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

Johannes Lahann(✉), Peter Pfeiffer, and Peter Fettke

German Research Center for Artificial Intelligence (DFKI) and Saarland University, Saarbrücken, Germany
{johannes.lahann,peter.pfeiffer,peter.fettke}@dfki.de

Abstract. Anomaly detection can identify deviations in event logs and allows businesses to infer inconsistencies, bottlenecks, and optimization opportunities in their business processes. In recent years, various anomaly detection algorithms for business processes have been proposed based on either process discovery or machine learning algorithms. While there are apparent differences between machine learning and process discovery approaches, it is often unclear how they perform in comparison. Furthermore, deep learning research in other domains has shown that advancements did not solely come from improved model architecture but were often due to minor pre-processing and training procedure refinements. For this reason, this paper aims to set up a broad benchmark and establish a baseline for deep learning-based anomaly detection of process instances. To this end, we introduce a simple LSTM-based anomaly detector utilizing a collection of minor refinements and compare it with existing approaches. The results suggest that the proposed method can significantly outperform the existing approaches on a large number of event logs consistently.

Keywords: Business process management · Anomaly detection · Deep learning · LSTM

1 Introduction

Anomaly detection deals with the identification of rare articles, objects, or observations that differ significantly from the majority of the data and therefore raise suspicions [16]. In the context of business process analysis, businesses apply anomaly detection to automatically detect deviations in event logs which can be a sign of inconsistencies, bottlenecks, and optimization opportunities in their business processes [7]. A typical approach to detect anomalous behavior in business processes is to apply conformance checking [12], i.e., evaluating the real occurred behavior that is recorded in event logs against the business process model that business experts previously designed. However, to do this, such a process model is required beforehand. More recently, a variety of deep learning-based anomaly detection algorithms with different architectures have been developed that are able to identify anomalous process behavior without requiring a process model or other prior knowledge about the underlying process. While there are apparent differences between the existing approaches, it is not clear how they perform in comparison. Furthermore, deep learning research in other domains has shown that advancements did not solely come from improved model architecture but are often due to minor training procedure refinements [6]. Thus, this paper aims to set up a broad benchmark between anomaly detection algorithms where we compare the performance of existing approaches with a simple LSTM-based anomaly detector that utilizes a number of minor refinements. The contribution of this paper is threefold:

- We examine a collection of different processing, model architecture, and anomaly score computation refinements that lead to significant model accuracy or run-time improvements.
- We show that the proposed methods lead to a significant performance improvement in comparison with state-of-the-art process mining-based and deep learning-based anomaly detection methods. To this end, we conduct experiments on the data sets from the Process Discovery Contests, the Business Process Intelligence Challenges, and additional synthetic event logs [7].
- We set up a comprehensive evaluation over a total of 328 different event logs, which can be utilized as a benchmark for further research.

The remaining sections of the paper unfold as follows: Sect. 2 introduces the reader to preliminary ideas of process mining and predictive process monitoring. Section 3 gives a brief overview of the approach before it discusses the applied refinements. Section 4 describes two experiments to evaluate the performance of the proposed approach. Section 5 shows the evaluation results covering an overall performance comparison with existing methods and a detailed analysis of the impact of different design decisions and refinements. Section 6 relates the developed approach to existing literature. Section 7 closes the paper with a summary of the main contributions and an outline of future work.

2 Preliminaries

This section introduces some preliminary concepts. In particular, we introduce the concepts of events, cases, and event logs and define next step prediction and (case-level) anomaly detection as we understand it during the scope of this paper.

Definition 1. *Event, Case, Event Log*
Let E be the universe of events. A case σ is a finite-length word of events, i.e. $\sigma \in E^* \wedge |\sigma| = n$, $n \in \mathbb{N}$. An event log is a multi-set of cases, i.e. $L \in \mathbb{B}(E^*)$.

