Tackling Multi-Class Imbalance in Next Activity Prediction with Class-Balanced Focal Loss

Xiaomeng He[1], Rafael Oyamada[1], Johannes De Smedt[1], Seppe vanden Broucke[1,2], and Jochen De Weerdt[1]

[1] Research Center for Information Systems Engineering (LIRIS), KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium
{xiaomeng.he, rafael.oyamada, johannes.desmedt, jochen.deweerdt}@kuleuven.be
[2] Department of Business Informatics and Operations Management, Ghent University, Tweekerkenstraat 2, 9000 Ghent, Belgium
seppe.vandenbroucke@ugent.be

Abstract. In Predictive Process Monitoring, machine learning models are increasingly being adopted for the next activity prediction task, which involves predicting the activity label of the subsequent event in a case given an ongoing process execution. In real-life event logs, there are typically many activity labels (classes), and a skewed distribution over these labels is often observed, with some activities occurring more frequently than others. Under such imbalance, predictive models trained with the standard cross-entropy loss tend to favor frequent classes, leading to poor performance over infrequent activities. Nevertheless, this model bias caused by multi-class imbalance has received limited attention in the current PPM literature and remains largely unsolved in the state-of-the-art. In this work, we explore the use of Class-Balanced Focal Loss (CBFL) as a solution to this challenge. CBFL is a loss re-weighting method that simultaneously up-weights minority classes and down-weights well-classified instances. To evaluate its effectiveness, we train a range of machine learning models with CBFL on four event logs. Our empirical results show that, in most cases, CBFL substantially improves performance on minority classes without degrading performance for majority classes, leading to gains in overall predictive performance.

Keywords: Predictive Process Monitoring · Next Activity Prediction · Multi-Class Imbalance · Class-Balanced Focal Loss

1 Introduction

As a subfield of Process Mining, Predictive Process Monitoring (PPM) focuses on building predictive models to estimate the future behavior of ongoing cases [6]. Among the various predictive tasks, next activity prediction involves predicting the activity label of the subsequent event given an incomplete trace [17]. Such predictions can, for example, inform resource planning or trigger timely interventions in response to expected actions.

Recently, a variety of machine learning (ML) and deep learning (DL) models have been adopted for next activity prediction [14]. These models make predictions by classifying each input sequence into one out of multiple classes, with each class corresponding to a possible next activity (label). In many real-world processes, however, some activities (e.g., routine tasks) tend to occur more frequently than others, which causes their corresponding classes to dominate the data and creates multi-class imbalance.

Under this imbalance issue, standard ML models tend to be biased toward frequent classes while underperforming on infrequent ones [9]. Consequently, existing methods often struggle to deliver reliable predictions for rare but potentially critical process behavior [13], such as unusual cancellations or rejections. Nevertheless, few studies in the PPM domain have specifically addressed this issue, and the only existing solution [11] is not well-suited to multi-class settings. Appropriate approaches to address multi-class imbalance in next activity prediction are still lacking in the PPM literature.

Beyond PPM, handling multi-class imbalance is an active research line in the broader ML field, where loss re-weighting has emerged as one of the most effective strategies [9]. It works by re-scaling the loss contributions of specific classes or instances, thereby guiding the model to focus more on underrepresented data.

Building on the principle of loss re-weighting, this study explores the use of Class-Balanced Focal Loss (CBFL) [7,10] as a solution to tackle multi-class imbalance in next activity prediction. CBFL combines class-level weighting with dynamic instance-level weighting, allowing it to simultaneously up-weight minority classes and down-weight well-classified instances, thereby improving the model's focus on difficult and under-represented samples. We validate our methodology by empirically benchmarking CBFL against the conventional cross-entropy loss (CEL). For this purpose, we train a range of models on four real-life event logs using both loss functions. Our experiments include XGBoost, Long Short-Term Memory networks (LSTMs), Transformers, and the recent Extended LSTMs (xLSTMs).

The remainder of this paper is structured as follows. Section 2 motivates this work by identifying gaps in the existing literature and illustrating the challenges of multi-class imbalance. Section 3 covers the necessary preliminaries, and Section 4 presents the CBFL method, demonstrating why it is well-suited for the next activity prediction task. Section 5 describes the experimental setup for the empirical evaluation and reports on the results. Section 6 concludes the paper, discusses limitations, and suggests directions for future work.

2 Current State and Limitations

This section reviews the related works, introduces the problem of multi-class imbalance, and highlights the challenges it presents, thereby motivating this study.

