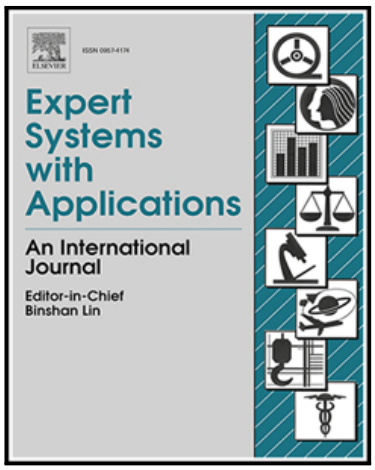


Multi-Task Learning for Predictive Process Monitoring with Temporal Graph Neural Networks on Object-Centric Event Logs

Dongseok Seo, Minseok Song

PII: S0957-4174(26)03172-6
DOI: https://doi.org/10.1016/j.eswa.2026.134266
Reference: ESWA 134266

To appear in: *Expert Systems With Applications*

Received date: 7 March 2026
Revised date: 9 August 2026
Accepted date: 1 September 2026

Please cite this article as: Dongseok Seo, Minseok Song, Multi-Task Learning for Predictive Process Monitoring with Temporal Graph Neural Networks on Object-Centric Event Logs, *Expert Systems With Applications* (2025), doi: https://doi.org/10.1016/j.eswa.2026.134266

Highlights

Multi-Task Learning for Predictive Process Monitoring with Temporal Graph Neural Networks on Object-Centric Event Logs

Dongseok Seo, Minseok Song

- We propose a Temporal GNN framework for predictive process monitoring on OCEL.
- A multi-task learning architecture jointly predicts multiple process outcomes.
- The proposed method captures continuous-time dependencies within object-centric event sequences.

Multi-Task Learning for Predictive Process Monitoring with Temporal Graph Neural Networks on Object-Centric Event Logs

Dongseok Seo[a], Minseok Song[a,*]

[a]Department of Industrial and Management Engineering, Pohang University of Science and Technology, 77 Cheongam-ro, Nam-gu, Pohang, 37673, South Korea

ARTICLE INFO

Keywords:
Predictive Process Monitoring
Temporal GNN
Object Centric Event Log
Multi-Task Learning

ABSTRACT

Predictive Process Monitoring (PPM) forecasts the future trajectory of ongoing process instances, including next activities and remaining execution times. While deep learning has advanced the field, modeling irregular temporal dynamics and long-range dependencies remains a significant challenge, especially within complex object-centric environments.

We propose a multi-task predictive framework based on the Temporal Graph Attention Network (TGAT). By transforming object-centric event logs into temporal graphs with time-aware edges, our approach explicitly encodes continuous time intervals directly into the attention mechanism. This allows the model to capture subtle temporal irregularities often missed by sequence-based methods. A shared temporal graph encoder is jointly optimized for three critical tasks: next activity prediction, next timestamp estimation, and remaining time prediction.

We evaluate our framework using four real-world BPI Challenge datasets (2014, 2016, 2017, and 2019). The results demonstrate that our model consistently outperforms or matches state-of-the-art sequence-based, graph-based, and Transformer architectures. Furthermore, attention visualizations and ablation studies confirm that integrating temporal graph modeling with multi-task learning significantly enhances predictive accuracy in complex process environments.

1. Introduction

Predictive Process Monitoring (PPM) provides the analytical foundation for proactive business process management. By forecasting future process behaviors—such as the next activity, the time until the following event, or the total remaining execution time—PPM enables organizations to anticipate delays, optimize resource allocation, and mitigate operational risks (Márquez-Chamorro, Resinas and Ruiz-Cortés, 2017; Maggi, Di Francescomarino, Dumas and Ghidini, 2014). As event data grows in both volume and complexity, PPM has evolved into a critical intersection of business process management and applied machine learning.

The emergence of Object-Centric Process Mining (OCPM) has revolutionized the field by addressing the limitations of traditional, activity-centric analysis. While classical process mining assumes that each event refers to a single case identifier, OCPM acknowledges the inherent complexity of modern enterprise systems, where an event can involve multiple interrelated objects. Central to this advancement is the Object-Centric Event Log (OCEL). Unlike traditional logs that flatten processes into single-case sequences, OCEL captures the intricate, many-to-many relationships between entities, such as orders, items, and invoices (Goossens, De Smedt and Vanthienen, 2024). While these logs preserve essential relational structures, most current PPM research still relies on "flattening" OCEL into simple time-ordered sequences to fit standard predictive models (Van der Aalst, 2019). Although this simplification retains basic temporal order, it often discards the rich relational signals inherent in multi-object interactions. The stage at which flattening occurs is, however, consequential: in the conventional pipeline the log is flattened at the input stage, before learning, so that the multi-object structure is discarded before the model ever sees it. In this work, OCEL is retained for instance construction — prediction targets, prefixes, and inter-event timings are all derived from the object-centric log — and each instance is derived under a single leading object type (Section 3); we treat this residual simplification — and its richer alternatives such as connected-component process executions — as a limitation and revisit it in Section 5.

Traditional neural architectures for PPM, such as LSTMs and GRUs, have dominated the field for the last decade, consistently outperforming classical machine learning approaches (Tax, Verenich, La Rosa and Dumas, 2017; Di Francescomarino, Ghidini, Maggi and Milani, 2018). However, these recurrent architectures suffer from two fundamental limitations: (i) they struggle to model long-range dependencies when inter-event time intervals vary significantly, and (ii) they are unable to incorporate the relational dependencies that define object-centric processes natively.

To address these architectural deficiencies, recent research has shifted toward Graph Neural Networks (GNNs) to encode the topological complexity of OCEL explicitly. Rather than treating events as isolated points in a sequence, GNNs represent events and objects as nodes in a dynamic graph, capturing the shared context among multiple process entities. However, while several studies have used static or heterogeneous graphs to represent object relations (Smit, Reijers and Lu, 2024; Gherissi, Acheli, Haddad and Grigori, 2024), these approaches often treat time as a secondary attribute—typically as a node feature—rather than as a primary structural component that governs the evolution of

*Corresponding author
seods99@postech.ac.kr (D. Seo); mssong@postech.ac.kr (M. Song)
ORCID(s): 0000-0002-6813-8853 (M. Song)

the graph. Recent benchmarks suggest that naively applying GNNs to OCEL without time-aware relational modeling fails to yield consistent improvements over classical sequence models (Galanti and de Leoni, 2023). This reveals a critical research gap: the potential of OCEL remains untapped because existing models lack a mechanism to synthesize complex structural relations with continuous temporal dynamics into a unified representation.

Temporal Graph Neural Networks (Temporal GNNs) offer a promising approach to these challenges by integrating timestamp information directly into the message-passing process (Rossi, Chamberlain, Frasca, Eynard, Monti and Bronstein, 2020). A notable advancement in this domain is the Temporal Graph Attention Network (TGAT) (Xu, Ruan, Korpeoglu, Kumar and Achan, 2020), which utilizes functional time encoding and temporal self-attention to capture complex, non-linear patterns across evolving interactions. Such architectures are uniquely suited for PPM, where the "rhythm" and inter-arrival times of a process are often as informative as the logical sequence of tasks. Despite this potential, the application of Temporal GNNs to OCEL-derived monitoring remains in its infancy, with few models capable of navigating the dual complexity of multi-object relations and continuous-time dynamics.

Recent work by Hennig and Schmidt has explored the application of Temporal Graph Networks (TGN) for predictive process monitoring(Hennig, Schmidt and Möhring, 2025), demonstrating the effectiveness of temporal graph representations for remaining time prediction. However, their approach is based on traditional case-centric event logs and primarily focuses on a single-task prediction setting. In contrast, our framework leverages object-centric event logs (OCEL) to construct object-centric sequences, enabling the modeling of multi-entity process behavior. Furthermore, we extend beyond single-task prediction by adopting a multi-task learning framework that jointly optimizes multiple interdependent prediction targets, including next activity prediction, next timestamp prediction, and remaining time prediction within a unified temporal graph learning architecture.

Limitation of existing approaches. Despite these advances, existing studies have largely addressed object-centric modeling, temporal graph learning, and multi-task prediction in isolation. Specifically, prior work either (i) models object-centric relationships without explicitly incorporating continuous-time dynamics, (ii) applies temporal graph neural networks in non-process or single-task settings, or (iii) adopts multi-task learning without integrating structural and temporal dependencies within a unified representation.

As a result, current approaches lack a cohesive framework that jointly captures the interplay between object-centric relational structure, temporal dynamics, and interdependent prediction targets.

Beyond structural and temporal modeling, an additional challenge in PPM arises from the inherent correlation between prediction targets, such as the next activity classification and the remaining time estimation. Multi-task learning (MTL) has emerged as a robust framework for exploiting these dependencies, improving the generalization of the model by allowing related tasks to share and refine internal representations (Zhang and Yang, 2021). Although recent studies have begun to explore multi-task architectures for PPM (Ke, Huan, Yifei and Chifeng, 2025), they often treat tasks as independent outputs at the final layer, lacking a deep structural justification for task synergy—particularly in a graph-based context. This suggests a significant opportunity to develop models in which object-centric relational features and temporal dynamics are jointly optimized to serve multiple predictive objectives simultaneously.

Building on these insights, this study proposes a **Multi-Task Temporal Graph Predictive Monitoring (MT-TGPM)** framework. By applying a Temporal Graph Attention Network to object-centric event sequences, our approach captures the intricate interplay between multi-object relations and continuous-time dynamics. The core contributions of this work are threefold:

- **Temporal Graph Representation of OCEL:** We transform complex object-centric event sequences into dynamic temporal graphs, employing a learnable time-encoding mechanism to effectively model the temporal irregularities and varying inter-arrival times inherent in real-world processes.
- **Integrated Temporal Graph and Multi-Task Learning Framework:** Unlike prior studies that address prediction tasks independently, we design a unified architecture in which a shared TGAT-based encoder jointly learns multiple interdependent tasks (next activity, next timestamp, and remaining time).
- **An Encoder–Objective Interaction Finding:** Beyond adopting a multi-task setting, we provide empirical evidence of a consistent interaction between the encoder family and the learning objective: temporal graph encoders (TGN, TGAT) tend to improve under joint multi-task training, whereas sequential encoders (LSTM, Transformer) tend to exhibit negative transfer under the same joint supervision (Section 4). This indicates that time-aware neighborhood aggregation is what allows heterogeneous supervision signals to reinforce, rather than interfere with, each other — a property of the combination rather than of any single component.
- **Empirical Validation on Large-Scale Datasets:** We rigorously validate our framework using four real-world datasets from the BPI Challenge (BPIC). The results demonstrate that the synergy between temporal graph modeling and multi-task learning consistently outperforms state-of-the-art non-temporal and single-task baselines across all metrics.

To support reproducibility, our implementation and the per-dataset preprocessing scripts are publicly available at `https://github.com/SeoDSeok/temporalGNN_ppm`.

The remainder of this paper is organized as follows: Section 2 reviews the related literature; Section 3 details the proposed methodology; Section 4 reports the experimental evaluation and ablation studies; and Section 5 concludes the paper.

2. Related works

2.1. Predictive Process Monitoring

Predictive Process Monitoring (PPM) focuses on forecasting the future behavior of running process instances by leveraging historical event data. Typical prediction targets include the next activity to be executed, the remaining processing time until completion, the time until the next event, or case-level results such as violations, risks, or costs (Márquez-Chamorro et al., 2017; Maggi et al., 2014). These predictions enable proactive operational interventions, such as strategic rescheduling of resources, prioritization of critical cases, or alerting stakeholders before Service Level Agreements (SLAs) are violated (Ceravolo, Comuzzi, De Weerdt, Di Francescomarino and Maggi, 2024). With the widespread adoption of process-aware information systems and event-driven architectures, PPM has emerged as a cornerstone of the Business Process Management (BPM) community.

Early approaches modeled PPM as a supervised learning problem centered on traditional case-centric event logs. Feature engineering played a vital role, with manually extracted control-flow patterns, aggregated resource statistics, and temporal indicators used to train tree-based or ensemble models (Márquez-Chamorro et al., 2017; Teinemaa, Dumas, Rosa and Maggi, 2019). The introduction of deep learning marked a paradigm shift from manual feature engineering to automated sequence learning (Kratsch, Manderscheid, Röglinger and Seyfried, 2021). In particular, Tax et al. demonstrated that long- and Short-Term Memory (LSTM) networks significantly outperform classical methods to predict the next event and its timestamp, while also enabling the generation of complete process traces (Tax et al., 2017; Rama-Maneiro, Vidal and Lama, 2021). Subsequent research has extended these models to incorporate high-dimensional event attributes, textual descriptions, and inter-case context, frequently employing attention mechanisms or Transformer architectures to capture complex dependencies (Bukhsh, Saeed and Dijkman, 2021; Pegoraro, Uysal, Georgi and van der Aalst, 2021; De Smedt and De Weerdt, 2024).

Despite these advances, recent surveys highlight two critical trends that suggest that current methodologies are reaching a plateau. First, most PPM methods still assume a traditional single case view, in which each event is strictly mapped to a single case, despite the inherently multi-object and cross-cutting nature of modern business processes (Ceravolo et al., 2024). Second, prediction tasks are typically configured as a single-task; separate models are often trained for the next activity and the remaining time, neglecting potential synergy and underlying correlations between these targets (Márquez-Chamorro et al., 2017). These observations motivate a transition toward representations that better capture multi-object behavior and architectures that jointly optimize multiple predictive objectives.

In addition to sequence-based models, GNN-based predictive process monitoring has developed into an active line of research. PROPHET represents prefix traces as heterogeneous multi-view graphs whose nodes encode characteristic values from the different views of an event log (activity, resource, discretized timestamps, trace attributes) and applies heterogeneous graph attention networks, together with an explainability component, for next-activity prediction (Pasquadibisceglie, Scaringi, Appice, Castellano and Malerba, 2024). Chiorrini et al. take a model-aware route: instance graphs are derived from a discovered Petri net via causal relations, enriched with temporal and data node features, and processed by deep graph convolutional networks for next-activity prediction (Chiorrini, Diamantini, Genga and Potena, 2023). PGTNet encodes each prefix as a compact directly-follows graph over event classes, attaches rich temporal edge features, and trains a GPS-style graph transformer for remaining-time prediction (Amiri Elyasi, van der Aa and Stuckenschmidt, 2024). A further representative, hereafter referred to as *DFG-GNN*, combines directly-follows graphs with graph neural networks for case-centric predictive monitoring (Lischka, Rauch and Stritzel, 2025). These approaches demonstrate that graph representations can effectively encode control-flow, data, and temporal information. However, they operate on case-centric logs and each addresses a single prediction task; moreover, time enters their representations as discretized categorical node values (PROPHET), as node features (Chiorrini et al.), or as edge features consumed only by the local message-passing blocks (PGTNet, whose global attention block is time-agnostic) — in none of them does continuous time directly modulate the attention weights of message passing. Notably, the authors of PGTNet themselves identify the extension to object-centric event logs and the investigation of multi-task learning as open future directions (Amiri Elyasi et al., 2024). The specific combination pursued in this work — OCEL-derived instances, continuous-time edge attributes that modulate attention weights, and joint multi-task objectives — therefore remains unexplored in this line of research.

Overall, the literature shows that (i) PPM is mostly studied on single-object, case-centric logs, (ii) predictions are typically configured as single-task models (e.g., only next activity or only remaining time), and (iii) temporal dependencies are modeled via RNN-style architectures rather than explicit temporal graph structures. Work leveraging OCEL and dynamic relational information for PPM remains limited, leaving room for approaches that better capture temporal and relational patterns simultaneously.

2.2. Object-centric event logs and graph-based PPM

Object-Centric Event Logs (OCEL) have been proposed to overcome the limitations of classical case-centric logs by

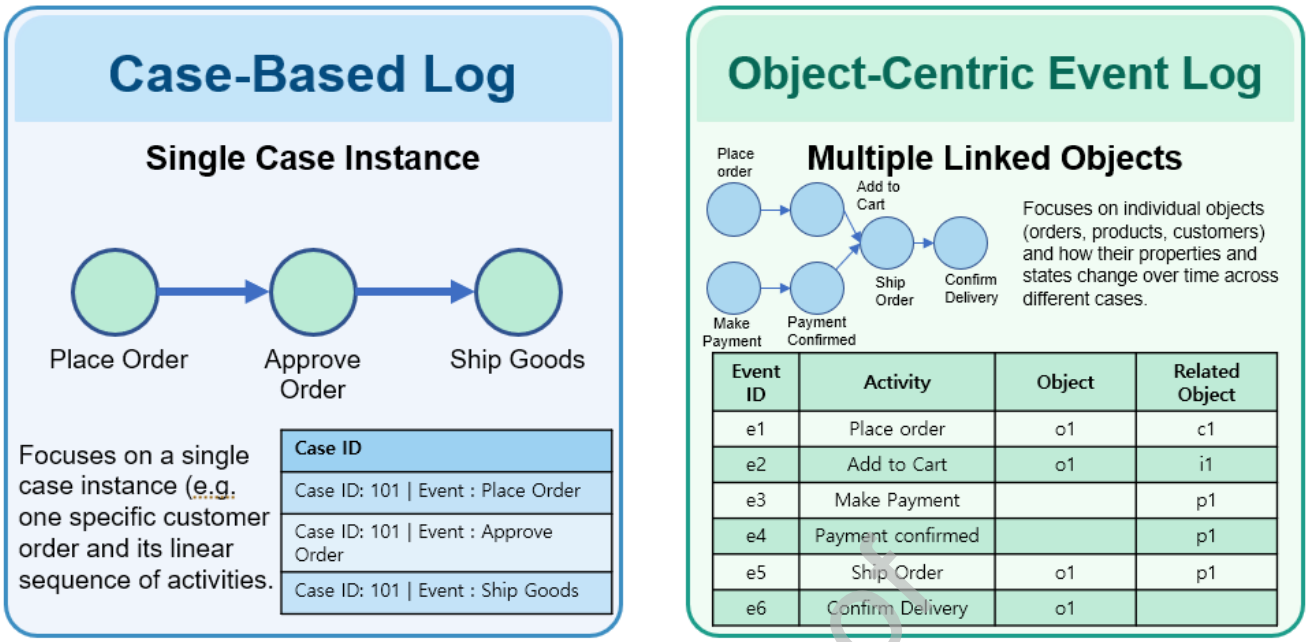


Figure 1: Comparison between traditional case-centric event logs and object-centric event logs (OCEL).

allowing a single event to be related to multiple business objects (e.g., orders, items, invoices and shipments) (Goossens et al., 2024). OCEL better reflects the intertwined nature of real-world processes, where different object types evolve concurrently and interact through shared events. However, this added expressiveness also complicates predictive modeling because the notion of "case" is no longer unique, and relational dependencies must be considered.

