


Task-Free Continual Learning with Dynamic Loss for Online Next Activity Prediction

Tamara Verbeek, Ruozhu Yao, and Marwan Hassani 

Eindhoven University of Technology, Eindhoven, The Netherlands
t.a.m.verbeek@student.tue.nl r.yao@student.tue.nl m.hassani@tue.nl

Abstract. Continual learning, known also as lifelong learning, aims at designing learning models that can continuously and autonomously adapt to varying data concepts without forgetting previously collected knowledge. Such concepts are referred to as tasks. Predictive business process monitoring, which predicts future process steps, is crucial in dynamic environments where tasks are not previously specified and processes frequently change or face unpredictability. However, many existing frameworks assume a static setting, ignoring dynamic nature and concept drifts in processes, leading to catastrophic forgetting—where training over new data adversely affects the performance on previously learned tasks. This paper presents TFCLPM, a framework for online next activity prediction that operates without relying on predefined tasks and employs continual learning techniques to reduce catastrophic forgetting. The methodology combines a Single Dense Layer neural network with a continual learning algorithm designed to retain challenging historical samples and include a regularizer to stabilize model parameters. Extensive experimental evaluations with synthetic and real-world event logs highlight our optimal configurations. The proposed framework’s performance is compared against three existing online next activity prediction methodologies. Results show significant improvements in prediction accuracy, especially in scenarios with gradual or recurrent drifts, highlighting the framework’s robustness and efficiency, even with large datasets.

Keywords: Next-Activity Prediction, Concept Drift, Catastrophic Forgetting, Continual Learning, Task-free Learning, Dynamic Loss Function

1 Introduction

Process mining involves analyzing event logs from business processes to discover, monitor, and improve real processes by extracting valuable insights and patterns. It bridges the gap between data and process management, enabling organizations to visualize, understand, and optimize their operational workflows [17]. A recent focus on predictive process monitoring underscores its vital significance, as it allows organizations to dynamically predict the future paths of individual process instances [5]. The ability to predict the next activity in predictive process monitoring is crucial for foreseeing and preparing for future actions in business processes. By leveraging this predictive power, organizations can manage resources such as manpower, materials, and time proactively, ensuring they are utilized effectively to address upcoming needs.

Traditionally, machine learning models are trained with separate training and test sets, assuming all data is available upfront. However, in real-world business scenarios, next activity prediction is more effective in an online setting, where the model continuously learns from incoming data. This allows the model to stay current with the latest information and adapt to changes in the data, known as concept drifts [20], which occur when the statistical properties of the target variable change over time. By continually integrating new data, the model maintains accuracy and relevance, effectively responding to dynamic business environments.

Continual learning, or lifelong learning, is essential for improving online prediction. It allows models to retain past knowledge while adapting to new data, maintaining a deep understanding of evolving data. In dynamic real-world environments, predicting next activities accurately is crucial, as it ensures that models remain effective and relevant amid changing patterns. This adaptability is vital for maintaining efficiency, enhancing decision-making, and enabling proactive responses in areas like business workflows, manufacturing, and customer service, where timely predictions directly affect performance. Various strategies for predicting future activities in an online context are discussed in [6, 8, 21].

In the context of this paper, a *task* is not conceptualized as a step in a process, but as a learning task where a model must execute actions like next activity prediction on a given data distribution. In dynamic environments with continuously evolving data, traditional next activity prediction methods often rely on predefined tasks or concept drift detection to trigger model updates. However, these approaches have limitations, as defining tasks in advance or detecting drifts can be impractical in real-world scenarios where changes are subtle and unpredictable. A task-free approach to continual next activity prediction offers significant advantages. It eliminates the need for explicit task definitions, allowing the model to adapt seamlessly to shifting data distributions without manual intervention. This flexibility is essential in environments where activities are not clearly split into specific tasks or where drifts occur gradually.

Rather than waiting for concept drift detection, it is more effective to update the model based on loss function behavior [3]. By monitoring for *plateaus* followed by *peaks*, we can pinpoint when the model’s performance stabilizes and then declines, indicating that an update is needed. This method ensures timely updates, maintaining model accuracy while avoiding unnecessary updates that could lead to overfitting or resource waste. Additionally, it is advantageous to use a dynamic loss function that adapts based on the significance of the model’s parameters [1] because it helps the model to selectively retain critical knowledge while adapting to new information. By weighting the loss according to parameter importance, the model can prioritize preserving crucial parameters that have a significant impact on performance, thereby minimizing catastrophic forgetting.