2.1 Related Work

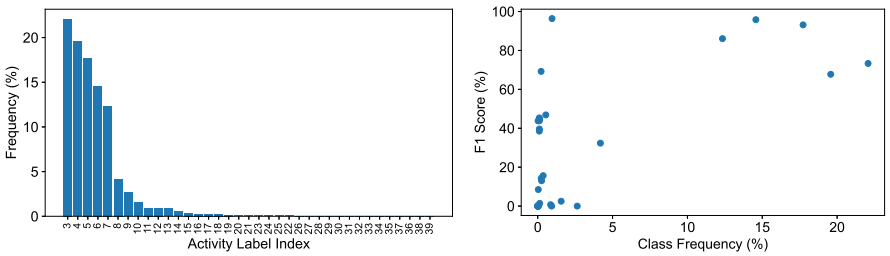
Next activity prediction is a critical research topic in PPM. Early works primarily rely on process-aware methods such as probabilistic finite automata [3], whereas recent research increasingly adopts DL-based approaches [14]. Although various DL architectures, such as Convolutional Neural Networks (CNNs) [13] and Graph Neural Networks (GNNs) [15], have been investigated, the inherently sequential nature of traces makes sequence models particularly prevalent [17]. Among these, LSTMs are the most commonly used architectures [5,8,18], while Transformer-based models have also emerged as promising alternatives [4], following their success in language modeling.

Although significant progress has been made in improving predictive accuracy through architecture design and data preprocessing, the issue of class imbalance remains largely under-investigated—despite its substantial impact on model performance. The

poor performance of predictive models on infrequent activity labels has already been noticed [13] and pointed out as a potential research direction in the literature [12]. Nevertheless, many well-known studies [5,8,14,18] ignore this issue and rely on accuracy metrics that primarily reflect performance on frequent classes; as a result, these metrics may provide an inflated view of model quality, even when rare classes are poorly predicted. To our knowledge, only Mehdiyev et al. [11] address this challenge to some extent by framing it as a binary classification problem: they isolate one rare activity as the minority class, group all others as the majority, and apply oversampling. However, this setup produces binary outputs that are not suitable for next activity prediction, and disregards activities that might impact business outcomes and key performance indicators.

2.2 Problem Illustration

Multi-class imbalance is common in real-world event logs. As depicted in Figure 1a, this imbalance is characterized by a skewed distribution of the target variable's classes (i.e., activity labels in the context of next activity prediction), where a few dominant classes (majority classes) contribute most instances, while many others (minority classes) are underrepresented by relatively few instances. In practice, multi-class imbalance presents challenges for predictive modeling, as standard ML models tend to be biased towards the majority classes. To illustrate this issue, Figure 1b plots per-class F1-scores against class frequency, based on experimental results from an LSTM model trained with CEL to predict the next activity. The model's performance on minority classes tends to be worse and exhibit greater variability compared to majority classes. This outcome is closely linked to CEL's equal weighting of all training instances. As the majority classes contribute more instances, their misclassification losses dominate the overall training loss, skewing the gradient updates in their favor and reducing attention to minority classes.



(a) Class distribution of the target variable in the training set

(b) Per-class F1-scores across classes with different frequencies

Fig. 1: Illustration of multi-class imbalance in BPIC2019 event log.

This model bias is concerning in many applications, because minority classes could represent infrequent yet important business activities. For example, in the BPIC 2017 event log, low-frequency events such as "*Application_Denied*" (0.32%), "*Application_Cancelled*" (0.88%), and "*Offer_Accepted*"(1.46%) are tied to key business decisions. Failing to accurately predict such events can result in missed opportunities for risk mitigation or strategic planning.

Therefore, given both the importance of addressing multi-class imbalance and the insufficient exploration of relevant ML solutions in the current PPM literature, this work aims to investigate the use of CBFL as a potential solution to bridge this gap. Although resampling is a possible alternative for handling class imbalance, this study focuses on loss re-weighting due to the known drawbacks of resampling [7,20], i.e., loss of information from under-sampling and overfitting from over-sampling.