To describe a case σ, we also use the notation $\sigma := \langle e_1, \ldots, e_n \rangle$. There are further attributes next to the activity associated with events such as resource, timestamp, and others. These attributes can add additional information that can also be utilized for analysis and predictive tasks.

One process prediction task that has been researched intensively in recent years and also plays a major role in the proposed anomaly detection approach in this paper is next step prediction. Next step prediction aims to forecast the direct continuation of an ongoing process instance based on all available information regarding the process instance. We define next step prediction as follows:

Definition 2. *Next Step Prediction*
Given a prefix $p_t = \langle e_1, ..., e_t \rangle$ *of a case* $\sigma = \langle e_1, ..., e_n \rangle$ *with* $0 <= t < n,\ t, n \in \mathbb{N}$*, we define Next Step Prediction as a relation* $NSP \subseteq E^* \times E$ *that predicts the next occurring event* e_{t+1} *based of the prefix* p_t*.*

Next, we can define anomaly detection of process instances. There is a distinction between attribute and case-level anomaly detection in the literature. While the former detects irregular attribute values on event-level, such as false activities, resources, or timestamps, the latter aims to classify anomalous cases. For the scope of this paper, we are only concerned with case-level anomaly detection, which we conceptualize as follows:

Definition 3. *Case-level Anomaly Detection*
We define a case-level anomaly detector as a function f *that receives a case* σ *and returns a label* $\ell \in \{0, 1\}$*, where 0 indicates a normal case and 1 indicates an anomalous case.*

One may notice that we do not specify what makes a case normal or anomalous. We argue that depending on the context, the criteria for an anomaly may differ. Hence, a more vague definition is beneficial. In the first conducted experiment, we understand anomaly detection similarly to conformance checking, i.e., a case is normal if it fits a hidden process model; else, it is anomalous. In the second experiment, synthetic events and attributes are injected into the data sets based on a predefined rule-set. A case is considered anomalous if it contains at least one of the injected values.

3 Proposed Approach

3.1 Overview

The proposed method investigates prediction-based anomaly detection with a deep neural network as the predictive model. The approach can be divided into two stages - first, we train an LSTM-based model to learn the behavior of the process, while in the second stage, the trained model is used to assess whether a given trace is anomalous or not. In the first stage, we train the prediction model to solve the next step prediction task. The idea is to teach the model a hidden representation that contains the most relevant information to predict the possible next events. To assess whether a trace σ is anomalous or not, we use the trained model to predict all the steps of a given case. If the predicted behavior of the neural network and the real behavior differ significantly in at least one of the events, we consider this observation a strong indicator that the case is suspicious. Therefore, we mark the case as anomalous. We introduce DAPNN (Detection of Anomalous Processes through Neural Networks), which utilizes a collection of changes and refinements to previous work [8,9] that together led to significant performance improvements in the conducted experiments. We generated fixed sliding windows and switched to a LSTM-based network architecture. Furthermore, we used multiple training methods to improve the convergence of the neural networks. Last, we added normalization to the anomaly score computation, which creates a comparable anomaly score throughout different event logs.

3.2 Approach Characteristics and Refinements

Data Processing. DAPNN is trained on windows extracted from the cases σ of size w. Given a window of the $w-1$ previous events, DAPNN's task is to predict the last event in the window. Thereby, we do not have to insert padding elements to counteract the different lengths of the prefixes. Furthermore, since the window size is usually much smaller than the maximum length of the prefixes, this results in a much faster training time. For a case $\sigma := \langle e_1, \ldots, e_n \rangle$ and a fixed window size w, we generate n-w windows $\langle e_t - w, e_t \rangle$, where $w < t \leq n$. In the conducted experiments, we used a fixed window size of 5. Next, we add special *Start* and *End* events to each case in the event log. Thereby, the next step prediction model can also learn to predict the beginning and the end of a case. This is especially effective since there are anomalous cases that only behave wrongly at the beginning or at the end.

