CaLenDiR: Mitigating Case-Length Distortion in Deep-Learning-Based Predictive Process Monitoring

Brecht Wuyts[1][0000−0001−6079−7515],
Seppe vanden Broucke[2,1][0000−0002−8781−3906], and Jochen De Weerdt[1][0000−0001−6151−0504]

[1] LIRIS, Faculty of Economics and Business, KU Leuven, Leuven, Belgium
[2] Department of Business Informatics and Operations Management, Ghent University
brecht.wuyts@kuleuven.be

Abstract. Predictive Process Monitoring (PPM) in Process Mining (PM) focuses on forecasting future aspects of ongoing business processes. Recently, Deep Learning (DL) models have emerged as top performers across various prediction tasks. However, prevalent practices in instance creation lead to *case-length distortion*, a problem where longer cases are overrepresented, biasing the training process towards these longer cases and distorting evaluation by skewing performance metrics, potentially leading to misleading assessments. To address this challenge, we introduce the *CaLenDiR* (Case Length Distribution-Reflective) framework for DL-based PPM, which aligns training and evaluation with the true case length distributions. *CaLenDiR training* includes *Uniform Case-Based Sampling (UCBS)* to ensure balanced case contributions and, for suffix prediction, employs *suffix-length-normalized loss functions* to prevent further exacerbation of training distortion. For evaluation, the framework proposes *case-based metrics* as an alternative to traditional, distorted instance-based metrics. Our extensive experiments demonstrate *CaLenDiR*'s effectiveness in improving model robustness and generalization, offering new insights into the interaction between log characteristics and model behavior.

Keywords: Process Mining · Predictive Process Monitoring · Deep Learning · Case-Length Distortion.

1 Introduction

Predictive Process Monitoring (PPM) focuses on developing techniques for the prediction of remaining runtime (e.g. [1,3,9,13]), outcome (e.g. [4,6,12,14]), next event (e.g. [1,2,9]), or even the entire suffix (e.g. [1,3,5,9,11,15]).

Despite the success of these DL-based techniques [3, 6, 9, 13], an issue we term *case-length distortion* arises from the prevalent approach to instance creation, where the overrepresentation of longer cases *(1) during training* reduces the models' generalization ability over the true underlying distribution of cases and their lengths, while *(2)* skewing the performance metrics *during evaluation*, in turn leading to a potentially misleading assessment of model performance. In the specific task of suffix prediction, this distortion is further amplified by non-normalized loss functions, which also overemphasize longer cases. To eliminate case-length distortion, we introduce the *CaLenDiR* (Case Length Distribution-Reflective) PPM framework, specifically designed for DL techniques and comprised of two primary components. First, *CaLenDiR training*, which employs *Uniform Case-Based Sampling (UCBS)*—applicable across all PPM prediction tasks—to ensure a balanced contribution of training instances from each case. For suffix prediction, *CaLenDiR* training further incorporates *Suffix-Length-Normalized Loss Functions* to mitigate the overemphasis on longer cases. This training approach aims to enhance the generalization ability and robustness of DL-based models. *Second*, beyond training, *CaLenDiR* introduces *Case-Based (CB) metrics* as an alternative to traditional, skewed Instance-Based (IB) metrics, ensuring that performance metrics accurately reflect the models' ability to generalize across the actual distribution of cases.

Extensive experiments applied to suffix prediction not only highlight the effectiveness of our framework in improving the generalization capabilities and robustness of DL-based techniques across various prediction tasks, but also delve deeply into the interplay between log characteristics, model properties, and prediction performance. By analyzing results against varying log peculiarities and model features, we uncover further insights and raise critical questions about the components essential for effective suffix prediction. This thorough analysis also explores the impact of these factors on evaluation setups, providing a nuanced understanding of how event log characteristics and model design influence overall performance.

2 Background & Related Work

2.1 DL-based PPM & Event Log Data

DL-based PPM Most DL-based PPM techniques rely on Long Short-Term Memory (LSTM) network. In outcome prediction, Hinkka et al. [4] were among the first to explore Recurrent Neural Networks (RNN), including LSTM variants. Weytjens et al. [14] demonstrated that Convolutional Neural Networks (CNNs) are a fast, competitive outcome prediction alternative to LSTMs. For runtime prediction, Navarin et al. [7] introduced an LSTM model for direct remaining time estimation. Tax et al. [9], and Camargo et al. [1] used LSTM networks for multi-task next-event prediction, targeting both activity labels and timestamps, with [1] also predicting roles. These trained models were further applied to suffix and remaining time predictions via external feedback loops. More recently, Philipp et al. [8] pioneered the use of Transformer components for next-event prediction, signaling a shift to more advanced architectures. More recently, a number of DL-based techniques for the particularly challenging task of suffix prediction have been proposed. Since our *CaLenDiR* framework is evaluated within this context, further details on DL-based suffix prediction techniques are provided in Sect. 2.2.

PPM neural networks, though varied in architecture, share a common training method. Data is processed in small subsets known as *batches*, with the model's predictions compared to actual targets using a *loss function*. The model's parameters are adjusted to minimize this error, and this process repeats across all batches in an *epoch*. Multiple epochs allow the model to refine its predictions and improve accuracy by repeatedly processing the dataset.

Event Log Data An *event log* $\boldsymbol{L} = \{\boldsymbol{\sigma}_i | 1 \leq i \leq |\boldsymbol{L}|\}$ records cases $\boldsymbol{\sigma}_i$ that represent a sequence of chronologically ordered events $\langle \boldsymbol{e}_{i,1},...,\boldsymbol{e}_{i,n_i}\rangle$, with n_i being the number of events executed for that particular case. For notational simplicity, unless explicitly needed, the subscript i referring to the case is omitted. An *event* is a tuple $\boldsymbol{e}=(a,c,t,f_1,...,f_m)$ with a the activity label, c the case ID, t the timestamp, and $f_1,...,f_m$ (with $m \geq 0$) the potential case and event features. All elements comprising the event tuple $\boldsymbol{e}$ can be accessed individually, and are denoted by means of the subscript of the event. E.g., the activity label of the j-th event $\boldsymbol{e}_j$ ($j \in \{1,...,n\}$) is denoted by a_j, while the timestamp of that same event is denoted by t_j.

Moreover, to enable algorithms to interpret (and predict) timestamps, it is necessary to convert them into a numerical proxy. As such, they are converted into the numerical feature *time elapsed since the previous event* $t^p_j = t_j - t_{j-1}$ ($\forall j = 2,...,n$), capturing the absolute amount of time elapsed since the previous event. For the first event $\boldsymbol{e}_1$, $t^p_1 = 0$.

PPM techniques are trained and evaluated on historical data. To do so, event log $\boldsymbol{L}$ is subdivided into a training log $\boldsymbol{L}_{train}$ ($\subset \boldsymbol{L}$) and test log $\boldsymbol{L}_{test}$ ($\subset \boldsymbol{L}$). Afterwards, for both $\boldsymbol{L}_{train}$ and $\boldsymbol{L}_{test}$, each case $\boldsymbol{\sigma}_i$ is parsed into n_i prefix-target pairs or instances $\left\{(\boldsymbol{\sigma}^p_{i,k}, \boldsymbol{y}_{i,k}) | 1 \leq k \leq n_i\right\}$ (e.g. [1, 3, 7, 12–15])[3] ultimately resulting in the training and test set of instances, $\boldsymbol{N}_{train}$ & $\boldsymbol{N}_{test}$. The prefix $\boldsymbol{\sigma}^p_{i,k} = \langle \boldsymbol{e}_{i,1},...,\boldsymbol{e}_{i,k}\rangle$ contains the first k events, mimicking a real-life unfinished case, serving as the input, while the target $\boldsymbol{y}_{i,k}$ encapsulates the ground-truth prediction target(s), the form of which depends on the specific prediction task at hand.