In contrast to traditional event logs, where each event is associated with a single case identifier, OCEL allows events to be linked to multiple objects simultaneously, producing a many-to-many relationship between events and process entities. For example, a single event such as "Create Invoice" may be related to both an order and multiple items, whereas in a case-centric log such relationships must be flattened into a single sequence — a transformation that often discards relational information and introduces ambiguity in process dependencies. Figure 1 illustrates this distinction: case-centric logs organize events as linear sequences tied to a single case, whereas OCEL preserves multi-object interactions at the cost of additional modeling complexity, namely (i) the notion of "case" is no longer unique and must be derived from a chosen object perspective, and (ii) relational dependencies between concurrently evolving objects must be considered. These limitations motivate the OCEL-aware case-generation procedure described in Step 1 of Section 3.

To make OCEL data amenable to machine learning, several encoding frameworks have been introduced. Adams et al. distinguish between sequence-based, graph-based, and tabular encodings for object-centric data, and show that naive flattening may lose necessary relational signals. In particular, they propose a graph-based OCEL encoding, hereafter referred to as *GOCEL*, which preserves object interactions at the cost of increased complexity (Adams, Park and van der Aalst, 2023). Comparative studies between classical and object-centric logs show that OCEL can improve prediction accuracy, but the best representation and model type depend on the prediction task and the characteristics of the process (Fioretto and Masciari, 2025).

A growing line of work explores Graph Neural Networks (GNNs) on top of OCEL. Galanti and de Leoni investigate whether GNNs actually bring benefits over strong non-graph baselines for object-centric predictive analytics, concluding that performance improvements are often modest and highly dependent on the chosen encoding and task (Galanti and de Leoni, 2023). Smit et al. propose HOEG, a heterogeneous object-centric event graph where events and objects form a multi-typed graph, and apply GNNs for remaining time prediction and outcome prediction (Smit et al., 2024). Their results show that graph-based models can be competitive, but they are typically evaluated on static snapshots rather than explicitly modeling temporal dynamics.

In parallel, several works adapt classical PPM techniques to OCEL by generating case notions and flattened traces from object-centric data, sometimes in combination with hand-made relational features (Adams et al., 2023). These studies demonstrate that well-designed flattening strategies can retain most of the predictive signal while simplifying the design of the model. However, they generally treat the resulting traces as purely sequential data, without further exploiting the underlying temporal graph structure induced by event adjacency.

2.3. Multi-task Learning

Multi-task Learning (MTL) trains a single model to solve multiple related tasks jointly, leveraging shared information across tasks to improve generalization and robustness. Caruana's classical work already emphasized that MTL can act as an inductive bias by "improving generalization by leveraging domain-specific information contained in training signals for related tasks" (Zhang and Yang, 2021). Modern deep MTL surveys distinguish hard parameter sharing, in which early layers are shared and later layers are task-specific, and soft parameter sharing, in which task-specific networks are regularized to remain similar (Vandenhende, Georgoulis, Van Gansbeke, Proesmans, Dai and Van Gool, 2021). These surveys also review architectural design, optimization strategies, and methods for learning task relationships (Ruiz, Alaíz and Dorronsoro, 2024).

MTL has been particularly successful in computer vision and natural language processing. In dense prediction tasks such as semantic segmentation and depth estimation, multi-task architectures share an encoder and use separate decoders, which produces better performance than isolated single-task networks (Vandenhende et al., 2021). In object detection frameworks such as Faster R-CNN and Mask R-CNN, classification and bounding-box (and sometimes mask) regression are trained jointly on a shared backbone, illustrating how related tasks can mutually regularize and exploit standard features. Similar patterns appear in NLP, where shared encoders enable concurrent tasks such as tagging, parsing, and textual inference (Zhang, Yu, Yu, Guo and Jiang, 2023).

In the PPM domain, although several studies have addressed different prediction targets, such as next activity, next timestamp, and remaining time, these tasks are typically modeled independently and their interactions are rarely analyzed. Although multi-task learning has occasionally been adopted in PPM settings, existing studies primarily report predictive performance rather than systematically investigate its effectiveness in exploiting shared temporal patterns across tasks(Wang, Sui, Liu, Shen, Li and Yu, 2024; Ke et al., 2025). Recent surveys on PPM further indicate that comprehensive analyzes of multi-task configurations remain scarce, leaving the potential benefits of jointly modeling correlated prediction objectives (e.g., next activity, inter-event time, and remaining time) largely underexplored (Ceravolo et al., 2024). This observation motivates our adoption of a multi-task learning architecture for PPM.

2.4. Temporal GNN

Graph Neural Networks (GNNs) have become a powerful tool for learning relational data, where instances are represented as nodes connected by edges encoding interactions or dependencies. Although early GNN models were designed for static graphs, many real-world systems—such as social networks, transaction logs, and sensor networks—evolve. This has led to the development of Temporal GNNs, which aim to model how node states and edges evolve as events occur.

Temporal Graph Networks (TGN) provide a generic framework for learning dynamic graphs in continuous-time represented as sequences of timestamped events. TGN combines a memory module with message passing and embedding updates, and can be instantiated to recover several existing dynamic graph models (Rossi et al., 2020). The Temporal Graph Attention Network (TGAT) proposes an alternative perspective: node embeddings are defined as functions of time, and neighborhood information is aggregated via self-attention, with functional time encoding that captures temporal patterns in node and edge features (Xu et al., 2020). TGAT has been shown to achieve strong performance in temporal node classification and link prediction tasks, and has been adopted in several application domains such as energy forecasting and infrastructure monitoring (Jia, Wang, Hosseini, Shou, Wu and Mao, 2024).

Despite this progress, applications of temporal graph neural networks to business process data remain relatively limited. Most existing process mining studies rely on static GNNs constructed from aggregated graphs or on purely sequential models. In contrast, temporal GNNs have only recently begun to attract attention in the PPM domain (Hennig and Schmidt, 2025). Although recent work has explored temporal GNN architectures such as TGN to model event-based process data, these approaches are typically evaluated in single-task settings and primarily focus on isolated prediction objectives (Hennig et al., 2025). Moreover, temporal graph models are more commonly assessed in social networks, citation networks, or interaction networks than in object-centric event logs derived from business processes (Rossi et al., 2020). To the best of our knowledge, no prior PPM approach simultaneously (i) represents OCEL-derived event sequences as temporal graphs, (ii) employs TGAT as a temporal graph encoder, and (iii) integrates a unified multi-task prediction framework for next activity, next inter-event time, and remaining time. This combination constitutes the specific gap that our work aims to address.

3. Methodology

This section details the proposed multi-task predictive monitoring framework, which leverages Temporal Graph Neural Networks (TGNNs) to handle the complexity of object-centric process data. The core of our architecture lies in its ability to transform Object-Centric Event Logs (OCELs) into dynamic, temporally structured graphs, which are subsequently processed by a Temporal Graph Attention (TGAT) mechanism. As illustrated in Figure 2, the methodology is structured into two primary phases:

(i) **OCEL-based Data Preprocessing:** In this stage, raw OCEL data are transformed into case-specific temporal sequences. These sequences are then converted into a graph representation where nodes represent events and objects, and edges capture their interactions over continuous time.

(ii) **Temporal Multi-Task Modeling:** A shared TGAT encoder learns high-dimensional latent representations

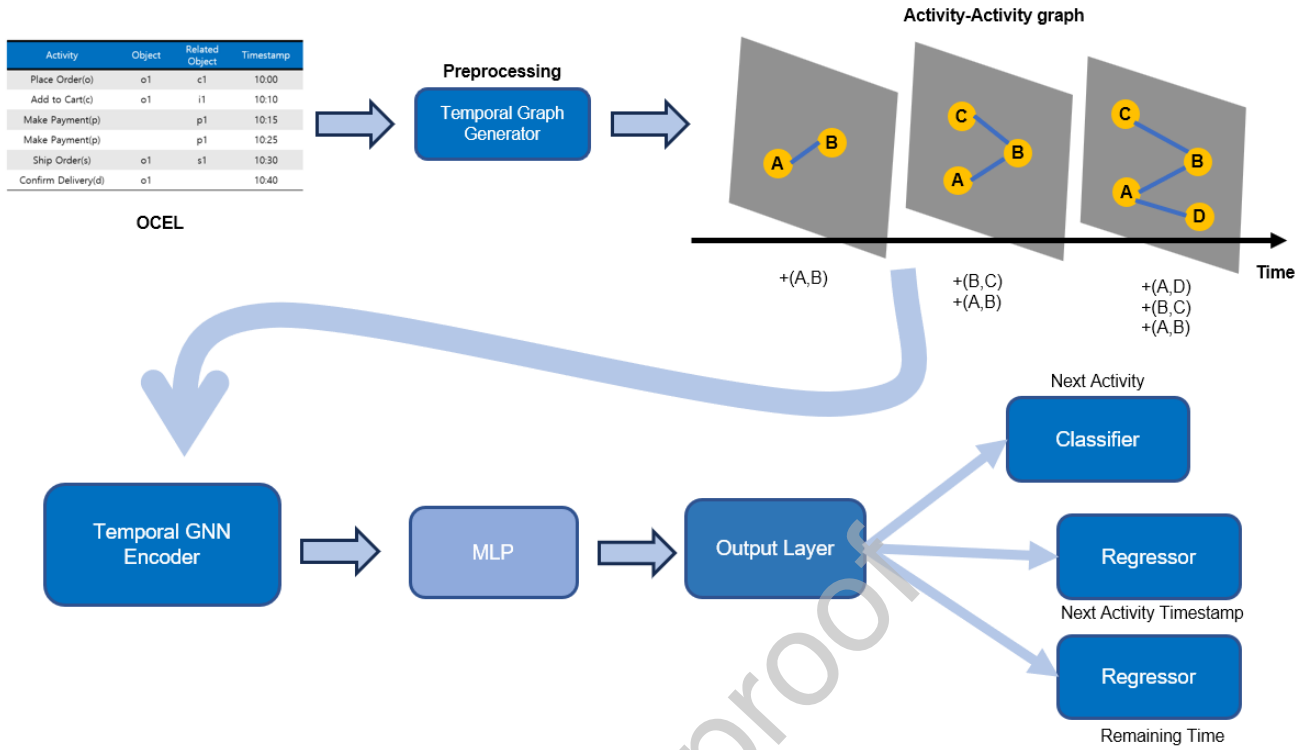


Figure 2: The architectural overview of the proposed Multi-Task Temporal Graph Predictive Monitoring (MT-TGPM) framework.

from the temporal graphs. These representations are then fed into multiple task-specific prediction heads that jointly optimize for next activity classification, next timestamp estimation, and remaining processing time prediction.

By integrating OCEL-aware feature engineering with temporal graph construction, the proposed approach preserves the intricate relational and temporal regularities inherent in modern business processes. This synthesis allows the model to leverage deep structural dependencies and continuous-time dynamics simultaneously, ultimately enhancing the precision of predictive process monitoring.

3.1. Data Preprocessing

Object-Centric Event Logs (OCEL) capture complex interactions between multiple business objects and events. However, most predictive tasks require a case-based sequence representation aligned with a specific process instance or object lifecycle. Consequently, the preprocessing stage restructures the OCEL into a format suitable for temporal graph modeling while preserving the causal and structural dependencies inherent in the data. The pipeline comprises four primary stages: case generation, Semantic Event Refinement, chronological sorting, and temporal graph construction.

Step 1: Case Generation

The initial step involves deriving meaningful "cases" from the multi-object environment of an OCEL. Unlike traditional event logs, OCEL does not enforce a single predefined case identifier; instead, it allows events to relate to multiple object types. Unlike classical case-based PPM approaches that rely on predefined case identifiers and flattened event logs, our framework leverages OCEL to derive object-centric instances that preserve execution semantics across related entities. To bridge this gap, we handle two distinct scenarios:

- **Explicit Case Identifiers:** If an explicit case identifier is provided (e.g., `ocel:oid` or a specific attribute like *Order ID*), events are grouped accordingly. This is typical in logs where a dominant object (e.g., a loan application) drives the lifecycle of the process instance.
- **Object type-Driven Sequences:** When a single case ID is absent, we derive sequences by selecting an object type whose lifecycle represents a coherent process behavior. In this setting, each individual object instance (e.g., a specific *Application ID*) is treated as an implicit case identifier. Events are grouped based on their association with the same object instance and then ordered chronologically to form a case-level sequence. For example, consider events associated with two application objects, *Application A1* and *Application A2*. All events linked to A1 are grouped together and sorted by timestamp to form one sequence, while events linked to A2 form a separate sequence. Each

such sequence is treated as an individual case for predictive modeling.

In both cases, each derived case corresponds to a sequence of events associated with the selected object instance. This step ensures that the model receives logically consistent input sequences while still reflecting the object-centric relationships present in OCEL.

Concrete case notions per dataset. For each dataset we designate one *leading* object type and let every instance of that type constitute a case: the *Interaction* object for BPI 2014, the customer *Session* for BPI 2016, the loan *Application* for BPI 2017, and the *Purchase Document* for BPI 2019. It is worth being precise about cardinality here, because two different relations are easily conflated. At the level of the raw object-centric log, the event-to-object relation is many-to many for *all* four datasets: a single event may refer to several object types and, within a type, to several instances. What differs across datasets is the cardinality *after* a leading type has been fixed, i.e. how many instances of that particular type a given event references. In BPI 2014 this residual relation is genuinely many-to-many: the leading type reference is a list, so an event associated with several *Interaction* instances is assigned to the sequence of every related instance — convergence is thus made explicit through event replication, following the standard flattening semantics for object-centric data (van der Aalst, 2019). In BPI 2016, 2017, and 2019, each event is processed under a single identifier of the leading type (the customer *Session*, the loan *Application*, and the *Purchase Document*, respectively), so after fixing the leading type each event belongs to exactly one case and no replication occurs. In all cases the resulting per-instance sequence is what the temporal graph is subsequently built on.

Relation to existing case-extraction techniques. Conceptually, the above procedure corresponds to flattening the OCEL under a leading object type, i.e., to deriving a case-centric view from an object-centric log (van der Aalst, 2019). More elaborate extraction techniques exist, most notably process executions based on connected components of the object interaction graph (Adams, Schuster, Schmitz, Schuh and van der Aalst, 2022). We deliberately adopt the simplest established extraction, for two reasons. First, the controlled variable in this study is the temporal representation: all compared models — object-centric and case-centric alike — consume exactly the same extracted instances, so that performance differences are attributable to the encoder rather than to the extraction step; introducing a novel or more elaborate extraction would confound the comparison. Second, prefix-based multi-task supervision requires well-defined per-instance labels (the next activity of a specific instance; the remaining time of a specific instance); connected-component process executions can merge many objects into a single large component under convergence, in which case these labels lose their operational meaning. Exploring richer execution-extraction strategies is left for future work (Section 5). Importantly, the contribution of this work does not lie in the extraction step itself but in the subsequent representation: the derived sequence is treated not as a static trace but as a continuous-time graph whose Δt edge attributes directly modulate the attention weights of the encoder (Section 3), a representation that the flattened trace alone does not provide to sequence models.

Step 2: Semantic Event Refinement

To capture the full granular complexity of business processes, we perform semantic enrichment of event labels. Raw OCEL data, such as the BPI Challenge 2017 dataset, often contain auxiliary columns describing event semantics: *Action* and *Lifecycle*.

As shown in Table 1, a single workflow activity can correspond to multiple combinations of action and lifecycle states. If only the original activity label was used, these execution-level distinctions would be ignored, resulting in ambiguous supervision signals. We redefine each event type by concatenating the *Activity*, *Action*, and *Lifecycle* attributes into a single **atomic activity label**.

This transformation enables the model to explicitly distinguish different execution states of the same activity, ensuring that the prediction task accurately reflects both the control-flow progression and execution semantics.

Step 3: Chronological Sorting

Once the cases are defined, the event sequences are sorted in strict chronological order. Sorting is crucial for capturing temporal dependencies and ensuring that graph edges represent valid state transitions. Given a case sequence $S = \{e_1, e_2, \ldots, e_n\}$, we ensure:

$$t(e_1) \leq t(e_2) \leq \cdots \leq t(e_n) \tag{1}$$

where $t(e_i)$ denotes the timestamp of the event e_i. This ordered structure serves as the basis for the subsequent temporal graph, reflecting the logical progression of the business process.

Step 4: Temporal Graph Construction

Modeling Scope Clarification. While OCEL inherently captures multi-object interactions, the proposed temporal graph is designed as a temporal backbone representation derived from object-centric sequences, rather than a full object-event relational graph. In particular, the current formulation focuses on modeling temporal dependencies among events within a given object's perspective, rather than explicitly capturing inter-object relationships. This design choice allows us to emphasize continuous-time dynamics via temporal graph attention, the primary focus of this work.