Model Architecture and Training. We decided to use a simple LSTM-based architecture. Each case σ is split into separate sequences along the attributes, which are processed by individual LSTM blocks. Each block consists of an embedding layer, two LSTM-layer with hidden layer size 25, followed by a softmax layer. This allows obtaining a probability distribution $\vec{p}$ per attribute found in the event log, which serves as the basis to assess whether σ is anomalous or not.

We train each neural network for up to 25 epochs utilizing early stopping, the learning rate finder, and cyclic learning rates [13]. While early stopping primarily reduces training time, we see a significant improvement in the robustness of the results through the latter two methods throughout the conducted experiments.

Anomaly Score Computation. After training the prediction model, we can utilize it to detect anomalies. To do this, we compute all windows for a given case σ and feed them through the prediction model. For a case with n events and m attributes, we compute $m \times n$ probability distributions $\vec{p}$. In order to obtain the anomaly scores, we apply a scoring function Θ and store the anomaly scores per case in a matrix $M_{anomaly}$. We define Θ as follows:

$$\Theta(\vec{p}, y) = \frac{max(\vec{p}) - p_y}{max(\vec{p})}$$

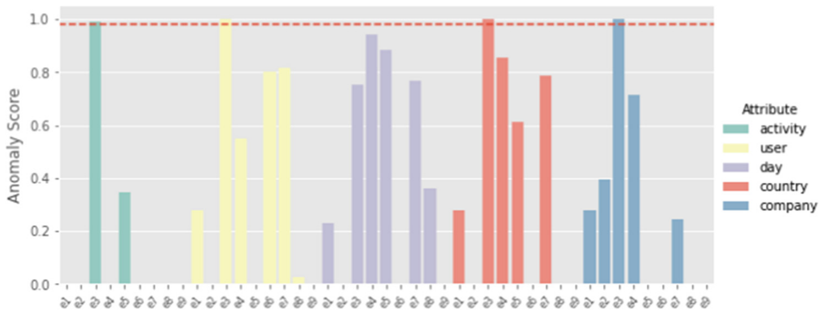


Fig. 1. Illustration of the anomaly scores of a case that resembles a skip sequence anomaly. For the 3rd predicted event, the threshold is exceeded for 4 out of 5 attributes.

y depicts the actual next occurred attribute in σ, and p_y represents the probability that the prediction model is assigned to the attribute y. The margin of $max(\vec{p})$ and p_y can be interpreted as a measure of certainty for an anomaly. If the margin is high, the prediction model is certain that another attribute should occur instead. Hence, this is a sign of an anomaly. By normalizing with $max(\vec{p})$, we make the anomaly score more robust so that it behaves similarly throughout all predictions. Additionally, it penalizes deviations stronger if it has low confidence regarding the occurred value. For example, if the predicted event has a probability of 0.75 and the occurred event has a probability of 0.25, the obtained anomaly score is $(0.75-0.25)/0.75 = 0.66$. However, if the predicted event has a probability of 0.5 and the occurred event has a probability of 0.0, the obtained anomaly score is $(0.5-0.0)/0.5 = 1.0$. The normalization pushes anomalies near 1.0 and enables easier differentiation between anomalies and normal events. Furthermore, it allows us to introduce a threshold that functions similarly to a significance measure, as the threshold is relatively stable over different event logs. Figure 1 shows the resulting anomaly scores for one particular case.

Anomaly Classification. Based on the anomaly scores, we can then determine if a case is anomalous, i.e., we define a function f that takes all anomaly scores M of a case and a threshold τ as input and outputs a label $l \in 0, 1$.

$$f(M_{anomaly}, \tau) = \begin{cases} 1, & \text{if } max(M_{anomaly}) > \tau \\ 0, & \text{otherwise} \end{cases}$$

The intuition behind the formula is that if a case contains at least one anomaly score greater or equal to the given threshold, it is flagged as an anomaly. In order to choose a suitable threshold, we compare different options:

– *Best Threshold*: we select the optimal threshold based on the achieved F1-score on the test set. I.e., we compute the F1-Score for all possible thresholds

Table 1. Data sets of experiment 1.