For instance, in remaining time prediction, the target $\boldsymbol{y}_k = r_k$ represents the total remaining runtime from the last observed prefix event $\boldsymbol{e}_k$ until the case completion $\boldsymbol{e}_n$, calculated as $r_k = t_n - t_k$. For next event

[3] Other approaches (e.g. [5, 9, 11]) slightly deviate, and construct $n_i - 1$ instances, with $2 \leq k \leq n_i$

prediction, the target $\boldsymbol{y}_k = a_{k+1}$ is the activity label of the event that directly follows the last observed prefix event $\boldsymbol{e}_k$ (e.g. [2]). Alternatively, other next event prediction approaches (e.g., [1,9,10]) opt for the multi-task target $\boldsymbol{y}_k = (a_{k+1}, t^p_{k+1})$, simultaneously predicting the next event's activity label and its timestamp (proxy). In (binary) outcome prediction, the target $\boldsymbol{y}_k = o$ with $o \in \{0,1\}$ represents a binary label indicating a particular outcome of the process. Note that the subscript k is omitted since true label o does not depend on event index k. Lastly, in suffix prediction, a commonly used (multi-task) target $\boldsymbol{y}_k$ is the sequence $\langle (a_{k+1}, t^p_{k+1}), ..., (a_n, t^p_n), (EOS) \rangle$, which includes the remaining activity labels and their corresponding timestamps, i.e. the activity and timestamp suffix, with an *End Of Sequence* (EOS) token added during preprocessing to denote the end of a case. Additionally, many suffix prediction techniques produce an additional scalar remaining runtime prediction $\hat{r_k}$, either directly (e.g. [3,15]), or indirectly (e.g. [1,5,9,11]).

2.2 Suffix Prediction

The majority of (DL-based) suffix prediction techniques are developed to jointly predict the activity suffix $\langle a_{k+1}, ..., a_n, EOS \rangle$, timestamp suffix $\langle t^p_{k+1}, ..., t^p_n \rangle$ and remaining runtime r_k, when being presented with a prefix $\boldsymbol{\sigma}^p_k$. As such, the multi-task target $\boldsymbol{y}_k$ comprises two sequences and one scalar target.

All suffix prediction networks are trained using a multi-task loss function for joint optimization. Most use an additive loss function, summing the individual loss functions for each target. Commonly used loss functions for these targets include:

1. **Activity Suffix Prediction - Categorical Cross Entropy**: Let $\hat{a}_{j,t}$ be the predicted probability for the t-th activity label in the suffix of the j-th instance, and $a_{j,t}$ be the true activity label. The cross-entropy loss for the activity suffix is given by:
$$\mathcal{L}_{\text{activity}} = -\frac{1}{\sum_{j=1}^{B} N_j} \sum_{j=1}^{B} \sum_{t=1}^{N_j} \sum_{c=1}^{C} a_{j,t,c} \log(\hat{a}_{j,t,c}) \tag{1}$$
where B is the batch size, N_j is the number of events in the (ground-truth) suffix of the j-th instance, and C is the number of possible activity labels.
2. **Timestamp Suffix Prediction - Mean Absolute Error (MAE)**: Let $\hat{t}^p_{j,t}$ be the predicted timestamp for the t-th event in the suffix of the j-th instance, and $t^p_{j,t}$ be the true timestamp. The MAE loss for the timestamp suffix is given by:
$$\mathcal{L}_{\text{timestamp}} = \frac{1}{\sum_{j=1}^{B} N_j} \sum_{j=1}^{B} \sum_{t=1}^{N_j} |\hat{t}^p_{j,t} - t^p_{j,t}| \tag{2}$$
3. **Remaining Runtime Prediction - Mean Absolute Error (MAE)**: Let $\hat{r}_j$ be the predicted remaining runtime for the j-th instance, and r_j be the true remaining runtime. The MAE loss for the remaining runtime is given by:
$$\mathcal{L}_{\text{runtime}} = \frac{1}{B} \sum_{j=1}^{B} |\hat{r}_j - r_j| \tag{3}$$

However, the way in which different techniques are trained to ultimately deliver these predictions, differs. Most recently, Wuyts et al. [15] proposed a Data-Aware (DA) encoder-decoder Transformer-based network, the only technique explicitly trained for jointly predicting all three targets, using the following composite loss function $\mathcal{L}_{\text{batch}} = \mathcal{L}_{\text{activity}} + \mathcal{L}_{\text{timestamp}} + \mathcal{L}_{\text{runtime}}$. Taymouri et al. [11] introduced a Non-Data-Aware (NDA) encoder-decoder LSTM network trained to generate activity and timestamp suffixes, with the remaining runtime implicitly derived by computing the sum over the predicted timestamp (proxy) suffix, i.e., $\hat{r}_k = \sum \hat{t}^p_{k+i}$. They furthermore complement supervised training with adversarial learning, resulting in performance gains for the largest beam widths. They used the following supervised loss function: $\mathcal{L}_{\text{batch}} = \mathcal{L}'_{\text{activity}} + \mathcal{L}'_{\text{timestamp}}$, with $\mathcal{L}'_{\text{activity}} =$

$\sum_{i=1}^{B}\sum_{t=1}^{N_i}\sum_{c=1}^{C} a_{i,t,c}\log(\hat{a}_{i,t,c})$, which is highly similar to Eq. 1, except for the omission of averaging, and with $\mathcal{L}'_{\text{timestamp}} = \sum_{i=1}^{B}\left((\sum \hat{t}^{p}_{k+i}) - \sum t^{p}_{k+i}\right)^2$. Ketykó et al. [5] compared various NDA suffix prediction models, including encoder-decoder LSTM and Transformer architectures. These seq2seq models were trained for activity and timestamp suffix prediction, with remaining runtime derived during inference ($\hat{r}_k = \sum \hat{t}^{p}_{k+i}$), using a weighted sum of categorical cross-entropy and Mean Squared Error (MSE) losses. The MSE loss is akin to the MAE loss (Eq. 2), but with $(\hat{t}_{i,t} - t_{i,t})^2$ replacing $|\hat{t}_{i,t} - t_{i,t}|$. - The DA LSTM-based technique by Gunnarsson et al. [3] is explicitly trained to predict the activity and remaining time suffix $\langle r_k, ..., r_{n-1}\rangle$, deviating from the common timestamp suffix approach. During inference, timestamp suffix predictions can be derived by subtracting consecutive timestamp predictions ($\hat{t}^{p}_{k+i} = \hat{r}_{k+i-1} - \hat{r}_{k+i}$), while only the first element of the predicted remaining time suffix ($\hat{r}_k$) is used for remaining time estimation. They used the following loss function: $\mathcal{L}_{\text{batch}} = \mathcal{L}_{\text{activity}} + \mathcal{L}'_{\text{runtime}}$. It should be noted that the latter component ($\mathcal{L}'_{\text{runtime}}$) is the MAE as defined in Eq. 2, except that it is computed over the predicted remaining time suffixes instead of the timestamp suffixes.

