IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.Doi Number

Object-Centered Petri Net Process Prediction: A Case Study of Multi-System Intelligent Healthcare

SHAO Chifeng[1,2], WANG Qianqian[2]

[1]School of Mathematics and Big Data, Anhui University of Science and Technology, Huainan 232001, China
[2]College of Information & Network Engineering, Anhui Science and Technology University, Fengyang 233100, China

Corresponding author: SHAO Chifeng (e-mail: 2022100081@aust.edu.cn).

This work was supported by the National Natural Science Foundation, China (No. 61572035, 61402011), the Key Project of Natural Science Research of Anhui Provincial Department of Education (2022AH051638), the Scientific Research Project of Anhui Science and Technology University (2021zryb31), the Talent Introduction Project of Anhui Science and Technology University (XWYJ202107).

ABSTRACT

To address the problem of business process prediction in smart healthcare involving the fusion of multimodal data from interactions among multiple systems, this paper proposes an object-centered explainable prediction method (OCPPP). The approach comprises three modules: (1) Utilizing control flow and data flow constraints between activities for process modelling, Algorithm 1 constructs a Petri net centered on the object to be predicted; (2) Recognizing that business activities share resources during system interactions, individual activities are analysed using AI models, with distinct models applied to different modalities of data in activity logs (e.g., multi-object detection for image data, time-series forecasting for text data); (3) Employing coloured Petri nets and Algorithm 2 for predicting activity durations, integrating outputs from various intelligent models to formulate predictions, including those for low-frequency events. The experimental outcomes indicate that during both individual component analysis and sequential forecasting phases, a versatile selection of AI models enables effective operation. Integrating Petri Nets for system-level predictions enhances explainability through six distinct service compositions. Furthermore, the introduction of precursor transitions and a waiting threshold facilitates the anticipation of infrequent behaviours, thereby augmenting the system's predictive capabilities for a broader spectrum of occurrences.

INDEX TERMS Object-Centered, Petri, Process Prediction, Intelligent Healthcare

I. INTRODUCTION

A. RESEARCH BACKGROUND

Business processes are regarded as the cornerstone of organizational operations, mapping the daily operational patterns and serving as a conduit for collaboration within departments and across organizational boundaries [1, 2]. At the core of this related domain, Business Process Management (BPM) has established a mature framework of knowledge, playing a pivotal role in enhancing organizational efficiency, with its significance further underscored in literature [3, 4].

In recent years, data-driven BPM techniques have flourished, encompassing process mining [5], model rectification [6], optimization [7], log generation [8], and process forecasting [9], attracting extensive attention both academically and in practical applications. These advancements have significantly contributed to the enhancement of organizational performance, particularly within Predictive Business Process Monitoring (PBPM), where predicting upcoming process characteristics has emerged at the forefront of research.

PBPM aims to anticipate the key features of future process instances, leading to the development of various solutions tailored for specific prediction tasks in recent times. These notably include forecasting the subsequent event sequences [9-11], remaining processing times [12, 13], and variables linked to outcomes [14-16]. Among these approaches, methods based on deep learning, especially Long Short-Term Memory (LSTM) networks

[17], have garnered significant attention. While these methods boost predictive performance, the opacity inherent in deep models raises concerns about model credibility; conversely, traditional prediction methods offer better interpretability but are less efficient.

Over the past few decades, rapid advancements in machine learning have equipped PBPM with a potent toolkit, with literature [18, 19] delving into this evolutionary trend. Machine learning technologies have permeated a wide array of fields, including natural language understanding [20, 21], anomaly detection [22, 23], image recognition [24], database knowledge discovery [25], and time series forecasting [26]. Defining objective functions to optimize algorithm performance is crucial, enabling quantitative assessments of their efficacy, as extensively discussed in [27, 28]. The rapid progression of artificial intelligence technology has given rise to diverse model architectures, furnishing effective tools for various tasks. Nonetheless, the "black box" nature of deep learning models has propelled explainability to the forefront of current research concerns.

Processes are seen as instantiated flows composed of individually executed events. However, inter-organizational processes are typically more intricate (as depicted in Fig.1), with several process instances executing concurrently and potentially interacting. This scenario resembles orchestration, where an instance of process P_1 interacts and synchronizes with multiple instances of a second process P_2, and vice versa. Furthermore, instances of P_2 may in turn interact with other instances. For example, in a hospital, multiple patients may undergo individual checkups. To economize, the hospital associates multiple patients' tests to a single testing window. Moreover, a patient's checkup may necessitate visits to different departments and windows. Interactions between patients and Windows are managed through instances of different processes: an instance of a patient checkup process can be linked with multiple instances of Windows checkup processes, and the reverse is also true. These interactions may involve resource conflicts or sharing, which can impact the occurrence of interacting activities.

In stark contrast to conventional global process prediction methodologies, Fig.1 highlights the article's focal point on a nuanced analysis of multiple attributes associated with individual activities (specifically, t6), along with the forecasting of individual time-series data (t6_i+n, where n=1,2,3,…). Furthermore, it delves into the intricate causal and shared relationships among distinct activities (t3, t8, tk), exploring both their sequential and interdependent nature. This comprehensive examination culminates in an integrated process prediction, providing a sophisticated and holistic understanding of the dynamic interplay within the system under study.

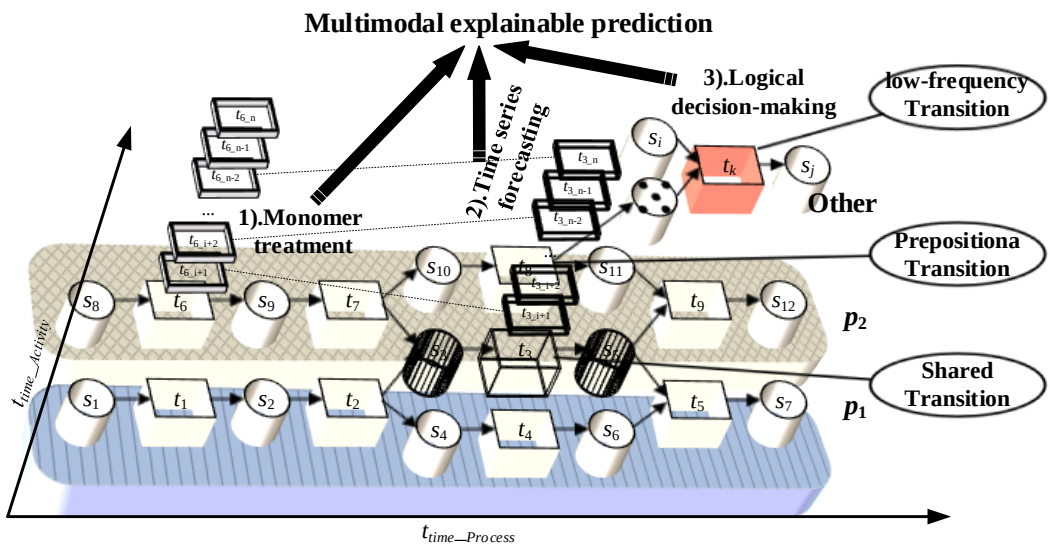


FIGURE 1. **Illustration of Resource Conflicts and Supply Issues in Multi-system Interactions.**

B. SOLUTION AND CONTRIBUTIONS

To tackle the inefficiencies of conventional process prediction and the explainability limitations in deep learning-based forecasting methods, we propose the Object-Centric Predictive Process Prediction (OCPPP) method. This approach integrates deep learning for individual activity prediction with traditional forecasting at the decision-making tier. The methodology's principles are elucidated with the assistance of Fig.1. The core idea revolves around dynamically assembling relevant activities for the current entity into a target business process for prediction. Depending on the actual data requirements, deep learning techniques such as YOLO[29] and LSTM[30] are selectively employed for data extraction and state forecasting from individual activity logs. The CPNtools[31] software is then used to simulate the predicted process, and low-frequency behaviours are forecasted based on triggers from preceding non-low-frequency activities.

The key contributions of this paper are summarised as follows:

1. Enhancing Activity Log Utilization: By applying AI models to predict the status of individual activity logs, we improve the efficiency of exploiting log attributes from an activity perspective.

2. Improved Explainability through Process View: Combining control flow modelling grounded in Petri Nets with the states of individual activities leads to explainable predictions, thereby increasing transparency compared to opaque deep learning models.

3. Addressing Low-Frequency Events: Given the potential inadequacy of event data for training models targeting low-frequency behaviours, our method predicts the preceding non-low-frequency activities to support low-frequency predictions.

4. Data Dimensionality and Online Prediction: Leveraging low-dimensional data for state prediction of individual activities minimizes the need to learn implicit relations, enabling real-time model updates. Integrating this with Petri Net control flow modelling facilitates online prediction of business processes centered on objects.

The subsequent sections are organized as follows: Section 2 introduces pertinent definitions; Section 3 outlines the steps for building process models using CPNs, discusses the handling of multimodal data, and explains how AI model outputs are leveraged with CPNs for interpretable forecasts. Section 4 presents the experimental setup, data, evaluation criteria for AI models, and the use of CPNTools for simulating medical processes. Lastly, Section 5 summarizes the paper and contemplates future directions for research.

II. PRELIMINARIES

This section describes the definitions of algorithm-related content. This paper uses deep learning algorithms to process activity data, predicts the control flow model and activity state based on Petri net, and characterizes the real activities in Petri net.

