Decision Support Systems 143 (2021) 113494

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Process data properties matter: Introducing gated convolutional neural networks (GCNN) and key-value-predict attention networks (KVP) for next event prediction with deep learning

Kai Heinrich [a,b], Patrick Zschech [c], Christian Janiesch [d,*], Markus Bonin [e]

[a] Otto-von-Guericke-Universität Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany
[b] Technische Universität Dresden, Helmholtzstraße 10, 01069 Dresden, Germany
[c] Friedrich-Alexander-Universität Erlangen-Nürnberg, Lange Gasse 20, 90403 Nürnberg, Germany
[d] Julius-Maximilians-Universität Würzburg, Sanderring 2, 97084 Würzburg, Germany
[e] Capgemini Deutschland GmbH, Löffelstraße 46, 70597 Stuttgart

ARTICLE INFO

Keywords:
Process mining
Predictive process monitoring
Machine learning
Deep learning
Gated convolutional neural network
Key-value-predict attention network

ABSTRACT

Predicting next events in predictive process monitoring enables companies to manage and control processes at an early stage and reduce their action distance. In recent years, approaches have steadily moved from classical statistical methods towards the application of deep neural network architectures, which outperform the former and enable analysis without explicit knowledge of the underlying process model. While the focus of prior research was on the long short-term memory network architecture, more deep learning architectures offer promising extensions that have proven useful for other applications of sequential data. In our work, we introduce a gated convolutional neural network and a key-value-predict attention network to the task of next event prediction. In a comprehensive evaluation study on 11 real-life benchmark datasets, we show that these two novel architectures surpass prior work in 34 out of 44 metric-dataset combinations. For our evaluation, we consider the effects of process data properties, such as sparsity, variation, and repetitiveness, and discuss their impact on the prediction quality of the different deep learning architectures. Similarly, we evaluate their classification properties in terms of generalization and handling class imbalance. Our results provide guidance for researchers and practitioners alike on how to select, validate, and comprehensively benchmark (novel) predictive process monitoring models. In particular, we highlight the importance of sufficiently diverse process data properties in event logs and the comprehensive reporting of multiple performance indicators to achieve meaningful results.

1. Introduction

Predictive process monitoring (PPM) aims to provide timely information about running process instances to identify risks and issues before or while they develop [1]. In this way, companies can derive recommendations for action to manage and control processes at an early stage [2] and reduce their action distance [3]. Depending on the monitoring focus and the richness of event data, PPM approaches support a variety of prediction tasks, such as forecasting remaining cycles times [4] or predicting next events in running process instances [5].

Prediction tasks are realized by methods from the fields of statistics and machine learning (ML) [6]. In ML, the focus is increasingly on artificial neural networks (ANN) that are organized in deep network architectures consisting of multiple processing layers. This allows ANNs to automatically process raw input data without manual feature engineering and learn patterns that are relevant for prediction tasks, which is commonly known as deep learning (DL). With this functionality, advanced DL networks are exceedingly outperforming classic ML techniques as well as statistical approaches, especially in tasks related to large and high-dimensional data, for example, in natural language processing (NLP) and computer vision [7]. In PPM, we face a similar development for predicting process behavior. While the first approaches focused on classical methods like hidden Markov models and support vector machines, recent work is steadily moving towards the application of DL models [e.g., [1,5]]. We observe considerable performance improvements for several prediction tasks such as outcome classification

* Corresponding author.
E-mail addresses: kai.heinrich@tu-dresden.de (K. Heinrich), patrick.zschech@fau.de (P. Zschech), christian.janiesch@uni-wuerzburg.de (C. Janiesch), markus.bonin@capgemini.com (M. Bonin).

https://doi.org/10.1016/j.dss.2021.113494
Received 7 July 2020; Received in revised form 8 January 2021; Accepted 8 January 2021
Available online 11 January 2021

[8] and next event prediction [9].

DL models come in different architectural variants that are mostly characterized by the type of processing units, layers, and connections they use [e.g., [10,11]]. Depending on the properties of the input data, the network architecture determines how relevant structures are automatically recognized by the model. The predominant type in the field of PPM is that of recurrent neural networks (RNN). This type of network is explicitly designed for sequential data properties present in process structures and the underlying event logs. Recurrent architectures offer internal feedback loops and therefore enable sequential pattern learning to model time dependencies by forming a memory. Motivated by the achievements in NLP, Evermann, et al. [5] were among the first to introduce a long short-term memory (LSTM) network as a particular type of RNN to the field of PPM. Much of the subsequent work at the intersection of PPM and DL was based on LSTMs, and researchers referred to it as a reference architecture for proposing further modifications or for adapting it to a variety of different PPM tasks [e.g., [12,13–18]]. Apart from this, only a few architectural DL alternatives have been proposed and tested in comprehensive evaluation studies with competitive results. Some of these exceptions are stacked autoencoders [9], memory-augmented networks [19], and convolutional neural networks (CNN) [20,21]. These examples emphasize that while LSTM is the predominant DL architecture today, it is by no means without competition. Prediction performance rather depends on the triangle of architecture, dataset, and hyperparameters [22].

Against this backdrop, we argue that there cannot be a *silver bullet* architecture for all circumstances. Instead, it rather requires a closer look at the properties of the process data in the corresponding event logs to choose the DL architecture that is best suited to adequately learn and capture the structural patterns within the data to make predictions with high quality.

In this work, we introduce two novel DL architectures, which have demonstrated promising results in sequential modeling tasks such as NLP, to the field of PPM and evaluate them with multiple datasets in varying circumstances. The first approach is a *gated convolutional neural network* (GCNN) as a non-recurrent network alternative. Due to its convolutional layers, the network can learn hierarchical structures in sequences, which are also present in business processes that are subject to non-linear execution patterns. The second approach is a *key-value-predict* attention network (KVP). Based on an advanced attention mechanism, the network can capture long-range connections in large sequences, which can also be observed in real-life processes with many involved activities.

In this respect, our contribution is threefold:

1. We introduce GCNN and KVP as two novel DL architectures into the field of PPM, and more specifically, the task of predicting next events in running instances based on past activities.
2. We conduct a comprehensive evaluation study based on 11 real-life benchmark datasets using multiple established evaluation metrics for assessing the multi-class prediction task.
3. We demonstrate that our approaches show competitive prediction results and exceed existing DL approaches for next event prediction on most datasets with regard to different evaluation metrics. We discuss the circumstances in which the individual DL architectures unfold their benefits.

Our article is structured accordingly: First, we discuss related work on predictive process monitoring and next event prediction in Section 2. Subsequently, we go into more details about previously applied DL architectures and outline GCNN and KVP as novel architectures in Section 3. In Section 4, we describe our study design, followed by the presentation of the results in Section 5. We summarize our findings and highlight potential for future work in Section 6.

2. Related work

Predictive process monitoring covers a variety of tasks, mainly expressed by the target to be predicted. Thus, related work can be found for performance predictions of remaining cycle times, delays, and next timestamps [e.g., [4,23,24]], predictions of partial or final process outcomes [e.g., [8,25,26]], the anticipation of business rule violations [e.g., [27,28]], anomaly classification [e.g., [29]], predictions regarding the next event [e.g., [5,9,13]] or a sequence of next events [e.g., [2]], and prediction-based resource allocation [e.g., [30]].

In this paper, we focus on the task of next event prediction in running process instances using previous activities, based on historical execution and context data recorded in event logs. This is an important task in PPM as it allows business analysts to proactively intervene and prevent undesired behavior in case of anticipated deviations in a timely manner [9]. The task has received special attention in recent years by researchers across different disciplines and is a valuable basis when introducing and comparing novel approaches into the field of PPM [1].

Traditionally, approaches for next event prediction largely relied on explicit process representations, that is, process models. For example, Breuker, et al. [31] proposed a probabilistic finite automaton (RegPFA) based on Bayesian regularization to derive a process model that can be used for predicting future behavior. The authors evaluated their approach with two publicly available data collections from the BPI Challenges 2012 (BPI'12) [32] and 2013 (BPI'13) [33], providing a baseline for benchmark purposes. Other exemplary approaches in this context cover state-transition models [34], sequential pattern mining [35], and hidden Markov models [e.g., [36,37]]. However, such approaches generally face the limitation that they rely on restrictive assumptions given by the explicit process representation [5].

Recent work is steadily moving away from explicit models towards DL approaches due to their capability of automated representation learning and superior prediction results [9]. The focus of this development is primarily on recurrent architectures and especially LSTMs as a subtype of RNNs (cf. Section 3 for further details). One of the first LSTMs was introduced by Evermann, et al. [5], consisting of an embedding layer for transforming event log features into a vector space and two subsequent LSTM layers for predicting upcoming events. Subsequently, Tax, et al. [13] proposed a network for multi-task prediction that consisted of three LSTM layers: an event prediction layer, a timestamp prediction layer, and a shared layer. Input features were transformed via one-hot encoding. Camargo, et al. [12] extended this approach by proposing a multi-task network with four LSTM layers for predicting the upcoming events, their timestamps, and associated resource attributes. Event and resource inputs were encoded with separate embedding layers. Both, Tax, et al. [13] and Camargo, et al. [12] evaluated their approaches with a BPI'12 subset and an additional event log from a ticketing system called Helpdesk [38]. Further LSTM variants were proposed by Tello-Leal, et al. [15] as well as Schönig, et al. [14]. The former introduced a network with a single LSTM layer, whereas the latter considered two LSTM layers for predicting next events in combination with related resource information.

Apart from LSTMs, only few architectural alternatives have been considered for next event prediction. Hinkka, et al. [39] considered an RNN with gated recurrent units (GRU). Such units consist of a cell structure that is less complex in contrast to LSTM cells [40]. Khan, et al. [19] use a memory-augmented neural network (MANN) with external memory units for enhanced learning capabilities of long sequences as compared to conventional LSTMs. Lin, et al. [41] introduced MM-Pred using a modulator, which customizes weights for events and their attributes, to combine separately encoded events and their attributes for PPM.

In contrast to recurrent architectures, Mehdiyev, et al. [9] proposed a network that consists of two stacked autoencoders (SAE) for dimensionality reduction and unsupervised feature learning, followed by a multi-layer perceptron with two hidden layers for activity prediction.

Input events are encoded into *n*-grams in combination with feature hashing to receive a reasonable vector size. The approach is comprehensively evaluated and compared to related work using samples from the Helpdesk, BPI'12, and BPI'13 datasets. Moreover, Di Mauro, et al. [21] and Pasquadibisceglie, et al. [20] introduce a CNN. While the former proposed a network with one embedding layer and three stacked CNN inception modules, the latter applied a CNN with three convolutional blocks to process a spatial representation of event log traces using a frequency-based encoding of activities and timestamps.

3. Deep learning architectures

Prior work has extended existing networks continuously and evaluated the results on benchmark datasets. However, little attention has been paid to the underlying properties of process data to discuss and select DL architecture for application given the specific properties of a dataset. In this section, we explain the structure of established DL architectures and introduce novel approaches relevant to PPM.

3.1. Recurrent neural networks and long short-term memory networks

ANNs consist of connected processing units (i.e., cells) that are organized into networks with multiple layers, including an *input layer*, one or more *hidden layers*, and an *output layer*. Processing units are coupled by weighted connections to other units from previous and subsequent layers in different forms. This gives ANNs the flexibility to come in various architectures and allows them to be modified for a variety of contexts and learning tasks [e.g., [10,11]].

RNNs are a particular type of ANN that specialize in processing sequential data structures such as time-series data and natural language. More specifically, each cell of an RNN is not only connected to a subsequent layer but also feeds information back into itself to maintain a state over time, which serves as a memory to capture time dependencies. The extent of this memory can be specified by the number of time steps to be considered for processing sequences of inputs with variable length. However, classic RNN architectures have difficulties in the training procedure when the data involves long-term dependencies. Due to the problem of vanishing gradients in the training phase, the magnitude of weights assigned to distant events from the beginning of the sequence decrease rapidly so that the model may fail to consider those events during prediction [5].