The contributions of this paper include: i) introducing the first use of continual learning through a dynamic loss function for task-free online next activity prediction; ii) conducting a comprehensive experimental evaluation using both

synthetic and real-world datasets with various types of concept drifts; and, iii) demonstrating that our approach outperforms existing methods.

The remainder of the paper is organized as follows: Section 2 reviews a required background and most related work, followed by preliminaries and problem formulation in Section 3. Section 4 details our main approach, TFCLPM (Task-Free Continual Learning for predictive Process Mining). A extensive experimental evaluation is presented in Section 5 while a conclusion of this paper is presented in Section 6.

2 Background and Memory Aware Synapses

Studies explore incremental techniques to update predictive models with new process execution data. Pauwels et al. [13] compare various update strategies, including re-training with and without hyperoptimization and incremental updates, demonstrating the effectiveness of incremental updates in maintaining model quality while offering real-time adaptability.

Continual learning aims to enable machine learning models to learn and adapt continuously over time, much like how humans assimilate knowledge throughout their lives. Within this framework, researchers have explored various methods, each offering unique perspectives and strategies. These methods include memory-based, architecture-based, regularization-based, and prompt-based approaches.

Memory-based approaches involve storing and retrieving past experiences or knowledge. These approaches can be divided into two categories: the first retains actual past experiences, as seen in methods like Experience Replay [15], iCaRL [14], DynaTrainCDD [8], Maximally Interfered Retrieval [2], and Gradient Episodic Memory [11]. The second category generates past experiences during training, exemplified by Generative Replay [9]. **Regularization-based methods** aim to prevent catastrophic forgetting by constraining weight updates during training. This constraint can be achieved by determining the significance of each parameter for past tasks, like in Elastic Weight Consolidation [7] or determining how crucial each weight in the network is, such as Memory Aware Synapses [1]. Alternatively, the importance of parameters can be assessed based on their impact on output sensitivity, with selective penalties applied to key parameters to mitigate forgetting, which is done in Learning without Forgetting [10]. **Architecture-based approaches**, in contrast, prioritize adjusting the neural network’s structure to integrate new data while preserving existing knowledge. One approach involves dynamic architectures, which expand the network by adding more neurons or layers for each task. This allows the model to continuously grow and adapt without forgetting previous knowledge, as exemplified by methods such as Progressive Neural Networks [16]. **Prompt-based approaches** (e.g. DualPrompt [19]), a more recent addition to the continuum, introduce a novel perspective on continual learning challenges. These methods entail attaching static or adaptable “instructions”, also referred to as prompts, to direct the model’s behavior. These prompts can take various forms, such as specific input patterns, embeddings, or task-specific tokens which help the model to recall and apply knowledge from earlier tasks.

We further elaborate on Memory Aware Synapses (MAS) [1] as a memory-based technique that our method, TFCLPM, builds on. First, it evaluates the significance of each weight, also called parameter, in the network. Once the network has been trained on a task, the importance of a parameter is assessed by determining how changes to it could impact the network’s overall performance. The importance Ω_{ij} of a particular parameter θ_{ij} is calculated using Eq. 1:

$$\Omega_{ij} = \frac{1}{N} \sum_{k=1}^N \|g_{ij}(x_k)\| \quad (1)$$

where N represents the total number of data samples and $g_{ij}(x_k)$ indicates how the network’s output changes for a given input x_k when θ_{ij} is adjusted (see Eq. 2). For networks with multiple outputs, a simplified approach is recommended. Instead of evaluating changes for each output separately, the overall magnitude of the output change is measured with:

$$g_{ij}(x_k) = \frac{\partial[\ell_2^2(F(x_k; \theta))]}{\partial\theta_{ij}} \quad (2)$$

where ℓ_2^2 is the squared l_2 norm of the function output.

This approach reduces complexity by concentrating on the total change rather than individual output variations. When the network is trained on a new task, MAS prevents the forgetting of previously learned tasks by penalizing significant changes to important parameters. The model’s objective is thus a balance between performing well on the new task and preserving crucial parameters: Objective = $\text{Ln}(\theta) + \lambda \times \text{Penalty}$, where λ is a regularization parameter that controls the penalty for altering important parameters.