3 Preliminaries

Event logs generated by information systems record business process executions and provide essential data for PPM. In event logs, an instance of a process execution is a case, which consists of a series of events, each reflecting an executed activity with a timestamp. An event can be defined as a tuple $e=(a,t,c,d_1,...,d_m)$, where $a \in A$ is the activity label (A is the set of all distinct activity labels), t is the timestamp, c is the case ID, and $d_1,...,d_m$ are optional attributes. Let $\pi_{\mathcal{A}}(e)=a$ be the function that maps an event e to its activity label a. A trace $\sigma_c=\langle e_1,e_2,...,e_{|\sigma_c|}\rangle$ consists of temporally ordered events with the same case ID c. A prefix of length k $(0<k<|\sigma_c|)$ is a partial trace: $hd^k(\sigma_c)=\langle e_1,...,e_k\rangle$. Based on these concepts, we define the next activity prediction task as follows:

Definition 1 (Next Activity Prediction)**.** The objective of next activity prediction is to learn a function that, given a prefix $hd^k(\sigma_c)=\langle e_1,...,e_k\rangle$, predicts the activity label of the next event e_{k+1}, i.e., $\pi_{\mathcal{A}}(e_{k+1})$.

To predict the next activity, ML models are typically trained using a loss function that defines the optimization objective, with CEL being the conventional choice. For each instance i, the predictive model outputs one probability $\hat{p}_{i,j}$ for each activity label $j \in A$. These probabilities are usually produced via a softmax function in ML models, such that $\sum_{j\in A}\hat{p}_{i,j}=1$. To compute CEL, let $a_i \in A$ be the true next activity label for instance i, and $\hat{p}_{i,a_i}$ be the predicted softmax probability assigned to a_i, the CEL for i is calculated as:

$$L_{\text{CEL}} = -\log(\hat{p}_{i,a_i}) \tag{1}$$

4 Class-Balanced Focal Loss for Next Activity Prediction

Class-Balanced Focal Loss (CBFL) is a loss re-weighting method that integrates the principles of Focal Loss and Class-Balanced Loss. Focal Loss [10] is a widely validated technique for DL tasks involving high class imbalance. It operates at the instance level by down-weighting well-classified instances, but can be combined with class-level weighting to further emphasize minority classes. To incorporate class-level weighting, a common approach is to simply assign class weights inversely proportional to class frequencies (inverse class weighting). Class-Balanced Loss [7] builds on this idea but offers a smoother weighting scheme (as detailed in Section 4.1).

CBFL extends the standard CEL with two additional weighting terms, as defined below:

$$L_{\text{CBFL}} = -\frac{1-\beta}{1-\beta^{n_{a_i}}}(1-\hat{p}_{i,a_i})^{\gamma}\log(\hat{p}_{i,a_i})$$

where a_i is the true next activity label of i, $\hat{p}_{i,a_i}$ is the predicted softmax probability assigned to a_i, n_{a_i} is the number of training instances in class a_i, and $\beta \in [0,1)$ and $\gamma \in [0,\infty)$ are tunable hyperparameters.

4.1 Class-Level Weighting

The Class-Balanced (CB) term $\frac{1-\beta}{1-\beta^{n_{a_i}}}$ assigns the same weight to all instances within a class, based on the class frequency n_{a_i} and a hyperparameter β. As a result, infrequent classes receive relatively higher weights, yet the CB term improves upon the naive inverse frequency weighting by up-weighting minority classes in a smoother manner. As illustrated in Table 1, this smoothing effect functions in two dimensions. First, it produces modest weight differences among sufficiently large classes (e.g., class with 10,000 to 100,000 instances), while still assigning significantly higher weights to rare classes. This aligns with the observation in Figure 1b that frequent classes generally perform well despite variations in class frequency. Second, it narrows the weight gap between the most and least frequent classes, which is particularly crucial for highly imbalanced datasets. For example, in BPIC2019, the most frequent class has over 100,000 times more instances than the least frequent class. In such cases, inverse class weighting may assign excessively large weights to rare classes, increasing the risk of overfitting. Additionally, the degree of smoothing is controlled by the hyperparameter β, and as β increases, the smoothing effect decreases.

Table 1: Illustrative examples of class weights produced by inverse class weighting and the CB term under different β and n_{a_i} values (weights are scaled by 1000 for readability). For instance, when β=0.9999, a class with 10 instances is assigned a weight of 100.05.