Moreover, earlier DL suffix prediction techniques, referred to as *Single-Event Prediction (SEP)* techniques, are explicitly trained for next event (instead of suffix) prediction, i.e. the joint prediction of the next event's activity and timestamp proxy (a_{k+1}, t^{p}_{k+1}). Only upon inference, they are leveraged for suffix generation by means of an iterative feedback loop, updating event prefixes after every consecutive prediction, with remaining time derived as in [5, 11]. Examples are the techniques proposed by [1, 9] (as discussed in Sect. 2.1), with [9] utilizing the unweighted sum of $\mathcal{L}^{1}_{\text{activity}}$ and $\mathcal{L}^{1}_{\text{timestamp}}$, which are similar to their suffix counterparts, Eq. 1 and 2 respectively, but with $N_j = 1$, and [1] adding a third component to the loss: the cross entropy loss $\mathcal{L}^{1}_{\text{role}}$ for predicting the next 'role' as well.

2.3 Case-Length Distortion

Without further domain knowledge or specific attention to a specific set of cases, one would prefer PPM techniques to generalize well over the underlying distribution of cases and their case lengths. However, the common practice of *instance creation* (Sect. 2.1) and, in the case of suffix prediction, the prevalent use of *non-normalized loss functions* in seq2seq DL architectures (Sect. 2.2, 4.1), introduces case-length distortions, hindering this ideal generalization and skewing evaluation metrics.

Instance Creation Creating instances by parsing each original case $\boldsymbol{\sigma}_i$ into n_i prefix-target pairs (Sect. 2.1) leads to an overrepresentation of longer cases among the derived instances in both the training $\boldsymbol{N}_{train}$ and test $\boldsymbol{N}_{test}$ sets, compared to their original distribution of case lengths. This artificially induced bias towards longer cases in $\boldsymbol{N}_{train}$ skews the models' learning process and negatively influences the degree to which the models generalize across the underlying case length distribution. Similarly, by default, evaluation metrics should measure models' true performance across all test log cases $\boldsymbol{\sigma}_i \in \boldsymbol{L}_{test}$, with each case weighted equally, regardless of its length and resulting instances in $\boldsymbol{N}_{test}$. However, evaluation metrics are typically computed by averaging over all instances in the test set $\boldsymbol{N}_{test}$, unintentionally amplifying the influence of longer cases (Sect. 3.3).

Figure 1 illustrates case-length distortion in the BPIC17[4] event log post-preprocessing. It compares the original case length distribution (GTCLD) with that after instance parsing (IBCLD). Note that the IBCLD represents the distribution of case lengths among the created instances, where each instance is labeled with the length of the original case it was derived from. The comparison shows a clear shift towards longer cases in the IBCLD, where they are overrepresented at the expense of cases pertaining to more representative lengths, as also reflected in the means and medians.

Additionally, event logs' case length distributions generally exhibit a pronounced right-skew, with outliers often containing significantly more events than the median. Further investigation revealed these extended cases to be frequently characterized by multiple repetitions of the same activity or group of activities,

[4] `https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b`

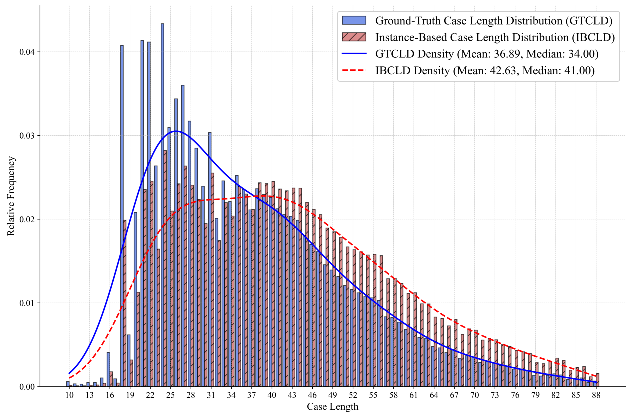

Fig. 1: Ground-Truth and Instance-Based Case Length Distributions for BPIC17 (post-preprocessing), highlighting the overrepresentation of longer cases in the instance-based distribution.

often executed in quick succession, which raises important considerations regarding their representativeness in modeling and whether models should be designed to accommodate or mitigate the influence of such disproportionately represented cases.

Non-Normalized Loss Functions In suffix prediction, instances originating from longer cases inherently possess longer ground-truth suffixes on average. Standard sequential loss functions, such as $\mathcal{L}_{\text{activity}}$ (Eq. 1) and $\mathcal{L}_{\text{timestamp}}$ (Eq. 2), average across all time steps within these suffixes for every instance in the batch, resulting in a greater number of terms in the loss calculation for instances from longer cases, disproportionately amplifying their influence on the overall loss. Consequently, these longer cases exert a stronger effect on the gradient updates during training, further skewing the models' learning process towards them.

3 Case Length Distribution-Reflective (*CaLenDiR*) PPM

In this section, we introduce the three core components of the *CaLenDiR* PPM framework designed to address case-length distortion: Uniform Case-Based Sampling (UCBS) (Sect. 3.1), Suffix-Length-Normalized Loss Functions for suffix prediction (Sect. 3.2), and Case-Based Evaluation Metrics (Sect. 3.3).

3.1 Uniform Case-Based Sampling (*UCBS*)

UCBS counteracts case-length distortion by equalizing the contribution of each case to the training set. At the start of each epoch, it samples a uniform number of instances from each case $\boldsymbol{\sigma}_i$ in training log $\boldsymbol{L}_{train}$, regardless of its original length. Let $\boldsymbol{N}_{\boldsymbol{\sigma}_i} = \left\{(\boldsymbol{\sigma}^p_{i,k}, \boldsymbol{y}_{i,k}) | 1 \leq k \leq n_i\right\}$ be the set of instances (i.e., prefix-target pairs) derived from case $\boldsymbol{\sigma}_i$. The training set $\boldsymbol{N}_{train} = \bigcup_{\boldsymbol{\sigma}_i \in \boldsymbol{L}_{train}} \boldsymbol{N}_{\boldsymbol{\sigma}_i}$ encompasses all instances derived from all cases $\boldsymbol{\sigma}_i \in \boldsymbol{L}_{train}$, with $\tilde{n}_{train} = \text{median}(\{n_i \,|\, \boldsymbol{\sigma}_i \in \boldsymbol{L}_{train}\})$ as the median case length. The *UCBS* procedure, detailed in Algorithm 1, begins by setting a random seed equal to the epoch number, ensuring slightly different sampled sets $\boldsymbol{N}^{(e)}_{train}$ each epoch e, which aids regularization and maintains reproducibility.

The procedure iterates over each case in $\boldsymbol{L}_{train}$, sampling $\tilde{n}_{train}$ instances from $\boldsymbol{N}_{\boldsymbol{\sigma}_i}$. If the number of instances for a case is less than $\tilde{n}_{train}$, sampling is done with replacement; otherwise, without replacement[5]. This design balances uniform instance contribution with diversity, reducing the risk of overfitting and improving training stability. The sampled instances are then aggregated to form the training set for epoch e: $\boldsymbol{N}^{(e)}_{train}$.