A modified version of MAS eliminates the need for explicit task definitions by continuously learning from a data stream and adapting to evolving data distributions [3]. The system identifies plateaus—moments when the network’s performance stabilizes—by tracking the mean and variance of losses in a sliding window (cf. Fig. 1). During these plateaus, the system saves snapshots of the network’s weights to compare current and past data streams, updating the importance weights of neurons to preserve knowledge while adapting to new data.

Additionally a “hard buffer” with a small set of challenging samples retained based on their high loss is introduced. This buffer helps evaluate neuron importance and contributes to creating a retraining dataset. Importance weights are calculated using a cumulative moving average, preventing rapid fluctuations and supporting stable learning. A regularization term is added to the loss function to retain critical parameters, avoiding overfitting. Overall, the system enables continual learning without predefined tasks, adapting to gradual data shifts and supporting continuous model retraining and performance improvement.

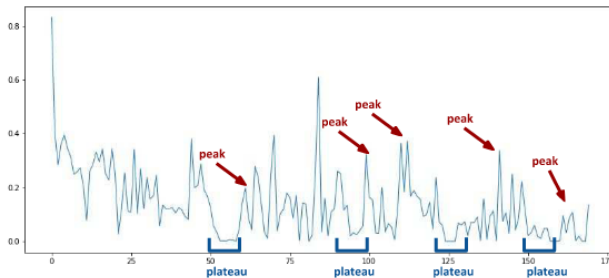


Fig. 1: Distinct plateaus and peaks are shown in loss values, which signal when to update the importance weights. X-axis represents update steps while y-axis shows the loss values [3].

3 Preliminaries and Problem Formulation

In this section we delve into the problem of utilising continual learning in process prediction. Assume we aim to develop an algorithm that continuously processes an event stream $S = \{e_1, e_2, \dots\}$ as events are generated, where $e = (c, a, t, v_1, \dots, v_A)$. An event e is a tuple of case identifier c , activity label a , timestamp t , and the values of the event attributes v_1, \dots, v_A . A case refers to a single instance of the process being analyzed or executed, encompassing all events, attributes, and contextual information associated with that instance. A case can include multiple traces, each representing different sequences of activities within the same instance. The stream S contains multiple traces. A trace $\sigma^{(i)} = \langle e_{1i}, \dots, e_{ni} \rangle$ denotes any finite sequence over the set of all events, related to a case. Given $\sigma^{(i)}$, the prefix represents the sequence of activities executed up to a certain point in a trace’s lifecycle. The prefix of length k is defined by $\sigma_{\leq k}^{(i)} = \langle e_{1i}, \dots, e_{ki} \rangle$. On the other hand, a suffix refers to the part of a process trace that occurs after a particular event or activity in that trace. Given $\sigma^{(i)}$, the suffix of events of length k is defined by $\sigma_{>n-k}^{(i)} = \langle e_{(n-k)i}, \dots, e_{ni} \rangle$. For each event e_i that occurs in stream S , the general problem is that we want to predict the next activity happening in the suffix $\sigma_{>n-k}^{(i)}$ based on the prefix $\sigma_{\leq k}^{(i)}$.

Definition 1 (Next activity prediction). *Let there be a sample of prefixes of sequences $\mathcal{P} = \{\sigma_{\leq k}^{(i)}\}_{i=1}^m$ where $2 \leq k < |\sigma^{(i)}|$ is the prefix length and m is the sample size. Given a prefix of an events sequence $\sigma_{\leq k}^{(i)}$, the next activity prediction is $\hat{a}_{(k+1)i}$ of the activity $a_{(k+1)i}$ happening in the beginning of the suffix $\sigma_{>n-k}^{(i)}$.*

Definition 2 (Online Next Activity Prediction). *We aim to perform ongoing predictions of the next activity on a stream of events. Upon the arrival of each new event e_{ki} , we utilize the prefix $\sigma_{\leq k}^{(i)}$ to forecast the subsequent activity $\hat{a}_{(k+1)i}$. The stream S may encompass multiple learning tasks \mathcal{T}_γ that the prediction model has to learn. We say that \mathcal{T}_p and \mathcal{T}_r , where $p \neq r$ represent two different learning tasks if they belong to two different processes separated by a concept drift, implying there is no relationship between these learning tasks.*

In the context of a stream of events involving multiple learning tasks, the distribution evolves over time, necessitating continuous updates to the model. In many scenarios, the model has to handle entirely new tasks that are distinct from previously encountered ones. This necessitates task incremental learning.