		Number of samples in a class (n_{a_i})				
		100,000	10,000	1,000	100	10
CB term	β=0.99	10.00	10.00	10.00	15.77	104.58
	β=0.999	1.00	1.00	1.58	10.50	100.45
	β=0.9999	0.10	0.16	1.05	10.05	100.05
inverse class weighting		0.01	0.10	1.00	10.00	100.00

4.2 Instance-Level Weighting

The focal term $(1-\hat{p}_{i,a_i})^{\gamma}$ assigns instance-specific weights based on the predicted softmax probability $\hat{p}_{i,a_i}$ for the true class and a hyperparameter γ. As $\hat{p}_{i,a_i}$ is updated after each training epoch, the focal term adapts dynamically during training. It down-weights well-classified instances (i.e., those with high $\hat{p}_{i,a_i}$) and emphasizes harder instances, with this effect becoming stronger as γ increases. The incorporation of instance-level weighting is motivated by the observation that classes with similar frequencies can still vary in predictive performance (Figure 1b), which cannot be accommodated by solely relying on frequency-based class-level weighting.

5 Empirical Evaluation

The experimental setup and corresponding results are discussed in this section. The code is available in our repository: https://github.com/Xiaomeng-He/CBFL.

5.1 Experimental Setup

Event Logs Empirical evaluation is performed on four real-life event logs. We select these datasets because they provide sufficient data for DL models and exhibit diversity

in class distributions. BPIC2017[1] records loan applications from a Dutch financial institution. BPIC2019[2] tracks a purchase-to-pay process for a multinational coatings company. BPIC2020 Request For Payment[3] (BPIC2020) records the process of handling expense reimbursement requests at a university. BAC[4] logs the luggage handling process of a European airport. Table 2 presents key statistics of the event logs after pre-processing. To describe the degree of multi-class imbalance in each dataset, we report max./min. class freq., i.e., the number of training instances in the most and least frequent classes, respectively, and majority percent, i.e., the percentage of training instances from majority classes (defined later in this section). Distributions of the target variables in the training, validation, and test sets are shown in Figure 2.

Table 2: Summary statistics of event logs. Case duration is recorded in days, except for BAC (recorded in seconds).

Event log	Num. cases	Num. events	Num. activities	Max.case length	Avg.case length	Max.case duration	Avg.case duration	Max./min. class freq.	Majority percent
BPIC2017	28,977	1,067,714	25	87	36.85	47.80	20.57	117,543/173	63.58%
BPIC2019	169,142	907,557	36	13	5.37	143.33	72.55	100,530/1	95.63%
BPIC2020	5,753	29,902	17	10	5.20	28.83	9.00	3,570/1	73.48%
BAC	362,506	1,964,221	57	26	5.42	6728.00	731.36	231,869/1	89.35%

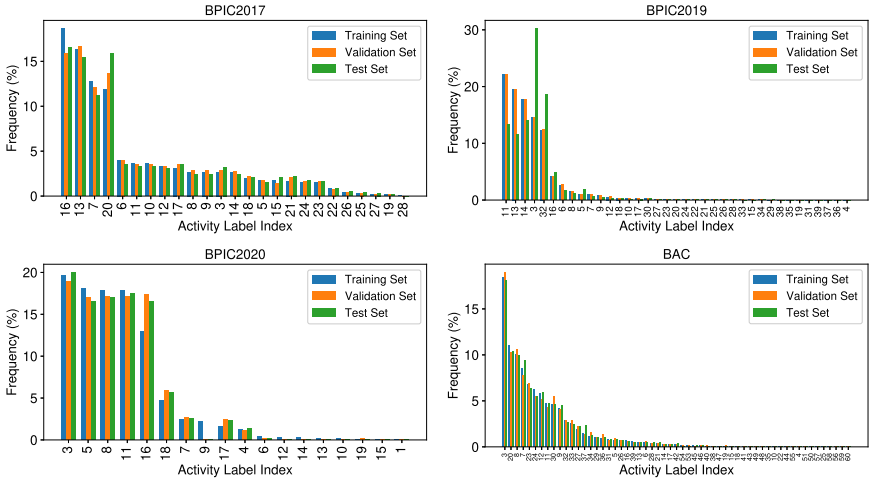


Fig. 2: Distribution of the target variable (represented by the activity label index).

Data Pre-processing and Splitting The data pre-processing procedure described in [21] is applied to remove chronological outliers, and cases with excessively long length or duration. Moreover, we adhere to a strict temporal train-test splitting: cases are

[1] 10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
[2] 10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1
[3] 10.4121/uuid:52fb97d4-4588-43c9-9d04-3604d4613b51
[4] This dataset is not publicly available due to business confidentiality restrictions.

ordered by their start times, with the last 20% assigned to the test set and the rest to the training set; cases that are still ongoing when the first test set case begins are excluded from training to prevent incomplete traces from distorting the target distribution. 20% of training cases serve as validation set for early stopping and hyperparameter tuning.