The ordered sequences are converted into directed temporal graphs, where nodes represent discrete events, and edges encode temporal transitions between them. This transformation procedure comprises two principal components:

(a) Calculating Temporal Edge Attributes For a target event e_i and any event e_j within its look-back window ($j < i$, $i - j \leq L$), we compute the elapsed time between them as

$$\Delta t_{ij} = t(e_i) - t(e_j). \tag{2}$$

Table 1
Example of Semantic Event Refinement (Mapping to Atomic Activities).

Original Activity	Action	Lifecycle	Refined Atomic Activity
W_Assess potential fraud	Deleted	WITHDRAW	W_Assess_fraud_Deleted_WITHDRAW
W_Assess potential fraud	Obtained	START	W_Assess_fraud_Obtained_START
W_Call incomplete files	Released	COMPLETE	W_Call_files_Released_COMPLETE
W_Validate application	Created	SCHEDULE	W_Validate_app_Created_SCHEDULE

The quantity Δt_{ij} is used as the *edge timestamp* of the edge (e_j, e_i) and is fed to the TGAT encoder, where it conditions the attention weight assigned to e_j when updating e_i (Section 3.2.1). The directly-following case $j = i - 1$ is thus the special case $\Delta t_{i,i-1}$. By explicitly representing these temporal intervals, the model can capture the characteristic temporal dynamics, or "rhythm," of the underlying process, thus facilitating the discrimination between typical (routine) transitions and atypical (anomalous) delays.

(b) Graph Representation Each case is formalized as a directed temporal graph $G = (V, E)$, where vertices V correspond to events and each event e_i is connected to its L most recent predecessors: $E = \{(e_j, e_i) \mid j < i,\ i - j \leq L\}$, where L is the look-back window size. The directly-follows relation $e_{i-1} \rightarrow e_i$ is therefore contained in E as the special case $j = i - 1$, but the neighborhood $N(e_i)$ is not restricted to it. The resulting graphs comply with:

- **Temporal Directionality:** All edges point from past to future ($j < i$), so no event can attend to information that was not yet available at its own timestamp; no causal or temporal shortcuts are introduced.
- **Temporal Awareness:** Each edge (e_j, e_i) carries the elapsed time Δt_{ij}, which modulates the attention weight α_{ij} within the TGAT layers, so that the same activity contributes differently depending on how long ago it occurred.
- **Process Fidelity:** The graph structure is constructed to faithfully replicate the actual execution trajectory of the process, allowing the model to capture and exploit the inherent structural dependencies among events.

The resulting output includes temporal graphs, node-level features (including activity encodings and associated attributes), and annotation labels corresponding to the multi-task learning objectives. An illustrative graph trace is presented in Figure 3.

3.2. Modeling

The proposed predictive monitoring framework is built upon a Temporal Graph Neural Network architecture that jointly learns three prediction targets: the next activity, the next inter-event time, and the remaining processing time of a running process instance. The model consists of three major components: (i) a Temporal Graph Attention Network (TGAT) encoder that transforms each event and its temporal neighborhood into latent node embeddings, (ii) a learnable time encoding module that converts raw inter-event intervals into high-dimensional temporal representations, and (iii) a multi-task output head composed of one classifier and two regressors. In the following subsections, we describe each component in detail.

3.2.1. Temporal Graph Encoder (TGAT)

Given a temporally ordered event sequence represented as a graph, the TGAT encoder aggregates information from temporal neighbors using attention-based message passing. Each event v is associated with an activity embedding $\mathbf{x}_v$ and a set of context nodes $N(v)$ consisting of the most recent events. TGAT computes an updated representation for v as:

$$\mathbf{h}_v = \sum_{u \in N(v)} \alpha_{vu}\, W\, \mathbf{x}_u \tag{3}$$

where W is a learnable transformation matrix and α_{vu} denotes the attention weight assigned to neighbor u. Unlike traditional GNNs, TGAT incorporates time-dependent attention by conditioning α_{vu} on both node features and their temporal encodings:

$$\alpha_{vu} = \frac{\exp\left(\phi(\mathbf{x}_v, \mathbf{x}_u, \mathrm{TE}(\Delta t_{vu}))\right)}{\sum_{u' \in N(v)} \exp\left(\phi(\mathbf{x}_v, \mathbf{x}_{u'}, \mathrm{TE}(\Delta t_{vu'}))\right)}, \tag{4}$$

where $\mathrm{TE}(\Delta t_{vu})$ denotes a continuous time encoding of the interval between events Δt_{vu}.

Multiple TGAT layers are stacked with residual connections and layer normalization, enabling the model to capture multi-hop temporal interactions within a case. This architecture allows the encoder to learn long-range temporal effects, such as cascading delays or rapid event bursts that significantly influence process behavior. Whereas prior temporal graph approaches such as TGN (Rossi et al., 2020) model temporal dynamics through a memory module combined with message passing, our framework instead employs temporal self-attention via TGAT (Xu et al., 2020) to directly capture continuous-time dependencies between events. Hennig and Schmidt (Hennig et al., 2025) apply TGN to case-centric event logs for single-task remaining-time prediction; our work differs in three respects: (i) it operates on object-centric event logs rather than case-centric logs, (ii) it jointly learns three prediction targets within a unified multi-task framework, and (iii) it replaces memory-based message passing with continuous-time self-attention, which we find more flexible for the irregular temporal patterns common

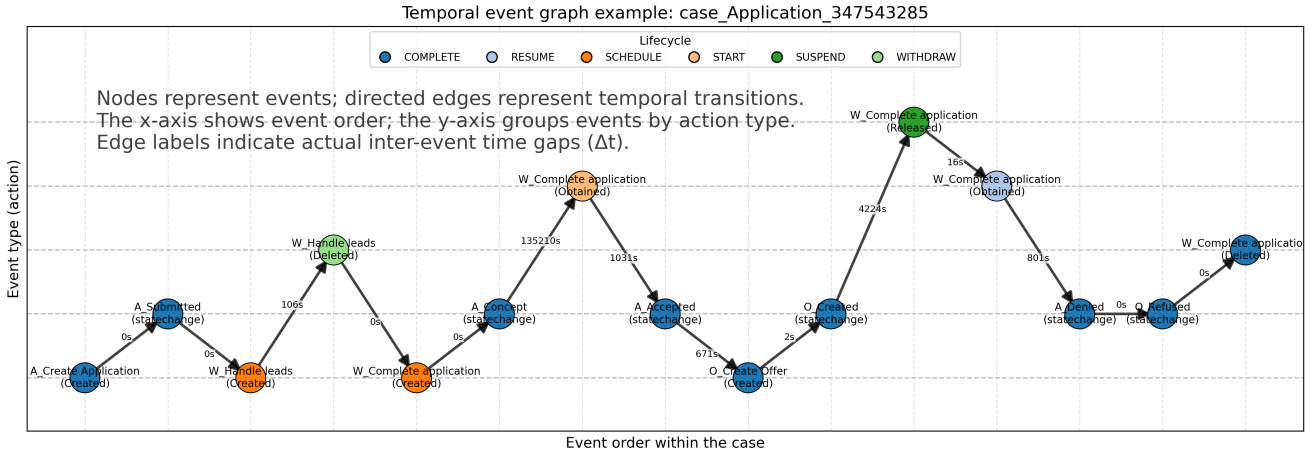


Figure 3: Illustrative example of an action-layered temporal case graph derived from a single application instance. Each node represents an event annotated with its activity and lifecycle state, while directed edges indicate temporal precedence; for readability only the directly-following edges are drawn, whereas the constructed graph additionally connects each event to its L most recent predecessors within the look-back window (Section 3, Step 4). Node colors encode lifecycle categories, and edge labels denote the actual inter-event time gaps (Δt). The x-axis reflects the sequential event order within the case, whereas the y-axis groups events by their action type, forming distinct horizontal layers for improved visual separation and interpretability.

in real-world business processes. Table 2 summarizes these differences. They are representational and task-level rather than cosmetic: Hennig and Schmidt build directly-follows temporal graphs in which nodes are distinct activities, so that the number of nodes is bounded by the activity set, whereas our instances are derived from OCEL and time is a first-class edge attribute that modulates attention; in TGN, time updates a latent memory state, whereas in our framework Δt directly modulates the attention weight between every pair of events; and the attention-based mechanism yields directly inspectable attention distributions, analyzed in Section 4, which a compressed memory state does not provide. Their study addresses a single task (remaining time) and explicitly notes that the effectiveness of their model for other tasks and domains remains open, and that adapting the graph or model architecture may be necessary for different predictions (Hennig et al., 2025). Our multi-task, OCEL-based formulation is therefore complementary rather than competing: it occupies exactly the design region their work leaves open, and our central empirical finding — that this joint objective helps temporal graph encoders while hurting sequential ones (Section 4) — is orthogonal to the single-task remaining-time question they study.

3.2.2. *Learnable Time Encoding*

The temporal encoding module plays a central role in enabling the attention mechanism to reason over irregular and highly variable event timestamps. Given a vector of time differences Δt between the current event and its context window, the module maps raw time intervals into a periodic embedding space using a combination of logarithmic scaling and learnable sinusoidal projections.

First, raw time differences are transformed as:

$$z = \log(1 + \Delta t) \tag{5}$$

which normalizes large temporal gaps and stabilizes training. Next, TGAT applies learnable temporal projections:

$$\mathbf{TE}(\Delta t) = \Big[\sin(z \cdot \omega + \phi),\ \cos(z \cdot \omega + \phi)\Big] \tag{6}$$

where ω and ϕ are trainable frequency and phase parameters.

This results in a temporal representation that captures both short-term and long-term periodic patterns. By concatenating $\mathbf{TE}(\Delta t)$ with node features before attention computation, TGAT learns to emphasize temporally relevant events while down-weighting obsolete or temporally distant information.

The significance of this module is further examined in our ablation study (subsection 4.4), where removing the time encoding results in a noticeable degradation in predictive performance in all tasks, confirming its essential role in modeling temporal dependencies in business processes.

3.2.3. *Multi-Task Prediction Heads*

The shared TGAT encoder outputs a latent representation $\mathbf{h}_t$ for each event. This embedding is passed through a fully connected neural network (MLP), which produces a unified feature representation used by three task-specific heads:

- **Next Activity Classifier:** Outputs a probability distribution over the activity vocabulary using a softmax layer.
- **Next Inter-Event Time Regressor:** Predicts the time interval to the next event using a regression layer optimized with MAE loss.
- **Remaining Time Regressor:** Estimates the remaining time until process completion, defined as

$$RT_t = t_{\text{end}} - t_t \tag{7}$$

Table 2
Positioning of the proposed framework relative to Hennig and Schmidt (Hennig et al., 2025).

Axis	Hennig and Schmidt (2025)	This work
Input log / instances	Case-centric event log with predefined case identifiers	OCEL; leading-object-type extraction with event replication under convergence (Section 3, Step 1)
Graph construction	Directly-follows temporal graph; nodes are distinct activities (node count bounded by the activity set)	Temporal graph over event occurrences with Δt-annotated edges
Temporal mechanism	TGN: recurrent memory module updated via message passing	TGAT: continuous-time multi-head self-attention with learnable log-scaled time encoding
Role of time	Updates a latent memory state	Δt directly modulates every pairwise attention weight
Learning objective	Single-task remaining-time prediction	Joint next activity, next inter-event time, and remaining time on a shared encoder
Interpretability	Memory state not directly interpretable	Attention weights inspectable; head specialization analyzed (Section 4)

The overall training objective is the unweighted sum of the three task losses:

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \mathcal{L}_{\Delta t} + \mathcal{L}_{\text{RT}}. \tag{8}$$

We deliberately refrain from task-specific weighting or loss balancing: no treatment of objective imbalance is applied, and the objective is the standard unweighted multi-task loss (equivalently, the uniform special case $\lambda_{\text{act}} = \lambda_{\Delta t} = \lambda_{\text{RT}} = 1$ of the general weighted formulation $\mathcal{L} = \lambda_{\text{act}}\mathcal{L}_{\text{act}} + \lambda_{\Delta t}\mathcal{L}_{\Delta t} + \lambda_{\text{RT}}\mathcal{L}_{\text{RT}}$). This choice isolates the contribution of the shared temporal graph encoder in the multi-task-versus-single-task comparison of Section 4 from any effect of loss balancing. Adaptive task-weighting schemes, such as uncertainty-based weighting or gradient normalization, are orthogonal to the contribution of this work and are left for future work (Section 5).

3.3. Illustrative Example of Temporal Graph Construction and Prediction

To elucidate the transition from raw object-centric data to multi-task inference, we provide a conceptual overview in Figure 4. This workflow demonstrates how the structural richness of Object-Centric Event Logs (OCEL) is synthesized into a temporally aware graph representation.

Construction of temporal graphs. Figure 4a depicts a simplified OCEL segment, where events are intrinsically linked to multiple business entities (e.g., orders and items). Unlike traditional flattening, which may obscure relational context, our approach derives temporally ordered prediction instances by sorting events chronologically within a specific object's lifecycle. For any prefix ending at time t, the model considers only historically observable events to ensure a realistic monitoring scenario.

Based on this prefix, we construct a temporal graph in which each node corresponds to an event and each event is connected to its L most recent predecessors within the look-back window, following the construction of Section 3 (Step 4). Each directed edge (e_j, e_i) is annotated with the elapsed time $\Delta t_{ij} = t(e_i) - t(e_j)$; the directly-follows pair is the special case $j = i - 1$. This representation preserves the original execution order while explicitly encoding continuous temporal information, enabling the model to capture irregular waiting times and temporal dependencies common in real-world processes.

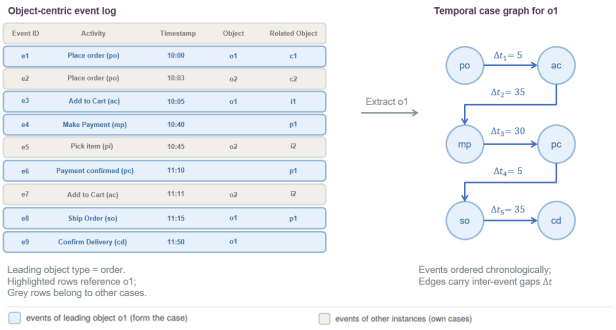
Figure 4b illustrates the operation of a single TGAT layer for the running example. The temporal neighborhood $N(v)$ of the target event comprises the L most recent past events of the prefix within the look-back window — not only its directly-preceding event. For each context event $u \in N(v)$, the key and value are formed by concatenating the event embedding with the continuous time encoding $\text{TE}(\Delta t_{vu})$ of its temporal distance to the target; multi-head attention yields weights α_{vu} over all context events in the window, and the weighted messages are aggregated into the updated representation of the target. The attention mechanism is therefore meaningful precisely because the neighborhood contains multiple past events with heterogeneous time gaps. In the figure, context events are arranged chronologically, and each edge is annotated with its time gap Δt and learned attention weight α (edge thickness proportional to α); the aggregated representation is then consumed by the three task-specific heads.

Multi-Task Inference Pipeline. The processing of this graph is illustrated in Figure 4b. The temporal graph is ingested by the **TGAT encoder**, which employs time-aware attention to aggregate neighborhood information. This mechanism generates a latent representation that encapsulates both the topological structure of the process and its temporal evolution.

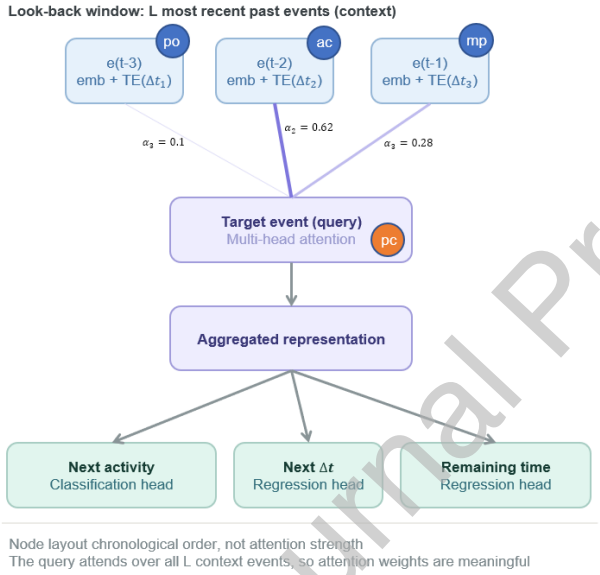
To exploit the inherent correlations between different process metrics, we apply three specialized prediction heads to this shared representation:

- **Next Activity Classification:** A softmax-based head to predict the most likely succeeding event type.
- **Next Inter-event Time Regression:** An estimator for the duration until the following activity occurs.
- **Remaining Time Regression:** A global estimator for the total time until process completion.

Layer-wise Temporal Attention. Figure 4b illustrates how the constructed temporal graph is processed by the proposed model. The temporal graph is fed into a Temporal Graph Attention Network (TGAT) encoder, which



(a) Case generation and temporal graph construction. The leading object instance (order o_1) is highlighted; only the events referencing o_1 are extracted and ordered chronologically to form the case sequence, while events of other object instances (greyed out) belong to their own cases. One temporal graph is derived per object instance of the leading type, with Δt-annotated edges connecting each event to its L most recent predecessors within the look-back window; only the directly-following edges are drawn for readability.



(b) Message passing and aggregation in one TGAT layer. The target event attends over its full temporal neighborhood — the L most recent events of the prefix — with keys and values formed from event embeddings concatenated with continuous time encodings TE(Δt). Edges are annotated with Δt and the learned attention weight α; edge thickness is proportional to α. The aggregated representation feeds the three task-specific prediction heads.