Definition 3 (Task Incremental Learning). *In task incremental learning, two tasks $n, m \in [1, \dots, N]$ have no correspondence to each other if $n \neq m$. Each task possesses unique objectives and is associated with a separate process, potentially necessitating the model to acquire new patterns, features, or behaviors.*

If we want to update the model after a concept drift, we aim to use as much data as possible. If a task reappears in a recurrent concept drift setup, it is advantageous to have stored data about this task to ensure that it is not forgotten. However, this poses challenges such as storage limitations or privacy concerns. Those often prevent the model from keeping all past data, letting it rely instead merely on recent data. This causes catastrophic forgetting [4].

Definition 4 (Catastrophic Forgetting). *Let there be a prediction model at any point in time that has learned a sequence of $\mathcal{T}_1 \dots \mathcal{T}_n$ learning tasks. When faced with the \mathcal{T}_{n+1} th task, a typical model tends to forget how to predict the next activities of the previously learned tasks.*

This phenomenon poses a significant challenge in dynamic environments. To address the issue of catastrophic forgetting, continual learning enables models to retain and incorporate knowledge from previous tasks while learning new ones.

Definition 5 (Continual Learning). *In continual learning, the model is designed to mainly mitigate catastrophic forgetting while quickly adapting to emerging tasks. This is done by leveraging the past knowledge in the knowledge base to help learn recurrent tasks. The objective is to optimize the performance of new tasks while minimizing performance degradation of previously learned tasks.*

Definition 6 (Continual Learning for Next Activity Prediction). *Consider a model that has learned to predict the next activities for an event stream S . Throughout S , concept drifts may occur. These concept drifts refer to the alteration in the learning task. The event stream contains a sequence of $\mathcal{T}_1 \dots \mathcal{T}_n$ learning tasks, where each learning task represents an individual process. A learning task for next activity prediction is defined by pairs of prefixes and suffixes of sequences \mathcal{P} . Given a prefix of an event sequence $\sigma_{\leq k}^{(i)}$, the learning task is to deliver a prediction \hat{a}_{k+1} of the activity a_{k+1} occurring in the suffix $\sigma_{>n-k}^{(i)}$ such that when presented with the \mathcal{T}_{n+1} th task, the model must accurately predict next activities for this task without suffering from catastrophic forgetting if it was a recurrent one. This necessitates maintaining the ability to predict the next activities for past prefixes while adapting to new patterns and variations in the event sequences as new events e arrive from S .*

Task-Free Continual Learning: In this work, we are designing the model in such a way that tasks are neither previously specified, nor their starting and ending timestamps are given. We refer to this by task-free continual learning. This imposes further challenges to the designed model to detect the drift and to also recognize the task while facing other continual learning challenges.

4 Task-Free Continual Learning for predictive Process Mining (TFCLPM)

We provide a comprehensive overview of the TFCLPM architecture, highlighting its critical components. The implementation is available on GitHub¹. Fig. 2 shows the framework architecture for training the model.

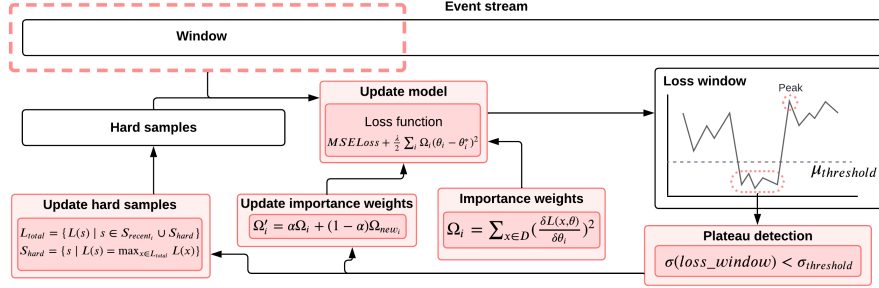


Fig. 2: Framework of the Task-Free Continual Learning for predictive Process Mining (TFCLPM).

