

Universiteit Leiden

Transformer-Based Pattern Mining and Quantification for Semiconductor Lithography Efficiency Loss Analysis

Jing Lu (s4185048)

Supervisors:
Matthijs van Leeuwen & Lincen Yang

Master Thesis– Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

June 22, 2026

Abstract

Lithography scanner productivity is a key determinant of semiconductor manufacturing throughput, where even small timing inefficiencies can lead to significant losses at the fab level. Understanding and quantifying the contribution of specific event sequence patterns to timing metrics is essential for targeted root-cause analysis and proactive productivity optimization.

This thesis proposes an end-to-end pipeline that combines a transformer-based timing prediction model with a post-hoc token replacement procedure to quantify the impact of sequential event patterns on scanner timing metrics. Two transformer architectures, Attention Pooling Based and CLS Token Based aggregation, are compared against a Naive time-difference baseline. Controlled experiments on a synthetic dataset with known ground-truth pattern impacts show that the CLS Token Based method consistently achieves the most accurate quantification across varying levels of data complexity, while both learned methods provide per-sequence pattern contributions that the Naive baseline cannot. Applied to real industrial logs from two scanners, the CLS Token Based method achieves R^2=0.950 (FLOH) and R^2=0.755 (WST) on one scanner, and R^2=0.861 (FLOH) and R^2=0.848 (WST) on the other scanner. Expert validation confirms that the majority of mined patterns for FLOH and WST are recognized as genuine productivity-relevant behaviors on both scanners. The pipeline is deployed as an interactive Spotfire dashboard supporting engineers in root-cause diagnosis of previously unknown productivity losses.

Acknowledgements

I am grateful to LIACS at Leiden University and ASML for providing me with the opportunity and the platform to complete this thesis project in an industry setting, which is exactly the kind of experience I had been hoping for.

In Leiden University, My deepest thanks go to my thesis supervisor Matthijs van Leeuwen. His guidance on the research direction has been extremely valuable throughout this project. Every time we met, he could quickly identify the key issues and point me in the right direction, which saved me from going down many wrong paths. His way of thinking about research problems has also influenced how I approach my own work. I would also like to thank Lincen Yang for reviewing my thesis and providing detailed, helpful feedback that significantly improved the final version.

At ASML, I would like to thank my manager Hrishikesh Salunkhe. He helped me shape the topic in the early stage of the project, connecting the research with real industry needs, and provided consistent guidance on progress throughout the internship. I would also like to thank my technical mentor Roel Borgers, who guided me through many technical details and helped me think through implementation challenges. I am also grateful to Fred Knijnenburg, who shared his deep domain knowledge with me in great detail and spent time reviewing my results and discussing their implications from an industry perspective. I would also like to thank Animesh Sahoo, Rathinjai Chauhan, Selin Tas, and all the other colleagues at ASML who helped me in various ways during this project. Their support and willingness to share knowledge made this internship a truly enriching experience.

Finally, I would like to thank my family and friends for their support and encouragement throughout this journey.

Contents

Acknowledgements **1**

1 Introduction **1**
1.1 Background . . . 1
1.1.1 Semiconductor manufacturing pipeline . . . 1
1.1.2 Critical timing metrics and their impact . . . 2
1.1.3 Current industry practices for productivity analysis . . . 3
1.1.4 Current research . . . 4
1.1.5 Research opportunities . . . 5
1.2 Research Questions . . . 6
1.3 Approach . . . 6
1.4 Contributions . . . 7
1.5 Thesis Overview . . . 7

2 Related Work **8**
2.1 Pattern Mining in Event Logs . . . 8
2.1.1 Sequential pattern mining . . . 8
2.1.2 Process mining . . . 9
2.1.3 Summary . . . 9
2.2 Pattern Mining for Feature Selection . . . 9
2.2.1 Target-guided pattern discovery . . . 9
2.2.2 Using patterns as features for machine learning . . . 9
2.2.3 Summary . . . 10
2.3 Sequence-to-Target Machine Learning . . . 10
2.3.1 Recurrent models . . . 10
2.3.2 Transformer-based models . . . 10
2.3.3 Summary . . . 11
2.4 Quantifying and Interpreting Sequence Models . . . 11
2.4.1 Post-hoc explanation methods . . . 11
2.4.2 Attention-based interpretability . . . 11
2.4.3 Mechanistic interpretability . . . 11
2.4.4 Unified prediction and explanation . . . 12
2.4.5 Summary . . . 12
2.5 Summary and Positioning . . . 12