Feature Encoding To construct the feature representation of each event, we follow the most commonly used setting in PPM [14]. The input includes activity labels and timestamps. Activity labels are encoded using one-hot encoding for XGBoost and embeddings for DL models. Two temporal features are derived from timestamps: (i) the time elapsed since the previous event in the trace; and (ii) the time elapsed since the start of the trace (i.e. the first event). Given the highly right-skewed distribution of these temporal features, a logarithmic transformation $ln(1+x)$ is applied, followed by min-max normalization. While prior studies have noted that additional features may improve predictive performance, we do not include them to ensure a systematic evaluation based on common features across all datasets, focusing on the models' intrinsic ability to handle class imbalance using the two different loss functions.

To prepare training instances for XGBoost, prefixes are extracted from each trace. For a trace σ_c, we use all possible prefixes with lengths $0<k<|\sigma_c|$. Prefixes are encoded using the index-based method [6] that preserves all events, and then flattened into a single feature vector for processing by XGBoost. For DL models, we use trace-based encoding to accelerate training, as recommended by [16], where each trace is treated as an instance under a many-to-many training architecture. Experiments show that this trace-based encoding achieves performance comparable to prefix-based encoding for DL models, and our encoding strategy for XGBoost also performs similarly to popular alternatives, including prefix aggregation encoding, which validates our choice. Full results can be found in the previously linked repository.

Implemented Architectures We train four models using both CEL and CBFL to comprehensively assess CBFL's effectiveness and applicability. Among these four architectures, XGBoost serves as a traditional ML baseline[5]. LSTMs and Transformers are included for their demonstrated strong performance in next activity prediction [4,14]. We also include xLSTMs, which, to our knowledge, have not been previously examined in the PPM context. Motivated by xLSTMs' recent success in other sequential modeling tasks such as language modeling [2] and time-series forecasting [1], as well as their superior performance over LSTMs in rare token prediction [2], we aim to investigate whether these strengths extend to next activity prediction.

Hyperparameter Tuning Hyperparameters related to model complexity and training are optimized separately for each architecture using grid search. When training with CBFL, the two hyperparameters β and γ are tuned on the validation set using grid search. Guided by [7,10] and initial experimental findings, we define the search space as $\beta \in \{0,0.99,0.995,0.999,0.9999\}$ and $\gamma \in \{0,0.5,1,2,3\}$. For comparison, the search space also includes pure class-level and instance-level weighting, obtained by setting $\gamma=0$ and $\beta=0$, respectively. We also examine inverse frequency weighting as a naive alternative to the CB term.

[5] For XGBoost, CBFL is implemented without the focal term, as XGBoost already emphasizes difficult instances by design.

Evaluation and Statistical Testing In classification, precision measures how often predicted positives are correct, and recall reflects the coverage of actual positives. Capturing more true positives may also introduce more false positives, yet a desirable model should achieve a balance to ensure complete and precise predictions. Therefore, we adopt F1-score, the harmonic mean of precision and recall, as our primary evaluation metric. When comparing model performance across majority and minority classes, we also report precision and recall for more granular insights. All metrics are computed per class and aggregated using macro-averaging to prevent majority classes from disproportionately influencing the evaluation results. For the formal definitions and computational details of these metrics in multi-class settings, we refer to [19].

In the results, we report overall performance (average across all classes), as well as separate averages for majority and minority classes. Majority classes are defined as those with frequencies above the 75th percentile; statistically, they represent the upper tail of the long-tailed distribution and account for a disproportionately large number of instances. The rest are considered minority classes.

For every dataset, each ML and DL model is trained and evaluated over 10 independent runs with different random seeds to account for stochasticity in training. We report the mean and standard deviation across these runs in Section 5.2. To assess whether differences between models trained with the two loss functions are statistically significant, we apply the Wilcoxon signed-rank test, comparing the paired results obtained from the same architecture on the same dataset under the two loss functions. A difference is considered significant when the p-value is below 0.05, the conventional threshold for statistical significance.

5.2 Results

This section presents performance comparisons of models trained with CEL and CBFL using different architectures, along with visualizations highlighting CBFL's advantages over inverse frequency weighting. In addition, we provide an analysis that does not rely on a fixed majority/minority threshold, but examines majority and minority F1-scores across all possible thresholds. In the following tables, all metric values are reported as percentages.