Algorithm 1 Uniform Case-Based Sampling (UCBS) Procedure

```
 1: Input: Training log $\boldsymbol{L}_{train}$, training set $\boldsymbol{N}_{train}$, median case length $\tilde{n}_{train}$, epoch number $e$
 2: Output: Sampled training set $\boldsymbol{N}^{(e)}_{train}$
 3: Set random seed $s = e$
 4: $\boldsymbol{N}^{(e)}_{train} \leftarrow \emptyset$
 5: for all $\boldsymbol{\sigma}_i \in \boldsymbol{L}_{train}$ do
 6:     $\boldsymbol{N}_{\boldsymbol{\sigma}_i} \leftarrow \{(\boldsymbol{\sigma}^p_{i,k}, \boldsymbol{y}_{i,k}) \in \boldsymbol{N}_{train} \mid 1 \leq k \leq n_i\}$
 7:     if $|\boldsymbol{N}_{\boldsymbol{\sigma}_i}| < \tilde{n}_{train}$ then
 8:         $\boldsymbol{N}^{(e)}_{\boldsymbol{\sigma}_i} \leftarrow$ sample $\tilde{n}_{train}$ instances from $\boldsymbol{N}_{\boldsymbol{\sigma}_i}$ with replacement
 9:     else
10:         $\boldsymbol{N}^{(e)}_{\boldsymbol{\sigma}_i} \leftarrow$ sample $\tilde{n}_{train}$ instances from $\boldsymbol{N}_{\boldsymbol{\sigma}_i}$ without replacement
11:     end if
12:     $\boldsymbol{N}^{(e)}_{train} \leftarrow \boldsymbol{N}^{(e)}_{train} \cup \boldsymbol{N}^{(e)}_{\boldsymbol{\sigma}_i}$
13: end for
14: return $\boldsymbol{N}^{(e)}_{train}$
```

3.2 Suffix-Length-Normalized Loss Functions

CaLenDiR's *Suffix-Length-Normalized Loss Functions* provide an alternative to the standard loss functions commonly used in seq2seq neural networks (e.g., [3,5,11,15]), which exacerbate case-length distortion. These functions normalize each instance's contribution by its suffix length N_j, rather than summing over all suffix elements in a batch, ensuring equal weighting of all instances and preventing overemphasis on longer cases. The normalized versions of Eq. 1 and 2 are provided in Eq. 4 and 5, respectively. This normalization can also be applied to other timestamp loss functions, such as Mean Squared Error (MSE) used in, e.g., [5]. For SEP techniques (e.g., [1,2,9]), suffix-length normalization is unnecessary, as they predict only the next event, with a single error per instance contributing to the loss.

$$\widetilde{\mathcal{L}}_{\text{activity}} = -\frac{1}{B}\sum_{j=1}^{B}\left(\frac{1}{N_j}\sum_{t=1}^{N_j}\sum_{c=1}^{C} a_{j,t,c}\log(\hat{a}_{j,t,c})\right) \tag{4}$$

$$\widetilde{\mathcal{L}}_{\text{timestamp}} = \frac{1}{B}\sum_{j=1}^{B}\left(\frac{1}{N_j}\sum_{t=1}^{N_j}|\hat{t}^p_{j,t} - t^p_{j,t}|\right) \tag{5}$$

[5] In the commonly applied instance creation approach, as described in Sect. 2.1, $|\boldsymbol{N}_{\boldsymbol{\sigma}_i}| = n_i$. We use the generic $|\boldsymbol{N}_{\boldsymbol{\sigma}_i}|$ to ensure UCBS is applicable to methods where this may not hold, such as those creating only $n_i - 1$ instances (i.e., $\boldsymbol{N}_{\boldsymbol{\sigma}_i} = \{(\boldsymbol{\sigma}^p_{i,k}, \boldsymbol{y}_{i,k}) \mid 2 \leq k \leq n_i\}$).

3.3 Case-Based Evaluation Metrics

The test log $\boldsymbol{L}_{test}$ is transformed into a test set $\boldsymbol{N}_{test} = \bigcup_{\boldsymbol{\sigma}_i \in \boldsymbol{L}_{test}} \boldsymbol{N}_{\boldsymbol{\sigma}_i}$ (Sect. 2.1, 3.1). In multi-task settings, the target $\boldsymbol{y}_{i,k}$ comprises multiple prediction tasks. Let $\boldsymbol{y}^x_{i,k} \in \boldsymbol{y}_{i,k}$ denote the ground truth for prediction task x and prefix $\boldsymbol{\sigma}^p_{i,k}$, and let $\widehat{\boldsymbol{y}}^x_{i,k}$ be the corresponding model prediction. The evaluation function $m_x : (\boldsymbol{y}^x_{i,k}, \widehat{\boldsymbol{y}}^x_{i,k}) \mapsto \mathbb{R}$ rates the prediction $\widehat{\boldsymbol{y}}^x_{i,k}$ against the ground truth $\boldsymbol{y}^x_{i,k}$. Finally, let $\alpha_{i,k}$ denote the pair $(\boldsymbol{\sigma}^p_{i,k}, \boldsymbol{y}_{i,k})$.The standard instance-based (IB) evaluation metric for prediction task x, averaging over all instances $\alpha_{i,k} \in \boldsymbol{N}_{test}$, is then defined as:

$$\overline{M}_{IB} = \frac{1}{|\boldsymbol{N}_{test}|} \sum_{i=1}^{|\boldsymbol{L}_{test}|} \sum_{\alpha_{i,k} \in \boldsymbol{N}_{\boldsymbol{\sigma}_i}} m_x(\boldsymbol{y}^x_{i,k}, \widehat{\boldsymbol{y}}^x_{i,k}) \tag{6}$$

In this approach, longer cases disproportionately influence the results due to their higher instance count. To mitigate this, we propose case-based (CB) evaluation metrics, where each test case $\boldsymbol{\sigma}_i$ is weighted equally, regardless of length. This is done by first averaging the score per case and then averaging across all cases in the test set, as shown in Eq. 7. This method provides a more balanced evaluation, accurately reflecting the model's performance across all cases in the test log $\boldsymbol{L}_{test}$.

$$\overline{M}_{CB} = \frac{1}{|\boldsymbol{L}_{test}|} \sum_{i=1}^{|\boldsymbol{L}_{test}|} \frac{1}{|\boldsymbol{N}_{\boldsymbol{\sigma}_i}|} \left(\sum_{\alpha_{i,k} \in \boldsymbol{N}_{\boldsymbol{\sigma}_i}} m_x(\boldsymbol{y}^x_{i,k}, \widehat{\boldsymbol{y}}^x_{i,k}) \right) \tag{7}$$

4 Experimental Setup

In our previous work [15], we introduced a Transformer-based seq2seq architecture for suffix prediction and re-implemented five existing architectures, applying consistent data preprocessing and scaling. All models were trained and evaluated on four real-life event logs using the standard approach. In this study, we replicate the experimental setup from [15] to evaluate *CaLenDiR* training for suffix prediction. The same models are retrained on the same event logs with identical hyperparameters, except for *(1)* the use of *UCBS* instead of processing the entire training set $\boldsymbol{N}_{train}$ each epoch (Sect. 3.1), and *(2)* suffix-length normalization of the loss functions (Sect. 3.2).

