Business Process Representation Learning

Dissertation zur Erlangung des Grades eines
Doktors der Wirtschaftswissenschaft (doctor rerum oeconomicarum)
der Fakultät für Empirische Humanwissenschaften und
Wirtschaftswissenschaft der Universität des Saarlandes

M. Sc. Peter Pfeiffer

UNIVERSITÄT
DES
SAARLANDES

Saarbrücken, April 2025

Erstgutachter	Univ.-Prof. Dr. Peter Loos
Zweitgutachterin	Prof. Dr. Jana-Rebecca Rehse
Amtierender Dekan	Univ.-Prof. Dr. Axel Mecklinger

Tag der Disputation 07.11.2025

Abstract

Companies generate vast amounts of digital trace data, stored in event logs, when performing business processes. Such data is rich in information, describing what actions have been performed, by whom, at what time, and in what context. Process mining (PM), a data-driven discipline in the field of business process management (BPM), deals with detailed analysis of event data on a process instance basis, aiming to obtain operational insights into how companies' processes have actually been performed. It has evolved as a major field of research, offering approaches for automatic event log analysis to enhance efficiency or ensure compliance, with growing industrial adoption. As PM applications increasingly require machine learning (ML) techniques, particularly for forward-facing support of business processes, the question becomes how to apply such techniques to event log data, since the characteristics of event logs and the processes they describe differ from common data modalities such as images and text. Given the complexity of event log data, applying ML techniques poses significant challenges, requiring adaptations of existing or the development of customized ML techniques to account for the characteristics of event logs. Representation learning, a research field in ML, deals with learning representations from data that make solving downstream tasks more effective and efficient by overcoming the laborious and expensive task of manual feature engineering. This thesis introduces the problem of business process representation learning, i.e., learning representations from event logs for solving BPM tasks like process prediction, anomaly detection, or task abstraction. By developing, analyzing, and applying various process representation models (RPMs), this thesis contributes to three research areas. First, on how to design PRMs in terms of architecture and training procedures to learn representations that follow the general priors of representation learning. Second, on the assessment of how well the models have learned characteristics of event log data and the underlying processes. Finally, on the application side, for which BPM tasks PRMs can be used, and how well they perform. The results demonstrate that PRMs are capable of learning accurate, context-specific representations from different concepts within event logs. These representations adhere to the principles of representation learning and can be utilized to solve real-world BPM tasks efficiently. Thereby, the results advance the fields of PM and ML, and especially their interplay.

Danksagung

Ich bin dankbar für die Gelegenheit, mit einer Vielzahl von Menschen zusammenzuarbeiten, deren Erfahrung, Wissen und Engagement diese Dissertation erst ermöglicht haben. Meinem Doktorvater, Prof. Dr. Peter Loos möchte ich für die Betreuung meiner Dissertation bedanken. Ich habe die Freiheit, welche mir mit dieser Position eröffnet wurde, immer zu schätzen gewusst. Deine stets konstruktive Art und positive Einstellung haben mir in dieser Zeit sehr geholfen. Insbesondere Dir möchte ich danken, mir die Möglichkeit gegeben zu haben diese Arbeit am Institut für Wirtschaftsinformatik (IWi) des Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) an der Universität des Saarlandes durchzuführen. In meiner Zeit als wissenschaftlicher Mitarbeiter, beginnend mit dem Junior Researcher im September 2019 bis hin zur Vollendung dieser Arbeit im April 2025 durfte ich wertvolle Erfahrungen im wissenschaftlichen sowie dem projektgetriebenen Arbeiten in einer Vielzahl von verschiedenen Projekten sammeln.

Ebenso möchte ich mich bei Prof. Dr. Jana-Rebecca Rehse für ihre Unterstützung sowie die Übernahme der Zweitkorrektur bedanken. Bereits während meines Studiums hast Du, damals noch selbst Doktorandin am IWi, mich fachlich sowie menschlich unterstützt. In den unzähligen Gesprächen und Diskussion mit Dir habe ich viele hilfreiche Dinge zur Promotion gelernt und die Ideen dieser Arbeit verfeinert.

Danken möchte ich auch allen Kollegen und promovierenden Mitstreitern am IWi, auch ehemalige, mit denen gemeinsam sich die Anstrengung und Hürden dieser Zeit überwinden ließen. Eure offenen Ohren und wertvolle Tipps haben zum erfolgreichen Abschluss dieser Arbeit beigetragen. Insbesondere Johannes Lahann, Nijat Mehdiyev, Sarah Rübel, Oliver Gutermuth, Chantale Lauer, Constantin Houy, Brian Willems, Alexander Berrang, Martin Scheid, Philip Hake, Sharam Dadashnia, Manuel Zapp sowie Prof. Dr. Peter Fettke möchte ich danken. Den wissenschaftlichen Hilfskräften, die mich in der Zeit am DFKI in den Projekten aber auch zu den Themen dieser Arbeit unterstützt haben möchte ich ebenso meinen Dank aussprechen wie den externen Projektpartner.

Bereits am Ende meines Bachelorstudienganges im Jahre 2017 durfte ich wertvolle Auslandserfahrung bei einem Praktikum für Carl Zeiss in Shanghai, China sammeln. Dir, lieber Hexin Wang, möchte ich herzlichst für diese Möglichkeit Danken. Gleichwohl möchte ich mich bei Prof. Dr. Avigdor Gal für das Hosting zum Forschungsaufenthalt an der Technion in Haifa, Israel im Sommer 2023 bedanken. Die

Erfahrungen dieser Zeit und eure Persönlichkeiten werden mein Leben auch weiterhin prägen.

Daneben hatte ich das große Glück viele schöne Stunden mit einer Reihe von Menschen außerhalb des Büroalltags auf Konferenzen oder Forschungsseminaren zu verbringen, aus denen auch oftmals eine engere Zusammenarbeit entstanden ist. Besonders Adrian Rebmann, Michael Grohs, Prof. Dr. Han van der Aa, Luka Abb, Diana Sola und Timo Nolle möchte ich hierbei erwähnen.

Schließlich gehört mein Dank auch meiner Familie und Freunden. Meiner Frau Larissa, für Dein Verständnis und Rückhalt, gerade in schwierigen Zeiten sowie die schönen Momente, die wir jetzt schon mit unserem Sohn Jakob verbringen dürfen. Meinen Eltern Stefan und Heidrun sowie Stiefeltern Manuela und Martin die mir all das erst ermöglicht und mich uneingeschränkt unterstützt haben. Ich kann von Glück sprechen, dass ihr mir ermöglicht habt, meinen Vorstellungen nachzugehen. Für die Ablenkung von der Dissertation und der Arbeit bedanke ich mich bei meinen Freunden – Boggi und Bert, auf eure Kappe geht auch ein Teil hiervon.

Zuletzt möchte ich noch eine Anekdote zu meinem verstobenen Großvater, Opa Rudi, erzählen. Dieser hatte mich in meiner Jugend eines Abends beim Spielen eines Computerspieles gesehen. Als er sah, wie schnell meine Finger über die Tastatur huschten, um die Computerfigur zu steuern, sprach er mich an und ermutigte mich ein Buch zu schreiben, da ich ja so flink mit dem Computer umgehen kann. Er würde auch gerne ein Buch schreiben — wenn er denn nur so gut mit dem Computer arbeiten könnte. Seine Idee erschien mir etwas weit hergeholt. Ich dachte nicht ansatzweise daran ein Buch zu schreiben. Ich bin mir auch bis heute nicht sicher, ob er tatsächlich beabsichtigte, dass ich ein Buch schreibe. Vielleicht wollte er auch einfach nur dass ich aufhöre das Computerspiel zu spielen. Während der Arbeit an dieser Thesis kamen mir seine Worte aber immer wieder in den Kopf. Die vergangenen Jahre waren eine sehr bereichernde Zeit, in der ich wahnsinnig viel zu den aktuellsten Technologien lernen durfte und die meinen weiteren Lebensweg prägen wird. Vielleicht wollte mein Opa mir damals auch die Möglichkeiten aufzeigen, die sich mit der für ihn neuen Technologie ergaben. In diesem Sinne kann ich jeden nur ermutigen, mit Technologien, insbesondere neuartigen, zu experimentieren, sie weiterzuentwickeln oder anderweitig sinnvoll zu nutzen. Und vielleicht doch darüber ein Buch zu schreiben.

Peter Pfeiffer

Contents

List of Figures

List of Tables

List of Abbreviations

AE	Autoencoder
AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations From Transformers
BPDP	Business Process Deviation Prediction
BPIC	Business Process Intelligence Challenge
BPM	Business Process Management
CIBE	Complex-Index-Based Encoding
CNN	Convolutional Neural Network
CV	Computer Vision
DAPNN	Detection of Anomalous Processes through Neural Networks
DFG	Directly-Follows-Graph
DSR	Design Science Research
FNN	Feedforward Neural Network
GAF	Gramian Angular Fields
GNN	Graph Neural Network
ICPM	International Conference on Process Mining
IS	Information Systems
LLM	Large Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
MPPN	Multi-Perspective Process Network
MobIS	Model-based Compliance in Information Systems
NLP	Natural Language Processing

PCA	Principal component analysis
PDC	Process Discovery Contest
PM	Process Mining
PPA	Predictive Process Analytics
PPM	Predictive Process Monitoring
PRM	Process Representation Model
RNN	Recursive Neural Network
RA	Research Area
RQ	Research Question
SOTA	State-of-the-art
XAI	Explainable Artificial Intelligence

Part I

Synopsis

Chapter 1

Introduction

1.1 Motivation

With increasing digitalization and the need for more efficiency in their operations, companies strive to analyze and optimize their processes from large amounts of stored data — for instance, through data- and artificial-intelligence (AI)-driven techniques (van der Aalst 2022; Weinzierl et al. 2024). Process mining (PM) is a data-driven discipline in business process management (BPM) that helps organizations analyze data from process executions. For instance, for uncovering the structure of their processes, finding bottlenecks, or identifying optimization opportunities (van der Aalst 2016). It allows companies to understand how their processes are executed in reality and improve them. In BPM, a business process is a coordinated procedure, consisting of activities and decisions performed in an organizational and technical environment to realize a business goal (Dumas et al. 2018; Weske 2019). Executing a business process with the support of information systems creates digital traces (vom Brocke et al. 2024).

The event log L stores the digital traces over a period as a collection. A single trace represents a process instance as a sequence of events $\langle e_0, e_1, \ldots, e_n \rangle$ with events being tuples of attribute values. Table 1 shows one trace from the MobIS event log (Scheid et al. 2018). In general, each event contains at least the mandatory information about the *activity* that has been performed, the *timestamp* when it was performed, as well as the *caseID* of the respective process instance. Additionally, there can be various other event attributes that describe the event's context. For instance, who executed the activity (attributes *type* and *resource*), or *cost* of the travel. Besides event attributes, trace attributes are characterized by having the same value for all events in a particular trace — for instance, the *travel start* date in Table 1. Each attribute gives one perspective on the event log, making it rich in information. The event log organizes the events per process instance, but in reality, events happen over time — and not necessarily per trace. We refer to the underlying system that generates the events as S. It contains the same events stored in L but represents process behavior in a less trace-encapsulated but time-ordered way.

Table 1: Case 1469 of the MobIS event log (Scheid et al. 2018).

activity	caseID	start	end	type	resource	travel start	travel end	cost
file travel request	1469	02.01.2017 05:59	02.01.2017 06:15	Employee	GV7676	09.04.2017	23.04.2017	4418,43
check if travel request needs preliminary price inquiry	1469	02.01.2017 06:15	02.01.2017 06:16			09.04.2017	23.04.2017	
decide on approval requirements	1469	02.01.2017 06:16	02.01.2017 06:16			09.04.2017	23.04.2017	
forward request to approver	1469	02.01.2017 06:16	02.01.2017 06:17			09.04.2017	23.04.2017	
decide on request	1469	02.01.2017 13:00	02.01.2017 13:04	Manager	TD8091	09.04.2017	23.04.2017	
check if booking is necessary	1469	04.01.2017 13:28	04.01.2017 13:29	Travel Department	JV9257	09.04.2017	23.04.2017	
prepare booking proposal	1469	06.01.2017 12:53	06.01.2017 14:49	Travel Department	JV9257	09.04.2017	23.04.2017	
send booking proposal to employee	1469	10.01.2017 14:10	10.01.2017 14:12	Travel Department	JV9257	09.04.2017	23.04.2017	
check booking proposal	1469	10.01.2017 17:34	10.01.2017 17:34	Employee	GV7676	09.04.2017	23.04.2017	
check if travel request is still needed and up to date	1469	16.01.2017 05:59	16.01.2017 06:46	Employee	GV7676	09.04.2017	23.04.2017	
request update of the booking proposal	1469	16.01.2017 06:46	16.01.2017 06:46	Employee	GV7676	09.04.2017	23.04.2017	
prepare booking proposal	1469	18.01.2017 14:30	18.01.2017 15:49	Travel Department	JV9257	09.04.2017	23.04.2017	
send booking proposal to employee	1469	23.01.2017 13:14	23.01.2017 13:16	Travel Department	JV9257	09.04.2017	23.04.2017	
check booking proposal	1469	23.01.2017 14:56	23.01.2017 14:59	Employee	GV7676	09.04.2017	23.04.2017	
book travel	1469	26.01.2017 15:58	26.01.2017 18:22	Travel Department	JV9257	09.04.2017	23.04.2017	
check if expense documents exist	1469	24.04.2017 15:22	24.04.2017 15:34	Employee	GV7676	09.04.2017	23.04.2017	
file travel expense report	1469	24.04.2017 15:34	24.04.2017 15:48	Employee	GV7676	09.04.2017	23.04.2017	
confirm travel expense report	1469	24.04.2017 15:48	24.04.2017 15:52	Employee	GV7676	09.04.2017	23.04.2017	
decide on travel expense approval	1469	25.04.2017 13:59	25.04.2017 14:06	Manager	JH852	09.04.2017	23.04.2017	
send original documents to archive	1469	28.04.2017 06:59	28.04.2017 07:53	Employee	GV7676	09.04.2017	23.04.2017	
calculate payments	1469	10.05.2017 08:59	10.05.2017 09:21	Accounting	GR7476	09.04.2017	23.04.2017	4364,5
pay expenses	1469	19.05.2017 14:36	19.05.2017 14:47	Accounting	GR7476	09.04.2017	23.04.2017	1679,5

Machine Learning in Process Mining Recently, an increasing number of PM approaches have made use of machine learning (ML) techniques, a subfield of AI, for analytical or predictive applications (Neu et al. 2021). The most prominent example is the prediction of future process behavior based on the observed process behavior. Such approaches, which are commonly referred to as process prediction, enable forward-directed support. They extend the scope of process mining from descriptive to predictive applications. For instance, for predicting unwanted process behavior and acting proactively instead of analyzing deviations that have already happened in event logs and acting reactively (Francescomarino and Ghidini 2022).

Process prediction has a long history, with first approaches emerging at the beginning of the 2000s (Reijers 2007). These approaches often relied on explicit representations of the process behavior, e.g., transition systems (Aalst et al. 2011), Markov models (Ruta and Majeed 2011), or formal models like Petri nets (van der Aalst 2013), with machine-learning-based prediction methods applied on top of it.

With the emergence of deep learning in the mid-2010s (Lecun et al. 2015), process prediction quickly adopted this new technique. The usage of deep recurrent neural networks for process prediction was motivated by their success in language modeling in the field of natural language processing (NLP), and the observation that language modeling and process prediction share some characteristics.[1] For instance, one can predict the next activity from a sequence of observed events analogous to predicting the next word in a sentence (Evermann et al. 2016).

[1] These similarities have motivated the usage of language modeling techniques for process prediction already before the success of deep learning, e.g., grammatical inference and n-gram approaches (Breuker et al. 2016).

Being able to train deep neural networks on large amounts of data marked a significant step in the field of ML. Instead of learning a mapping from (hand-crafted) features to outputs, such models can discover the features themselves (Goodfellow et al. 2016, p.4). With their introduction to the task process prediction, explicit models such as Petri nets were no longer required to describe the process behavior. Instead, neural networks are trained on sequences of previous events to predict the next step, the outcome, or other future characteristics (Evermann et al. 2016; Rehse et al. 2018; Tax et al. 2017). Thereby, the ML model[2] learns an internal, "implicit" (Evermann et al. 2017) representation of the process behavior.

Representation Learning Such internal representations of data that deep learning models learn in their layers during training (Hinton et al. 2006) are of interest in the research field of representation learning (Bengio et al. 2013; Goodfellow et al. 2016). As an area in ML, it is concerned with learning representations from data that make solving tasks more effective and efficient (Goodfellow et al. 2016, p.173 ff.). It combines various concepts from deep learning (Goodfellow et al. 2016, p.557), especially self- or semi-supervised techniques (Ziv and Lecun 2024). By learning representations from data, one overcomes the laborious and expensive task of manual feature engineering (Goodfellow et al. 2016, p.4) by exploiting the enormous amounts of unlabeled data available (Bishop and Bishop 2024, p.188). Further, learned representations often perform much better and can be easily adapted to new tasks (Goodfellow et al. 2016, p.4).

Learned representations are numerical feature vectors $r \in R^n$, generated by the ML model, as the output of one of its layers; typically one of the last layers. By training a neural network, we do not explicitly tell the model how the representation r should look. Instead, the representation is "shaped" by the training objective. Some general priors for useful representations exist that can guide the development of dedicated techniques (Bengio et al. 2013). These priors aim to learn the underlying factors that generated the data. Learning the multiple explanatory factors of variation that generated the data, the hierarchical organization of concepts in the data, or shared factors across tasks are, among other priors, crucial for developing effective representations learning approaches (Bengio et al. 2013; Goodfellow et al. 2016).

Autoencoders (AE) are an example of representation learning models. They are trained to preserve as much information from the data in their internal representation between the encoder and decoder as possible (Goodfellow et al. 2016, p.4). Convolutional neural networks (CNN) trained to classify objects in a self-supervised way learn representations of samples that resemble semantic clusters (Ben-Shaul et al. 2023). Language models such as BERT or GPT are referred to as "language representation models" (Devlin et al. 2019). Their ability to uncover the structure of the data plays a vital role in making solving downstream tasks easier (Bishop and Bishop 2024, p.22) or enabling few-shot learning (Goodfellow et al. 2016, p.536). Representation learning is central to deep learning (Bishop and Bishop 2024, p.189) — and the success of large language models (LLMs) (Bishop and Bishop 2024, p.6).

[2] If not explicitly stated otherwise, we refer to an ML model as being a deep neural network.

Process Representation Models (*PRMs*) In process mining, learning representation from event logs has not been researched as extensively as for images or natural language. Most predictive approaches on event logs are inspired by NLP techniques but solve a single task at a time. However, due to differences between natural language and event sequences, learning the explanatory factors of variation in event logs requires developing new neural network architectures and training procedures that adapt to the characteristics of events, traces, and processes. Some characteristics, like the procedural nature of business processes or the interplay between different attributes within a trace, are considered an open challenge for ML techniques (Ceravolo et al. 2023). Other challenges deal with analyzing the organization of concepts in the learned embeddings or evaluating the accuracy of models in representing the process. Finally, it has to be researched for which tasks learned representations can be used and how well *PRMs* perform.

Central Research Question This thesis explores representation learning for event log data, driven by the curiosity whether the success of representation learning for other modalities also works for processes in event logs. The central research question that this thesis contributes to is about realizing process representation models that learn accurate representations r of process concepts from event logs for solving various BPM tasks (see Section 1.4).

In accordance with representation learning models for other data modalities, we refer to a representation learning model for event logs as a process representation model, short *PRM*. The idea of business process representation learning is closely related to the idea of process mining, particularly process discovery. However, we aim to learn a (high-dimensional) feature vector r, sometimes also called an embedding, instead of a graph-like or formal model. Thus, we focus less on representations that are communicable and more on learning accurate representations of the behavior of the underlying concepts in the data. This will be discussed in more detail in the following sections.

1.2 Structure

The thesis is structured in two parts. Part I *Synopsis* summarizes the contributions of this thesis and is structured into five chapters.

This chapter, Chapter 1, introduces the thesis. The following sections outline the research objective, challenges, questions, and methods, giving an overview of the publications and contributions made in this PhD thesis. The following three chapters present the results organized by research areas. Chapter 2 presents results on the architecture and training of *PRMs*, Chapter 3 presents results on the evaluation of *PRMs*, and Chapter 4 presents results on the application of *PRMs*. Finally, Chapter 5 concludes Part I by summarizing the contributions, discussing limitations and implications, and outlining future work.

Part II contains the individual publications of this thesis.

1.3 Research Areas and Challenges[3]

This thesis researches representation learning for event log data, specifically to obtain knowledge on how representations r of concepts in event logs can be learned and whether they are useful in application. This knowledge contributes to developing a business process representation learning model *PRM*.

To achieve this objective, three research areas (RA) with accompanying research questions (RQ) (see Section 1.4) have been defined. Each research area focuses on a particular aspect of *PRMs* concerning their architecture and training (RA 1), their intrinsic assessment (RA 2), and their extrinsic assessment through evaluation and demonstration in application (RA 3). Each area comes with its challenges, technical approaches, and research needs.

RA 1 *PRM* Architecture and Training

RA 2 *PRM* Assessment (Intrinsic Evaluation)

RA 3 *PRM* Application and Demonstration (Extrinsic Evaluation)

Figure 1 shows how *PRMs* are embedded into the research areas. The data modality *PRMs* are designed for are event logs L containing digital traces of a process performed in the system S. A process representation model is learning representations r of the process's concepts captured in L. These representations are assessed without application context before being used to solve real-world tasks.

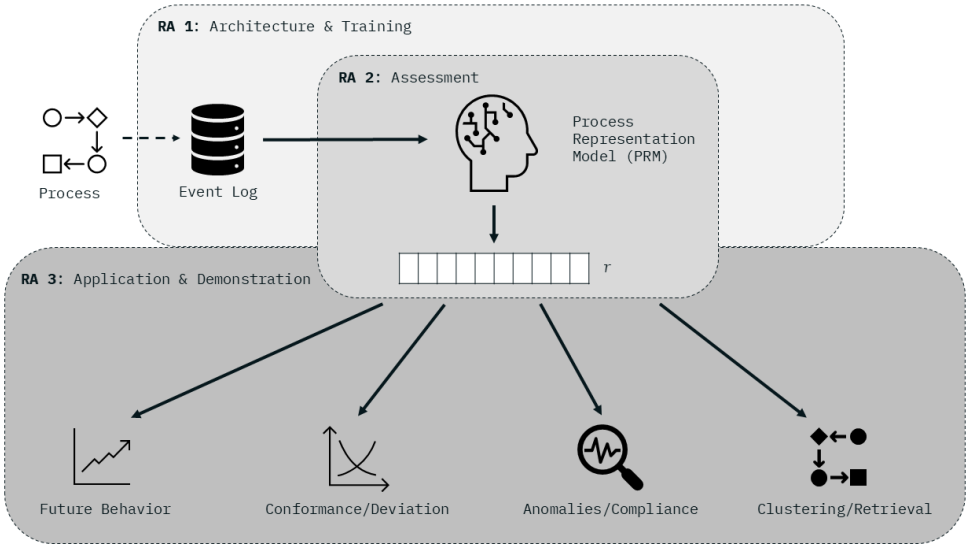


Figure 1: Overview of business process representation learning embedded into the research areas RA1-3 (own figure).

[3] This section is based on Pfeiffer (2022) and Pfeiffer (2023) and has been extended.

The separation into three research areas follows from three early observations made in this doctoral thesis. First, applying neural networks to event logs is challenging since existing types of neural networks do not fully account for the characteristics of the data, which requires extensive modifications to their architecture and training procedure (RA 1). Second, differentiating between the performance of a *PRM* in learning the data, i.e., the process (RA 2) and its performance on downstream tasks in application (RA 3), similar to how language representation models are evaluated (Jurafsky and Martin 2025, p.38), allows gaining detailed insights into their internal working, aiding further development. Further, the three research areas align with typical characterizations of machine learning models and the research method used. Technically, since machine learning can be defined over the task T (RA 3), experience E (RA 1), and performance measure P (RA 2 and RA 3) as described by Goodfellow et al. (2016, p.103). Methodologically, since the research areas are aligned with the design and empirical cycles (see Section 1.5).

The first research area (RA 1) deals with building model architectures and training procedures that fit event logs L and allow for learning the underlying process that generated the data. This requires a data and business understanding to know which features are essential for application since the architecture and training method of the *PRM* are decisive for how the representation r will be designed.

Research area 2 (RA 2) is concerned with assessing how accurately the learned representations r represent the process. For instance, how the *PRM* clusters process instances internally, or how well the *PRM* deals with changes in the process. This type of evaluation is called intrinsic and assesses the model itself and not specifically for one task (Jurafsky and Martin 2025).

Research area 3 (RA 3) analyzes how well the *PRM* performs on actual tasks and in comparison to existing, specialized methods. This involves benchmarking against state-of-the-art (SOTA) approaches and baselines on various tasks. Further, it is demonstrated how the learned representation r can be used for different tasks by clustering, fine-tuning, or additional end-to-end training of the whole *PRM*. This end-to-end evaluation is called extrinsic (Jurafsky and Martin 2025).

The separation into three research areas thus enables a cyclic development approach where detailed and purpose-oriented contributions aligned with model usage are obtained. The following presents the current state, challenges, and research needs per research area in more detail.

1.3.1 *PRM* Architectures and Training Procedures

What makes designing a neural network architecture that learns accurate representations from event logs challenging is adapting it to the specific characteristics of business processes. Representation learning aims to learn the underlying factors of the data-generating process (Goodfellow et al. 2016, p.552), for which some general priors exist that can be translated to business process representation learning. For instance, a *PRM* must be capable of processing all relevant information in the event

log to learn the explanatory factors of variation in business processes (see *multiple explanatory factors* (Goodfellow et al. 2016, p.555)), i.e., it must be capable of considering the event attributes. Event attributes are important for many BPM tasks like monitoring and analysis (Weinzierl et al. 2024) and essential information for prediction (Evermann et al. 2016). The hierarchical organization of the data as events being characterized through a set of different event attributes, traces being sequences of events, and the processes being a collection of concurrently running traces with interdependencies is also important to consider (see *hierarchical organization of explanatory factors* in (Goodfellow et al. 2016, p.555))[4]. Another critical characteristic of event log data is the procedural nature of the process that the log describes, which manifests as traces in event logs being sequences of events (Ceravolo et al. 2023). Considering these characteristics, designing a network architecture[5] that can process and learn from it is a challenge on its own.

We highlight this challenge using the MobIS event log. A trace of this event log is displayed in Table 1 and is considered a typical example of such data. Besides the mandatory attributes *activity* and timestamps (*start* or *end*), it contains various other attributes. However, note that the *caseID* attribute is always ignored in process prediction, given that it is a unique identifier for traces. Attributes *resource* and *type* describe the person and the department of the person who executed the *activity*. Additionally, the *travel start* and *travel end* denote the respective dates. Attribute *cost* signifies the cost related to the travel. We refer to each attribute as one perspective of the event log.

The first observation is that attributes are of different types. Attributes *activity*, *resource*, and *type* can be considered categorical, *start*, *end*, as well as *travel start* and *travel end* are temporal information, and *cost* is numerical. The *activity* contains semantic information, too. The timestamps contain different information. *travel start* and *travel end* relate to two days in a year. In contrast, the attributes *start* and *end* denote the execution time of the activity. The weekday can also be of interest for certain tasks.

As the learned representation is supposed to be used for various tasks, no available information should be ignored. For instance, detecting compliance issues requires analyzing all attributes and their interplay throughout the trace (Baier et al. 2020). A compliance issue can arise if, for example, the payment activity has been performed after the trip ended. This requires considering at least the attributes *activity*, *start*, and *travel end*. Another issue requires checking whether the travel request has been approved by the responsible manager of the employee who files the request, in case the *cost* exceeds a certain threshold. Other tasks where context information in conjunction with the control flow is of significant importance are deviation prediction (Grohs et al. 2023) or task abstraction (Rebmann et al. 2023). Including all available information is thus relevant for concepts at all hierarchical levels.

[4] Ruta and Majeed (2011) have described processes very similarly as "a hierarchical encapsulation of sequential data" (Ruta and Majeed 2011, p.389).

[5] With architecture, we refer to the type and structure of layers used in the *PRM* (Goodfellow et al. 2016, p.197).

The shown trace is from one event log only. Other event logs can comprise a very different composition of attributes. Some event logs only contain the attributes *activity* and *timestamp*. Other event logs, like BPIC 2017, include many more attributes than MobIS. There are also event logs with textual notes assigned to specific events (Cabrera et al. 2023) or CAD data that contains information about which process steps will be taken (Mehdiyev et al. 2022).

In general, a large variety of architectures exist that can be used to learn from data. Feed-forward neural networks are versatile and can approximate almost any function (Goodfellow et al. 2016, p.198). Nevertheless, specialized architectures have shown to perform better for specific data modalities (Goodfellow et al. 2016, p.167), by including priors on how to perceive the data.

For instance, recurrent neural networks (RNNs) are specialized in processing sequential data. They have been shown to work very well for text (sequences of words) or time series (sequences of numerical values). Convolutional neural networks (CNN) are specialized for data that has a grid-like topology (Goodfellow et al. 2016, p.330). They use the convolution function in their layers (Goodfellow et al. 2016, p.330) and are frequently used for images, which are typically represented as matrices. Another specialized neural network type is graph neural networks (GNN) for data with complex dependencies between objects, which have been extensively researched over the past years (Wu et al. 2021). Since graphs are a very generic data modality, GNNs can be used for many other data modalities, too. In recent times, attention-based network architectures (Vaswani et al. 2017), so-called transformer networks, have shown to work very well for multiple data modalities. Although initially developed for textual data (Vaswani et al. 2017), they have been adapted for images, videos, speech, and other domains (Bishop and Bishop 2024, p.357-403).

Choosing the right architecture is critical for successfully solving a ML task. By selecting a specific network architecture, one is implicitly stating some prior beliefs about what function they should learn (Goodfellow et al. 2016, p.201). The prior beliefs that existing network architectures place on the data do not necessarily match event log data. Recurrent neural networks are designed for one-dimensional, sequential data, while CNNs are optimized explicitly for two-dimensional, grid-like data (Bishop and Bishop 2024, p.407). Considering each perspective in an event log as one dimension, different tricks must be applied if processing data with more than one (for RNN) or more than two (for CNN) perspectives with existing architectures.

The initial paper that used LSTM models for event log data (Evermann et al. 2016) concatenated the *activity* and *resource* of events into a single string. They argued that event log data shares some characteristics with natural language, which is why LSTM models are a reasonable choice. The approach to concatenate attributes, however, does not scale, as adding more attributes exponentially increases the number of unique values, i.e., the vocabulary.[6] In turn, each element in the vocabulary is scarce, increasing the prediction task's difficulty (Evermann et al. 2016). Following

[6] In consequence, for the BPIC 2013 event log, they did not use the *resource* but the organizational group, which has fewer different values and is less granular.

approaches using LSTM models processed each perspective in a separate LSTM network, e.g., Tax et al. (2017), Nolle et al. (2016), or Gunnarsson et al. (2023). This, however, increases the model size and, therefore, does not scale very well, either. This problem is known as the curse of dimensionality in machine learning (Goodfellow et al. 2016, p.155). In addition, many approaches using RNNs do not explicitly encode the timestamps but use the time perspective only implicitly through the sequential processing of RNNs.

Other approaches transform traces to 2D matrices and utilize convolutional architectures, e.g., (Pasquadibisceglie et al. 2020), to overcome the scaling problem of RNNs. Their ability to process event log data, therefore, depends on how traces are transformed to 2D data and the ability of the CNN to capture the sequential nature. GNNs, which became popular recently, excel in learning local connections in graph-structured data. Event log data can also be represented as a graph with connections between different objects involved in the process (Adams et al. 2022). However, GNNs suffer in capturing long-range interactions and are therefore outperformed by other network types, e.g., transformer models on tasks requiring capturing long-range dependencies (Dwivedi et al. 2022).

In addition to the architecture of a neural network, training the network requires a training objective and method that gives the expected result. For representation learning, where we aim to obtain a versatile representation, adequate training procedures have to be defined that guide the network in learning to preserve the desired feature of the input. The difficulty in representation learning is that there is no clear target to optimize for as there is in supervised training settings (Bengio et al. 2013, p.5). The objective to learn a representation that makes solving other tasks easier cannot be translated to a cost function that we can minimize in the same way as in supervised settings (Bengio et al. 2013). Instead, one has to experiment with different training strategies and general-purpose priors, which makes solving other tasks easier.

The training methods are often coupled with the type of network. For instance, an autoencoder is usually trained in a self-supervised way by reconstructing the input. Thereby, a representation r is learned that is usually much lower in dimension than the input. For language modeling, predicting next or missing words in sentences has resulted in a representation that can be used for many different text classification tasks (Jurafsky and Martin 2025; Devlin et al. 2019). Other frequently used training strategies for representation learning include multi-view learning, multi-task learning, transfer learning, and unsupervised pre-training (Goodfellow et al. 2016).

Similar to the model architecture, the training strategies have to be adapted to fully account for event log and process characteristics. If using multiple event attributes in the input but predicting only the next activity, attributes that the model does not find relevant for the next activity might not be represented in r. Instead, the training objective should be adapted to account for the multivariate nature of event logs. Training objectives that focus on learning representations of the data that and consider the general priors of representation learning are infrequently researched for

event log data. Instead, most approaches focus on solving one specific task in a supervised way.