Effectiveness of CBFL Compared to CEL, training with CBFL generally improves the overall F1-score, with a few exceptions observed. As shown in Table 3, XGBoost, LSTM, and xLSTM achieve statistically significant improvements in F1-score on three of the four datasets, with BPIC2019 being the exception. For the Transformer model, CBFL yields statistically significant improvements on BPIC2020, while differences between the two loss functions are not significant on the remaining three datasets.

When trained with CEL, all models perform markedly worse on minority classes than on majority classes, although the extent of this gap varies by dataset. CBFL reduces this gap by enabling models to capture more true positives from minority classes, leading to consistent recall improvements for minority classes on all datasets and F1-score gains on three of them (Table 4a). Regarding the majority classes (Table 4b), the F1-scores are largely not negatively affected when switching to CBFL. The only exception is BPIC2020, where the F1-score under CEL is already near perfect; however, the score under CBFL remains competitively high. Moreover, CBFL also yields higher precision for the majority

classes across all datasets. This suggests that improving the model's sensitivity to minority classes also contributes to a reduction in false positive predictions for majority classes.

Table 3: Overall next activity prediction performance. The table reports the mean and standard deviation of F1-scores under two loss functions, together with the p-value. Results where CBFL outperforms CEL with statistical significance are highlighted in bold.

	BPIC2017		BPIC2019		BPIC2020		BAC	
	CEL	CBFL	CEL	CBFL	CEL	CBFL	CEL	CBFL
XGBoost	76.37±0.21	**78.45±0.21**	28.28±0.10	27.34±0.10	50.88±0.20	**53.16±0.42**	43.79±0.07	**45.44±0.04**
	$p=0.002$		$p=0.002$		$p=0.002$		$p=0.002$	
LSTM	76.23±0.58	**77.50±0.44**	28.92±0.32	**29.64±0.40**	48.40±2.28	**50.65±1.24**	42.71±0.56	**44.23±0.55**
	$p=0.002$		$p=0.004$		$p=0.006$		$p=0.004$	
Transformer	77.66±0.47	78.08±0.57	29.90±0.49	29.50±0.76	48.41±1.97	**50.18±0.66**	43.38±0.83	43.56±1.12
	$p=0.160$		$p=0.105$		$p=0.002$		$p=0.557$	
xLSTM	76.90±0.80	**78.30±0.69**	28.91±0.55	29.42±0.51	49.22±0.33	**51.68±0.85**	43.01±0.57	**43.99±0.79**
	$p=0.010$		$p=0.131$		$p=0.002$		$p=0.013$	

CBFL Tuning Hyperparameter tuning reveals that model performance is more sensitive to the choice of β than γ, and that an appropriately tuned CB term (i.e., with a well-chosen β) always yields higher F1-scores than the inverse frequency weighting. This finding, illustrated in Figure 3, supports the use of the more sophisticated CB term. Additionally, the trade-off between precision and recall is also evident during the tuning of β and γ. Since a model that balances precision and recall is generally preferred, the reported CBFL results in Table 3 and Table 4 correspond to the configuration of β and γ that yields the highest F1-score on the validation set.

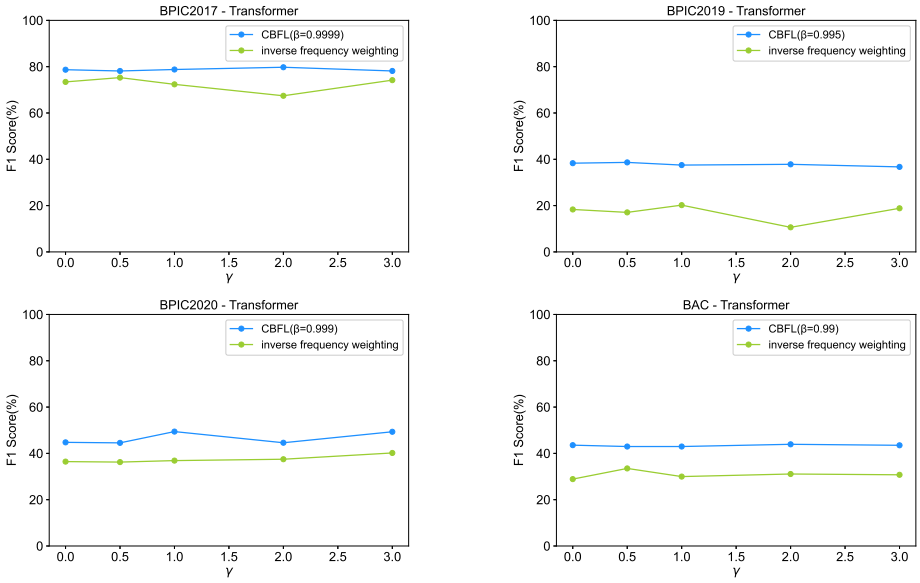


Fig. 3: F1-scores on the validation set, obtained using the CB term (with the tuned β) and inverse frequency weighting combined with the focal term at different γ values.