While [15] primarily reported conventional Instance-Based (IB) metrics, this paper expands the analysis by including Case-Based (CB) metrics (Sect. 3.3, 4.2) for both *CaLenDiR*-trained models and those trained using the standard approach in [15]. To provide a comprehensive perspective, we also report IB metrics for both training approaches. This dual reporting offers a more nuanced understanding of case-length distortion and the impact of *CaLenDiR* training on model performance and generalization. By directly comparing the evaluation metrics of *CaLenDiR*-trained models with those trained using the standard approach, we isolate the impact of the training methodology, clearly measuring the improvements facilitated by *CaLenDiR*.

4.1 Data & Models

The four real-life event logs are: BPIC17[6], BPIC17-DR, BPIC19[7], and BAC. Table 1 summarizes their main properties, including the average and standard deviation of the case length (event count) and case duration. BPIC17-DR is a variant of BPIC17 with subsequent repetitions of the same activity removed to enhance

[6] `https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b`

[7] `https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1`

Table 1: Overview of the models (left) and event logs (right) included in the experimental comparison [15].

Benchmark	seq2eq	DA	Explicit RT prediction	Explicit timestamp prediction	Default loss	*CaLenDiR* loss
SEP-LSTM	×	×	×	✓	$\mathcal{L}^1_{\text{activity}}+\mathcal{L}^1_{\text{timestamp}}$	$\mathcal{L}^1_{\text{activity}}+\mathcal{L}^1_{\text{timestamp}}$
CRTP-LSTM *(NDA)*	✓	×	✓	×	$\mathcal{L}_{\text{activity}}+\mathcal{L}'_{\text{runtime}}$	$\widetilde{\mathcal{L}}_{\text{activity}}+\widetilde{\mathcal{L}}'_{\text{runtime}}$
CRTP-LSTM	✓	✓	✓	×	$\mathcal{L}_{\text{activity}}+\mathcal{L}'_{\text{runtime}}$	$\widetilde{\mathcal{L}}_{\text{activity}}+\widetilde{\mathcal{L}}'_{\text{runtime}}$
ED-LSTM	✓	×	×	✓	$\mathcal{L}_{\text{activity}}+\mathcal{L}_{\text{timestamp}}$	$\widetilde{\mathcal{L}}_{\text{activity}}+\widetilde{\mathcal{L}}_{\text{timestamp}}$
SuTraN *(NDA)*	✓	×	✓	✓	$\mathcal{L}_{\text{activity}}+\mathcal{L}_{\text{timestamp}}+\mathcal{L}_{\text{runtime}}$	$\widetilde{\mathcal{L}}_{\text{activity}}+\widetilde{\mathcal{L}}_{\text{timestamp}}+\mathcal{L}_{\text{runtime}}$
SuTraN	✓	✓	✓	✓	$\mathcal{L}_{\text{activity}}+\mathcal{L}_{\text{timestamp}}+\mathcal{L}_{\text{runtime}}$	$\widetilde{\mathcal{L}}_{\text{activity}}+\widetilde{\mathcal{L}}_{\text{timestamp}}+\mathcal{L}_{\text{runtime}}$

Log	Cases - Events	Variants	avg. - SD Length	avg. - SD Duration
BPIC17	30,078 – 1,109,665	14,745	36.90 – 14.55	20.52 – 10.81 (days)
BPIC17-DR	30,078 – 704,202	3,592	23.42 – 6.85	20.52 – 10.81 (days)
BPIC19	181,395 – 986,077	5,767	5.44 – 1.78	71.76 – 36.78 (days)
BAC	362,563 – 1,767,186	13,496	4.87 – 2.49	732 – 912.24 (sec.)

data quality. The BAC log, sourced from a major European airport's luggage handling system, is not publicly available. Adopting a chronological 75-25 % out-of-time train-test split, each event log $\boldsymbol{L}$ is subdivided into a training and test log ($\boldsymbol{L}_{train}$ & $\boldsymbol{L}_{test}$). The former was further divided into final training and validation logs ($\boldsymbol{L}_{train}$ & $\boldsymbol{L}_{val}$) by assigning the last 20 % of cases to $\boldsymbol{L}_{val}$. The final train, validation and test sets ($\boldsymbol{N}_{train}$, $\boldsymbol{N}_{val}$ & $\boldsymbol{N}_{test}$) are derived from $\boldsymbol{L}_{train}$, $\boldsymbol{L}_{val}$ and $\boldsymbol{L}_{test}$ as discussed in Sect. 2.1. Please refer to [15] for further details regarding the preprocessing pipeline. Table 1 summarizes the six implementations, re-trained using the *CaLenDiR* framework. The *'DA'* column indicates whether the architecture is *Data-Aware*, leveraging (all) available payload data beyond just the timestamp and activity information of the prefix events. *SEP-LSTM* is a re-implementation of [9], trained solely for next event prediction, only generating suffixes during inference using an external feedback loop (Sect. 2). *CRTP-LSTM* re-implements the DA architecture from [3], while *ED-LSTM*, an encoder-decoder LSTM, is based on NDA techniques from [5, 11]. *SuTraN* is the encoder-decoder Transformer network from [15]. *SuTraN* and *CRTP-LSTM* are the only DA models, with NDA versions (*SuTraN (NDA)* and *CRTP-LSTM (NDA)*) also implemented.

During inference, all implementations generate an activity suffix, timestamp suffix, and remaining runtime prediction, though the specific targets trained for, differ (Sect. 2.2). Each implementation uses a simple additive loss function, summing the losses for its explicit prediction targets. The individual loss components (Table 1) are standardized across implementations to ensure a level playing field. For *CaLenDiR* training, seq2seq models use Suffix-Length-Normalized Cross Entropy (Eq. 4) for activity suffix prediction and Normalized MAE (Eq. 5) for timestamp suffix prediction. In *CRTP-LSTM*, the MAE is applied to remaining time suffixes instead of timestamp suffixes. *SuTraN*, which predicts all three targets, adds a third loss function (Eq. 3) for remaining time prediction, but no normalization is needed since it outputs a scalar instead of a sequence. *SEP-LSTM*, trained only for next event prediction, uses non-sequential cross-entropy $\mathcal{L}^1_{\text{activity}}$ and MAE $\mathcal{L}^1_{\text{timestamp}}$ for next activity and timestamp prediction (Sect. 2.2), respectively, which do not require normalization. Aside from these training specifics, all other hyperparameters and inference settings, were kept consistent with those in [15].

4.2 Evaluation - Uniform Case-Based vs. Instance-Based

We evaluate the three prediction tasks using (normalized) Damerau-Levenshtein Similarity (DLS) for activity suffix prediction and mean absolute error (MAE) for both timestamp suffix and remaining time prediction. DLS measures sequence similarity on a scale from 0 (completely different) to 1 (identical) and is calculated as $DLS(X,Y)=1-\frac{DL(X,Y)}{\max(|X|,|Y|)}$, where $DL(X,Y)$ is the Damerau-Levenshtein distance between sequences X and Y, and $\max(|X|,|Y|)$ the length of the longer sequence. For activity suffix prediction, DLS is computed for each instance $\alpha_{i,k}=(\boldsymbol{\sigma}^p_{i,k},\boldsymbol{y}_{i,k})$ in $\boldsymbol{N}_{test}$ as $DLS(\boldsymbol{y}^{activity}_{i,k},\widehat{\boldsymbol{y}}^{activity}_{i,k})$, where $\boldsymbol{y}^{activity}_{i,k}$ is the ground-truth activity suffix and $\widehat{\boldsymbol{y}}^{activity}_{i,k}$ the predicted suffix. The IB and CB metrics ($\overline{DLS}_{IB}$ & $\overline{DLS}_{CB}$) are derived by setting $m_x(\boldsymbol{y}^x_{i,k},\widehat{\boldsymbol{y}}^x_{i,k})$ to $DLS(\boldsymbol{y}^{activity}_{i,k},\widehat{\boldsymbol{y}}^{activity}_{i,k})$ in Equations 6 and 7 respectively. The IB and CB metrics for remaining time ($\overline{MAE}^{runtime}_{IB}$ & $\overline{MAE}^{runtime}_{CB}$) are calculated by setting $m_x(\boldsymbol{y}^x_{i,k},\widehat{\boldsymbol{y}}^x_{i,k})$ to $|\widehat{r}_{i,k}-r_{i,k}|$. In contrast, the MAE for timestamp suffix prediction is computed across the $N_{i,k}$ predicted timestamps in the suffix. Let $\widehat{t}^{p}_{i,k;t}$ and $t^{p}_{i,k;t}$ represent the predicted and true timestamps, respectively. The IB MAE is given by:

$$\overline{MAE}^{timestamp}_{IB} = \frac{1}{|\boldsymbol{N}_{test}|}\sum_{i=1}^{|\boldsymbol{L}_{test}|}\sum_{\alpha_{i,k}\in\boldsymbol{N}_{\boldsymbol{\sigma}_i}}\sum_{t=1}^{N_{i,k}}|\widehat{t}^{p}_{i,k;t}-t^{p}_{i,k;t}|$$

To counter the resulting additional distortions (cf. Sect. 2.3 and 3.2), the CB MAE includes both case-based weighting and suffix-length normalization:

$$\overline{MAE}^{timestamp}_{CB} = \frac{1}{|\boldsymbol{L}_{test}|}\sum_{i=1}^{|\boldsymbol{L}_{test}|}\frac{1}{|\boldsymbol{N}_{\boldsymbol{\sigma}_i}|}\left(\sum_{\alpha_{i,k}\in\boldsymbol{N}_{\boldsymbol{\sigma}_i}}\frac{1}{N_{i,k}}\sum_{t=1}^{N_{i,k}}|\widehat{t}^{p}_{i,k;t}-t^{p}_{i,k;t}|\right)$$

5 Results

Tables 2a and 2b present the IB and CB performance metrics for models trained using the standard approach [15] and the *CaLenDiR* approach. The best results are in bold and underlined, while the second-best are only in bold. Table 2c shows the percentage changes in these metrics for *CaLenDiR* training compared to standard training, calculated as $\frac{x_{calendir}-x_{standard}}{x_{standard}}$. Our analysis mainly focuses on the undistorted CB metrics unless otherwise noted.

Table 2: Performance comparison across different techniques and datasets

(a) Standard Training

Model	Damerau-Levenshtein Similarity (DLS)								MAE Timestamp Suffix ($\text{MAE}_{\text{suffix}}$)								MAE Remaining Runtime ($\text{MAE}_{\text{runtime}}$)							
	BPIC17-DR		BPIC17		BPIC19		BAC		BPIC17-DR		BPIC17		BPIC19		BAC (sec.)		BPIC17-DR		BPIC17		BPIC19		BAC (sec.)	
	IB	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**
SEP-LSTM	0.6733	0.6902	0.2160	0.2823	0.8425	0.8183	0.7206	0.7488	**1178**	**1219**	762	1048	16604	13930	113	64	10139	9699	11823	11683	30572	33645	420	262
CRTP-LSTM (NDA)	0.6525	0.6734	0.3357	0.3692	0.8435	0.8188	0.7320	0.7640	1391	1313	1009	1248	17182	14511	113	66	8931	8520	8906	8788	29323	33400	318	204
CRTP-LSTM	**0.6741**	**0.7149**	**0.4095**	**0.4660**	**0.8522**	**0.8427**	**0.8374**	**0.8647**	1556	1414	996	1184	**15708**	**13572**	112	61	**8000**	**7287**	**8685**	**8059**	**21345**	**24360**	**301**	**191**
ED-LSTM	0.6737	0.6902	0.3239	0.3623	0.8477	0.8201	0.7424	0.7663	1200	1224	**739**	**993**	16485	13810	**108**	61	9705	9298	12160	11889	31000	33914	338	217
SuTraN (NDA)	0.6723	0.6895	0.2669	0.3161	0.8435	0.8193	0.7355	0.7645	1201	1224	**745**	1008	16621	13897	109	**61**	8896	8452	8860	8754	29209	33387	308	200
SuTraN	**0.7274**	**0.7621**	**0.3843**	**0.4621**	**0.8699**	**0.8601**	**0.8461**	**0.8698**	**1157**	**977**	749	**882**	**14542**	**12446**	**106**	**58**	**7727**	**6945**	**7913**	**7260**	**20182**	**23462**	**290**	**183**

(b) CaLenDiR Training

Model	Damerau-Levenshtein Similarity (DLS)								MAE Timestamp Suffix ($\text{MAE}_{\text{suffix}}$)								MAE Remaining Runtime ($\text{MAE}_{\text{runtime}}$)							
	BPIC17-DR		BPIC17		BPIC19		BAC		BPIC17-DR		BPIC17		BPIC19		BAC (sec.)		BPIC17-DR		BPIC17		BPIC19		BAC (sec.)	
	IB	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**
SEP-LSTM	0.6738	0.6931	0.2132	0.2818	0.8425	0.8217	0.7421	0.7764	**1190**	1218	**759**	1001	16909	14139	112	60	9821	9405	11940	11590	30134	33324	382	234
CRTP-LSTM (NDA)	0.6268	0.7014	**0.4285**	**0.5207**	0.7851	0.7784	0.7077	0.7527	1655	1298	1160	1136	18005	15046	122	72	9878	8775	10252	9157	33173	33730	331	201
CRTP-LSTM	**0.6868**	**0.7551**	**0.4700**	**0.5523**	**0.8498**	**0.8451**	**0.8338**	**0.8665**	1525	1318	1088	1209	**16207**	14105	**109**	**54**	**8035**	**7072**	**8503**	**7513**	**22509**	**24938**	**301**	**177**
ED-LSTM	0.6485	0.7141	0.3862	0.4809	0.7931	0.7883	0.7127	0.7564	1265	1181	894	948	16748	**14061**	115	62	9231	8795	10817	10078	36490	36624	385	252
SuTraN (NDA)	0.6521	0.7137	0.3704	0.4669	0.7907	0.7870	0.7097	0.7535	1288	**1153**	909	**930**	17260	14455	115	62	8943	8422	9015	8657	29716	32734	313	195
SuTraN	**0.7278**	**0.7848**	0.3795	0.4780	**0.8656**	**0.8612**	**0.8386**	**0.8708**	**1187**	**943**	**765**	**833**	**14620**	**12341**	**107**	**52**	**7770**	**6939**	**7859**	**7092**	**20076**	**23146**	**285**	**169**