Events from the event stream are stored in a window, which is a temporary storage for the most recent events. When 500 events accumulate, chosen following the experiments in [8], a model update is initiated. The events from the window are combined with hard samples to form the retraining dataset. At initialization, the hard samples dataset is empty. This retraining dataset is then fed into the Single Dense Layer (SDL) model, which undergoes several epochs to assimilate the new information. During model updates, a dynamic loss function is employed, combining the Mean Squared Error (MSE) loss with the Memory Aware Synapses (MAS) regularizer term defined as in Eq. 3:

$$MAS_{regularizer} = \frac{\lambda}{2} \sum_i \Omega_i (\theta_i - \theta_i^*)^2 \quad (3)$$

where θ_i represents the current value of a model parameter and θ_i^* is the value of that parameter prior to the update. This regularizer applies the calculated importance weights Ω_i to penalize significant changes in critical weights.

After each model update, the loss values for all samples in the retraining dataset are computed. For each sample, we determine the loss by evaluating $L_n(F(x; \theta), y)$, and then plotting the resulting values. Once the plot is generated, we need to analyze it to identify whether it contains a peak or a plateau. This involves examining the plot to detect regions where the loss value either reaches a maximum (peak) or stabilizes (plateau). To detect if a peak occurred, the condition $\mu(loss_window) > \mu' + \sigma'$ is checked, where μ' and σ' represent the mean and standard deviation of the previous window. Next, we calculate the

¹ <https://github.com/TamaraVerbeek/TFCLPM>

variance of the losses from the retraining dataset. We then verify if the conditions $\mu(l_{window}) < \mu_{th}$ and $\sigma(l_{window}) < \sigma_{th}$ are met, where μ_{th} and σ_{th} represent the predefined thresholds. This step is crucial for assessing whether the variance and the mean of the loss values are within acceptable limits, which helps in determining if the model requires an update based on stability criteria. If both metrics fall below a predetermined threshold, the importance weights and hard samples are updated accordingly.

Importance weights and hard samples update At initialization, the number of hard samples, denoted as H , is configured as a hyperparameter. To manage the hard samples effectively, we first calculate the importance of each sample in the retraining dataset based on its loss value, which has been previously assessed to determine the need for updates. Samples with higher loss values are deemed more challenging and therefore more critical for model performance. Consequently, the H samples with the highest losses are selected and designated as hard samples.

As the model undergoes updates, the importance weights are revised accordingly. In parallel, the Memory Aware Synapses (MAS) regularizer is adjusted to reflect these changes. The importance values used in the MAS regularizer are updated using Eq. 4:

$$\Omega'_i = \alpha\Omega_i + (1 - \alpha)\Omega_{new_i} \quad (4)$$

where α is set to 0.5, indicating that the updated importance values are influenced equally by both the previous and current values. This balanced approach ensures that the regularizer effectively accounts for both historical and recent data, facilitating more stable and accurate model updates. The new importance values Ω_{new_i} are determined by the Eq. 1.

5 Experimental Evaluation

Evaluation Metrics To effectively measure the performance of our proposed approach, it is essential to have evaluation metrics that offer a nuanced understanding of our model’s performance dynamics.

Accuracy at a given event index is a performance metric used to assess the effectiveness of a predictive model at a specific position within the sequence of events. In mathematical terms, if we denote γ as the size of the average accuracy window, and \hat{y}_j as the predicted event at index j based on preceding events, with y_j representing the actual event at index j , the accuracy at index i can be expressed as: $accuracy_i = \frac{1}{\gamma} \sum_{j=i-\gamma}^i 1\{\hat{y}_j = y_j\}$ (5). The formula computes the ratio of correct predictions from j up to event index i , averaging the predictions in the window. **Average accuracy** is calculated by averaging across all events to determine whether each one is predicted correctly. **Running Time** encapsulates the total duration the approach requires to process the entire event stream.

The Datasets We provide an overview of the datasets utilized in our study comprising a diverse selection of both real-world and synthetic datasets.

Synthetic Datasets: Business Process Drift The *Business Process Drift* dataset is a synthetic compilation designed to serve as a benchmark for the study

of business process changes. To simulate drift in a log, the authors of [12] systematically altered a base model by applying one of twelve simple change patterns, resulting in a total of 5.0000 events per simulated log containing 17 unique events. For more details on these simple change patterns, we refer interested readers to [12].