	# 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

and choose the threshold with the highest F1-Score. Note that this heuristic requires labels and thus is not applicable in practice in an unsupervised scenario. However, it is still relevant as it allows us to measure the maximal achievable performance with the underlying prediction model.
- *Fixed Threshold:* we set a fixed threshold that we use throughout all experiments. We achieved reasonable results with a threshold of 0.98.
- *Anomaly Ratio*: we pick a threshold based on the total number or the ratio of predicted anomalies.
- *Elbow and Lowest Plateau Heuristic*: we utilize heuristics based on the anomaly ratio per potential threshold as introduced in [8].

4 Experimental Setup

4.1 Experiment 1

The first experiment compares the performance of the proposed anomaly detection approach with process discovery algorithms on the Process Discovery Contests 2020 and 2021 [14,15]. The process discovery contest (PDC) aims to assess tools and techniques that discover business process models from event logs. To this end, synthetic data sets are generated that comply with general concepts that influence process mining algorithms.

While the process discovery is designed to evaluate process discovery algorithms, it measures their performance indirectly through a classification task, identifying process cases that fit a hidden process model. Hence this task can also be accomplished through anomaly detection. Regarding the experimental setup, we follow the instructions from the process discovery contest. In particular, we consider the data sets from PDC 2020 and PDC 2021. Table 1 highlights the most important characteristics and statistics about the data sets. To achieve maximal comparability with the other algorithms that took part in the challenges, we also trained the next step prediction model on the training logs and measured the performance on the test sets.

4.2 Experiment 2

The second experiment provides a comparison with other machine learning-based anomaly detection approaches on the data sets generated by Nolle et al. [8].The synthetic event logs are based on six process models with a different number of activities, model depths, and model widths, which are created randomly with the PLG2 framework [5]. Additionally, the authors utilized the event logs from the BPI Challenges 12, 13, 15, and 17. Subsequently, a variety of artificial anomalies was added to some of the cases of all event logs (Table 2):

– Skip: One or multiple events are skipped.
– Insert: Random events are inserted.
– Rework: Events are executed multiple times.
– Late: Events are shifted forward.
– Early: Events are shifted backward.
– Attribute: Other attribute values of some events are altered.

Table 2. Data sets of experiment 2.

	# 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

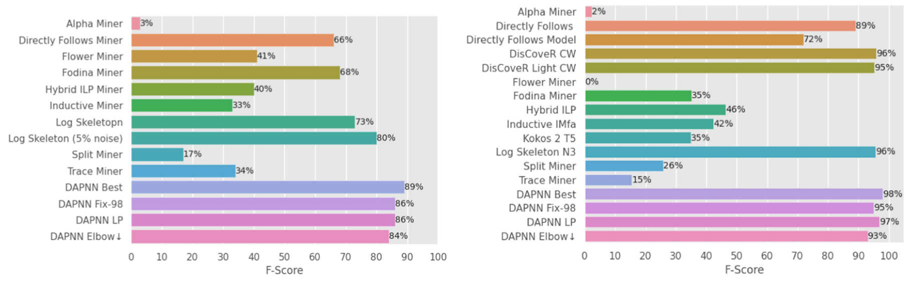
4.3 Evaluation Metrics

In order to evaluate the performance of the approach, we use the F1 score, which is a common choice for evaluating anomaly detection. The F1 score is computed by the harmonic mean of precision and recall. The precision measures how precisely anomalies can be identified, i.e., how many of the predicted anomalies are actual anomalies. The recall measures how many anomalies are identified and how many anomalies are not recognized by the model:

$$F1\text{-}Score = \frac{2 * (precision * recall)}{(precision + recall)}$$

To comply with the specifications of the Process Discovery Contest and achieve comparability with the existing methods, we use an adapted version of the F1-Score in experiment 1, which is calculated by the balanced mean of the true positive rate *tpr* and the true negative rate *tnr*:

$$F\text{-}Score = \frac{2 * (tpr * tnr)}{(tpr + tnr)}$$



(a) PDC 2020 (b) PDC 2021

Fig. 2. Comparison by F-Score of the *DAPNN* approach with existing approaches extracted from the PDC website.