(c) Percentage Change Comparison

Model	Damerau-Levenshtein Similarity (DLS)								MAE Timestamp Suffix ($\text{MAE}_{\text{suffix}}$)								MAE Remaining Runtime ($\text{MAE}_{\text{runtime}}$)							
	BPIC17-DR		BPIC17		BPIC19		BAC		BPIC17-DR		BPIC17		BPIC19		BAC (sec.)		BPIC17-DR		BPIC17		BPIC19		BAC (sec.)	
	IB	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**	**IB**	**CB**
SEP-LSTM	+0.07	+0.42	-1.30	-0.18	+0.00	+0.42	+2.98	+3.69	+1.02	-0.08	-0.39	-4.48	+1.84	+1.50	-0.88	-6.25	-3.14	-3.03	+0.99	-0.80	-1.43	-0.95	-9.05	-10.69
CRTP-LSTM (NDA)	-3.94	+4.16	+27.64	+41.03	-6.92	-4.93	-3.32	-1.48	+18.98	-1.14	+14.97	-8.97	+4.79	+3.69	+7.96	+9.09	+10.60	+2.99	+15.11	+4.20	+13.13	+0.99	+4.09	-1.47
CRTP-LSTM	+1.88	+5.62	+14.77	+18.52	-0.28	+0.28	-0.43	+0.21	-1.99	-6.79	+9.24	+2.11	+3.18	+3.93	-2.68	-11.48	+0.44	-2.95	-2.10	-6.78	+5.45	+2.37	+0.00	-7.3
ED-LSTM	-3.74	+3.46	+19.23	+32.74	-6.44	-3.88	-4.00	-1.29	+5.42	-3.51	+20.97	-4.53	+1.60	+1.82	+6.48	+1.64	-4.88	-5.41	-11.04	-15.23	+17.71	+7.99	+13.91	+16.13
SuTraN (NDA)	-3.00	+3.51	+38.78	+47.71	-6.26	-3.94	-3.51	-1.44	+7.24	-5.80	+22.01	-7.74	+3.84	+4.02	+5.50	+1.64	+0.53	-0.35	+1.75	-1.11	+1.74	-1.96	+1.62	-2.50
SuTraN	+0.05	+2.98	-1.25	+3.44	-0.49	+0.13	-0.89	+0.11	+2.59	-3.48	+2.14	-5.56	+0.54	-0.84	+0.94	-10.34	+0.56	-0.09	-0.68	-2.31	-0.53	-1.35	-1.72	-7.65
Average Change	**-1.45**	**+3.36**	**+16.31**	**+23.88**	**-3.49**	**-1.99**	**-1.53**	**-0.03**	**+5.54**	**-3.47**	**+11.49**	**-4.86**	**+2.63**	**+2.35**	**+2.89**	**-2.62**	**+0.68**	**-1.47**	**+0.67**	**-3.67**	**+6.01**	**+1.18**	**+1.47**	**-2.25**

As expected, *CaLenDiR*-training generally improves CB metrics across all three prediction tasks (see *Average Change* row in Table 2c. However, exceptions include BPIC19, where CB DLS, $\text{MAE}_{\text{suffix}}$, and $\text{MAE}_{\text{runtime}}$slightly decline by approximately 1.99%, 2.35%, and 1.18%, respectively, and a negligible decrease of 0.03% in CB DLS on the BAC log. Overall, these results indicate that *CaLenDiR* training enhances the generalization and robustness of suffix prediction models. Conversely, *CaLenDiR* training is anticipated to reduce IB performance, which is supported by the average IB percentage changes. A notable exception is the IB DLS on BPIC17, where *CaLenDiR* training leads to a significant 16.31% improvement (see infra).

Eliminating case-length distortion also has important implications for the insights derived from evaluation and benchmarking setups. This is evident in the shifts in CB rankings among *CaLenDiR*-trained models (Table 2b) versus the standard approach (Table 2a). Rankings shift across all logs for activity suffix (DLS) and timestamp suffix prediction ($\mathrm{MAE}_{\mathrm{suffix}}$), and on three of four logs for remaining time prediction ($\mathrm{MAE}_{\mathrm{runtime}}$), except for BPIC17-DR, where models retain their "standard' rankings. These shifts indicate that susceptibility to case-length distortion is model-dependent. The changes in IB rankings for all logs for DLS and $\mathrm{MAE}_{\mathrm{runtime}}$, and for three of four logs for $\mathrm{MAE}_{\mathrm{suffix}}$, further support this hypothesis.

The more pronounced the right-skewness in a log's case length distribution, the greater the improvements from *CaLenDiR* training across all prediction tasks (Table 2c). This skewness is most pronounced in BPIC17 (Fig. 1, Table 1) and least in BPIC19 (Table 1).

In BPIC17, cases frequently include immediate repetitions of identical activities, a pattern that becomes more prevalent with longer cases. This repetitiveness, previously examined in [15], likely introduces noise into the data. The BPIC17 results support the hypothesis that lengthy, repetitive cases contribute more noise than valuable information to the predictive models. *CaLenDiR* training leads to the largest improvements across all event logs: DLS increases by 23.88%, $\mathrm{MAE}_{\mathrm{suffix}}$decreases by 4.86%, and $\mathrm{MAE}_{\mathrm{runtime}}$reduces by 3.67%. Notably, the IB DLS metric, still influenced by the overrepresentation of these lengthy cases, improves significantly by 16.31%, contrary to trends in other logs. The expected performance drop on longer cases did not materialize, suggesting that improvements on more moderate, representative cases far outweigh any negative impact on IB DLS. This further supports the hypothesis that extended BPIC17 cases distort predictive signals relevant to representative cases while lacking meaningful, learnable patterns themselves. The repetitive events in lengthy BPIC17 cases are also primarily executed in quick succession, leading to numerous near-zero values in their ground-truth timestamp suffixes, which disrupts the predictive signal. *CaLenDiR* training, by reducing the overrepresentation of these cases, results in an 11.49% decrease in IB $\mathrm{MAE}_{\mathrm{suffix}}$performance but a 4.86% improvement in CB $\mathrm{MAE}_{\mathrm{suffix}}$, the largest changes across all logs. This suggests that *CaLenDiR*-trained models are less prone to overfitting on low elapsed time predictions, enhancing robustness and improving the overall predictive signal. As a result, these models better predict timestamps for the more representative BPIC17 cases, reflected in the significant CB improvements and deterioration of the IB metrics due to the continued bias from near-zero targets.

BPIC19 has the least right-skewed distribution of case lengths among all logs, suggesting a lower presence of case-length distortion. Despite dedicated preprocessing efforts to address observed drift in BPIC19 (cf. [15, 16]), minor distributional changes persist, such as an increase in the frequency of slightly longer cases in the (out-of-time) test set. Additionally, BPIC19 uniquely contains a significant number of (training) cases with only two events. *CaLenDiR* training in turn reduces the disproportionate impact of these slightly longer cases while balancing the importance of these short outliers, likely explaining the generally divergent BPIC19 results. This raises questions about whether these short outliers should be further addressed as well.

However, as with all logs, the impact of *CaLenDiR* training on BPIC19 remains both model- and prediction task-dependent. Interestingly, data-aware (DA) models, i.e. CRTP-LSTM and SuTraN, are not adversely affected and even - although negligible - show an increase in CB DLS performance. This might suggest that the longer cases, more frequent in the test set, occupy a distinct region in the feature hyperspace, making these DA models more resilient to activity suffix prediction under moderate drift conditions. For timestamp suffix and runtime prediction however, only *SuTraN* remains resilient.