LSTMs address this problem by introducing advanced long-term memory cells based on gating components with forget functions for unimportant parts and emphasize functions for important parts of the sequence. In this way, the network can control which information will be forgotten and which information will be maintained for long periods of time [42].

As described in the previous section, LSTMs are organized in different layers, mostly depending on the target(s) to be predicted and the type of feature encoding. Since ANNs operate on real-valued data, event log features need to be encoded accordingly. A common option is to use embeddings as an input layer by which events and related attributes are first transformed into an *n*-dimensional vector space and subsequently processed by one or more LSTM layers. Alternatively, input data can be transformed into a feature vector via one-hot-encoding that is directly fed into the LSTM. However, this results in large feature vectors and correspondingly large network layers in case of event logs with many process activities [5].

3.2. Gated convolutional neural networks

A *GCNN* is a non-recurrent network alternative to capture long-term dependencies while avoiding sequential operations for better parallelizability. Thus, recurrent connections typically applied in RNNs are replaced by gated temporal convolutions. In general, convolutional operations are responsible for hierarchical feature learning. For this purpose, multiple convolutional layers are stacked into a network to extract hierarchical structures over increasingly larger contexts with more abstract features. In this way, long-term dependencies can be modeled over a context of size N and kernel width k [43].

To overcome the problem of vanishing gradients, GCNNs contain gating mechanisms like those introduced by LSTMs. More specifically, gated linear units (GLU) are employed to keep control over the information that should be propagated through the hierarchy of layers. Additionally, convolutions and GLUs can be wrapped in residual blocks adding the input of the block to the output for further mitigation of the degradation problem [44]. Since GCNNs have demonstrated state-of-the-art results in the field of NLP [43,45], we assume that they have the potential to advance PPM as well. In this work, we adopt the approach proposed by Dauphin, et al. [43] and apply it to next event prediction.

Formally speaking, the GCNN should produce a representation $H = [h_0, \ldots, h_N]$ of the context for each event $w_0, \ldots, w_N$ to predict the next event $P(w_i | h_i)$. This is done by convolving the inputs with a function f to obtain $H = f * w$. The input of the network is a sequence of events $w_0, \ldots, w_N$ represented by embeddings $E = [D_{w0}, \ldots, D_{wN}]$. The hidden layers of the network $h_0, \ldots, h_L$ are computed as follows

$$h_l(X) = (X*W + b) \otimes \sigma(X*V + c) \tag{1}$$

where m, n are respectively the number of input and output feature maps, k is the patch size, $X \in \mathbb{R}^{N \times m}$ is the input of layer h_l, $W \in \mathbb{R}^{k \times m \times n}$, $b \in \mathbb{R}^n$, $V \in \mathbb{R}^{k \times m \times n}$, $c \in \mathbb{R}^n$ are learned parameters, and σ is the sigmoid function. The output of each layer is a linear projection $X * W + b$ modulated by the gates $\sigma(X * V + c)$. In comparison with a regular CNN that centers a window around a time step, the causal convolution in (1) does not include future time steps in the context, ensuring real prediction power based on past inputs instead of including future information that will not be present for new data. The gating operation in (1) uses simple sigmoid gating to filter in the important parts of the context that serve as input to the next hidden layer.

3.3. Key-value-predict attention networks

Another approach for extending DL architectures for sequential modeling is that of attention mechanisms. Attention layers assist in identifying global dependencies disregarding sequential distance (e.g., time-based). They consist of context and attention vectors, as well as attention weights. Vaswani, et al. [46] extended attention mechanisms by introducing *key-value attention*. Later, Daniluk, et al. [47] improved the approach and proposed *key-value-predict attention* for LSTMs that separates output vectors (2) into a key, a value, and a predict part. The newly introduced attention layer can identify correlations between two elements that have different positions within an input sequence. The context vector contains a summarizing vector that comprises the predictions of past iterations. This separation improves the performance for predicting the next token in very large sequences. We adopt their approach for next event prediction and apply the definitions by Daniluk, et al. [47] to augment LSTMs with KVP attention.

Formally speaking, KVP operates as follows: Let $Y_t = [h_{t-L} \ldots h_{t-1}]$ be a memory of previous L output vectors of an LSTM unit, where k is the output dimension and h_t is the output representation at time step t. To determine the probability distribution y_t for the next event, the following applies:

$$\begin{bmatrix} k_t \\ v_t \\ p_t \end{bmatrix} = h_t \in \mathbb{R}^{3k} \tag{2}$$

$$\mathrm{M_t} = \phi\left(\mathrm{W^Y}[\mathrm{k_{t-L}} \ldots \mathrm{k_{t-1}}] + \left(\mathrm{W^h k_t}\right)1^{\mathrm{T}}\right) \in \mathbb{R}^{\mathrm{k \times L}} \tag{3}$$

$$\alpha_{\mathrm{t}} = \mathrm{softmax}\left(\mathrm{w^T M_t}\right) \in \mathbb{R}^{1 \times \mathrm{L}} \tag{4}$$

$$r_t = [v_{t-L} \ldots v_{t-1}]a^T \in \mathbb{R}^k \quad (5)$$

$$h_t^* = \phi(W^r r_t + W^x p_t) \in \mathbb{R}^k \quad (6)$$

$$y_t = \text{softmax}\left(W^* h_t^* + b\right) \in \mathbb{R}^{|V|} \quad (7)$$

with W^Y, W^h, W^r, $W^x \in \mathbb{R}^{k \times k}$, $W^* \in \mathbb{R}^{|V| \times k}$ being trainable projection matrices, $w \in \mathbb{R}^k$ being a trainable vector, $b \in \mathbb{R}^{|V|}$ being a bias vector, and ϕ being the hyperbolic tangent function. The output vector h_t is divided into three equal parts (2): key (k_t), value (v_t) and predict (p_t). p_t is used to encode the next-event distribution, k_t serves as key, and v_t serves as value for an attention mechanism. The attention distribution α_t is computed from a comparison of the key at time step t with the previous L keys, which is then used to obtain a weighted context representation $r_t \in \mathbb{R}^k$ from values associated with these keys. The final representation h_t^* is computed from a non-linear combination of the attention-weighted representation r_t and the respective encoding of the next-event distribution p_t and subsequently used to predict the next event probability using y_t. In comparison to concurrent context extraction in GCNN, KVP uses a window of the history L, sequentially determining important context events by the comparison in (3) that is used to calculate attention weights in (4). The weights then serve as input to (5) resulting in weighted context values, so that only the important context information is passed through to be used in the subsequent prediction (6) and (7).

4. Study design

Our study design generally follows that of existing studies on next event prediction to ensure comparability of the results [5,9]. That is, we use the same publicly available benchmark datasets, apply established evaluation metrics for multi-class prediction, and compare our results to related work. However, we deviate from previous studies as we go into further details about relevant circumstances and discuss process data properties with respect to the suitability of the novel architectures.

4.1. Datasets and subsets

We used five different dataset collections of real-life event logs for our evaluation: BPI'11 [48], BPI'12 [32], BPI'13 [33], Helpdesk [38], and EnvLog [49]. On the one hand, we considered these datasets because they are widely used for benchmark purposes in the field of PPM [5,9]. On the other hand, we looked for datasets that support a wider range of circumstances regarding their process data properties. Based on those datasets, we additionally consider two typical set of actions performed on process data:

- **Process data focusing.** Limiting the database by only *focusing* on selected process instances or subprocesses of the top-level process,
- **Process data augmentation.** Enriching the database by *augmenting* it with additional activity and/or instance attributes.

We summarize the properties of the top-level datasets used in this study, along with their focusing subsets in Table 1. The metrics to describe process data properties are the number of process instances (#I), the number of unique process variants (#V), the number of events (#E), the number of activity types (#A), minimum, average, and maximum statistics of events and activity types per instance, activity-to-instance ratio (sparsity), variant-to-instance ratio (variation), and average-based event-to-activity ratio (repetitiveness).

In the following, we describe the chosen datasets in more detail:

- **BPI'11** contains event logs from process executions related to cancer treatments of patients in the Gynecology department of a Dutch academic hospital. Each process instance is related to the treatment of one patient.
- **BPI'12** comprises data from a loan application process of a large Dutch financial institute. For focusing, the process can be divided into three subprocesses related to the work (*W*), the application (*A*), and the offer (*O*) of the loan application process. Additionally, the *W* process contains three different lifecycle transitions, schedule, start, and complete (*Cmpl*), whereas *A* and *O* only contain events with the lifecycle transition complete. Analogous to [5,9,13], BPI'12 can be considered as multiple subsets with varying circumstances. Furthermore, for augmentation purposes, the dataset can be enriched by activity attributes (e.g., organizational resources executing an activity (*org:resource*)), and instance attributes (e.g., amount of requested loan (*amnt_req*)).
- **BPI'13** comprises event logs of an incident and problem management system at Volvo IT in Belgium. The event logs can be divided into two separate subsets handling *incidents* and *problems* independently. Analogous to [5,8,9], problem cases can be separated into *open* and *closed* problems for focusing purposes. For augmentation purposes, the subsets can be enriched by additional attributes as well.
- **Helpdesk** contains events from an Italian software company and details the processes of a ticketing system of an IT helpdesk. It has been used widely for evaluation purposes [9,13,19–21] and does not contain any further attributes besides case and activity number as well as a timestamp.
- **EnvLog** contains data from an environmental building permit application process ('WABO') from five different anonymous Dutch municipalities. The events of the five processes are comparable, as the same labels were used across all five municipalities. For augmentation purposes, the dataset can be enriched with attributes at different levels (e.g., human resource involved (*org:resource*)).

4.2. Process data properties

In general, process event logs such as the ones summarized in Table 1 exhibit certain properties affecting next event prediction performance in critical ways. Since these properties can be determined prior to the analysis, they enable the suggestion of appropriate DL architectures

Table 1
Selected datasets and process data properties.

Dataset	Instances (#I)	Variants (#V)	Events (#E)	Activities (#A)	min/avg/max E per I	min/avg/max A per I	Sparsity	Variation	Repetitiveness
BPI'11	1143	981	150,291	624	1/131/1814	1/33/113	0.5459	0.8583	3.97
BPI'12 (all events)	13,087	4366	262,000	36	3/20/175	3/12/32	0.0028	0.3336	1.67
BPI'12Cmpl	13,087	4336	164,506	23	3/13/96	3/8/20	0.0018	0.3313	1.63
BPI'12WCmpl	9658	2263	72,413	6	1/7/74	1/2/6	0.0006	0.2343	3.50
BPI'12A	13,087	17	60,849	10	3/5/8	3/5/8	0.0008	0.0013	1.00
BPI'12O	5015	168	31,244	7	3/6/30	3/5/6	0.0014	0.0335	1.20
BPI'13Incidents	7554	2278	65,533	13	1/9/123	1/4/9	0.0017	0.3016	2.25
BPI'13Problems	2306	454	9011	7	1/4/35	1/3/6	0.0030	0.1969	1.33
BPI'13Closed	1487	327	6660	7	1/4/35	1/3/6	0.0047	0.2199	1.33
Helpdesk	3804	154	13,710	9	1/4/14	1/3/5	0.0024	0.0405	1.33
EnvLog	787	781	34,848	331	2/44/93	2/44/92	0.4206	0.9924	1.00

based on dataset properties. Kratsch, et al. [8] propose three *process data properties* that can affect DL model performance substantially. For our analysis, we specifically focus on two of these (*variation* and *repetitiveness*). Following [2], we take a more general stance on the process data property *sparsity* as the ratio between activity labels and traces rather than solely considering payload attributes:

- **Sparsity:** A high activity-to-instance ratio can be an issue for process prediction. That is, there are only (relatively) few instances in the dataset to train a model in relation to the complexity of the process resulting in high sparsity because some or even most activities are not instantiated often and, thus, are more difficult to predict. For example, although the average number of activities in the EnvLog dataset is only 44 (compared to 20 for BPI'12), the dataset only provides 787 instances for 331 possible activities resulting in a high sparsity of 0.42, whereas BPI'12 has a comparably low sparsity of 0.0028.
- **Variation:** The number of observed unique activity combinations measures the variant-to-instance ratio that describes the potential variation in the process data. Low variation indicates standardized process behavior, while high variation indicates highly individual behavior resulting in many variants. While the complexity of the process data in terms of possible activity types and the average number of events can be low, the variation can still be comparably high. For example, the BPI'12WCmpl dataset contains 2263 variants, whereas BPI'12A and BPI'12O comprise 17 and 169, respectively.
- **Repetitiveness:** The number of loops and iterations, resulting in many repetitions of activities can be expressed by comparing the average number of events with the average number of activities. High event-to-activity ratio processes usually contain few activities that are repeated often, while a low event-to-activity ratio indicates many specific and rare combinations rather than many loops. However, this is not necessarily a sign of standardization, as we can see from EnvLog where we have high variation (0.99) and low repetitiveness (1.00) compared to BPI'11 with high repetitiveness (3.97), but also a high variation (0.85).