3 Problem Formulation **13**
3.1 Task Definition . . . 13
3.2 Formal Problem Definition . . . 13

4 Methodology **15**
4.1 End-to-End Pipeline Overview . . . 15
4.2 Pattern Mining and Consolidation . . . 17
4.2.1 PrefixSpan Mining . . . 17
4.2.2 Overlap-Ratio Consolidation . . . 17
4.3 Baseline: Naive Time Difference Method . . . 17
4.4 Overview of Transformer-Based Methods . . . 18
4.5 Shared Transformer Architecture . . . 18
4.6 Attention Pooling Based Method . . . 19
4.6.1 Architecture . . . 19
4.6.2 Quantification via Token Replacement . . . 20
4.7 CLS Token Based Method . . . 21
4.7.1 Architecture . . . 21
4.7.2 Quantification via Token Replacement . . . 21

5 Synthetic Dataset Experiments **23**
5.1 Evaluation Metrics . . . 23
5.2 Experimental Setup . . . 24
5.2.1 Factors to Mirror Real Event Logs . . . 24
5.2.2 Data Generation . . . 24
5.2.3 Ground-Truth Causal Patterns . . . 25
5.2.4 Factor Controlled Experiment . . . 26
5.2.5 Implementation Details . . . 27
5.3 Results and Analysis . . . 28
5.3.1 Prediction Performance . . . 29
5.3.2 Per-Pattern Global Quantification . . . 30
5.3.3 Gradient Analysis . . . 32
5.3.4 Computational Complexity . . . 38

6 Real Data Experiments **40**
6.1 Experimental Setup . . . 40
6.1.1 Data Source and Preprocessing . . . 40
6.1.2 Pattern Mining and Consolidation . . . 42
6.1.3 Model Training and Pattern Quantification . . . 43
6.1.4 Evaluation Metrics . . . 44
6.2 Dataset . . . 44
6.2.1 Scanner A . . . 45
6.2.2 Scanner B . . . 45
6.3 Results and Analysis . . . 48
6.3.1 Prediction Performance Across Targets . . . 48
6.3.2 Pattern Quantification Results and Expert Review . . . 50

6.3.3 Qualitative Expert Feedback on Unrecognized Patterns 55
6.4 Interactive Spotfire Dashboard for Pattern Exploration 56
6.4.1 Analysis Pipeline 56
6.4.2 Interactive Dashboard 56

7 Conclusion **58**
7.1 Conclusions 58
7.2 Strengths and Limitations 60
7.3 Future Work 61
References 65

Chapter 1

Introduction

Lithography scanner productivity is an essential KPI of semiconductor manufacturing throughput, where even small timing inefficiencies can lead to significant losses at the fab level. This chapter provides background on lithography scanner operation, critical timing metrics, and existing industrial and research practices to motivate the research questions addressed in this thesis.

1.1 Background

Semiconductor manufacturing is a highly complex process chain in which lithography plays a central role. The lithography scanners studied in this thesis are manufactured by ASML, the world's leading supplier of photolithography systems, whose tools are used in virtually all high-volume semiconductor manufacturing facilities. Lithography tools are the most expensive tools in the fab, and are therefore intentionally chosen as the bottleneck toolset to balance capital investment and maximize overall throughput. Consequently, even small inefficiencies in scanner operation can propagate through the manufacturing pipeline and result in significant losses in wafers per day (WPD), directly impacting cost per wafer and fab productivity.

1.1.1 Semiconductor manufacturing pipeline

The semiconductor manufacturing pipeline consists of a large number of tightly coupled process steps, including deposition, lithography, etching, and metrology, which are repeated across many process steps to build up the final device structure. Within this pipeline, lithography defines critical dimensions and overlay accuracy, making it both technologically and economically critical. Modern fabs therefore optimize around lithography capacity, treating scanners as the constraint resource whose productivity largely determines overall fab output.

Within a lithography cell, wafers are transported from the track into the scanner, processed through a sequence of measurement, exposure, and handling steps, and returned to the track. This sequence inside the scanner, commonly referred to as the life of a wafer, is highly parallelized: while one wafer is being exposed, others are simultaneously measured, aligned, or transferred. The Measure stage performs wafer alignment and levelling to determine the precise position before exposure, while the Expose stage applies the patterned light dose to transfer the circuit design onto the wafer. Scanner architectures such as TWINSCAN [2] intentionally decouple measurement and exposure stages to maximize utilization of the exposure lens, the most expensive component of

the system. As a result, productivity is largely governed by the temporal coordination between multiple wafers and the scanner's internal components and processing stages. Figure 1.1 illustrates this parallelised flow.