In conclusion, the multivariety and heterogeneity in event attributes, their connectivity, and the procedural nature of processes make building a model architecture a challenge in its own. The focus of learning an accurate and multi-purpose representation from event logs before solving a specific task, despite having huge success in other fields, has been researched very little. More research is required to develop architectures and accommodating and appropriate training methods to learn representations r from event log data.

1.3.2 *PRM* Assessment (Intrinsic)

The second research area deals with evaluating the quality of the learned representation before any application. Given the black-box nature of neural network models, the internal representation r is difficult to inspect by looking at it — different from a model discovered with process discovery techniques. A good representation makes solving subsequent tasks, sometimes also called downstream tasks, easier (Bengio et al. 2013). Since evaluations on downstream tasks are time-consuming and often require additional procedures after representation learning, alternatives that are easy and (computationally) cheap to implement are required. Such an evaluation is often referred to as intrinsic evaluation (Jurafsky and Martin 2025, p.38) For instance, solving tasks that work in the embedding space of the representations or the analysis of the pre-training tasks to understand the model's performance, independent of any application. They also aim to answer how well a model has learned the data concerning the priors.

Previous research on representation learning has found that the representations learned within neural networks can have interesting properties (Goodfellow et al. 2016, p.526 ff.). Since this representation is a feature vector, clustering techniques or event arithmetic operations can be applied. For instance, it has been found that the internal representation of words in neural network models, often called word embeddings, depicts semantic similarities when being trained in predicting missing words in sentences in a self-supervised way (Mikolov et al. 2013). Further, arithmetic operations can be performed on such word representations that resemble surprising semantic logic. Also, for other domains such as image data, learned representations have been found to resemble semantic clusters (Xie et al. 2016). For event log data, such techniques can support semantic tasks on processes such as clustering of process models (Koninck et al. 2018; Seeliger et al. 2021).

Besides inspecting the representation r itself, one can assess the *PRM* by examining its performance on the pre-training task. For event log data, predicting characteristics of the next event is often employed. Predicting the next word in a sentence is considered a self-supervised pre-training task in language modeling. For BPM, predicting the next activity in a running process instance has been conceptualized as a downstream task, as it helps process owners anticipate future process behavior and act accordingly (Ruta and Majeed 2011). Both tasks, predicting next words

in sentences or activities in traces, do not require expert labels and can thus be classified as an intrinsic evaluation. Therefore, we can also assess a *PRM* based on its performance in this task.

However, certain characteristics of processes, such as parallel behavior, can threaten reliable evaluations and comparisons of next element prediction models on event logs. For instance, if two activities can be executed in any order, the ground truth label used to compute accuracy might not be as deterministic as assumed, negatively impacting the performance measure. Further, it also impacts core ML objectives, such as generalization. Language models trained in predicting next words are assessed with evaluation metrics based on cross-entropy. However, the applicability of these metrics has not been researched for process prediction so far. Therefore, more research is required on how *PRMs* can be assessed without expert-labeled data, how reliable the currently available evaluation procedures are, what influences this has on other machine learning objectives, and what alternatives exist.

1.3.3 *PRM* Application and Demonstration (Extrinsic)

Although learning representation from data is an objective on its own, the final aim is to make subsequent tasks easier (Bengio et al. 2013). Therefore, representation models have to be tested on various downstream tasks after inspecting and evaluating them intrinsically to obtain a final assessment of their applicability. Their performance must be compared to existing techniques to assess their usefulness. Thus, research area 3 deals with the applicability of *PRMs* to the field of BPM.

BPM consists of an extensive range of tasks, from which many can already be supported by machine learning techniques (Weinzierl et al. 2024). As a representation model is of predictive nature, we mainly focus on predictive tasks. However, analytical tasks are also of interest — for instance, the analysis of historical traces for anomalies or compliance issues. Another interesting analytical task is abstracting events to higher-level activities, i.e., task abstraction for a more straightforward interpretation of actions in event logs.

Existing neural network models for machine learning tasks are often explicitly developed for one task and evaluated on that task only. For instance, a wide range of process prediction techniques solve the next step prediction task. However, most of these approaches have only been applied to the next step prediction task. At the same time, it has not been researched whether the learned representation also helps for other tasks like outcome prediction. This is surprising since only a few changes are required, and most characteristics are shared between those tasks.

Consequently, it is unclear whether machine learning models trained on event logs can be applied or help to solve multiple prediction tasks - with or without learning representations. Thus, more research has to be conducted for what tasks process representation models can be used, whether and how they can be applied to solve multiple tasks, and how well a representation model performs compared to existing, specialized models.

1.4 Research Questions

After outlining the research areas and their challenges, appropriate research questions (RQ) can be formulated. The research questions are aligned with the research areas. Each contribution to the research areas will therefore contribute answers to the research questions. First, the overall research question is presented before outlining the separate research questions for each research area.

Overall RQ: How can we realize process representation models *PRMs* that learn accurate representations from event logs for solving various BPM tasks?

Three research questions, one for each area, have been formulated.

RQ 1 How do *PRMs* need to be designed in terms of architecture and training procedure to learn accurate representations of concepts found in event log data?

RQ 2 How can *PRMs* and the representations be evaluated intrinsically?

RQ 3 For which BPM applications can *PRMs* be used, and how do they perform?

RQ 1 This research question is to be answered with contributions in RA 1, i.e., *PRM* architectures and training methods. Different architectures and training methods have to be developed and tested on event log data. On the architecture side, this includes existing ones, like RNNs, and newly developed architectures tailored for event log data. Similarly, existing machine learning training methods, e.g., those used for language representation learning, are employed and adapted for this type of data. Contributions are expected to answer which architecture and training method are best suited for learning representations r from event log data, thus guiding the design of a sophisticated *PRM*.

RQ 2 Contributions to RA 2 are expected to answer how the learned representations can be evaluated intrinsically. This includes techniques and approaches that allow assessing the quality of the representation r and the *PRM* without application context, e.g., by clustering the learned representations. Furthermore, self-supervised tasks and the interpretation of their results are critically examined. New insights on how to inspect r and interpret the performance of the *PRM* in self-supervised tasks are expected to be made.

RQ 3 The final research question is supposed to give insights about the applicability and performance of *PRMs* in application through contributions on RA 3. This involves testing the feasibility of using representations r for different tasks. Furthermore, techniques and approaches on using r for different tasks have to be researched, e.g., through fine-tuning or clustering. Finally, it is to be researched whether *PRMs* perform better than specialized approaches on those tasks. The results are expected to validate the applicability of the idea of *PRMs*.

1.5 Research Method

This thesis has been conducted in the field of Business Informatics[7] following the popular design science research (DSR) method. Design science focuses on the creation of artifacts that are discipline-oriented (Hevner et al. 2004) and the study of these artifacts in context (Wieringa 2014, p.3) — for instance, by creating and assessing artifacts such as novel methods, techniques, or algorithms used in software systems for more efficient data transmission (Wieringa 2014, p.3). The artifacts created aim to improve "something" in context, so understanding the context is important in DSR (Wieringa 2014). Finally, the results must be communicated to an appropriate audience, including technicians and practitioners (Hevner et al. 2004).

The research in this thesis focuses on business process representation learning, using ML techniques, with a practical impact on the domain of BPM. It is driven by the curiosity about whether and how the ideas of representation learning, which have been shown to be an enabler for many astonishing results in ML, can also work for data generated by business processes. Thus, the research is objective-driven in building such representation models. To make progress on these points, research questions have been defined (see Section 1.4). Throughout this research, several artifacts have been designed (see Table 3), and knowledge has been gained about the performance of these artifacts in practice. In sum, they aim to understand whether and how representation learning models for event logs can be built and how they should be designed. As many techniques come from the ML domain, contributions are also made in this discipline, e.g., how to learn from and evaluate on data representing real-world business processes.

Different versions of DSR exist that slightly differ in the way research is conducted. This thesis uses the design science research for information systems and software engineering approach by Wieringa (2014), featuring an iterative research approach with design and empirical cycles. Design cycles solve design problems and iterate over problem investigation, treatment design, and treatment validation. In contrast, empirical cycles answer knowledge questions about the artifact in context (Wieringa 2014), e.g., through testing or case studies (Wieringa 2014, p.110). As the research on *PRMs* in this thesis is conducted iteratively, with design problems and knowledge questions to be answered, the DSR method of Wieringa (2014) fits better than other sequential methods, e.g., Peffers et al. (2007). For instance, for developing a *PRM*, different techniques, algorithms, and models have to be designed and iteratively refined. Further, knowledge of how these designs perform in practice is collected to validate the feasibility of these choices. By iterating over these activities, the design problem of designing a *PRM* can be answered.

We start by positioning the research conducted in this thesis within the DSR framework presented in Wieringa (2014), which is an extended version of the framework presented by Hevner et al. (2004) (see, e.g., (Wieringa 2014, p.6)). This framework

[7] English translation of "Wirtschaftsinformatik" also known as *Business & Information Systems Engineering* (BISE).

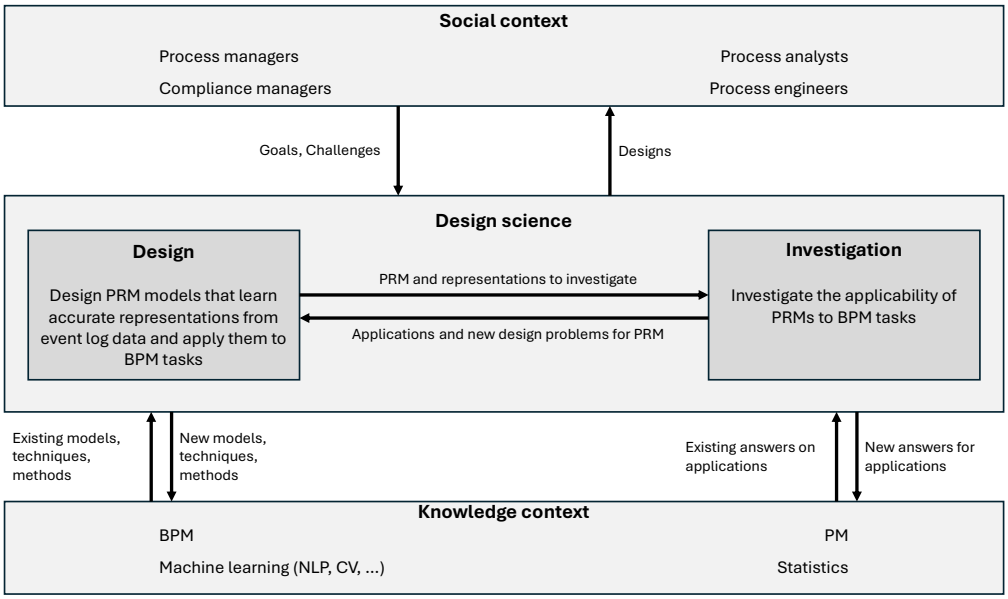


Figure 2: Positioning of this thesis within the DSR framework of Wieringa (2014) (own figure).

contains the stakeholders of the artifact and the knowledge used to design the artifact, extending the context of the design science project beyond the artifact itself. Figure 2 shows the framework applied to the research in this thesis, with a condensed version of design and investigation stemming from Section 1.3 and Section 1.4.

The stakeholders of this research are the users of *PRMs*, i.e., individuals who use ML techniques to analyze event log data. This includes process managers, data scientists, compliance managers, and other individuals who benefit from accurate process predictions and analysis. It also includes engineers who aim to develop new predictive or analytical solutions for event log data, as they can benefit from existing models that they can use to realize their applications. The context is the field of process analysis in BPM, where insights and knowledge about processes in organizations are to be drawn. Existing knowledge to solve design problems and answer knowledge questions is used from BPM and external domains. As this thesis is also positioned in the area of artificial intelligence, it uses existing knowledge from ML, statistics, NLP, and computer vision. Gained insights throughout this thesis add knowledge to these domains. For instance, new designs of neural network architectures for event log data and knowledge of whether such models can be used to solve various tasks.

The research questions introduced in Section 1.4 can now be classified according to the DSR method (Wieringa 2014). The overall research question can be classified as

a design problem. There are many solutions to build a *PRM*, which are evaluated by the utility in terms of stakeholder goals. Following the goal hierarchy of DSR projects (Wieringa 2014, p.14), the design of a *PRM* that generally improves the performance and efficiency of event log analysis constitutes a higher-level research goal, also referred to as an artifact design goal. Underlying this artifact design goal, many knowledge goals or instrumental design goals can be answered (Wieringa 2014, p.14). Examples for such questions include how *PRMs* interact in their context or how to simulate *PRMs* in context. For exploratory research projects like this, the hierarchy of these goals can be fuzzy, and some goals may be speculative — for instance, whether *PRMs* can reach a similar utility level compared to language models such as BERT. In the following, we will describe the three research questions as design problems or knowledge questions, and what type of contributions will be expected while answering these questions.

RQ 1 asks for designs of *PRMs* in terms of architecture and training methods that improve their capability to learn from event log data. This constitutes a design problem, as the solutions to this problem are design options and choices for *PRMs*. Further, there is not a single solution but potentially many solutions that have to be evaluated based on their utility.

The second research question contains design problems and knowledge questions. For instance, designing evaluation methods for assessing the accurateness in representing the process, which does not require expert labels, constitutes a design problem. On the other side, the question whether the commonly used metrics to evaluate the performance of process prediction models are reliable qualifies as a knowledge question, according to Wieringa (2014). It asks for knowledge about the metrics without improving them. Designing an alternative assessment procedure that overcomes these problems is a design problem.

Finally, RQ 3 investigates how the artifact performs in context. This involves designing methods for solving known BPM and PM tasks with *PRMs*. It also involves knowledge questions that answer how *PRMs* will perform in these applications and whether they perform better or worse than specialized approaches. Answers to these knowledge questions trigger new design problems for *PRMs*, e.g., if additional features need to be included to solve specific tasks. Contributions to research question 3 will allow the evaluation of the overall artifact design goal.

This research has been organized in cycles, not necessarily following a single, strictly sequential structure (Wieringa 2014, p.32). In fact, design cycle(s) (used to design and investigate artifacts) and empirical cycle(s) (to answer knowledge questions) were performed continuously and often in parallel. For instance, designing a new architecture and training method for event log data (targeting RQ1) and its evaluation, i.e., whether this *PRM* performs better than existing ones (targeting RQ3), were conducted in conjunction. During these cycles, new insights on how the model organizes the internal representation were obtained, which initiated, for instance, a new design cycle targeting RQ2. By doing so, fruitful interactions between design, evaluation, and answers to knowledge questions could be used most effectively.

1.6 Publications

This thesis consists of a series of 12 peer-reviewed and published research papers. Nine papers have been published at conferences such as the *International Conference on Business Process Management (BPM)* or the *International Conference on Process Mining (ICPM)*, as well as the workshops associated with those conferences. Three manuscripts have been published in scientific journals related to BPM. One in the *Enterprise Modelling and Information Systems Architectures – International Journal of Conceptual Modeling (EMISAJ)*. The second journal paper — published in *Information Systems* — is an invited journal extension of the best papers at *ICPM 2023* (paper number 8) that got significantly extended. Paper number 12 — published in a special issue of the *Business & Information Systems Engineering (BISE)* journal — is a significantly extended version of paper number 9.

Two of the papers are single-author papers. One has been published for the doctoral consortium at the *BPM 2022*, while the other was accepted as a student paper on the *AAAI Bridge Program: Artificial Intelligence and Business Process Management* in 2023, where it was presented as a poster. Part II will list other papers published during the PhD that did not contribute to the research areas. Table 2 gives an overview of the publications that contribute to this thesis and the research area they contribute to.

Table 2: Contributing publications by publication year

#	Ref	Article/Outlet	RA
	2020		
P1	Baier, Dunzer, Fettke, Houy, Matzner, Pfeiffer, Rehse, Scheid, Stephan and Stierle (2020)	**The MobIS-Challenge 2019 – A Report on the WI-2019-Workshop on Model-Based Compliance in Information Systems** — *Enterprise Modelling and Information Systems Architectures — International Journal of Conceptual Modeling (EMISAJ)*	1, 2, 3
	2021		
P2	Pfeiffer, Lahann and Fettke (2021)	**Multivariate Business Process Representation Learning Utilizing Gramian Angular Fields and Convolutional Neural Networks** — *International Conference on Business Process Management (BPM) 2021* (LNCS)	1, 2, 3

#	Ref	Article/Outlet	RA
	2022		
P3	Pfeiffer (2022)	**Business Process Representation Learning** — *International Conference on Business Process Management (BPM) 2022 – Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track* (CEURS)	1
	2023		
P4	Pfeiffer, Lahann and Fettke (2023)	**The Label Ambiguity Problem in Process Prediction** — *International Conference on Business Process Management (BPM) 2022 – AI4BPM Workshop* (LNBIP)	2
P5	Lahann, Pfeiffer and Fettke (2023)	**LSTM-based Anomaly Detection of Process Instances: Benchmark and Tweaks** — *International Conference on Process Mining (ICPM) 2022 – ML4BPM Workshop* (LNBIP)	1, 3
P6	Rebmann, Pfeiffer, Fettke and van der Aa (2023)	**Multi-perspective Identification of Event Groups for Event Abstraction** — *International Conference on Process Mining (ICPM) 2022 – EDBA Workshop* (LNBIP)	1, 3
P7	Pfeiffer (2023)	**Towards Process Representation Models for Business Process Management** — *AAAI Bridge Program: Artificial Intelligence and Business Process Management (AI4BPM)*	1
P8	Grohs, Pfeiffer and Rehse (2023)	**Business Process Deviation Prediction** — *International Conference on Process Mining (ICPM) 2023* (IEEE)	1, 3
	2024		
P9	Abb, Pfeiffer, Fettke and Rehse (2024)	**A Discussion on Generalization in Next-Activity Prediction** — *International Conference on Business Process Management (BPM) 2023 – AI4BPM Workshop* (LNBIP)	2
P10	Pfeiffer and Fettke (2024)	**Trace vs. Time: Entropy Analysis and Event Predictability of Traceless Event Sequencing** — *International Conference on Business Process Management (BPM) 2024 – Forum* (LNBIP)	1, 2, 3

#	Ref	Article/Outlet	RA
	2025		
P11	Grohs, Pfeiffer and Rehse (2025)	**Proactive Conformance Checking: An Approach for Predicting Deviations in Business Processes** — *Information Systems (IS)*	1, 3
P12	Pfeiffer, Abb, Fettke and Rehse (2025)	**Learning from the Data to Predict the Process: Generalization Capabilities of Next Activity Prediction Algorithms** — *Business & Information Systems Engineering (BISE)*	2, 3

1.7 Contributions Overview

As indicated by the previous Table 2, the contributing publication often did not contribute to a single research area. Instead, they mostly communicated results for more than one. This has two primary reasons. First, due to the research method, where several cycles were executed simultaneously, multiple results were available at the same time. Further, results from multiple cycles were combined to form a proper scientific publication. For instance, publishing a new architecture and training method requires an extensive evaluation of its quality and positioning against the state of the art. Publishing the model without experimental analysis, i.e., without assessing its performance in the expected environment, is insufficient.

To give a better overview of how the publications are aligned with research questions, Table 3 lists the contributions of each publication to the research area and questions. The following chapters will discuss all contributions in detail per research area.

Table 3: Overview of publications and their contributions to the research questions. The section discussion the contribution is mentioned in brackets.

#	**RA1**	**RA2**	**RA3**
P1	**Context-aware autoencoder** (AE) for event logs (2.2)	Clustering of the **latent space** (3.2.1)	Application of the **reconstruction error** for **compliance checking** on the MobIS challenge (4.3.1)
P2	**Multi-Perspective Process Network** (**MPPN**) for event logs predicting **all attributes in the next event** (2.3)	**Semantic trace clustering** based on representations r (3.2.2)	Application and benchmark of the MPPN in **process prediction**: (4.2.1) **Next step prediction** (activity, resource, duration) **Outcome prediction** (activity, resource, duration)
P3	Conceptualization of ***PRMs*** (2.4)	—	—
P4	—	Description and discussion of the **label ambiguity problem** in process prediction and its influence on performance assessment (3.3.1)	—
P5	—	—	Application and benchmark of the DAPNN on the **Process Discovery Contest** 2020 and 2021 (4.3.3) Application and benchmark of the DAPNN in **anomaly detection** (4.3.3)

#	RA1	RA2	RA3
P6	MPPN trained with **masked event prediction** (2.3.2.1)	—	Application of the MPPN for **contextual event abstraction** (4.3.4)
P7	Refined the concept of ***PRMs*** (2.4)	—	—
P8	Description of the **deviation prediction task** and its challenges for model training (2.3.2.2) MPPN as a **pre-trained model** for the business process deviation prediction task (2.3.2.2)	—	Application and benchmark of using the **pre-trained embeddings** of the MPPN for deviation prediction (4.2.2.1) Application and benchmark of the predicted **suffixes** of the MPPN for deviation prediction (4.2.2.3)
P9	—	Discussion and analysis of the **accuracy limit** in process prediction due to label ambiguity (3.3.2) Critical discussion of **generalization** and its evaluation for process prediction tasks (3.3.2)	—
P10	Extension of *PRMs* to learn from and make predictions on the **original event sequence generated by** S instead of the traces in L (2.5)	**Cross-entropy and its link to entropy** as a alternative to accuracy for the evaluation of *PRMs* (3.4)	Comparison of three next step prediction models in predicting the **next activity in** S or in the traces in L (4.4)
P11	Extension of deviation prediction to **deviation pattern prediction** (2.3.2.2) **MPPN** tailored for **deviation** and **deviation pattern** prediction (2.3.2.2)	—	Application and benchmark of the **MPPN end-to-end for deviation prediction** (4.2.2) Application and benchmark of the **MPPN end-to-end for deviation pattern prediction** (4.2.2)
P12	—	Conceptualization of **generalization** for next activity prediction based on ML (3.3.3)	**Systematical evaluation of generalization** capabilities of different next activity prediction models (LSTM, MPPN) (4.2.1)

Chapter 2

Research Area 1 - Architectures and Training Procedures of Process Representation Models

2.1 Overview

This chapter contains the contributions to RA 1 by introducing network architectures and accompanying training methods for *PRMs*. We present an LSTM-based autoencoder-style model called S2SA that learns a compact representation of traces containing various event attributes. This approach has been developed to classify traces as compliant or non-compliant based on the reconstruction error in an unsupervised way. It has been presented as a contribution to the MobIS Challenge 2019 Baier et al. (2020) and extended in a seminar work.

Building on this approach, the Multi-Perspective Process Network (MPPN) was developed as an enhancement over the S2SA. Its architecture is specifically tailored to learn from any combination of event logs' categorical, numerical, and temporal attributes. The two-stage training procedure allows it to be used for various tasks. The original approach was presented at the BPM conference in 2021 (Pfeiffer et al. 2021) and has been used in following works for different tasks.

The concept and idea of representation learning on event log data have been described in Pfeiffer (2022) and extended towards process representation models in Pfeiffer (2023).

Finally, a new approach capable of learning inter-trace behavior from the system S is presented in Pfeiffer and Fettke (2024). In contrast to existing process prediction methods that are trace-centric and make predictions from and for a single trace only, the approach is able to recognize and learn effects between multiple traces, extending the idea of *PRMs*.

2.2 S2SA Model for the MobIS-Challenge[8]

The MobIS-Challenge 2019 This section describes the sequence-to-sequence trace autoencoder (S2SA) approach developed for the MobIS-Challenge 2019, presented at the accompanying Wirtschaftsinformatik conference (WI) 2019 workshop titled "MobIS-Challenge for Students and Doctoral Candidates: Model-Based Compliance in Information Systems". A description of the challenge has been added to the appendix in Chapter III. Following the workshop, a paper describing the challenge, the data, and a summary of the two submissions to the workshop has been published in the EMISA journal (Baier et al. 2020). This publication contains the submitted autoencoder approach in section 7. After the challenge, the S2SA approach was extended during a seminar work. This resulted in the case-to-case autoencoder C2CAE that was not published but is included in the appendix for completeness in Chapter III.

The S2SA aims to identify compliance issues unsupervised through the reconstruction error when generating the full trace from the representation r again. It remarks the first neural network trained in the PhD, where important experience was gained in applying neural networks for event log data.

For the challenge, a dataset in the form of an event log was released that describes the business travel management process of a medium-sized software consulting company, with 6,555 traces. It was generated by simulation of a real process, supported by an information system keeping track of all relevant steps. Compliance issues were added by manually altering certain events. The task of the challenge was to analyze the event log with process mining techniques and develop approaches to find and classify compliance issues. Apart from a short challenge description, which can be found in the appendix in section Chapter III, no additional information about the compliance issues, e.g., the number or types of issues, were communicated. Therefore, we analyzed the event log and found four compliance rules. The traces not complying with these four rules served as the hold-out test set for validation.

Sequence-to-Sequence Trace Autoencoder S2SA As only this description but no ground truth data about the compliance issues were given, the idea was to detect the compliance issues unsupervised using an autoencoder that works in a sequence-to-sequence way, similar to anomaly detection in text or time series. In general, a sequence-to-sequence autoencoder consists of an encoder and a decoder. The encoder transforms the input sequence into a low-dimensional latent representation from which the decoder tries to reconstruct the original sequence again. The training objective is to reconstruct the original input as closely as possible. Thereby, essential characteristics of the input data are supposed to be contained in the learned representation r.

[8] This section is based on Baier, Dunzer, Fettke, Houy, Matzner, Pfeiffer, Rehse, Scheid, Stephan and Stierle (2020).

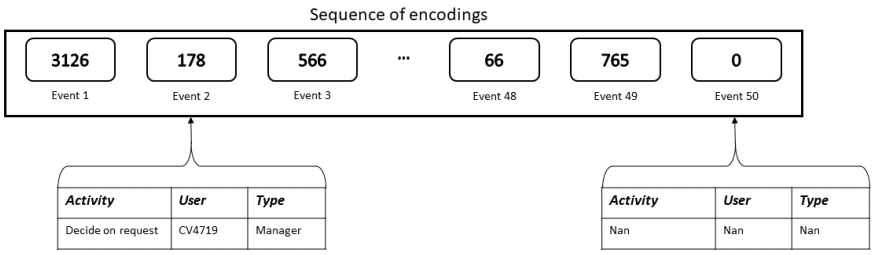


Figure 3: Encoding of events for the S2SA (taken from the research project in Chapter III).

As we can assume that the majority of traces in the event log describe compliant behavior, the AE is supposed to learn to reconstruct the traces in a compliant way. If a non-compliant trace is given as input, depicting an exceptional trace, the AE is supposed to make errors in reconstructing the non-compliant behavior in the trace from its latent representation. This will result in a higher reconstruction error than the reconstruction of compliant cases since the model is not trained well in reconstructing infrequent, non-compliant behavior. The reconstruction error is later used to identify non-compliant cases by classifying those cases as non-compliant that exceed a certain reconstruction error, requiring no manual analysis.

Following the case description, compliance issues manifest in various process perspectives. Checking and reconstructing the control flow alone does not suffice to detect all compliance issues. This requires the autoencoder to consider these attributes in the event log in its encoder as well as the decoder. To reduce the size of the neural network, all numerical attributes were dropped. Only the attributes *activity*, *resource*, and *type* (the role of the resource) found in the MobIS event log were used. In the event log, a total of 3,742 unique combinations of these attributes exist. Each unique combination was assigned a number that is used to encode the event. The autoencoder processes each trace $\langle e_0, e_1, \ldots, e_n \rangle$ as a single sequence of integer values $\langle i_0, i_1, \ldots, i_{50} \rangle$ where each integer i represents the unique *activity*, *resource*, and *type* combination of that event. All traces are padded with zero values to length 50. The encoding is visualized in Figure 3.

The architecture of the S2SA is shown in Figure 4. It consists of a learnable embedding layer that transforms the integer encoding into a high-dimensional vector. Following this embedding layer, an LSTM layer processes the sequence of encoded events. The output of the LSTM layer is passed through the bottleneck consisting of two fully connected layers. The last layer features two neurons, resulting in a representation r of each trace of size 2 (red layer). From this low-level representation, the decoder reconstructs the original sequence of events using several fully connected and one LSTM layer. The reconstruction error was computed as the mean squared error between the input and output sequences of integer-encoded events. It is used as loss during training as well as separating traces into compliant and non-compliant. The model was trained on 90% of the traces while 10% were left out to compute the validation loss to trigger early stopping.

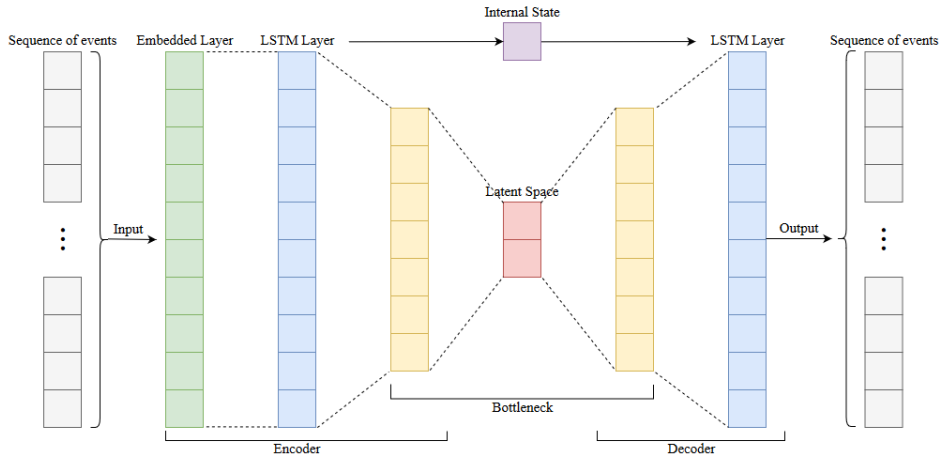


Figure 4: The architecture of the S2SA (Baier et al. 2020).

Results The S2SA model can learn a representation from whole traces with the attributes *activity*, *resource*, and *type*, making it a *PRM*. For this model, the design choice was made to encode each unique combination of these attributes into a single integer value instead of having separate models or layers per attribute. In result, the S2SA model is almost identical to other sequence-to-sequence autoencoder models, which have been shown to perform well for other sequential data. Further, the models' capacity in the number of neurons is smaller than having separate networks per attribute. This also allows the training of all layers with all data.

Training to reconstruct the original data from an encoded, latent representation has been shown to lead to very useful representations. During the time of that research, i.e., around 2018, autoencoders have been successfully used for learning representations of various data (Tschannen et al. 2018), showing that by training the AE model, important features of the input data are contained in the representation. Therefore, relevant information about the data has to be contained in the input. The S2SA model can only consider the three categorical event attributes but no numerical attributes, such as *cost*, or other important information, such as the date when the travel started or ended. However, this decision had to be made as the number of unique attribute value combinations otherwise would have increased drastically, also increasing the model size. The embedding layer becomes especially large if the number of unique values is increased. If the model capacity becomes too high in relation to the available data, the model might not be trained well or overfit, as too little data is available. In both situations, the reconstruction error will not be useful in separating compliant from non-compliant behavior.

However, even though design decisions to limit the model size have been implemented, the S2SA model did not decrease the reconstruction error during training as much as expected. This indicates that the model is not accurate in reconstructing the traces and, as a consequence, might not have learned normative, compliant

behavior from the data. Further analysis is shown in Subsection 3.2.1 and Subsection 4.3.1.

In conclusion, the model can process and learn a representation from traces. However, the model is limited in the attributes it can process, limiting the information in its representation r. Further, the reconstruction error does not decrease as much as expected, indicating that not enough data is available or the model capacity is too high. Further, by mapping the distinct values of the attributes *activity*, *resource*, and *type* to a single integer value, relations between those attributes might not be represented naturally.

Extension as Seminal Work The autoencoder approach for the MobIS-Challenge 2019 has been extended to the C2CAE in a seminal work using the same event log and task. This work has not been published but is added to the appendix Chapter III for completeness. The extended model does not contribute to the thesis. However, important findings have been made that influenced the design of the MPPN, which is described in the following section. Therefore, we will shortly explain the changes to the S2SA.

Instead of encoding and embedding all attributes into a single embedding, each attribute is encoded and embedded separately. Each attribute is handled by a separate sub-model, called "branch", both in the encoder and decoder, allowing them to adapt to the specific behavior of each attribute. For categorical attributes, the sub-models consist of an embedding layer and several linear layers. For numerical attributes, they consist of LSTM layers only. The outputs of the sub-models are concatenated in the central layers of the autoencoder to have a single representation r containing all attributes. While this architecture is more complex, it allows having information from all perspectives, including numerical and temporal ones, in the representation r, leading to a better performance in reconstructing traces. While the model performs better than the S2SA, the model's size is still very high in contrast to the event log size. Therefore, other architectures are required to handle the diversity of event logs.

2.3 Multi-Perspective Process Network (MPPN)[9]

The Multi-Perspective Process Network (MPPN) is an evolution over the S2SA and C2CAE models presented in the previous section, aimed at overcoming their limitations in architecture and training. It combines ideas from different ML areas that have been shown to perform well in representation learning, such as multi-view learning, multi-task training, gramian angular fields for time series learning, and the self-supervised pre-training paradigm. The major areas of improvement include:

1. It applies the multi-view learning paradigm (Ziv and Lecun 2024, p.3) to event logs by treating each perspective as an individual view of the process
2. Instead of embedding layers that can become huge and require large amounts of data to be trained properly, all event and trace attributes (categorical, numerical, and temporal) are first transformed to gramian angular fields (GAF) (Wang and Oates 2014), i.e., 2D matrices, from which the important features per perspective are extracted using a pre-trained CNN model
3. The MPPN is pre-trained in a self-supervised way by predicting the attribute values of the next event given a trace prefix through multi-task learning. This pre-training task trains the model to learn a representation that contains information from all perspectives of interest
4. After pre-training, the learned representations can be used for analytical tasks. Further, the model can be fine-tuned in a supervised way for various tasks by replacing the last layers with task-specific layers

As a result, the model is smaller in the number of neurons than the AE model and can be pre-trained on prefixes instead of traces, resulting in more available samples. Further, it scales well with more attributes as only a few layers increase by a constant factor per perspective added, which are much smaller than the size of an embedding layer. This allows learning from any combination and number of categorical, numerical, and temporal attributes found in the event log of interest. Predicting next elements from a prefix is supposed to be easier than reconstructing the whole traces from a latent representation. Besides learning representations for traces[10], it can also be trained to learn representations for events. Due to the separation in pre-training and fine-tuning, it can be applied to various tasks. This makes it one of the most sophisticated prediction models for event logs available.