4.3. Classification properties and evaluation metrics

To handle stochastic influences when training DL models, for example, due to random parameter initialization and arbitrary order of training samples, we used 10-fold cross-validation to perform our evaluation [5]. The results serve for statistical analyses and classifier comparison [50,51] based on two *classification properties*.

- **Generalization:** To assess the classification property of generalization, we provide both training accuracy (Tr) and validation accuracy (Va) results of the cross-validation. In addition, we calculated the metrics of precision, recall, and *F*1-score for the validation data [52,53].
- **Class imbalance:** The distribution of activity instances to activity types in a dataset can deviate heavily from a uniform distribution resulting in some rare activities (i.e., classes) with only a few instances. This potentially signifies the classification property of class imbalance. Thus, we considered multiple variants of averaging the metrics over all activity classes: i) macro averaging (*ma*), which does not take the sizes of the l classes into account, ii) micro averaging (*mi*), which accounts for the difference in class distribution, and iii) a weighted average (*w*) that is based on the number of class occurrences s_i for each class i in the dataset of size n [9].

We set $\beta = 1.0$ for the *F*1-score. The metrics are calculated using the outputs of the one-vs-all confusion matrix for each class: *tp* for true positives, *tn* for true negatives, *fn* for false negatives, and *fp* for false positives. Our metrics are defined as follows:

$$\text{accuracy} = \frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i + tn_i}{tp_i + fn_i + tn_i + fp_i} \tag{8}$$

$$\text{precision}_{\text{weighted}} = \frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i}{tp_i + fp_i} \tag{9}$$

$$\text{precision}_{\text{macro}} = \frac{1}{l}\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i} \tag{10}$$

$$\text{precision}_{micro} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i + fp_i)} \tag{11}$$

$$\text{recall}_{\text{weighted}} = \frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i}{tp_i + fn_i} \tag{12}$$

$$\text{recall}_{\text{macro}} = \frac{1}{l}\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i} \tag{13}$$

$$\text{recall}_{micro} = \frac{\sum_{i=1}^{l} tp_i}{\sum_{i=1}^{l}(tp_i + fn_i)} \tag{14}$$

$$F1\text{-score}_{\text{weighted}} = \frac{(\beta^2+1)\text{precision}_{\text{weighted}} * \text{recall}_{\text{weighted}}}{\beta^2 \text{precision}_{\text{weighted}} + \text{recall}_{\text{weighted}}} \tag{15}$$

$$F1\text{-score}_{\text{macro}} = \frac{(\beta^2+1)\text{precision}_{\text{macro}} * \text{recall}_{\text{macro}}}{\beta^2 \text{precision}_{\text{macro}} + \text{recall}_{\text{macro}}} \tag{16}$$

$$F1\text{-score}_{\text{micro}} = \frac{(\beta^2+1)\text{precision}_{\text{micro}} * \text{recall}_{\text{micro}}}{\beta^2 \text{precision}_{\text{micro}} + \text{recall}_{\text{micro}}} \tag{17}$$

4.4. Technical implementations

We set up a development environment in Python using different ML packages and frameworks to implement our study design: OpenXes and NLTK were used for data transformation, Tensorflow and Keras for implementing the DL architectures, Scikit-opt for parameter tuning, and Scikit-learn, Pandas and Seaborn for data management and evaluation [54]. Furthermore, we used the Process Mining Toolkit (ProM) [55] for calculating the descriptive statistics of the event logs in Table 1 and for creating subsets of the BPI'12 and BPI'13 datasets.

For the extraction of features from the event logs, we followed Evermann, et al. [5] and created tokens as a concatenation of *activity names* (e.g., "Approve Loan", "Submit Order"), *lifecycle transitions* (e.g., "Start", "Complete"), and, for some experiments, additional *event attributes* such as organizational resources (e.g., "Unit A", "Supervisor"). Thus, an exemplary event within a sequence takes the form "Approve Loan–Started–Unit A", with all unique events that follow this form constituting the vocabulary of an event log.

To instantiate the novel DL architectures of GCNN and KVP, we adapted existing implementations[1] from NLP applications and modified them regarding feature preparation, parameter tuning, and validation procedures. For benchmarking purposes, we reconstructed two additional DL networks, which have demonstrated superior prediction

[1] GCNN: https://github.com/astanway/gated-conv-nets | KVP: https://github.com/asahi417/LSTMCell/blob/master/lstm_cell/kvp_attention_cell.py

performances across a variety of datasets for PPM: an LSTM as introduced by Evermann, et al. [5] and an SAE as proposed by Mehdiyev, et al. [9].

More specifically, we implemented a GCNN using depth-wise 2D convolution with eleven hidden layers wrapped in a flexible number of residual blocks [1:10]. KVP and LSTM were implemented with up to three hidden RNN layers. All three networks use embedding layers as input for feature encoding. In contrast, the SAE is based on *n*-grams and feature hashing for feature encoding as proposed by the authors. Since no source code is available for replication, we reconstructed the SAE in an educated guess using a flexible number of hidden layers [3:7].

Regarding further hyperparameters, we kept several of them unchanged, as suggested by related work. For several other parameters such as the size of embeddings or the prefix length of preceding event sequences (i.e., sequence length and kernel width for GCNN, number of time steps for LSTM and KVP), we searched optimal settings. Table 2 details the hyperparameters that we considered for optimization. We applied a grid search method using Bayes optimization [56] based on the intervals given in Table 2 and checked if results are close to boundaries to adjust the intervals if necessary. The specifications of all networks can be reviewed in our source code [57].

All networks were trained on a high-performance computing cluster using 64 nodes, each with 2× Intel(R) Xeon(R) CPU E5-E5-2680 v3 (12 cores) @2.50GHz, no multi-threading, 64 GB RAM (2.67 GB per core), 128 GB SSD, 4× NVIDIA Tesla K80 (12 GB GDDR RAM). The training procedure generally covered 100 epochs. However, in most cases, the results converged much faster towards a stable level, especially for datasets with a lower vocabulary (e.g., BPI'12A and BPI'12O). After running all epochs, the model with the best validation accuracy was chosen.

5. Results

In the following, we present the results of our experiments in a stepwise manner. First, we compare our results to the related work and outline some tendencies of the networks' strengths. Subsequently, we go into further detail by providing a statistical model comparison and discussing the effects of process data properties on our networks' prediction qualities. Thereafter, we consider their classification properties to address dataset challenges. In the last step, we discuss the effects of process data focusing and augmentation. A comprehensive overview of our results with all metrics across all models and datasets can be found in [57].

5.1. Comparison with related work

For a comparison with related work, we consider the metrics of accuracy, precision, recall, and *F*1-score in their weighted forms. Table 3 summarizes the results by reporting mean values and standard deviations for each metric across all folds from cross-validation. Both KVP and GCNN achieve competitive results across all datasets and even surpass prior approaches in 34 out of 44 metric-dataset combinations (KVP in 20, GCNN in 14).

Table 2
Optimized hyperparameters of the implemented DL architectures.

GCNN [43]	KVP [47]	LSTM [5]	SAE [9]
Residual blocks [1:10]	Hidden RNN layers [1:3]	Hidden RNN layers [1:3]	Hidden layers [3:7]
Kernel width [3:5]	Hidden Units [256:612]	Hidden Units [256:612]	Hidden units [10:500]
Sequence length [3:15]	Time steps [5:30]	Time steps [5:30]	Learning rate [0.001:1]
Minibatch size [20:300]	Learning rate [0.8:1]	Learning rate [0.8:1]	Learning decay [0.8:0.99]
Embedding size [128:256]	Keep probability [1:10]	Keep probability [1:10]	Momentum [0.7:0.9]
	Batch size [10:30]	Batch size [10:30]	Activation [ReLu, Sigmoid]
	Embedding size [256:612]	Embedding size [256:612]	Batch size [20:100]
	Attention window [2:6]		

With respect to specific metrics, we can distinguish several tendencies. When focusing on accuracy as the most commonly reported metric across the related work, GCNN outperforms alternative approaches in 6 of 11 datasets. In comparison to previous studies, we can see differences in accuracy of up to 8% (e.g., 7.2% for Helpdesk compared to [21], 8% for BPI'12 compared to [12], 4.9% for BPI'12A compared to [9]). In contrast, when focusing on the *F*1-score as a weighted mean of precision and recall, KVP is more dominant and exceeds the performance of other approaches in 7 out of 11 datasets. Here, we obtain differences of up to 16.8% as observed in the Helpdesk dataset in comparison to [9].

With respect to datasets, we can report some tendencies as well. While GCNN generally performs well across all metrics on BPI'12A and BPI'12O, KVP shows remarkable results on BPI'11 and EnvLog. This demonstrates that there is no *silver bullet* approach dominating the results across all datasets and metrics. It rather requires a closer look at the effects of process data and classification properties on model performance, which we will present in the subsequent sections.

Furthermore, our reconstructed SAE performs rather poorly and the results are generally lower than those reported by Mehdiyev, et al. [9]. An exception is the dataset BPI'12WCmpl where both SAE models are among the best results. Interestingly, this is the only dataset that is characterized by a low degree of sparsity and variation in combination with a high degree of repetitiveness. This indicates that SAE can handle these circumstances exceptionally well. However, due to the poor results of our SAE, we did not include it in the further analyses.

In contrast, we used our LSTM as a baseline for subsequent analyses since the model shows promising results and outperforms other LSTM approaches on several datasets (e.g., Helpdesk compared to [13], BPI'12 compared to [5,12], BPI'13 compared to [5]).

5.2. Statistical model comparison and effects of process data properties

To assess our DL models' prediction qualities in more detail, we performed a statistical model comparison. Hutson [58] suggests that additional statistical testing should be done to underline the comparison efforts for classifiers in any setting. We performed the Iman and Davenport omnibus test followed by the Friedman post hoc test with Bergmann and Hommel's correction for comparison over all datasets using a significance level of $\alpha = 0.05$ [51,59]. The omnibus test showed significant overall differences for *F*1-score ($p < 0.0001$) and validation accuracy ($p = 0.01$). Post hoc analysis on *F*1-scores, however, only revealed significant differences between GCNN and KVP ($p < 0.001$). Post hoc results for validation accuracy only indicated significant differences between GCNN and LSTM ($p = 0.02$). As shown in Fig. 1, we can confirm that GCNN performed better on average when only looking at accuracy, which is often considered by practitioners and researchers alike. However, accuracy does not distinguish between positive and negative classes, while the *F*1-score as a weighted mean between precision and recall does [60]. This confirms that prediction methods should not be evaluated by a single metric. Instead, a multi-perspective view is necessary to assess a model's prediction qualities and its suitability for the task at hand [52].

Furthermore, we found that GCNN lacks consistency, showing large deviations from its mean, especially when looking at the *F*1-score, in comparison to LSTM and KVP, which show less deviation. The results concur with the notion that there is no *silver bullet* algorithm that performs best in every situation.