Table 4: Next activity prediction performance for minority and majority classes. Best F1-scores are bold and underlined; best precision and recall are bold only.

	BPIC2017			BPIC2019			BPIC2020			BAC		
	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec
Baseline: CEL												
XGBoost	72.81	79.26	71.01	17.09	25.11	15.45	36.90	**41.07**	36.70	32.32	50.54	28.28
LSTM	72.63	79.82	71.57	17.90	26.39	16.73	33.03	35.36	33.23	30.79	48.60	26.93
Transformer	74.44	**81.43**	73.28	**19.18**	**27.06**	18.14	33.01	35.10	33.17	31.57	49.19	28.41
xLSTM	73.46	80.33	72.34	17.89	25.34	16.64	34.23	37.73	34.39	31.24	49.31	27.78
CBFL												
XGBoost	**75.53**	76.39	76.22	15.78	21.53	14.48	42.31	39.86	**50.79**	**34.58**	**52.92**	**31.02**
LSTM	74.32	75.75	75.97	18.98	23.98	**21.91**	38.04	39.47	40.57	32.89	49.16	30.87
Transformer	75.09	77.15	**77.61**	18.69	26.23	18.85	38.79	39.25	43.73	31.84	49.41	30.55
xLSTM	75.33	76.52	77.31	18.69	22.84	21.82	**40.95**	40.50	44.93	32.68	50.05	30.60

(a) Minority Classes

	BPIC2017			BPIC2019			BPIC2020			BAC		
	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec
Baseline: CEL												
XGBoost	90.59	88.59	**92.75**	60.62	63.01	61.62	95.06	92.09	98.97	78.27	81.90	79.08
LSTM	90.63	88.78	92.61	60.77	66.98	61.99	95.08	91.74	99.49	78.03	81.56	78.96
Transformer	90.56	88.83	92.42	**60.87**	66.88	**62.06**	95.14	91.86	99.49	78.13	81.74	78.96
xLSTM	**90.66**	88.82	92.66	60.75	65.96	62.00	**95.25**	91.93	**99.64**	77.98	81.51	78.88
CBFL												
XGBoost	90.16	89.25	91.12	60.71	63.23	61.76	90.86	94.42	87.97	**78.28**	**81.91**	**79.09**
LSTM	90.24	**89.38**	91.18	60.43	**68.04**	61.45	91.69	94.97	89.64	77.92	81.61	78.78
Transformer	90.02	88.84	91.34	60.72	67.83	61.96	88.36	**97.42**	84.23	77.98	81.42	78.91
xLSTM	90.18	89.27	91.16	60.43	67.12	61.32	88.76	97.05	84.70	78.00	81.70	78.78

(b) Majority Classes

Threshold-Independent Analysis In addition to Table 4 that is based on a fixed majority/minority threshold, we also provide a more comprehensive performance comparison across all possible thresholds. Figure 4 presents an illustrative analysis on the BAC dataset using the LSTM architecture. All classes are sorted by descending frequency, and the x-axis indicates the number of classes included in the majority class group, starting from the most frequent. For example, $x=5$ means that the top 5 most frequent classes are considered majority, with the remaining classes as minority. Moving from left to right along the x-axis, the majority group expands by incrementally including infrequent classes, while the minority group shrinks by incrementally excluding frequent classes.

This visualization confirms that CBFL's effectiveness is consistent across different definitions of majority/minority threshold (the threshold used for Table 4 is marked by red line). Regardless of the threshold, the general trend holds: CBFL yields higher F1-scores for the minority group than CEL, while majority group performance remains comparable under the two loss functions.

Full versions of the visualization shown in Figure 4, extended to all datasets and architectures, are provided in our repository. The repository also includes a detailed report with additional performance metrics such as AUC-PR and accuracy.

Fig. 4: F1-scores across different majority/minority thresholds.

6 Conclusion

Multi-class imbalance is a common challenge in next activity prediction, yet it has received relatively little attention in the PPM literature. Since training with the conventional CEL inherently biases models toward majority classes, this work investigates the effectiveness of CBFL, an alternative loss function that combines class-level and instance-level reweighting, to address this issue. Empirical results show that training with CBFL generally enhances performance on minority classes without negatively impacting performance on majority classes, leading to improvements in overall model performance.