Figure 4: Illustrative example of temporal graph construction and prediction from object-centric event logs.

aggregates information from past events using time-aware attention mechanisms. The resulting hidden representation summarizes both the structural ordering of events and their temporal context. On top of this shared representation, three task-specific prediction heads are applied: (i) a classifier for next activity prediction, (ii) a regressor for next inter-event time estimation, and (iii) a regressor for remaining time prediction.

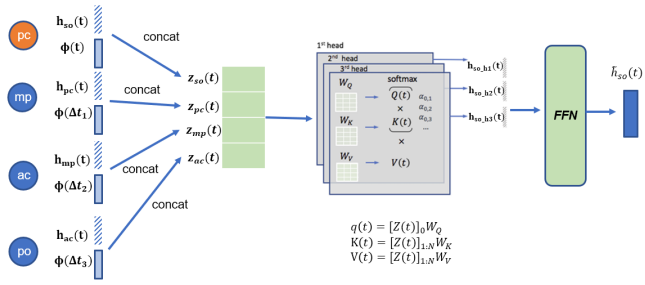


Figure 5: Illustration of the TGAT layer with time-aware attention. Node representations are combined with continuous-time encodings and aggregated via multi-head temporal attention.

The internal computation of a TGAT layer is further detailed in Figure 5. For each target event, the TGAT layer combines event embeddings with continuous-time encodings via concatenation and applies multi-head attention to selectively weight past events. This design enables the model to assign higher importance to temporally relevant events while down-weighting less informative ones, thereby effectively modeling long-range dependencies and irregular temporal dynamics.

Through this example, we highlight how temporal graph construction and time-aware attention jointly enable expressive modeling of event sequences derived from object-centric logs. This illustrative workflow also clarifies the role of temporal information in supporting accurate and robust multi-task predictive process monitoring.

4. Experiments

This section presents an extensive experimental evaluation of the proposed multi-task Temporal GNN framework. The experiments are designed to assess three main aspects: (i) the overall predictive performance of the model on multiple object-centric process datasets, (ii) the effectiveness of temporal graph encoding compared to sequence-based and time-agnostic baselines, (iii) the benefits of multi-task learning for jointly predicting control-flow and time-related targets.

We conducted experiments on four public benchmark datasets from the Business Process Intelligence (BPI) Challenges, converted into Object-Centric Event Logs (OCEL). The evaluation follows a prefix-based predictive monitoring protocol and considers three prediction tasks: next activity classification, next inter-event time estimation, and remaining processing time prediction. To ensure a fair comparison, all baseline models are trained and evaluated in the same data splits and experimental settings.

4.1. Data

We evaluate the proposed multi-task Temporal GNN framework on Object-Centric Event Log (OCEL) versions

of four widely used Business Process Intelligence (BPI) Challenge datasets: BPI 2014, 2016, 2017, and 2019. These datasets are public benchmarks that have been repeatedly adopted in predictive process monitoring research due to their realistic process variability, noise, and complex temporal dynamics.

Although the original BPI Challenge datasets are inherently case-centric and defined with explicit case identifiers, our study operates on their Object-Centric Event Log (OCEL) representations. These OCEL datasets are obtained through established transformation approaches that convert traditional event logs or event knowledge graphs into object-centric formats, enabling events to be associated with multiple objects instead of a single predefined case notion (Khayatbashi, Hartig and Jalali, 2023). As a result, unlike classical event logs, the OCEL representation used in this work does not rely on a single explicit case identifier. Instead, it captures multi-object interactions by allowing each event to be linked to multiple object instances, thereby providing a richer and more flexible representation of real-world processes.

- **BPI Challenge 2014 (Rabobank ICT / ITIL service management):** This dataset originates from the ICT department of Rabobank Group and contains records extracted from an ITIL-based service management tool. It is provided as multiple CSV files with identifiers such as interaction, incident, and change numbers, along with an associated activity log4TU.ResearchData (2014).
- **BPI Challenge 2016 (UWV customer journey / multi-channel interactions):** Collected by UWV (Dutch Employee Insurance Agency), this dataset captures customer contacts over an eight-month period across multiple channels, including website clickstreams, messages, call center interactions, and complaints. The presence of heterogeneous event sources makes this dataset particularly suitable for temporal predictive tasks4TU.ResearchData (2016).
- **BPI Challenge 2017 (loan application process):** This dataset describes the online loan application process at a Dutch financial institution, in which each application may involve multiple offers and repeated interactions. Such characteristics reflect realistic branching behavior and complex object interactions4TU.ResearchData (2017).
- **BPI Challenge 2019 (purchase order handling / multi-item purchase documents):** Originating from a multinational coatings company, this dataset records the handling of purchase orders across subsidiaries. Each purchase document may contain multiple line items, and the log includes complex variants such as two-way and three-way matching as well as consignment flows. The dataset is large-scale, comprising hundreds of thousands of cases and over one million events4TU.ResearchData (2019).

Table 3
Datasets.

Datasets	Event	Object	Activity
BPIC 2014	690622	228885	330
BPIC 2016	7360146	748913	620
BPIC 2017	1202267	106162	66
BPIC 2019	1595923	330685	42

To support object-centric predictive monitoring, all datasets are represented in the OCEL format, which allows events to be associated with multiple object instances of potentially different object types. This representation preserves the concurrent evolution of multiple business entities (e.g., applications, offers, orders, items) and enables richer modeling of real-world processes than traditional case-centric logs.

In practice, predictive tasks such as next activity, next inter-event time, and remaining time prediction require well-defined running instances. Following common practice in object-centric predictive analytics, we therefore generate prediction instances by selecting an appropriate case notion or object perspective and extracting time-ordered event sequences for each instance. Information about specific data is shown in Table 3.

4.2. Experimental Setting

4.2.1. Prefix-based Evaluation Protocol

We adopt a prefix-based evaluation protocol: for each sequence of cases with events $(e_1, \dots, e_n)$, we create training examples from prefixes $(e_1, \dots, e_k)$, where $k < n$. Each prefix forms an input instance, and the labels are derived from the immediate next event and the case end time:

- **Next activity label**: a_{k+1}
- **Next inter-event time label**: $t(e_{k+1}) - t(e_k)$
- **Remaining time label**: $t(e_n) - t(e_k)$

To avoid information leakage, the split is performed at the case level (i.e., all prefixes of a case belong to the same split). We report results on the held-out test split and use a validation split for model selection and early stopping.

4.2.2. Temporal Graph Construction and Model Outputs

Following the construction procedure detailed in Section 3, each prefix is converted into a directed temporal graph in which each event node is connected to its L most recent predecessors within the look-back window, and each edge (e_j, e_i) carries the elapsed time $\Delta t_{ij} = t(e_i) - t(e_j)$ as an attribute (Section 3, Step 4). The only node feature is the refined atomic activity label produced by the Semantic Event Refinement step (Step 2 of Section 3), which concatenates the original Activity, Action, and Lifecycle attributes; no other event attributes (e.g., resource, object-type indicators) are used as node features. The activity label enters the model through a learnable embedding layer of dimension 20 (cf.

Table 4), which is functionally equivalent to a one-hot encoding followed by a learnable linear projection. Temporal information is conveyed exclusively through edge attributes rather than node features, and this event-centric design deliberately isolates temporal dependency modeling within object-centric sequences.

The TGAT encoder produces a latent representation that is forwarded to a shared multi-layer perceptron (MLP), followed by three task-specific heads for next activity, next timestamp, and remaining time prediction. We use an MLP rather than a recurrent module such as LSTM here because temporal aggregation is already performed by the TGAT encoder: the MLP never sees the raw event sequence and serves only as a lightweight task-specific head on top of the already temporally-aggregated representation. Placing an LSTM at this stage would duplicate the temporal modeling that the encoder provides and shift part of the temporal capacity into the head, which would obscure the contribution of the encoder in cross-model comparisons. We therefore follow the standard "expressive encoder + simple head" pattern from graph representation learning. Combined with hard parameter sharing across the three heads, this design lets the model learn a shared temporal representation while maintaining task-specific outputs.

4.2.3. Baseline

To comprehensively evaluate the effectiveness of the proposed approach, we compare it with a diverse set of baseline models that span the major modeling paradigms used in predictive process monitoring: (i) non-sequential aggregation (MLP), (ii) recurrent sequence modeling (LSTM, GRU), (iii) Transformer-based self-attention (ProcessTransformer), (iv) traditional process-aware modeling (DFG-GNN), (v) object-centric graph embedding (GOCEL), (vi) heterogeneous graph modeling (HOEG), (vii) static or weakly-temporal graph neural networks (GGCN), and (viii) continuous-time temporal graph networks (TGN) and (ix) graph-transformer-based predictive monitoring with explicit temporal features (PGTNet). Each baseline is included as a representative of its paradigm , tuned per dataset under the protocol of Section 4.2.5, so that the comparison isolates the contribution of object-centric temporal graph attention rather than confounding it with model-family effects. First, we include a simple multilayer perceptron (MLP) as a non-sequential baseline that relies solely on aggregated prefix features, without explicitly modeling temporal dependencies. This baseline quantifies the benefit of sequence-aware and graph-based representations.

Second, we consider sequence-based deep learning models, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks. These models have been widely adopted in predictive process monitoring and represent the standard approach to modeling event sequences in temporal order. Both models are trained on the same prefix-based instances and input features as our proposed method to ensure a fair comparison.

In addition, we include DFG-GNN (Lischka et al., 2025), a Directly-Follows Graph (DFG)–based predictive approach combined with graph neural networks, as a representative traditional process mining baseline. Unlike purely deep sequential or other graph neural models, DFG-GNN explicitly models process flow dependencies through transition relations between activities, providing an interpretable process-centric representation commonly used in conventional predictive process monitoring. Including this baseline enables comparison between modern deep learning approaches and traditional process-aware predictive modeling strategies.

To further evaluate recent sequence-based predictive monitoring approaches, we additionally include the ProcessTransformer (Bukhsh et al., 2021), a Transformer-based architecture that leverages self-attention to capture long-range dependencies in event sequences. ProcessTransformer represents a strong attention-based sequential baseline and enables comparison between sequence-based self-attention and graph-based temporal attention mechanisms.

We further include PGTNet (Amiri Elyasi et al., 2024), a process graph transformer network originally proposed for remaining-time prediction, as a representative of graph-transformer-based predictive monitoring with explicit temporal features. PGTNet encodes each prefix as a compact directly-follows graph over event classes, with edge features carrying occurrence counts and temporal information, processed by GPS layers that combine local message passing with global attention. Following the uniform adaptation strategy described in Section 4.2.5, its graph representation and encoder are left intact; the output stage is extended with three task-specific heads applied to the graph-level representation, augmented with the representation of the current event class so that the next-activity and next-timestamp heads can condition on the current process state. PROPHET (Pasquadibisceglie et al., 2024) and the instance-graph approach of Chiorrini et al. (Chiorrini et al., 2023) target next-activity prediction and depend, respectively, on rich multi-view event attributes (resources, data attributes) and on a discovered process model for constructing instance graphs; since our unified protocol restricts all models to the activity label and Δt for cross-model fairness and does not assume a discovered model, reproducing these methods without those ingredients would misrepresent their performance. The heterogeneous-graph and instance-graph paradigms they represent are covered in the comparison by HOEG and DFG-GNN, respectively, and we state this scoping decision explicitly.

Third, we include graph-based predictive models to assess the advantage of explicit relational and structural modeling. In particular, we compare with the Graph Gated Convolutional Network (GGCN)(Ke et al., 2025), which has demonstrated strong performance in recent object-centric predictive process monitoring studies, especially in multi-task learning settings. GGCN is therefore included as a competitive state-of-the-art baseline for multi-task prediction on object-centric event logs.

We additionally compare with GOCEL (Adams et al., 2023), the graph-based OCEL encoding framework proposed by Adams et al., which preserves object-centric relational structures through graph representations. This baseline serves as a representative object-centric graph embedding approach for predictive process monitoring.

Furthermore, we include the Heterogeneous Object Event Graph (HOEG) approach (Smit et al., 2024), which models OCEL data as heterogeneous object-event interaction graphs. HOEG has recently demonstrated strong performance for remaining time prediction and therefore provides an important reference for evaluating heterogeneous graph modeling strategies in object-centric predictive monitoring.

We also compare with the Temporal Graph Network (TGN) (Hennig et al., 2025), a representative temporal graph neural network that models dynamic graphs through a memory module combined with continuous-time message passing. Hennig and Schmidt introduce this model to predictive process monitoring for single-task remaining-time prediction, building directly-follows temporal graphs whose nodes are distinct activities; it is the closest temporal-graph reference to our approach and therefore an important comparison point. To evaluate it within our unified multi-task protocol, we retain its memory-based temporal message passing and, following the uniform adaptation strategy of Section 4.2.5, attach the three task-specific heads to the pooled graph representation. We note that this baseline is temporal-graph-based but, like its original formulation, operates on the case-level directly-follows construction rather than on object-centric interaction graphs.

To distinguish object-centric predictive monitoring approaches from conventional predictive process monitoring methods, we categorize the baselines into two groups: non-object-centric PPM approaches (MLP, LSTM, GRU, DFG-GNN, ProcessTransformer, PGTNet, and TGN) and object-centric PPM approaches (GOCEL, GGCN, HOEG, and the proposed TGAT). We place TGN (Hennig et al., 2025) in the non-object-centric group because, although it is temporal-graph-based, its directly-follows construction operates on case-centric logs rather than on object-centric interaction graphs; the proposed model is thus the only approach that is simultaneously object-centric and temporal-graph-based. This categorization enables a more systematic evaluation of the impact of object-centric relational modeling and temporal graph representations.

All baseline models are trained using the same data splits, input features, and prefix-based evaluation protocol to ensure fair and consistent comparison among methods. All results in the main text are reported under the temporal case-level split, in which training and testing cases are chronologically separated to evaluate future generalization under realistic predictive monitoring conditions; random-split results are provided in Appendix A solely for comparability with prior work.

4.2.4. Evaluation Metrics

The proposed framework addresses predictive process monitoring as a multi-task learning problem, comprising one classification task and two regression tasks. To adequately evaluate the performance of each task, we use task-specific evaluation metrics commonly used in the predictive process monitoring literature.

Next Activity Prediction The next activity prediction task is formulated as a multi-class classification problem, where the objective is to predict the activity type of the next event given a prefix event graph. We evaluate this task using classification accuracy (ACC), which measures the proportion of correctly predicted activity labels among all test samples. Formally, the accuracy is defined as:

$$\mathrm{ACC} = \frac{1}{B}\sum_{i=1}^{B} \mathbb{I}\left(\hat{y}_{\mathrm{act},i} = y_{\mathrm{act},i}\right), \tag{9}$$

where B denotes the total number of test samples, $\hat{y}_{\mathrm{act},i}$ is the predicted activity label for the i-th sample, and $y_{\mathrm{act},i}$ is the corresponding ground-truth label. The indicator function $\mathbb{I}(\cdot)$ returns 1 if the prediction is correct and 0 otherwise.

In addition to standard accuracy, we report Top-3 accuracy to account for the inherent uncertainty and ambiguity in process execution. Top-3 accuracy measures whether the ground-truth next activity appears among the three most probable predictions output by the model. This metric provides a more flexible assessment of predictive performance in complex process environments where multiple next activities may be plausible.

In addition to accuracy, we report the F1-score to account for potential class imbalance in event logs and to provide a more comprehensive evaluation of classification performance. The F1-score is defined as the harmonic mean of precision and recall:

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}, \tag{10}$$

where precision and recall are computed based on the predicted and ground-truth activity labels. For the multi-class setting, we report the macro-averaged F1-score across all activity classes.

Time-related Regression Tasks The remaining two tasks predict continuous-time quantities: the inter-event time until the next activity and the remaining processing time of the case. The inter-event time prediction task estimates the time interval between the current event and the subsequent event, whereas the remaining time prediction task estimates the time until the process instance completes.

For both regression tasks, we use the mean absolute error (MAE) as the evaluation metric, as it provides an interpretable measure of the average prediction error in the original time scale. The MAE for inter-event time prediction is defined as:

$$\mathrm{MAE}_{\mathrm{next}} = \frac{1}{B}\sum_{i=1}^{B} \left|\hat{y}_{\mathrm{next},i} - y_{\mathrm{next},i}\right|, \tag{11}$$

and the MAE for the remaining time prediction is defined as:

$$\mathrm{MAE}_{\mathrm{remain}} = \frac{1}{B}\sum_{i=1}^{B}\left|\hat{y}_{\mathrm{remain},i} - y_{\mathrm{remain},i}\right|, \tag{12}$$

where $\hat{y}_{\mathrm{next},i}$ and $\hat{y}_{\mathrm{remain},i}$ denote the predicted next inter-event time and remaining time for the i-th sample, respectively, and $y_{\mathrm{next},i}$ and $y_{\mathrm{remain},i}$ denote the corresponding ground-truth values. All metrics are computed on the test set and reported separately for each dataset.

4.2.5. Experimental environment

All experiments are conducted in a Linux-based server environment using Visual Studio Code for development and execution. The models are implemented in Python (version 3.11) using PyTorch (version 2.5.1) and the corresponding graph learning library (e.g., PyTorch Geometric version 2.4.0). The experimental server is equipped with an NVIDIA TITAN RTX GPU with 24GB memory, running CUDA version 12.1 and driver version 530.41.03.