4.4 Reproducibility

All code used for this paper, including the implementation of *DAPNN* as well as the quantitative comparison with traditional and machine learning-based anomaly detection approaches, is available in our git repository[1].

5 Results

5.1 Overall Performance on the Process Discovery Contest

Figure 2 shows the performance of the *DAPNN* approach in comparison with existing process discovery algorithms as described in experiment 1. In PDC 2020, the *DAPNN* approach reaches an F-Score of 89% with the optimal heuristic, outperforming all other existing methods. Moreover, the *DAPNN* models with the other heuristics do not perform significantly worse. In PDC 2021, *DAPNN Best* and the *DAPNN LP* reach the highest F-Score with 98% and 97% respectively. The *DAPNN Fix-98* performs similarly to the DisCoverR CW, the DisCoveR Light CW, and the Log Skeleton N3 model. Since the latter models have not been applied to the PDC 2020, it would be interesting to see how they compare with the *DAPNN* approach. The results suggest that the *DAPNN* approach can effectively identify the non-fitting cases in the PDC contests and is able to reach state-of-the-art performances. The DAPNN approach can not be used straightforwardly for process discovery as it does not directly output a process model. However, they seem to be superior in the detection of cases that do not fit the underlying process.

5.2 Overall Performance in Comparison with Other Anomaly Detection Approaches

Table 3 presents the results of experiment 2. It compares the F1-Score of 19 approaches on 40 event logs. Note that the event logs are grouped together as

[1] https://github.com/jolahann/dapnn.

Table 3. Comparison by F1-Score of the *DAPNN* approach with existing unsupervised anomaly detection approaches extracted from [8].

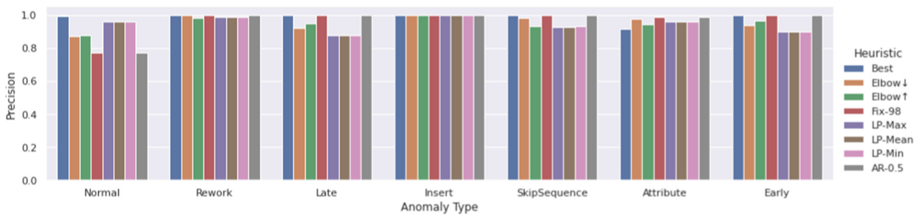
	BPIC12	BPIC13	BPIC15	BPIC17	Gigantic	Huge	Large	Medium	P2P	Paper	Small	Wide	Mean
Likelihood	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
OC-SVM	0.545	0.243	0.255	0.351	0.291	0.228	0.237	0.289	0.271	0.486	0.248	0.306	0.312
Naive	0.551	0.209	0.172	0.313	0.34	0.404	0.41	0.387	0.479	0.5	0.49	0.438	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

shown in Table 2 highlighting the mean F1-Score over the event logs of one data group. For example, the column *BPIC13* reports the mean F1-Score over all three event logs of the BPIC 2013. We reported the performance of the *DAPNN* approach with all heuristics. However, for the other approaches, only the performance with the *LP-Mean* heuristic is reported. The DAPNN approach reached top results on all examined event logs. In terms of the heuristics, the LP heuristics achieved better results than the elbow heuristics, followed by the anomaly ratio and the fixed threshold. Additionally, $DAPNN_{Best}$ reached a very high F1-Score for all synthetic event logs. This suggests that the prediction model is able to correctly separate anomalous and normal cases by assigning a higher anomaly score to anomalous events for most of the cases. However, the determination of the correct threshold is still a major challenge, as the $DAPNN_{LP-Max}$ with the second highest mean F1-Score performs significantly worse than the $DAPNN_{Best}$.