The initial MPPN model is described in Pfeiffer et al. (2021), presented at BPM 2021 will be shown in the next subsection. Over the PhD, the model has been used extensively for different tasks, some of which required changes to the model architecture or training procedure. These include event abstraction as presented in Rebmann et al. (2023) and deviation (pattern) prediction in Grohs et al. (2023) and Grohs et al. (2025). Further, it is included as a state-of-the-art process prediction model in Abb et al. (2024) to evaluate generalization capabilities.

[9] This section is based on Pfeiffer, Lahann and Fettke (2021).
[10] See Subsection 3.2.2 for how to obtain representations for whole traces using the MPPN

2.3.1 The initial MPPN[11]

A large variety of different ML models exists for process prediction approaches. While such models have been shown to reach high accuracies on tasks like next activity prediction, outcome prediction, or remaining time prediction, most of these models are task-specific, i.e., they are trained and evaluated on a single task only. Furthermore, they are limited to a fixed subset of attributes found in the event log[12]. These characteristics limit the model's versatility as the representations they learn do not contain all the information found in the event log. Further, no existing model has been designed and evaluated on different tasks.

The MPPN has been developed to overcome these limitations, consisting of an adaptable architecture and a multi-task learning objective. In combination, they ensure that all information given to the model is contained in its representation r. As it can process categorical, temporal, and numerical attributes, the model is called multivariate. Figure 5 shows the data transformation and architecture of the MPPN model. The pipeline from the event log to a representation of the trace in the MPPN (top to bottom in Figure 5) consists of the following steps (for each trace):

1. **Input:** Each perspective of a trace is first transformed into a sequence of numerical values that can be displayed as a time series (interpreting the event index as time steps). The time series are transformed to GAFs (Wang and Oates 2014) that represent the correlations between attribute values over time as a 2D matrix, which can also be visualized

2. **Feature extraction:** A pre-trained CNN extracts features from each GAF, resulting in a feature vector per perspective

3. **Multi-view fusion:** The features from each perspective are combined in a pooling layer before a fully connected network learns features across perspectives, resulting in the representation r per trace (denoted as FV in Figure 5)

4. **Task heads:** A flexible number of task-specific layers (called heads) is used for pre-training and can be replaced afterwards to fine-tune the model for different tasks based on the learned representation

To process all information in the event log, each perspective of the event log, i.e., each sequence of trace attribute values, is treated as a view of the process. Each view is interpreted as a time series and transformed into a GAF. By interpreting the perspectives as time series and transforming them into 2D matrices, features from the event log can be extracted using a pre-trained CNN. Different procedures are used to transform the attribute value sequence into sequences of numerical values depending on the attribute type. Categorical attributes are encoded as integer values. Attributes that are timestamps are transformed to the time in seconds passed since the earliest timestamp in the log. Numerical attributes are used as they are. If an attribute is a trace attribute, it is duplicated to the trace length

[11] This subsection is based on Pfeiffer, Lahann and Fettke (2021).
[12] For reference, see table 1 in Pfeiffer et al. (2021).

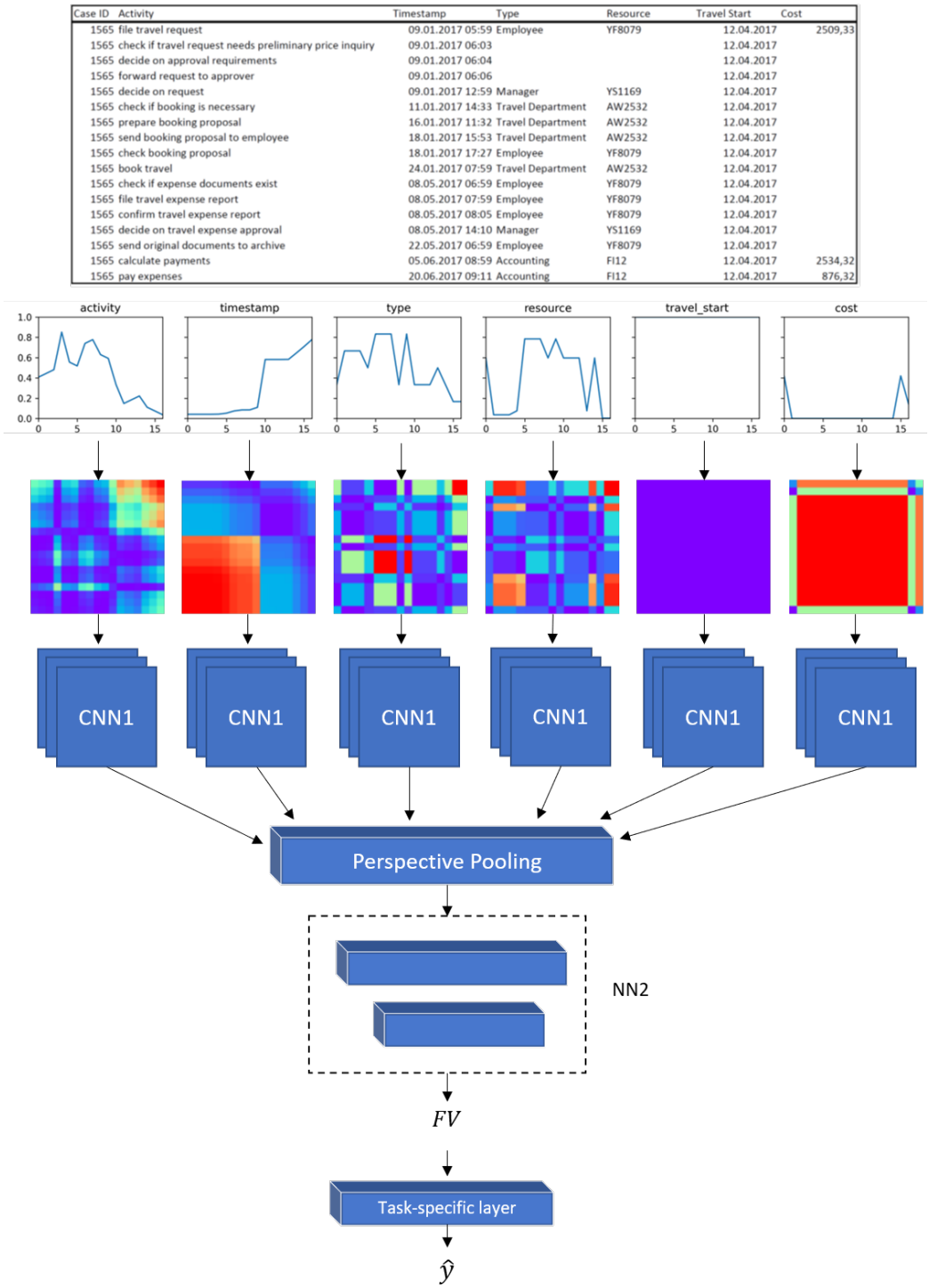


Figure 5: The architecture of the MPPN model, exemplified on a trace from the MobIS-challenge event log (Pfeiffer et al. 2021).

before being encoded in the same way as the event attributes. If an attribute is of a different type and cannot be interpreted as a categorical, temporal, or numerical perspective, a strategy for transformation has to be developed.

Once the perspectives have been transformed into equally length numerical sequences, they can be interpreted and visualized as time series where the index of the events is considered as the time steps (see Figure 5). This also allows applying the GAF transformation. Gramian angular fields have been proposed as a transformation of time series to inspect them visually with CNNs, generally improving the performance in time series tasks (Wang and Oates 2014). While humans can interpret time series plots easily, such plots are very scarce, i.e., most of the plot is empty, and only a fine line is drawn, containing little information for CNNs. In a GAF, the same information is represented as the trigonometric sum between each point in time. This representation also contains information about the temporal correlation within different time intervals, making it richer for ML interpretation. The GAF representation of each perspective is shown as a colorized image in Figure 5. For an event log with n attributes of interest, n GAFs are created. Each GAF represents the correlations in attribute values of one perspective in a trace.

As all perspectives are uniformly transformed into images, i.e., 2D matrices, a CNN can be applied to extract features. For this, a pre-trained Alexnet (Krizhevsky et al. 2012) is used for all GAF-encoded perspectives, i.e., only a single CNN is required. As GAF images differ from the natural images in ImageNet that Alexnet is trained on, Alexnet has been fine-tuned once on GAF images, resulting in a model called CNN1. This model is applied simultaneously on all n GAF-encoded perspectives, giving n feature vectors that are pooled in the perspective pooling layer into a single feature vector, denoted FV.

Using the pooled feature vector containing the combined information from all perspectives, a fully connected neural network, called NN2, learns dependencies and correlations between perspectives. Together with CNN1, they form the network that learns the representation r of a trace, containing information from all perspectives given as input. The last part of the MPPN is a flexible number of sub-networks called heads, each consisting of task-specific layers that perform classification or regression tasks. For instance, to train the MPPN in predicting the next activity given a trace prefix, a single head can be added using the learned representation to solve a classification task over activities. If the minutes until the process instance is finished are of interest, a single head performing regression can be added. It is also possible to use multiple heads where each head solves a different task, i.e., training them in a multi-task way.

To obtain rich representations, the MPPN is pre-trained once per event log, predicting the next event's attribute values for all attributes used in the input, i.e., the next step prediction task with all attributes simultaneously. For categorical attributes, a classification head is added. For numerical and temporal attributes, a regression head is added. The training loss to minimize is computed as the sum of the cross-entropy for categorical attributes and the mean squared error for the

regression heads. If n perspectives are given as input, the model solves a multi-task problem consisting of n views of the event log and n tasks. This ensures that all information contained in the trace is also included in the learned representation r.

After pre-training, the model can be used for next step prediction tasks without modification. For other tasks, the heads can be removed and other heads added. The remaining part of the MPPN stays the same and does not need any training. For instance, to perform outcome prediction instead of next step prediction. One can also remove all heads and use the learned representations as feature vectors for analysis, e.g., for clustering, retrieval, or anomaly detection. Apart from training the head(s) only, one can fine-tune the whole MPPN if the task requires it. This makes the MPPN a powerful model, given its flexibility in processing any combination of event attributes and versatility in solving tasks.

Results The design of the MPPN, in combination with the training objective, makes it a universal *PRM*. It can process and learn representations containing all categorical, temporal, and numerical event attributes as well as trace attributes of interest in the given event log. This remarks a significant improvement over the S2SA model and other existing prediction models for event logs. Once pre-trained on an event log, the learned representations can be used to solve various prediction tasks by training a smaller model.

During pre-training to predict the attribute values of the next event given a trace prefix, the model has been shown to be very precise. As the self-supervised pre-training objective for representation learning is also a downstream task in BPM, the model can be used to obtain multivariate representations and solve next-step prediction tasks. In one training run, the model can simultaneously predict all next attribute values, including frequently predicted attributes like *activity*, *resource*, and *timestamp*. Previous approaches had to train a dedicated model for predicting each attribute. In contrast, the MPPN can solve all tasks simultaneously with at least the same performance as specialized models[13].

While the architecture of the MPPN looks complex at first, it is streamlined towards event logs. CNN1, extracting features from GAF-encoded perspectives, is a single pre-trained model applied simultaneously to all perspectives. As it got pre-trained on GAF images once, its parameters require only very little, if any, optimization for a new event log. Thus, they do not need to be trained during representation learning as they are already pre-trained to extract features from GAF images. In contrast to models that utilize embedding layers or one-hot encodings for categorical and other transformations for different attribute types, the MPPN transforms all attribute types uniformly to GAFs and uses a single CNN for feature extraction. The other part of the model, i.e., the fully connected model NN2, is also comparably small as it consists of two layers only. On the MobIS-Challenge 2019 event log with attributes *activity*, *resource*, and *type*, the MPPN model consists of 36.248 neurons (without CNN1 as this is pre-trained) while the S2SA model consists of 91.521 neurons from

[13] For reference, see, e.g., table 4 in Pfeiffer et al. (2021).

which around 1/3 of them are used in the embedding layer. This means that the whole trainable part of the MPPN model is only a little bigger than the embedding layer of the S2SA.

If using more attributes, the size of the MPPN increases by a constant factor in NN2 (the size of the output of CNN1) and by the size of its heads (also a constant factor), scaling linearly with the additional information in the attributes. Using more attributes should increase the amount of information more than linearly due to dependencies and correlations between event attributes. In contrast, using embedding layers increases the size of a neural network more drastically.

On the downside, the model training is slowed down by the creation of GAFs. The longer the sequence, the bigger the GAF matrix size, and the longer the GAF computation takes. This drastically increases the overall training time. Further, changing the input sequence length requires pre-training the CNN1 for that length again, as the size of the GAFs changes, too. By transforming all attributes to GAFs, the semantic information of the data is lost, which reduces the applicability of the MPPN for certain tasks, e.g., anomaly detection (Rebmann 2024). The attributes are encoded independently from each other but uniformly to integer values. While the GAF transformation ensures that different values are encoded as different colors, the same color in different GAF-encoded perspectives has different meanings, as it represents information from different perspectives. This could potentially "blur" certain information as they cannot be distinguished. While the MPPN has been adapted to also learn event representations (see subsubsection 2.3.2.1), it cannot learn event and trace representations simultaneously, which would benefit learning hierarchical representations from event logs.

2.3.2 Modifications of the MPPN

In two subsequent publications, the MPPN has been adapted for specific tasks. The task of multi-perspective event abstraction, which has been tackled in one publication (Rebmann et al. 2023), called for representations of events instead of representations of traces. For this, the MPPN has been adapted to predict masked event attributes instead of next event attributes to obtain multi-perspective event embeddings. In Grohs et al. (2025), the MPPN has been tailored for deviation prediction, which introduces a number of new challenges that prediction models have to adapt to. The required changes made in both papers will be described below.

2.3.2.1 MPPN for Masked Event Prediction[14]

In this publication, the task of event abstraction has been tackled, i.e., grouping events recorded in event logs at a very low level into higher-level activities. Often, low-level activities are too fine-granular for manual analysis. By grouping them into high-level activities, more insights into the process can be obtained. Many existing approaches require domain knowledge for event abstraction, while the MPPN-based

[14] This subsection is based on Rebmann, Pfeiffer, Fettke and van der Aa (2023).

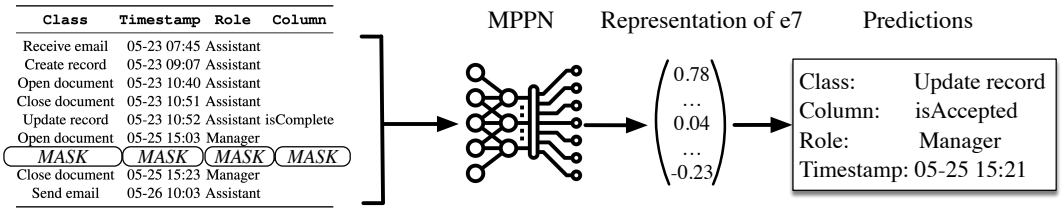


Figure 6: Masked event prediction task with MPPN (Rebmann et al. 2023).

approach works in an unsupervised way by clustering and grouping the event representations. Thus, in order to obtain meaningful results for abstraction, the event representations need to contain contextual information. For each event, its representations should contain multi-perspective information about the context of the event (i.e., about the other attribute values) as well as information about its context in terms of surrounding events in its trace. The representations are the base of a pipeline to discover and suggest event groups for higher-level activities.

Since the original MPPN model is trained to predict the next attribute values for a prefix, this results in representations for partial traces. For event abstraction, representations for events are required. Instead of being trained to predict next attribute values, the MPPN was modified to predict masked event attribute values in a given trace. This is similar to the masked word prediction task, also known as the cloze task, used in language modeling, e.g., in BERT (Devlin et al. 2019).

Figure 6 shows the masked event prediction task for the MPPN to obtain multi-perspective event representations. During training, for a given trace, all attribute values of a random event are masked by replacing them with a dedicated *MASK* value. The task of the MPPN is to predict the original attribute values that were masked, given the remaining events in the trace. Thereby, the learned representation r after NN2 contains information about the events' context.

If the MPPN is able to predict the attribute values accurately, it indicates that the representations r capture the events' context well. After training, representations for each event are generated by masking the particular event in the trace — the representation of the MPPN after NN2 is used as the representation of the event.

Results An event log has been simulated from a complex Petri net containing known mappings from the transitions to high-level activities. This event log is used to test the abstraction approach, which consists of the MPPN, generating event embeddings, and a pipeline to discover and suggest event groups. The MPPN performed exceptionally well by reaching almost 100% accuracy in predicting missing event attributes on the simulated event log. In the experiment, meaningful event abstractions were found, significantly simplifying the original process (see Subsection 4.3.4). Since the event representations form the base for the pipeline, they must contain contextual information. This shows that the MPPN can be used to obtain contextual event representations by modifying the training objective only.

2.3.2.2 MPPN for Deviation Prediction[15]

Business processes are subject to an increasing number of internal and external regulations. Knowing in advance whether running process instances are in danger of violating such regulations would allow process managers to proactively mitigate the risk and costs associated with it. Since deviations are rare, depend on contextual data in the prefix, and can form complex deviation patterns, the task is non-trivial to be solved with prediction methods. Two publications (Grohs et al. 2023, 2025) of this PhD focus on proactively mitigating the risk of deviations in a supervised way since the identification of deviations by definition depends on predefined to-be process documentations, e.g., to-be process models. In the first publication, the idea of deviation prediction and its challenges have been presented. This paper has been extended towards deviation pattern prediction, i.e., frequently co-occurring patterns of deviations.

Table 4: Example of deviation types and deviation pattern labeling. Labels of individual deviation types d_1 and d_2 as well as of deviation pattern c_1 for prefixes of t_1 (Grohs et al. 2025).

Trace Prefix	**Labels**		
	Individual Deviations		**Deviation Patterns**
	$\mathbf{d_1 : (\gg, APP)}$	$\mathbf{d_2 : (APP, \gg)}$	$\mathbf{c_1 : \{d_1, d_2\}}$
⟨SUB⟩	1	1	1
⟨SUB, PAR⟩	1	1	1
⟨SUB, PAR, PRE⟩	1	1	1
⟨SUB, PAR, PRE, ACC⟩	1	1	1
⟨SUB, PAR, PRE, ACC, FIN⟩	1	1	1
⟨SUB, PAR, PRE, ACC, FIN, REG⟩	0	1	0
⟨SUB, PAR, PRE, ACC, FIN, REG, APP⟩	0	0	0
⟨SUB, PAR, PRE, ACC, FIN, REG, APP, ACT⟩	0	0	0

SUB = A_SUBMITTED; PAR = A_PARTLY SUBMITTED;
PRE = A_PREACCEPTED; ACC = A_ACCEPTED; FIN = A_FINALIZED;
REG = A_REGISTERED; APP = A_APPROVED; ACT = A_ACTIVATED

From a machine learning perspective, the task is a dynamic multi-label classification problem: Given a trace prefix, the task is to predict which deviations and deviation patterns will occur in the future of that prefix. Since a trace can deviate in more than one way, there can be multiple deviations and deviation patterns to predict. Often, the context attributes of the trace prefix are decisive for deviations, which is why the prediction model must consider them. Further, deviation types and patterns are dynamic, meaning that if a deviation has occurred in the prefix and will no longer occur in its future, it should no longer be predicted. For a single trace, this can mean that all prefixes up to a certain position should be classified as deviating, while after that position they should no longer be classified as deviating. Table 4 shows a simplified example trace and its labeling, with two deviation types (log and model move on *APP*) that form one deviation pattern. As one can see, the trace is labeled differently per deviation type (indicated by *1* for deviating and *0* for non-deviating). A deviation pattern should be predicted as long as all of its deviation types are occurring in the future.

[15] This subsection is based on Grohs, Pfeiffer and Rehse (2025).

Finally, deviations occur infrequently since most traces constitute normative behavior. Not all deviations and deviation patterns are equally infrequent, which the model has to account for. There should not be too many false alarms handed over to a process manager by the prediction model to foster trust in the model. Thus, the prediction model has to balance precision and recall for a good *F1-score* while ensuring that all deviations are detected to minimize the risk associated with them.

In both publications, a specialized business process deviation prediction model (BPDP) was developed and tested, and the MPPN was used in different settings to benchmark the BPDP. Three different ways to use the MPPN have been tested that follow common machine learning paradigms.

1. The learned embeddings of the MPPN after pre-training have been used as input for another classification model that performs deviation prediction
2. The MPPN has been fine-tuned end-to-end for deviation and deviation pattern prediction by replacing the next-step prediction heads with heads that perform binary classification per deviation type. Each head is specialized in predicting one deviation type. Further, the event log was undersampled to account for the imbalance in training targets, and a deviation-specific loss was applied. The customized loss function puts more weight on the deviating classes than on non-deviating classes to force the model to detect non-conformance accurately
3. The MPPN has been used to predict whole suffixes for trace prefixes aligned to the to-be process model to detect deviations. As the suffixes reflect the expected future behavior, future deviations are detected by aligning them

Result In the conducted experiments, the adapted MPPN reached acceptable to good *F1-scores* in both individual deviation and deviation pattern prediction, compared to the result of the specialized BPDP model. This shows that the learned representations are a good base for deviation prediction by fine-tuning the original, pre-trained model. The features learned and included in the representation are useful for deviation prediction and make solving the task easier. Further, the MPPN accounts for the specific challenges of deviation prediction, which include a strong imbalance in the target labels, dynamic labeling of traces, and the multi-label nature of deviations. The performance of the MPPN on this task will be further explained and discussed in Subsection 4.2.2.

2.3.3 Summary of MPPN

The MPPN model has been used in different settings for various tasks by only slightly modifying its general design. This shows that the core of the MPPN model is versatile. Further, since the model has been successfully used for various tasks, it shows that the representations learned by the MPPN contain useful information that makes solving subsequent tasks easier, resembling the goal of representation learning (Bengio et al. 2013). This makes it a very versatile *PRM* and the first ML model tailored explicitly to learning representations from event log data.

2.4 Concept of Process Representation Models[16]

In the domain of BPM, predictive methods have been developed and used as task-specific models only. The idea of having an ML model that learns a representation useful to solve various tasks has been researched little. Given that process prediction approaches are frequently inspired by NLP methods, where representation learning has been an essential driver for progress (Bishop and Bishop 2024, p.189), it is surprising that the idea has not been described for event log data before. As part of this thesis, initial ideas for representation learning on event log data and its challenges have been formulated in Pfeiffer (2022). A following publication (Pfeiffer 2023) extended and updated the idea towards *PRMs*.

Both publications highlight the success of representation learning techniques used in models trained on large amounts of data for computer vision, language understanding, and other data domains. Since the data modality of an event log is very different from text or images, more research is required to investigate which existing techniques and concepts can be adapted and for which parts new ideas are required. Describing cases as a single sequence of homogeneous tokens, like sentences, misses the point that events consist of multiple attributes with different types. In comparison to time series, traces are not determined by time but by the behavior of the underlying process, e.g., the tasks to fulfill the process objective or other traces that run simultaneously. In contrast, adding temporal information in process prediction can be challenging. Thus, applying existing ML techniques to event logs can be challenging. If doing so, certain trade-offs in what information to use in neural networks and how to process it have to be made.

Results The results presented in the publications are conceptual, including conceptual descriptions for representation learning for event log data and process representation models, a concept of how to realize a *PRM*, as well as future work on datasets, tasks, and advanced methods. They form the base of the overall research goal of this thesis.

Representation learning on event logs aims to learn generic representations of the concepts found in the data, including events, cases, and processes. For this, new model architectures and training procedures are required to preserve the structure and semantics of the data. Other characteristics of event representations that could be useful include the positional information of the event, the surrounding context of the event in the trace, and its semantics. Case representations should aggregate the semantics of all events and other behavioral characteristics of the case. The representations model should also be able to generate representations for event logs not trained on. Since event logs can differ a lot, it has to be researched which features are transferable and which are specific to certain event logs. Finally, for certain tasks that require domain knowledge, an additional training phase with labeled data might be needed, e.g., to train in separating fitting from unfitting traces.

[16] This section is based on Pfeiffer (2022) and Pfeiffer (2023).

While the MPPN model learns and preserves many of these characteristics in the representation, it does not fulfill all requirements. For instance, the semantic information in events cannot be preserved appropriately when transforming attribute values into GAFs. This requires training the model for each event log and prevents it from learning generic features of processes. Also, it cannot learn representations for events and traces simultaneously.

A concept for an advanced *PRMs* is introduced to overcome these limitations. This model is inspired by the BERT language representation model (Devlin et al. 2019) but features a hierarchical architecture and training objective to learn event- and trace-level features simultaneously. Instead of GAF encoding, it encodes all event attributes to tokens and processes them with the transformer architecture. This preserves the semantic information contained in events. In addition to the positional encoding introduced with BERT (Devlin et al. 2019), an event encoding indicates which tokens belong to the same event. Additionally, special tokens similar to the CLS token in BERT are used for case- or process-level classification tasks. Such a model could be pre-trained on many event logs to learn generic characteristics of processes. Afterwards, the representations can be used in an unsupervised way, e.g., clustering or for event abstraction. For other tasks, the representations can be used to train a model for downstream tasks like process prediction. Also, few-shot learning is a promising method for learning transferable features across different event logs.

Such models' performance should be assessed on a large number of datasets and multiple tasks. For this, research is required on which other tasks in process mining can be solved with such techniques, which have so far not been solved. Further, evaluating the models intrinsically and extrinsically allows for a more detailed assessment of their capabilities. This allows assessing how good the learned representations are, separately from their performance on downstream tasks.

2.5 PRM learning beyond single Traces[17]

Process mining offers powerful techniques for analyzing real-world event data, stored as traces in event logs. Each trace describes the behavior of a process instance as a sequence of events, ordered as recorded by the information system. Real-world processes are performed by humans and machines, often in collaboration, to achieve a specific objective, forming a socio-technical system S. In such a system, there can be arbitrary dependencies between the actions carried out by the system participants, caused, e.g., by shared resources, bottlenecks, or deadlines. Some of these dependencies not only impact actions within a single trace but also influence all actions that happen in a certain period. Thus, they affect the behavior in different traces, as such dependencies can span across traces. Analyzing one trace at a time might conceal such trace-spanning behavior that can be useful for certain types of analysis, particularly for predictions.

Figure 7 shows the actions carried out over a certain period in the incident management process from the BPIC 2013 incidents event log. It shows seven traces (one line per trace), where each dot represents one event e. The events are ordered and spaced by their *timestamp* while different colors indicate different *activities*. One can find similar behavior when investigating the data per trace. Incidents start at some point, information gets collected, and the incident is investigated before being closed (red-colored events). Besides this trace behavior, we can also find additional behavior when looking beyond single traces. For instance, the red-colored events highlighted in the red boxes are executed successively throughout the traces within a short period. While Earl performs the first two occurrences, the later occurrence of this pattern originates from Juan. As it shows, there is "system" behavior across traces that remains unseen or is observed differently when analyzing events as traces.

However, analyzing event log data as traces aids process analysis since the behavior can be represented and inspected with a logical relationship induced by the *caseID*, e.g., product or request ID. For instance, a process model for the behavior seen in Figure 7 will likely be too complex to analyze by humans. However, for ML models, the additional information found across traces can be helpful in making predictions. Further, for learning accurate representations of event logs, the information on what is happening surrounding a single trace is important contextual information to include in the feature vector of the trace, similar to the event representations learned with MPPN in subsubsection 2.3.2.1. To find out how much information the original event sequence generated by S contains, and how much more difficult it is to learn in contrast to the trace-organized event log L, their entropy has been investigated, and the predictability has been tested.

Entropy, as defined by Shannon (1948), measures the level of information or surprise in random variables. For sequential data, where elements are correlated, it is measured as the entropy rate based on *Ngrams*, i.e., sequences of length n. The entropy rate quantifies the change in entropy if using one more element, i.e., if increasing the

[17] This section is based on Pfeiffer and Fettke (2024).

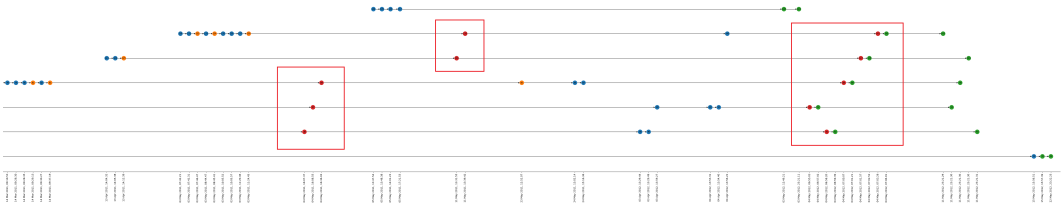


Figure 7: Some traces from the BPIC 2013 incidents event log with highlighted behavior spanning across traces (adapted from Pfeiffer and Fettke (2024)).

sequence length from n to $n+1$. The limit of the entropy rate, if it exists, quantifies the entropy of the underlying stochastic process that generated the sequences.

For sequences generated by S and stored in L, three different entropy rate estimators have been defined, which differ in how the sequences are built. We denote the original sequence of events as generated by S, which is only ordered by time and not by traces, as ES. TS denotes the sequence of events if first ordered by the *caseID* and then concatenating all traces in the event log L into a long sequence. If the traces are not concatenated into a long sequence but kept separate, the entropy rate estimator works on the traces in L.

All entropy rate estimators only consider the activities, i.e., the control flow of the process, as the event frequencies would otherwise tend to become very low, which negatively impacts their reliability. The entropy rate estimator H_n^{ES} estimates the entropy rate by building *Ngrams* of *activities* on the original sequence of events ES. H_n^{TS} estimates the entropy rate on TS with *Ngrams* potentially spanning from the end of one trace to the start of another trace. In contrast, H_n^L builds *Ngrams* from activities within traces in L, i.e., by respecting traces, only[18]. While all estimators use the same data, the resulting *Ngrams* and their frequencies differ as the events are ordered and *Ngrams* are constructed differently. With increasing n, the entropy rate estimators consider longer activity correlations. Lower entropies indicate less surprise, while higher entropies indicate more complex sequences.

There is a close link between the entropy estimation and the cross-entropy estimation used in many machine learning models that estimate the probability of next activities. Given a sequence with low entropy, estimating the next activity should be easier than estimating the next activity on a sequence with higher entropy. If a prediction model reaches a low cross-entropy in estimating next activities, it has accurately learned the statistical distribution of activities generated by S. Thus, we use the next activity prediction task to test how feasible it is to learn the behavior of S from traces in L or the original sequence of events ES.

In order to quantify the difference between learning process behavior from S based on traces or the original event sequence ES, three prediction types of the next activity prediction task have been formulated. They all estimate the probability $p(act_n \mid h)$ with act_n as the next activity and h as the history of previous activities.

[18] H_n^L is identical to an existing entropy rate estimator introduced in Back et al. (2019).

They all work on the same data, but differ in how the history of previous activity h is built and which activity act_n to predict.

- PPM_L: Models that perform this task predict the next activity in a trace given a prefix of activities from the same trace, which is the next step prediction task conducted in current literature. These models are called PPM_L as they perform on traces in the event log L. Here, act_n and h are drawn from the same trace, and no information from other traces is used.
- PPM_{TS}: This task is almost identical to PPM_L with the only difference that the windows can span multiple traces. The activities are still ordered as the traces in L. In conclusion, the history can contain activities from the previous trace, or act_n be from the following trace.
- PPM_{ES}: This is the most complex task where prefixes and next activities are samples from ES. This means there is no trace information as the events do not carry the *caseID* nor are the events ordered by trace as done in TS or L.

For all tasks, a window size of a maximum of 128 activities is used. The data is split into training and test sets with a ratio of 70/30 on a temporal basis. Three existing predictive approaches are compared. A random forest model with 100 trees has been chosen as a baseline. The LSTM-based model with 2 LSTM layers is selected as a state-of-the-art model. Further, a transformer-based model with four attention heads is trained to test whether any improvements can be made using this architecture. Note that this research focuses on a feasibility study of the different tasks rather than developing a dedicated architecture for this problem. If the task is found to be feasible, a dedicated architecture is worth developing.

Results After motivating and conceptualizing the idea of learning the behavior of S from its generated events, the entropy rates of different event orderings and sequencing techniques can be compared before comparing and testing the predictability. The entropy rates should indicate how feasible it is to make next activity predictions on ES compared to trace predictions using L. If the entropy of ES is too high or the rates do not reduce reliably, learning S will be very difficult.

Figure 8 compares the three entropy rate estimators on four real-life event logs. The lines are solid as long as the estimators are reliable, for which a constraint on the size of *Ngrams* has been implemented. The constraint is violated if there are insufficient statistics to compute reliable estimations. The estimations become unreliable at a certain point as the number of different *Ngrams* in comparison to their frequency increases disproportionately. Since the constraint is very strict, the lines are drawn dotted further.

The entropy rate estimators on ES and TS are more reliable for larger n than the existing entropy rate estimators based on traces in L. However, the latter one decreases quickly, proving that analyzing events ordered by traces is the least complex. For some event logs, especially BPIC 2013, the difference between H_n^{ES} and H_n^{TS} is not significant, while for others, the difference is considerably large.