Definition 1[32] A simple (with k colours) colour Petri net is a quintuple $\sum = (S, T; F, W, M)$

Among them $(S, T; F)$ is a net,

$$W : F \to \{0,1,2,\cdots\}^k$$

$$M : S \to \{0,1,2,\cdots\}^k$$

Yes $t \in T$, if

$$s \in {}^{\bullet}t \to M(s) \geq W(s,t)$$

Then transition t has the trigger authority $(M[t>$ at the marker M; under the marker M, transition t is triggered, resulting in a new marker $M'(M[t>M')$.

$$M'(s) = \begin{cases} M(s) - W(s,t), if \quad s \in {}^{\bullet}t - t^{\bullet} \\ M(s) + W(t,s), if \quad s \in t^{\bullet} - {}^{\bullet}t \\ M(s) - W(s,t) + W(t,s), if \quad s \cap {}^{\bullet}t - t^{\bullet} \\ M(s), other \end{cases}$$

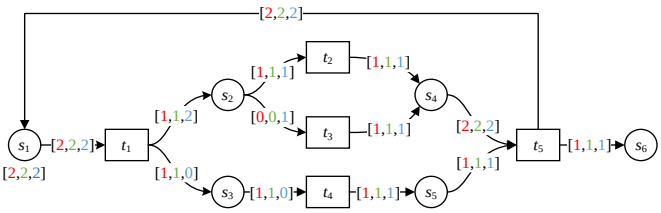


FIGURE 2. A simple colour Petri net.

Fig.2 illustrates a simple coloured Petri net, $\sum_1$, depicted graphically, where places and annotations on directed arcs are all three-dimensional vectors, indicating that $\sum_1$ is a colour Petri net incorporating three colours. Let vector $X_{i[1]}$ =Red, $X_{i[2]}$=Green, and $X_{i[3]}$=Blue. Under the initial marking M_0 of the net system, place s_1 contains 2 tokens of Red, 2 of Green, and 2 of Blue. Places $s_{[i]}$ (for i=2, 3, 4, 5, 6) hold no tokens (the absence of notation implies a vector [0,0,0]). As inferred from the weights on the input and output arcs of transition t_1, when there are no fewer than 2 tokens of Red, Green, and Blue in place s_1, transition t_1 can be activated. Execution of t_1 reduces each of the Red, Green, and Blue tokens in s_1 by 2, while it increments s_2 with 1 Red token, 1 Green token, and 2 Blue tokens, and s_3 with 1 Red token and 1 Green token.

Definition 2 For a set N of nets $\sum$ and given $\left\{\sum_i, \sum_j \in N \middle| T_i \cap T_j \neq \phi\right\}$, if nets $\sum_i$ and $\sum_j$ share a set of transitions $T_{shared} = T_i \cap T_j$, then these transitions are considered shared among the nets. For a shared transition $t_i \in T_{shared}$, when transition t_i participates in characterizing the Process (process representation) in both nets $\sum_i$ and $\sum_j$, the attribute sets $Attributes\_\sum_i(t_i)$ of t_i in $\sum_i$ and $Attributes\_\sum_j(t_i)$ of t_i in $\sum_j$ might not be identical. In this case, the transition t_i is referred to as a resource-sharing transition between $\sum_i$ and $\sum_j$.

Definition 3 For a Petri net $\sum$, let $t_i\bullet$ and $\bullet t_j$ denote the postset of transition t_i and the preset of transition t_j, respectively. If $t_i,\ t_j \in T \wedge t_i\bullet \cap \bullet t_j \neq \phi$, the transition t_i is considered a predecessor of transition t_j. Let $S_{Associate}(t_i, t_j) = t_i\bullet \cap \bullet t_j$ be the set of places and $T_{Front} = \bullet S_{Associate}(t_i, t_j)$ the set of transitions. When transition T_{Front} is fired, if it enables the conditions for transition t_j to fire, the set T_{Front} is referred to as the precondition transition set of transition t_j.

The following definition is provided to facilitate the depiction of complex business processes: A process diagram comprises multiple intersecting business streams, where, for reasons of system security or privacy protection, these streams do not communicate with one another during execution and maintain logs separately per stream. As illustrated in Fig.1, the network structure features transitions (t_8) acting as antecedents to transitions (t_k), where all places within the marked subset must be allocated the required number of tokens for the transition ($\bullet\ t_k$) to fire. In this schema, the shaded portion of p_1, combined with places s_3, s_5, and the resource-sharing transition t_3, forms a subnet $\sum_1$ starting from place s_1 and terminating at place s_7. Similarly, the shaded part of p_2, when joined with places s_3, s_5, and the same resource-sharing

transition t_3, constitutes a subnet $\sum_2$ commencing at place s_8 and concluding at place s_{12}.

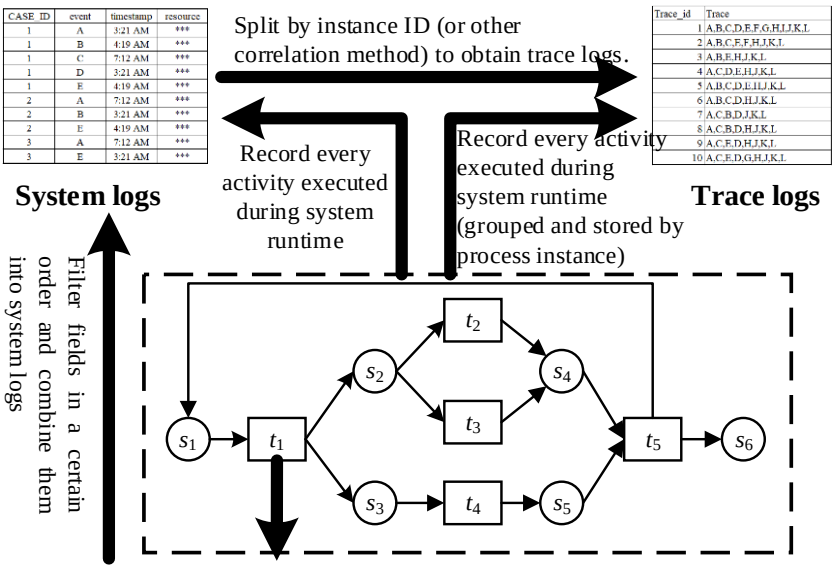
Definition 4 For a net $\sum$, during its operation, whenever a transition $t_i \in T$ fires, information about the firing of t_i is recorded as an entry $Inf(t_i)$ formatted as <ID, PID, UserID, timestamp, Other_Info>. The collection of these entries, denoted as $L\_Inf(t_i)$, constitutes the activity log for transition t_i.

Definition 5 For net $\sum$, during its operation, every time a transition $t_i \in T$ triggers, the firing information of t_i), denoted as $Inf(t_i)$, is documented following the schema <CASE_ID, event, timestamp, Other_Info>. Given $t_i \in T, i = 1,2,3,...,T_{.size}$, the ensemble of these $Inf(t_i)$ forms the system log of the net $\sum$, referred to as $L\_Inf(\sum)$.

Definition 6 For net $\sum$, in a particular network execution instance, the firing sequence of transition $t_i \in T, i = 1,2,3,...,T_{.size}$ is documented as $Inf(Trace)$ = < t_{start} ,…, t_{i-1} , t_i , t_{i+1} ,…, t_{end} >, where each t_i denotes a transition fired in sequence. The collection of these sequences, referred to as $L\_Inf(Trace)$, composes the trace log of the net $\sum$.

Definition 7 For net $\sum$, during network execution, the firing information of a transition $t_i \in T$ is documented as <ID, PID, UserID, timestamp, Other_Inf>, where Other_Inf encompasses one or more forms of data beyond text, such as audio, images, or video. Such a transition t_i is referred to as a multimodal transition.

Prevalent forms of log data include activity logs and system logs, as illustrated in Fig.3, with their attributes and dimensions potentially varying due to differences in database tables. Within the realm of process mining, the commonly utilized type of log is the trace log, typically stored in file formats such as CSV or XES. Activity logs document the execution information of individual system activities, with each activity recorded in corresponding data tables. Conversely, system logs encapsulate summary information of multiple activity executions throughout the entire system's runtime, featuring relatively standardized attributes and dimensions.



CASE_ID	event	timestamp	resource
1	A	3:21 AM	***
1	B	4:19 AM	***
1	C	7:12 AM	***
1	D	3:21 AM	***
1	E	4:19 AM	***
2	A	7:12 AM	***
2	B	3:21 AM	***
2	E	4:19 AM	***
3	A	7:12 AM	***
3	E	3:21 AM	***

Trace_id	Trace
1	A,B,C,D,E,F,G,H,I,J,K,L
2	A,B,C,E,F,H,J,K,L
3	A,B,E,H,J,K,L
4	A,C,D,E,H,J,K,L
5	A,B,C,D,E,I,I,J,K,L
6	A,B,C,D,H,J,K,L
7	A,C,B,D,J,K,L
8	A,C,B,D,H,J,K,L
9	A,C,E,D,H,J,K,L
10	A,C,E,D,G,H,J,K,L

ID	PID	USER_ID	text_Area	image_path	video_id	voice	timestamp	other
1	20230401_1981	scf	This is a good day,	E:\YourPhoto\12.jpg	E:\YourVideo\34.rmvb	E:\YourVoice\	2023/4/1 7:31	short_mssage
2	20230401_1981	scf	I love you!	E:\YourPhoto\13.jpg	E:\YourVideo\35.rmvb	E:\YourVoice\	2023/4/1 8:24	short_mssage
3	20230401_1981	scf	You are a good girls,	E:\YourPhoto\14.jpg	E:\YourVideo\36.rmvb	E:\YourVoice\	2023/4/1 9:13	short_mssage
4	20230401_1981	wy	***	E:\YourPhoto\15.jpg	E:\YourVideo\37.rmvb	E:\YourVoice\	2023/4/1 10:12	we_chat
5	20230401_1981	wy	***	E:\YourPhoto\16.jpg	E:\YourVideo\38.rmvb	E:\YourVoice\	2023/4/1 10:31	we_chat
6	20230401_1981	wy	***	E:\YourPhoto\17.jpg	E:\YourVideo\39.rmvb	E:\YourVoice\	2023/4/1 10:31	we_chat
7	20230401_2036	wy	***	E:\YourPhoto\18.jpg	E:\YourVideo\40.rmvb	E:\YourVoice\	2023/4/1 11:12	we_chat
8	20230401_2036	wy	***	E:\YourPhoto\19.jpg	E:\YourVideo\41.rmvb	E:\YourVoice\	2023/4/1 11:16	we_chat
9	20230401_2036	wy	***	E:\YourPhoto\20.jpg	E:\YourVideo\42.rmvb	E:\YourVoice\	2023/4/1 11:19	we_chat
10	20230401_2036	wy	***	E:\YourPhoto\21.jpg	E:\YourVideo\43.rmvb	E:\YourVoice\	2023/4/1 11:30	we_chat

FIGURE 3. The correlation of activity logs, system logs, trace logs, and model predictions.