In addition to the DA models, SEP-LSTM, the only model trained for next-event rather than suffix prediction, also shows mild DLS and $\mathrm{MAE}_{\mathrm{runtime}}$ improvements with *CaLenDiR* on BPIC19. This suggests that *CaLenDiR* aids SEP-LSTM by minimizing case length distortion, allowing it to focus on the prevalent short-case patterns typical of BPIC19. Similar trends are observed in BAC, where SEP-LSTM shows the largest improvements in DLS (+3.69%) and $\text{MAE}_{\text{runtime}}$ (−10.69%), moving from the worst to the third-best performing model. These results underscore SEP-LSTM's effectiveness in shorter logs, where simpler, consistent patterns prevail. Conversely, SEP-LSTM struggles with longer sequences due to training-inference discrepancies, as seen in BPIC17-DR, where it reports the mildest DLS and $\text{MAE}_{\text{suffix}}$ improvements (+0.42% and −0.08%), and in BPIC17, where it uniquely shows a slight DLS decrease (−0.18%), contrasting with the major improvements reported for other models. This highlights SEP-LSTM's sensitivity to case length and its reliance on environments where the noise from atypical long cases is reduced.

Beyond SEP-LSTM, also both DA models consistently improved on the BAC log, unlike their non-data-aware (NDA) counterparts, which saw slight decreases in DLS and $\text{MAE}_{\text{suffix}}$and smaller gains in $\text{MAE}_{\text{runtime}}$. This suggests that irregular longer cases and associated case-length distortion significantly disrupt the predictive signal from additional payload data in this log.

6 Conclusion

This study introduced the *CaLenDiR* PPM framework, which addresses case-length distortion in DL-based PPM techniques to improve model generalization, performance, and robustness. At the core of *CaLenDiR* is the *UCBS* technique, eliminating distortion originating from the prevalent approach to *instance creation* during training. For suffix prediction, *CaLenDiR* incorporates *Suffix-Length-Normalized Loss Functions* to tackle additional distortions specific to the training of suffix prediction models. Furthermore, to prevent these same sources of distortion from skewing evaluation, *CaLenDiR* includes generic *Case-Based (CB) metrics* as the more representative alternative to the widely used Instance-Based (IB) metrics.

We evaluated the effectiveness of the *CaLenDiR* methodology through experiments focused on suffix prediction. Using multiple real-life event logs and six different DL-based suffix prediction techniques, each model was trained twice per log: once using prevalent training methods and once using *CaLenDiR*-training. This approach allowed for a direct performance comparison, isolating the impact of *CaLenDiR*-training. The results highlight the importance of addressing this distortion in PPM and demonstrate the effectiveness of the proposed *CaLenDiR* framework in improving model performance and robustness. The changes in model rankings across various metrics underscore significant implications for performance benchmarking, revealing that susceptibility to case-length distortion is model-dependent and that addressing it alters competitive standings. Our deeper analysis of log characteristics revealed that the more right-skewed the case length distribution, the more pronounced the case-length distortion, and hence the greater the effect of *CaLenDiR*. By investigating log peculiarities following distinctive results, we uncovered further insights and raised important questions regarding the critical components required for effective suffix prediction, and the intricate interplay between these components, event log characteristics and model performance.

Future work could explore the effectiveness of the *CaLenDiR* framework across different prediction tasks beyond suffix prediction.

To promote transparency and collaboration within the research community, we have made the code for this study and the implementation of all *CaLenDiR* components available to the public at `https://github.com/BrechtWts/CaLenDiR-PPM`. The repository includes detailed documentation and supplementary materials to facilitate reproducibility, support the integration of the *CaLenDiR* framework into other research projects, and contribute to the advancement of the PPM field.

References

1. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate lstm models of business processes. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) Business Process Management. pp. 286–302. Springer International Publishing, Cham (2019)

2. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decision Support Systems **100**, 129–140 (2017). `https://doi.org/https://doi.org/10.1016/j.dss.2017.04.003`, smart Business Process Management
3. Gunnarsson, B.R., Broucke, S.v., De Weerdt, J.: A direct data aware lstm neural network architecture for complete remaining trace and runtime prediction. IEEE Transactions on Services Computing **16**(4), 2330–2342 (2023). `https://doi.org/10.1109/TSC.2023.3245726`
4. Hinkka, M., Lehto, T., Heljanko, K., Jung, A.: Classifying process instances using recurrent neural networks. In: Daniel, F., Sheng, Q.Z., Motahari, H. (eds.) Business Process Management Workshops. pp. 313–324. Springer International Publishing, Cham (2019)
5. Ketykó, I., Mannhardt, F., Hassani, M., van Dongen, B.F.: What averages do not tell: predicting real life processes with sequential deep learning. In: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. p. 1128–1131. SAC '22, Association for Computing Machinery, New York, NY, USA (2022), `https://doi.org/10.1145/3477314.3507179`
6. Kratsch, W., Manderscheid, J., Röglinger, M., Seyfried, J.: Machine learning in business process monitoring: A comparison of deep learning and classical approaches used for outcome prediction. Business & Information Systems Engineering **63**(3), 261–276 (Jun 2021), `https://doi.org/10.1007/s12599-020-00645-0`
7. Navarin, N., Vincenzi, B., Polato, M., Sperduti, A.: Lstm networks for data-aware remaining time prediction of business process instances. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–7 (2017). `https://doi.org/10.1109/SSCI.2017.8285184`
8. Philipp, P., Jacob, R., Robert, S., Beyerer, J.: Predictive analysis of business processes using neural networks with attention mechanism. In: 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). pp. 225–230 (2020). `https://doi.org/10.1109/ICAIIC48513.2020.9065057`
9. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Proceedings of the 29th International Conference on Advanced Information Systems Engineering. pp. 477–492. Springer (2017)
10. Taymouri, F., Rosa, M.L., Erfani, S., Bozorgi, Z.D., Verenich, I.: Predictive business process monitoring via generative adversarial nets: The case of next event prediction. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) Business Process Management. pp. 237–256. Springer International Publishing, Cham (2020)
11. Taymouri, F., Rosa, M.L., Erfani, S.M.: A deep adversarial model for suffix and remaining time prediction of event sequences. In: Demeniconi, C., Davidson, I. (eds.) Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021. pp. 522–530. SIAM (2021), `https://doi.org/10.1137/1.9781611976700.59`
12. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Transactions on Knowledge Discovery from Data (TKDD) **13**(2), 1–57 (2019)
13. Verenich, I., Dumas, M., Rosa, M.L., Maggi, F.M., Teinemaa, I.: Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. ACM Trans. Intell. Syst. Technol. **10**(4) (jul 2019), `https://doi.org/10.1145/3331449`
14. Weytjens, H., De Weerdt, J.: Process outcome prediction: Cnn vs. lstm (with attention). In: Del Río Ortega, A., Leopold, H., Santoro, F.M. (eds.) Business Process Management Workshops. pp. 321–333. Springer International Publishing, Cham (2020)
15. Wuyts, B., vanden Broucke, S., De Weerdt, J.: SuTraN: an encoder-decoder transformer for full-context-aware suffix prediction of business processes. In: Lu, X., Pufahl, L., Song, M. (eds.) 2024 6th International Conference on Process Mining (ICPM) (2024)
16. Wuyts, B., Weytjens, H., vanden Broucke, S., De Weerdt, J.: DyLoPro: Profiling the dynamics of event logs. In: Di Francescomarino, C., Burattin, A., Janiesch, C., Sadiq, S. (eds.) Business Process Management. pp. 146–162. Springer Nature Switzerland, Cham (2023)