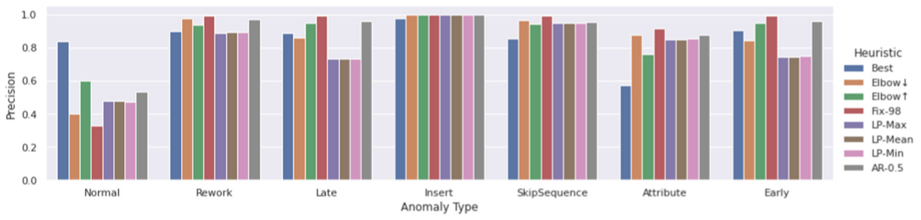
The results also show a clear performance gap between the synthetic event logs and the event logs of the BPI challenges. This can be explained by two reasons: On the one hand, the algorithms are only asked to find the artificial anomalies. However, it is unclear whether and how many unknown anomalies were already included in the original event logs that are not labeled as such. On the other hand, it might be the case that the synthetic event logs cover processes with simpler characteristics. In contrast, the processes of the BPI challenges are more difficult to comprehend for the approaches.

5.3 Detection of Anomaly Types

Given that we have different anomaly types, one question is how well the model can classify each anomaly type. Figure 3 compares the precision of the *DAPNN* with each heuristic for each anomaly type for the datasets from the second experiment. Aside from the best heuristic, there is no clear winner recognizable (Note that for multi-attribute event logs, we approximated the best heuristic with Naive Bayes optimization. Hence it is only a lower bound for the actual best score and can, in some cases, be lower than the scores of the other heuristics). The Elbow, Fix-98, and AR-0.5 heuristics tend to produce more false positives but have a slightly higher precision while detecting the anomalies. In contrast, the LP heuristics produce fewer false positives. Thus, the heuristics should be chosen based on the requirements of the business scenario.



(a) Synthetic Event Logs



(b) BPI Challenge Logs

Fig. 3. Detection precision of the *DAPNN* approach for each anomaly type and heuristic

6 Related Work

Originally, anomaly detection on business process data was performed by evaluating process cases captured in an event log against a predefined process model [12]. However, this requires a reference model of the underlying process, which is not always available. To overcome this problem, Bezerra et al. define an anomalous case as an irregular execution that differs from a process model that was dynamically discovered by a process discovery algorithm [1]. The approach follows the hypothesis that anomalous cases are rare and differ significantly from normal cases. Therefore, the process discovery algorithm will focus on modeling the normal cases. Hence, the mined process model will require considerable modifications in order to fit anomalous cases leading to a high alignment score. According to this idea, the authors propose an anomaly detection approach that samples process cases from a discovered process model. If a case in the original event log does not correspond to one of the sampled cases, it is flagged as an anomaly. Building on this, Bezerra et al. introduce two parameters, fitness model degree and appropriateness of a process model, in order to formalize the degree of an anomaly [3]. Furthermore, they introduce two new variants of their anomaly detection approach, including a threshold and an iterative version [2]. Both of the approaches make use of the conformance fitness of each case according to the discovered model. Similarly, in the Process Discovery Contest, the detection of anomalous cases is used to measure the quality of the process discovery approaches [14,15]. Each process discovery approach is first trained on a training event log before assessing the F1-Score over a test log with normal and anomalous process cases, i.e., process cases that fit or do not fit a hidden process model.

More recently, a variety of model-less anomaly detection approaches have been developed that are able to detect anomalous process behavior without requiring an explicit process model. Böhmer et al. proposed a multivariate technique that builds up an extended likelihood graph on multiple event attributes in order to identify the anomalies [4]. Nolle et al. introduced three different deep learning-based anomaly detection approaches. In [7] they proposed a deep autoencoder to capture anomalous process cases. First, an autoencoder is trained by mapping each process case to itself. Afterward, the reconstruction error is calculated for each case. If the reconstruction error succeeds a predefined threshold, the case is flagged as an anomaly. Then, the same authors proposed BINET, which consists of a next step prediction model and a heuristic [8]. The heuristic determines if the deviation of the model predictions is a significant sign of a potential anomaly. Last, the same authors proposed DeepAlign [10], an extension of the previous approach that can also be used to correct the anomalous process behavior. The core components of the model are bidirectional LSTMs and beam search. Finally, Pauwels et al. developed an anomaly detection method based on Bayesian Networks [11].