The transformation from activity logs to system logs can be conceptualized as the summarization of activity logs, wherein system logs retain associated attributes reflecting system behaviours. Meanwhile, converting system logs into trace logs necessitates a transformation tailored to the structure of the system logs. This conversion poses a significant challenge when dealing with system logs that lack clear identification of data flow ownership.

III. Multimodal interpretable prediction methods

This section primarily introduces how a multimodal explainable prediction approach is employed to address challenges in a case study of medical business process prediction. Specifically, focusing on the object-centric principle, it outlines the customization of a process model for the medical examination project workflow based on the CPN (Colored Petri Net) model, tailored to the object's

needs. This approach brings to light issues such as waiting times due to shared transition resource constraints and variations in execution delays for identical transitions under different contexts. To tackle these problems, this section proposes a multimodal analysis method. It conducts data mining on the activity logs of the business process, encompassing both image information extraction and time-series data prediction. Concurrently, leveraging the CPN model, it performs a composite prediction of disordered behaviours, aiming to achieve the objectives of forecasting the duration of the business process and analyzing the low-frequency behaviours following multiple preceding transitions. This comprehensive strategy enhances the explainability and accuracy of predictions in the complex medical workflow environment.

A. CPN MODELLING

Existing business process prediction methodologies predominantly cater to static workflows, whereas adjustments to these methods, particularly the conventional probabilistic forecasting and deep learning-based approaches, become notably cumbersome when processes undergo modifications. In the context of healthcare processes, variability significantly differs from the perspectives of hospitals, physicians, and patients. Consequently, practical implementation of process prediction methodologies necessitates agile adaptation and updating by real-world scenarios, concurrently taking into account the implications of workflow dynamics. Moreover, distinctive features of individual processes may warrant the employment of different predictive techniques and models.

This section harnesses CPNTools for model simulation, adopting an object-centric approach to construct the workflow model based on the specified needs of these objects. Beyond the diagrammatic representation of the process model, fundamental elements are also configured, such as in a health checkup scenario where the number of examination items is fixed and individuals navigate through them. The foundational declarations include:

The personnel set `colset PEOPLE = with XianwenFang | XianjinFang | HuanFang | ChifengShao | LiliWang | DuoqinLi | Others timed;` wherein `Others` signifies the current queue of attendees, while the rest symbolize the subjects of prediction, and `timed` denotes the temporal dimension support for this ensemble.

A queue list for individuals is established as `colset PEOPLEList = list PEOPLE;`, functioning as a token cache representing the queue. Corresponding instances, `var people: PEOPLE; var peopleList: PEOPLEList;`, facilitate object instantiation within flow arcs. To emulate the duration of process execution, a time set `colset E = unit with e timed;` is employed.

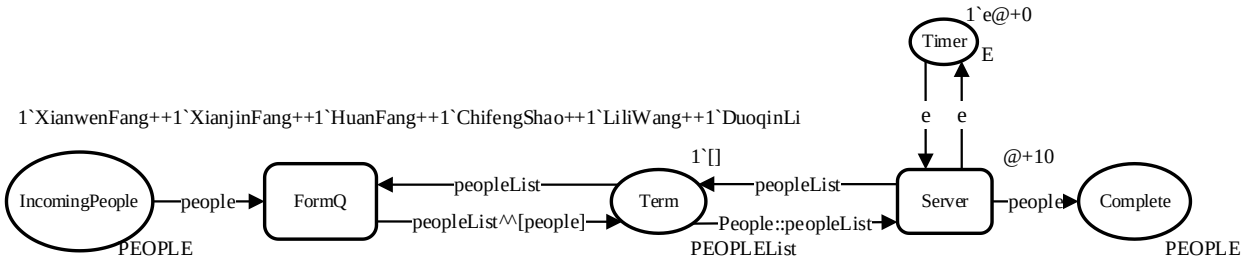


FIGURE 4. CPN model with queue and resource access restrictions.

Fig.4 illustrates a queue model where the current state holds five tokens in `IncomingPeople`, with only transition `FormQ` capable of triggering progression to the next state. Place `Term` serves as a queue repository, with both `FormQ` and `Server` transitions being triggerable at this point; if `Server` is triggered, it consumes one Token from `Timer` (recovering after `t` time units) to advance to the subsequent state. While `Server` is delayed, additional triggering of `FormQ` increments the count of waiting tokens in `Term`.

Regarding arc syntax, `[XianwenFang, XianjinFang]^^[HuanFang]` yields `[XianwenFang, XianjinFang, HuanFang]`, illustrating list concatenation. For the arc from transition `FormQ` to place `Term` annotated `peoplelist^^[people]`, it signifies that upon `FormQ`'s activation, the current occupants of `Term` are concatenated with the outgoing tokens from `IncomingPeople` into `Term`.

The operation `ChifengShao::[DuoqinLi]` results in `[ChifengShao, DuoqinLi]`, prepending `ChifengShao` to the list. Annotation `people::peoplelist` on the arc from place `Term` to transition `Server` implies that `Term` merges `people` into `peoplelist`, subsequently feeding the first element of `peoplelist` into `Server`.

Place `Timer` functions as a governor for parallel execution instances of `Server`, allowing further triggering only if `Timer` holds an available token. It restricts both the number of concurrent executions of the transition and the execution duration, with the limitation of adapting to the input of Tokens. Reflective of real-world scenarios, such as manual inspections where worker efficiency varies with prolonged working durations, and queue lengths exhibit patterned changes over time, these two factors impact the total duration of executing medical examination sequences differently. The following section elucidates the application of deep learning methodologies for data mining in activity logs to address this complexity.

B. Multimodal analysis

In certain business processes, multimodal data types like text, images, and videos may be incorporated, yet conventional process mining techniques often fail to leverage this rich multimodal data effectively. Furthermore, most artificial intelligence-driven process prediction methodologies are primarily geared toward forecasting sequential activities alone. The integration of AI models, therefore, stands as a prominent research frontier within process mining. Grounded in practical necessities and real-world constraints, this work adopts a fusion of two models – yolo_v7 and LSTM – to address these limitations.

The ensuing subsections delve into the application of these models: the utilization of yolo_v7 for object detection within image data extracted from activity logs, and the deployment of LSTM for time-series prediction based on sequential data present in the same activity logs. This dual-model approach aims to enhance the comprehensiveness and accuracy of process predictions by tapping into the diverse modalities inherent in modern business process data.

1) EXTRACTION OF QUEUE DATA

With the evolution of artificial intelligence (AI) technology, it has become feasible to employ AI models to scrutinize image data within log records, thereby yielding more accurate insights into both activity states and overall process conditions. This heightened precision is particularly evident when delving deeper into multimodal transitions, where the presence of image data within activity logs $L\_Inf(t_i)$ allows for a more meticulous depiction of the current status.

YOLO, short for You Only Look Once, represents a one-stage object detection model that excels in real-time systems due to its suitability for rapid detection tasks. As depicted in Fig.5, an image undergoes convolutional operations to extract salient features, concurrently forecasting bounding boxes and the probabilities of object categories encapsulated within these boxes. This approach marries feature extraction with prediction in a streamlined manner, enhancing the efficiency and responsiveness of image-based analyses in dynamic environments.

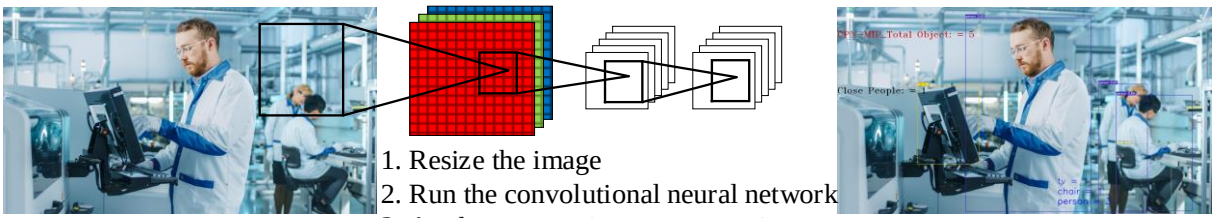


FIGURE 5. Schematic Diagram of YOLO Object Detection.

The YOLO framework operates as follows: Initially, through a sequence of convolutional operations, the input image is divided into a grid, with each cell in the grid responsible for detecting objects whose arithmetic centre falls within its boundaries.