Real-World Datasets The *BPI Challenge 2020* dataset captures two years of travel expense claims at a university, with 6.000 to 10.000 cases showing gradual drift. It includes five subsets where only three are used in this research: Domestic Declarations and Request for Payment average 5 events per case while International Declarations has 11 on average. The *BPI Challenge 2015* dataset merges building permit applications from five municipalities into one event stream, with four sudden concept drifts, 5.600 cases, 181 unique events, and an average of 37 events per case. The *BPI Challenge 2017* dataset includes 5.168 loan application cases with 45 unique events and avg of 18 events/case.

The Competitors The performance of the model is compared with three competitors. Among these competitors are two baseline methods. The **Incremental Update** ($w = 1$) approach [13] involves updating the model every month to incorporate the most recent data. In the **Incremental Update** ($w = \text{Last Drift}$) approach [13], the model is updated after each window of data based on the historical data up to the last observed concept drift. Next to this, a state-of-the-art method is used as a competitor. The method **DynaTrainCDD** [8] distinguishes itself through its advanced concept drift detection algorithm called PrefixCDD [6]. It continually monitors process data for deviations and utilizes Prefix Trees to represent and analyze process sequences efficiently. These detected drifts dynamically dictate the frequency of updates and the selection of datasets for retraining.

Parameter Selection The selection of parameters for our framework was driven by a combination of empirical experimentation and theoretical considerations. The primary goal was to optimize accuracy. After experimentation and analysis, the number of hard samples is set to 100, the MAS weight is set to 0.5, and the mean and variance thresholds for detecting a plateau are 0.2 and 0.1, respectively.

5.1 The Results

Tab. 1 displays the average accuracy across all methods and datasets. Notably, TFCLPM achieves the highest average accuracy for the synthetic datasets. DynaTrainCDD also shows strong performance, closely trailing TFCLPM. In contrast, both Incremental Update ($w = 1$) and Incremental Update ($w = \text{Last Drift}$) lag significantly behind TFCLPM and DynaTrainCDD. Since these datasets include recurring concept drifts, this metric demonstrates that our approach is highly effective in reducing catastrophic forgetting. For the real-world datasets, TFCLPM achieves the highest accuracy on the BPIC2020 datasets (DomesticDeclarations, InternationalDeclarations, and RequestForPayment), but not on BPIC2017 or BPIC2015. This indicates that TFCLPM maintains consistent performance in datasets with gradual drift. Similar to the synthetic datasets, DynaTrainCDD closely follows TFCLPM, while Incremental Update ($w = 1$) and Incremental

	[13] ($w = \text{Last Drift}$)	[13] ($w = 1$)	DynaTrainCDD [8]	TFCLPM
IRO5000	<u>75.35</u>	75.33	<i>79.61</i>	80.43
ORI5000	<u>75.73</u>	74.21	<i>80.84</i>	81.88
RIO5000	75.02	<u>75.08</u>	<i>80.15</i>	81.06
ROI5000	<u>75.66</u>	73.87	<i>81.27</i>	82.85
OIR5000	68.23	<u>68.88</u>	<i>75.73</i>	77.36
InternationalDeclarations	<u>82.00</u>	80.79	<i>82.30</i>	82.83
DomesticDeclarations	83.72	<u>84.00</u>	<i>87.93</i>	88.69
RequestForPayment	83.73	<u>83.74</u>	<i>85.33</i>	87.65
BPI Challenge 2017	<u>75.47</u>	69.59	84.82	<i>83.49</i>
BPI Challenge 2015	<i>74.75</i>	75.25	69.16	<u>74.01</u>

Table 1: Average accuracy for all methods and datasets. **Bold** denotes the highest accuracy, *italic* the second highest, and underlined the third highest.

Update ($w = \text{Last Drift}$) lag significantly behind the other two methods. Arguably, BPIC2017 lacks concept drifts, allowing us to evaluate if our approach remains effective in their absence. In this case, our approach ranks second-best, while DynaTrainCDD achieves the highest average accuracy. However, both of these leading methods significantly outperform the other two approaches, highlighting their advantages. BPIC2015 is a challenging dataset due to its inclusion of four distinct processes and many unique events, which poses difficulties for our approach in handling these variations which shows in the lower average accuracy.