Since predictive process monitoring inherently involves forecasting future process behavior, we adopt a temporal case-level split as the evaluation setting of this paper: cases are ordered chronologically by their start time, earlier cases are used for training, and the most recent cases are reserved for testing, with an intermediate held-out validation set used for model selection and early stopping (80%/10%/10% at the case level, so that every prefix of a given case belongs to the same split and no information leaks between training and evaluation). This setting simulates real-world deployment, where models are trained on historical data and applied to future executions. This is also in line with recent PPM practice: both PROPHET (Pasquadibisceglie et al., 2024) and the instance-graph approach of Chiorrini et al. (Chiorrini et al., 2023) evaluate on chronologically ordered 2/3–1/3 splits. For comparability with prior work that reports random case-level splits under a no-concept-drift assumption (Galanti, De Leoni, Navarin and Marazzi, 2023; Ke et al., 2025), the corresponding random-split results are reported in Appendix A; no conclusion in the main text relies on them.

We train all models using the Adam optimizer for up to 50 training epochs. Early stopping is applied based on validation performance to prevent overfitting, and training is terminated if the validation metric does not improve for 10 consecutive epochs. To ensure fairness and reproducibility, all baselines — both the non-object-centric models (MLP, LSTM, GRU, ProcessTransformer, DFG-GNN, PGTNet, TGN) and the object-centric graph-based models (GOCEL, HOEG, GGCN, and the proposed TGAT) — are trained and evaluated under a unified experimental protocol, including identical case-level data splits and the same prefix-based instance generation procedure. All models receive the same underlying event information, including one-hot encoded activity labels and temporal information derived from inter-event time gaps Δt, while the incorporation of temporal information follows the design of each architecture. Shared optimization settings — optimizer, learning rate, batch size, dropout rate, maximum epochs, and early-stopping patience (Table 4) — are kept identical across all models, so that no model benefits from a larger training budget than another. Architecture hyperparameters, in contrast, are tuned independently for every model on every dataset. For each model we perform a grid search over the search space in Table 5 (hidden/embedding dimension, number of layers, number of attention heads, memory dimension, and look-back window, as applicable to the architecture), evaluate all applicable combinations on the validation split, and select the configuration with the best validation performance; no configuration is transferred across datasets or across models. The selected configurations are reported in Table 6 and differ substantially across datasets, confirming that no single configuration is optimal for all logs. All reported test results are means over 5 random seeds with standard deviations, and the difference between the proposed model and the strongest baseline in each dataset–metric cell is assessed with a one-sided Wilcoxon signed-rank test across seeds ($p < 0.05$); the full mean±std results with significance markers are given in Appendix B.

Table 4
Shared training and optimization settings used across all models whenever applicable.

Hyperparameter	Value
Input dimension	20
Learning Rate	0.001
Batch Size	256
Dropout Rate	0.2
Optimizer	Adam
Maximum Epochs	50
Early Stopping Patience	10

For baselines that were not originally proposed for multi-task object-centric prediction, we adopt a uniform adaptation strategy: the original input pipeline and encoder are left intact, and the output stage is extended with task-specific heads so that the encoder's final hidden state feeds three parallel heads (next-activity classification, next-timestamp regression, remaining-time regression). This applies to both sequence-based encoders and Transformer-based encoders. For example, LSTM and GRU, which are commonly used as single-task next-activity predictors, are trained jointly on all three targets by attaching a softmax head and two regression heads to their final hidden state; ProcessTransformer (Bukhsh et al., 2021), originally designed for remaining-time prediction only, is augmented with an additional next-activity classification head and a next-timestamp regression head on top of its existing self-attention encoder; DFG-GNN, originally proposed for case-centric prediction, is applied to the same prefix-based OCEL instances used by the other baselines so that the comparison is restricted to the encoder rather than the data interface. Throughout this procedure no model is given access to information that the others do not see, and no encoder is modified internally.

Table 5
Hyperparameter search space per model family. For each model and each dataset, all applicable combinations are evaluated on the validation split, and the best-performing configuration is selected independently.

Hyperparameter	Search space	Applies to
Hidden / embedding dimension	{128, 256, 512}	all models
Number of layers	{2, 3, 4}	all except MLP
Number of attention heads	{1, 2, 3, 4}	ProcessTransformer, PGTNet, TGAT
Memory dimension	{256, 512}	TGN
Look-back window L	{5, 10, 20}	LSTM, GRU, ProcessTransformer, TGN, TGAT

Table 6
Configuration selected per model and per dataset by validation performance. Because the applicable hyperparameters differ across model families, the "Format" column states the fields reported for each model, in order; fields that do not apply to a model family are omitted (cf. Table 5). Here h denotes the hidden/embedding dimension, ℓ the number of layers, a the number of attention heads, L the look-back window, and m the memory dimension. Selected configurations differ across datasets, confirming that no single configuration is optimal for all logs.

Model	Format	BPI 2014	BPI 2016	BPI 2017	BPI 2019
MLP	h	128	256	512	512
LSTM	$h/\ell/L$	512/2/5	512/4/20	256/4/20	512/3/20
GRU	$h/\ell/L$	256/2/5	512/2/5	256/2/10	128/4/10
ProcessTransformer	$h/\ell/a/L$	512/3/3/10	256/3/2/10	512/4/4/5	512/3/1/20
DFG-GNN	h/ℓ	512/3	256/3	128/4	512/4
PGTNet	$h/\ell/a$	512/3/1	512/3/2	256/3/1	256/4/4
TGN	$h/\ell/m/L$	256/2/256/20	512/3/512/20	512/3/512/20	256/4/512/10
GOCEL	h/ℓ	256/2	512/4	256/4	256/4
HOEG	h/ℓ	512/4	128/4	128/2	128/4
GGCN	h/ℓ	512/4	128/4	512/3	256/2
TGAT (ours)	$h/\ell/a/L$	256/3/2/10	128/2/1/20	512/3/3/20	256/2/2/20

4.3. Experiments Results

4.3.1. Overall performance comparison

Table 7 reports the overall predictive performance of all models across the four BPI Challenge datasets under the temporal case-level split, in which training and testing cases are chronologically separated to evaluate future generalization capability under realistic predictive monitoring conditions. All values are means over 5 random seeds under per-dataset tuned configurations; random-split results are reported in Appendix A for comparability with prior work only. The results demonstrate several consistent patterns.

First, sequence-based deep learning models (LSTM and GRU) substantially outperform the non-sequential MLP baseline in next activity prediction accuracy across all datasets, confirming the importance of modeling temporal order in predictive process monitoring. In particular, LSTM attains the best accuracy and macro F1 on BPI 2016 and remains the strongest sequential competitor on BPI 2017, indicating that sequential dependency modeling remains highly effective for activity prediction tasks. However, their advantage does not carry over to the time-related regression tasks: on BPI 2014, BPI 2017, and BPI 2019, the best MAE_{next} and MAE_{rem} values are obtained by graph-based rather than sequence-based encoders, especially on datasets with irregular inter-event times.

Second, among the object-centric graph-based baselines, GOCEL, HOEG, and GGCN consistently trail the sequential models on the classification metrics, indicating that relational structure alone — without an explicit representation of continuous time — is insufficient for next-activity prediction. The two temporal-graph baselines behave differently: TGN, which encodes time through a latent memory state, attains the best MAE_{rem} on BPI 2017 (0.2845) and is competitive on the BPI 2016 time metrics, whereas PGTNet, whose global attention is time-agnostic, is mid-ranked throughout. This contrast supports the view that how time enters the encoder — rather than the mere use of a graph — is what drives performance on the temporal tasks.

Third, the proposed TGAT attains the best result in 11 of the 16 dataset–metric cells and the second-best result in 4 further cells. It is the best model on every metric of BPI 2019 and on three of four metrics of BPI 2014 and BPI 2017, and it obtains the lowest MAE_{next} on all four datasets, which is the metric most directly tied to the modeling of inter-event intervals. The exceptions are informative rather than incidental. On BPI 2016, LSTM attains a higher accuracy

Table 7
Overall performance comparison across BPI Challenge datasets under the temporal case-level split, grouped by non-object-centric PPM and object-centric PPM approaches. ACC denotes accuracy and F1 denotes macro F1-score. All values are means over 5 random seeds under the per-dataset tuned configurations of Table 6; the corresponding standard deviations and Wilcoxon signed-rank significance markers are reported in Appendix B. The best results are highlighted in **bold**, while the second-best results are underlined. For MAE metrics, lower values indicate better performance.

Category	Model	BPI 2014				BPI 2016				BPI 2017				BPI 2019			
		ACC	F1	MAE_{next}	MAE_{rem}	ACC	F1	MAE_{next}	MAE_{rem}	ACC	F1	MAE_{next}	MAE_{rem}	ACC	F1	MAE_{next}	MAE_{rem}
Non-OCPM	MLP	0.2610	0.1440	0.4522	0.5448	0.6812	0.1118	0.0495	0.0279	0.5222	0.4400	0.4895	0.3471	0.4677	0.2245	0.0611	0.1344
	LSTM	0.3360	0.1630	0.4551	0.5371	**0.7030**	**0.1536**	0.0488	0.0288	0.8103	0.6947	0.4255	0.2967	0.7522	0.3415	0.0463	0.1285
	GRU	0.3444	0.1668	0.4550	0.5412	0.7012	0.1332	0.0512	0.0312	0.8022	0.6889	0.4289	0.3112	0.7420	0.3312	0.0489	0.1255
	ProcessTransformer (Bukhsh et al., 2021)	0.3422	**0.1725**	0.4951	0.5579	0.6978	0.1225	0.0499	**0.0275**	0.8035	0.6887	0.4289	0.3315	0.7533	0.3412	0.0608	0.1279
	DFG-GNN (Lischka et al., 2025)	0.3415	0.1512	0.4892	0.5677	0.6844	0.1022	0.0612	0.0399	0.7899	0.6788	0.4667	0.3389	0.7414	0.3518	0.0662	0.1571
	PGTNet (Amiri Elyasi et al., 2024)	0.3222	0.1655	0.4811	0.5555	0.6999	0.1016	0.0477	0.0298	0.8014	0.6618	0.4445	0.2985	0.7212	0.3612	0.0661	0.1611
	TGN (Hennig et al., 2025)	0.3040	0.0855	0.4898	0.5477	0.7011	0.1311	0.0466	0.0278	0.7988	0.6882	0.4255	**0.2845**	0.6488	0.1879	0.0566	0.1388
OCPM	GOCEL (Adams et al., 2023)	0.2612	0.1099	0.5688	0.6142	0.6970	0.0822	0.0577	0.0544	0.7108	0.5335	0.5118	0.3441	0.5022	0.1321	0.0615	0.1778
	HOEG (Smit et al., 2024)	0.2740	0.1113	0.5477	0.5978	0.6992	0.0514	0.0578	0.0489	0.7612	0.5998	0.4987	0.3311	0.6022	0.1745	0.0497	0.1477
	GGCN (Ke et al., 2025)	0.2725	0.0955	0.5144	0.5788	0.7009	0.0518	0.0499	0.0388	0.7446	0.4878	0.4588	0.3021	0.6244	0.1588	0.0575	0.1342
	TGAT (ours)	**0.3489**	0.1711	**0.4495**	**0.5330**	0.7022	0.1455	**0.0462**	0.0285	**0.8118**	**0.7042**	**0.4244**	0.2966	**0.7544**	**0.3644**	**0.0459**	**0.1224**

(0.7030 vs. 0.7022) and macro F1 (0.1536 vs. 0.1455), and ProcessTransformer attains the lowest MAE_{rem} (0.0275 vs. 0.0285); BPI 2016 is dominated by high-frequency website clickstream events with short and regular inter-event gaps (cf. the dataset description in Section 4.1 and the small absolute MAE values in Table 7), so that the temporal irregularity our encoder is designed to exploit is largely absent and a well-tuned sequential encoder suffices. On BPI 2014, ProcessTransformer attains a marginally higher macro F1 (0.1725 vs. 0.1711), and on BPI 2017, TGN attains a lower MAE_{rem} (0.2845 vs. 0.2966). We therefore do not claim uniform superiority on every cell; the claim supported by these results is that TGAT is the only model that is either best or second-best in 15 of 16 cells, i.e., the only model that performs strongly across all four datasets and all three tasks simultaneously, whereas each competing baseline is strong on some cells and markedly weak on others.

Finally, compared with strong Transformer-based baselines, TGAT demonstrates greater robustness on datasets with higher temporal variability. This suggests that explicitly encoding temporal differences within a graph-based attention mechanism is more effective than relying solely on positional or sequential encoding. A comparison with the random-split results in Appendix A further reinforces this observation, as TGAT exhibits smaller degradation from random to chronological evaluation than several sequence-based baselines, particularly for the temporal regression metrics. Overall, these results indicate that temporal graph modeling provides a meaningful inductive bias for predictive process monitoring, particularly when multiple temporally correlated tasks are learned jointly. To further understand why temporal graph modeling improves performance, the following subsection analyzes the learned temporal representations and attention patterns in TGAT.

4.3.2. Analysis of Temporal Graph Modeling

This subsection provides an in-depth analysis of why temporal graph modeling, and in particular the proposed TGAT-based approach, yields strong predictive performance in predictive process monitoring. We focus on the roles of continuous-time modeling and attention-based memory mechanisms, and on their interaction with multi-task learning.

Learning continuous temporal signals In real-world business processes, the occurrence and timing of the next activity are strongly influenced by elapsed time since previous events, such as recency and duration effects. Unlike discrete-time sequence models, TGAT directly models continuous temporal signals through explicit time encoding. Specifically, TGAT employs sinusoidal time-encoding functions to transform time differences into continuous-valued representations, allowing the model to learn fine-grained sensitivity to temporal gaps between events. This enables the encoder to distinguish short delays from long waiting periods and to adapt its attention weights accordingly, rather than treating time as an implicit byproduct of event order.

Selective attention versus cumulative memory Recurrent models such as LSTM and GRU compress the entire event history into a single hidden state, effectively accumulating past information regardless of its relevance to the current prediction. In contrast, TGAT relies on an attention-based mechanism that selectively focuses on temporally and contextually important past events. This selective memory mechanism allows TGAT to assign higher importance to critical historical events while attenuating irrelevant or outdated information. As a result, TGAT is particularly effective for long event sequences, where decisive events may be temporally distant yet remain influential on future behavior.

Modeling irregular temporal intervals Process events are inherently time-irregular, with non-uniform intervals between successive activities. Sequence-based models typically assume uniform temporal progression or rely on implicit representations of time, which can lead to information loss. TGAT explicitly incorporates time differences as edge attributes in the temporal graph, treating the elapsed time itself as a first-class feature. This design allows the model to preserve temporal irregularities without distortion, avoiding the information loss commonly observed in purely order-based models.

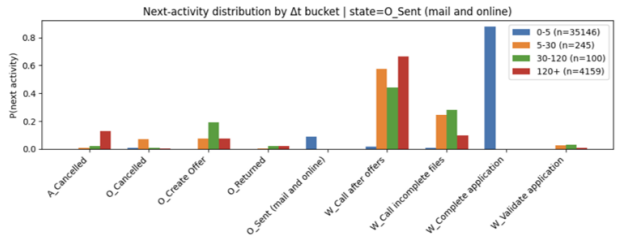


Figure 6: Next-activity distribution by Δt bucket | state = O_Sent(mail and online)

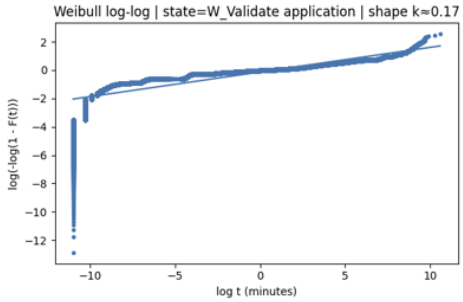


Figure 7: Weibull log-log plot | state=W_Validate application

State-dependent activity distributions across time buckets To further analyze the effect of temporal information, we examine how the distribution of next activities varies with elapsed time under a fixed process state. For a given activity state s_t, we divide the elapsed time Δt into four buckets: $[0, 5)$, $[5, 30)$, $[30, 120)$, and $[120, +\infty)$ minutes. Empirical analysis of the BPI Challenge 2017 dataset reveals that a substantial portion of activities (approximately 58% out of 26 activities) exhibit markedly different next-activity distributions in these time buckets.

For example, when Δt falls within the shortest interval $[0, 5)$, transitions to *W_Complete application* occur with high probability. In contrast, longer waiting times (e.g., $[30, 120)$ or 120+ minutes) substantially increase the likelihood of transitions to activities such as *W_Call incomplete files* or *W_Call after offers*. These observations confirm that the elapsed time fundamentally alters the process behavior, justifying the need for time-aware modeling. The results for this example are shown in Figure 6.

Weibull log–log analysis of waiting time distributions We further investigate temporal dynamics using Weibull log–log plots to assess whether waiting times until the next activity follow exponential or non-exponential distributions. If the waiting time were exponentially distributed, the probability of the next event would be memoryless, i.e.,

$$\Pr(\text{next} \mid s_t, \tau) = \Pr(\text{next} \mid s_t) \tag{13}$$

However, empirical results show significant deviations from the exponential behavior. In particular, for activities such as *W_Validate application*, the estimated Weibull shape parameter satisfies $k \ll 1$, indicating that the probability of observing the next event decreases as time elapses. This non-memory property implies that elapsed time τ plays a crucial role in determining future behavior, a role that time-agnostic models cannot capture. The graph for this example is shown in Figure 7.

Queueing and resource effects The elapsed time also implicitly reflects system-level factors such as queueing delays and resource availability. Longer waiting times often indicate resource contention, preparation delays, or the accumulation of pending projects, which, in turn, influence subsequent activity choices. By incorporating elapsed time directly into the temporal graph, TGAT can encode latent queueing effects without explicitly modeling resource states.