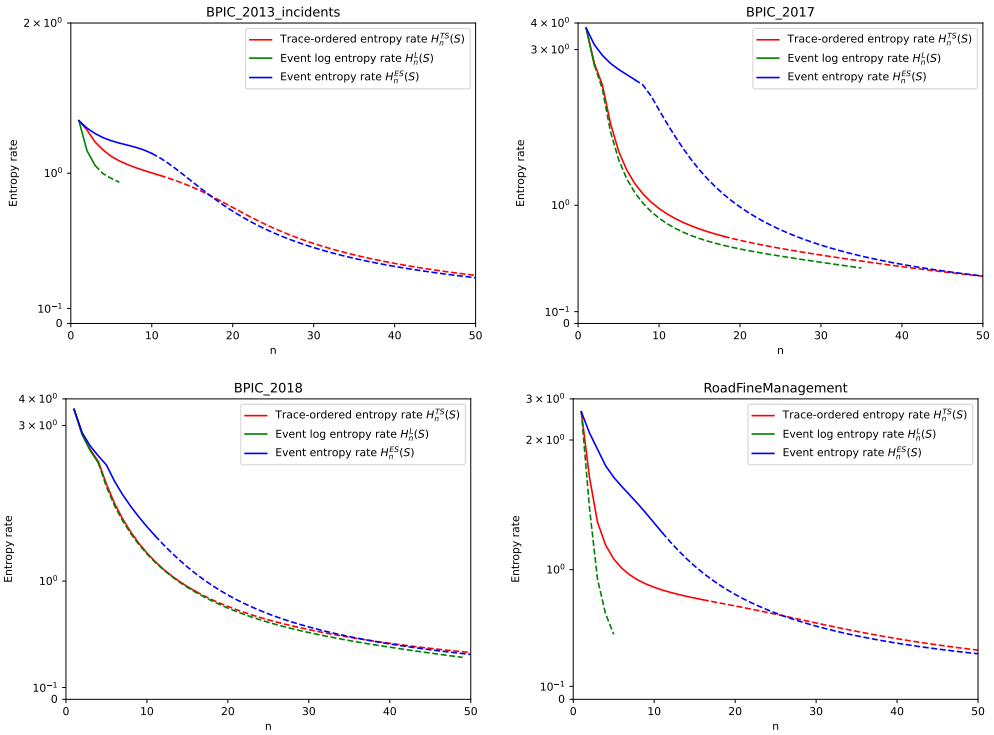


Figure 8: Comparison of different entropy rate estimators. The dotted lines indicate how the entropy rates continue after the constraint is violated (Pfeiffer and Fettke 2024).

However, considering that TS is ordered by the *caseID* while ES is not, another phenomenon must give ES a certain structure of repeating *Ngrams*. Although using the same data, the *Ngrams* built on ES contain different behavior from those built on TS, as the activities are ordered differently. We suspect, as motivated, that human behavior or behavior caused by other participants is the reason for the similar slopes of entropy rates and the little difference between the curves.

This also motivated testing the predictability of next activities with the three different prediction models. As seen in Table 5, all prediction models reach a lower cross-entropy than the unigram entropy rate, indicating that the models learn to estimate the next activities more accurately than statistical guessing. Analogous to the entropy rate results, making predictions on L gave the best results, i.e., the lowest cross entropy, followed by making predictions on TS and ES. The predictability results will be discussed in more detail in Section 3.4 and Section 4.4.

Although some limitations apply and the entropy rates are affected by undersampling, the results encourage future work on learning the system's behavior S that generated the event log from the original sequence instead of traces. We suspect that more behavior between traces, i.e., inter-trace behavior, leads to more similar entropy rate slopes. In contrast, if the traces are executed separately with less inter-trace behavior, the entropy rates might be somewhat dissimilar, with more significant differences between the slopes.

Table 5: Performance of different PPM models on the test split (Pfeiffer and Fettke 2024).

		PPM_{ES}			PPM_{TS}			PPM_{L}		
Cross-entropy (CE) in log_2 (lower is better)										
	H_1^{ES}	RF	LSTM	TF	RF	LSTM	TF	RF	LSTM	TF
BPIC 13 I	1.35	1.27	**1.07**	1.22	1.26	**1.04**	1.27	1.10	**0.97**	0.98
BPIC 17	3.79	2.56	**1.61**	1.63	0.75	0.49	**0.47**	0.61	**0.46**	0.47
BPIC 18	3.57	2.35	**2.11**	2.29	2.30	2.23	**2.13**	0.87	**0.57**	0.62
RFM	2.64	1.50	**1.30**	1.33	0.87	**0.64**	0.66	**0.67**	**0.67**	**0.67**
Accuracy (higher is better)										
BPIC 13 I		69.53%	**70.48%**	69.18%	64.94%	**67.53%**	61.48%	68.82%	**69.67%**	65.59%
BPIC 17		51.58%	**63.22%**	62.86%	86.15%	87.63%	**87.90%**	86.80%	**87.98%**	87.81%
BPIC 18		51.50%	**58.27%**	55.61%	59.76%	61.30%	**65.65%**	81.60%	**85.52%**	84.37%
RFM		65.05%	**66.81%**	65.57%	81.50%	**83.05%**	82.90%	80.91%	**80.92%**	**80.92%**

The event sequence ES is much less structured than TS or L and contains inter-case behavior by design. The prediction task on ES is much more complex than on the other two sequences, so the prediction models perform considerably well. Further, more complex neural architectures like the transformer seem more suitable for more complex and larger event logs like BPIC 2017 and BPIC 2018, while smaller models like LSTM perform better for simpler processes like BPIC 2013. Thus, the prediction models open up new research avenues for future work.

2.6 Summary and Conclusion

This chapter has summarized all results contributing on how to design and train process representation models. Starting with the autoencoder, a very early approach to learning a latent representation of traces containing information from multiple process perspectives, over the more advanced MPPN, to prediction approaches that take the whole behavior of the underlying system S into account when learning representations. Further, the general concept of process representation models has been discussed in two publications.

Regarding publications, the concept of *PRMs* has been presented (Pfeiffer 2022, 2023). Two unique neural network architectures for *PRMs* (Baier et al. 2020; Pfeiffer et al. 2021) and two adaptations of it have been presented (Rebmann et al. 2023; Grohs et al. 2023, 2025). Further, existing process prediction methods have been adapted to learn the behavior of S directly from its sequence instead of traces in L (Pfeiffer and Fettke 2024).

Adapting neural models for the characteristics of event log data and the challenges of BPM tasks is not straightforward but requires modifications that are not trivial. Sometimes, creative solutions combining different ideas of machine and representation learning, such as those used for MPPN, are needed. However, the adaptations have been shown to work well for the specifics of event log data, indicating that adapted architectures benefit more than standard ones.

For trace-centric predictions, the MPPN model performing next attribute values prediction has shown to be versatile and flexible in the attributes it can learn from, as well as accurate in its pre-training objective, indicating that important information from the different perspectives is included in the learned representation. After pre-training the model, representations r or the whole model can be utilized to solve various tasks, as shown in Chapter 4.

Future work should focus on *PRMs* that use information from other traces to take the contextual information of traces running concurrently into account, enriching the representation with the "state" of the system S. Such models should still be trace-centric, i.e., consider the events with relation to traces[19]. In terms of predictions, they should be able to make trace-centric predictions and more general predictions, e.g., on how the process will evolve in the future. Importantly, they should not be restricted to information from single traces only. Since representation learning aims to learn the underlying data-generating process and its causes (Goodfellow et al. 2016, p.524), learning from the original event sequence ES, containing trace information, should fit the aim better.

[19] This ability would mark an improvement over the prediction models presented in Pfeiffer and Fettke (2024) where the trace notion has been removed.

Chapter 3

Research Area 2 - Assessment of Process Representation Models

3.1 Overview

This chapter presents contributions on how to assess *PRMs* intrinsically. This includes the analysis of the representation space as well as the self-supervised pre-training task(s). These results contribute to RA 2.

The first subsection focuses on the representations learned by *PRMs*. First, the trace embeddings learned by the S2SA autoencoder, presented in Baier et al. (2020), are analyzed. Similarly, the trace representations of the MPPN model, presented in Pfeiffer et al. (2021), are analyzed.

The second subsection focuses on the self-supervised pre-training task(s) used to train *PRMs*. Predicting the next element in a sequence is a frequently employed self-supervised pre-training task. For process prediction, however, predicting the next activity to happen is considered a downstream task, too, as it allows anticipating future process behavior. This creates a discrepancy in evaluating and interpreting the model's performance, which has been analyzed in a series of publications. In Pfeiffer et al. (2023), the observation has been made that many prefixes have more than one valid continuation option, negatively affecting performance metrics such as accuracy. The implications of this phenomenon on the performance assessment are critically examined and discussed in Abb et al. (2024), especially on the intrinsic ability of the model to generalize. Finally, Pfeiffer et al. (2025) presents a conceptual frame for generalization in process prediction based on generalization in machine learning and shows empirical results on how well existing process prediction models can generalize in next activity prediction in various synthetic and real-world scenarios.

Lastly, Pfeiffer and Fettke (2024) presents the link between entropy on event logs and cross-entropy in next-step prediction as an alternative to accuracy that takes the stochastic nature of the data into account.

3.2 Analyzing the Internal Representation of *PRMs*

3.2.1 Trace Representations of the S2SA Autoencoder[20]

The S2SA approach, presented in Section 2.2, learns representations of traces during training. Constrained by the architecture, the representations contain information about the *activity*, *resource*, and *type* only, but no information about other attributes. Given that the event log contains many traces that are performed similarly, e.g., having the same control flow variant and resources, those traces should form clusters in the representation space.

The internal representation is analyzed for such clusters. This natural clustering, being one of the priors for representation learning (Bengio et al. 2013), would indicate that the model has learned the data in a meaningful way. After training, all traces are given to the autoencoder again. Since the bottleneck of the autoencoder consists of a layer with two neurons only, the representations are feature vectors of size two that can be plotted directly.

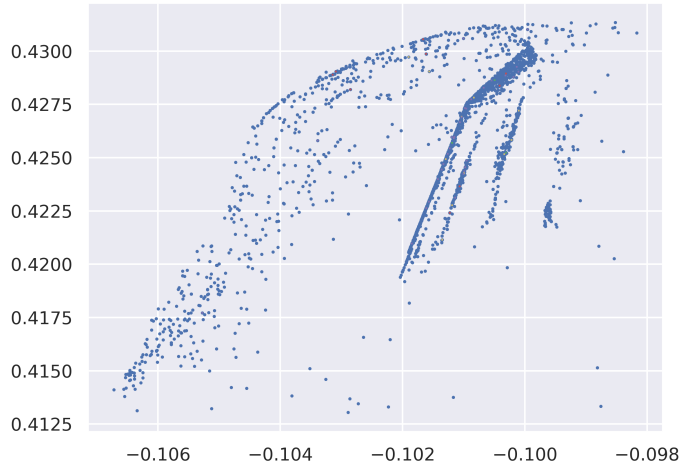


Figure 9: Internal representation space of the S2SA model (own figure).

Results Figure 9 shows the internal representations of traces from the bottleneck of the S2SA. Although some grouping of traces can be seen, there is no clustering as we would have expected. A lot of traces form a large group in the upper right. Apart from that, all other traces are scattered. Thus, natural clustering is not achieved. As noticed during training, the reconstruction error is not reduced sufficiently, which can explain why the embedding is not as organized as expected. This means that the reconstruction ability of the model is not very good.

[20] This subsection is based on Baier, Dunzer, Fettke, Houy, Matzner, Pfeiffer, Rehse, Scheid, Stephan and Stierle (2020).

3.2.2 Trace Representations of the MPPN[21]

The MPPN model, presented in Section 2.3, performed accurately in predicting next attribute values from prefixes. Thus, its internal representation might be more organized than that of the S2SA. For this analysis, the MobIS event log has been used with attributes *activity*, *resource*, *type*, *cost*, *timestamp*, and *travel_start*.

After pre-training the MPPN, the representations r for all traces were generated by passing complete traces to the MPPN. The representation is the output of NN2. Since this feature vector contains information from all attributes, its dimensionality is higher and thus has to be reduced for visualization. For this, the principal component analysis (PCA) has been applied to the set of representations, reducing the dimensionality of each feature vector linearly to two.

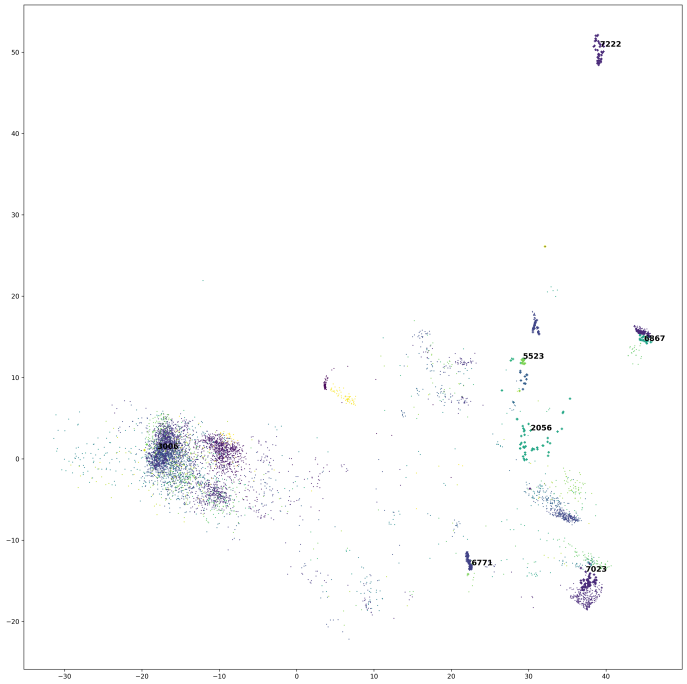


Figure 10: Internal representation space of the MPPN model (Pfeiffer et al. 2021).

Results Figure 10 shows the resulting representation space of the MPPN model. Different colors indicate different control flow variants. As one can see, various clusters are formed by the MPPN, making the representation space much more organized than the S2SA. Traces with the same control flow variant are often in the same clusters. However, some clusters have the same color but are apart, indicating that not only the control flow is decisive for clustering. The traces in these clusters have additional syntactical and semantical similarities, as will be demonstrated in Section 4.3. This proves that the MPPN is capable of natural clustering, indicating a major improvement over the S2SA model.

[21] This subsection is based on Pfeiffer, Lahann and Fettke (2021).

3.3 Analyzing the PRM Pre-Training Task[22]

Inspired by next word prediction models, process prediction models for event log data are frequently trained on the next step prediction task (Evermann et al. 2016). Initially and most commonly, the activity attribute of the next event is predicted while the performance is measured and compared based on accuracy, i.e., the share of samples where the predicted next activity is equal to the ground truth next activity.

In general, predicting next elements of an incomplete sequence is used likewise for many other data modalities. Most prominently for language models, building upon the seminal work of Shannon (1951) that analyzed the probabilities and entropy of *Ngrams* and next words in natural language. Such models, predicting next words (or tokens), are evaluated using cross-entropy and perplexity (Murphy 2022; Jurafsky and Martin 2025), which is computed on the model's learned probability distribution. Given that these measures are calculated on the data itself, they can indicate the quality of the model with respect to the data, i.e., intrinsically. However, the usefulness of the language model is evaluated by solving downstream tasks, i.e., extrinsically. The cross-entropy in next-word prediction is a quick and cheap measure for the quality of the model that sometimes serves as an approximation to the model's performance on downstream tasks (Jurafsky and Martin 2025).

In process prediction, however, the next step prediction task is commonly understood to enable proactive situation awareness by predicting the future of ongoing process instances (Ruta and Majeed 2011). Few other works consider the next activity prediction task as a pre-training or proxy task for solving downstream tasks, e.g., Nolle et al. (2016) for anomaly detection or Koninck et al. (2018) for learning embeddings of activities and traces. Thus, the task is commonly evaluated with accuracy while different models are compared in that way (Neu et al. 2021).

With the use of deep neural networks in process prediction, performance in terms of accuracy has increased over model-based or traditional ML-based methods (Pfeiffer et al. 2023). However, the performance of process prediction models has saturated soon, and no significant increase in the commonly used event logs has been observed for the last years (Abb et al. 2024). This resulted in a huge number of publications, where most models achieved very similar accuracy. Only minimal differences could be noted, with no model reaching an outstanding performance. Since more sophisticated network architecture and training procedures usually lead to better performance on ML tasks, stagnating performance is unusual.

These observations led to skepticism about the next step prediction task, motivating a series of three papers that critically analyzed the next activity prediction task in detail. The first two papers investigate the conceptual issues with the next step prediction task (Pfeiffer et al. 2023; Abb et al. 2024). The third paper (Pfeiffer et al. 2025) finally conceptualizes generalization in next activity prediction and evaluates this capability in existing models.

[22] This section is based on Pfeiffer, Lahann and Fettke (2023); Abb, Pfeiffer, Fettke and Rehse (2024); and Pfeiffer, Abb, Fettke and Rehse (2025).

3.3.1 Label Ambiguity in Process Prediction[23]

The first paper in that series introduced and quantified the label ambiguity problem in process prediction. Process prediction models are trained on a multitude of traces. Each trace is one example of how the process can behave. Thus, the next activity found for one prefix of one trace might not be the only acceptable way for the process to continue. There can be different options for how this prefix can proceed, caused by parallel paths or loops in the process. Consequently, there can be more than one ground truth label for the same prefix.

The term label ambiguity describes the uncertainty of ground truth labels in predictive tasks (Gao et al. 2017), initially introduced for image segmentation problems where different experts can label the sample differently. For process prediction tasks, the term label ambiguity has been defined referring to the uncertainty of the ground truth label derived from the event log due to typical characteristics of business processes. It affects the theoretical optimal performance a prediction model can achieve and the interpretation of its performance, preventing an objective estimation of how accurate such models are.

Formally, process prediction models are trained on samples (x, y) of prefixes x and targets y. x is a sequence of previous events $\langle e_1, \ldots, e_n \rangle$ while y, for next-activity prediction, is the *activity* attribute of the next event e_{n+1}. The target y thus serves as the ground truth for the next step given the input x. During training and evaluation, x is given to the prediction model and the prediction $\hat{y}$ is compared to the ground truth label y. Based on this comparison, performance metrics like accuracy or precision can be computed.

Table 6: Prefix $\langle A, B, C \rangle$ with three different options found in an event log, how it can continue (Pfeiffer et al. 2023).

Prefix	⟨A, B, C⟩		
Sequences	⟨A, B, C, D, E⟩	⟨A, B, C, L, M, K⟩	⟨A, B, C, M, L, K⟩

Table 6 shows a prefix and three different continuation options. Such a situation can occur if there is a decision point after activity C with concurrency between activities L and M. Consequently, there are three valid next steps for the same prefix, but each sample (x, y) contains only one instance. Such samples are called *ambiguity candidates*.

Besides the activities, real-life event logs contain contextual attributes that might resolve the ambiguity and clarify the next step. For instance, that activity D follows if costs are higher than 500. To determine how many *ambiguity candidates* can be found in real-life event logs and whether context attributes resolve the ambiguity, commonly used event logs in process prediction benchmarks were analyzed.

[23] This subsection is based on Pfeiffer, Lahann and Fettke (2023).

Table 7: Number of cases, training examples, ambiguity candidates, as well as the ratio of ambiguous samples in commonly used event logs for process prediction (Pfeiffer et al. 2023).

Dataset	Helpdesk	BPIC 2012	BPIC 2017	BPIC 2013
#Cases	3,804	13,087	31,512	7,554
#Training examples	9,007	249,113	1,170,758	8,945
#Ambiguity candidates	5,006	103,326	497,099	1,296
Percentage	55.58%	40.67%	42.46%	14.49%

Results Table 7 shows the number and percentage of ambiguity candidates among the prefixes in the respective event logs. As can be seen, the share of ambiguity candidates is very high among all logs, except the BPIC 2013 incidents event log. The manual analysis of context attributes did not give indications that resolve the ambiguity. Thus, other alternatives must be found to deal with the large share of ambiguous samples.

If next-step prediction is used as a pre-training task to learn the process behavior before solving other tasks, label ambiguity is less of a problem, as the model's performance is evaluated on another task. In situations where the performance in predicting the next activity is of interest, alternative metrics can be used. For instance, by taking a probabilistic view on the result of the prediction model and the assessment method. Alternatively, next-activity prediction can be formulated as a multi-label classification problem. This, however, causes problems if the next activities are mutually exclusive, e.g., at decision points. This would require expert knowledge to construct ground truth labels y per input x.

In conclusion, constructing ground truth labels from existing event logs is complicated and error-prone. Expert knowledge might be required to obtain valid ground truth labels for assessing with accuracy.

3.3.2 Discussing Generalization in Next-Activity Prediction[24]

Following the conceptualization and quantification of the label ambiguity problem, a preceding publication further analyzes the next step prediction task for conceptual issues and discusses their impact on generalization. Two validity issues have been observed that make the next activity evaluation setup unreliable. First, if splitting the traces randomly or by time, a significant overlap between training and evaluation samples with the same prefix x in the train and test set is introduced. Such an example in the test set is a trivial prediction, as the model can simply repeat what it has seen during training and does not need to learn generally valid relationships in the data. Since many traces in event logs follow the same few control flow variants, it can be considered a serious problem when evaluating process prediction tasks. By building prefixes from traces, the portion of duplicates can increase even further. Since such predictions are trivial, the prediction model should reach an accuracy higher than the share of leaked examples. Otherwise, the model only reproduces

[24] This subsection is based on Abb, Pfeiffer, Fettke and Rehse (2024).

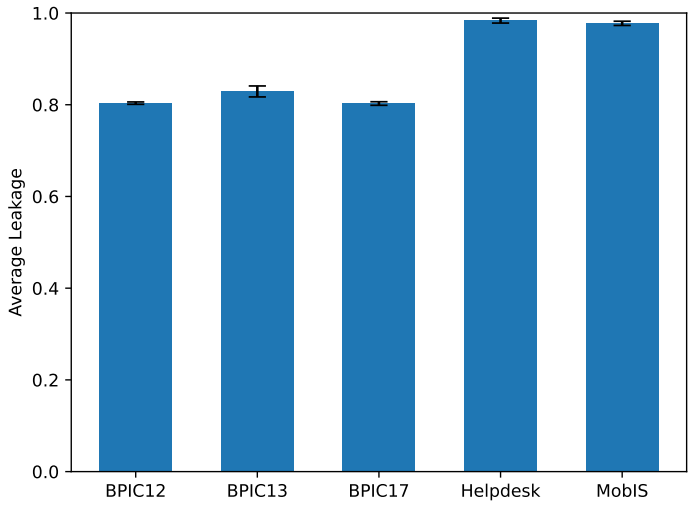


Figure 11: Average example leakage across all splits on the five event logs (Abb et al. 2024).

what it has seen during training. Interestingly, the performance has not yet been analyzed with the share of leaked samples in mind. Thus, it is unknown whether existing next step prediction models can actually make correct predictions for unseen samples and how well they perform in such situations.

Second, the maximum accuracy that can be reached is limited to a value lower than 100%, related to the percentage of ambiguous candidates in the event log. If there are multiple continuation options y for the same prefix x and no decisive feature in the prefix to determine which option to choose, the model cannot reach perfect accuracy.

Both observations have been validated on commonly used event logs using a standard evaluation procedure. This includes splitting the event log randomly or by time in 6 different splits, building prefixes on each split, and training and testing the prediction models on each of it.

Results The first observation, having duplicate examples in training and test sets, is referred to as example leakage. To quantify example leakage, only the control flow of prefixes is considered. Note that this approximates the "true" example leakage since no context attributes are considered, which can be decisive for the next activity.

As Figure 11 shows, the commonly used event logs, including many BPIC logs, exhibit a high example leakage of over 80% in the prefixes x. For some event logs, it is even close to 100%. Thus, as most samples have been seen before, one cannot conclude how well these models would perform on unseen samples and therefore, how well they can generalize.

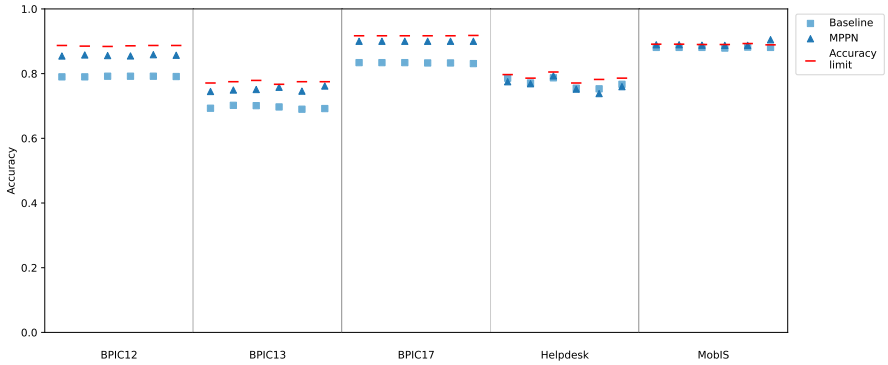


Figure 12: The accuracy limit as well as the accuracy of the baseline and MPPN model on all splits (Abb et al. 2024).

While the example leakage can be considered a lower limit for process prediction accuracy, there is also an upper limit for process prediction accuracy. This upper limit results from traces affected by label ambiguity, i.e., where the same prefix x has multiple valid next steps y. To quantify the effect of label ambiguity on accuracy, an experiment was conducted where a prediction model was designed to always make perfect predictions.

The best prediction a model can make is to predict the label y that is most commonly associated with a prefix x in the test set, i.e., following the "true" probability distribution. If a prefix x is given with more than one option on how to continue, the label associated with this prefix most frequently found in the test set is chosen as the prediction. Thus, the perfect prediction model makes predictions according to the test set. Such a model is illegitimate in machine learning as it has access to the usually secret test set. It is used to demonstrate the effect of label ambiguity only. The accuracy that this model reaches is called the accuracy limit.

In addition, the accuracy of a simple Ngram baseline model and the state-of-the-art process prediction model using context attributes (MPPN) are reported. This allows for setting the theoretical values in relation to the performance that can be reached with existing prediction models. Figure 12 shows the respective accuracy values for all five event logs and per split. Some interesting observations can be made. The accuracies are in a relatively small window on all event logs, especially on the Helpdesk and MobIS event logs, which feature a very high example leakage (see Figure 11). Further, no model reaches a higher accuracy than the accuracy limit. Even more interesting is the observation that on Helpdesk and MobIS event logs, the models reach a much lower accuracy than the example leakage. Given that leaked samples should be trivial predictions and the accuracy of any model, therefore, higher than the leakage, this shows a conceptual issue in evaluating process prediction models with accuracy. In combination with the observation that the example leakage is dangerously high, we conclude that this evaluation setup does not allow for measuring generalization. This calls for a discussion of what generalization in process prediction means.

Table 8: Example event log $L1$ with concurrency in activities *C1*, *C2*, and *C3* (Abb et al. 2024).

Event Log $L1$
⟨A, B, C1, C2, C3, D, E⟩
⟨A, B, C2, C1, C3, D, E⟩
⟨A, B, C2, C3, C1, D, E⟩
⟨A, B, C3, C1, C2, D, E⟩
⟨A, B, C3, C2, C1, D, E⟩

As a first step, a conceptualization of generalization for process prediction is presented, oriented on generalization in machine learning. The definition of generalization in general machine learning refers to the ability of the model to make correct predictions on samples not seen during training. This ensures that the model is robust enough to work well in application and ensures that the model has learned generally valid features instead of memorizing the samples. This requires building test sets that contain different samples from the training set.

Several prediction scenarios inspired by real-world process behavior are formulated and presented. They are classified into three generalization types: unseen control-flow variants, unseen attribute value combinations, and unseen attribute values. For each scenario, a situation is constructed that results in an unseen prefix to put in the test set and an expected prediction that the prediction model should make. These scenarios guide the evaluation procedure for generalization in process prediction. Two examples of such generalization scenarios will be given using the event log $L1$ in Table 8.

- Given $L1$ and the unseen prefix ⟨A, B, C1, C3, C2, D⟩, we expect the prediction model to predict E. Although the model has not seen this prefix due to a new order of $C1$, $C2$, and $C3$, it should have learned that the case always continues with E after D, regardless of the order of the previous activities.
- Given $L1$ and the unseen prefix ⟨A, B, C1, C3, C2⟩, we expect the prediction model to predict D. Again, the prediction model should have learned that regardless of the order of $C1$, $C2$, and $C3$, D always follows.

In total, 11 different scenarios are described, with 7 of them including context attributes, i.e., the model must be context-aware to make correct predictions for those unseen prefixes. These scenarios can be used to split existing event logs into distinct training and test event logs. Further, new event logs can be generated that are dedicated to evaluating the generalization capabilities of the process prediction model. Since real-life event logs are usually more complex, a mix of scenarios might be required to obtain logs with a realistic level of unseenness.

3.3.3 Conceptualizing Generalization in Next Activity Prediction[25]

Generalization is a fundamental goal of machine learning (Goodfellow et al. 2016) and known to be required for process prediction methods since processes tend to change over time (Ruta and Majeed 2011). As extensively researched in the previous two publications (Pfeiffer et al. 2023; Abb et al. 2024), existing research in process prediction has not adequately addressed this concept as the currently employed evaluation setup does not account for realistic testing conditions as expected in application. In turn, it suffers from high example leakage and is problematic for measuring and reporting the performance. Thus, a third publication in this stream of research extends the existing publication in three ways:

1. Example leakage and label ambiguity are introduced formally and related to common evaluation failures in machine learning research, extending the theoretical foundation of those observations
2. Generalization in next activity prediction is conceptualized based on a discussion on the goal and challenges of generalization in general machine learning
3. Two next activity prediction models are evaluated in their ability to generalize in synthetic and real scenarios based on the proposed conceptualization

By defining generalization based on machine learning research, a significant contribution is made that aligns generalization in process prediction with generalization in machine learning while still considering the specific challenges of process prediction. While this paper focuses specifically on next activity prediction, the conceptualization of generalization applies similarly to other process prediction tasks since many characteristics and challenges are similar.

Results The two conceptual issues found in next-step prediction evaluation procedures, i.e., label ambiguity and example leakage, have been linked to acknowledged evaluation failures in machine learning research (Liao et al. 2021). The problem of having many duplicates between the training and the test set is known as the issue of contaminated data (Liao et al. 2021). It leads to an overestimation of the model's performance. In contrast to this stands the issue of using accuracy as a point measure for intrinsically stochastic data. This has been described as label ambiguity in the first publication in this series of research (Pfeiffer et al. 2023) and is linked to label error as described in (Liao et al. 2021). Reporting and measuring the performance in that way does not meaningfully capture the ability of the prediction model to make accurate predictions. It can lead to an underestimation of the model's performance. Both phenomena in combination lead to the conclusion that we do not know how well existing prediction models perform on unseen prefixes, i.e., how well they generalize.

In general machine learning, generalization aims to make correct predictions for unseen samples. When using machine learning techniques, several challenges arise, such as the incompleteness of the data, the distributions found in the data, the

[25] This subsection is based on Pfeiffer, Abb, Fettke and Rehse (2025).

capacity of the model, and the correct splitting of the available data. Depending on the data and the task, generalization is defined differently concerning the challenges that arise.

Thus, the publication defines generalization in next activity prediction and discusses the challenges of the data and task. We define generalization in next activity prediction as the ability of the model to make correct predictions for unseen trace prefixes. Note that this definition, which follows the general definition of machine learning, has not yet been applied to next activity prediction. Instead, generalization has been defined differently, e.g., as the ability to generate unseen trace suffixes from seen prefixes, which is much more difficult.

In the publication, three major challenges have been identified that make next activity generalization difficult to achieve

1. The training data contains only a small fraction of the process behavior
2. The empirical distribution of variants in the training event log can be different from the distribution found in application
3. Process prediction models have enough capacity to perfectly fit the training log, which would, without any countermeasures, not allow it to make correct predictions for unseen prefixes

In process mining, one typically assumes that the event log is incomplete with respect to the behavior of the process. Consequently, the data on which a process prediction model is trained contains an even smaller fraction of the total behavior of the process since a part of the data has to be held back for testing. Due to this incompleteness, we have to assume that we are confronted with prefixes during application that were not seen. The Pareto distribution of trace variants typically found in event logs leads to a large number of repetitive prefixes (note that the targets usually follow a different distribution). This leads to many leaked prefixes between the training and the test set. As processes tend to change over time, the distribution of variants likely changes, i.e., the prefixes in application are likely different than those found during training. When splitting the data into training and test sets, this must be considered.

To measure the generalization as the expected error in application, the data has to be split in such a way that the test set reflects the conditions to be expected in application. Splitting by time should give the most realistic setting compared to splitting randomly or by variant, as natural distribution changes are maintained. During training, regularization techniques should be implemented as the prediction model's capacity (in terms of neurons and layers) can quickly surpass the complexity of the training task. The number of unique prefix variants can grow to ten to some hundred thousand, while a neural network quickly grows to even more parameters.

An alternative to accuracy, which has been shown to be problematic, should be considered for performance assessment. Instead, we argue for using cross-entropy as an additional performance metric that is not as biased as accuracy. Cross-entropy eval-

uates the model's performance probabilistically instead of using an "all-or-nothing" manner of accuracy that cannot deal with the variability of process continuations. A cross-entropy estimation has been defined for the next activity prediction task.

Finally, analogous to In-Domain and Out-of-Domain generalization in ML, these two types of generalization are also defined for process prediction tasks. While the In-Domain setting deals with settings in which the distribution of examples in the training and test sets is identical, or at least not too different, Out-of-Domain settings concern changes in data that are more significant.