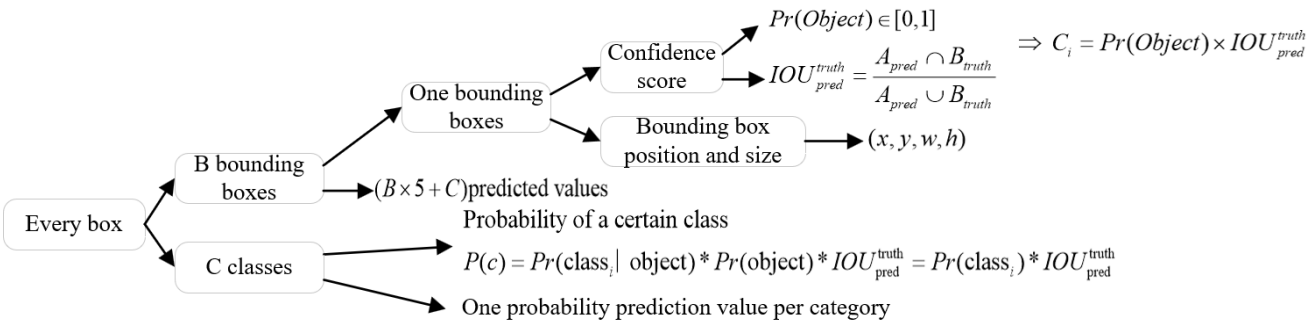


FIGURE 6. Principles Underlying YOLO Object Detection.

For each grid cell, YOLO predicts B bounding boxes along with their respective confidence scores, and additionally, C class probabilities for classification purposes. Each bounding box is represented by (x, y, w, h), signifying the centre coordinates and the width and height of the box, respectively (where x and y denote offsets from the top-left corner of the grid cell, measured in units relative to the cell size). The width and height predictions, w and h, are ratios relative to the width and height of the entire image.

Confidence scores encapsulate two dimensions: one reflects the probability of an object existing within the box, denoted as $Pr(Object)$, which equals 1 if the box contains an object and 0 otherwise; the other gauges the accuracy of the box, quantified by the Intersection over Union (IOU) between the predicted and ground truth boxes, calculated as $IOU^{truth}_{pred} = (A_{pred} \cap B_{truth}) / (A_{pred} \cup B_{truth})$. Hence, the confidence score for a bounding box can be defined as $Pr(Object) \times IOU^{truth}_{pred}$. In total, each box requires

predictions for 5 elements (x, y, w, h, c), with the initial four defining the box's position and size, and the last denoting its confidence.

Each grid cell also estimates probabilities for C classes, signifying the likelihood of the object(s) within the cell's bounding box(es) belonging to each class. Classification is only attempted when $Pr(Object)=1$, so these probabilities represent conditional probabilities for each class given there is an object, i.e., $Pr(\text{class}_i|\ \text{object})$.

Consequently, the class confidence for each bounding box can be computed as:

$$P(c)=Pr(\text{class}_i|\ \text{object})*Pr(\text{object})*IOU_{\text{pred}}^{\text{truth}}$$
$$=Pr(\text{class}_i)*IOU_{\text{pred}}^{\text{truth}}$$

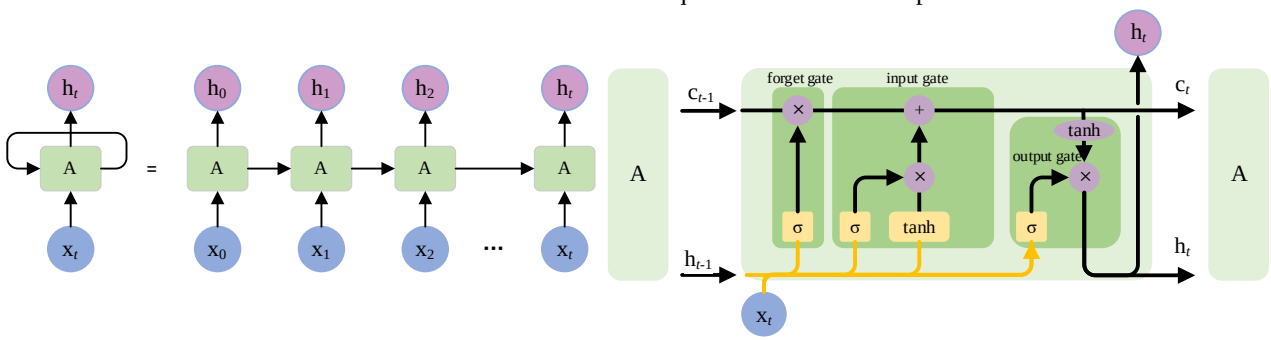
In summary, each grid cell is tasked with predicting a total of (B*5+C) values, where B represents the number of bounding boxes per grid cell and C denotes the total number of classes. Given 20 classes, S=7, and B=2, the ultimate prediction output would constitute a tensor of dimensions 7×7×(2×5+20).

By applying Yolo_v7 to the context of manufacturing processes, real-time monitoring of processing queues becomes feasible, transforming video image data into numerical information for the purposes of prediction, optimization, and management.

2) QUEUE DATA PREDICTION

With the advancement of artificial intelligence technologies, AI models can now analyze sequential data within logs to predict future activity and process states. This is particularly pertinent in the in-depth analysis of resource-sharing transitions t_i, where the activity logs $L\_Inf(t_i)$ containing temporal information facilitate predictions about the impending states of individual transitions t_i shared across multiple systems.

Long Short-Term Memory (LSTM), an enhancement upon Recurrent Neural Networks (RNNs) [29], is a model designed to better apprehend sequential patterns. Illustrated in Fig.7(b), LSTM's architecture incorporates mechanisms such as state carryover and forget gates, enhancing its capacity to capture temporal dynamics and improve sequence modelling. In business process forecasting, LSTM proves instrumental in predicting sequential data related to queue statuses, and resource utilization, among others, where Section 1) EXTRACTION OF QUEUE DATA facilitates the transformation of image data into sequential data for such predictive tasks.



(a)RNN (b)LSTM

FIGURE 7. Network Architecture Diagram for Queue Prediction Model.

At every time step t, a Recurrent Neural Network (RNN) considers both its current state from the previous time step and the input at the current time step t. It aggregates this information to produce an output at time t and, concurrently, updates its state for the next time step. Initially, the state is initialized as a vector of zeros.

$$state\_t=0$$
$$for\ input\_t\ in\ input\_sequence:$$
$$output\_t=f(input\_sequence, state\_t)$$
$$state\_t=output\_t$$

The function f is responsible for transforming the input and the state into the output, typically involving two matrices, W and U, and a bias vector b, followed by an activation function. It can be formally expressed as follows:

$$f=activation\left(dot(\mathbf{W}, input)+dot(\mathbf{U}, state)+\mathbf{b}\right)$$

In comparison to traditional RNNs, LSTM introduces a mechanism that enables the propagation of information across multiple time steps. At any given time step, the output is influenced by three components: the current input, the current state, and the carried-over information from prior time steps. This can be depicted as follows:

$$output\_t=activation\begin{pmatrix}dot(state\_t, \mathbf{U}_o)\\+dot(input\_t, \mathbf{W}_o)\\+dot(C\_t, V_o)+\mathbf{b}_o\end{pmatrix}$$

$$i\_t=activation\left(dot(state\_t, \mathbf{U}_i)+dot(input\_t, \mathbf{W}_i)+\mathbf{b}_i\right)$$

$$f\_t=activation\left(dot(state\_t, \mathbf{U}_f)+dot(input\_t, \mathbf{W}_f)+\mathbf{b}_f\right)$$

$$k\_t=activation\left(dot(state\_t, \mathbf{U}_k)+dot(input\_t, \mathbf{W}_k)+\mathbf{b}_k\right)$$

These three values are then combined to update the state.

$$C\_t:\ \ c\_t+1=i\_t*k\_t+c\_t*f\_t$$

This paper employs the Min-Max normalization technique for preprocessing the input data fed into the LSTM model. Continuous attribute values are normalized directly, whereas discrete attribute values are first encoded

numerically to represent different categories before undergoing normalization.

C. EXPLAINABLE PREDICTIONS

Process prediction, often tailored to static workflows, is conventionally categorized into two primary approaches: 1) Traditional forecasting methods that estimate the probabilities of event occurrences and subsequent correlations, offering moderate accuracy levels but with higher interpretability. 2) Deep learning-based process predictions, which excel in forecasting remaining system durations and upcoming activities, yield better predictive performance. However, the black-box nature of deep learning undermines the interpretability and reliability of prediction algorithms. They also demonstrate limited adaptability to common modifications such as resource constraints or other conditional changes affecting individual activities. To enhance both the interpretability and effectiveness of prediction methodologies, this work introduces an object-centric Petri net modelling approach and a multimodal explainable prediction framework. The subsequent sections detail the construction of business processes and the development of state prediction methodologies.

1) CONSTRUCTING BUSINESS PROCESSES

During business execution, phenomena such as concept drift frequently arise, where the actual execution deviates from the predefined process, necessitating subsequent adjustments to the process model to align with real-world operations. This underscores the need for considering the adaptability of prediction models in business process forecasting. Real-world scenarios are replete with intertwined business processes, each encompassing a multitude of intersecting execution instances. Building upon this understanding, we proceed to outline an intelligent methodology for constructing predictive process models based on activity correlations, tailored to the known objectives of the entities involved. This approach aims to systematically anticipate and accommodate the intricate dynamics of interwoven processes underpinning practical business demands.