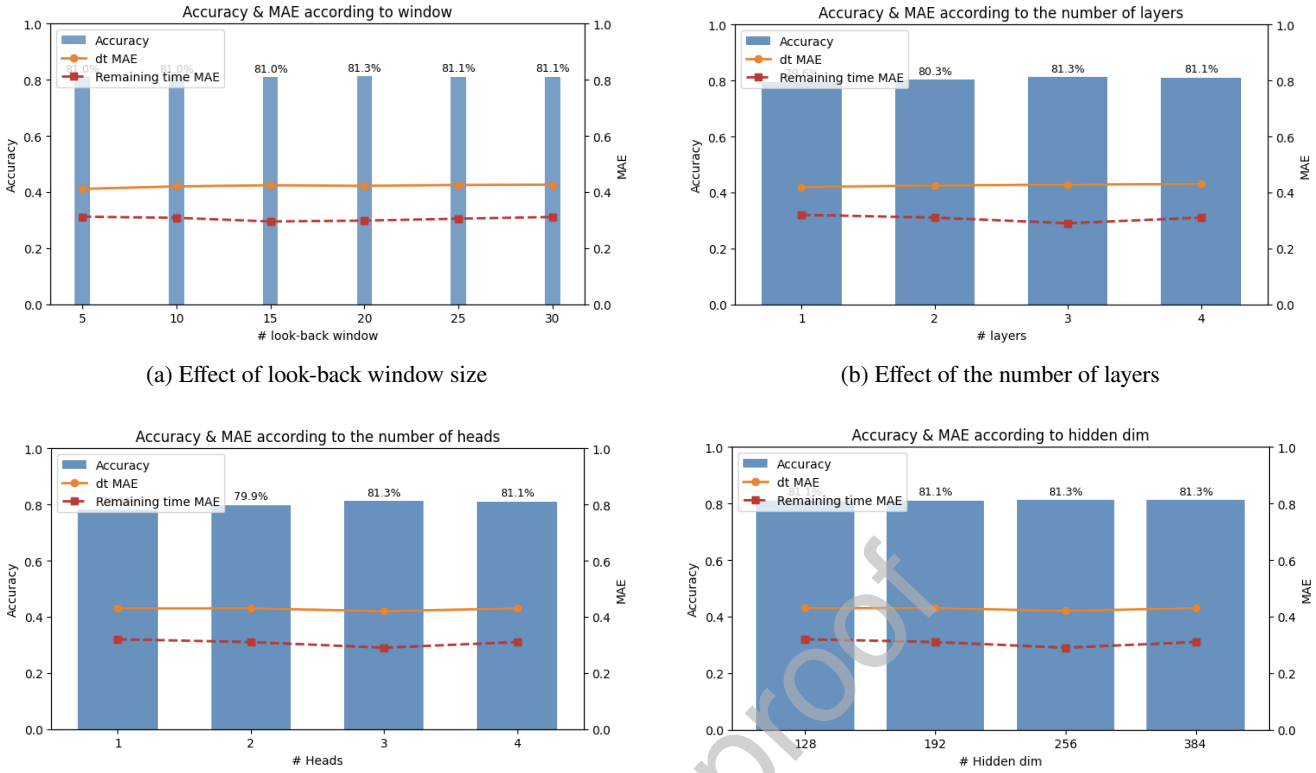
Overall, these analyses demonstrate that temporal graph modeling with attention-based time encoding provides a principled and empirically effective framework for predictive process monitoring, especially in object-centric and irregularly timed process environments.

4.3.3. *Hyperparameter sensitivity analysis*

To better understand the behavior and robustness of the proposed TGAT-based multi-task PPM model, we conduct a systematic hyperparameter sensitivity analysis focusing on four key components: (i) the look-back window size, (ii) the number of temporal GNN layers, (iii) the number of attention heads, and (iv) the hidden embedding dimension. These hyperparameters directly control the temporal receptive field, representation capacity, and the model's attention expressiveness, and are therefore critical for temporal process modeling. This analysis deliberately sweeps values beyond the search space of Table 5 (e.g., a window size of 30 and a single-layer variant) to characterize robustness outside the tuned range; unless stated otherwise, the results in this subsection are reported on BPI 2017.

Effect of the look-back window size The look-back window determines how many past events are included when constructing prefix-based temporal graphs. This parameter is particularly important in PPM, as different processes may depend on short-term reliability or a longer execution history. As illustrated in Figure 8a, performance remains relatively stable across window sizes ranging from 5 to 30 events, with a slight peak around a medium-sized window. This suggests that TGAT can effectively attend to the most relevant past events even when longer prefixes are provided, due to its attention mechanism. The robustness with respect to window size also indicates that the model does not rely on strict fixed-length histories, unlike traditional sequence models.

Effect of the number of layers We analyze the impact of stacking multiple TGAT layers, as deeper architectures allow the model to aggregate information from temporally more distant events. As shown in Figure 8b, increasing the



(a) Effect of look-back window size

(b) Effect of the number of layers

(c) Effect of the number of attention heads

(d) Effect of hidden embedding dimension

Figure 8: Hyperparameter sensitivity analysis of the proposed TGAT-based multi-task model. We report next-activity accuracy (bars) and MAE for next inter-event time and remaining time (lines).

number of layers from one to three consistently improves the accuracy of next-activity and reduces both next-time stamp MAE and remaining-time MAE. This indicates that multi-hop temporal aggregation helps capture longer-term dependencies in the process. However, adding a fourth layer yields no further improvements and slightly degrades performance, suggesting diminishing returns and potential over-smoothing. Based on this observation, three layers provide a good trade-off between expressiveness and stability.

Effect of the number of attention heads Multi-head attention enables the model to capture heterogeneous temporal patterns by attending to different aspects of the temporal neighborhood. We therefore vary the number of attention heads from one to four. As shown in Figure 8c, increasing the number of heads improves performance by up to three heads, after which the gains saturate. This confirms that multiple attention heads are beneficial for modeling diverse temporal signals, such as short delays versus long waiting times, while excessive heads may introduce redundancy without additional benefit.

Effect of the hidden embedding dimension Finally, we investigate the influence of the hidden embedding dimension, which controls the capacity of node representations. As shown in Figure 8d, increasing the hidden dimension from 128 to 256 leads to consistent improvements in both classification accuracy and regression errors. Larger dimensions beyond this range provide marginal gains, indicating that moderate embedding sizes are sufficient to encode temporal and contextual information in event sequences. This result suggests that the proposed model achieves strong performance without requiring excessively large representations.

Summary Overall, the sensitivity analysis demonstrates that the proposed TGAT-based multi-task framework is robust across a wide range of hyperparameter settings. The results also provide empirical guidance for selecting model configurations that balance temporal expressiveness, computational efficiency, and predictive accuracy.

4.3.4. Temporal attention visualization

To interpret how the proposed TGAT encoder exploits temporal dependencies, we visualize the learned temporal attention weights on a representative prediction instance from BPI Challenge 2017. Specifically, we consider a prefix whose current activity is `O_Created`, where the model correctly predicts the next activity `O_Sent (mail and online)`. We plot the attention distribution over historical events for each attention head, revealing that different heads attend to different parts of the execution history.

Figure 9, 10, 11 summarizes the attention patterns of three heads. **Head 0** assigns non-negligible weights to early-stage events in the prefix, suggesting that the model uses long-range historical context to stabilize the prediction. **Head 1** focuses sharply on a specific mid-prefix event (e.g.,

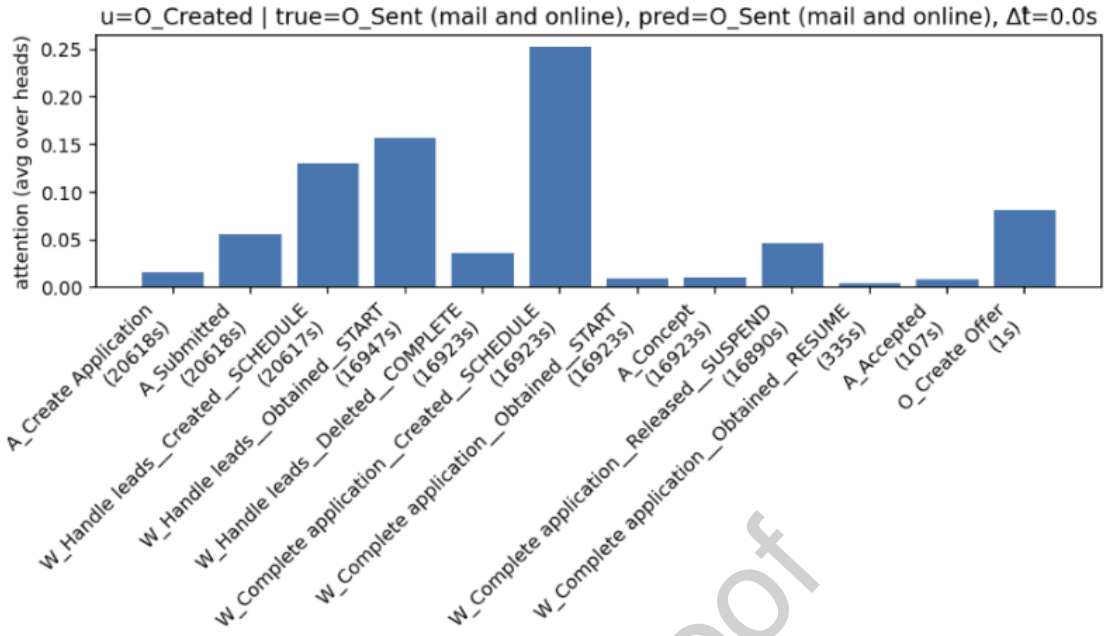


Figure 9: Head 0: global/early-history emphasis. Temporal attention visualization for a representative prediction example (current activity: `O_Created`, predicted next activity: `O_Sent (mail and online)`).

Table 8
Impact of time representation on predictive performance, reported on the BPI 2017 dataset under the temporal case-level split. Higher accuracy and lower MAE indicate better performance. The best results are highlighted in **bold**, while the second-best results are underlined.

Method	Accuracy	$\mathbf{MAE}_{\mathbf{next}}$	$\mathbf{MAE}_{\mathbf{rem}}$
MLP + feature (absolute time)	0.5944	0.4898	0.3471
MLP + feature (Δt)	0.6475	0.4772	0.3331
GNN + feature (absolute time)	0.3988	0.4755	0.3235
GNN + feature (Δt)	0.4224	0.4517	0.3112
GNN + feature (absolute time) + GRU	0.7855	0.4495	0.3002
GNN + feature (Δt) + GRU	0.8005	0.4422	0.2985
Temporal GNN	**0.8118**	**0.4244**	**0.2966**

`W_Handle leads`), indicating that this intermediate workflow step is highly informative for forecasting `O_Sent`. In contrast, **Head 2** places the largest weight on the most recent events, reflecting a recency-driven mechanism that captures short-term dynamics. Overall, these results provide qualitative evidence that multi-head temporal attention in TGAT acts as a mixture of temporal experts: some heads specialize in long-term context, while others capture mid-term triggers or short-term recency signals. This behavior aligns with the observed robustness of our temporal graph modeling across different look-back windows and supports the effectiveness of attention-based temporal aggregation for PPM.

4.3.5. Impact of time representation

In this experiment, we analyze the impact of different time representations on predictive performance, with a particular focus on distinguishing among *absolute time*, *time-difference* (Δt), and *temporal attention-based modeling*. We compare several architectural variants to isolate how temporal information is incorporated into the learning process.

Specifically, we consider three groups of models. First, we evaluate non-graph baselines (MLP) augmented with time features, using either absolute timestamps or inter-event time differences. Second, we examine static GNN encoders in which temporal information is treated as an additional node or edge feature but does not directly influence the neighborhood aggregation weights. Third, we compare these approaches with a full Temporal GNN model, where temporal information directly modulates the attention mechanism during message passing.

Table 8 summarizes the results. For MLP-based models, the incorporation of Δt consistently improves performance over absolute time, indicating that relative temporal information is more informative for PPM tasks than raw timestamps. A similar trend is observed for GNN models with time features, where Δt yields better results than absolute

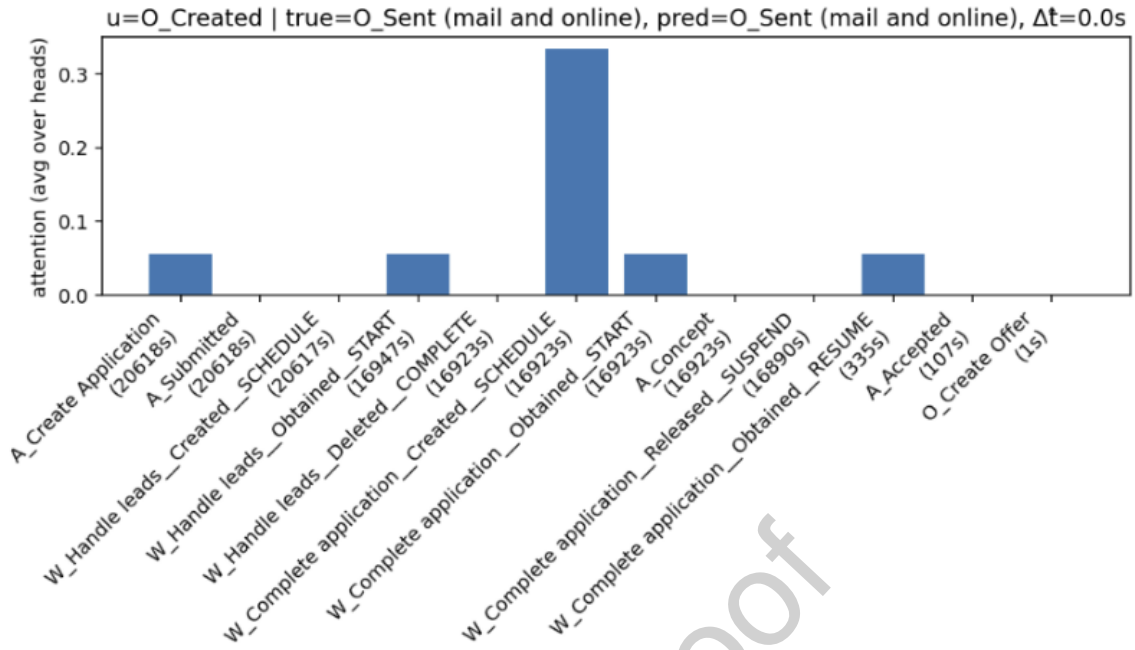


Figure 10: Head 1: mid-prefix event focus. Each subplot shows the attention weights over past events for one attention head.

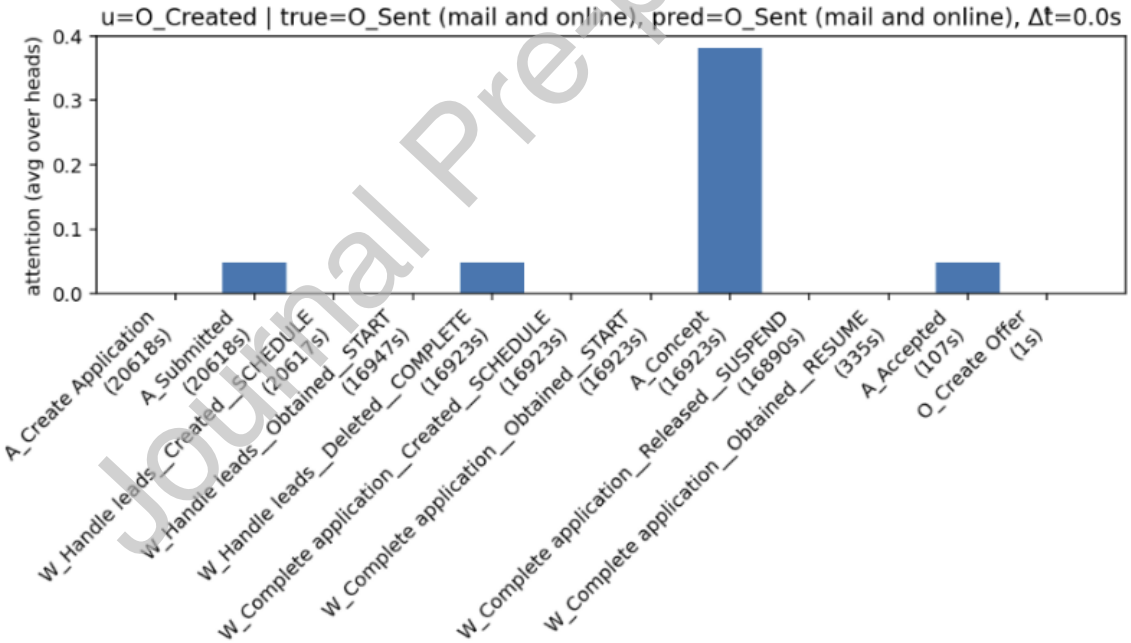


Figure 11: Head 2: recency emphasis. The results illustrate complementary temporal focus patterns across attention heads.

time, but overall performance remains limited due to time-agnostic aggregation.

Introducing a GRU on top of GNN features significantly improves performance, confirming the benefit of sequential modeling. However, even in this setting, representations based on Δt outperform absolute time, suggesting that relative temporal signals are easier to exploit in sequence models.

The Temporal GNN achieves the best performance across all evaluation metrics. Unlike feature-based approaches, the Temporal GNN directly integrates time differences into the attention weights, allowing the model to dynamically

Table 9
Comparison between multi-task and single-task learning across temporal graph and sequential models. Best results are highlighted in **bold**. Lower values indicate better performance for MAE.