A process prediction model is said to generalize well if it performs well on the training data and minimizes the gap between the training and the test error, measured with cross-entropy. This translates to a process prediction model that accurately predicts how processes continue and does so equally well on the training as well as on the unseen examples in the test set.

In the experimental part, different event logs were used to evaluate the generalization capabilities of existing next activity prediction models using the generalization definition defined beforehand. We decided to simulate various scenarios to assess the generalization capabilities per process characteristic and difficulty. First, synthetic event logs were simulated based on the nine generalization scenarios presented in the previous publication Abb et al. (2024), resulting in 18 event logs with two complexity levels per scenario. Since real-life processes contain combinations of the presented scenarios, an additional process model containing all scenarios has been simulated. This resulted in three additional event logs with increasing difficulty.

For each scenario, a total of 10,000 traces have been simulated. They are split with a ratio of 80/20 so that no prefixes are leaked from the training to the test set with regard to the generalization characteristic that is evaluated in that scenario. As a result, the models are evaluated on unseen prefixes only, while the total number of variants in the test set corresponds to 20% of the process behavior in terms of variants. This ensures that the models are trained on a fraction of the total process variability and that all test examples are unseen.

Further, the models were also evaluated on real-life event logs (BPIC 12 and 17) in two settings. Since we could not split the event logs by characteristics of the traces, as we did for the synthetic scenarios, we opted for a temporal split that naturally imitates changes through time. Both logs were split such that the first half of the traces (50%) ended in the training split and the last half in the test split. Surprisingly, there is still a high example leakage of more than 70%. Therefore, we created an additional setting in which all prefixes from the test set were removed with a control flow found in the training split. While the first setting contains all prefixes, all duplicated prefixes from the test set were removed in the second setting. As a result, the test set consists of unseen prefixes only.

Table 9 shows the results of the experiments using the 21 synthetic event logs with different scenarios. Except for scenarios *CF5*, introducing unseen activities in the prefixes, as well as *Comb3*, at least one model generalizes well with respect to the

Table 9: Cross-entropy (CE) and accuracy (ACC) on the training and test split (Pfeiffer et al. 2025).

		LSTM				MPPN			
		Train		Test		Train		Test	
		CE	ACC	CE	ACC	CE	ACC	CE	ACC
Simple	CF1	0.001	100.00%	0.002	100.00%	0.0	100.00%	0.0	100.00%
	CF2	0.841	52.28%	0.844	52.60%	0.800	56.11%	1.037	49.51%
	CF3	0.000	100.00%	0.000	100.00%	0.0	100.00%	0.0	100.00%
	CF4	0.256	92.19%	0.236	93.47%	0.253	92.32%	1.253	82.85%
	CF5	0.177	88.63%	0.266	85.74%	0.009	99.72%	0.926	84.16%
	ATT1	0.176	88.62%	0.177	88.05%	0.040	97.51%	0.041	97.41%
	ATT2	0.702	50.41%	0.700	61.26%	0.026	99.15%	0.090	96.16%
	ATT3	0.177	88.70%	0.177	88.67%	0.007	99.78%	0.009	99.71%
	ATT4	0.700	54.00%	0.698	55.26%	0.087	96.91%	0.326	90.91%
Advanced	CF1	0.001	100.0%	0.001	100.0%	0.0	100.0%	0.0	100.0%
	CF2	0.829	55.76%	0.895	53.22%	0.798	57.52%	1.491	48.06%
	CF3	0.000	100.0%	0.000	100.0%	0.0	100.0%	0.001	99.43%
	CF4	0.205	92.04%	0.205	92.45%	0.200	92.26%	0.231	91.20%
	CF5	0.177	88.53%	0.821	75.00%	0.007	99.80%	2.720	56.12%
	ATT1	0.176	88.72%	0.176	88.66%	0.014	99.56%	0.014	99.54%
	ATT2	0.701	51.03%	0.702	50.75%	0.104	96.18%	0.145	94.15%
	ATT3	0.177	88.31%	0.177	88.60%	0.005	99.83%	0.006	99.78%
	ATT4	0.699	54.24%	0.697	55.18%	0.055	98.05%	0.321	91.92%
Combinations	Comb1	0.233	83.82%	0.233	83.85%	0.074	97.07%	0.169	93.05%
	Comb2	0.232	83.34%	0.251	83.60%	0.076	97.00%	0.324	87.63%
	Comb3	0.349	75.29%	0.379	75.45%	0.068	97.63%	2.319	56.96%

specifics of the scenarios in each setting. In general, the LSTM model generalizes very well on the control-flow scenarios (*CF1* - *CF5*) while the MPPN, as expected, performs better in scenarios with context attributes. However, while the MPPN usually reaches a lower training error, its generalization error is often higher. In all scenarios, the MPPN shows that using context attributes in addition to the control flow can lead to better performance. However, the results also show that unseen context attributes, e.g., *ATT4*, *Comb2*, and *Comb3*, can make generalization more challenging. In scenario *Comb3*, the performance of the MPPN decreases considerably, from which we conclude that the MPPN model struggles to generalize in the presence of unseen combinations of activity and price attribute values. Similarly, unseen activity attribute values used in *CF5* are also challenging for existing next activity prediction models.

Table 10 shows the result on the event logs BPIC 12 and BPIC 17. Again, both models perform considerably well. The LSTM model maintains a smaller generalization error, while the MPPN can reach higher overall performance values. These results are in line with the results on the synthetic scenarios. The performance of

Table 10: Cross-entropy (CE) and accuracy (ACC) for setting *50:50* (no removal of leaked prefixes) and setting *unseen* (all leaked prefixes removed) (Pfeiffer et al. 2025).

	LSTM				MPPN			
	Train		Test		Train		Test	
	CE	ACC	CE	ACC	CE	ACC	CE	ACC
BPIC 12 *50:50*	0.381	85.29%	0.394	84.55%	0.448	83.36%	0.448	83.13%
BPIC 12 *unseen*	0.381	85.29%	0.566	81.49%	0.448	83.36%	0.890	72.22%
BPIC 17 *50:50*	0.376	87.22%	0.349	87.20%	0.246	91.13%	0.241	91.21%
BPIC 17 *unseen*	0.376	87.22%	0.549	81.15%	0.246	91.13%	0.513	82.70%

the prediction models stays almost unchanged in the *50:50* setting but drops considerably in the *unseen* setting. Thus, we conclude that prediction models can still be enhanced in their ability to predict the continuation options for unseen prefixes.

Finally, we investigate scenario *CF2* in more detail as the performance of the models on this event log split seems insufficient at first. As seen in Table 9, the models reach only around 50% accuracy. However, as we will show by inspecting the probability distribution learned by the model during training, the low(er) accuracies are due to multiple continuation options and the absence of decisive attributes to determine which of the options follows. For instance, in *CF2*, there are up to five different activities to follow for each unseen prefix in the test set.

Table 11: Probabilities given by softmax of LSTM for activities with unseen prefix on *CF2* simple. Activities in bold are ground truth activities y (Pfeiffer et al. 2025).

Unseen prefix	A	B	C	D	E	F	G	H	I	J
$\langle A, B, C, E \rangle$	0.00	0.00	0.00	**0.22**	0.01	**0.30**	**0.23**	**0.24**	0.00	0.00
$\langle A, B, C, F, D \rangle$	0.00	0.00	0.00	0.01	**0.28**	0.00	**0.36**	**0.35**	0.00	0.00
$\langle A, B, C, D, E, G \rangle$	0.00	0.00	0.00	0.01	0.00	**0.49**	0.01	**0.48**	0.00	0.00
$\langle A, B, C, D, E, F, G \rangle$	0.00	0.00	0.00	0.01	0.01	0.01	0.01	**0.97**	0.00	0.00

Table 11 shows the softmax probability the LSTM model predicted for the unseen prefixes in CF2 simple. As can be seen, the model gives high probabilities for all next activities that can potentially follow (highlighted in bold). Only very low probabilities are predicted for activities that cannot be followed. This shows that the model accurately learned the process behavior and that accuracy gives an unreliable picture of the true performance of the model in the next activity prediction task.

In conclusion, the results show that process prediction models can generalize very well and learn correct continuation options even for unseen process behavior. These observations are contrary to previous research. Interestingly, prediction models learn concurrency, as shown in Table 11, and can make accurate predictions even if activities not part of the training set are in the test prefixes. These observations prove that *PRMs* learn process behavior accurately - if measured adequately.

3.4 Internal Validity through Entropy-based Measures[26]

The evaluation setup of next activity prediction models using accuracy has shown to suffer from issues that are of a conceptual nature, as discussed through Section 3.3. These problems have a strong impact when evaluating, comparing, and interpreting different prediction models on the next-step prediction task. For representation learning, where this task is used during pre-training (e.g. in Pfeiffer et al. (2021)), the quality of the learned representation is of interest instead of the quality of the model to predict a single next activity correctly. Therefore, an accurate measure allowing for the interpretation of the quality of the model to represent the data is required. This measure should give information on how accurately the *PRM* has learned the underlying process using the NSP task. While the previous Subsection 3.3.3 focused on generalization in NSP, another publication deals with the question of how to close the learned representation reflects the underlying process.

Language models, learning textual data by estimating probabilities for next words, use perplexity to express their predictive power (Murphy 2022; Jurafsky and Martin 2025). This metric is calculated based on the cross-entropy of the language model. The idea is that the model should give high probabilities to next words (Jurafsky and Martin 2025, p.214), i.e., not to be "perplexed" when seeing the (actual) next word. Aggregated and normalized over a corpus consisting of sentences of natural language, it should give an estimation of how accurately next words in the whole corpus can be estimated. The lower the perplexity, the better the corpus has been learned. Those measures are based on information theory and are well established.

A similar way of evaluation could be applied to *PRMs*, too, with the event log being the corpus to learn. Instead of using accuracy, the cross-entropy is used as a metric of how accurately, in a probabilistic sense, the model has learned the data. As described in Section 2.5, traces as well as the sequence of events generated by the system S can be described as stochastic processes. The *Ngram* entropy quantifies the randomness of a sequence of elements and measures how well the next activity can be estimated from the previous n activities. Since the entropy and cross-entropy are closely related, the performance of a *PRM* can be set in relation to the *Ngram* entropy to find out how accurately the data has been learned.

In previous work, the term label ambiguity has been used to describe the situation where multiple valid next activities are possible for the same prefix (Pfeiffer et al. 2023). Originally, this term denotes the uncertainty about the labels. This, however, might not be the correct interpretation for such situations. Instead, the fact that there is not always one activity how a process instance can proceed can be described as stochasticity in processes. For example, there can be a 70% chance that *C1* follows and a 30% chance that *C2* follows. This is not a problem in the process or the data, but simply a characteristic of it. Thus, using stochastic measures that take this characteristic of processes into account can help to overcome the problems emerging with accuracy.

[26] This section is based on Pfeiffer and Fettke (2024) and Pfeiffer, Abb, Fettke and Rehse (2025).

Guided by this idea, Pfeiffer and Fettke (2024) discusses the link between estimating next activities in traces using entropy and predicting next activities from prefixes with prediction models. This way, the stochasticity is taken into consideration. Further, the performance of different process prediction models is compared against the *Ngram* entropy.

Results Instead of assessing process prediction models with accuracy using the prefix leakage as a lower bound, cross-entropy assesses against entropy (rate) as the lower bound. The entropy rate gives information on how "surprising" a sequence is. Thus, a sequence with lower entropy should be easier to learn than a sequence with higher entropy. Cross-entropy (estimation) builds upon entropy to compare how closely the learned probability distribution of a prediction model reflects the probabilities found in real data. Due to the close link between estimating the entropy for an *Ngram*, i.e., the entropy rate of a sequence, and the probability of the next element given previous events, the cross-entropy that a prediction model reaches can be compared to the entropy rate of the underlying sequence.

If the cross-entropy is lower than the entropy (rate), the models have learned the statistical distribution of prefixes accurately. The longer the *Ngrams* become, i.e., the further the n of the entropy rate increases, the more challenging it will be to reach a lower cross-entropy as the entropy rates are computed based on probabilities for *Ngrmas* on the whole event log while the prediction models have to predict the next activity and are trained on the training set only.

Setting the performance of a *PRM* in relation to this and interpreting it with the knowledge of the entropy (rate) should therefore be more precise than using accuracy. Further, interpreting the performance with no knowledge about the complexity of the underlying data can be dangerous. For instance, an accuracy of 80% is commonly assumed to be a very good accuracy, indicating that the model is accurate. If, however, 80% of the process is strictly sequential, i.e., 80% of prefixes x have a single next activity y, then the accuracy of the model only indicates that it has learned sequential patterns. In such situations, the entropy rate would also be low. The prediction model has to reach a lower cross-entropy than the entropy rate of the data. In contrast, an accuracy of 50% does not necessarily mean that the model is bad if the process has many options how to continue (as shown in Subsection 3.3.3). For such processes, the event log should have a high entropy rate.

As the results of the experiments on various real-life event logs show, existing prediction models can reach a lower cross-entropy than the respective unigram entropy. This shows that the models can learn longer-range dependencies in sequences of events. For some event logs, however, the cross-entropies are not too much lower than the uni- or bigram entropy rates. This indicates that although the *PRM* models reach very high accuracies, there is much more improvement possible. The results are further discussed in Section 4.4.

3.5 Summary and Conclusion

This chapter has summarized all contributions on the intrinsic evaluation of *PRMs* and their learned representations. All these methods do not need labeled data from experts and can be used to evaluate the models with the data given for training.

While the trace representations of the S2SA model (Baier et al. 2020) do not show signs of natural clustering, most likely due to the suboptimal reconstruction performance, the representation space of the MPPN (Pfeiffer et al. 2021) is much more organized. As will be demonstrated in Section 4.3, traces that are similar in multiple dimensions are placed close to each other in the representation space. This shows strong signs of natural clustering.

Besides analyzing the internal organization of traces in *PRMs*, the next step prediction task, which has shown some unusual progress pattern, has been critically examined (Pfeiffer et al. 2023; Abb et al. 2024; Pfeiffer et al. 2025). It has been found that the evaluation setup used in next-activity prediction suffers from multiple validity issues that negatively affect the performance of prediction models and, more severely, the interpretation of their actual quality. In consequence, generalization as a major goal of machine learning, cannot be assessed reliably.

A framework consisting of the goals and challenges of generalization in process prediction has therefore been introduced (Pfeiffer et al. 2025), and the generalization capabilities of existing process prediction models have been assessed along several synthetic and real-world scenarios. The results suggest that existing models can generalize surprisingly well and better than initially assumed. Further, cross-entropy as an alternative evaluation metric to accuracy has been motivated and used during the interpretation of the results. As it has shown, the accuracy does not reliably report the quality of the model in predicting next activities.

Similarly, Pfeiffer and Fettke (2024) also motivated the use of cross-entropy for evaluating prediction models on event log data. Different from Pfeiffer et al. (2025), its usage is motivated based on entropy estimations of event sequences. By setting the performance of a *PRM* in relation to the entropy of the underlying (system) sequences, a more reliable way of evaluating and interpreting the performance is achieved.

More research is required on evaluating *PRMs* with entropy-based measures and whether more insights can be gained about what such models have learned about the process. Further, the representation space of *PRMs* should be analyzed in more detail as it can reveal interesting properties about what the model has learned.

Chapter 4

Research Area 3 - Application and Demonstration of Process Representations Models

4.1 Overview

Process Representation Models are designed to support solving different tasks on event log data. Thus, after designing their architecture, training the model, and evaluating its intrinsic quality, their usefulness has to be demonstrated through various experiments on different downstream tasks. Evaluating and comparing the performance of a model on different tasks is the most important type of evaluation. This chapter presents all contributions that apply *PRMs* to real-world tasks that have a benefit for the user, thereby contributing to RA 3.

This chapter is organized by the downstream tasks. First, Section 4.2 summarizes the contributions to process prediction, i.e., predicting future process instance behavior. This includes publications on next step prediction (Pfeiffer et al. 2021; Abb et al. 2024), outcome prediction (Pfeiffer et al. 2021), and deviation prediction and deviation pattern prediction (Grohs et al. 2023, 2025). Section 4.3 presents results on trace classification tasks and trace retrieval, including publications on compliance classification (Baier et al. 2020), and anomaly and conformance checking (Lahann et al. 2023).

Another interesting task is grouping lower-level events into higher-level activities, i.e., event abstraction (Rebmann et al. 2023) presented in Subsection 4.3.4. Finally, predictions on the underlying system S (Pfeiffer and Fettke 2024) that generated the traces are presented in Section 4.4.

4.2 Predictions of Future Process Instance Behavior

Predicting the future behavior of process instances is the most frequently researched predictive task in BPM from early on (Ruta and Majeed 2011). This section demonstrates how *PRMs* can be used for such tasks. There is a wide range of process instance characteristics to predict. Contributions within this thesis have been made to the tasks of next-step prediction, outcome prediction, as well as deviation prediction.

4.2.1 Next Step and Outcome Prediction[27]

Given its versatility, the MPPN model has been evaluated and benchmarked against state-of-the-art prediction models on the next step and outcome prediction task, using a large variety of event logs. Table 12 shows an overview of the event logs, their characteristics, and the attributes used as input for the MPPN model for next step and outcome prediction. In this experiment, cases containing more than 64 events were removed since those were mostly outliers and would have increased the training time drastically. Six prediction tasks have been tackled:

- $NSP_{activity}$: prediction of the next activity
- $NSP_{resource}$: prediction of the resource of the next activity
- $NSP_{timestamp}$: prediction of the duration till the next activity will be executed
- $OUT_{activity}$: prediction of the last activity in the trace
- $OUT_{resource}$: prediction of the resource of the last activity
- $OUT_{timestamp}$: prediction of the duration till the trace ends

For all tasks, prefixes of all lengths were built and given as input to the prediction models. Before being fine-tuned on these tasks, the MPPN was pre-trained once per event log. Each event log was split into a training, validation, and test set with a ratio of 70/10/20. While the test set remains the same in all runs, the remaining log was split randomly into training and validation from run to run. In total, 10 runs were performed, and the performance on the test set was averaged across all runs. Early stopping and cycling learning rates were used.

The MPPN has been compared to four state-of-the-art prediction models that contain early and very recent approaches. The approach of Evermann et al. (2016) is the first LSTM model developed for process prediction. Another LSTM model has been developed by Camargo Camargo et al. (2019) featuring three different architectures on how to unify attribute perspectives (*Ca_Spez*, *Ca_concat*, and *Ca_full*). Similarly, the approach by Tax et al. (2017) that utilizes LSTM, too, also offers three different ways to combine the information from different perspectives (*Tax_Spez*, *Tax_Mixed*, and *Tax_Shared*). MiDA, a state-of-the-art approach developed by Pasquadibisceglie et al. (2020), builds on CNNs and can process attributes *activity*, *resource*, and *timestamp*. Those models were re-implemented based on the original papers and

[27] This subsection is based on Pfeiffer, Lahann and Fettke (2021).

Table 12: Event logs, statistics, and attributes used in the experiments (Pfeiffer et al. 2021).

	#Traces	#Events	Avg. trace length	Avg. trace duration	Input Attributes		
					categorical	numerical	temporal
Helpdesk	4,580	21,348	4.66	62.9 days	activity, resource		timestamp
BPIC12	13,087	262,200	20.04	150.2 days	activity, resource	AMOUNT_REQ	timestamp
BPIC12_Wc	9,658	72,413	7.50	95.6 days	activity, resource	AMOUNT_REQ	timestamp
BPIC13_CP	1,487	6,660	4.48	426.5 days	activity, resource, resource country, organization country, organization involved, impact, product, org:role		timestamp
BPIC17_O	42,995	193,849	4.51	23.9 days	activity, Action, NumberOfTerms, resource	FirstWithdrawalAmount, MonthlyCost, OfferedAmount, CreditScore	timestamp
BPIC20_RFP	6,886	36,796	5.34	31.6 days	org:role, activity, resource, Project, Task, OrganizationalEntity	RequestedAmount	timestamp
MobIS	6,555	166,512	25.40	1194.4 days	activity, resource, type	cost	timestamp

the provided source code. To ensure a unified test setting, some aspects of the original implementations have been altered, like prefix generation, train-test-splitting, hyperparameters, or pre-processing steps. All models were trained and tested on the same training, validation, and test splits.

Results Table 13 shows the results of the benchmark, i.e., the average accuracy (for classification tasks) and mean absolute error for the prediction of timestamps (in days) with the standard deviation. As the results show, there is no single superior model that performs best in all tasks. Nevertheless, the MPPN model performs best on most datasets and tasks, especially on the categorical next step and outcome prediction tasks. On the regression tasks, the MPPN performs well but often a little worse than the best-performing model. The MPPN is also very constant, as the standard deviation is low in most settings.

Overall, the results show that the MPPN is effective, constant, and requires less hyperparameter tuning than other models. While the existing models often have a larger standard deviation in their performance results, the MPPN performs very reliably. The instability of the other models could be caused by the use of embedding layers, which can become huge if attributes have many values and are therefore difficult to train properly. In contrast, using GAFs in combination with pre-training makes the MPPN more robust, as fewer parameters are required. Fewer parameters can be trained more efficiently, while the unsupervised pre-training conditions those parameters. This allows for the effective use of more attributes than existing models and validates the intended effects of the design choice.

The MPPN has also been used as a state-of-the-art model in the accuracy limit analysis for next activity prediction, as presented in Subsection 3.3.1. In the experiment conducted in that paper, the MPPN reaches accuracy values very close to or even on par with the accuracy limit, showing that the model predicts the next activities almost as well as theoretically possible.

Table 13: Bold indicates the best-performing model in that task (Pfeiffer et al. 2021).

Dataset	Model	$NSP_{activity}$	$NSP_{resource}$	$OUT_{activity}$	$OUT_{resource}$	$NSP_{timestamp}$	$OUT_{timestamp}$
Helpdesk	Evermann	0.651±0.128	0.222±0.005	**0.994±0.000**	0.811±0.000	—	—
	Ca_Spez.	0.693±0.168	0.289±0.071	**0.994±0.000**	0.811±0.000	7.95±0.576	6.654±0.101
	Ca_concat	0.696±0.116	0.421±0.035	**0.994±0.000**	0.811±0.000	7.63±0.052	6.739±0.253
	Ca_full	0.774±0.077	0.432±0.000	**0.994±0.000**	0.811±0.000	5.308±0.288	7.018±0.225
	Tax_Spez.	0.763±0.082	—	**0.994±0.000**	—	7.777±0.526	6.895±0.253
	Tax_Mixed	0.3±0.003	—	**0.994±0.000**	—	14.849±0.034	7.197±0.101
	Tax_Shared	0.793±0.004	—	**0.994±0.000**	—	5.088±0.129	6.67±0.100
	MiDA	0.693±0.120	0.263±0.089	**0.994±0.000**	0.811±0.000	**4.898±0.043**	**6.629±0.166**
	MPPN	**0.805±0.003**	**0.691±0.006**	**0.994±0.000**	**0.847±0.008**	5.197±0.126	6.691±0.089
BPIC12	Evermann	0.595±0.107	0.149±0.000	0.417±0.000	0.172±0.000	—	—
	Ca_Spez.	0.795±0.030	0.333±0.282	0.417±0.000	0.177±0.015	0.693±0.208	7.82±0.033
	Ca_concat	0.74±0.071	0.426±0.164	0.417±0.000	0.172±0.000	0.722±0.206	7.849±0.076
	Ca_full	0.756±0.064	0.283±0.197	0.417±0.000	0.184±0.025	0.687±0.226	6.649±0.084
	Tax_Spez.	0.585±0.194	—	0.417±0.000	—	0.734±0.155	7.477±0.127
	Tax_Mixed	0.615±0.182	—	0.417±0.000	—	0.544±0.205	6.678±0.101
	Tax_Shared	0.824±0.008	—	0.487±0.019	—	**0.542±0.167**	6.693±0.080
	MiDA	0.565±0.123	0.149±0.000	0.417±0.000	0.172±0.000	0.625±0.041	**6.587±0.047**
	MPPN	**0.846±0.006**	**0.775±0.002**	**0.53±0.005**	**0.316±0.004**	0.82±0.079	6.694±0.066
BPIC12_Wc	Evermann	0.774±0.000	0.104±0.000	0.435±0.000	0.11±0.000	—	—
	Ca_Spez.	0.775±0.002	0.104±0.000	0.435±0.000	0.113±0.005	1.799±0.088	8.31±0.058
	Ca_concat	0.794±0.027	0.104±0.000	0.435±0.000	0.115±0.013	1.843±0.169	8.333±0.041
	Ca_full	0.792±0.026	0.104±0.000	0.443±0.026	0.112±0.005	1.81±0.125	7.455±0.063
	Tax_Spez.	0.713±0.081	—	0.435±0.000	—	1.765±0.098	7.932±0.086
	Tax_Mixed	0.774±0.000	—	0.435±0.000	—	**1.595±0.064**	7.51±0.106
	Tax_Shared	0.773±0.001	—	0.537±0.057	—	1.645±0.070	**7.409±0.110**
	MiDA	0.805±0.022	0.104±0.000	0.435±0.000	**0.155±0.028**	1.767±0.115	7.424±0.103
	MPPN	**0.815±0.006**	**0.237±0.011**	**0.558±0.01**	0.147±0.009	1.761±0.061	7.528±0.072
BPIC13_CP	Evermann	0.417±0.113	0.082±0.005	**1.0±0.000**	0.211±0.000	—	—
	Ca_Spez.	0.481±0.090	0.086±0.000	**1.0±0.000**	0.211±0.000	50.927±2.339	137.718±3.125
	Ca_concat	0.524±0.004	0.086±0.000	**1.0±0.000**	0.211±0.000	51.672±3.62	139.412±5.032
	Ca_full	0.493±0.065	0.106±0.035	**1.0±0.000**	0.211±0.000	67.168±7.233	137.193±6.280
	Tax_Spez.	0.502±0.067	—	**1.0±0.000**	—	50.785±4.395	140.481±4.913
	Tax_Mixed	0.309±0.003	—	**1.0±0.000**	—	112.867±0.279	176.167±0.930
	Tax_Shared	0.51±0.011	—	**1.0±0.000**	—	**47.741±1.217**	144.528±21.964
	MiDA	0.434±0.110	0.083±0.005	**1.0±0.000**	0.211±0.000	54.949±4.044	128.185±10.555
	MPPN	**0.562±0.009**	**0.178±0.024**	**1.0±0.000**	**0.216±0.008**	54.922±3.948	**127.824±3.806**
BPIC17_O	Evermann	0.818±0.000	0.067±0.005	0.509±0.032	0.186±0.041	—	—
	Ca_Spez.	0.818±0.000	0.064±0.000	0.513±0.018	0.192±0.000	3.628±0.057	9.604±0.017
	Ca_concat	0.818±0.000	0.226±0.261	0.501±0.027	0.192±0.000	3.611±0.082	9.606±0.014
	Ca_full	0.818±0.000	0.081±0.048	0.52±0.001	0.192±0.000	3.627±0.105	9.519±0.025
	Tax_Spez.	0.67±0.065	—	0.454±0.019	—	3.529±0.019	9.688±0.145
	Tax_Mixed	0.726±0.178	—	0.458±0.014	—	3.999±0.503	9.768±0.184
	Tax_Shared	0.818±0.000	—	0.519±0.000	—	3.531±0.037	9.47±0.021
	MiDA	**0.836±0.030**	0.064±0.000	**0.828±0.002**	0.192±0.000	**3.297±0.037**	**8.946±0.059**
	MPPN	0.818±0.000	**0.553±0.061**	0.518±0.001	**0.208±0.001**	3.567±0.068	9.534±0.016
BPIC20_RFP	Evermann	0.699±0.099	0.817±0.084	**0.957±0.000**	**0.958±0.000**	—	—
	Ca_Spez.	0.756±0.087	0.841±0.020	**0.957±0.000**	**0.958±0.000**	2.556±0.142	6.068±0.185
	Ca_concat	0.704±0.09	**0.997±0.000**	**0.957±0.000**	**0.958±0.000**	2.631±0.199	6.062±0.079
	Ca_full	0.804±0.025	**0.997±0.001**	**0.957±0.000**	**0.958±0.000**	2.634±0.252	5.931±0.117
	Tax_Spez.	0.791±0.085	—	**0.957±0.000**	—	2.269±0.085	5.933±0.087
	Tax_Mixed	0.431±0.252	—	**0.957±0.000**	—	3.827±2.194	8.55±0.058
	Tax_Shared	**0.849±0.001**	—	**0.957±0.000**	—	**2.12±0.095**	**5.468±0.181**
	MiDA	0.55±0.109	**0.997±0.001**	**0.957±0.000**	**0.958±0.000**	2.673±0.173	5.842±0.086
	MPPN	**0.849±0.001**	**0.997±0.000**	**0.957±0.000**	**0.958±0.000**	3.018±0.849	6.495±0.909
MobIS	Evermann	0.767±0.140	0.163±0.000	0.798±0.000	0.075±0.000	—	—
	Ca_Spez.	0.87±0.040	0.163±0.000	0.798±0.000	0.075±0.000	4.648±0.560	30.106±0.814
	Ca_concat	0.836±0.034	0.163±0.000	0.798±0.000	0.075±0.000	4.801±0.525	30.133±0.526
	Ca_full	0.838±0.038	0.163±0.000	0.798±0.000	0.075±0.000	3.966±0.922	24.449±0.354
	Tax_Spez.	0.85±0.079	—	0.798±0.000	—	3.919±0.968	28.236±1.569
	Tax_Mixed	0.545±0.188	—	0.798±0.000	—	2.333±0.602	21.384±0.977
	Tax_Shared	0.926±0.008	—	0.805±0.009	—	**2.323±0.638**	**20.963±0.420**
	MiDA	0.7±0.154	0.163±0.000	0.798±0.000	0.075±0.003	2.992±0.372	24.498±0.405
	MPPN	**0.934±0.003**	**0.536±0.026**	**0.812±0.002**	**0.121±0.023**	4.827±0.420	22.454±1.011

Additionally, the results of the MPPN on the generalization scenarios, presented in Subsection 3.3.3, prove that the model can generalize to unseen prefixes with unseen context attribute values (Pfeiffer et al. 2025). Even in more complex settings like the advanced scenarios, the combinations of scenarios, or real-life event logs, the MPPN model reaches low cross-entropy and high accuracy.

4.2.2 Deviation and Deviation Pattern Prediction[28]

Deviations in business processes can have a negative impact on organizations as they risk becoming non-compliant, delaying processes, or resulting in other unwanted behavior. Conformance checking, as a technique to ensure compliance, is reactive and cannot detect or prevent deviations from occurring in the future of running process instances. To predict deviation in business processes, the MPPN has been adapted for predicting deviations and deviation patterns before they occur by modifying its pre-processing and training procedure (see section 2.3.2.2).

Three different ways to use the MPPN have been tested. In one paper, only the pre-trained embeddings of the MPPN are used as a feature vector for deviation prediction. In the second paper, the pre-trained MPPN is fine-tuned end-to-end on deviation prediction. Finally, the MPPN has been used to predict trace suffixes to detect and separate compliant from non-compliant behavior.

4.2.2.1 MPPN Embeddings for Deviation Prediction

In Grohs et al. (2023), the embeddings of the pre-trained MPPN were used for deviation prediction and compared against a specifically developed deviation prediction model called BPDP. The MPPN embeddings were obtained by pre-training the MPPN once per event log in the standard pre-training procedure to predict the next attribute values of the attributes found in the event log. Afterwards, the output of the final layer before the heads (*NN2* in Figure 5), a feature vector of size 128, is used to train a classifier for deviation prediction.

The BPDP model either uses the MPPN embeddings or a complex-index-based encoding (CIBE) (Leontjeva et al. 2016) for deviation prediction. It is a specifically developed machine learning model tailored towards deviation prediction, consisting of a shallow neural network made of fully connected layers. For each deviation type, a separate BPDP model is trained.

In all versions, a weighted loss function is used that puts more weight on the deviation class than on the conforming class (16 vs. 1). The training samples are undersampled per deviation type using the one-sided selection algorithm (Kubat and Matwin 1997). This does not balance the whole training set but ensures that important information from the minority class is preserved. Traces are split randomly in training (2/3) and test (1/3).

[28] This subsection is based on Grohs, Pfeiffer and Rehse (2023) and Grohs, Pfeiffer and Rehse (2025).

Table 14: Descriptive statistics for evaluation event logs (Grohs et al. 2023).

Log L		**Traces**	**Events**	**Trace Attr.**	**Trace Length**			**Dev. Types**	**Deviating Traces**		
					min.	**avg.**	**max.**		**min.**	**avg.**	**max.**
BPIC 12	A	13,087	60,849	1	3	4.7	8	3	399	927	1,191
	O	5,015	31,244	1	3	6.2	30	8	20	984	1,761
BPIC 20	Dom. Dec.	10,500	56,437	4	1	5.4	24	19	1	252	2,154
	Int. Dec.	6,449	72,151	17	3	11.2	27	46	1	292	1,701
	RfP	6,886	36,796	8	1	5.3	20	23	1	116	1,027
	Prep.	2,099	18,246	16	1	8.7	21	41	1	64	530
MobIS		3,354	55,809	1	11	16.6	49	43	1	182	1011

Table 14 shows the event logs, their characteristics, as well as the number and statistical information on the deviation types. Those event logs were selected as the to-be process models were available for these logs. To-be process models are required to obtain labels for training and testing. The event logs differ in size, trace lengths, attributes, deviation types, and class imbalance, resulting in a realistic setup for experimentation. All traces in the event log were aligned to the to-be process model to generate labels.