In Algorithm 1, steps (2-9) traverse the given set of business requirements to identify if there exists a network where transitions t_i and t_j represent data-constrained transitions, marking them accordingly. Steps (10-15) iterate through the same requirement set to locate data-constrained transitions within networks that include business-related transitions. Steps (16-34) meticulously navigate the requirement collection: if the current transition lacks predecessors, it is appended directly to the network's tail. If there is a single predecessor transition present in the network, it is appended following its predecessor; otherwise, the addition is postponed until a later traversal. If the current transition has multiple predecessors and all are present in the network, it is attached after the last one; if not all predecessors are found, the process is deferred. Ultimately, a sequence of networks with predecessor-marked transitions is obtained, which, when sequentially connected, forms the predictive process network.

Algorithm 1: Creation of Predictive Process Model

Input： Workflow to be constructed, denoted as ∑, Set of business requirements, T, Existing network set, N,
Output： Predictive Process Model, represented as ∑.

```
01  List[Net] N, List[Transition] T, Process ∑;
02  for i in T:
03     for j in T:
04        for k in N
05           if in N[k] ti.input_flow.data∩
tj.output_flow.data != Null:
06              label(ti.pre=tj);
07        end for
08     end for
09  end for
10  for i in T:
11     for j in N[i]:
12        if  ti.input_flow.data∩
N[i][j].output_flow.data != Null:
13           label(ti.pre=N[i][j]);
14     end for
15  end for
16  for i in T:
17     if ti.pre== Null:
18        ∑.append(ti);
19     else if ti.pre in net:
20        if ti.pre==1:
21           j=Net.indexof(ti.pre);
22           ∑[j].append(ti);
23        else if ti.pre>1:
24           if ti.pre all in net:
25              var z=0;
26              for k in ti.pre:
27                 if Net.indexof(ti.pre)>z:
28                 z=Net.indexof(ti.pre)
29              end for
30              ∑[z].append(ti);
31           else put into end to execute;
32     else if ti.pre not in Net:
33        put into end to execute;
34  end for
35  return ∑;
```

Fig.8 presents a concrete example. Algorithm 1 leverages the set of required activities and their corresponding process sets to uncover relationships among the required activities and data-constrained transitions governing activity execution across distinct workflows. By exploiting these relationships, an expanded set of interconnected activities meeting the business requirements is synthesised to yield the desired predictive process.

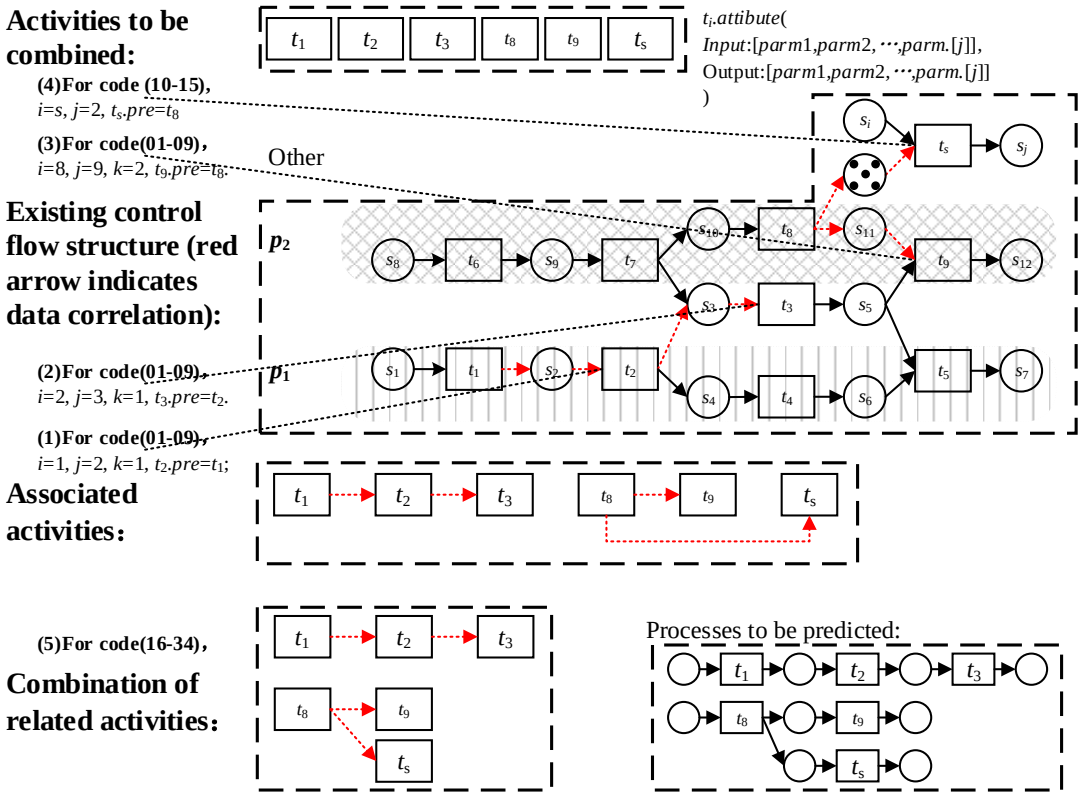


FIGURE 8. **Creating a Predictive Process Model for Upcoming Tasks Based on Specified Business Requirements and Existing Network Sets.**

2) STATE PREDICTION APPROACH

When predicting the state of business processes, time is commonly used as the metric, driving the evolution of the system through temporal changes. Within process nets, sequential, selection (branching), and concurrent structures are prevalent. For sequential blocks, the method involves updating the time upon completion of each activity and then forecasting the state of the subsequent activity. In selection blocks, the time is updated post-execution of a chosen path, followed by a renewed prediction for the next activity's state. Concurrent structures necessitate separate process predictions for each path, with the time being updated after the longest path concludes, leading to a fresh prediction for the upcoming activity.

In the absence of sequential chaining behaviour, the duration of various combinations can be influenced by the evolving activity states over time. Algorithm 2 outlines the state prediction approach tailored to structures lacking sequential chaining behaviour, acknowledging the impact of varying time-dependent activity combinations on structural persistence.

In Algorithm 2, steps (01-08) traverse all possible sequences of activity occurrences, calculating the total duration for each sequence. Step (02) initializes the reference time for the prediction algorithm. Steps (03-06) iterate through the current sequence of events, computing the cumulative duration for each activity; within this, step (04) updates the reference time for the prediction algorithm, while step (05) refreshes the total duration for the ongoing sequence. The variable TIME_total retains the aggregate durations corresponding to all possible sequences of activity execution.

Algorithm 2 adopts a control flow perspective, systematically combining sequences of business activities and integrating deep learning prediction techniques to forecast the remaining time of business processes. By examining different sequences of activity execution from a control flow standpoint and leveraging advanced machine learning, it offers a comprehensive approach to estimating process timelines.

Algorithm 2: Remaining Time Prediction for Combined Processes

Input: Permutations and combinations of the set of transitions to be predicted, A_T, Initial timestamp, time_init, Transition state prediction function predict(), Accumulated duration set, TIME_total.
Output: TIME_total;

```
01   for i in A_T:
02       time=time_init;
03       for j in A_T[i]:
04           time += predict(A_T[i][j]);
05           TIME_total[i]= TIME_total[i]+
predict(A_T[i][j]);
06       end for
07   end for
08   return TIME_total.
```

For infrequent transitions t_j with precondition transition sets T_{Front}, an analysis of the associated data is conducted: Infrequent transitions have sparse and less favourable log data for prediction. From a control flow perspective, however, predicting the triggering of transitions t_j within the precondition set T_{Front} can offer substantial reference for anticipating the occurrence of infrequent transition.

Algorithm 3 initiates from a control flow viewpoint, integrating the state prediction of precondition transition sets T_{Front} to forecast the emergence of infrequent transitions t_j. This approach capitalizes on understanding the dynamics of preceding transitions to inform predictions about those that occur less frequently, thereby enhancing the overall predictive power of the model.

Algorithm 3: Method for Antecedent Behavior Prediction of Low-Frequency Events

Input: Infrequent transition(s), denoted as t_j, Set of preceding transitions, represented as T_{Front}, Set of timepoints for triggering low-frequency events, Time_L, Set of non-infrequent behaviour thresholds, Threshold, Transition state prediction function, *predict*(), Set of timeslots where predictions for non-infrequent behaviours meet the threshold, Time_H_List, Prediction time range, Time_predict, Notably, Time_H_List.size= T_{Front}.size.
Output: Time_L

```
01  for i in T_Front :
02      Time_H_List=0;
03      for j in Time_predict:
04          if predict(j) > Threshold[i],
Time_H_List[i].add(j);
05      end for
06  end for
07  Time_L= Time_H_List[0];
08  for i in Time_H_List:
09      if Time_H_List[i]=null, return Time_L=null;
10      else Time_L=Time_L ⋂ Time_H_List[i];
11  end for
12  return Time_L.
```

In Algorithm 3, steps (01-06) traverse the set of preceding transitions for infrequent transitions, calculating time intervals where the prediction of non-infrequent behaviours satisfies a predetermined threshold (Threshold). Step (02) initializes the time when the set of preceding transitions meets the threshold. Step (03) forecasts the upcoming occurrences of these preceding transitions for a defined period. Step (04) adds times meeting the threshold to the set of predicted satisfaction times for non-infrequent behaviours. Steps (07-11) then iterate through the set of times where predictions for non-infrequent behaviours satisfy the Threshold, identifying the intersection of all these time intervals. If non-empty, this intersection represents the potential timing for the infrequent behaviour to occur. Step (07) initializes the time for triggering the infrequent behaviour. Step (08) traverses the set of times where non-infrequent behaviours meet the threshold. If any prediction set within (09) is empty, the infrequent behaviour will not be triggered. Conversely, if all prediction sets in (10) contain values (are non-empty), their intersection is taken, indicating that all preceding transitions have met the threshold and thus triggering the infrequent behaviour. The set Time_L retains the potential timestamps for the infrequent transition's initiation.