Model	Dataset	Accuracy	MAE_{next}	MAE_{rem}
LSTM (Multi-task)	BPI 2014	0.3360	0.4551	0.5371
	BPI 2016	0.7030	0.0488	0.0288
	BPI 2017	0.8103	0.4255	0.2967
	BPI 2019	0.7522	0.0463	0.1285
LSTM (Single-task)	BPI 2014	0.3496	**0.4478**	**0.5251**
	BPI 2016	**0.7055**	0.0466	0.0290
	BPI 2017	0.8112	0.4233	0.2967
	BPI 2019	0.7535	0.0460	0.1249
ProcessTransformer (Multi-task)	BPI 2014	0.3422	0.4951	0.5579
	BPI 2016	0.6978	0.0499	0.0275
	BPI 2017	0.8035	0.4289	0.3315
	BPI 2019	0.7533	0.0608	0.1279
ProcessTransformer (Single-task)	BPI 2014	0.4195	0.4545	0.5324
	BPI 2016	0.7047	0.0499	**0.0266**
	BPI 2017	0.8101	**0.4227**	0.2903
	BPI 2019	0.7540	0.0559	0.1247
TGN (Multi-task)	BPI 2014	0.3040	0.4898	0.5477
	BPI 2016	0.7011	0.0466	0.0278
	BPI 2017	0.7988	0.4255	**0.2845**
	BPI 2019	0.6488	0.0566	0.1388
TGN (Single-task)	BPI 2014	0.3012	0.4999	0.5478
	BPI 2016	0.6933	0.0477	0.0295
	BPI 2017	0.7891	0.4358	0.3217
	BPI 2019	0.5818	0.0588	0.1396
TGAT (Multi-task)	BPI 2014	0.3489	0.4495	0.5330
	BPI 2016	0.7022	**0.0462**	0.0285
	BPI 2017	**0.8118**	0.4244	0.2966
	BPI 2019	**0.7544**	**0.0459**	**0.1224**
TGAT (Single-task)	BPI 2014	**0.4259**	0.4522	0.5373
	BPI 2016	0.6983	0.0496	0.0287
	BPI 2017	0.7937	0.4441	0.3015
	BPI 2019	0.7112	0.0477	0.1248

adjust the influence of historical events based on their temporal distance. This results in superior accuracy for next-activity prediction and lower MAE for both next-timestamp and remaining-time prediction.

Overall, these results demonstrate that (i) relative time representations are more effective than absolute timestamps, and (ii) explicitly modeling time as a first-class component of message passing, rather than as a static feature, is critical for accurate predictive process monitoring.

4.3.6. Multi-task vs single-task learning

We isolate the contribution of joint optimization by training each model in two regimes: a *multi-task* setting in which next activity, next timestamp, and remaining time are predicted jointly, and a *single-task* setting in which a separate model is trained for each target. To benchmark the proposed framework against state-of-the-art alternatives, the comparison covers two temporal graph models (TGN and TGAT) and two strong sequential baselines (LSTM and ProcessTransformer) under both regimes, so that multi-task TGAT is evaluated directly against the best single-task predictors reported in the recent literature rather than only against its own single-task counterpart.

Table 9 reports the results on the four BPI datasets. Benchmarked against the strongest single-task predictors, multi-task TGAT attains the best accuracy on BPI 2017 and 2019, the lowest MAE_{next} on BPI 2016 and 2019, and the lowest MAE_{rem} on BPI 2019. On the remaining cells it trails the best single-task predictor, in most cases only marginally (e.g., accuracy of 0.7022 versus 0.7055 on BPI 2016 and MAE_{next} of 0.4244 versus 0.4227 on BPI 2017), with the largest gap on BPI 2017 MAE_{rem}, where the memory-based multi-task TGN leads (0.2845 versus 0.2966). The one clear exception is BPI 2014 next-activity accuracy, where TGAT and ProcessTransformer in their single-task forms reach 0.4259 and 0.4195 against 0.3489 for multi-task TGAT; we attribute this gap to the limited process-variant diversity of BPI 2014, which weakens the auxiliary supervision signal available to joint training.

Two complementary patterns emerge from the table. First, the temporal graph models (TGN and TGAT) benefit from joint training: in most cells their multi-task variants match or exceed their single-task counterparts, indicating that time-aware neighborhood aggregation can absorb gradients from heterogeneous objectives. Second, the sequential baselines (LSTM and ProcessTransformer) follow the opposite trend, with single-task variants generally outperforming the multi-task ones. This contrast suggests that recurrent and self-attention architectures are more susceptible to negative transfer once activity and time signals share a single hidden state.

The advantage of multi-task TGAT is therefore not uniform across every cell but arises, where it does, from the interaction between time-aware attention and joint supervision: the next-time and remaining-time targets surface waiting and bottleneck patterns that the temporal graph propagates and that the activity head reuses. This mechanism is unavailable to flat sequential encoders, which explains both why their best results occur in the single-task regime and why multi-task TGAT remains competitive with — and on several datasets superior to — state-of-the-art single-task models, while the residual gaps on the remaining cells stay small.

We note that Table 9 reports single-configuration point estimates rather than multi-seed means; several of the multi-task-versus-single-task gaps are therefore small relative to the seed-level variation reported in Appendix B. We accordingly describe this encoder-dependent pattern as a consistent tendency—with visible exceptions such as the next-activity accuracy of TGAT on BPI 2014—rather than as a significance-tested effect.

4.4. Ablation study

To better understand the contribution of each architectural component in the proposed TGAT-based multi-task predictive monitoring model, we conducted a comprehensive ablation study on four BPI Challenge datasets (2014, 2016, 2017, and 2019). In each ablation setting, one component is removed or simplified while all other configurations are kept identical to those of the full model. We report the impact on next-activity prediction accuracy, next-timestamp MAE, and remaining-time MAE.

Table 10
Ablation study results across BPI Challenge datasets.

	BPI 2014			BPI 2016			BPI 2017			BPI 2019		
Model	ACC	MAE_{next}	MAE_{rem}	ACC	MAE_{next}	MAE_{rem}	ACC	MAE_{next}	MAE_{rem}	ACC	MAE_{next}	MAE_{rem}
TGAT (full)	**0.3489**	**0.4495**	**0.5330**	**0.7022**	**0.0462**	**0.0285**	**0.8118**	**0.4244**	**0.2966**	**0.7544**	**0.0459**	**0.1224**
w/o time enc.	0.3193	0.5846	0.5369	0.6995	0.0485	0.0312	0.8023	0.4666	0.3189	0.7102	0.0541	0.1572
w/o ctx time	0.3293	0.5928	0.5367	0.6994	0.0512	0.0333	0.8010	0.4555	0.3321	0.7289	0.0521	0.1344
w/o ctx	0.2610	0.5995	0.5789	0.6789	0.0477	0.0310	0.7110	0.4568	0.3039	0.6316	0.0511	0.1445
w/o multi-head	0.3444	0.4512	0.5571	0.7009	0.0471	0.0299	0.8009	0.4459	0.3012	0.7219	0.0490	0.1271
w/o act. emb.	0.3119	0.4512	0.5571	0.7009	0.0471	0.0295	0.6885	0.4591	0.3111	0.7215	0.0470	0.1251

Ablation settings We consider the following variants:

- **w/o time encoding**: removes the Δt-based time encoding from the temporal attention mechanism, preventing the model from explicitly exploiting continuous time gaps.
- **w/o context time**: ignores temporal distances between historical events and the current event, and only uses the last L events as unordered contextual information.
- **w/o context**: disables context-based attention and updates node representations using only residual connections and feed-forward networks.
- **w/o multi-head**: replaces multi-head attention with a single attention head.
- **w/o activity embedding**: removes activity-specific embeddings and retains only auxiliary feature information.

Overall results Tables 10 summarize the ablation results in the four datasets.

Discussion Across all datasets, removing *time encoding* or *context time* consistently degrades both classification and regression performance, confirming that explicitly modeling continuous temporal gaps is critical for predictive process monitoring. The performance drop is particularly pronounced in BPI 2014 and 2019, which exhibit a high temporal irregularity.

Disabling *context attention* results in the most severe accuracy degradation, indicating that selectively attending to relevant historical events is more effective than compressing the entire history into a fixed representation. The impact of removing *multi-head attention* is comparatively smaller, but still observable, suggesting that different heads capture complementary temporal patterns. Finally, removing *activity embeddings* consistently degrades performance, underscoring the importance of explicitly modeling activity semantics alongside temporal information.

In general, the ablation results validate that the proposed TGAT architecture benefits from the joint integration of temporal encoding, context-aware attention, and activity-aware representations.

5. Conclusions and future work

In this paper, we investigated predictive process monitoring (PPM) from the dual perspectives of temporal graph modeling and multi-task learning. Unlike prior approaches that address object-centric modeling, temporal graph learning, and prediction tasks in isolation, our work provides a unified framework that jointly captures structural, temporal, and task-level dependencies within a single architecture. By leveraging Object-Centric Event Logs (OCEL), we derived temporally ordered prediction instances and modeled each prefix as a dynamic temporal graph. This formulation enables the explicit representation of complex temporal dependencies and irregular inter-event intervals. Based on this architectural foundation, we proposed a multi-task predictive framework utilizing the Temporal Graph Attention Network (TGAT) to jointly optimize three critical objectives: next activity prediction, next inter-event time estimation, and remaining time prediction. This joint design enables the model to capture cross-task temporal dependencies, which are not effectively leveraged in prior single-task or loosely coupled approaches.

Extensive experimental evaluation on four real-world BPI Challenge datasets (2014, 2016, 2017, and 2019) demonstrates that our proposed MT-TGPM framework is the only model that ranks first or second in 15 of the 16 dataset–metric cells under the temporal split — i.e., the only approach that performs strongly across all four datasets and all three tasks simultaneously — attaining the lowest MAE_{next} on every dataset while remaining statistically competitive in the cells where a specialized baseline leads (Appendix B). The results highlight that temporal graph modeling is particularly robust in process environments characterized by irregular "rhythms" and long-range temporal dependencies. Furthermore, the effect of the multi-task formulation is encoder-dependent: temporal graph encoders (TGN and TGAT) tend to benefit from joint supervision, whereas sequential encoders tend to exhibit negative transfer under the same objective (Section 4.3.6). This indicates that activity-flow and time-related signals reinforce each other only when aggregated through time-aware attention, rather than providing an unconditional benefit. These results indicate that the performance gains stem not only from individual components, but from their tight integration within a unified temporal graph-based multi-task framework.

Through rigorous ablation studies and attention visualizations, we further established that explicitly encoding continuous time gaps through a learnable mechanism is vital for predictive accuracy. Our findings confirm that temporal information should not be treated as a secondary auxiliary feature, but rather as a first-class structural component that directly modulates message passing and attention weights. Qualitative analysis of learned attention patterns also provides evidence that different attention heads specialize in capturing distinct process contexts, such as early-stage stabilizers, mid-prefix triggers, or short-term recency effects. This further confirms that the effectiveness of the proposed approach arises from the synergistic interaction between temporal encoding, attention mechanisms, and multi-task learning, rather than from a simple combination of existing techniques.

Despite these promising results, several limitations remain to be addressed in future work. First, while our framework utilizes OCEL to define meaningful prediction instances, the core temporal modeling still operates on flattened event sequences. Extending the temporal graph attention mechanism to fully exploit multi-object interactions and inter-object relational dependencies is a priority, including the adoption of richer process-execution extraction techniques (e.g., connected-component process executions) in place of the leading-object flattening used in this study. Second, the current model assumes reliable timestamp data; thus, incorporating uncertainty-aware modeling or robust handling of noisy and missing temporal information would enhance its practical applicability in volatile industrial settings. Third, the multi-task objective of this study is deliberately kept as an unweighted sum of the task losses, so that the multi-task-versus-single-task comparison in Section 4.3.6 isolates the effect of the shared temporal representation from any effect of loss balancing (Section 3.2.3). Adaptive task-weighting schemes, such as uncertainty-based weighting or gradient normalization, are orthogonal to the contribution of this work; exploring such dynamic task-weighting strategies to balance multi-task gradients is therefore a natural next step. Future research will also investigate the framework's scalability for online streaming predictive monitoring.

Declaration of interests

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Minseok Song reports financial support was provided by National Research Foundation of Korea. Minseok Song reports financial support was provided by Institute of Information & Communications Technology Planning & Evaluation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

4TU.ResearchData (2014). Bpi challenge 2014 (ocel). https://data.4tu.nl/datasets/7d097cec-7304-4b85-9e78-a3ca1cc44c40/1.

4TU.ResearchData (2016). Bpi challenge 2016 (ocel). https://data.4tu.nl/datasets/85513b02-29a5-49cb-b196-69408c9dc67d.

4TU.ResearchData (2017). Bpi challenge 2017 (ocel). https://data.4tu.nl/datasets/8889ca3f-97cf-459a-b630-30b81d8664b.

4TU.ResearchData (2019). Bpi challenge 2019 (ocel). https://data.4tu.nl/datasets/46a7e15b-10c7-4ab2-988d-ec67d8ea8515.

Van der Aalst, W. (2019). Object-centric process mining: dealing with divergence and convergence in event data, in: International Conference on Software Engineering and Formal Methods, Springer. pp. 3–25.

van der Aalst, W.M.P. (2019). Object-centric process mining: Dealing with divergence and convergence in event data, in: International Conference on Software Engineering and Formal Methods (SEFM), Springer. pp. 3–25.

Adams, J.N., Park, G., van der Aalst, W. (2023). Preserving complex object-centric graph structures to improve machine learning tasks in process mining. Engineering Applications of Artificial Intelligence 125, 106764.

Adams, J.N., Schuster, D., Schmitz, S., Schuh, G., van der Aalst, W.M.P. (2022). Defining cases and variants for object-centric event data, in: International Conference on Process Mining (ICPM), IEEE. pp. 128–135.

Amiri Elyasi, K., van der Aa, H., Stuckenschmidt, H. (2024). PGTNet: A process graph transformer network for remaining time prediction of business process instances, in: International Conference on Advanced Information Systems Engineering (CAiSE), Springer. pp. 124–140.

Bukhsh, Z.A., Saeed, A., Dijkman, R.M. (2021). Processtransformer: Predictive business process monitoring with transformer network. arXiv preprint arXiv:2104.00721 .

Ceravolo, P., Comuzzi, M., De Weerdt, J., Di Francescomarino, C., Maggi, F.M. (2024). Predictive process monitoring: concepts, challenges, and future research directions. Process Science 1, 2.

Chiorrini, A., Diamantini, C., Genga, L., Potena, D. (2023). Multi-perspective enriched instance graphs for next activity prediction through graph neural network. Journal of Intelligent Information Systems 61, 5–25.

De Smedt, J., De Weerdt, J. (2024). Predictive process model monitoring using long short-term memory networks. Engineering Applications of Artificial Intelligence 133, 108295.

Di Francescomarino, C., Ghidini, C., Maggi, F.M., Milani, F. (2018). Predictive process monitoring methods: Which one suits me best?, in: International Conference on Business Process Management, Springer. pp. 462–479.

Fioretto, S., Masciari, E. (2025). A comparative analysis of predictive process monitoring: object-centric versus classical event logs: S. fioretto, e. masciari. Knowledge and Information Systems , 1–44.

Galanti, R., De Leoni, M., Navarin, N., Marazzi, A. (2023). Object-centric process predictive analytics. Expert Systems with Applications 213, 119173.

Galanti, R., de Leoni, M. (2023). Predictive analytics for object-centric processes: Do graph neural networks really help?, in: International Conference on Business Process Management, Springer. pp. 521–533.

Gherissi, W., Acheli, M., Haddad, J.E., Grigori, D. (2024). Predictive process monitoring using object-centric graph embeddings, in: International Conference on Service-Oriented Computing, Springer. pp. 55–67.

Goossens, A., De Smedt, J., Vanthienen, J. (2024). Object-centric event logs: Characteristics, comparative analysis and road map, in: International Conference on Business Process Management, Springer. pp. 37–54.

Hennig, M.C., Schmidt, R. (2025). Leveraging temporal graphs for enhancing transformer-based predictive process monitoring, in: International Conference on Business Process Management, Springer. pp. 291–307.

Hennig, M.C., Schmidt, R., Möhring, M. (2025). Temporal gnns for remaining time prediction: an evaluation, in: 2025 7th International Conference on Process Mining (ICPM), IEEE. pp. 1–8.

Jia, Y., Wang, J., Hosseini, M.R., Shou, W., Wu, P., Mao, C. (2024). Temporal graph attention network for building thermal load prediction.