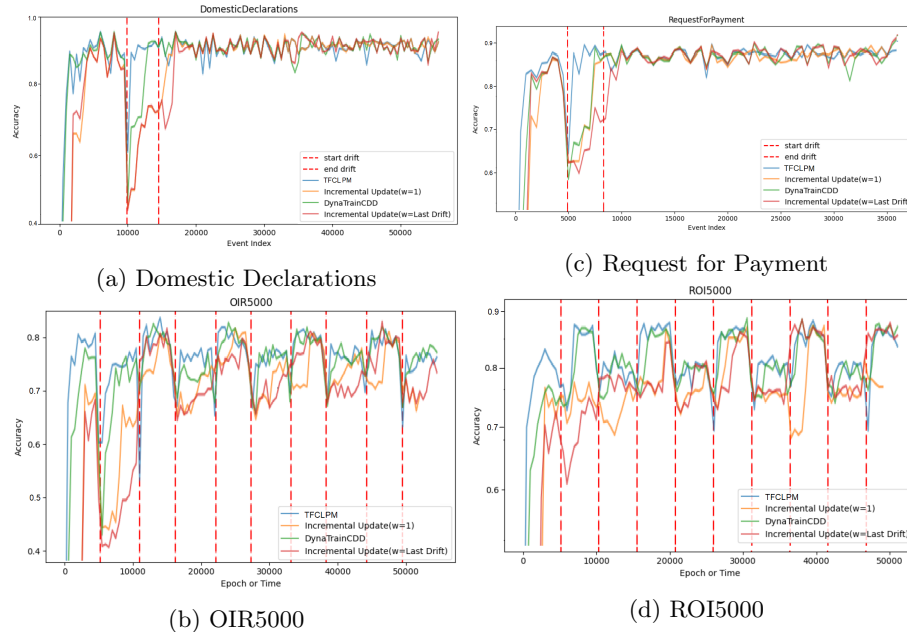


Fig. 3: Accuracy at a given event index for four different datasets. Vertical dashed lines represent a start (or an end) of a drift.

Fig. 3a and Fig. 3c illustrate the advantages of avoiding reliance on concept drift detection systems. It is evident that our approach restores accuracy much more quickly compared to Incremental Update ($w = 1$), Incremental Update

($w = \text{Last Drift}$), and DynaTrainCDD. The competitors take longer to detect concept drift, resulting in a slower return to an acceptable accuracy.

Fig. 3b and Fig. 3d demonstrate the effectiveness of our approach on datasets with recurrent concept drifts, highlighting the significance of the hard buffer and importance weights. Our method consistently achieves high accuracy faster than the others. In the OIR5000 dataset, when the drift occurs from the first to the second concept, DynaTrainCDD, Incremental Update ($w = 1$), and Incremental Update ($w = \text{Last Drift}$) encounter difficulties, whereas our approach swiftly recovers to around 75%. Although DynaTrainCDD gradually improves and eventually matches TFCLPM’s accuracy, our approach remains quicker to recover after each drift. A similar trend is observed in the ROI5000 dataset, where DynaTrainCDD lags slightly in regaining accuracy compared to our method’s rapid recovery. Tab. 2 demonstrates the running time for all approaches. In-

	[13] ($w = \text{Last Drift}$)	[13] ($w = 1$)	DynaTrainCDD [8]	TFCLPM
IOR5000	<u>305.03</u>	48.35	400.33	<i>112.67</i>
ORI5000	<u>257.98</u>	39.06	311.72	<i>131.76</i>
OIR5000	<u>297.69</u>	29.54	402.03	<i>104.33</i>
RIO5000	<u>241.52</u>	26.31	451.87	<i>97.33</i>
ROI5000	<u>215.18</u>	45.87	351.22	<i>120.11</i>
DomesticDeclarations	988.50	69.44	<u>775.95</u>	<i>267.69</i>
InternationalDeclarations	1801.68	87.95	<u>953.96</u>	<i>377.53</i>
RequestForPayment	<u>517.30</u>	50.76	530.05	<i>118.11</i>
BPI Challenge 2017	1605.53	361.68	<u>3198.75</u>	<i>1226.71</i>
BPI Challenge 2015	<u>5893.64</u>	2774.51	7194.67	<i>3518.07</i>

Table 2: Running times in Seconds for all methods and datasets.

cremental Update ($w = 1$) consistently performs efficiently across all scenarios while Incremental Update ($w = \text{Last Drift}$) often has longer running times, particularly with sudden drifts, due to its reliance on the last drift for updates. Our method offers stable and faster running times compared to DynaTrainCDD and Incremental Update ($w = \text{Last Drift}$). This is because our method maintains a constant retraining dataset size, ensuring a robust efficiency. On the other hand, Incremental Update ($w = \text{Last Drift}$) uses a varying dataset size based on drift detection, leading to longer running times in stable environments. Additionally, DynaTrainCDD runs a concept drift detection algorithm in parallel that further increases its running time. Our approach does not require an explicit drift detection component, resulting in its consistently higher efficiency.