7 Conclusion

This paper analyzed multivariate anomaly detection for detecting anomalous process instances (case-based anomaly detection) through LSTM neural networks. We showed that by various refinements in terms of data processing, neural network architecture, and anomaly score computation, we could improve the anomaly detection quality significantly. We evaluated the proposed approach against existing approaches on 328 different real-life and synthetic event logs. We were able to improve the mean F-Score on the PDC 2020 by 6% and the PDC 2021 by 2.3%. In comparison with the machine learning-based models, we achieved a performance gain of 26.1%. Additionally, the paper provides a benchmark for anomaly detection of process cases and can serve as a baseline for further research. In the future, we plan to investigate which design decisions lead to the highest performance improvements and which process features and anomalous behaviors are most difficult for neural networks to understand. Furthermore, we want to extend the anomaly score computation to support continuous attributes.

References

1. Bezerra, F., Wainer, J.: Anomaly detection algorithms in logs of process aware systems. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 951–952 (2008)
2. Bezerra, F., Wainer, J.: Algorithms for anomaly detection of traces in logs of process aware information systems. Inf. Syst. **38**(1), 33–44 (2013)
3. Bezerra, F., Wainer, J., van der Aalst, W.M.P.: Anomaly detection using process mining. In: Halpin, T., et al. (eds.) BPMDS/EMMSAD -2009. LNBIP, vol. 29, pp. 149–161. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01862-6_13
4. Böhmer, K., Rinderle-Ma, S.: Multi-perspective anomaly detection in business process execution events. In: Debruyne, C., et al. (eds.) OTM 2016. LNCS, vol. 10033, pp. 80–98. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48472-3_5
5. Burattin, A.: PLG2: multiperspective process randomization with online and offline simulations. In: Azevedo, L., Cabanillas, C. (eds.) Proceedings of the BPM Demo Track 2016 Co-located with the 14th International Conference on Business Process Management (BPM 2016), Rio de Janeiro, Brazil, 21 September 2016. CEUR Workshop Proceedings, vol. 1789, pp. 1–6. CEUR-WS.org (2016)
6. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks (2018). https://doi.org/10.48550/ARXIV.1812.01187. https://arxiv.org/abs/1812.01187
7. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Analyzing business process anomalies using autoencoders. Mach. Learn. **107**(11), 1875–1893 (2018). https://doi.org/10.1007/s10994-018-5702-8
8. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: BINet: multi-perspective business process anomaly classification. Inf. Syst. **103**, 101458 (2022)
9. Nolle, T., Seeliger, A., Mühlhäuser, M.: BINet: multivariate business process anomaly detection using deep learning. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 271–287. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_16
10. Nolle, T., Seeliger, A., Thoma, N., Mühlhäuser, M.: DeepAlign: alignment-based process anomaly correction using recurrent neural networks. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds.) CAiSE 2020. LNCS, vol. 12127, pp. 319–333. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49435-3_20
11. Pauwels, S., Calders, T.: An anomaly detection technique for business processes based on extended dynamic Bayesian networks. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp. 494–501 (2019)
12. Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
13. Smith, L.N.: Cyclical learning rates for training neural networks (2017)
14. Verbeek, E.: Process discovery contest 2020 (2021)
15. Verbeek, E.: Process discovery contest 2021 (2021)
16. Zimek, A., Schubert, E.: Outlier detection. In: Liu, L., Ozsu, M. (eds.) Encyclopedia of Database Systems. Springer, Cham (2017). https://doi.org/10.1007/978-1-4899-7993-3_80719-1