To understand the differences in performance, we looked into the different datasets to reveal which process data properties exert certain

Table 3
Results obtained in comparison to benchmark approaches.

Dataset / Model	Accuracy	Precision (*w*)	Recall (*w*)	*F*1-score (*w*)
BPI'11				
LSTM (ours)	0.621 ± 0.031	**0.775 ± 0.028**	0.621 ± 0.031	0.665 ± 0.028
SAE (ours)	0.422 ± 0.005	0.430 ± 0.008	0.422 ± 0.005	0.380 ± 0.004
GCNN (ours)	0.565 ± 0.077	0.557 ± 0.018	0.546 ± 0.016	0.528 ± 0.017
KVP (ours)	**0.691 ± 0.032**	0.728 ± 0.016	**0.692 ± 0.021**	**0.700 ± 0.018**
BPI'12				
LSTM [5]	–	0.859 ± 0.005	–	–
LSTM (ours)	0.841 ± 0.022	0.924 ± 0.017	0.841 ± 0.023	0.867 ± 0.021
LSTM [12]	0.786	–	–	–
SAE (ours)	0.690 ± 0.013	0.745 ± 0.010	0.690 ± 0.013	0.684 ± 0.012
GCNN (ours)	**0.866 ± 0.007**	0.864 ± 0.008	**0.866 ± 0.007**	0.844 ± 0.005
KVP (ours)	0.855 ± 0.027	**0.930 ± 0.023**	0.855 ± 0.027	**0.876 ± 0.030**
BPI'12Cmpl				
LSTM [5]	–	0.788 ± 0.006	–	–
LSTM (ours)	0.757 ± 0.020	0.870 ± 0.013	0.760 ± 0.003	0.799 ± 0.006
SAE (ours)	0.686 ± 0.004	0.741 ± 0.005	0.686 ± 0.004	0.678 ± 0.004
GCNN (ours)	**0.805 ± 0.015**	0.807 ± 0.016	**0.805 ± 0.015**	0.769 ± 0.008
KVP (ours)	0.756 ± 0.019	**0.875 ± 0.009**	0.759 ± 0.005	**0.800 ± 0.005**
BPI'12WCmpl				
RegPFA [31]	0.719	–	0.578	–
LSTM [5]	–	0.658 ± 0.020	–	–
LSTM (ours)	0.648 ± 0.032	0.805 ± 0.051	0.665 ± 0.009	0.715 ± 0.019
LSTM [13]	0.760	–	–	–
LSTM [12]	0.778	–	–	–
SAE [9]	**0.831**	0.811	**0.832**	–
SAE (ours)	0.807 ± 0.002	0.794 ± 0.003	0.807 ± 0.002	**0.795 ± 0.002**
MANN [19]	0.777	–	–	–
CNN [20]	0.782	–	–	–
GCNN (ours)	0.765 ± 0.014	0.773 ± 0.013	0.765 ± 0.014	0.710 ± 0.017
KVP (ours)	0.645 ± 0.022	**0.829 ± 0.041**	0.662 ± 0.010	0.723 ± 0.018
BPI'12A				
RegPFA [31]	0.801	–	0.723	
LSTM [5]	–	0.832 ± 0.010	–	–
LSTM (ours)	0.813 ± 0.022	0.912 ± 0.018	0.817 ± 0.013	0.848 ± 0.012
SAE [9]	0.824	0.852	0.824	0.817
SAE (ours)	0.630 ± 0.005	0.610 ± 0.008	0.630 ± 0.005	0.602 ± 0.006
GCNN (ours)	**0.873 ± 0.015**	**0.924 ± 0.005**	**0.873 ± 0.015**	**0.880 ± 0.010**
GCNN (org: resource)[a]	**0.877 ± 0.007**	**0.918 ± 0.004**	**0.877 ± 0.007**	**0.882 ± 0.007**
KVP (ours)	0.824 ± 0.011	0.915 ± 0.014	0.823 ± 0.009	0.853 ± 0.010
BPI'12O				
RegPFA [31]	0.811	–	0.647	–
LSTM [5]	–	0.836 ± 0.010	–	–
LSTM (ours)	0.816 ± 0.016	0.893 ± 0.013	0.820 ± 0.010	0.833 ± 0.011
SAE [9]	0.821	0.847	0.822	–
SAE (ours)	0.768 ± 0.004	0.815 ± 0.005	0.768 ± 0.004	0.732 ± 0.005
GCNN (ours)	**0.861 ± 0.008**	0.895 ± 0.006	**0.861 ± 0.008**	**0.857 ± 0.008**
GCNN (org: resource)[a]	**0.885 ± 0.008**	0.904 ± 0.003	**0.885 ± 0.008**	**0.882 ± 0.008**
KVP (ours)	0.843 ± 0.013	**0.909 ± 0.008**	0.833 ± 0.005	0.847 ± 0.004
BPI'13Incidents				
RegPFA [31]	0.714	–	0.377	–
LSTM [5]	–	0.735 ± 0.044	–	–
LSTM (ours)	0.697 ± 0.031	0.793 ± 0.028	**0.698 ± 0.031**	**0.734 ± 0.031**
SAE [9]	0.663	0.648	0.664	0.647
SAE (ours)	0.436 ± 0.030	0.601 ± 0.087	0.436 ± 0.030	0.463 ± 0.020
GCNN (ours)	**0.740 ± 0.044**	0.591 ± 0.011	0.676 ± 0.009	0.613 ± 0.011
KVP (ours)	0.666 ± 0.030	**0.795 ± 0.037**	0.666 ± 0.030	0.706 ± 0.027
BPI'13Problems				
RegPFA [31]	**0.690**	–	0.521	–
LSTM [5]	–	0.628 ± 0.086	–	–
LSTM (ours)	0.649 ± 0.023	0.811 ± 0.062	0.648 ± 0.019	0.699 ± 0.022
SAE [9]	0.662	0.641	**0.662**	–
SAE (ours)	0.484 ± 0.045	0.407 ± 0.031	0.484 ± 0.045	0.428 ± 0.020
GCNN (ours)	0.654 ± 0.013	0.668 ± 0.026	0.654 ± 0.013	0.631 ± 0.011
KVP (ours)	0.650 ± 0.022	**0.828 ± 0.043**	0.648 ± 0.016	**0.702 ± 0.022**
BPI'13Closed				
LSTM (ours)	**0.682 ± 0.024**	0.853 ± 0.064	**0.682 ± 0.025**	0.734 ± 0.025
SAE (ours)	0.496 ± 0.042	0.422 ± 0.027	0.496 ± 0.042	0.440 ± 0.018
GCNN (ours)	0.665 ± 0.024	0.641 ± 0.029	0.665 ± 0.024	0.642 ± 0.024
KVP (ours)	0.680 ± 0.024	**0.870 ± 0.064**	0.680 ± 0.025	**0.741 ± 0.027**
Helpdesk				
LSTM (ours)	0.849 ± 0.019	**0.912 ± 0.008**[b]	0.853 ± 0.009	**0.880 ± 0.009**[b]
LSTM [13]	0.712	–	–	–
SAE [9]	0.782	0.632	0.781	0.711
SAE (ours)	0.743 ± 0.009	0.658 ± 0.014	0.743 ± 0.009	0.687 ± 0.012
MANN [19]	0.714	–	–	–
CNN [20]	0.739	–	–	–
CNN [21]	0.785 ± 0.005	–	–	–
GCNN (ours)	**0.857 ± 0.019**	0.847 ± 0.021	**0.857 ± 0.019**	0.848 ± 0.020
KVP (ours)	0.849 ± 0.034	**0.911 ± 0.007**[b]	0.852 ± 0.009	**0.879 ± 0.008**[b]
EnvLog				
LSTM (ours)	0.679 ± 0.016	0.713 ± 0.016	0.672 ± 0.012	0.683 ± 0.013
SAE (ours)	0.524 ± 0.003	0.710 ± 0.005	0.524 ± 0.003	0.579 ± 0.004
GCNN (ours)	0.615 ± 0.018	0.636 ± 0.016	0.615 ± 0.018	0.595 ± 0.018
KVP (ours)	**0.776 ± 0.051**	**0.806 ± 0.036**	**0.782 ± 0.043**	**0.787 ± 0.040**

[a] Further values for process data augmentation can be found in [57].
[b] We consider LSTM and KVP to perform equally well since LSTM shows higher mean values, while KVP shows lower standard deviations.

influences on the prediction qualities of the three DL models. Fig. 2 summarizes the performances in terms of accuracy and *F*1-score for the six top-level datasets. Several tendencies can be observed. For example, all three models perform equally well with high prediction qualities for

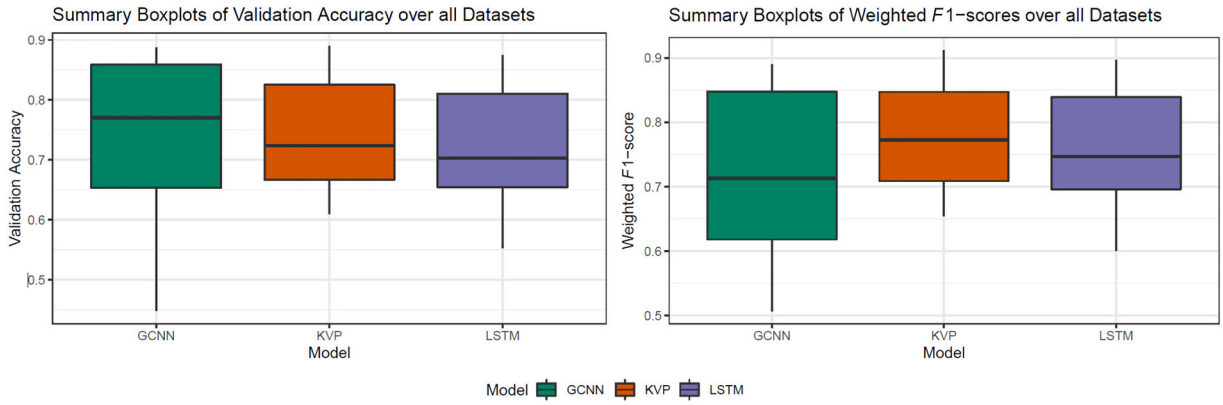

Fig. 1. Overall performance of our models.

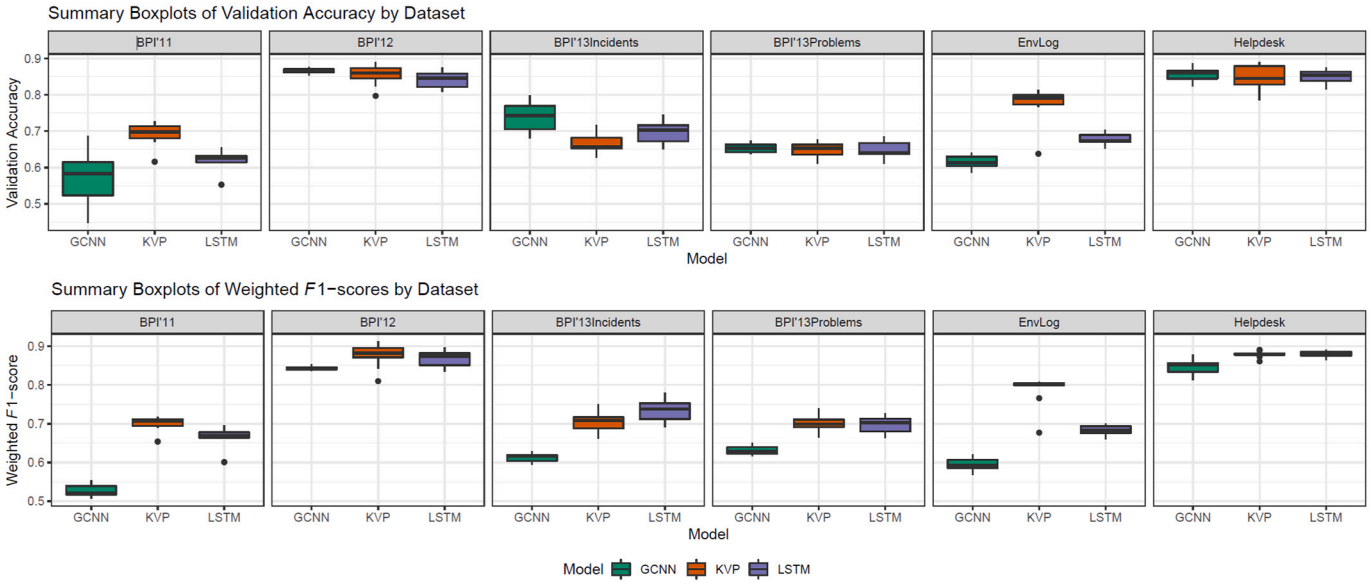

Fig. 2. Model performance by top-level dataset.