IV. EVALUATION

This study conducts a series of experiments on real-world system logs to validate the feasibility of the proposed object-centric Petri net modelling and multimodal explainable prediction methodologies. This section commences with an introduction to the activity logs utilized, experimental setup, and evaluation metrics employed for assessing the proposed methods. Experiments proceed along three dimensions: queue length detection, queue size prediction, explainability, and rare event forecasting. The influence of varying time periods and different activity execution sequences on prediction tasks is analyzed, with CPNtools employed to simulate prediction tasks, thereby validating the explainability of the proposed approach.

The methods outlined herein were implemented using the PyTorch library in Python 3.8. All experiments were conducted on a Windows 11 system equipped with an Intel® Core™ i7-12700 CPU, GeForce RTX 3090 GPU, and 64 GB of RAM. The experimental dataset comprised three components: (1) the COCO2017 dataset for validating the YOLO model's performance; (2) selected surveillance images from the First Affiliated Hospital of the University of Science and Technology of China's West Campus, used for training and validating the YOLO model's performance in identifying queue sizes; and (3) registration data from three departments spanning 16 weeks, Monday through Friday, from 8:00 to 16:00, with data sampled every 5 minutes, to train and validate the time-series prediction models for forecasting registration trends in each department. Considering practical needs and real-world constraints, this work further trains the YOLO_v7 pre-trained model on hospital data for queue length detection and employs LSTM for time-series data handling. With advancements in deep learning and consideration of real-life scenarios, State-of-the-Art (SOTA) models can be updated accordingly. The configuration of experimental parameters is detailed in Table I.

D. QUEUE LENGTH DETECTION

The performance of trained deep learning models was assessed using multi-class metrics, with specific evaluation criteria including:

$$Precision = \frac{1}{m}\sum_{i=1}^{m}\frac{tp_i}{tp_i + fp_i}$$

$$Recall = \frac{1}{m}\sum_{i=1}^{m}\frac{tp_i}{tp_i + fn_i}$$

$$F_1 - score = \frac{1}{m}\sum_{i=1}^{m} 2 * \frac{P_i * R_i}{P_i + R_i}$$

TABLE I
CONFIGURATION OF MODEL PARAMETERS

YOLO:
('--epochs', type=int, default=300) # Number of training epochs ('--lr0', type=float, default=(1, 1e-5, 1e-1), # Initial learning rate (SGD=1E-2, Adam=1E-3) ('--lrf', type=float, default=(1, 0.01, 1.0), # Final OneCycleLR learning rate (lr0 * lrf)
LSTM:
('--epochs', default=100, type=int) # Number of training iterations ('--layers', default=2, type=int) # Number of LSTM layers ('--input_size', default=2, type=int) # Dimensionality of input features ('--hidden_size', default=32, type=int) # Dimensionality of the hidden layer ('--lr', default=0.001, type=float) # Learning rate ('--sequence_length', default=6, type=int) # Sequence length, typically uses the past half-hour's data to predict the queue size for the next half-hour

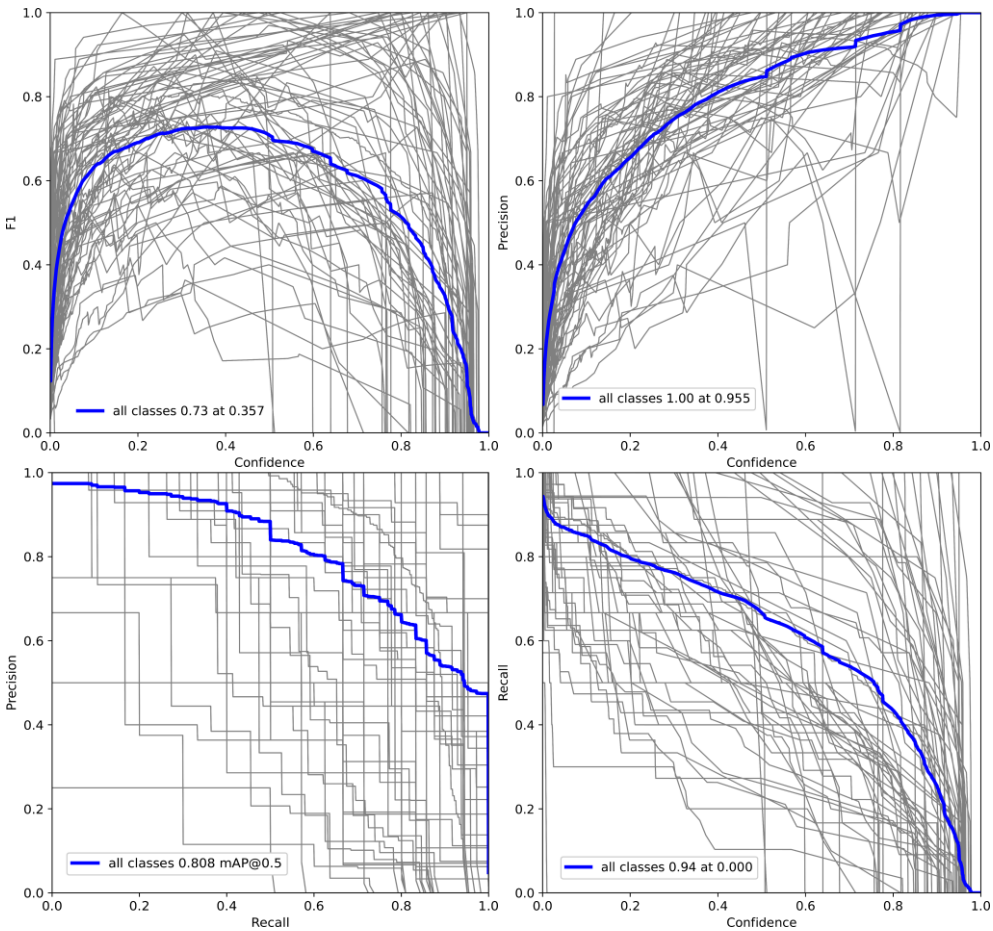


FIGURE 9. Evaluation Metrics Curves for Queue Length Detection Model Based on YOLO_v7: PR, F1, RC, PC.

In Fig.9. The experimental results demonstrate the robust performance of the YOLO model in object detection tasks. Specifically, the model achieved an impressive mean Average Precision (mAP) of 0.87 when evaluated with an

Intersection over Union (IoU) threshold of 0.5 across all categories, indicating a high level of detection accuracy. Furthermore, the F1-score versus confidence relationship revealed an optimal balance between precision and recall, with the F1-score peaking at 0.73 at a confidence threshold of 0.357, demonstrating the model's capability to detect target objects accurately while minimizing false positives and negatives. The recall-confidence curve showed a gradual decrease in recall as the confidence threshold increased, with maximum recall of 0.94 at zero confidence, highlighting the model's tendency to retain more predictions at lower thresholds, albeit with an increased risk of false positives. Conversely, the precision-confidence curve indicated a steady rise in precision with increasing confidence, reaching a perfect score of 1 at a threshold of 0.955, where all predicted targets were correctly identified without any false positives. However, such a high confidence threshold in practice may lead to a significant drop in recall, potentially missing many actual targets. In summary, the YOLO model displays exceptional detection capabilities, with fine-tuning of confidence thresholds offering opportunities to balance precision and recall for specific applications.

The object detection model, trained on both the COCO2017 dataset and the hospital image dataset, effectively identifies and categorizes humans, thereby counting the number of individuals and determining queue lengths. The recognition efficacy is visually demonstrated in Fig.10.

FIGURE 10. **Hospital Queue Length Acquisition.**

A. Queue Size Prediction

Experiments were conducted utilizing 16 weeks of registration data from three departments, encompassing two dimensions: the number of patients waiting for registration and the current time (Monday through Friday, 8:00 to 16:00, with data sampled every 5 minutes). The dataset was partitioned into training and testing subsets according to the following ratio:

trainx, trainy = X[:int(0.8 * total_len)], Y[:int(0.8 * total_len)]

testx, testy = X[int(0.8 * total_len):], Y[int(0.8 * total_len):]

This division ensures that approximately 80% of the data is allocated for training, with the remainder used for testing the model's predictive capabilities.

As presented in Table Ⅱ, the evaluation metrics of LSTM models trained on data from three distinct departments reveal that Model 3 exhibits exceptional performance across multiple crucial indicators. Specifically, Model 3 boasts a minimal Mean Squared Error (MSE) of 0.0023641 and a low Root Mean Squared Error (RMSE) of 0.0486221, indicating negligible deviations between its predictions and actual

values, thereby demonstrating a high level of predictive accuracy. Furthermore, the substantial reduction in the Mean Absolute Error (MAE) of Model 3, which stands at 0.0360433, underscores the high stability and reliability of its predictive outcomes. Notably, the R^2 score of Model 3 reaches an impressive 0.9536967, nearly approaching the ideal value of 1, which robustly validates its formidable capacity in explaining data variability and accurately capturing as well as predicting key patterns and trends within the data.

In contrast, while Models 1 and 2 do not match the performance of Model 3, they nonetheless demonstrate certain predictive capabilities. Model 1 outperforms Model 2 in terms of MSE, RMSE, and MAE, indicating relatively superior predictive precision and stability. Conversely, Model 2 exhibits weaker performance across all evaluation metrics, particularly in the R^2 score, which is significantly lower than the other two models. This observation underscores the importance of potentially focusing on enhancing the predictive power and data fitting capabilities of Model 2 in future model optimization efforts.