Energy and Buildings 321, 113507.
Ke, L., Huan, F., Yifei, X., Chifeng, S. (2025). Multi-task prediction method based on ggcn for object centric event logs. IEEE Access .
Khayatbashi, S., Hartig, O., Jalali, A. (2023). Transforming event knowledge graph to object-centric event logs: A comparative study for multi-dimensional process analysis, in: International Conference on Conceptual Modeling, Springer. pp. 220–238.
Kratsch, W., Manderscheid, J., Röglinger, M., Seyfried, J. (2021). Machine learning in business process monitoring: A comparison of deep learning and classical approaches used for outcome prediction. Business & Information Systems Engineering 63, 261–276.
Lischka, A., Rauch, S., Stritzel, O. (2025). Directly follows graphs go predictive process monitoring with graph neural networks. arXiv preprint arXiv:2503.03197 .
Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C. (2014). Predictive monitoring of business processes, in: International Conference on Advanced Information Systems Engineering, Springer. pp. 457–472.
Márquez-Chamorro, A.E., Resinas, M., Ruiz-Cortés, A. (2017). Predictive monitoring of business processes: a survey. IEEE Transactions on Services Computing 11, 962–977.
Pasquadibisceglie, V., Scaringi, R., Appice, A., Castellano, G., Malerba, D. (2024). PROPHET: Explainable predictive process monitoring with heterogeneous graph neural networks. IEEE Transactions on Services Computing 17, 4111–4124.
Pegoraro, M., Uysal, M.S., Georgi, D.B., van der Aalst, W. (2021). Text-aware predictive monitoring of business processes, in: Business Information Systems, pp. 221–232.
Rama-Maneiro, E., Vidal, J.C., Lama, M. (2021). Deep learning for predictive business process monitoring: Review and benchmark. IEEE Transactions on Services Computing 16, 739–756.
Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., Bronstein, M. (2020). Temporal graph networks for deep learning on dynamic graphs. arXiv preprint arXiv:2006.10637 .
Ruiz, C., Alaíz, C.M., Dorronsoro, J.R. (2024). A survey on kernel-based multi-task learning. Neurocomputing 577, 127255.
Smit, T.K., Reijers, H.A., Lu, X. (2024). Hoeg: A new approach for object-centric predictive process monitoring, in: International Conference on Advanced Information Systems Engineering, Springer. pp. 231–247.
Tax, N., Verenich, I., La Rosa, M., Dumas, M. (2017). Predictive business process monitoring with lstm neural networks, in: International Conference on Advanced Information Systems Engineering, Springer. pp. 477–492.
Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M. (2019). Outcome-oriented predictive process monitoring: Review and benchmark. ACM Transactions on Knowledge Discovery from Data (TKDD) 13, 1–57.
Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., Van Gool, L. (2021). Multi-task learning for dense prediction tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 44, 3614–3633.
Wang, J., Sui, Y., Liu, C., Shen, X., Li, Z., Yu, D. (2024). Multi-task learning with multi-gate mixture of transformer-based experts for predictive process monitoring in manufacturing. Science Progress 107, 00368504241292196.
Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., Achan, K. (2020). Inductive representation learning on temporal graphs. arXiv preprint arXiv:2002.07962 .
Zhang, Y., Yang, Q. (2021). A survey on multi-task learning. IEEE Transactions on Knowledge and Data Engineering 34, 5586–5609.
Zhang, Z., Yu, W., Yu, M., Guo, Z., Jiang, M. (2023). A survey of multi-task learning in natural language processing: Regarding task relatedness and training methods, in: Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pp. 943–956.

A. Random-split results (for comparability with prior work)

The main text evaluates all models under the temporal case-level split (Section 4.2.5), which reflects the forecasting nature of predictive process monitoring. Because two of the compared approaches were originally reported under random case-level splits (Galanti et al., 2023; Ke et al., 2025), this appendix reports the corresponding random-split results solely for comparability with that line of work. The random split assumes the absence of concept drift and is not the recommended evaluation protocol for predictive process monitoring; no conclusion in the main text relies on the numbers below.

B. Statistical robustness of the overall comparison

This appendix complements the overall comparison of Table 7 by reporting the statistical robustness of all results under the temporal case-level split. For every model and dataset, test performance is reported as mean $\pm$ standard deviation over 5 random seeds under the per-dataset tuned configurations selected in Table 6; all other factors — the data splits, the prefix generation procedure, and the shared optimization settings of Table 4 — are held fixed, so that the only source of variation between runs is the random initialization and the shuffling of the training batches. For readability the results are split into two tables: Table B.1 covers BPI 2014 and BPI 2016, and Table B.2 covers BPI 2017 and BPI 2019.

Significance testing protocol

Mean values alone do not establish whether a difference between two models is systematic or merely a consequence of run-to-run variation. We therefore assess each difference with a *paired Wilcoxon signed-rank test*, a non-parametric counterpart of the paired t-test. The test operates on the paired differences $d_k = m_k^{\mathrm{TGAT}} - m_k^{\mathrm{base}}$ for $k = 1, \ldots, 5$, where m_k^{TGAT} and m_k^{base} denote the test-set value of a given metric obtained by the proposed model and by the competing model on the k-th seed. The differences are ranked by absolute magnitude, the ranks are signed according to the direction of each difference, and the test statistic is the sum of the positive signed ranks. Unlike the paired t-test, the procedure assumes neither normality nor equal variances of the differences, which is appropriate here because the metrics are either bounded (ACC, macro F1) or strictly positive and right-skewed (the MAE metrics), and because the number of runs is small. Pairing by seed is essential: it removes the between-seed variation to which both models are equally exposed, so that the test isolates the difference attributable to the encoder rather than to the initialization.

The test is applied per dataset–metric cell. In each cell we first identify the *strongest baseline* — the competing model with the best mean value of that metric on that dataset, which differs from cell to cell (LSTM for the classification metrics of BPI 2016 and BPI 2017, TGN for MAE_{rem} of BPI 2017,

Table A.1
Overall performance comparison across BPI Challenge datasets under the random case-level split (reported for comparability with prior work only; see Section 4.2.5), grouped by non-object-centric PPM and object-centric PPM approaches. ACC denotes accuracy and F1 denotes macro F1-score. The best results are highlighted in **bold**, while the second-best results are underlined. For MAE metrics, lower values indicate better performance.

Category	Model	BPI 2014				BPI 2016				BPI 2017				BPI 2019			
		ACC	F1	MAE_{next}	MAE_{rem}	ACC	F1	MAE_{next}	MAE_{rem}	ACC	F1	MAE_{next}	MAE_{rem}	ACC	F1	MAE_{next}	MAE_{rem}
Non-OCPM	MLP	0.2616	0.1415	0.4492	0.5340	0.6914	0.1124	0.0490	0.0266	0.5250	0.4412	0.4880	0.3461	0.4696	0.2243	0.0612	0.1300
	LSTM	0.3515	0.1819	0.4500	0.5330	**0.7095**	0.1132	0.0460	0.0265	0.8155	0.7004	0.4240	0.2940	**0.7665**	0.3512	0.0413	0.1202
	GRU	0.3510	0.1779	0.4512	0.5340	0.7081	0.1150	0.0478	0.0277	0.8153	0.7002	0.4240	0.2980	0.7647	0.3508	0.0412	0.1231
	ProcessTransformer (Bukhsh et al., 2021)	**0.3566**	**0.1882**	0.4482	0.5390	0.7081	0.1167	0.0470	0.0265	0.8154	0.7014	**0.4141**	0.2999	0.7619	0.3555	0.0511	0.1208
	DFG-GNN (Lischka et al., 2025)	0.3457	0.1541	0.4556	0.5444	0.7022	0.1003	0.0512	0.0333	0.8011	0.6997	0.4414	0.3014	0.7544	**0.3611**	0.0588	0.1331
	PGTNet (Amiri Elyasi et al., 2024)	0.3211	0.1645	0.4811	0.5565	0.6978	0.1016	0.0466	0.0298	0.8011	0.6617	0.4455	0.2985	0.7215	0.3615	0.0661	0.1615
	TGN (Hennig et al., 2025)	0.3082	0.0875	0.4885	0.5440	0.7054	0.1329	0.0460	0.0264	0.8109	0.6962	0.4241	**0.2832**	0.6546	0.1920	0.0512	0.1302
OCPM	GOCEL (Adams et al., 2023)	0.2645	0.1144	0.5442	0.5912	0.6995	0.0812	0.0557	0.0522	0.7242	0.5512	0.4992	0.3241	0.5112	0.1444	0.0599	0.1522
	HOEG (Smit et al., 2024)	0.2777	0.1124	0.5441	0.5888	0.7000	0.0522	0.0555	0.0412	0.7645	0.6012	0.4882	0.3011	0.6222	0.1775	0.0488	0.1412
	GGCN (Ke et al., 2025)	0.2707	0.0946	0.5122	0.5710	0.7011	0.0519	0.0480	0.0341	0.7536	0.4924	0.4512	0.2999	0.6162	0.1509	0.0522	0.1305
	TGAT (ours)	0.3521	0.1742	**0.4480**	**0.5310**	0.7082	**0.1402**	**0.0450**	**0.0264**	**0.8155**	**0.7066**	0.4238	0.2950	0.7610	0.3603	**0.0411**	**0.1201**

ProcessTransformer for macro F1 of BPI 2014, and so on) — and then test the proposed model against that single baseline. Comparing against the strongest baseline rather than against all ten baselines is deliberately conservative: it evaluates the claim in the form in which it would be contested, and it avoids the inflation of the family-wise error rate that ten pairwise tests per cell would introduce. We use the one-sided form of the test, with the alternative hypothesis that the proposed model is *better* than the strongest baseline on the metric in question (higher for ACC and macro F1, lower for the MAE metrics), and a dagger (†) marks the cells in which this alternative is accepted at $p < 0.05$. The absence of a dagger indicates only that the proposed model is not significantly better than the strongest baseline in that cell; it does not by itself establish that the baseline is significantly better.

Results

Three observations follow from Tables B.1 and B.2.

First, seed-to-seed variability is small and comparable across models: no standard deviation exceeds 0.006 on any metric, and the spread of the proposed model is of the same order as that of the baselines. The ranking reported in Section 4.3.1 is therefore stable and is not an artefact of a particular initialization. The same figures, however, caution against over-reading the narrowest margins: the accuracy gap of 0.0008 between LSTM and the proposed model on BPI 2016, for instance, is a small fraction of the corresponding standard deviations and does not by itself distinguish the two models.

Second, the proposed model is significantly better than the strongest baseline in ten of the sixteen dataset–metric cells. The advantage is complete on BPI 2019, where the improvement is significant on all four metrics, and it extends to three of four metrics on BPI 2017 (ACC, macro F1, MAE_{next}) and to both MAE metrics on BPI 2014. Notably, MAE_{next} — the metric most directly tied to the modeling of inter-event intervals — is significantly improved on all four datasets, which is the pattern one would expect if the gain originated in the continuous-time encoding rather than in model capacity alone.

Third, the cells without a dagger are informative in their own right, and we do not claim significance where the data do not support it. On BPI 2016 the proposed model is significantly better only on MAE_{next}: LSTM is ahead on accuracy and, more substantially, on macro F1 (0.1536 versus 0.1455), and ProcessTransformer attains the lowest MAE_{rem}. BPI 2016 is the log in which the temporal irregularity that our encoder is designed to exploit is least pronounced, so that a well-tuned sequential encoder is already close to the ceiling on the classification tasks. On BPI 2017, TGN retains a clear advantage on MAE_{rem} (0.2845 versus 0.2966), indicating that a latent memory state remains an effective inductive bias for long-horizon time regression — a limitation of the proposed architecture that we state explicitly rather than obscure. On BPI 2014 the macro F1 difference with respect to ProcessTransformer is small and not significant. Taken together, the picture is not one of uniform dominance but of consistent strength across heterogeneous logs: the proposed model is the only one that is at or close to the top of every dataset–metric cell simultaneously, whereas each baseline that overtakes it in one cell falls markedly behind in several others (LSTM on BPI 2019 macro F1, TGN on all BPI 2019 metrics, ProcessTransformer on the BPI 2019 MAE_{next}).

Table B.1
Statistical robustness of the overall comparison under the temporal case-level split on BPI 2014 and BPI 2016: test performance reported as mean ± standard deviation over 5 random seeds. A dagger (†) marks cells in which the proposed model is significantly better than the strongest baseline of that dataset–metric cell (one-sided paired Wilcoxon signed-rank test across seeds, $p < 0.05$; see the protocol described above). ACC and F1 denote accuracy and macro F1-score; for MAE metrics, lower values are better.

Category	Model	BPI 2014				BPI 2016			
		ACC	F1	MAE_{next}	MAE_{rem}	ACC	F1	MAE_{next}	MAE_{rem}
Non-OCPM	MLP	0.2610 ±0.0020	0.1440 ±0.0028	0.4522 ±0.0028	0.5448 ±0.0044	0.6812 ±0.0022	0.1118 ±0.0021	0.0495 ±0.0044	0.0279 ±0.0035
	LSTM	0.3360 ±0.0022	0.1630 ±0.0032	0.4551 ±0.0035	0.5371 ±0.0042	0.7030 ±0.0039	0.1536 ±0.0044	0.0488 ±0.0024	0.0288 ±0.0028
	GRU	0.3444 ±0.0025	0.1668 ±0.0033	0.4550 ±0.0024	0.5412 ±0.0043	0.7012 ±0.0039	0.1332 ±0.0016	0.0512 ±0.0043	0.0312 ±0.0033
	ProcessTransformer (Bukhsh et al., 2021)	0.3422 ±0.0033	0.1725 ±0.0046	0.4951 ±0.0027	0.5579 ±0.0033	0.6978 ±0.0037	0.1225 ±0.0027	0.0499 ±0.0038	0.0275 ±0.0026
	DFG-GNN (Lischka et al., 2025)	0.3415 ±0.0027	0.1512 ±0.0029	0.4892 ±0.0045	0.5677 ±0.0045	0.6844 ±0.0035	0.1022 ±0.0036	0.0612 ±0.0047	0.0399 ±0.0038
	PGTNet (Amiri Elyasi et al., 2024)	0.3222 ±0.0034	0.1655 ±0.0030	0.4811 ±0.0027	0.5555 ±0.0022	0.6999 ±0.0035	0.1016 ±0.0033	0.0477 ±0.0026	0.0298 ±0.0039
	TGN (Hennig et al., 2025)	0.3040 ±0.0036	0.0855 ±0.0042	0.4898 ±0.0029	0.5477 ±0.0024	0.7011 ±0.0019	0.1311 ±0.0036	0.0466 ±0.0048	0.0278 ±0.0038
OCPM	GOCEL (Adams et al., 2023)	0.2612 ±0.0019	0.1099 ±0.0039	0.5688 ±0.0038	0.6142 ±0.0025	0.6970 ±0.0026	0.0822 ±0.0025	0.0577 ±0.0031	0.0544 ±0.0044
	HOEG (Smit et al., 2024)	0.2740 ±0.0032	0.1113 ±0.0039	0.5477 ±0.0033	0.5978 ±0.0045	0.6992 ±0.0048	0.0514 ±0.0037	0.0578 ±0.0032	0.0489 ±0.0021
	GGCN (Ke et al., 2025)	0.2725 ±0.0043	0.0955 ±0.0046	0.5144 ±0.0029	0.5788 ±0.0034	0.7009 ±0.0044	0.0518 ±0.0036	0.0499 ±0.0024	0.0388 ±0.0018
	TGAT (ours)	0.3489 ±0.0030	0.1711 ±0.0029	0.4495 ±0.0030†	0.5330 ±0.0025†	0.7022 ±0.0038	0.1455 ±0.0021	0.0462 ±0.0041†	0.0285 ±0.0028

Table B.2
Statistical robustness of the overall comparison under the temporal case-level split on BPI 2017 and BPI 2019: test performance reported as mean ± standard deviation over 5 random seeds. A dagger (†) marks cells in which the proposed model is significantly better than the strongest baseline of that dataset–metric cell (one-sided paired Wilcoxon signed-rank test across seeds, $p < 0.05$; see the protocol described above). ACC and F1 denote accuracy and macro F1-score; for MAE metrics, lower values are better.

Category	Model	BPI 2017				BPI 2019			
		ACC	F1	MAE_{next}	MAE_{rem}	ACC	F1	MAE_{next}	MAE_{rem}
Non-OCPM	MLP	0.5222 ±0.0044	0.4400 ±0.0022	0.4895 ±0.0048	0.3471 ±0.0033	0.4677 ±0.0049	0.2245 ±0.0031	0.0611 ±0.0026	0.1344 ±0.0043
	LSTM	0.8103 ±0.0047	0.6947 ±0.0038	0.4255 ±0.0044	0.2967 ±0.0050	0.7522 ±0.0038	0.3415 ±0.0028	0.0463 ±0.0030	0.1285 ±0.0045
	GRU	0.8022 ±0.0044	0.6889 ±0.0033	0.4289 ±0.0041	0.3112 ±0.0037	0.7420 ±0.0025	0.3312 ±0.0031	0.0489 ±0.0047	0.1255 ±0.0039
	ProcessTransformer (Bukhsh et al., 2021)	0.8035 ±0.0025	0.6887 ±0.0023	0.4289 ±0.0022	0.3315 ±0.0037	0.7533 ±0.0045	0.3412 ±0.0022	0.0608 ±0.0033	0.1279 ±0.0037
	DFG-GNN (Lischka et al., 2025)	0.7899 ±0.0040	0.6788 ±0.0041	0.4667 ±0.0035	0.3389 ±0.0039	0.7414 ±0.0041	0.3518 ±0.0038	0.0662 ±0.0039	0.1571 ±0.0042
	PGTNet (Amiri Elyasi et al., 2024)	0.8014 ±0.0037	0.6618 ±0.0046	0.4445 ±0.0037	0.2985 ±0.0037	0.7212 ±0.0038	0.3612 ±0.0032	0.0661 ±0.0051	0.1611 ±0.0040
	TGN (Hennig et al., 2025)	0.7988 ±0.0035	0.6882 ±0.0048	0.4255 ±0.0039	0.2845 ±0.0022	0.6488 ±0.0037	0.1879 ±0.0018	0.0566 ±0.0027	0.1388 ±0.0037
OCPM	GOCEL (Adams et al., 2023)	0.7108 ±0.0019	0.5335 ±0.0041	0.5118 ±0.0026	0.3441 ±0.0041	0.5022 ±0.0045	0.1321 ±0.0058	0.0615 ±0.0045	0.1778 ±0.0044
	HOEG (Smit et al., 2024)	0.7612 ±0.0033	0.5998 ±0.0041	0.4987 ±0.0038	0.3311 ±0.0044	0.6022 ±0.0038	0.1745 ±0.0022	0.0497 ±0.0035	0.1477 ±0.0047
	GGCN (Ke et al., 2025)	0.7446 ±0.0020	0.4878 ±0.0035	0.4588 ±0.0027	0.3021 ±0.0031	0.6244 ±0.0031	0.1588 ±0.0032	0.0575 ±0.0028	0.1342 ±0.0038
	TGAT (ours)	0.8118 ±0.0035†	0.7042 ±0.0035†	0.4244 ±0.0030†	0.2966 ±0.0042	0.7544 ±0.0046†	0.3644 ±0.0018†	0.0459 ±0.0033†	0.1224 ±0.0034†