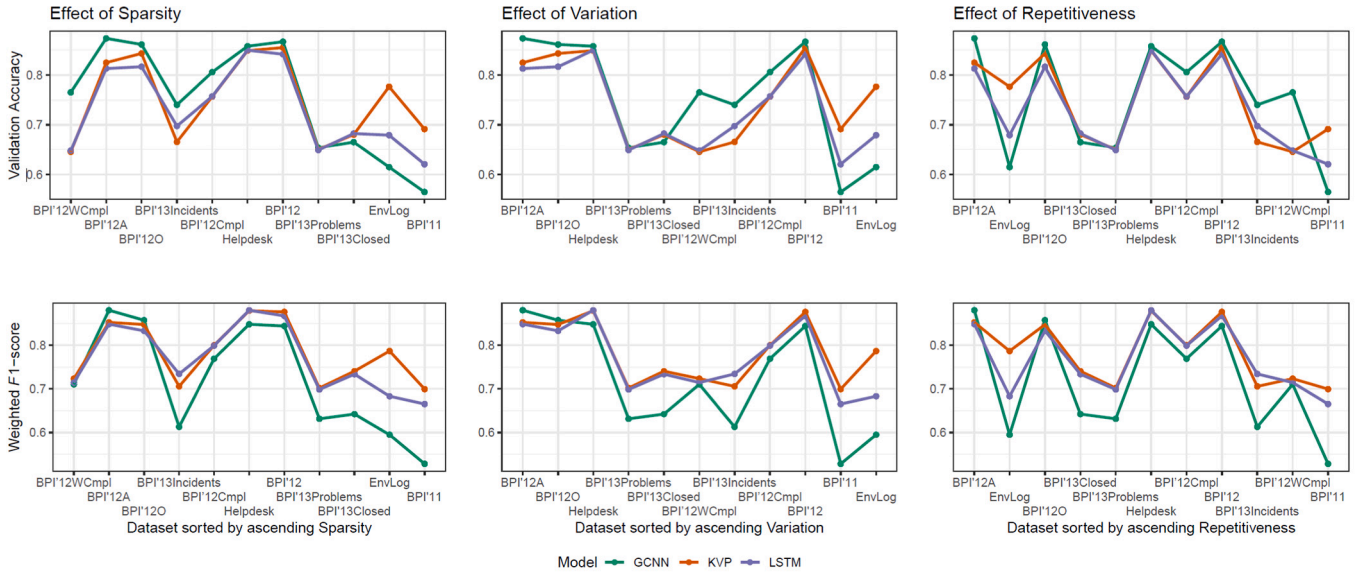

Fig. 3. Model performance by process data properties.

BPI'12 and Helpdesk, both of which show similar characteristics in terms of a low degree of sparsity (0.0028 and 0.0024) and low repetitiveness (1.67 and 1.33). Likewise, we can see the tendency that KVP outperforms the other two models for BPI'11 and Envlog. Both datasets are subject to a high degree of sparsity (0.55 and 0.42).

By sorting the datasets based on their process data properties such effects can be recognized more clearly. Fig. 3 shows the effects of *sparsity*, *variation*, and *repetitiveness* on model performance. The datasets on the *x*-axis are sorted in ascending order of the respective property (i.e., in the top-left plot, BPI'12WCompl shows the least amount of sparsity and BPI'11 the highest amount). We find that sparsity and variation affect prediction performance, especially on the extreme ends: GCNN performs well in situations with low sparsity and low variation, as reflected by the datasets BPI'12A and BPI'12O. In contrast, KVP, with its advanced attention mechanism, excels when faced with complex datasets exhibiting high amounts of sparsity and variation (BPI'11, EnvLog). Thus, applying the previously suggested statistical testing procedure, we found that performance differs for high variation and high sparsity (BPI'11, EnvLog) at $p < 0.001$.

Looking at the effect of repetitiveness, we found that the model performance can only be significantly distinguished for very high repetitiveness: Considering the BPI'11 dataset with high sparsity, KVP outperforms LSTM and GCNN, with LSTM outperforming GCNN at $p < 0.001$ when considering accuracy and *F*1-Score. In the case of the low sparsity dataset BPI'12WCmpl, GCNN outperforms the other models at $p < 0.001$ when looking at accuracy. Likewise, when comparing the equally low repetitive EnvLog and BPI'12A datasets, we find that the difference in performance stems almost entirely from the EnvLog dataset, since in addition to having low repetitiveness, in comparison to BPI'12A, it also has a high sparsity and variation. This leads to the conclusion that the results are rather influenced by sparsity and that the effects can only be observed as interactive components rather than isolated.

5.3. Effects of classification properties

Apart from dealing with process data properties, a general hurdle to overcome – especially in multi-class prediction problems – is the presence of the classification property *class imbalance*. Since the datasets show a very heterogeneous class structure, we checked for differences in performance metrics by taking single-class performances into account using macro-scores. Fig. 4 depicts distribution plots of the micro-macro-difference of *F*1-scores for the six top-level datasets. The individual plots are ordered by average micro-macro-differences. Thus, the largest differences can be observed in Helpdesk and BPI'13Incidents, whereas in BPI'12 with similar process data properties, the differences are considerably smaller. This can be explained by the different distributions of the activity classes as shown in Fig. 5, applying the same order of plots. The first few datasets are characterized by an abrupt drop in frequencies with a few dominating activities and many rare activities, while the distribution in datasets like EnvLog and BPI'12 decreases more evenly, resulting in a lower degree of class imbalance.

Although all three models generally have problems with imbalanced data and correctly predicting rare activities, we can observe several tendencies confirming the previously observed benefits of KVP and GCNN (cf. Fig. 4). Thus, it is noticeable that GCNN performs considerably better in predicting rare activities than its competitors in situations with low sparsity and low variation as represented by Helpdesk (0.0024, 0.04) and BPI'13Problems (0.003, 0.2). In contrast, it performs worse when faced with a high degree of sparsity, as given in BPI'11 (0.546). This is where KVP excels due its advanced attention mechanisms. Thus, even when treating every activity equally independent of its rarity, KVP shows the best performance in both datasets with large sequences and high sparsity (BPI'11, EnvLog) as opposed to LSTM and GCNN. However, in the case of EnvLog, the difference in performance is much smaller and less robust in contrast to LSTM since both recurrent models perform equally well on this dataset.

Another issue raised in the context of learning algorithms is the classification property *generalization* about the quality of the respective algorithm when applied to new data. To measure that quality, we show the distribution of differences between training and validation accuracy for the top-level datasets in Fig. 6. The plots are ordered by average training-validation-difference. While there is barely any difference, or even a negative difference, indicating well generalizable models, for the less complex datasets (BPI'12, Helpdesk, BPI'13Problems, and BPI'13Incidents), we find considerable differences for the complex datasets (BPI'11, EnvLog) that indicate a strong degree of overfitting to the training data. KVP outperforms the other algorithms in BPI'12 and EnvLog. GCNN generalizes well in both BPI'13 datasets while showing worse performance and comparatively large variance when applied to BPI'11. While LSTM tends to overfit for EnvLog, it generalizes well when applied to BPI'11, indicating that LSTM deals well with sparse and repetitive process data in terms of generalization.

5.4. Effects of process data focusing and augmentation

First, we consider the effect of *process data focusing*. As observed in the example of the BPI'12 subsets, different changes in the process data properties may occur after filtering. Thus, subsets can either show an increased or reduced level of complexity as focusing does not necessarily

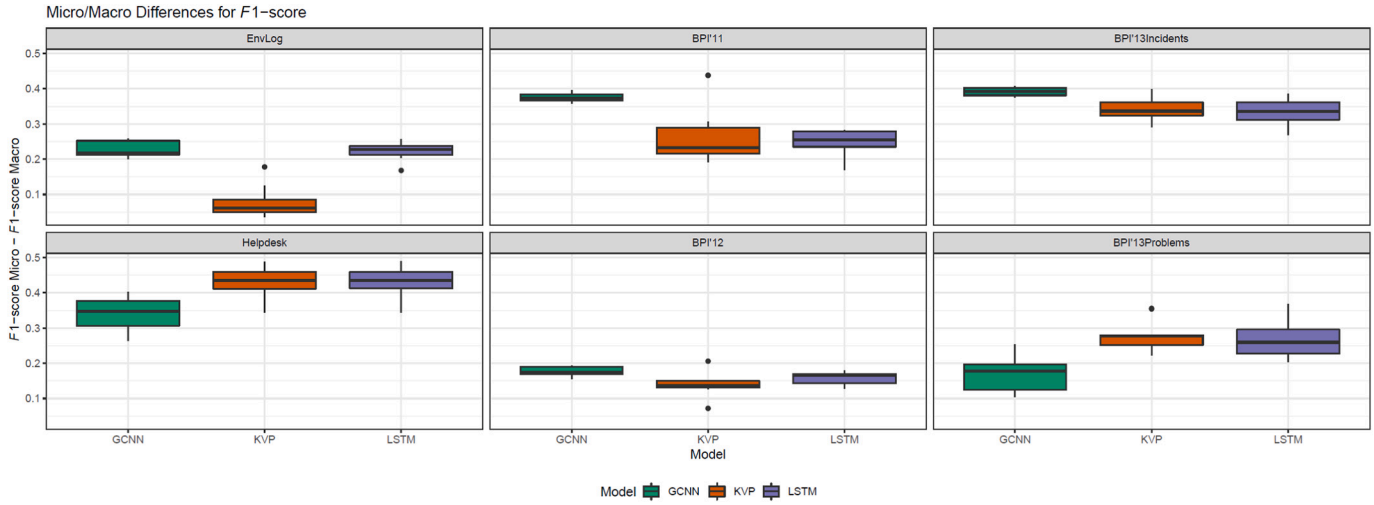

Fig. 4. Class imbalance effect by top-level dataset.

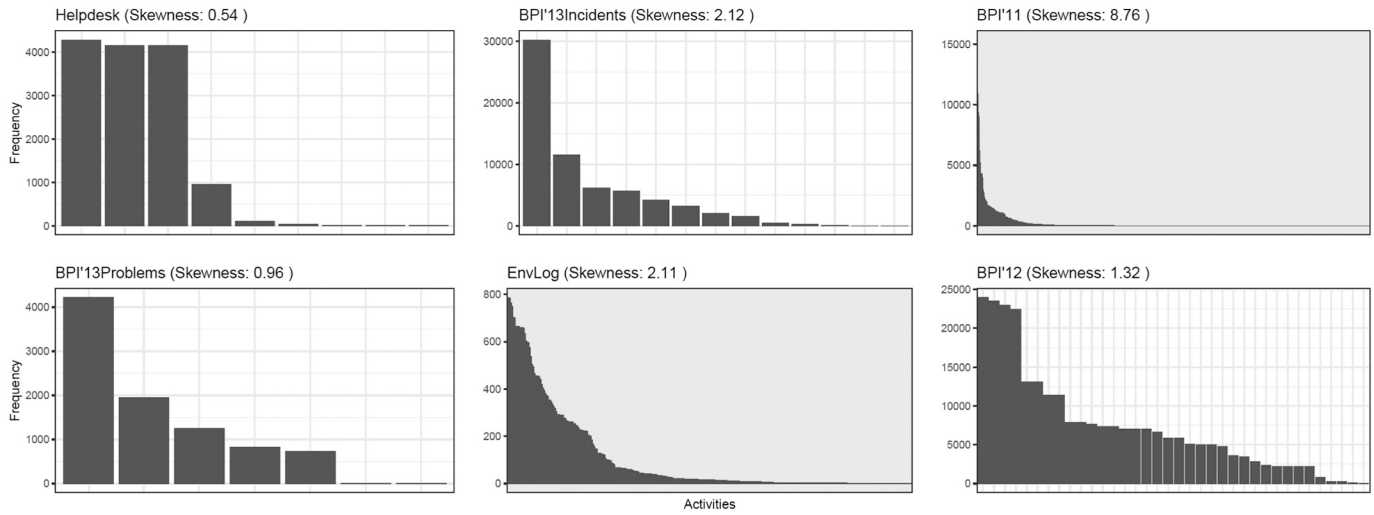

Fig. 5. Activity distribution for selected top-level datasets.