The BPDP model, trained either with CIBE (denoted $BPDP_{CIBE}$) or MPPN (denoted $BPDP_{MPPN}$) embeddings, was compared against three other approaches that can be used to detect deviations. The approach developed by Genga et al. (2019) uses subjective logic for deviation detection, relying on the statistical distribution of deviations. Further, the BPDP model is compared against an approach building on decision trees, i.e., CatBoost, a machine learning technique that works well when few samples are available or the dataset is imbalanced. For CatBoost, the same loss weighting is used as for BPDP. Finally, the MPPN is utilized to predict the suffixes for trace prefixes, i.e., obtaining full traces, that are aligned to the to-be process model to detect non-conforming behavior. Section 4.2.2.3 gives more details on this procedure.

The performance of the deviation prediction models is assessed using precision, recall, and AUC_{ROC} as commonly done for tasks with strong class imbalance. Precision and recall are calculated separately for the deviation (*Dev*) and non-deviating class (*No Dev*) to get a detailed quality assessment. As the deviations are more important than compliant behavior, a high *Dev* recall is particularly interesting. The AUC_{ROC} quantifies how well the prediction model can distinguish between deviating and non-deviating behavior by comparing the true and false-positive rates. A value above 0.7 is considered acceptable, and above 0.8 is excellent.

Results Table 15 shows the result of the experiment. BPDP performs best across all tasks compared to the baseline models. Concerning the embeddings, the version of BPDP using CIBE encoding performs better in all but one log than the version that utilizes the pre-trained embeddings of the MPPN. Nevertheless, the AUC_{ROC} values of BPDP with MPPN embeddings are also respectable, given that the embed-

Table 15: Precision, recall, and AUC_{ROC} for all event logs. Boldness indicates the best score for this event log (Grohs et al. 2023).

Log	Baselines						BPDP			
	Genga et. al.		CatBoost		Suffix Prediction		$BPDP_{CIBE}$		$BPDP_{MPPN}$	
	Dev	No Dev	Dev	No Dev	Dev	No Dev	Dev	No Dev	Dev	No Dev
BPIC 12A	0.2136	0.9214	0.1694	0.9622	0.3766	0.9256	0.1620	0.9669	0.1405	0.9480
	0.0678	0.9654	0.7110	0.6797	0.0967	0.9895	**0.8084**	0.6563	0.6636	0.5733
	0.5166		0.6954		0.5431		**0.7324**		0.6185	
BPIC 12O	0.1462	0.8663	0.2277	0.9566	0.3282	0.8811	0.2019	0.9625	0.1798	0.9652
	0.1340	0.8403	0.6034	0.7148	0.2128	0.8504	0.7981	0.5348	**0.8089**	0.4588
	0.4872		0.6591		0.5316		**0.6665**		0.6339	
BPIC 20 Dom. Dec.	0.1961	0.7318	0.4035	0.9977	0.1934	0.9964	0.1401	0.9982	0.0314	0.9980
	0.2040	0.7205	0.5110	0.9930	0.2900	0.9868	**0.7619**	0.8897	0.6459	0.7876
	0.6372		0.7511		0.7023		**0.8258**		0.7168	
BPIC 20 Int. Dec.	0.1738	0.8823	0.3648	0.9955	0.1096	0.9911	0.0720	0.9938	0.0741	0.9973
	0.1648	0.8684	0.3866	0.9884	0.2652	0.9732	**0.6333**	0.8223	0.6239	0.8456
	0.5796		0.6969		0.6338		0.7270		**0.7348**	
BPIC 20 RfP	0.1480	0.7352	0.4630	0.9979	0.2259	0.9972	0.0402	0.9979	0.0291	0.9985
	0.1244	0.7273	0.3967	0.9965	0.2064	0.9908	**0.6888**	0.8353	0.6486	0.8180
	0.5762		0.6961		0.6335		**0.7620**		0.7333	
BPIC 20 Prep.	0.1475	0.8705	0.3032	0.9949	0.1373	0.9943	0.0457	0.9969	0.0270	0.9971
	0.1067	0.8629	0.2727	0.9946	0.2447	0.9804	**0.5566**	0.8514	0.5511	0.7664
	0.5521		0.6333		0.6284		**0.7040**		0.6587	
MobIS	0.1211	0.8355	0.1363	0.9296	0.1176	0.9697	0.0993	0.9748	0.0956	0.9971
	0.1245	0.8415	0.2254	0.8907	0.2063	0.9599	**0.7162**	0.5906	0.5644	0.7391
	0.5461		0.5729		0.5958		**0.6534**		0.6518	

dings are learned by predicting the next step, which is a much shorter forecasting problem than predicting whether a deviation will occur somewhere in the future of a trace prefix. The AUC_{ROC} of the BPDP version with CIBE encoding is often only minimally higher. This shows that the embeddings contain a lot of useful information for deviation prediction, enabling similar performance as specialized machine learning models. The better performance of the $BPDP_{MPPN}$ model on BPIC 20 Int. Dec. shows that the learned embeddings sometimes even contain more information than manual encodings, which is an interesting observation.

4.2.2.2 MPPN End-to-End for Deviation and Deviation Pattern Prediction

In a following paper (Grohs et al. 2025), the idea of deviation prediction has been extended to deviation pattern prediction. Often, deviating process behavior does not consist of a single activity but multiple activities being executed differently as defined in the to-be process. Thus, multiple deviation types co-occur together as the result of the same underlying non-compliant behavior and hence have to be predicted simultaneously for the same prefix. In this publication, such co-occurring deviation types are referred to as deviation patterns. The previously developed BPDP model is extended by a second model, called DPP, which predicts deviation patterns based on the output of the individual deviation type prediction model IDP. An extensive model architecture experiment and hyperparameter optimization have been performed to find the best neural network type for the BPDP model. This also includes experimenting of whether BPDP can predict all deviation types at once instead of training a separate model per deviation type.

Table 16: Precision, recall, and AUC_{ROC} for individual deviation prediction (IDP) for all event logs. Boldness indicates best score for this event log (Grohs et al. 2025).

Log		Baselines								BPDP	
		Genga et al.		CatBoost		MPPN		Suffix			
		Dev	No Dev	Dev	No Dev	Dev	No Dev	Dev	No Dev	Dev	No Dev
BPIC12A	**Prec.**	0.2136	0.9214	0.1694	0.9622	0.1040	0.9869	0.3766	0.9256	0.1620	0.9669
	Rec.	0.0678	0.9654	0.7110	0.6797	**0.9648**	0.3021	0.0967	0.9895	0.8084	0.6563
	AUC	0.5166		0.6954		0.6335		0.5431		**0.7324**	
BPIC12O	**Prec.**	0.1462	0.8663	0.2277	0.9566	0.1627	0.9901	0.3282	0.8811	0.2019	0.9625
	Rec.	0.1340	0.8403	0.6034	0.7148	**0.8146**	0.4240	0.2128	0.8504	0.7981	0.5348
	AUC	0.4872		0.6591		0.6193		0.5316		**0.6665**	
Dom.	**Prec.**	0.1961	0.7318	0.4035	0.9977	0.0688	0.9974	0.1934	0.9964	0.1401	0.9982
	Rec.	0.2040	0.7205	0.5110	0.9930	0.3182	0.8982	0.2900	0.9868	**0.7619**	0.8897
	AUC	0.6372		0.7511		0.6082		0.7023		**0.8258**	
Int.	**Prec.**	0.1738	0.8823	0.3648	0.9955	0.0864	0.9981	0.1096	0.9911	0.0720	0.9938
	Rec.	0.1648	0.8684	0.3866	0.9884	0.5367	0.8779	0.2652	0.9732	**0.6333**	0.8223
	AUC	0.5796		0.6969		0.7073		0.6338		**0.7270**	
RfP	**Prec.**	0.1480	0.7352	0.4630	0.9979	0.0686	0.9980	0.2259	0.9972	0.0402	0.9979
	Rec.	0.1244	0.7273	0.3967	0.9965	0.3574	0.8878	0.2064	0.9908	**0.6888**	0.8353
	AUC	0.5762		0.6961		0.6226		0.6335		**0.7620**	
Prep.	**Prec.**	0.1475	0.8705	0.3032	0.9949	0.0811	0.9961	0.1373	0.9943	0.0457	0.9969
	Rec.	0.1067	0.8629	0.2727	0.9946	0.2851	0.9284	0.2447	0.9804	**0.5566**	0.8514
	AUC	0.5521		0.6333		0.6067		0.6284		**0.7040**	
MobIS	**Prec.**	0.1211	0.8355	0.1363	0.9296	0.1555	0.9975	0.1176	0.9697	0.0993	0.9748
	Rec.	0.1245	0.8415	0.2254	0.8907	0.4421	0.7617	0.2063	0.9599	**0.7162**	0.5906
	AUC	0.5461		0.5729		0.6019		0.5958		**0.6534**	

Additionally, the MPPN has been trained directly on the deviation and deviation pattern prediction task. This allows assessing whether any performance gains can be expected over using the embeddings. For this, the model is pre-trained once per event log and then fine-tuned to simultaneously predict all deviation types (or patterns) by adding the respective number of heads. Compared to BPDP, which trains one model per deviation type, this requires training only one prediction model per event log. The MPPN model performs undersampling and loss weighting per imbalance across the event log. The same event logs and baseline models as in the original publication (Grohs et al. 2023) were used in the experiment.

Results Table 16 shows the recall, precision, and AUC_{ROC} of the MPPN in comparison to the other baseline and the specialized BPDP model called IDP. Throughout the experiments, the BPDP IDP version predicting all deviation types at once (called IDP-collective in Grohs et al. (2025)) has been shown to perform much worse than training a separate model per deviation type. Therefore, only the separate version is used in the following experiments.

Unexpectedly, the performance of the MPPN trained end-to-end did not increase compared to using the MPPN embeddings as reported in Table 13. In contrast, it decreased on average over all event logs. Only for BPIC 2020 Int, an acceptable AUC_{ROC} is reached with MPPN. For the other event logs, lower values are obtained. It has been expected that the performance should increase when training

Table 17: Precision, Recall, and AUC_{ROC} for deviation pattern predictions (DPP) for all event logs. Boldness indicates the best score for this event log (Grohs et al. 2025).

Log		Baselines						DPP	
		CatBoost		MPPN		Suffix			
		Dev	No Dev	Dev	No Dev	Dev	No Dev	Dev	No Dev
BPIC12A	Prec.	0.1716	0.9177	0.1665	0.9531	0.0000	0.9012	0.2927	0.9692
	Rec.	0.3145	0.8342	0.7413	0.5865	0.0000	0.9965	**0.8765**	0.5250
	AUC	0.5743		0.6639		0.4983		**0.7507**	
BPIC12O	Prec.	0.2120	0.9248	0.1870	0.9807	0.0228	0.8608	0.1859	0.9914
	Rec.	0.4769	0.7840	0.8595	0.6559	0.0551	0.7122	**0.9654**	0.4849
	AUC	0.6304		**0.7577**		0.3836		0.7251	
Dom.	Prec.	0.2327	0.9951	0.0641	0.9989	0.1794	0.9991	0.0543	0.9993
	Rec.	0.1988	0.9966	0.2976	0.9944	**0.6703**	0.9857	0.5490	0.9522
	AUC	0.5977		0.6460		**0.8245**		0.7387	
Int.	Prec.	0.4267	0.9974	0.0926	0.9993	0.0479	0.9979	0.0311	0.9986
	Rec.	0.3736	0.9992	0.5004	0.9609	0.1799	0.9795	**0.6322**	0.8598
	AUC	0.6863		**0.7307**		0.5762		0.7265	
RfP	Prec.	0.2957	0.9963	0.0000	0.9993	0.0743	0.9992	0.0462	0.9996
	Rec.	0.2801	0.9092	0.0000	1.0000	0.2923	0.9905	**0.7292**	0.8997
	AUC	0.6588		0.5000		0.6398		**0.8013**	
Prep.	Prec.	0.2957	0.9963	0.0549	0.9977	0.0435	0.8965	0.0619	0.9973
	Rec.	0.2801	0.9092	0.2507	0.9723	0.1393	0.8875	**0.6881**	0.8668
	AUC	0.6588		0.6115		0.5610		**0.7692**	
MobIS	Prec.	0.2440	0.9539	0.1295	0.9982	0.2576	0.9490	0.1257	0.9724
	Rec.	0.1715	0.9715	**0.6038**	0.7614	0.3440	0.9518	0.5458	0.7694
	AUC	0.5715		**0.6861**		0.5119		0.6576	

the MPPN end-to-end. As this did not happen, it has to be expected that predicting all deviation types at once, which are imbalanced irregularly, leads to decreasing performance. For instance, as shown in Table 14, some event logs contain up to 46 deviation types that have between a single to a thousand occurrences. Since the MPPN is trained to predict all 46 irregularly imbalanced types at once, it might fail to optimize for all of them, leading to a low AUC_{ROC}.

For deviation pattern prediction, the MPPN model performs better in comparison to the specialized DPP model as shown in Table 17. On two event logs, the MPPN reached an acceptable AUC_{ROC} while it performed best on three event logs. As there are fewer deviation patterns as deviation types, the lower number of prediction targets might lead to a better performance in this task.

In conclusion, the MPPN is able to predict deviations and deviation patterns. A high number of deviation (patterns) overstrains the model. Nevertheless, the results are remarkable given that the model predicts all deviation and deviation types at once and does not need a separate model per deviation (pattern) as the specialized IDP and DPP models.

4.2.2.3 MPPN Suffix Prediction

For deviation prediction, the MPPN has also been used to predict whole suffixes of trace prefixes that are aligned for deviation "prediction". For this, no deviation labels and no training phase are required. Instead, the MPPN predicts suffixes for trace prefixes, which the model can do after pre-training. Given its multi-perspective nature and pre-training objective, the MPPN predicts all event attributes of the next event given trace prefixes. The predicted event attributes can be appended to the existing trace prefix, which is given to the MPPP again. This allows for iteratively predicting next events until the whole suffix is predicted, i.e., until the end-activity according to the to-be process model is predicted. The idea behind this procedure is that the model predicts different continuations for deviating than for non-deviating traces, as it has inherently learned how deviating traces differ from non-deviating ones through pre-training.

Results Results of this procedure are reported in Table 16 for individual deviation prediction and in Table 17 for deviation pattern prediction. Given the high *No Dev* precision and recall values reported, the predicted suffixes are accurate in the sense that the MPPN performs very well in predicting the normative process behavior, i.e., non-deviating process behavior. Since only a few traces contain deviating behavior, it is no surprise that the model learned the compliant behavior. On the downside, the model reaches very low precision and recall for the *Dev* class, although undersampling and loss weighting are applied. Only for a few event logs, the MPPN could reach values better than random guessing. This indicates that the model was not able to reconstruct non-deviating behavior. However, for some event logs, like for BPIC 20 Dom, the suffix prediction approach performed surprisingly well. For this event log, the model was the best-performing model across all approaches and performed even better than DPP. Thus, the model must also have learned how to predict non-compliant behavior. This is encouraging for further research into the suffix prediction power of the MPPN. For instance, whether the model can predict other deviation types than the ones obtained through alignment. This would potentially allow for detecting other deviation types that are yet not present in the event log.

4.3 Unsupervised Trace Classifications, Clustering, and Retrieval

Grouping events and traces can help to separate and organize them in semantic or task-specific ways. This section presents how *PRMs* cluster and separate events and traces, and how this can be used for solving downstream tasks. In contrast to the previous section, all approaches presented in this section work unsupervised, i.e., do not require expert labels for the downstream task to solve. Applications on compliance classification (Baier et al. 2020), trace clustering and retrieval (Pfeiffer et al. 2021), anomaly and conformance classification (Lahann et al. 2023), as well as event abstraction (Rebmann et al. 2023), will be shown.

4.3.1 Compliance Classification using S2SA[29]

The S2SA model, introduced in Section 2.2, was trained to reconstruct traces from its internal, latent representation. Since most traces in the MobIS event log follow compliant behavior, it should have learned the compliant behavior of process traces. Therefore, traces can be classified into compliant and non-compliant based on the reconstruction error of the trace. This way, the model can be used to find non-compliant traces unsupervised without defining the normative behavior beforehand.

Figure 13: The reconstruction error per event for all traces in the MobIS event log (Baier et al. 2020).

[29] This subsection is based on Baier, Dunzer, Fettke, Houy, Matzner, Pfeiffer, Rehse, Scheid, Stephan and Stierle (2020).

After training the S2SA model, all traces in the log are given to the model, and the reconstruction error per trace is computed. The reconstruction error was normalized by the trace length to compensate the higher reconstruction error of longer traces. A trace was marked as non-compliant if the normalized reconstruction error exceeded 1.5 times the average reconstruction error across all traces. In total, 1,500 traces were marked as non-compliant. To validate whether those traces are valid compliance issues, they were compared to those identified manually beforehand, i.e., by the rule-based analysis in Table 18.

Table 18: Traces identified through manual analysis in comparison to the S2SA (Baier et al. 2020).

Rule	# of traces found manually	# of traces found by S2SA	Percentage
1	2	0	0.00%
2	14	7	50.00%
3	45	10	22.20%
4	246	45	18.29%

Results Figure 13 plots the normalized reconstruction error of all traces, i.e., the reconstruction error per event. Each point represents one trace, while longer traces have darker shades. The non-compliant traces found by the manual analysis are marked with crosses in different colors. Table 18 shows how many of the traces found by manual analysis to be non-compliant have also been found by the S2SA approach.

Unfortunately, the S2SA approach identified only a few traces that were identified manually. It missed many non-compliant traces, indicating a low recall, which is undesired for compliance checking. Also, many traces marked as non-compliant (1,500 in total) by the S2SA are compliant in reality, resulting in a low precision. Nevertheless, 50% of the traces that violate the four-eye principle (rule 2) are also found by S2SA. Rules 3 and 4 deal with the cost and timely reporting of travel expenses, which require checking attributes the S2SA is not considering. Therefore, it is not expected that the model would be able to classify them correctly.

A deeper analysis of the results indicated that the S2SA model tends to assign higher reconstruction errors for longer traces or traces that follow an uncommon variant. A longer trace does not indicate non-compliant behavior. Similarly, traces that follow infrequent variants resemble behavior that is not necessarily non-compliant. As already observed in Subsection 3.2.1, this does not resemble natural or semantic clustering in the way required for this task. From a machine learning point of view, infrequent behavior can be described as an anomaly. However, anomalous behavior resulting from infrequent traces does not signify non-compliance. To improve unsupervised compliance checking, the model architecture and training must be improved to account for these characteristics of process traces.

4.3.2 Trace Retrieval using MPPN [30]

The internal representation of traces learned by the MPPN forms natural clusters as shown in Subsection 3.2.2. Each cluster contains traces that are similar to each other. However, as indicated by different colors, the traces in the clusters do not strictly follow the same control flow. Instead, they are of different variants. While the trace clusters have so far only been inspected visually, the representation space can also be used to find traces that are "semantically" similar.

To find such traces, the case-based case retrieval task has been formulated and demonstrated with the MPPN. It allows retrieving cases that are similar in multiple perspectives and dimensions to a query case. Given a query case c_{query}, the task is to generate a set of cases C such that all cases in this set C have similar characteristics to c_{query}. In classical process mining, this requires filtering the log for the characteristics of the trace that are of interest, e.g., to filter for traces that have the same control flow and were performed by the same resource. However, if another resource frequently performs this variant or the resource changes its name, filtering will not show those traces, even though they have the same variant. If one activity in a loop is executed more often than before, it is also not found, as it resembles a different variant, but technically only differs by the activities in the loop. Similarly, for attributes that are numerical or temporal, filtering is complicated. Using a vector-space model as the MPPN could overcome these limitations, as traces with similar characteristics are close to each other in their internal representation space R^n. This case-based case retrieval using the MPPN would allow the retrieval of cases similar to a query case in multiple perspectives and dimensions, and beyond the strict similarity checks used in filters.

After pre-training the MPPN, the representations r of each trace in the event log are computed through inference. Each r is a feature vector of size 128, resulting from the output of the *NN2*. Two feature vectors can be compared by their cosine similarity. The higher the cosine similarity, the more similar the traces are. Given a query trace c_{query}, the set of feature vectors is searched for other traces with a small cosine distance. The top 5 most similar traces of C are returned.

Results Figure 14 shows the internal representation of traces in the MPPN after PCA (see Subsection 3.2.2 for more details) and Table 19 the retrieved traces for 4 example query traces. The query traces are also marked in the PCA plot. *FV* distance denotes the cosine distance between the traces, *DLD* the Damerau-Levenshtein distance computed on the activities of the control flow, and the other three columns the mean absolute error (*MAE*) for the three perspectives *cost*, *travel_start* and *timestamp*.

[30] This subsection is based on Pfeiffer, Lahann and Fettke (2021).

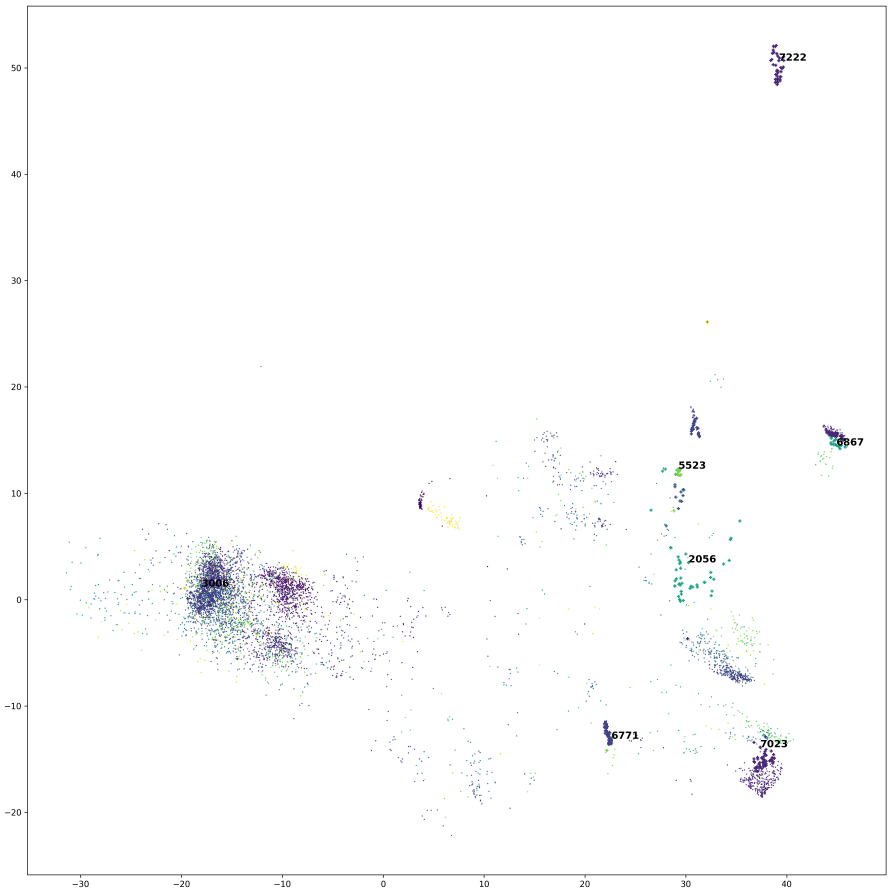


Figure 14: Visualization of the representation space learned by the MPPN on all MobIS cases. Different colors indicate different control-flow variants (Pfeiffer et al. 2021).

As one can see, the distance in control flow (DLD) is very small, indicating that the control flow is the most decisive feature present in the representation space. The retrieved traces are also very similar in the other perspectives, explaining the clusters:

- Traces retrieved for 3006 all follow the most frequent variant.
- The retrieved traces for query case 5523 have different control flow variants but start and end with the same activities. They all have costs below 100.
- For query case 2056, the retrieved traces loop through the same activity but with a different frequency.
- Traces retrieved for case 7222 consist of the first two activities, are performed around the same date, and with costs around 200.

This shows that the representations of the MPPN can be used for case-based case retrieval, considering the context of traces and other process characteristics.

Table 19: Similarities in the perspectives of the retrieved cases (Pfeiffer et al. 2021).

c_{query}	$\hat{C}$ ID	FV distance	DLD	MAE *timestamp*	MAE *cost*	MAE *travel_start*
5523	5511	0.00411	0	101.66	240	0.53
	5613	0.00479	0	154.82	154	6.47
	5036	0.01665	5	1307.83	244	28.53
	5911	0.01755	8	1004.02	203	24.47
	6034	0.01937	8	917.40	253	31.93
	5980	0.02088	8	933.96	237	28.55
	5868	0.02115	8	1045.91	69	21.55
2056	4819	0.01388	0	2066.35	49	174.00
	4960	0.02068	5	3587.52	218	181.33
	4765	0.02295	5	729.69	253	169.15
	4497	0.02340	5	632.51	217	153.00
	4715	0.02428	5	717.51	263	167.00
	5044	0.02453	5	847.96	375	188.00
	4657	0.02465	5	689.45	233	162.39
7222	7109	0.00006	0	14.71	97	7.35
	7092	0.00006	0	16.86	94	8.42
	7073	0.00012	0	18.01	77	9.00
	7090	0.00015	0	17.10	24	8.54
	7133	0.00016	0	11.72	32	5.86
	7231	0.00017	0	0.54	100	0.27
	7052	0.00021	0	18.01	41	9.00
3006	3227	0.00048	0	55.97	392	153.00
	2403	0.00105	0	164.74	1123	0.00
	2624	0.00118	0	54.72	501	30.00
	3748	0.0012	0	206.65	662	153.00
	2859	0.00123	0	103.17	629	38.00
	2861	0.0014	0	89.77	474	52.00
	2116	0.00153	0	287.39	250	40.00

4.3.3 Anomaly and Conformance Classification[31]

Anomaly detection in business process analysis deals with automatically detecting deviating process instances in event logs, i.e., in historical traces, instead of predicting them as done in Subsection 4.2.2. Such anomalies can be introduced by inconsistencies in process executions or bottlenecks, showing optimization opportunities for business processes. In process mining, conformance checking is typically applied to detect anomalies. However, this requires a to-be process model beforehand to compare and evaluate the traces against.

Another approach to detecting anomalies is through the use of machine learning models. Such approaches do not need a process model or other prior knowledge about the process. Instead, they aim to learn the normative process behavior by predicting next events. If the predicted and actual next event found in the trace differ by a certain margin, an anomalous event is found. This also means that the anomaly can only be detected in the moment it occurs.

[31] This subsection is based on Lahann, Pfeiffer and Fettke (2023).

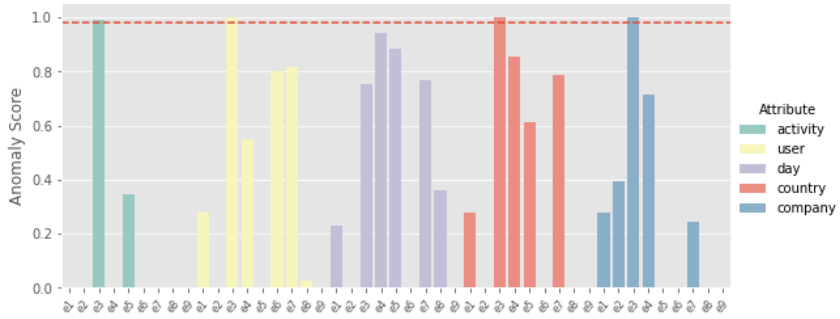


Figure 15: Illustration of the anomaly scores of a case that resembles a skip sequence anomaly. For the 3rd predicted event, the threshold was exceeded for 4 out of 5 attributes (Lahann et al. 2023).

Anomalies are broadly defined as any deviation from the normative behavior. In contrast, conformance checking aims to identify traces that do not fit a to-be process model. Typically, the behavior that does not fit is identified as a pattern of activities. This means that deviating behavior is defined with respect to the to-be process, limiting what is considered deviating. It also requires that all normative behavior is defined. If some wanted behavior has not been modeled, conformance checking might fail to identify the respective deviations from it. The publication aims to detect any anomalous behavior in traces using an LSTM-based process prediction model. Besides any other deviating behavior, this includes non-conforming traces that could be identified with conformance checking.

For this, a process prediction approach optimized for anomaly detection called DAPNN, which consists of an LSTM-based prediction model and an anomaly classification component, has been introduced. The LSTM model is similar to the one proposed in Nolle et al. (2019), featuring an individual LSTM and embedding block per event log perspective. Given a window of a trace as input, i.e., a sequence of events $\langle e_i, \ldots, e_j \rangle$, each block predicts one attribute value of the next event. The model is trained for up to 25 epochs using early stopping.

After training, the model is used for anomaly classification. All trace windows are fed through the DAPNN model to obtain the predicted next attribute probability distributions. A scoring function computes an anomaly score per trace by comparing the predicted probability of next attributes with the probability of the actual next attribute. If at least one anomaly score, i.e., of one attribute value, in the trace exceeds a certain threshold, the trace is marked as anomalous. This means that if one attribute differs between the predicted and actual values, the trace is marked as anomalous. Figure 15 illustrates a trace window with nine events and five perspectives where the anomaly score of the third event exceeds the threshold in 4 out of 5 perspectives.

Different techniques to set the threshold have been tested and are compared in the experiments. Note that the version *DAPNN Best* is not applicable in practice as it sets the threshold based on the labels in the test set. It serves as the maximal achievable performance to reach with the prediction model.

Table 20: Data sets of experiment 1 (Lahann et al. 2023).

	# Logs	# Cases	# Activities	# Events	# Anomalies
PDC 2020 Train	192	1000	16-38	8867-70106	0 / ∼ 200
PDC 2020 Test	192	1000	16-38	8764-68706	412-515
PDC 2021 Train	480	1000	37-65	9867-32009	0 / ∼ 200
PDC 2021 Test	96	250	35-64	6612-11860	125

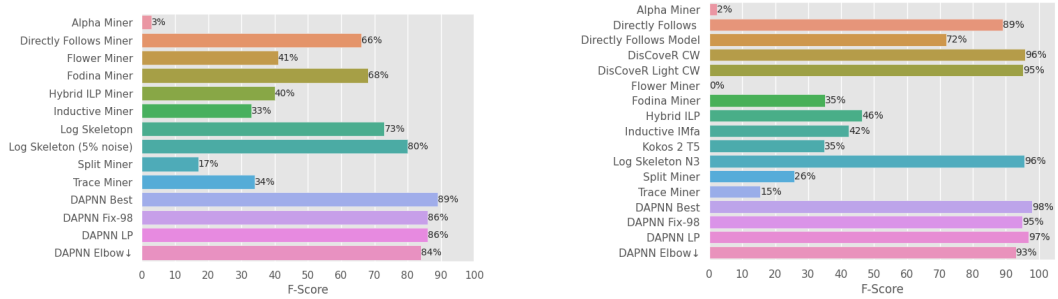
Two types of experiments are conducted. The first experiment tests the approach on the event logs provided by the Process Discovery Contest (PDC) 2020 and 2021 and compares its performance to the approaches that participated. This contest aims to evaluate different process discovery techniques by comparing how accurate they are in classifying traces into fitting and non-fitting ones on a large set of event logs. While it is meant to evaluate process discovery algorithms through a classification task, the DAPNN can also solve it as an anomaly classification task.

Table 20 shows the statistics of the PDC event logs. To replicate the contest accurately, the DAPNN model is trained on the training logs and tested on the test logs. The performance is measured with the F-score, an adapted version of the F1-score.

Table 21: Data sets of experiment 2 (Lahann et al. 2023).

	# Logs	# Cases	# Activities	# Events	# Attributes	# Anomalies
BPIC12	1	13087	73	289892	0	3927
BPIC13	3	819-7554	11-27	4068-81524	7	162-2257
BPIC15	5	832-1409	417-491	46110-62667	6	232-438
BPIC17	2	31509-42995	17-53	285211-1269176	2	9398-13193
Gigantic	4	5000	152-157	38774-42711	1-4	1499-1553
Huge	4	5000	109	46919-53627	1-4	1416-1479
Large	4	5000	85	61789-67524	1-4	1482-1529
Medium	4	5000	65	38990-41991	1-4	1459-1550
P2p	4	5000	27	48477-53193	1-4	1430-1563
Paper	1	5000	27	66814	1	1466
Small	4	5000	41	53437-56695	1-4	1481-1529
Wide	4	5000	58-69	39678-41910	1-4	1436-1513

The second experiment compares the DAPNN approach to an existing machine-learning-based business process anomaly detection approach developed by Nolle et al. (2019) called BINet. The same authors also generated event logs with different numbers of activities and complexities and included various types of synthetic anomalies in the traces. The characteristics of these event logs are shown in Table 21.



(a) PDC 2020 (b) PDC 2021

Figure 16: Comparison by F-Score of the *DAPNN* approach with existing approaches extracted from the PDC website (Lahann et al. 2023).

Results Figure 16 shows and compares how the DAPNN approach performs using different techniques to set the threshold on the PDC data. On PDC 2020, it performs best across all models, regardless of the threshold technique used. On PDC 2021, it performs best using the *LP* heuristic across all models.

Table 22: Comparison by F1-Score of the *DAPNN* approach with existing unsupervised anomaly detection approaches extracted from Nolle et al. (2019) (Lahann et al. 2023).