Three queue size prediction models were separately trained using three datasets, each corresponding to a different queue. The results of these predictions, juxtaposed against the actual data, are depicted in Fig.11. These predictive models effectively mirror the fluctuations in queue sizes, thereby reflecting the waiting situations for diagnostic activities in each department (in this context, all departments are assumed to use machinery for inspections, with a default completion time of 3 minutes per inspection).

TABLE II
EVALUATION METRICS FOR LSTM MODELS

Department	Mean Squared Error	Root Mean Squared Error	Mean Absolute Error	R-squared Score
1	0.0086111	0.0927961	0.0743720	0.8279177
2	0.0097239	0.0986100	0.0789896	0.7736935
3	0.0023641	0.0486221	0.0360433	0.9536967

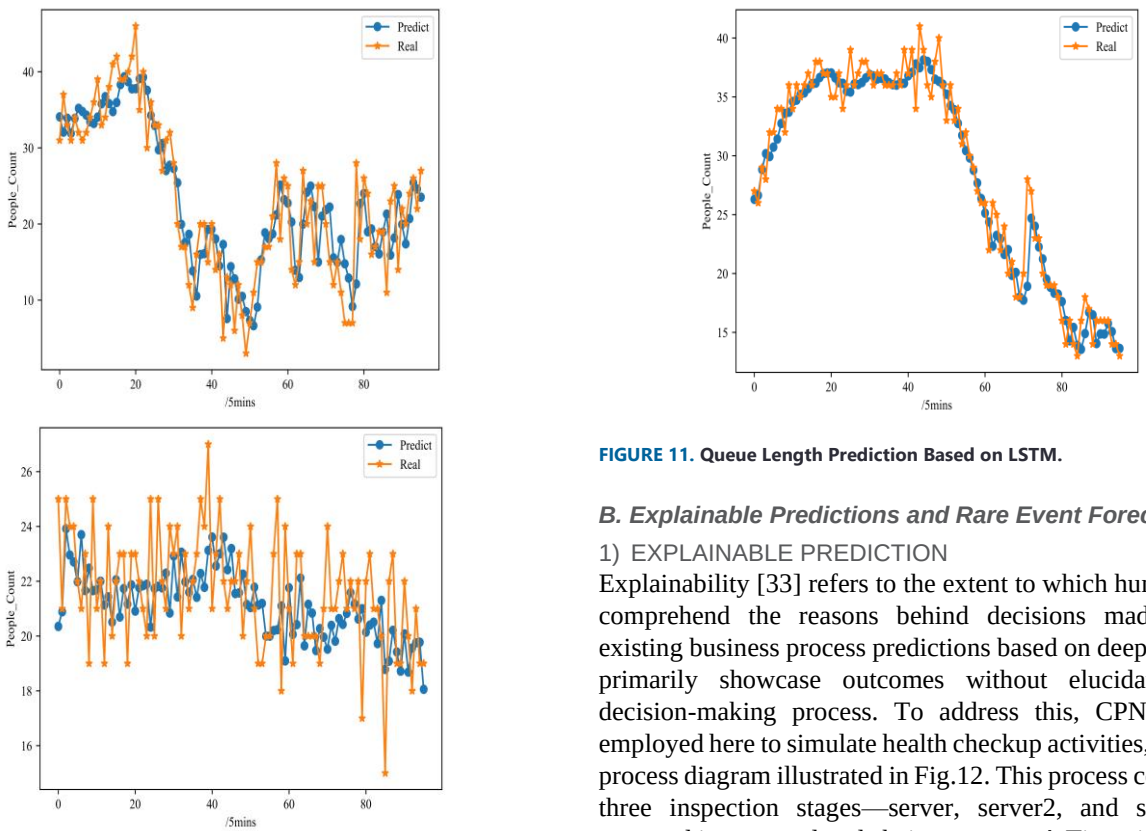


FIGURE 11. Queue Length Prediction Based on LSTM.

B. Explainable Predictions and Rare Event Forecasting

1) EXPLAINABLE PREDICTION

Explainability [33] refers to the extent to which humans can comprehend the reasons behind decisions made. Most existing business process predictions based on deep learning primarily showcase outcomes without elucidating the decision-making process. To address this, CPNTools is employed here to simulate health checkup activities, with the process diagram illustrated in Fig.12. This process comprises three inspection stages—server, server2, and server3—arranged in an unordered chain structure. A Timer is utilized to limit the number of server instance executions, with "@+3" in the top-right corner of the server transition indicating a 3-minute duration for its execution. By harnessing process predictions to determine the number of individuals in queues during various time periods and simulating these with Others

elements, a higher level of explainability is achieved compared to black-box prediction approaches.

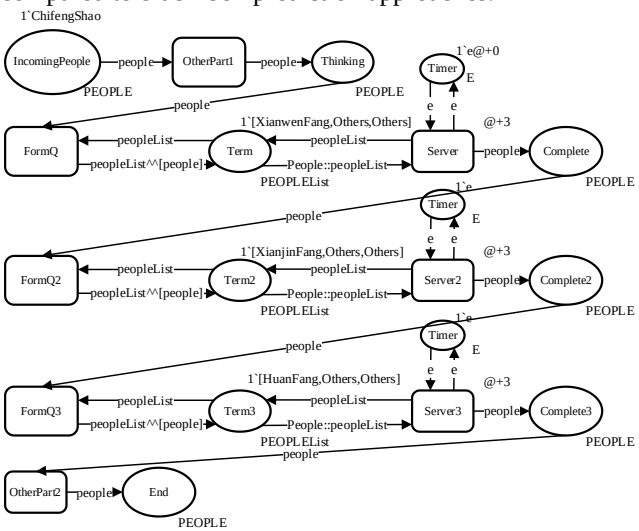


FIGURE 12. Simulation of Health Checkup Process Based on CPNtool.

For the three non-interfering inspections, there exist six distinct sequences. Regarding subjects arriving at the hospital at different times and undergoing inspections in varying orders, Fig.13 illustrates the total duration for each combination of tests.

The optimal inspection sequence is not consistent for subjects arriving at differing times. For instance, if one arrives at 8:30, the fastest sequence is server3, server2, server1. Conversely, at 9:00, both sequences—server2, server3, server1 and server3, server2, server1—yield equivalent durations. Arriving at 9:30, the most efficient sequence becomes server2, server3, server1.

2) RARE EVENT FORECASTING

During the health checkup process, when a participant abandons a current examination due to an excessively long queue, this behaviour is classified as a low-frequency event. For the three examinations in question, the precondition transitions for such low-frequency events can be established as follows: the waiting counts at server, server2, and server3 exceed 90% of their respective daily maximums. In other words, upon arrival, if the subject finds the queues for all three checks surpassing expectations, it triggers the low-frequency behaviour, leading to the abandonment of the health check.

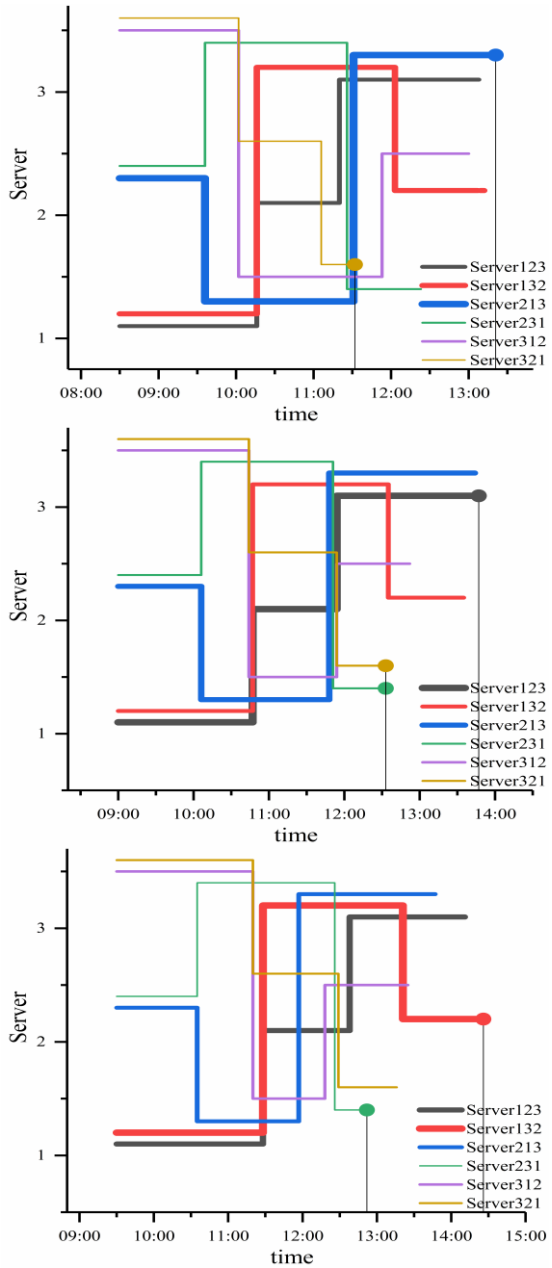


FIGURE 13. Duration Performance of Six Service Sequence Combinations.

As illustrated in Fig.14, the maximum waiting times for server, server2, and server3 on that day were 118.0224 minutes, 71.7515 minutes, and 114.4294 minutes, respectively. Consequently, the 90% thresholds were set at 106.2202 minutes, 64.5764 minutes, and 102.9865 minutes, respectively. Based on LSTM prediction outcomes, the timestamps when preceding transitions meet the 90% threshold are visualized in Fig.11.

According to Algorithm 3, it is revealed that at 9:00, 9:20, 9:30, 9:40, and within the interval of 9:50 to 10:00, all precondition transitions satisfied the criterion of exceeding 90% of the day's maximum waiting duration, thereby triggering the low-frequency behaviour of abandoning the health check.