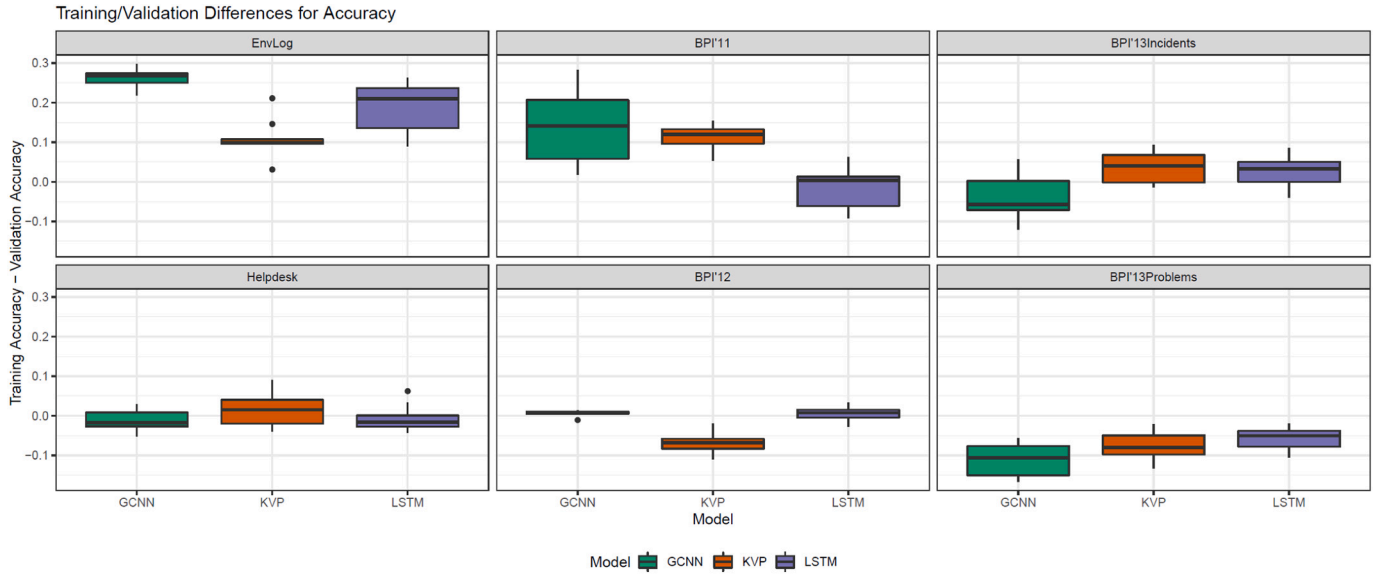

Fig. 6. Overfitting effect by top-level dataset.

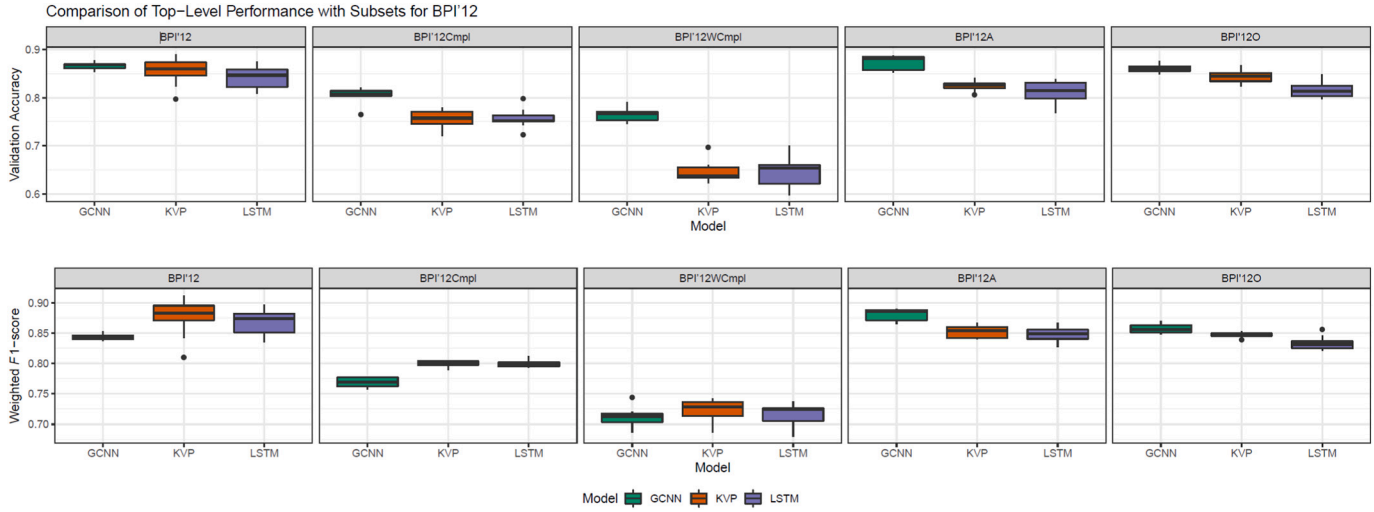

Fig. 7. Performance comparison for BPI'12 focusing.

entail a projection and, thus, simplification, but may as well entail the accentuation of already existing process data properties and the reduction of training data.

When we look at event sequences with the lifecycle transition complete as reflected by BPI'12WCmpl and BPI'12Cmpl, the resulting subsets have a reduced average amount of events. However, they still show a high amount of variation, resulting in a higher degree of complexity (cf. Table 1). Consequently, we encounter decrease in performance for all three models when comparing the results between the subprocesses and the top-level dataset as illustrated in Fig. 7. Nevertheless, there are different tendencies. When only considering accuracy, GCNN outperforms the other models at $p < 0.001$. Contrary, when looking at the *F*1-score, we find that GCNN performs significantly worse than KVP and LSTM at $p < 0.001$, while there is no significant difference between KVP and LSTM at $p = 0.08$. Consequently, we cannot state that any model shows superior performance across both metrics.

When focusing on the subprocesses A and O, on the other hand, we see a strong simplification of the process data properties in terms of much shorter sequences and reduced degrees of sparsity, variation, and repetitiveness. However, instead of witnessing a performance improvement across all three models, only GCNN is capable to benefit from the simplified process structures, whereas LSTM and KVP even perform slightly worse as opposed to the entire dataset of BPI'12 with all events. Consequently, we find that for the subsets A and O, GCNN outperforms the other two models at $p < 0.001$, when considering both accuracy and *F*1-score, confirming our finding that GCNN excels in situation with rather simple process data properties, such as low sparsity and low variation.

Additionally, we also considered a comparison between the BPI'13Problems dataset and the subprocess of closed problems. However, no particular tendency could be observed, except that all three models perform slightly better in the subprocess (cf. Table 3) although the properties of the subset show a marginally higher degree of sparsity (0.003 vs. 0.0047) and variation (0.1969 vs. 0.2199).

As a second effect, we examine *augmentation* by considering additional attributes as input for the prediction. As already noted by Evermann, et al. [5], including attributes may contribute additional information and thus improve predictive power. However, at the same time, this also leads to an increased number of unique tokens resulting in a larger vocabulary (cf. Section 4.4). Thus, predictive performance might be impaired. Consequently, we observed that augmentation could yield improvements but also lead to overfitting. However, the effect differs for the different models and datasets. Fig. 8 shows the augmentation of BPI'12A and EnvLog with additional attributes as examples for datasets with low and high complexity.

Considering the activity attribute *org:resource* for BPI'12A leads to slight improvements for GCNN in terms of accuracy and *F*1-score, and considerably less performance fluctuations between the folds. However, it slightly decreases the performance for LSTM and KVP (cf. Fig. 8). We observed a similar effect for the datasets BPI'12O and BPI'13Problems for all three models when adding the activity attributes *org:res* and *org:group* respectively. In this way, GCNN even achieved the best overall results for BPI'12A (accuracy: 87.7%, *F*1-score: 88.2%) and BPI'12O (accuracy: 88.5%, *F*1-score: 88.2%) (cf. Table 3).

Adding the instance attribute *amnt_req* to BPI'12A with *org:resource* causes a major drop in performance for LSTM and KVP and only a slight decrease for GCNN. While GCNN is very stable when adding additional attributes, KVP and LSTM are rather sensitive, resulting in large performance differences when both, *org:resource* and *amnt_req* are added.

When looking at the more complex EnvLog dataset, we encounter an immediate drop in performance for both recurrent approaches, especially for KVP that outperforms the other models without augmentation. Contrary, GCNN stays at a similar level and even shows slightly better results than KVP based on the additional attribute.

With regard to the overfitting effect of data augmentation, we show the difference in training and validation accuracy in Fig. 9. The plots indicate that the augmentation of the BPI'12A dataset with *amnt_req* leads to overfitting for KVP, while augmentation by both attributes leads to an increased overfitting effect for all three models. In the case of the complex dataset EnvLog, we observe that the generally high effect of overfitting prior to augmentation is, as expected, further intensified and that even the well-performing KVP strongly suffers from this effect when enriching the input.

In summary, we found that GCNN, in contrast to LSTM and KVP, is fairly robust to augmentation and can even improve performance when adding attributes as long as the increase in unique event-attribute-combinations is manageable and overfitting is negligible. The effect of better dealing with augmented events can be attributed to the network's capability of hierarchical feature learning. Starting from specific event-attribute-combinations the network builds increasingly abstract features through the hierarchy of layers. In this way, GCNN can capture sequential structures while using the predictive power of low-level features.

6. Discussion and concluding remarks

Our research focused on the performance of different DL architectures in varying circumstances given by distinct process event logs.

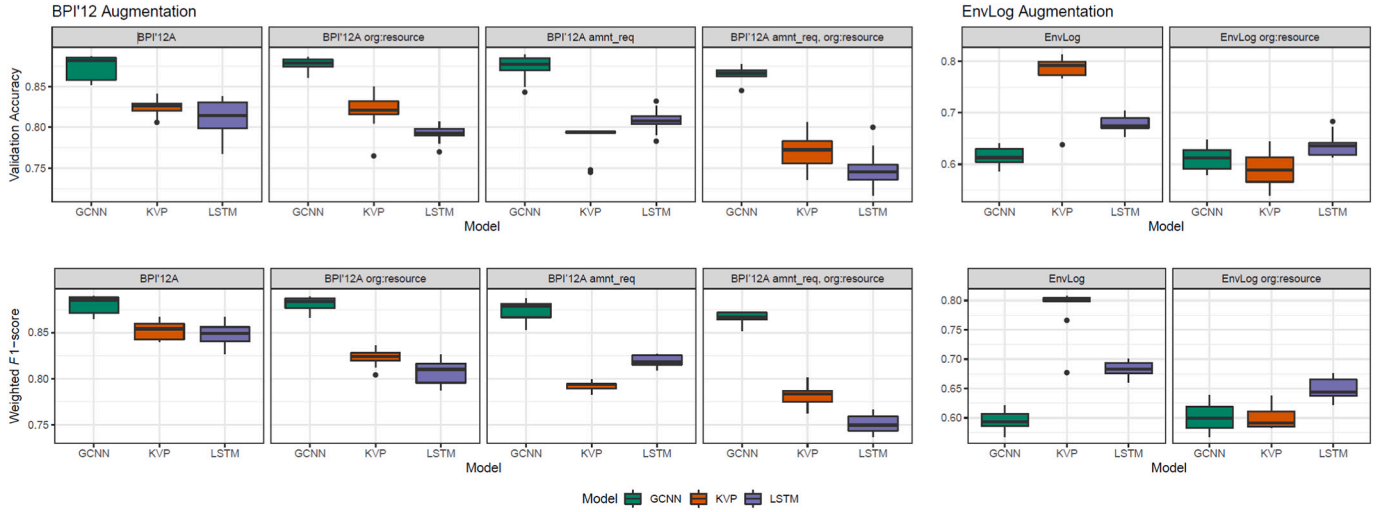

Fig. 8. Performance comparison for BPI'12A and EnvLog augmentations.