This work has several limitations. First, the effectiveness of CBFL is sensitive to the choice of hyperparameters β and γ, which requires careful tuning. Second, while CBFL accounts for class frequency and instance difficulty, it does not consider the business importance of activities, which may vary by context and require business expertise. In future work, we aim to extend loss re-weighting techniques to other PPM tasks including suffix prediction, and investigate loss functions based on additional factors such as the frequency of process variants.

Acknowledgments This work was supported by the Research Foundation Flanders (FWO) under grant number G039923N, and Internal Funds KU Leuven under grant number C14/23/031.

References

1. Auer, A., Podest, P., Klotz, D., Böck, S., Klambauer, G., Hochreiter, S.: TiRex: Zero-Shot Forecasting Across Long and Short Horizons with Enhanced In-Context Learning (2025), https://arxiv.org/abs/2505.23719
2. Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., Hochreiter, S.: xLSTM: Extended long short-term memory. In: NeurIPS 2024. vol. 37, pp. 107547–107603 (2024)

3. Breuker, D., Matzner, M., Delfmann, P., Becker, J.: Comprehensible predictive models for business processes. MIS Q. **40**(4), 1009–1034 (2016)
4. Bukhsh, Z.A., Saeed, A., Dijkman, R.M.: Processtransformer: Predictive business process monitoring with transformer network (2021), https://arxiv.org/abs/2104.00721
5. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate LSTM models of business processes. In: BPM 2019. LNCS, vol. 11675, pp. 286–302. Springer, Cham (2019)
6. Ceravolo, P., Comuzzi, M., De Weerdt, J., Di Francescomarino, C., Maggi, F.M.: Predictive process monitoring: concepts, challenges, and future research directions. Process Sci. **1**(1), 2 (2024)
7. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples. In: CVPR 2019. pp. 9260–9269 (2019)
8. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decis. Support Syst. **100**, 129–140 (2017)
9. Johnson, J.M., Khoshgoftaar, T.M.: Survey on deep learning with class imbalance. J Big Data **6**, 27 (2019)
10. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. In: ICCV 2017. pp. 2980–2988 (2017)
11. Mehdiyev, N., Evermann, J., Fettke, P.: A novel business process prediction model using a deep learning method. Bus Inf Syst Eng **62**, 143–157 (2020)
12. Neu, D.A., Lahann, J., Fettke, P.: A systematic literature review on state-of-the-art deep learning methods for process prediction. Artif Intell Rev **55**, 801–827 (2022)
13. Pasquadibisceglie, V., Appice, A., Castellano, G., Malerba, D.: Using convolutional neural networks for predictive process analytics. In: ICPM 2019. pp. 129–136. IEEE (2019)
14. Rama-Maneiro, E., Vidal, J.C., Lama, M.: Deep learning for predictive business process monitoring: Review and benchmark. IEEE Trans. Serv. Comput. **16**(1), 739–756 (2023)
15. Rama-Maneiro, E., Vidal, J.C., Lama, M.: Embedding graph convolutional networks in recurrent neural networks for predictive monitoring. IEEE Trans. Knowl. Data Eng. **36**(1), 137–151 (2024)
16. Roider, J., Zanca, D., Eskofier, B.M.: Efficient training of recurrent neural networks for remaining time prediction in predictive process monitoring. In: BPM 2024. LNCS, vol. 14940, pp. 238–255. Springer, Cham (2024)
17. Tax, N., Teinemaa, I., van Zelst, S.J.: An interdisciplinary comparison of sequence modeling methods for next-element prediction. Softw Syst Model **19**, 1345–1365 (2020)
18. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer, Cham (2017)
19. Terven, J., Cordova-Esparza, D.M., Romero-González, J.A., Ramírez-Pedraza, A., Chávez-Urbiola, E.A.: A comprehensive survey of loss functions and metrics in deep learning. Artif Intell Rev **58** (2025)
20. Wang, S., Yao, X.: Multiclass imbalance problems: Analysis and potential solutions. IEEE Trans. Syst. **42**(4), 1119–1130 (2012)
21. Weytjens, H., De Weerdt, J.: Creating unbiased public benchmark datasets with data leakage prevention for predictive process monitoring. In: Business Process Management Workshops. BPM 2021. LNBIP, vol. 436, pp. 18–29. Springer, Cham (2021)