	BPIC12	BPIC13	BPIC15	BPIC17	Gigantic	Huge	Large	Medium	P2P	Paper	Small	Wide	Mean
Likelihood	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
OC-SVM	0.545	0.243	0.255	0.351	0.291	0.228	0.237	0.289	0.271	0.486	0.248	0.306	0.312
Naive	0.551	0.209	0.172	0.313	0.34	0.404	0.41	0.387	0.479	0.5	0.49	0.438	0.391
Naive+	0.551	0.209	0.173	0.276	0.383	0.454	0.49	0.439	0.48	0.5	0.488	0.469	0.409
Sampling	0.546	0.207	0.172	0.323	0.446	0.491	0.494	0.465	0.49	0.495	0.492	0.486	0.426
t-STIDE+	0.678	0.319	0.287	0.324	0.406	0.446	0.453	0.429	0.509	0.404	0.531	0.471	0.438
DAE	0.595	0.207	0.0	0.295	0.627	0.703	0.713	0.708	0.708	0.463	0.716	0.697	0.536
Likelihood+	0.625	0.445	0.329	0.399	0.665	0.676	0.622	0.654	0.611	0.656	0.688	0.637	0.584
BINetv2	0.607	0.397	0.375	0.43	0.68	0.704	0.71	0.719	0.768	0.757	0.775	0.733	0.638
BINetv1	0.621	0.398	0.346	0.469	0.711	0.713	0.713	0.734	0.768	0.739	0.772	0.761	0.645
BINetv3	0.664	0.446	0.362	0.489	0.662	0.693	0.692	0.709	0.769	0.791	0.762	0.738	0.648
$DAPNN_{FIX-98}$	0.636	0.425	0.459	0.565	0.735	0.776	0.744	0.789	0.842	0.898	0.847	0.805	0.71
$DAPNN_{AR-0.5}$	0.658	0.443	**0.484**	0.621	0.74	0.776	0.744	0.789	0.842	0.898	0.847	0.805	0.721
$DAPNN_{Elbow\downarrow}$	0.656	0.448	0.465	0.564	0.766	0.84	0.817	0.824	0.932	0.965	0.945	0.887	0.759
$DAPNN_{Elbow\uparrow}$	0.688	0.446	0.461	**0.689**	**0.829**	0.88	0.78	0.859	0.852	0.893	0.931	0.903	0.768
$DAPNN_{LP-Min}$	**0.72**	**0.473**	0.475	0.569	0.813	**0.939**	**0.927**	**0.899**	**0.973**	**0.996**	**0.973**	**0.955**	**0.809**
$DAPNN_{LP-Mean}$	**0.72**	**0.473**	0.475	0.569	0.813	**0.939**	**0.927**	**0.899**	**0.973**	**0.996**	**0.973**	**0.955**	**0.809**
$DAPNN_{LP-Max}$	**0.72**	**0.473**	0.475	0.57	0.813	**0.94**	**0.928**	**0.899**	**0.973**	**0.996**	**0.973**	**0.955**	**0.809**
$DAPNN_{Best}$	0.726	0.618	0.501	0.803	0.964	0.969	0.982	0.98	0.993	1.0	0.995	0.987	0.876

Table 22 shows how the DAPNN approach compares to the BINet and baseline models on the event logs of experiment 2. The DAPNN approach reached the best F1-scores across all examined event logs, with the *LP* heuristic performing the best. Overall, it shows that different anomaly types and unfitting traces in particular can be detected accurately with next-step prediction models. The *LP* heuristic seems to be an accurate metric to separate non-fitting or anomalous traces from those that contain normative behavior. It also shows that machine learning models, particularly neural networks, can accurately solve trace classification in an unsupervised way. Given the results, these models must have learned an accurate representation of the process behavior; an observation that is also made in a later publication (see Subsection 3.3.3 and Pfeiffer et al. (2025)).

4.3.4 Event Abstraction[32]

Events recorded by information systems are often too fine-granular for meaningful analysis. As a result, the process models discovered are overly complex and neither useful nor interpretable by humans. Event abstraction can help to overcome these issues by lifting the low-level events recorded in event logs to high-level activities. Supervised techniques require expert knowledge about the high-level activities upfront, e.g., high-level process models or event patterns, which is often not available in practice. Unsupervised techniques do not require any input about the high-level activities. Instead, they rely on control-flow similarities between low-level events. However, existing techniques do not consider other similarities in events, e.g., temporal relations or interactions between resources.

To overcome these issues, an approach has been developed that groups events based on their context and enables users to inspect the groups, guiding them towards identifying meaningful high-level activities. The adapted MPPN introduced in subsubsection 2.3.2.1 learns event representations that capture contextual dependencies between low-level events. For instance, events performed around the same date or by the same resource. Based on the representations, the approach identifies and suggests groups of events that the user can inspect.

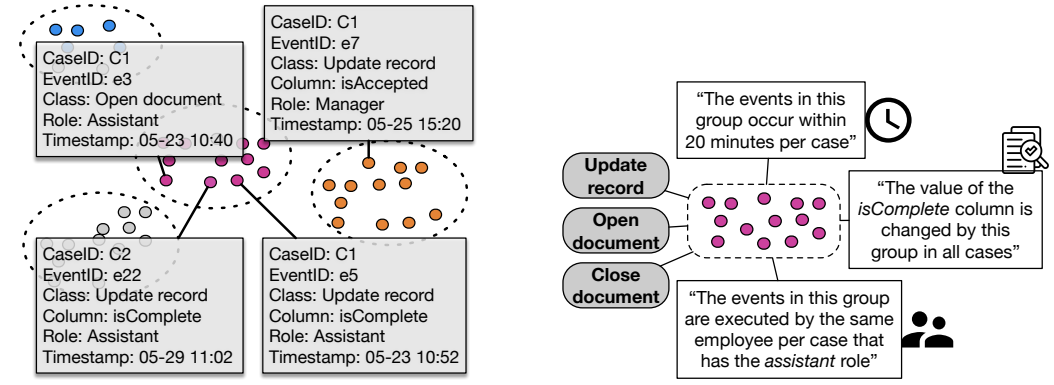
Table 23: A single case of a request handling process recorded on a low level (Rebmann et al. 2023).

CaseID	EventID	Class	Timestamp	Role	Column
C1	e1	Receive email	05-23 07:45	Assistant	
C1	e2	Create record	05-23 09:07	Assistant	
C1	e3	Open document	05-23 10:40	Assistant	
C1	e4	Close document	05-23 10:51	Assistant	
C1	e5	Update record	05-23 10:52	Assistant	isComplete
C1	e6	Open document	05-25 15:03	Manager	
C1	e7	Update record	05-25 15:20	Manager	isAccepted
C1	e8	Close document	05-25 15:23	Manager	
C1	e9	Send email	05-26 10:03	Assistant	

Table 23 illustrates the event abstraction problem, where individual events of a request-handling process are logged in the database as low-level events. One trace consists of different high-level activities, indicated by different colors. From the event names, e.g., *Open document*, one cannot infer which high-level activity has been performed. In reality, each color refers to one high-level activity. The blue events record that a new request has been received, the red events refer to checking the required documents, the brown events refer to a decision about a request, and the grey events represent the notification about the outcome of the request.

The problem of event abstraction in such a situation can be described as creating an $n : m$ mapping between low-level events and high-level activities. Each event with

[32] This subsection is based on Rebmann, Pfeiffer, Fettke and van der Aa (2023).



(a) Multi-perspective event groups (b) Suggestion of a group of events

Figure 17: Illustration of the event abstraction problem (Rebmann et al. 2023).

the same class (i.e., the activity attribute value) can relate to different high-level activities, e.g., *Open document*. Additionally, each high-level activity can relate to any number of low-level events. The context of events is therefore of major importance in separating events with the same class into different high-level activities.

Therefore, the goal is to group events with similar context, i.e., multi-perspective event groups, and make meaningful suggestions to the user as illustrated in Figure 17. The MPPN is primarily used for the former part of learning event embeddings that allow grouping them. Whether events are correlated, e.g., whether they are often performed in a short period or by the same resource, cannot be identified by inspecting individual traces. Rather, the entire event log has to be inspected for this. The MPPN adapted with masked event prediction can be used for this, as it learns event abstraction r that contains information about the context of individual events as well as information about the surrounding events.

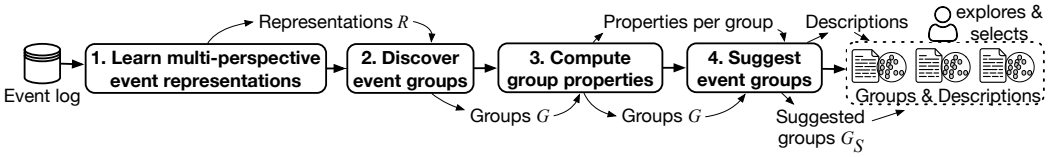
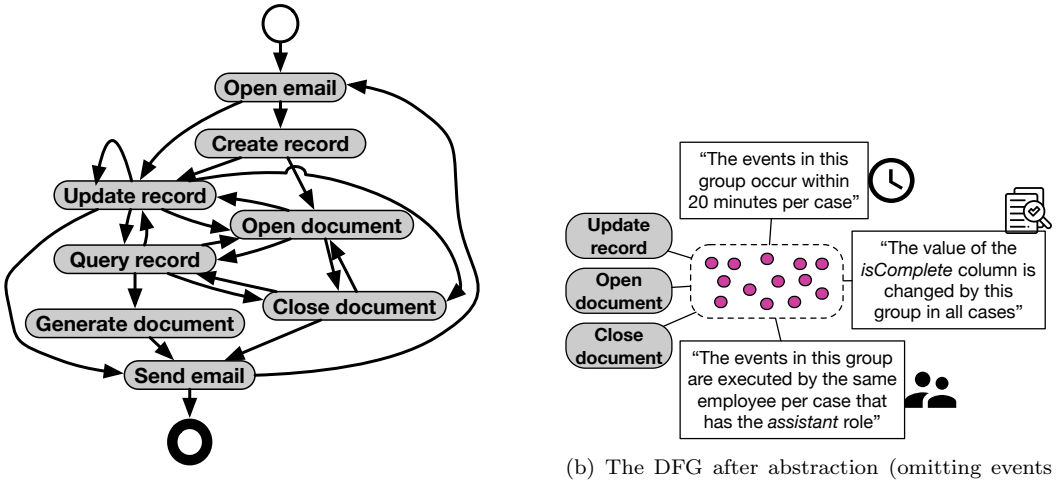


Figure 18: The main steps of the event abstraction approach (Rebmann et al. 2023).

The whole pipeline of the approach is shown in Figure 18. After training and generating event representations for all events in the event log, groups are discovered through clustering using PCA and the k-means algorithm with the elbow method to select an appropriate number of clusters. This clustering gives a grouping G as illustrated in Figure 17 (a). The next step computes various properties for each group, which are used to assess how interesting each group is and to generate textual descriptions for the user, which are communicated in the next step. These properties include the number of event classes, resources, or roles per group, or temporal properties. The last step selects interesting groups based on the group



(a) The low-level DFG before abstraction

(b) The DFG after abstraction (omitting events not included in the suggested groups)

Figure 19: The approach in application (Rebmann et al. 2023).

properties and generates textual descriptions for communication. The final output is a set of interesting event groups suggested for abstraction, which the user can inspect and select. The finally selected event groups provide a mapping from low-level events to high-level activities. To evaluate the approach, a high-level process and a corresponding low-level Petri net have been modeled and simulated that contain $n : m$ relations. After training the MPPN in masked event prediction, which reached very good results, event groups were discovered and suggested based on the learned representations.

Result The approach identified three groups of events that exactly resemble high-level activities. The fourth group suggested represents the whole initial phase of the process, consisting of four high-level activities, which the approach could not discriminate. One reason could be that all events have happened at the beginning of their case, which does not give enough contextual information.

To highlight the usefulness, the high-level activities suggested are mapped to the low-level events in the process model, omitting events not present in the suggested groups. Figure 19 shows the initial DFG of the low-level process and the abstracted high-level DFG. As can be seen, the approach was able to group events so that a meaningful structure became visible, which is not visible in the original DFG. Interestingly, the approach uncovered the hidden choice between doing a thorough examination or not, which becomes not visible from the original model. Note that the labels of the abstracted high-level activities are assigned manually, as this is not yet supported by the approach[33]. Overall, the results indicate the potential of the approach, especially the ability of the MPPN to learn multi-perspective event abstractions with very similar utility as trace representations shown in Subsection 3.2.2.

[33] Assigning meaningful labels to the abstracted high-level activities could be done automatically using LLMs, which were not available when working on this paper.

4.4 Predictions of Future Process Behavior[34]

Almost all predictive techniques for business processes are trace-centric in the sense that they use information from within a single trace only and make predictions about the future behavior of that same single trace. However, in reality, multiple process instances are running simultaneously that influence each other as described and measured with entropy rate estimators in Section 2.5. This inter-trace behavior, which remains concealed if working on single traces only, is an essential characteristic of processes to consider when making predictions. As seen in Figure 7, including information from other traces in the history h should make predictions for single traces more precise, as it gives the prediction model access to inter-trace behavior.

On the prediction target side, there are differences between predicting single-trace behavior and the underlying system S behavior. From a single user point of view, e.g., as a customer who has ordered some goods, knowing how this single process instance will behave in the future is of interest. Also, for analyzing micro-level future trace behavior, it is of interest to monitor and control timely delivery or deviations. However, from a business point of view, aggregated prediction might be more relevant. Such predictions are on the macro level. For instance, predictions that answer how many traces will start and end over the next 7 days, how the utilization of machines will be, whether additional personnel are required, or how many deviations will occur. These prediction tasks require having information from the whole system S.

Since the entropy rates of trace or system sequences (ES or TS) can behave similarly, as investigated in Section 2.5, this has motivated analyzing the predictability of the system, i.e., the original event sequence ES instead of the traces in L. If it is possible to predict the next step in the system sequence ES, i.e., the sequence of events as generated by S, chances are high that the behavior of the system S can be learned; and prediction tasks like the ones described above can be solved.

Three prediction tasks have been formulated in Section 2.5 that differ in the way the history is built and the target activity is sampled. Existing process prediction models use the sequence of events in the trace as the only information for predicting the next step y in the same trace, i.e., they estimate the probability $p(y \mid h)$ with h being the sequence of previous events in the trace and y being the next activity in that trace. In contrast, the prediction models for the underlying system S use all events in a certain period, i.e., windows of ES, as history h to predict the next activity in ES. This means that both the window of previous events and the next activity no longer relate to traces. Instead, the next activity of S is predicted using the previous events in ES.

This task, referred to as PPM_{ES} is considered to be much more difficult than learning to predict next activities from and within traces only, since the data is no longer trace-centric as it is implicitly through the trace ordering for the other two tasks PPM_L and PPM_{TS}.

[34] This section is based on Pfeiffer and Fettke (2024).

Table 24: Performance of different PPM models on the test split (Pfeiffer and Fettke 2024).

		PPM_{ES}			PPM_{TS}			PPM_L		
Cross-entropy (CE) in log_2 (lower is better)										
	H_1^{ES}	RF	LSTM	TF	RF	LSTM	TF	RF	LSTM	TF
BPIC 13 I	1.35	1.27	**1.07**	1.22	1.26	**1.04**	1.27	1.10	**0.97**	0.98
BPIC 17	3.79	2.56	**1.61**	1.63	0.75	0.49	**0.47**	0.61	**0.46**	0.47
BPIC 18	3.57	2.35	**2.11**	2.29	2.30	2.23	**2.13**	0.87	**0.57**	0.62
RFM	2.64	1.50	**1.30**	1.33	0.87	**0.64**	0.66	**0.67**	**0.67**	**0.67**
Accuracy (higher is better)										
BPIC 13 I		69.53%	**70.48%**	69.18%	64.94%	**67.53%**	61.48%	68.82%	**69.67%**	65.59%
BPIC 17		51.58%	**63.22%**	62.86%	86.15%	87.63%	**87.90%**	86.80%	**87.98%**	87.81%
BPIC 18		51.50%	**58.27%**	55.61%	59.76%	61.30%	**65.65%**	81.60%	**85.52%**	84.37%
RFM		65.05%	**66.81%**	65.57%	81.50%	**83.05%**	82.90%	80.91%	**80.92%**	**80.92%**

Results Table 24 shows the results of the prediction experiments. All prediction models reach a lower cross-entropy than the respective unigram entropy rate, indicating that the models learn to estimate the next activities more accurately than statistical guessing. Further, the cross-entropies are also lower than Ngram entropy rates for higher values of n. This can be seen by relating the reached cross-entropy to the Ngram entropy rates shown in the plots[35] in Figure 8. For instance, the LSTM-based PPM_L on BPIC 13 I reaches a cross-entropy of 0.97, which is roughly the value of $n \approx 4$ for the respective entropy rate estimator H_n^L. In comparison, the LSTM-based PPM_{ES} model reaching a cross-entropy of 1.07 is in the area of n between 11 and 12 for H_n^{ES}. This shows that the models learn patterns and long-term statistical effects in the event sequences.

However, for some logs, the cross-entropy is not too much lower than the uni- or bigram entropy rates. This indicates that although the PRM models reach very high accuracies, much more improvement is possible. While a higher accuracy might not be measurable, the cross-entropy can be further reduced. If evaluating with accuracy, one likely cannot see or measure such improvements as the accuracy is limited in the same way as shown in Section 3.3. However, when using cross-entropy, further reduction can be measured. In fact, the improvement in accuracy between PPM_L on BPIC 13 I and PPM_{ES} is very low while the reduction in cross-entropy in relation to the entropy indicates that the model PPM_{ES} has learned the statistical distribution of activities much better than PPM_L.

Comparing the different prediction tasks, making predictions on L, i.e., trace-centric, is the easiest, which is unsurprising as its entropy rate is the lowest. Further, this task is tackled most frequently when making predictions for business processes. Estimating the next activity on TS is, for most event logs, only a little more difficult or even easier for the Road Fine Management (RFM) event log. On ES, the estimations are more difficult, which aligns with the results from the entropy rate analysis.

[35] Note that this procedure has not been explicitly shown in the paper.

Interestingly, for BPIC 2013 I, the cross-entropy on ES is only slightly higher than on TS or L, indicating that next activity predictions are similarly difficult.

Given that predictions on traces within L are possible, as extensively shown in this thesis, and the observation that the entropy rates do not differ too much between the different event orderings, this validates that making predictions about S is feasible. It also indicates that learning the behavior of S using ES might be possible. These predictability experiments are only a feasibility study, and the prediction model PPM_{ES} is not adequately adapted for this task. For instance, as the *caseID* is removed from the events in ES, as commonly done for process prediction tasks, the model has no information to learn that there are traces at all. For the model, ES is a single sequence of events. In reality, the events belong to multiple process instances. Therefore, this information should be encoded somehow for the model, e.g., by enumerating the traces in the window with an artificial *ID* without exposing the actual *caseID*. Doing so requires new prediction model architectures and training objectives, triggering a new design cycle on RQ 1 again.

As predictions about the system S seem plausible, future prediction models should focus on learning different future characteristics of single traces or the whole process. For instance, how many traces will end within the next couple of days, how many will start, and how (and which) will deviate. Further, generative *PRMs* could predict the whole continuation of the system S just like suffix prediction for traces. Given the more information available in ES compared to traces in L, more advanced and helpful real-life prediction tasks can be formulated.

4.5 Summary and Conclusion

This chapter has summarized how predictive techniques, building on *PRMs*, can be used for solving real-world BPM tasks on event log data. The diversity of the tasks solved shows that such techniques can be used for a large variety of problems that are of interest in practice. Tasks like next step and outcome prediction, trace and event retrieval, or anomaly detection can be solved without expert labels. For more complex problems, expert labels are required to fine-tune the pre-trained models on the specific task, e.g., deviation prediction.

The MPPN has been demonstrated on five different types of tasks - next step and outcome prediction in Section 4.2, deviation and deviation pattern prediction in Subsection 4.2.2, trace clustering and retrieval in Section 4.3, and event clustering for abstraction in Subsection 4.3.4. As it outputs softmax values, it is also applicable in the developed pipeline for anomaly detection and trace classification shown in Subsection 4.3.3. The MPPN has been shown to perform very well on these tasks, outperforming existing models or reaching respectable performance compared to task-specific models. Given that the model has been developed as a general-purpose representation learning model, its wide range of applications and the good results are remarkable. Altogether, the array of successful applications finally proves that the concept of a *PRM* works in practice.

New application avenues emerge through learning the system S based on the original event sequence ES instead of trying to learn process behavior from the traces in L. As shown, this task is feasible and enhances existing process prediction techniques with inter-trace information. This allows solving macro-level prediction tasks on the underlying system S in addition to trace-centric, micro-level prediction tasks for which process prediction models are currently used. Developing such models should be researched more extensively in the future to enable richer and more versatile *PRMs*.

Chapter 5

Conclusion

5.1 Summary

This doctoral thesis has introduced the problem of business process representation learning and the concept of process representation models *PRMs* that learn representations r of different concepts in event logs L, such as events, traces, and multi-trace behavior, for solving real-world BPM tasks. In the previous chapters, the main contributions of this doctoral thesis have been highlighted per research area. These show that the idea of neural network models learning vector-space representations r of different concepts from event logs is feasible and promising. Different architectures and training procedures for *PRMs* have been introduced in Chapter 2. The accurateness and quality of these representations have been analyzed in Chapter 3. Finally, Chapter 4 has demonstrated various applications of *PRMs* to real-world problems. Collectively, these contributions provide various design principles for realizing *PRMs*, contributing to the overall research question and validating the thesis idea.

As an example of a *PRM*, the MPPN has been developed, analyzed, and demonstrated. Given this model's extensive analysis and testing - intrinsically and extrinsically on various benchmarks and tasks - combined with its versatility and performance throughout those tasks, the MPPN is considered one of the most sophisticated representation learning models for event logs. Thereby, it realizes the concept of *PRMs* to be applicable for many different tasks with comparable or superior performance to existing and task-specific models.

A significant number of ideas and concepts of representation and machine learning have been applied in this thesis to process prediction. These include multi-view learning, originating from computer vision, used in the MPPN; time-series classification with gramian angular fields; and the idea of semantic clustering as done in NLP and computer vision. More general ML concepts include next and masked element predictions used in various ways for the MPPN, e.g., in anomaly and trace classification or for system learning; multi-task learning for the S2SA and the MPPN;

learning to encode and decode traces in the S2SA and the analysis of the resulting embeddings; and the use of entropy and cross-entropy for evaluation. Further, the aim and challenges of generalization in general machine learning have been adapted and applied to process prediction. The critical investigation of the existing evaluation pipeline used for this task and its linking to common evaluation failures in ML has led to a profound understanding of generalization for process prediction tasks.

In the following, the contributions per research area will be summarized, limitations discussed in Section 5.2, and future research avenues presented in Section 5.3.

Summary of Contributions to Research Area 1 Concerning RA 1, two specialized neural network architectures have been proposed and tested, three different training procedures for *PRMs* have been proposed, and two additional tasks dealing with process deviations have been formalized. The following list briefly summarizes the contributions.

- The S2SA autoencoder-type *PRM* has been proposed and characterized as being trained to reconstruct whole traces with a limited set of attributes
- The MPPN model, being an evolution over the S2SA, has been introduced with the training objective to predict the next events' attribute values, able to include any combination of categorical, numerical, and temporal attributes
- The idea, concept, and challenges in developing *PRMs* have been described in two publications
- The MPPN has been adapted for masked event attribute prediction instead of next event prediction, i.e., solving the cloze task instead of the next element prediction task
- The MPPN has been adapted for the newly introduced deviation and deviation pattern prediction task
- Finally, the idea of learning from and making predictions for the underlying event sequence of the system S rather than the traces in the event log L has been conceptualized and formalized as an evolution over the existing process prediction models, allowing for the incorporation of inter-trace behavior

Summary of Contributions to Research Area 2 Contributions to RA 2 include the analysis of the representation spaces of the S2SA and the MPPN, a critical analysis of the next activity prediction task, conceptualization and evaluation of the generalization capabilities of process prediction models, and alternative evaluation procedures using entropy-based measures. The following list briefly summarizes the contributions.

- The representation spaces of the S2SA and MPPN have been analyzed for clusters, with the representations learned by the MPPN being much more organized and showing strong signals of natural clustering

- The evaluation of process prediction models on the next activity prediction task using accuracy has been critically analyzed, with the observation being made that many prefixes have multiple valid continuation options, described as label ambiguity, affecting the reliability of the accuracy metric
- The next activity prediction task has been further analyzed for example leakage and the accuracy limit, indicating that generalization cannot be assessed adequately using the current evaluation procedure
- Based on the previous findings, a concept and framework for generalization assessment in next activity prediction has been developed, building upon the definition of generalization in general machine learning. The experiments on synthetic and real-life event logs indicate that next activity prediction models generalize surprisingly well to unseen process behavior in the next activity prediction task. It also validates that accuracy is unreliable in communicating the capability of the model to learn process behavior, while cross-entropy expresses this capability much better
- *PRMs* trained to accurately predict the next activity for prefixes learn what next activities can follow, i.e., a very accurate representation of process (instance) behavior. Contextual *PRMs* like the MPPN adapt the probabilities for next activities depending on the contextual attributes present in the prefix. Thus, they learn how processes are structured and can generalize to unseen process behavior represented by unseen prefixes. This shows a strong ability to learn intrinsic process representations
- An alternative evaluation procedure based on cross-entropy, being motivated by Ngram entropy rates and the language modeling objective, has been conceptualized and tested for the next activity pre-training task in process prediction

Summary of Contributions to Research Area 3 Finally, several applications from process instance prediction over trace retrieval to multi-trace system predictions have been shown as contributions to RA 3. The promising performance of these applications concludes that *PRMs* are realistic. The following list briefly summarizes the contributions.

- Several predictive applications on process instance level have been demonstrated, including next step and outcome prediction; deviation and deviation pattern prediction with embedding, end-to-end, and suffix prediction approaches, showing strong performance in comparison to other approaches
- Several applications on process instance level have been demonstrated, including compliance classification, trace retrieval, anomaly and trace classification, and event abstraction, showing promising results
- Finally, predictions on the underlying system S that generated the traces show the feasibility of learning inter-trace and multi-trace behavior, enabling predictive applications about the behavior of the underlying system

Summary of Contributions on Representation Learning This research focused on the realization of *PRMs* with contributions to applications for BPM. Nevertheless, many contributions also relate to the theory and practice of representation learning as a research field of machine learning, as discussed in Section 1.4. The following will briefly summarize how this thesis contributes to representation learning by relating the contributions to the priors of representation learning according to Bengio et al. (2013) and Goodfellow et al. (2016).

- Event attribute values in event logs constitute the *multiple explanatory factors* that one aims to learn in representation learning, representing the individual sources of variation in real world processes, with using them for prediction improving the results and thus the quality of the representation as shown with the MPPN (Pfeiffer et al. 2021), its intrinsic evaluation (clustering in Pfeiffer et al. (2021) and generalization in Pfeiffer et al. (2025)), and its performance on downstream tasks, e.g., deviation prediction (Grohs et al. 2025)
- The MPPN has learned a representation of traces that *naturally clusters* them into groups that are semantically similar (manifolds) as shown in Pfeiffer et al. (2021). By changing the training objective, clusters of events with similar properties have been learned (Rebmann et al. 2023)
- This clustering also shows the *hierarchical organization of explanatory factors* where the traces are made up of events, the events are made up of attributes, and the clusters of traces (Pfeiffer et al. 2021) or events (Rebmann et al. 2023) being organized by different attribute value combinations
- *Self-supervised learning* by predicting the attribute values of the next or a missing event has shown to be beneficial for learning representations that contain the explanatory factors. Such representations contain features that can be shared across tasks, as seen by their usefulness for a variety of different tasks like conformance classification (Lahann et al. 2023), deviation prediction (Grohs et al. 2025), or retrieval (Pfeiffer et al. 2021; Rebmann et al. 2023)
- The prior of *temporal/spatial coherence* is one that we argue to be important for process representation learning in the sense that nearby observations tend to be associated. We can assume that events recorded after another or within a short period are associated. This typically holds for traces but might also hold for event sequences, i.e., ES, as shown in Pfeiffer and Fettke (2024).

Insights on the Design of a *PRM* With the overall research question of this thesis being a design goal (see Section 1.4), we briefly summarize what knowledge has been gained on realizing process representation model.

- The contributions confirm that neural network learning process traces as sequences, e.g., sequential neural networks such as LSTM models, are a good fit for event log data if only the control-flow perspective is of interest. Such models achieve solid performance on many tasks, confirming that the control flow contains many explanatory features of processes

- If the event log consists of multiple perspectives, i.e., features context attributes, they should be considered since it leads to performance improvements of the *PRM* and a richer and more realistic internal representation
- Processing event logs featuring multiple perspectives with neural networks is challenging. Specific changes to the architecture of neural networks and their training objective have to be implemented. Further, achieving generalization can become more challenging. Architectures like the MPPN can process and learn multi-perspective event log data very accurately and often with better performance than sequential architectures. However, more work is required on how to make the models generalize to unseen context attribute combinations
- Many techniques from language representation learning can be applied to business process representation learning, such as next element prediction, the cloze task, or the use of embeddings for tasks like retrieval
- Considering the hierarchy of event log data and learning this as events, traces, and multi-trace sequences gives a realistic internal representation of the data
- Evaluating *PRMs* with probabilistic measures such as cross-entropy gives more accurate insights into their performance than using point measures such as accuracy. Especially for the intrinsic assessment, such evaluation procedures should be used instead of or in addition to established point measures
- The learned representations r can be used in different ways. For instance, without modifications for clustering or retrieval. The representations r can also be used as input for another neural network to solve a specific task. Since this network does not need to process the raw input data but instead the feature vector, this allows lightweight task-solving. Further, fine-tuning the whole *PRM* on a specific task is also possible. Thus, designing better *PRMs* learning from the data enables various application possibilities.
- Overall, when learning representations and building architectures, one should consider the *explanatory factor prior*, the *hierarchical prior*, the *semi-supervised prior*, the *shared feature prior*, the *natural clustering prior*, and the *temporal and spatial coherence prior*, which have been shown to be helpful in this thesis. These priors can also help develop ML and representation learning methods for data with similar properties

Finally, the way the research in this thesis has been conducted can be followed by other researchers as a framework for representation learning on event log data, separated into three research areas RA 1, RA 2, and RA 3. It is compatible with design science research, allowing an iterative approach to enhance process representation learning by researching their architecture and training RA 1, process learning capabilities RA 2, and evaluation in application RA 3. Further, it separates the evaluation of *PRMs* into an intrinsic and extrinsic assessment, a novel evaluation approach in process prediction - and probably also for process mining in general. Future research can build on this framework to make progress on business process representation learning.

5.2 Discussion and Limitations

Significant contributions to business process representation learning have been made in this doctoral thesis. They extend the knowledge of how to learn rich representations from event log data as well as predictive methods in process mining. The concept of *PRMs* (see Chapter 2), inspired by language representation models, has been shown to work similarly for event logs. The learned representations accurately represent the concepts found in the data (see Chapter 3), allowing for solving analytical to forward-facing predictive tasks. This ability makes them versatile for problem-solving. Contributions for research area RA 2 notably advanced the understanding of how machine learning methods learn business processes from event log data. For instance, the observations that *PRMs* organize traces in semantic clusters or generalize very well. However, the accuracy seems suboptimal, and using (cross-) entropy for assessment allows a more accurate interpretation of their actual performance. As shown in Chapter 4, one model, trained on a pretraining task like next step prediction, can be used for many downstream tasks with little to no modifications. These tasks include future predictions of single to multiple process instances, anomaly detection, trace and event retrieval, and deviation prediction.

Overall, each contribution has demonstrated individual parts of representation learning on event logs, showing the idea's feasibility. Nevertheless, more research is required on that topic, and results need to be reproduced by other researchers and on different data to validate the concept of process representation models comprehensively. For now, the developed models are less mature than language representation models. In the following, several aspects and limitations of this thesis are discussed.

As this thesis is the first work that defines the idea of representation learning on event log data, no other representation learning model for event logs is known against which the approaches developed in this thesis could be compared. For the same reason, no dedicated representation learning benchmark (suite) covering many different prediction tasks with ground truth labels can be used to evaluate the developed models. The absence of a standardized benchmark suite for representation learning or other representation learning models for event logs prevents an objective positioning of the developed approaches. The development of such a suite has been studied during this PhD. However, the diversity of different process prediction tasks with non-standard definitions and often heterogeneous datasets made this very difficult.

The research conducted in this thesis is also limited by the number and quality of the event logs publicly available. It is a well-known problem in the research community that there is a limited number of event logs that, in addition, were never intended to be used for machine learning or benchmarking. The lack of publicly available data and benchmarks is a problem the whole BPM community is aware of. This has lately been criticized in the keynote by Alexander Serebrenik at BPM 2024 (Marella et al. 2024) with him stating that the repetitive use of the same data[36] creates a "community bias" that can threaten progress in a field - which inevitably also affects

[36] In particular he mentioned the BPIC event logs.

the results in this thesis. The size of the available logs is limited to some thousand to ten thousand traces with a limited set of attributes. This also limits the size of the *PRM* to be trained, as scaling the model should be followed by more data (Hoffmann et al. 2022). Therefore, it is unknown whether the scaling law, which is an essential part of the success of deep learning in recent times (Pearce et al. 2024), also holds for representation learning on event logs. Whether more event data, larger models, and more compute would also lead to more accurate *PRMs* remains open. Similarly, the absence of more real-world event logs from companies containing many more traces with many more attributes limits the insights on the findings' transferability. For instance, how such models would perform on event logs with even more attributes.

Not all aspects of representation learning have been validated. For instance, it was not shown that a better *PRM* leads to better performance on downstream tasks - a finding often made in representation learning on images or text. Also, not all priors for representation learning have been considered when developing *PRMs*, nor tested within this thesis. So far, the models developed do not consider the semantics of event attributes, which is an essential characteristic of processes (Yuan et al. 2024; Rebmann 2024). Considering this type of information or other general priors of representation learning could enhance the existing models. The generalization of representations or *PRMs* across different event logs has intentionally not been researched so far. Therefore, it is not known which feature can be used across event logs. Such models are referred to as foundation models and should be researched in future work. Learning features that can be used for solving different tasks is considered a prior for training models that can generalize across various domains, e.g., from one event log to another.