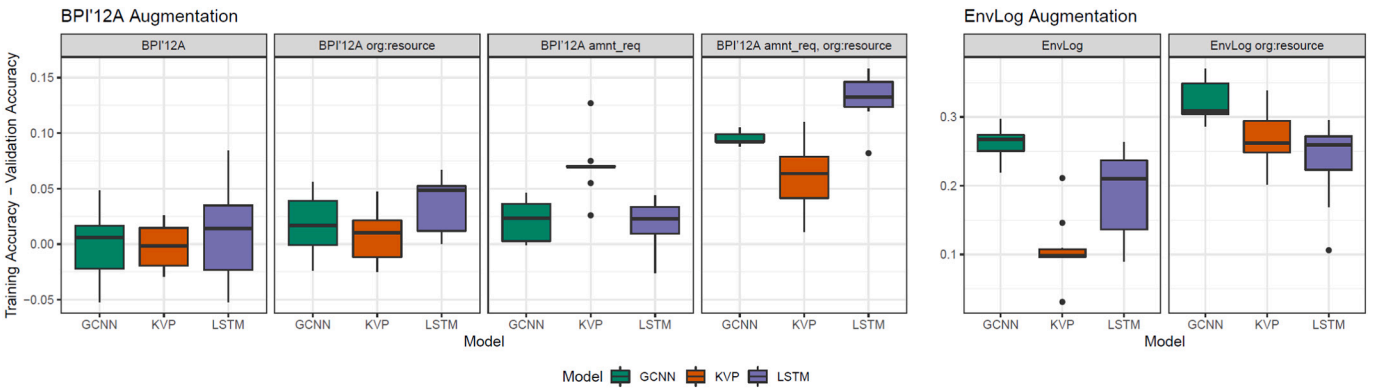

Fig. 9. Overfitting effect for BPI'12A and EnvLog subsets resulting from augmentations.

Against this backdrop, we have introduced GCNN and KVP as two novel DL architectures into the field of PPM and conducted a comprehensive evaluation study based on 11 real-life benchmark datasets. GCNN and KVP show competitive prediction results and exceed the performance of existing DL models in 34 out of 44 metric-dataset combinations.

As process data properties vary greatly between event logs, our research also shows that it is unlikely that there will ever be a *silver bullet* approach for next event prediction. Rather, we suppose that the choice of a specific DL architecture should incorporate its capability to deal with process data properties such as *sparsity*, *variation*, and *repetitiveness*.

In this respect, we found that KVP performs particularly well on complex datasets that are subject to a high degree of sparsity and high variation. The network's superiority in this situation comes with the advanced attention mechanism that separates output vectors into a key, value, and predict part to capture long-range dependencies. Its attention mechanism passes important context through the event sequence by calculating global context weights and, thus, prevents important context from being neglected, even in long sequences. Contrary, GCNN with its capability of hierarchical feature learning excels in situations at the other end of the spectrum. This can be explained by GCNN's focus on extracting local context at the cost of lacking global attention weighting that result in favorable results for short range dependencies but may be less favorable for long sequences. It outperforms the competitors on datasets with low sparsity and low variation, and it is fairly robust to augmentation when enriching events with additional attributes. The strengths of both networks are also apparent when looking at their capability to predict rare activities as a general challenge when facing class imbalance in event logs. The suitability of both networks for specific datasets and the connection of their underlying architectures to prediction quality could be investigated further with a confirmatory explainable artificial intelligence (XAI) study to reveal the detailed weighting process for each prediction.

LSTM's performance as the most popular DL architecture in PPM often lies between the extrema and offers a robust solution, although it outperforms GCNN and KVP in only 7 of the 44 dataset-metric combinations. That being said, the same order of magnitude of performance differences cannot be observed for these datasets with mid-level process data properties in terms of sparsity and variation. That is, in these cases, the choice of DL architecture becomes less critical. Lastly, we noticed an indication that the SAE network by Mehdiyev, et al. [9] performs exceptionally well in situations with high repetitiveness while facing a low degree of variation and sparsity. However, this observation could not be further substantiated as our reconstructed SAE generally showed poor prediction qualities and there was no other dataset in our collection covering this combination of process data properties.

At this point, we also see great potential for further investigations. Currently, our focus was on real-life datasets to review the effects of DL architecture choice across various realistic circumstances. For future work, it seems necessary to generate synthetic datasets in which properties like sparsity, variation, and repetitiveness can be adjusted in a controlled fashion to investigate said impact on classifiers. In the meantime, we suggest validating advances in PPM with DL on sufficiently diverse event logs to avoid only incidental improvements. In this respect, our selection of datasets, including BPI'11, BPI'12, BPI'13, Helpdesk, and EnvLog, could serve as a minimal baseline as they cover a broad spectrum of different process data property combinations.

Moreover, our study shows the importance of reporting multiple metrics to convey an unbiased picture of DL architecture performance. Most of prior studies focus on single metrics to demonstrate the performance in comparison to related work. However, this might not reveal a classifier's full prediction qualities and its suitability for a certain task. For example, while our GCNN shows best results in terms of accuracy across of a majority of datasets, its performance is inferior to KVP when considering the balance between precision and recall as measured by the *F*1-score. Likewise, a single metric is not sufficient to report qualities regarding further classification properties such as class imbalance and generalization. In this sense, our study can serve as an evaluation framework to guide researchers and practitioners alike on how to comprehensively evaluate (novel) PPM approaches.

Naturally, there are some limitations to our work. As Weinzierl, et al. [61] point out, there are situations where encoding may matter more than DL architecture choice. In our work, we did not investigate different encoding options but followed the example of Evermann, et al. [5] by creating event tokens and transforming them into a vector space via embedding layers. Considering the trade-off between generalization and optimization [58], we compromised. We did not use exactly the same hyperparameter configurations as given for prior models, but our decisions were inspired by the respective studies. As a result, we kept several parameters unchanged as suggested by previous work, while we chose several crucial parameters to be specified dynamically via hyperparameter optimization. This conveys some sense of optimization as it would have been done for real-life applications. To ensure comparability, we used multiple common datasets of prior work and established an architecture baseline with an own LSTM implementation to balance the triangle of architecture, dataset, and hyperparameters [22].

Lastly, novel architectures have been and will be proposed continuously in the coming months and years. Echoing the call by Hutson [58], our findings underline that it is important to install a common core of sufficiently distinct process event logs and report metrics comprehensively. With respect to novel architectures, generative adversarial networks (GAN) seem to be a promising avenue to improve the performance in next event prediction [62]. However, in the future, it will not be enough to only focus on performance improvements as there is a trade-off between model performance and model explainability [63]. Harl, et al. [64] and Mehdiyev and Fettke [65] have recognized this and started to experiment with XAI visualizations for PPM.

Declarations of interest

None.

Funding

This research and development project is funded by the Bayerische Staatsministerium für Wirtschaft, Landesentwicklung und Energie (StMWi) within the framework concept "Informations- und Kommunikationstechnik" (grant no. DIK0143/02) and managed by the project management agency VDI+VDE Innovation + Technik GmbH.

Acknowledgement

A prior version of this article has been published as "Heinrich, K., Zschech, P., Janiesch, C., Bonin, M., Ein Vergleich von aktuellen Deep-Learning-Architekturen für die Prognose von Prozessverhalten. Proceedings of the 15. Internationale Tagung Wirtschaftsinformatik (WI). Potsdam, 2020, pp. 876-892. https://doi.org/10.30844/wi_2020_i1-heinrich. It has been revised and extended with permission.