6 Conclusion

In this work, we proposed TFCLPM, a novel approach for continual next-activity prediction aimed at mitigating catastrophic forgetting. This approach operates without predefined tasks, allowing it to adapt flexibly to evolving data. It addresses catastrophic forgetting by maintaining a buffer of hard samples and employing a dynamic loss function. We evaluated its performance using various metrics, including average accuracy and accuracy at a given event index. Through experiments on multiple synthetic and real-life datasets, we observed that TFCLPM achieves the highest average accuracy on eight out of ten datasets. It quickly recovers high accuracies after experiencing a concept drift and demonstrates minimal forgetting in cases of recurrent drifts compared to other methods.

Our method also shows stable running times compared to competitors. We plan to explore further continual learning methods for possibly other downstream prediction tasks, in a similar setup to [18] but for other prediction tasks.

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: ECCV. pp. 139–154 (2018)
2. Aljundi, R., Caccia, L.: Online continual learning with maximally interfered retrieval. In: NIPS (2019)
3. Aljundi, R., Kelchtermans, K., Tuytelaars, T.: Task-free continual learning. In: CVPR. pp. 11254–11263 (2019)
4. Chrysakis, A., Moens, M.F.: Online continual learning from imbalanced data. In: ICML. pp. 1952–1961. PMLR (2020)
5. Ferilli, S., Angelastro, S.: Activity prediction in process mining using the WoMan framework. *JiIS* **53**, 93–112 (2019)
6. Huete, J., Qahtan, A.A., Hassani, M.: PrefixCDD: Effective online concept drift detection over event streams using prefix trees. In: COMPSAC. pp. 328–333 (2023)
7. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., et al.: Overcoming catastrophic forgetting in neural networks. *CoRR* **abs/1612.00796** (2016)
8. Kosciuszek, T., Hassani, M.: Online next activity prediction under concept drifts. In: CAiSE Workshops. pp. 335–346 (2024)
9. Lesort, T., Caselles-Dupré, H., Garcia-Ortiz, M., Stoian, A., Filliat, D.: Generative models from the perspective of continual learning. In: IJCNN. pp. 1–8. IEEE (2019)
10. Li, Z., Hoiem, D.: Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* **40**(12), 2935–2947 (2017)
11. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. *Advances in neural information processing systems* **30**, 6470–6479 (2017)
12. Maaradji, A., Dumas, M., La Rosa, M., Ostovar, A.: Fast and accurate business process drift detection. In: BPM. pp. 406–422 (2015)
13. Pauwels, S., Calders, T.: Incremental predictive process monitoring: The next activity case. In: BPM. pp. 123–140 (2021)
14. Rebuffi, S., Kolesnikov, A., Sperl, G., Lampert, C.H.: iCaRL: Incremental classifier and representation learning. In: CVPR. pp. 5533–5542 (2017)
15. Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G.: Experience replay for continual learning. *Advances in Neural Information Processing Systems* **32** (2019)
16. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. *CoRR* **abs/1606.04671** (2016), <http://arxiv.org/abs/1606.04671>
17. Van Der Aalst, W.: Process mining: Overview and opportunities. *ACM Transactions on Management Information Systems (TMIS)* **3**(2), 1–17 (2012)
18. Verbeek, T., Hassani, M.: Handling catastrophic forgetting: Online continual learning for next activity prediction. In: CoopIS. p. to appear (2024)
19. Wang, Z., Zhang, Z., Ebrahimi, S., Sun, R., Zhang, H., Lee, C.Y., Ren, X., Su, G., Perot, V., Dy, J., et al.: DualPrompt: Complementary prompting for rehearsal-free continual learning. In: ECCV. pp. 631–648 (2022)
20. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. *Machine learning* **23**(1), 69–101 (1996)
21. Wolters, L., Hassani, M.: Predicting activities of interest in the remainder of customer journeys under online settings. In: ICPM Workshops. pp. 145–157 (2022)