While the models have been applied to many different applications, as demonstrated in Chapter 4, the number of tasks used is still limited, with more tasks to be solved by *PRMs* to explore. Further, the artifacts have not been validated in the social context outlined by the research method (Wieringa 2014), e.g., through case studies or stakeholder interviews. Although substantial time has been spent obtaining real data from companies willing to test these techniques in practice, this has not worked out so far.

The MPPN has been shown to suffer from different conceptual limitations, as discussed in Section 2.3. Also, it has not been consistently used for all tasks throughout this thesis, like trace classification for anomalies or trace fitting. It is also not the best model across all the tasks it has been used for. For instance, it performs less well in predicting temporal characteristics of running process instances (see Subsection 4.2.1) or in deviation predictions (see Subsection 4.2.2) where the targets are strongly imbalanced. These limitations can be explained by design choices of the MPPN, i.e., in its architecture and training setup. However, they prevent demonstrating a similar success for representation learning as shown with representation models for language, e.g., BERT (Devlin et al. 2019), which performs better than existing models on almost all tasks it has been tested on.

Alternative model designs like graph neural networks have not been included in the thesis. While other researchers have successfully applied such models to BPM tasks, the author of this thesis has had little success in their application for representation learning. As a consequence, the results have not been published. Similarly, reinforcement learning, as another promising learning strategy for representation learning, has not been researched.

In summary, representation learning for event logs has been found to be promising, with first *PRMs* giving promising results. However, the representations do not yet have the same quality as representations for images or text, and their usage has not been researched as extensively as for these modalities.

5.3 Future Research Avenues

Various future research avenues open up by this thesis, driven by the results of this thesis and by the general progress on AI, especially GenAI. In RA 1, the features that should be included in the representations need further conceptualization driven by the general priors of representation learning (Bengio et al. 2013) and specific priors that are important for processes. The transformer architecture, in combination with adapted positional encoding, offers new possibilities for learning representations from traces and multi-trace sequences. However, it has to be researched how such models can effectively process mixtures of categorical, numerical, and temporal attributes to overcome the limitations of the MPPN. Additionally, models of instruction type that perform different prediction tasks based on a specific token (Li and Liang 2021) or prompt in the input could be used to make such models even more versatile without requiring extensive training of the whole model. For instance, given a trace prefix and an instruction token, to predict different future characteristics.

Regarding RA 2, perplexity should be researched intensively. Just as the branching factor of language models (Jurafsky and Martin 2025) relates to the structure of a language, it could relate to the structure of the process for *PRMs*. Further, other forms of generalization should be researched, as well as how to obtain formal process models from the learned representation.

For RA 3, other applications that have so far not been researched should be tested — for instance, simulation. All future research requires more event log data, preferably actual company data, and accompanying use cases. The concept of *PRMs* needs to be further validated through large-scale experiments. Another promising avenue is embedding process representation learning into the field of process science as it allows analytical and predictive tasks on digital trace data (vom Brocke et al. 2024).

Beyond this thesis, a foundation model for business process data is of interest, i.e., a large-scale model that can subsequently be adapted to solve multiple different tasks (Bishop and Bishop 2024). Figure 20 shows a possible research path for such a foundation model along two dimensions. The application dimension shows the ability of such models from performing single and multiple tasks, to simulation, and finally, autonomy, i.e., to performing tasks without human supervision. The

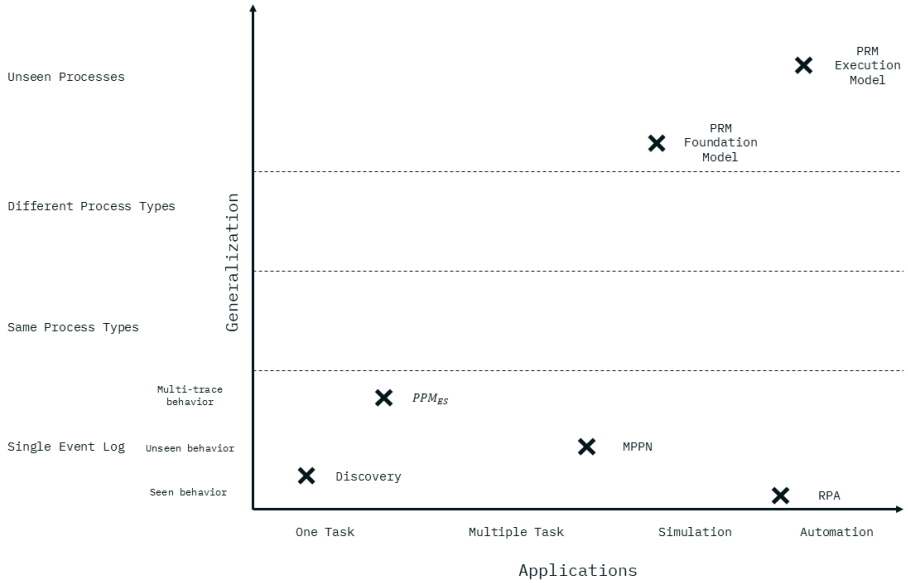


Figure 20: Future research path for *PRMs* (own figure).

generalization dimension signifies how far such models can generalize in a processual perspective and how well they can deal with unseen behavior to unseen processes. Seen behavior requires the least generalization, followed by generalization to unseen and multi-trace behavior for single event logs. Next comes generalization across the same process from different companies to generalization across multiple different process types. Finally, models would need to generalize to unseen processes.

Discovery and RPA are added as two examples of existing techniques. Process models obtained by discovery can be used for a few tasks (van der Aalst 2022) on seen behavior. However, if unseen traces occur or other tasks are to be performed, such models fall short (see, e.g., Tax et al. (2017)). RPA can be used to automate known tasks that have limited variability. The MPPN model can solve multiple tasks and handle a certain degree of unseen behavior, if used on the same process from the same company. Models trained to learn the system behavior of S from ES, i.e., PPM_{ES}, can generalize to multi-trace behavior. So far, they have only been used for one task.

The next step involves developing models that have learned how a certain type of process works, e.g., the purchase-to-pay process. They can generalize to tasks on such a process type from different companies. One step further, models that can generalize beyond a single process type to different process types are to be developed. For instance, they work for different process types from different companies with company-specific fine-tuning to learn context information. Finally, a foundation model for processes is capable of generalizing to unseen processes and can easily be adapted for new tasks. Following, execution models that can also perform (unseen) processes, capable of automating a large variety of processes and tasks, are possible.

Bibliography

Aalst, W. M. P. V. D., Schonenberg, M. H. and Song, M. (2011), 'Time prediction based on process mining', *Information Systems* pp. 450–475.

Abb, L., Pfeiffer, P., Fettke, P. and Rehse, J.-R. (2024), A discussion on generalization in next-activity prediction, *in* 'International Conference on Business Process Management Workshops', pp. 18–30.

Adams, J. N., Park, G., Sergej, L., Schuster, D. and van der Aalst, W. M. P. (2022), A framework for extracting and encoding features from object-centric event data, *in* 'International Conference on Service-Oriented Computing'.

Back, C. O., Debois, S. and Slaats, T. (2019), 'Entropy as a measure of log variability', *Journal on Data Semantics* pp. 129–156.

Baier, S., Dunzer, S., Fettke, P., Houy, C., Matzner, M., Pfeiffer, P., Rehse, J.-R., Scheid, M., Stephan, S. and Stierle, M. (2020), 'The mobis-challenge 2019', *Enterprise Modelling and Information Systems Architectures (EMISAJ)* .

Ben-Shaul, I., Shwartz-Ziv, R., Galanti, T., Dekel, S. and LeCun, Y. (2023), Reverse engineering self-supervised learning, *in* 'Advances in Neural Information Processing Systems', pp. 58324–58345.

Bengio, Y., Courville, A. and Vincent, P. (2013), 'Representation learning: A review and new perspectives', *IEEE transactions on pattern analysis and machine intelligence* pp. 1798–1828.

Bishop, C. M. and Bishop, H. (2024), *Deep Learning Foundations and Concepts*, Springer International Publishing.

Breuker, D., Matzner, M., Delfmann, P. and Becker, J. (2016), 'Comprehensible predictive models for business processes', *MIS quarterly* pp. 1009–1034.

Cabrera, L., Weinzierl, S., Zilker, S. and Matzner, M. (2023), Text-aware predictive process monitoring with contextualized word embeddings, *in* 'International Conference on Business Process Management Workshops', pp. 303–314.

Camargo, M., Dumas, M. and lez Rojas Gonzá Oscar (2019), Learning accurate lstm models of business processes, *in* 'International Conference on Business Process Management', pp. 286–302.

Ceravolo, P., Junior, S. B., Damiani, E. and Aalst, W. V. D. (2023), 'Tailoring machine learning for process mining', *IEEE Access* pp. 24583–24595.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019), Bert: Pre-training of deep bidirectional transformers for language understanding, *in* 'Proceedings of the 2019 Conference of the North', Association for Computational Linguistics, pp. 4171–4186.

Dumas, M., Rosa, M. L., Mendling, J. and Reijers, H. A. (2018), *Fundamentals of Business Process Management*, Springer Berlin Heidelberg.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T. and Beaini, D. (2022), Long range graph benchmark, *in* 'Advances in Neural Information Processing Systems', pp. 22326–22340.

Evermann, J., Rehse, J.-R. and Fettke, P. (2016), A deep learning approach for predicting process behaviour at runtime, *in* 'International Conference on Business Process Management Workshops'.

Evermann, J., Rehse, J.-R. and Fettke, P. (2017), 'Predicting process behaviour using deep learning', *Decision Support Systems* pp. 129–140.

Francescomarino, C. D. and Ghidini, C. (2022), *Predictive Process Monitoring*, pp. 320–346.

Gao, B.-B., Xing, C., Xie, C.-W., Wu, J. and Geng, X. (2017), 'Deep label distribution learning with label ambiguity', *IEEE Transactions on Image Processing* pp. 2825–2838.

Genga, L., Francescomarino, C. D., Ghidini, C. and Zannone, N. (2019), Predicting critical behaviors in business process executions: When evidence counts, *in* 'International Conference on Business Process Management Forum', pp. 72–90.

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press.

Grohs, M., Pfeiffer, P. and Rehse, J.-R. (2023), Business process deviation prediction: Predicting non-conforming process behavior, *in* 'International Conference on Process Mining', pp. 113–120.

Grohs, M., Pfeiffer, P. and Rehse, J.-R. (2025), 'Proactive conformance checking: An approach for predicting deviations in business processes', *Information Systems* p. 102461.

Gunnarsson, B. R., Broucke, S. V. and Weerdt, J. D. (2023), 'A direct data aware lstm neural network architecture for complete remaining trace and runtime prediction', *IEEE Transactions on Services Computing* pp. 2330–2342.

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004), 'Design science in information systems research', *MIS quarterly* pp. 75–105.

Hinton, G. E., Osindero, S. and Teh, Y. W. (2006), 'A fast learning algorithm for deep belief nets', *Neural Computation* pp. 1527–1554.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J. W. and Sifre, L. (2022), Training compute-optimal large language models, *in* 'Advances in Neural Information Processing Systems', Neural information processing systems foundation.

Jurafsky, D. and Martin, J. H. (2025), *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3rd edition. Online manuscript released January 12, 2025. https://web.stanford.edu/ jurafsky/slp3*, 3rd edn.

Klein, S., Lahann, J., Mayer, L., Neu, D., Pfeiffer, P., Rebmann, A., Scheid, M., Willems, B. and Fettke, P. (2020), Business process intelligence challenge 2020: Analysis and evaluation of a travel process, *in* 'Business Process Intelligence Challenge at the International Conference on Process Mining'.

Koninck, P. D., vanden Broucke, S. and Weerdt, J. D. (2018), act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes, *in* 'International Conference on Business Process Management', pp. 305–321.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems'.

Kubat, M. and Matwin, S. (1997), Addressing the curse of imbalanced training sets: One-sided selection, *in* 'International Conference on Machine Learning'.

Lahann, J., Pfeiffer, P. and Fettke, P. (2023), Lstm-based anomaly detection of process instances: Benchmark and tweaks, *in* 'International Conference on Process Mining Workshops', pp. 229–241.

Lecun, Y., Bengio, Y. and Hinton, G. (2015), 'Deep learning', *Nature 2015 521:7553* pp. 436–444.

Leontjeva, A., Conforti, R., Francescomarino, C. D., Dumas, M. and Maggi, F. M. (2016), Complex symbolic sequence encodings for predictive monitoring of business processes, *in* 'International Conference on Business Process Management', pp. 297–313.

Li, X. L. and Liang, P. (2021), 'Prefix-tuning: Optimizing continuous prompts for generation', *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference* pp. 4582–4597.

Liao, T., Taori, R., Raji, I. D. and Schmidt, L. (2021), Are we learning yet? a meta review of evaluation failures across machine learning, *in* 'Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)'.

Marella, A., Resinas, M., Jans, M. and Roseman, M. (2024), *Business Process Management - 22nd International Conference BPM 2024*, Springer Nature Switzerland.

Mehdiyev, N., Mayer, L., Lahann, J. and Fettke, P. (2022), 'Deep learning-based clustering of processes and their visual exploration: An industry 4.0 use case for small, medium-sized enterprises', *Expert Systems* .

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013), 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781* .

Murphy, K. P. (2022), *Probabilistic machine learning: an introduction*, MIT press.

Neu, D. A., Lahann, J. and Fettke, P. (2021), 'A systematic literature review on state-of-the-art deep learning methods for process prediction', *Artificial Intelligence Review* .

Nolle, T., Luettgen, S., Seeliger, A. and Mühlhäuser, M. (2019), 'Binet: Multi-perspective business process anomaly classification', *Information Systems* p. 101458.

Nolle, T., Seeliger, A. and Mühlhäuser, M. (2016), Unsupervised anomaly detection in noisy business process event logs using denoising autoencoders, *in* 'Discovery Science: 19th International Conference, DS 2016', pp. 442–456.

Pasquadibisceglie, V., Appice, A., Castellano, G. and Malerba, D. (2020), Predictive process mining meets computer vision, *in* 'International Conference on Business Process Management', pp. 176–192.

Pearce, T., Rashid, T., Bignell, D., Georgescu, R., Devlin, S. and Hofmann, K. (2024), Scaling laws for pre-training agents and world models, *in* 'arXiv preprint arXiv:2411.04434'.

Peffers, K., Tuunanen, T., Rothenberger, M. A. and Chatterjee, S. (2007), 'A design science research methodology for information systems research', *Journal of Management Information Systems* pp. 45–77.

Pfeiffer, P. (2022), Business process representation learning, *in* 'Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2022 co-located with 20th International Conference on Business Process Management (BPM 2022)'.

Pfeiffer, P. (2023), Towards process representation models for business process management, *in* 'AAAI 2023 Bridge Program on Artificial Intelligence and Business Process Management'.

Pfeiffer, P., Abb, L., Fettke, P. and Rehse, J.-R. (2025), 'Learning from the data to predict the process', *Business & Information Systems Engineering* pp. 1–27.

Pfeiffer, P. and Fettke, P. (2021), 'Applications of rpa in manufacturing', *Robotic Process Automation: Management, Technology, Applications* pp. 315–346.

Pfeiffer, P. and Fettke, P. (2024), Trace vs. time: Entropy analysis and event predictability of traceless event sequencing, *in* 'International Conference on Business Process Management Forum', pp. 72–89.

Pfeiffer, P., Lahann, J. and Fettke, P. (2021), Multivariate business process representation learning utilizing gramian angular fields and convolutional neural networks, *in* 'International Conference on Business Process Management', pp. 327–344.

Pfeiffer, P., Lahann, J. and Fettke, P. (2023), The label ambiguity problem in process prediction, *in* 'International Conference on Business Process Management Workshops', pp. 37–44.

Pfeiffer, P., Sander, H., Fettke, P. and Reisig, W. (2022), Towards a standard process enabling ai-support for safety and conformity of medical devices, *in* 'The International Health Data Workshop (HEDA-2022), co-located with Petri Nets 2022', CEUR-WS.

Rebmann, A. G. (2024), Semantics-aware event data transformation for process mining, PhD thesis.

Rebmann, A., Pfeiffer, P., Fettke, P. and van der Aa, H. (2023), Multi-perspective identification of event groups for event abstraction, *in* 'International Conference on Process Mining Workshops', pp. 31–43.

Rehse, J.-R., Dadashnia, S. and Fettke, P. (2018), 'Business process management for industry 4.0 - three application cases in the dfki-smart-lego-factory', *it - Information Technology* pp. 133–141.

Reijers, H. (2007), 'Case prediction in bpm systems: a research challenge', *Journal of Korean Institute of Industrial Engineers* pp. 1–10.

Ruta, D. and Majeed, B. (2011), Business process forecasting in telecom industry, *in* 'IEEE GCC Conference and Exhibition (GCC)', pp. 389–392.

Scheid, M., Rehse, J.-R., Houy, C. and Fettke, P. (2018), 'Data set for mobis challenge 2019'.

Seeliger, A., Luettgen, S., Nolle, T. and Mühlhäuser, M. (2021), Learning of process representations using recurrent neural networks, *in* 'International Conference on Advanced Information Systems Engineering', pp. 109–124.

Shannon, C. E. (1948), 'A mathematical theory of communication', *The Bell system technical journal* pp. 379–423.

Shannon, C. E. (1951), 'Prediction and entropy of printed english', *Bell system technical journal* pp. 50–64.

Striewe, M., Forell, M., Houy, C., Pfeiffer, P., Schiefer, G., Schüler, S., Soyka, C., Stottrop, T., Ullrich, M., Fettke, P., Loos, P., Oberweis, A. and Schaper, N. (2021), 'Kompetenzorientiertes e-assessment für die grafische, konzeptuelle modellierung', *HMD Praxis der Wirtschaftsinformatik 2021 58:6* pp. 1350–1363.

Tax, N., Verenich, I., Rosa, M. L. and Dumas, M. (2017), Predictive business process monitoring with lstm neural networks, *in* 'International Conference on Advanced Information Systems Engineering', pp. 477–492.

Tschannen, M., Bachem, O. and Lucic, M. (2018), Recent advances in autoencoder-based representation learning, *in* 'Third workshop on Bayesian Deep Learning (NeurIPS 2018)'.

Ullrich, M., Fettke, P., Pfeiffer, P., Schüler, S. and Striewe, M. (2022), 'Workshop zur modellierung in der hochschullehre', pp. 174–175.

Ullrich, M., Forell, M., Houy, C., Pfeiffer, P., Schüler, S., Stottrop, T., Willems, B., Fettke, P. and Oberweis, A. (2021), Platform architecture for the diagram assessment domain, *in* 'Software Engineering (Satellite Events)'.

Ullrich, M., Pfeiffer, P., Schiefer, G., Soyka, C., Stottrop, T., Striewe, M., Fettke, P., Loos, P., Oberweis, A. and Schaper, N. (2022*a*), E-assessment-plattform für die grafische modellierung, *in* '20. Fachtagung Bildungstechnologien (DELFI)', Gesellschaft fur Informatik (GI).

Ullrich, M., Pfeiffer, P., Schiefer, G., Soyka, C., Stottrop, T., Striewe, M., Fettke, P., Loos, P., Oberweis, A. and Schaper, N. (2022*b*), Piloteinsatz einer e-assessment-plattform für die grafische modellierung, *in* '20. Fachtagung Bildungstechnologien (DELFI)', Gesellschaft fur Informatik (GI), pp. 233–234.

van der Aalst, W. M. P. (2013), 'Business process management: A comprehensive survey', *ISRN Software Engineering* pp. 1–37.

van der Aalst, W. M. P. (2016), *Process Mining: Data Science in Action*, 2nd edn, Springer Publishing.

van der Aalst, W. M. P. (2022), *Process Mining: A 360 Degree Overview*, pp. 3–34.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017), Attention is all you need, *in* 'Advances in neural information processing systems', pp. 5998–6008.

vom Brocke, J., van der Aalst, W. M. P., Berente, N., van Dongen, B., Grisold, T., Kremser, W., Mendling, J., Pentland, B. T., Roeglinger, M., Rosemann, M. and Weber, B. (2024), 'Process science: the interdisciplinary study of socio-technical change', *Process Science* p. 1.

Wang, Z. and Oates, T. (2014), Encoding time series as images for visual inspection and classification using tiled convolutional neural networks, *in* 'Workshop at the Twenty-Ninth AAAI conference on Artificial Intelligence'.

Weinzierl, S., Zilker, S., Dunzer, S. and Matzner, M. (2024), 'Machine learning in business process management: A systematic literature review', *Expert Systems with Applications* .

Weske, M. (2019), *Business Process Management*, 3 edn, Springer Berlin Heidelberg.

Wieringa, R. J. (2014), 'Design science methodology: For information systems and software engineering', *Design Science Methodology: For Information Systems and Software Engineering* pp. 1–332.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. and Yu, P. S. (2021), 'A comprehensive survey on graph neural networks', *IEEE Transactions on Neural Networks and Learning Systems* pp. 4–24.

Xie, J., Girshick, R. and Farhadi, A. (2016), Unsupervised deep embedding for clustering analysis, *in* 'Proceedings of The 33rd International Conference on Machine Learning, PMLR'.

Yuan, J., Grigori, D. and van der Aa, H. (2024), Enhancing predictive process monitoring using semantic information, *in* 'International Conference on Process Mining Workshops'.

Ziv, R. S. and Lecun, Y. (2024), 'To compress or not to compress—self-supervised learning and information theory: A review', *Entropy* p. 252.

Part II

Individual Publications

Publication Overview

Table 25 lists all publications contributing to the thesis. In addition, Table 26 lists those publications that have been published during the PhD phase but which did not contribute to the thesis. For each publication, we report the ranking of the publication venue. For conference publications, we report the ranking according to CORE 2023 and the VHB Publication Media Rating 2024. If the rating is listed in brackets, it denotes that the general ranking of the main conference was used for ranking purposes. For instance, if the venue is a workshop that is not included in the ranking.

- Core 2023[1] (CORE) ranks venues from **A*** (flagship conferences) to **C** (sound and satisfactory)
- VHB Publication Media Rating 2024 - Section Wirtschaftsinformatik[2] (VHB) ranks scientific journals and conference proceeding from **A+** (outstanding and world-leading journals/proceedings of outstanding and world-leading conferences) to **D** (other academic journals/proceedings of other academic conferences, workshops/forums)
- For journal publication, the impact factor[3] (IF) of the outlet is reported additionally

In the following, for each individual publication, the page number of the thesis will be shown in the bottom right in a small box to distinguish it from the individual publication's page number. Note that each publication has its own page number in different positions, which does not align with the page numbering of the thesis. Further, the publication number and title is displayed in the footer.

[1] `https://portal.core.edu.au/conf-ranks/`

[2] `https://www.vhbonline.org/fileadmin/vhb/Services/vhb-rating/WI/VHB_Rating_2024_Area_rating_WI.pdf`

[3] Taken from the journal's homepage as of 6. April 2025.

Table 25: Contributing Publications.

#	Ref	Article/Outlet	CORE	IF	VHB
	2020				
P1	Baier, Dunzer, Fettke, Houy, Matzner, Pfeiffer, Rehse, Scheid, Stephan and Stierle (2020)	**The MobIS-Challenge 2019 – A Report on the WI-2019-Workshop on Model-Based Compliance in Information Systems** — *Enterprise Modelling and Information Systems Architectures — International Journal of Conceptual Modeling (EMISAJ)*			C
	2021				
P2	Pfeiffer, Lahann and Fettke (2021)	**Multivariate Business Process Representation Learning Utilizing Gramian Angular Fields and Convolutional Neural Networks** — *International Conference on Business Process Management (BPM) 2021* (LNCS)	A		B
	2022				
P3	Pfeiffer (2022)	**Business Process Representation Learning** — *International Conference on Business Process Management (BPM) 2022 – Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track* (CEURS)	(A)		(B)
	2023				
P4	Pfeiffer, Lahann and Fettke (2023)	**The Label Ambiguity Problem in Process Prediction** — *International Conference on Business Process Management (BPM) 2022 – AI4BPM Workshop* (LNBIP)	(A)		C
P5	Lahann, Pfeiffer and Fettke (2023)	**LSTM-based Anomaly Detection of Process Instances: Benchmark and Tweaks** — *International Conference on Process Mining (ICPM) 2022 – ML4BPM Workshop* (LNBIP)	(B)		(B)

#	Ref	Article/Outlet	CORE	IF	VHB
P6	Rebmann, Pfeiffer, Fettke and van der Aa (2023)	**Multi-perspective Identification of Event Groups for Event Abstraction** — *International Conference on Process Mining (ICPM) 2022 – EDBA Workshop* (LNBIP)	(B)		(B)
P7	Pfeiffer (2023)	**Towards Process Representation Models for Business Process Management** — *AAAI Bridge Program: Artificial Intelligence and Business Process Management (AI4BPM)*	(A*)		(A)
P8	Grohs, Pfeiffer and Rehse (2023)	**Business Process Deviation Prediction** — *International Conference on Process Mining (ICPM) 2023* (IEEE)	B		B
	2024				
P9	Abb, Pfeiffer, Fettke and Rehse (2024)	**A Discussion on Generalization in Next-Activity Prediction** — *International Conference on Business Process Management (BPM) 2023 – AI4BPM Workshop* (LNBIP)	(A)		C
P10	Pfeiffer and Fettke (2024)	**Trace vs. Time: Entropy Analysis and Event Predictability of Traceless Event Sequencing** — *International Conference on Business Process Management (BPM) 2024 – Forum* (LNBIP)	(A)		C
	2025				
P11	Grohs, Pfeiffer and Rehse (2025)	**Proactive Conformance Checking: An Approach for Predicting Deviations in Business Processes** — *Information Systems (IS)*		3.0	B
P12	Pfeiffer, Abb, Fettke and Rehse (2025)	**Learning from the Data to Predict the Process: Generalization Capabilities of Next Activity Prediction Algorithms** — *Business & Information Systems Engineering (BISE)*		7.9	B

Table 26: Other publications not contributing to the thesis ordered by date.

#	Ref	Article/Outlet	CORE	IF	VHB
13	Klein, Lahann, Mayer, Neu, Pfeiffer, Rebmann, Scheid, Willems and Fettke (2020)	**Business Process Intelligence Challenge 2020: Analysis and evaluation of a travel process** — *BPI Challenge in conjunction with the International Conference on Process Mining (ICPM) 2020*	(B)		(B)
14	Ullrich, Forell, Houy, Pfeiffer, Schüler, Stottrop, Willems, Fettke and Oberweis (2021)	**Platform Architecture for the Diagram Assessment Domain** — *Software Engineering (SE) 2021 - Satellite Events*			
15	Striewe, Forell, Houy, Pfeiffer, Schiefer, Schüler, Soyka, Stottrop, Ullrich, Fettke, Loos, Oberweis and Schaper (2021)	**Competence-oriented E-assessment of Graphical, Conceptual Modelling** — *HMD Praxis der Wirtschaftsinformatik*			C
16	Pfeiffer and Fettke (2021)	**Applications of RPA in manufacturing** — *Robotic Process Automation, Management, Technology, Applications* (De Gruyter STEM)			
17	Ullrich, Fettke, Pfeiffer, Schüler and Striewe (2022)	**Workshop zur Modellierung in der Hochschullehre** — *Modellierung 2022 – Satellite Events*			

#	Ref	Article/Outlet	CORE	IF	VHB
18	Ullrich, Pfeiffer, Schiefer, Soyka, Stottrop, Striewe, Fettke, Loos, Oberweis and Schaper (2022*a*)	**E-Assessment-Plattform für die grafische Modellierung** — *20. Fachtagung Bildungstechnologien (DELFI) 2020* (LNI)			
19	Ullrich, Pfeiffer, Schiefer, Soyka, Stottrop, Striewe, Fettke, Loos, Oberweis and Schaper (2022*b*)	**Piloteinsatz einer E-Assessment-Plattform für die grafische Modellierung** — *20. Fachtagung Bildungstechnologien (DELFI) 2020* (LNI)			
20	Pfeiffer, Sander, Fettke and Reisig (2022)	**Towards a Standard Process enabling AI-support for Safety and Conformity of Medical Devices** — *The International Health Data Workshop (HEDA) in conjunction with International Conference on Application and Theory of Petri Nets and Concurrency 2022* (CEURS)	(B)		

Paper 1

The MobIS-Challenge 2019 — A Report on the WI-2019-Workshop on Model-Based Compliance in Information Systems

Authors	Stephan Baier, Sebastian Dunzer, Peter Fettke, Constantin Houy, Martin Matzner, Peter Pfeiffer, Jana-Rebecca Rehse, Martin Scheid, Sebastian Stephan, Matthias Stierle, Brian Willems

Year	2020
Published in	*Enterprise Modelling and Information Systems Architectures – International Journal of Conceptual Modeling (EMISAJ)*

Paper 2

Multivariate Business Process Representation Learning Utilizing Gramian Angular Fields and Convolutional Neural Networks

Authors	Peter Pfeiffer, Johannes Lahann, Peter Fettke
Year	2021
Published in	*International Conference on Business Process Management (BPM) 2021*, Lecture Notes in Computer Science (LNCS) volume 12875

Paper 3

Business Process Representation Learning

Author	Peter Pfeiffer

Year	2022
Published in	*International Conference on Business Process Management (BPM) 2022 – Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track*, CEUR Workshop Proceedings volume 3216

Paper 4

The Label Ambiguity Problem in Process Prediction

Authors	Peter Pfeiffer, Johannes Lahann, Peter Fettke
Year	2023
Published in	*International Conference on Business Process Management (BPM) 2022 – AI4BPM Workshop*, Lecture Notes in Business Information Processing (LNBIP) volume 460

Paper 5

LSTM-Based Anomaly Detection of Process Instances: Benchmark and Tweaks

Authors	Johannes Lahann, Peter Pfeiffer, Peter Fettke
Year	2023
Published in	*International Conference on Process Mining (ICPM) 2022 – ML4BPM Workshop*, Lecture Notes in Business Information Processing (LNBIP) volume 468

Paper 6

Multi-perspective Identification of Event Groups for Event Abstraction

Authors	Adrian Rebmann, Peter Pfeiffer, Peter Fettke, Han van der Aa
Year	2023
Published in	*International Conference on Process Mining (ICPM) 2022 – EDBA Workshop*, Lecture Notes in Business Information Processing (LNBIP) volume 468

Paper 7

Towards Process Representation Models for Business Process Management

Author	Peter Pfeiffer

Year	2023
Published in	*AAAI Bridge Program: Artificial Intelligence and Business Process Management (AI4BPM)*

Paper 8

Business Process Deviation Prediction: Predicting Non-Conforming Process Behavior

Authors	Michael Grohs, Peter Pfeiffer, Jana-Rebecca Rehse

Year	2023
Published in	*International Conference on Process Mining (ICPM) 2023*, IEEE

Paper 9

A Discussion on Generalization in Next-Activity Prediction

Authors	Luka Abb, Peter Pfeiffer, Peter Fettke, Jana-Rebecca Rehse
Year	2024
Published in	*International Conference on Business Process Management (BPM) 2023 – AI4BPM Workshop*, Lecture Notes in Business Information Processing (LNBIP) volume 492

Paper 10

Trace vs. Time: Entropy Analysis and Event Predictability of Traceless Event Sequencing

Authors	Peter Pfeiffer, Peter Fettke
Year	2024
Published in	*International Conference on Business Process Management (BPM) 2024 – Forum*, Lecture Notes in Business Information Processing (LNBIP) volume 526

Paper 11

Proactive conformance checking: An approach for predicting deviations in business processes

Authors	Michael Grohs, Peter Pfeiffer, Jana-Rebecca Rehse
Year	2025
Published in	*Information Systems (IS) volume 127*

Paper 12

Learning from the Data to Predict the Process: Generalization Capabilities of Next Activity Prediction Algorithms

Authors	Peter Pfeiffer, Luka Abb, Peter Fettke, Jana-Rebecca Rehse

Year	2025
Published in	*Business & Information Systems Engineering (BISE)*

Part III

Appendix

Appendix

Overview

- MobIS-Challenge 2019 Description
- Seminal work of the S2SA (Forschungsprojekt)

MobIS-Challenge 2019 Description

The following description (Scheid et al. 2018) has been published along the event log for the MobIS-Challenge:

The provided process log describes the execution of a business travel management process in a medium-sized consulting company. This process is fully covered and logged by the company's internal workflow management system, for which each employee has access. The tasks and rights are assigned according to the employee's role in the company (normal employee, manager, director, etc.). An internal travel department is responsible for booking the business trips, in accordance with the respective employee. The process consists of two parts, planning before the trip and accounting for expenses after it. For planning a trip, its dates and expected costs are specified first. If the help of the travel department is required, a preliminary price inquiry can be issued first. When the costs are known, a formal business trip request is filed, which has to be approved by the manager. As soon as the trip has been approved, the trip can be booked. After the trip, the travel expenses have to be accounted for. Therefore, potentially incurred cost positions are proved with a receipt and a travel expense report is generated, which again has to be approved by the manager. After the approval, travel expenses are reimbursed by the accounting department. Regarding the process compliance, the company has given itself a number of compliance rules, which are checked in an annual review. Employees are only allowed to take business trips which are necessary for the project. This necessity is checked by the respective manager. Employees should estimate their travel-related costs as realistically as possible, to facilitate travel controlling. Trips are approved by means of the four-eye-principle. To ensure accurate books and accounts, travel expenses should be reported as soon as possible after the end of a trip. To control compliance with those rules, the company has collected all travel-related system data from 2017 into the provided log.

Research Project - Unsupervised Business Process Compliance Checking

Peter Pfeiffer

May 26, 2020