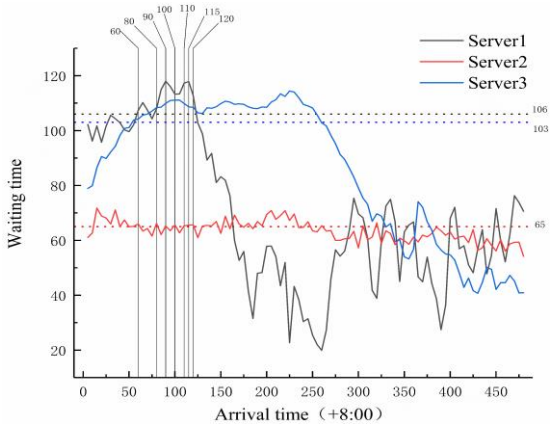


FIGURE 14. Timestamps When All Precondition Transitions Are Triggered.

V. CONCLUSION

The proposed object-centric business process forecasting methodology is a groundbreaking innovation, rooted in its ability to harness the power of deep learning for extracting critical insights from a blend of graphical and sequential data in activity logs. The AI model, designed with versatility in mind, can seamlessly adapt to a myriad of business contexts, ensuring its applicability across diverse domains.

Moving beyond conventional methods, the approach innovatively employs Petri net control flow modeling to dynamically construct predictive process flows based on the specific requirements of objects. This dynamic assembly not only streamlines business operations but also enables real-time responsiveness to changing conditions, thereby enhancing overall process efficiency.

The crux of the method lies in its sophisticated treatment of transition triggers. By leveraging the operational status of the Petri net and deep learning models, it forecasts activity states and the net's status, fostering a comprehensive understanding of multi-instance, multi-system dynamics. The shared transition prediction mechanism enhances the consideration of triggers across various instances, while the prediction of precondition transition sets bolsters the prediction of infrequent transitions. This addresses the challenge of forecasting low-frequency behaviors, even when data is scarce, thus improving the accuracy of process predictions.

Future research directions hold immense potential. One such avenue involves the exploration of temporal dynamics, delving into how the efficiency of transition execution evolves over time. This could provide deeper insights into process performance trends and enable more proactive management strategies. Another intriguing prospect is the utilization of alternative deep learning architectures in log data mining, which could potentially unlock new levels of predictive power and uncover hidden patterns.

In essence, this framework ushers in a new era of business process forecasting, characterized by enhanced nuance and adaptability. It paves the way for more effective decision-making in intricate business landscapes, where real-time understanding and anticipation of process dynamics can significantly impact operational success. The implications of this research extend beyond the realm of process forecasting, offering potential transformative impacts on business strategy, operations, and decision support systems. As such, it holds great promise for shaping the future of process management in the digital age.

REFERENCES

[1] GRISOLD T, GROSS S, STELZL K, et al. The five diamond method for explorative business process management[J]. Business & Information Systems Engineering, 2022, 64(2): 149-166.
[2] MALINOVA M, GROSS S, MENDLING J. A study into the contingencies of process improvement methods[J]. Information Systems, 2022, 104: 101880.
[3] VAN DER AALST W M P. Business process management: a comprehensive survey. ISRN Softw Eng 2013: 1–37[Z]. 2013.
[4] BEVERUNGEN D, BUIJS J C, BECKER J, et al. Seven paradoxes of business process management in a hyper-connected world[J]. Business & Information Systems Engineering, 2021, 63: 145-156.
[5] KERMANI MA, SEDDIGHI HR, MAGHSOUDI M. Revolutionizing Process Mining: A Novel Architecture for ChatGPT Integration and Enhanced User Experience through Optimized Prompt Engineering. arXiv preprint arXiv:2405.10689. 2024 May 17.
[6] ZHANG L, FANG X, SHAO C, et al. Alternative Model Repair Based on the Predictable Fitness[J]. Journal of Computer Research and Development, 2022, 59(11): 2618-2634. (in Chinese)
[7] FANG X, ZHAO F, FANG H, et al. The Fusion Analysis Method about the Change Region of the Business Process Model Based on Behavior Inclusion in Petri Net[J]. Chinese Journal of Computers, 2018, 41(3): 695-708. (in Chinese)
[8] SHAO C, FANG X, WANG W. Generation of controlled logs based on extended Petri Net[J]. Computer Engineering and Design, 2022, 43(3): 876-885. (in Chinese)
[9] TAX N, VERENICH I, LA ROSA M, et al. Predictive Business Process Monitoring with LSTM Neural Networks[M]DUBOIS E, POHL K. Advanced Information Systems Engineering: Vol. 10253. Cham: Springer International Publishing, 2017: 477-492.
[10] MEHDIYEV N, EVERMANN J, FETTKE P. A Novel Business Process Prediction Model Using a Deep Learning Method[J]. Business & Information Systems Engineering, 2020, 62(2): 143-157.
[11] PASQUADIBISCEGLIE V, APPICE A, CASTELLANO G, et al. Using Convolutional Neural Networks for Predictive Process Analytics[C]2019 International Conference on Process Mining (ICPM). Aachen, Germany: IEEE, 2019: 129-136.
[12] WEYTJENS H, DE WEERDT J. Learning Uncertainty with Artificial Neural Networks for Improved Remaining Time Prediction of Business Processes[M]. arXiv, 2021.
[13] CAMARGO M, DUMAS M, GONZÁLEZ-ROJAS O. Learning Accurate LSTM Models of Business Processes[C]//HILDEBRANDT T, VAN DONGEN B F, RÖGLINGER M, MENDLING J. Business Process Management. Cham: Springer International Publishing, 2019: 286-302.
[14] PASQUADIBISCEGLIE V, APPICE A, CASTELLANO G, et al. ORANGE: Outcome-Oriented Predictive Process Monitoring Based on Image Encoding and CNNs[J]. IEEE Access, 2020, 8: 184073-184086.
[15] TEINEMAA I, DUMAS M, ROSA M L, et al. Outcome-Oriented Predictive Process Monitoring: Review and Benchmark[J]. ACM Transactions on Knowledge Discovery from Data, 2019, 13(2): 1-57.
[16] FOLINO F, FOLINO G, GUARASCIO M, et al. Learning Effective Neural Nets for Outcome Prediction from Partially Labelled Log Data[C]//2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). Portland, OR, USA: IEEE, 2019: 1396-1400.
[17] VAN HOUDT G, MOSQUERA C, NÁPOLES G. A review on the long short-term memory model[J]. Artificial Intelligence Review, 2020, 53(8): 5929-5955.

[18] JORDAN M I, MITCHELL T M. Machine learning: Trends, perspectives, and prospects[J]. Science, 2015, 349(6245): 255-260.
[19] MITCHELL T, BUCHANAN B, DEJONG G, et al. Machine Learning[J]. Annual Review of Computer Science, 1990, 4(1): 417-433.
[20] CHOWDHARY K, CHOWDHARY K R. Natural language processing[J]. Fundamentals of artificial intelligence, 2020: 603-649.
[21] OTTER D W, MEDINA J R, KALITA J K. A survey of the usages of deep learning for natural language processing[J]. IEEE transactions on neural networks and learning systems, 2020, 32(2): 604-624.
[22] SHAUKAT K, LUO S, VARADHARAJAN V, et al. A survey on machine learning techniques for cyber security in the last decade[J]. IEEE Access, 2020, 8: 222310-222354.
[23] CHANDOLA V, BANERJEE A, KUMAR V. Anomaly detection: A survey[J]. ACM computing surveys (CSUR), 2009, 41(3): 1-58.
[24] LU D, WENG Q. A survey of image classification methods and techniques for improving classification performance[J]. International journal of Remote sensing, 2007, 28(5): 823-870.
[25] FRAWLEY W J, PIATETSKY-SHAPIRO G, MATHEUS C J. Knowledge discovery in databases: An overview[J]. AI magazine, 1992, 13(3): 57-57.
[26] BIGGIO B, ROLI F. Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning[J]. Pattern Recognition, 2018, 84: 317-331.

SHAO CHIFENG received the B.S. and M.S. degrees from the Anhui University of Science and Technology, China, in 2016 and 2021, respectively. He is currently pursuing a PhD degree in information security engineering at the Anhui University of Science and Technology.

He is a Teaching Assistant at the Computer Department, College of Information and Network Engineering, Anhui Science and Technology University, China. His research interests include Petri nets, process mining and AI Security.

WANG QIANQIAN received an M.A. degree from the Anhui University of Science and Technology, China, in 2020.

She is currently a Lecturer at the Department of Mathematics, Anhui Science and Technology University, China. Her research interest is Petri net theory and applications.

[27] MITCHELL T M. Does Machine Learning Really Work? [J]. AI Magazine, 1997, 18(3): 11-11.
[28] NEUMANN J von, MORGENSTERN O. Theory of Games and Economic Behavior: 60th Anniversary Commemorative Edition[M]Theory of Games and Economic Behavior. Princeton University Press, 2007.
[29] WANG C Y, BOCHKOVSKIY A, LIAO H Y M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors[C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023: 7464-7475.
[30] YU Y, SI X, HU C, et al. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures[J]. Neural Computation, 2019, 31(7): 1235-1270.
[31] JENSEN K, KRISTENSEN L M, WELLS L. Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems[J]. International Journal on Software Tools for Technology Transfer, 2007, 9(3): 213-254.
[32] WU Z. Introduction to Petri Nets[M]. Beijing: Mechanical Industry Press, 2006. (in Chinese)
[33] MILLER T. Explanation in artificial intelligence: Insights from the social sciences[J]. Artificial Intelligence, 2019, 267: 1-38.