References

[1] A.E. Márquez-Chamorro, M. Resinas, A. Ruiz-Cortés, Predictive monitoring of business processes: a survey, IEEE Trans. Serv. Comput. 11 (2018) 962–977.
[2] C. Di Francescomarino, C. Ghidini, F.M. Maggi, G. Petrucci, A. Yeshchenko, An eye into the future: leveraging A-priori knowledge in predictive business process monitoring, in: Proceedings of the 15th International Conference on Business Process Management (BPM), Barcelona (2017) 252–268.
[3] M. zur Mühlen, R. Shapiro, Business Process Analytics, Handbook on Business Process Management 2, Springer, Berlin, 2010, pp. 243–263.
[4] A. Rogge-Solti, M. Weske, Prediction of business process durations using non-Markovian stochastic petri nets, Inf. Syst. 54 (2015) 1–14.
[5] J. Evermann, J.-R. Rehse, P. Fettke, Predicting process behaviour using deep learning, Decis. Support. Syst. 100 (2017) 129–140.
[6] G. Shmueli, O.R. Koppius, Predictive analytics in information systems research, Manag. Inf. Syst. Q. 35 (2011) 553–572.
[7] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[8] W. Kratsch, J. Manderscheid, M. Röglinger, J. Seyfried, Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction, Bus. Inform. Syst. Eng. (2020) online first.
[9] N. Mehdiyev, J. Evermann, P. Fettke, A novel business process prediction model using a deep learning method, Bus. Inf. Syst. Eng. 62 (2020) 143–157.
[10] S. Leijnen, F. van Veen, The Neural Network Zoo, Proceedings 47, 2020, p. 9.
[11] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M.P. Reyes, M. Shyu, S. Chen, S. S. Iyengar, A survey on deep learning: algorithms, techniques, and applications, ACM Comput. Surv. 51 (2018) 92.
[12] M. Camargo, M. Dumas, O. González-Rojas, Learning accurate LSTM models of business processes, in: Proceedings of the 17th International Conference on Business Process Management (BPM). Wien, 2019, pp. 286–302.
[13] N. Tax, I. Verenich, M. La Rosa, M. Dumas, Predictive business process monitoring with LSTM neural networks, in: Proceedings of the 29th International Conference on Advanced Information Systems Engineering (CAiSE). Essen, 2017, pp. 477–492.
[14] S. Schönig, R. Jasinski, L. Ackermann, S. Jablonski, Deep learning process prediction with discrete and continuous data features, in: Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), Setubal, 2018, pp. 314–319.
[15] E. Tello-Leal, J. Roa, M. Rubiolo, U. Ramirez-Alcocer, Predicting activities in business processes with LSTM recurrent neural networks, in: Proceedings of the 2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K), Santa Fe, 2018, pp. 1–7.
[16] A. Metzger, J. Franke, T. Jansen, Data-driven deep learning for proactive terminal process management, in: Proceedings of the 17th International Conference on Business Process Management (BPM) Industry Forum. Wien, 2019, pp. 190–201.
[17] S. Weinzierl, S. Zilker, M. Stierle, G. Park, M. Matzner, From predictive to prescriptive process monitoring: Recommending the next best actions instead of calculating the next most likely events, in: Proceedings of the 15. Internationale Tagung Wirtschaftsinformatik, Potsdam, 2020, pp. 1–5.
[18] S. Weinzierl, M. Stierle, S. Zilker, M. Matzner, A. Next Click, Recommender system for web-based service analytics with context-aware LSTMs, in: Proceedings of the 53rd Hawaii International Conference on System Sciences (HICSS), Maui, HI, 2020, pp. 1542–1551.
[19] A. Khan, H. Le, K. Do, T. Tran, A. Ghose, H. Dam, R. Sindhgatta, Memory-augmented neural networks for predictive process analytics, arXiv 1802 (2018) 00938.
[20] V. Pasquadibisceglie, A. Appice, G. Castellano, D. Malerba, Using convolutional neural networks for predictive process analytics, in: Proceedings of the 2019 International Conference on Process Mining (ICPM), Aachen, 2019, pp. 129–136.
[21] N. Di Mauro, A. Appice, T. Basile, Activity prediction of business process instances with inception CNN models, in: Proceedings of the XVIIIth International Conference of the Italian Association for Artificial Intelligence (AI*IA). Rende, 2019, pp. 348–361.
[22] R.P.W. Duin, Superlearning and neural network magic, Pattern Recogn. Lett. 15 (1994) 215–217.
[23] G. Park, M. Song, Predicting performances in business processes using deep neural networks, Decis. Support. Syst. 129 (2020), 113191.
[24] N. Navarin, B. Vincenzi, M. Polato, A. Sperduti, LSTM networks for data-aware remaining time prediction of business process instances, in: Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, 2017, pp. 1–7.
[25] I. Teinemaa, M. Dumas, M.L. Rosa, F.M. Maggi, Outcome-oriented predictive process monitoring, ACM Trans. Knowl. Discov. Data 13 (2019) 1–57.
[26] M. Hinkka, T. Lehto, K. Heljanko, A. Jung, Classifying process instances using recurrent neural networks, in: Proceedings of the 1st International Workshop on Artificial Intelligence for Business Process Management (AI4BPM). Sydney, 2018, pp. 313–324.
[27] A. Leontjeva, R. Conforti, C. Di Francescomarino, M. Dumas, F.M. Maggi, Complex symbolic sequence encodings for predictive monitoring of business processes, in: Proceedings of the 13th International Conference on Business Process Management (BPM), Innsbruck, 2015, pp. 297–313.
[28] C. Di Francescomarino, M. Dumas, M. Federici, C. Ghidini, F.M. Maggi, W. Rizzi, Predictive business process monitoring framework with hyperparameter optimization, in: Proceedings of the 28th International Conference on Advanced Information Systems Engineering (CAiSE). Ljubljana, 2016, pp. 361–376.
[29] T. Nolle, S. Luettgen, A. Seeliger, M. Mühlhäuser, BINet: Multi-perspective Business process anomaly classification, arXiv 1902 (2019) 03155.
[30] G. Park, M. Song, Prediction-based resource allocation using LSTM and minimum cost and maximum flow algorithm, in: Proceedings of the 2019 International Conference on Process Mining (ICPM), Aachen, 2019, pp. 121–128.
[31] D. Breuker, M. Matzner, P. Delfmann, J. Becker, Comprehensible predictive models for business processes, MIS Q. 40 (2016) 1009–1034.
[32] B.F. van Dongen, BPI Challenge 2012, 2012, https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f (accessed 2020-05-17).
[33] W. Steeman, BPI Challenge 2013, 2013, https://doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07 (accessed 2020-05-17).
[34] M. Le, B. Gabrys, D. Nauck, A hybrid model for business process event prediction, in: Proceedings of the 32nd International Conference on Innovative Techniques and Applications of Artificial Intelligence (SGAI), London, 2012, pp. 179–192.
[35] M. Ceci, P.F. Lanotte, F. Fumarola, D.P. Cavallo, D. Malerba, Completion time and next activity prediction of processes using sequential pattern mining, in: Proceedings of the 17th International Conference on Discovery Science (DS). Bled, 2014, pp. 49–61.
[36] G.T. Lakshmanan, D. Shamsi, Y.N. Doganata, M. Unuvar, R. Khalaf, A markov prediction model for data-driven semi-structured business processes, Knowl. Inf. Syst. 42 (2015) 97–126.
[37] M. Unuvar, G.T. Lakshmanan, Y.N. Doganata, Leveraging path information to generate predictions for parallel business processes, Knowl. Inf. Syst. 47 (2016) 433–461.
[38] I. Verenich, Helpdesk, 2016, https://doi.org/10.17632/39bp3vv62t.1 (accessed 2020-05-17).
[39] M. Hinkka, T. Lehto, K. Heljanko, Exploiting event log data-attributes in RNN based prediction, in: Proceedings of the 23rd European Conference on Advances in Databases and Information Systems (ADBIS). Bled, 2019, pp. 405–416.
[40] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, 2014, pp. 1724–1734.
[41] L. Lin, L. Wen, J. Wang, MM-Pred: A deep predictive model for multi-attribute event sequence, in: Proceedings of the 2019 SIAM International Conference on Data Mining (SDM), Calgary, 2019, pp. 118–126.
[42] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.
[43] Y.N. Dauphin, A. Fan, M. Auli, D. Grangier, Language modeling with gated convolutional networks, in: Proceedings of the 34th International Conference on Machine Learning (ICML) 70, 2017, pp. 933–941. Sydney.
[44] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770–778.
[45] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, T. Chen, Recent advances in convolutional neural networks, Pattern Recogn. 77 (2018) 354–377.
[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention Is all you Need, in: Proceedings of the 31st Conference on Advances in Neural Information Processing Systems (NeurIPS), Long Beach, CA, 2017, pp. 5998–6008.
[47] M. Daniluk, T. Rocktäschel, J. Welbl, S. Riedel, Frustratingly short attention spans in neural language modeling, in: Proceedings of the 5th International Conference on Learning Representations (ICLR), Toulon, 2017, pp. 1–10.
[48] B.F. van Dongen, Real-Life Event Logs - Hospital Log, 2011, https://doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54 (accessed 2020-05-17).
[49] J.C.A.M. Buijs, Environmental Permit Application Process ('WABO'), CoSeLoG Project – Municipality 4, 2014, https://doi.org/10.4121/uuid:e8c3a53d-5301-4afb-9bcd-38e74171ca32 (accessed 2020-05-17).

[50] R.P.W. Duin, A note on comparing classifiers, Pattern Recogn. Lett. 17 (1996) 529–536.
[51] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[52] M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Inf. Process. Manag. 45 (2009) 427–437.
[53] M. Shepperd, D. Bowes, T. Hall, Researcher bias: the use of machine learning in software defect prediction, IEEE Trans. Softw. Eng. 40 (2014) 603–616.
[54] F. Chollet, Deep Learning with Python, Manning, Shelter Island, NY, 2018.
[55] W.M.P. van der Aalst, B.F. van Dongen, C. Günther, A. Rozinat, E.M.W. Verbeek, A. J.M.M. Weijters, ProM: The process mining toolkit, in: Proceedings of the 7th International Conference on Business Process Management (BPM) Demonstration Track, Ulm, 2009, pp. 1–4.
[56] A. Klein, S. Falkner, S. Bartels, P. Hennig, F. Hutter, Fast Bayesian optimization of machine learning Hyperparameters on large datasets, in: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Lauderdale, FL, 2017, pp. 528–536.
[57] K. Heinrich, P. Zschech, C. Janiesch, M. Bonin, Supplementary Material for "Process Data Properties Matter: Introducing GCNN and KVP for Next Event Prediction with Deep Learning". http://doi.org/10.23728/b2share.08b7ff704f724b94a61b4a6cac0fe1e0, 2021 (accessed 2021-01-12).
[58] M. Hutson, Core progress in AI has stalled in some fields, Science 368 (2020) 927.
[59] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.
[60] M. Sokolova, N. Japkowicz, S. Szpakowicz, Beyond Accuracy, F-score and ROC: a family of discriminant measures for performance evaluation, in: Proceedings of the 19th Australian Joint Conference on Artificial Intelligence (AI). Hobart, 2006, pp. 1015–1021.
[61] S. Weinzierl, S. Zilker, J. Brunk, K. Revoredo, A. Nguyen, M. Matzner, J. Becker, B. Eskofier, An empirical comparison of deep-neural-network architectures for next activity prediction using context-enriched process event logs, arXiv 2005 (2020) 01194.
[62] F. Taymouri, M. La Rosa, S. Erfani, Z.D. Bozorgi, I. Verenich, Predictive business process monitoring via generative adversarial nets: The case of next event prediction, in: Proceedings of the 18th International Conference on Business Process Management (BPM), Wien, 2020, pp. 237–256.
[63] J. Wanner, K. Heinrich, C. Janiesch, P. Zschech, How much AI Do you require? Decision factors for adopting AI technology, in: Proceedings of the 41st International Conference on Information Systems (ICIS), AIS Virtual Conference, 2020, pp. 1–17.
[64] M. Harl, S. Weinzierl, M. Stierle, M. Matzner, Explainable predictive business process monitoring using gated graph neural networks, J. Decis. Syst. (2020) 1–16, forthcoming.
[65] N. Mehdiyev, P. Fettke, Explainable artificial intelligence for process mining: A general overview and application of a novel local explanation approach for predictive process monitoring, in: W. Pedrycz, S.-M. Chen (Eds.), Interpretable Artificial Intelligence: A Perspective of Granular Computing, Springer, Cham, 2021.

Kai Heinrich is an assistant professor with the Otto-von-Guericke-Universität Magdeburg. Before that, he worked as a research assistant and post-doc at the Chair of Business Informatics, especially Intelligent Systems and Services, at the Technische Universität Dresden. His research is at the intersection of artificial intelligence, decision support systems, and human-computer interaction. Research core topics include the design of AI-based decision support systems and the study of interactions between humans and AI-based systems. Common fields of application include industrial application fields like manufacturing, maintenance, or agrarian management as well as health care and finance. He has authored scholarly publications in leading international journals in the field of information systems such as Business & Information Systems Engineering, International Journal of Intelligent Information Technologies as well as in various conference proceedings including ICIS, ECIS, AMCIS, and HICSS.

Patrick Zschech is an assistant professor with the Friedrich-Alexander-Universität Erlangen-Nürnberg. Before that, he worked as a research assistant and post-doc at the Chair of Business Informatics, especially Intelligent Systems and Services, at the Technische Universität Dresden. Additionally, he worked for the IT service provider Robotron Datenbank-Software GmbH as a project member and an instructor for data science qualification programs. In his research, Patrick focuses on the selection, evaluation, and application of data-driven methods for the development of analytical information systems. His main interests are in the areas of machine learning, computer vision, process mining, and industry 4.0. Patrick regularly publishes his research results in international journals and conferences related to the field of decision support and information systems, such as Business & Information Systems Engineering, HMD Praxis der Wirtschaftsinformatik, Controlling & Management Review as well as in various conference proceedings including ICIS, ECIS, and WI.

Christian Janiesch is assistant professor with the Julius-Maximilians-Universität Würzburg. Before, Christian worked full-time at the Westfälische Wilhelms-Universität in Münster, at the SAP Research Center Brisbane at SAP Australia Pty Ltd., and at the Karlsruhe Institute of Technology. His research is at the intersection of business process management and business analytics with frequent applications in the Industrial Internet of Things. He is on the Department Editorial Board for BISE and has authored over 150 scholarly publications. His work has appeared journals such as the Journal of the Association for Information Systems, Communications of the Association for Information Systems, Information & Management, Business & Information Systems Engineering, Future Generation Computer Systems, Information Systems as well as in various major international conferences including ICIS, ECIS, BPM, and HICSS and has been registered as U.S. patents.

Markus Bonin completed his master's degree in Business Informatics with a focus on Business Intelligence and Data Sciences at the Technische Universität Dresden. His recent research work focuses on the application of deep learning architectures in the field of process prediction. Currently, he works at Capgemini Deutschland GmbH and designs an advanced analytics platform for the industrialization of data science solutions. Besides, he uses machine learning to solve natural language processing problems.