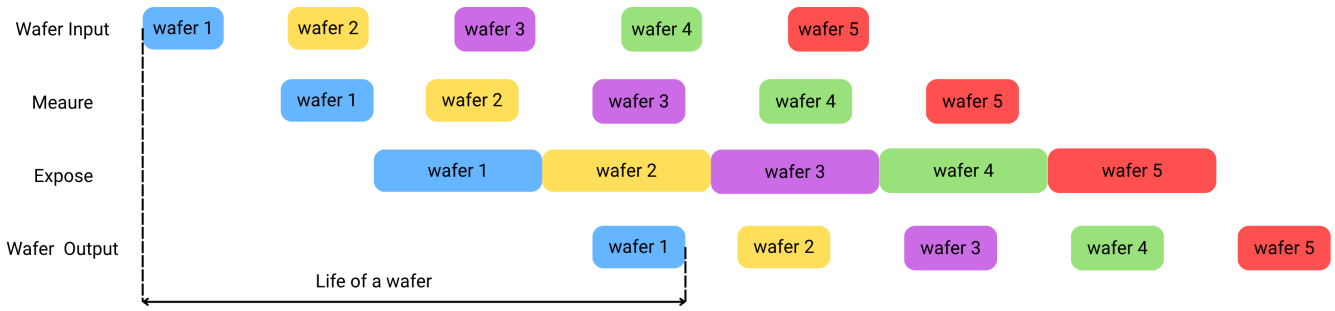


Figure 1.1: Life of a wafer inside a TWINSCAN lithography scanner. Five wafers are processed in parallel across the stages Wafer Input, Measure, Expose, and Wafer Output. The overlapping execution of measurement and exposure stages is central to maximizing scanner throughput.

1.1.2 Critical timing metrics and their impact

Scanner productivity is quantified through a hierarchy of timing metrics that capture losses at different granularities. At the high level, customers evaluate productivity in terms of wafers per day (WPD), which depends on throughput, utilization, availability, and rate efficiency. In this work, we focus specifically on timing-related losses that directly affect rate efficiency and lot throughput.

Scanner rate efficiency losses. Rate efficiency (RE) measures how closely a scanner operates relative to a reference throughput defined by an idealized reference job. Deviations arise from exposure losses, inter-wafer delays, and lot overheads. Exposure losses are often process-configuration-dependent, while inter-wafer delays can depend on both the process configuration and the dynamic interaction of wafer handling, alignment, and exposure activities. Lot overheads, in contrast, are largely independent of the process configuration and are primarily driven by operational and logistical factors. Understanding and reducing these losses is crucial for improving effective throughput without changes to hardware.

LOH (Lot Overhead). At the lot level, productivity losses are captured by lot overhead (LOH), which represents non-productive time before the first wafer and after the last wafer of a lot is exposed. A *lot* is a batch of wafers (typically 1–25) that are processed together as a unit through the scanner under a common process recipe. LOH is commonly decomposed into front lot overhead (FLOH) and back lot overhead (BLOH), reflecting activities such as lot queuing, wafer handling, reticle exchange, alignment initialization, illumination changes, and finalization steps.

WET (Wafer Expose Time). At the wafer level, wafer expose time (WET) is defined as the duration between the start and end of exposure for a single wafer. WET primarily reflects scan speed, dose, number of exposures, and settling times. While WET is often treated as deterministic for a given process configuration, in practice it can exhibit variability due to control actions, alignment behavior, or subtle interactions with upstream measurement results.

WST (Wafer Swap Time). Wafer swap time (WST) captures the interval between the end of exposure of one wafer and the start of exposure of the next. It includes a rich set of actions such as chuck exchange (CEX), reticle alignment, preparation steps, and, in some regimes, waiting time for the measure side to complete. Because WST aggregates many heterogeneous actions and conditional branches, it is a significant contributor to inter-wafer delays. Importantly, WST is highly sensitive to the temporal pattern of events occurring during a wafer's life.

Scanner monitoring and event logs. Modern lithography scanners are composed of multiple subsystems, each of which continuously reports low-level system events as structured event codes. These codes capture the full sequence of operations performed across the scanner's internal components during each wafer's life. Each event is associated with a timestamp, making the combined log a fine-grained temporal trace of scanner behavior, covering actions such as wafer loading, measurement steps, alignment, chuck exchange, and exposure. Event logs from different scanner subsystems are reported and stored centrally in the PMA (Performance Monitoring and Analysis) database on the cloud, where they can be accessed directly by ASML engineers without depending on customer-side data sharing. This persistent, machine-side availability makes event logs a reliable and scalable foundation for productivity diagnostics.

Unexplained and unquantified productivity losses. Despite the availability of well-defined timing metrics such as WET, WST, and LOH, customers could experience RE losses that cannot be fully explained using existing abstractions. In some cases, these losses are known qualitatively, but cannot be quantified at a pattern or root-cause level. In other cases, the losses remain entirely unexplained. A key difficulty is that many of these losses arise from complex interactions between events across the wafer life, which are not captured by aggregate timing statistics. Consequently, there is a need for methods that can operate directly on raw event logs, diagnose the presence of timing-relevant behavioral patterns, and quantitatively attribute their impact. Such quantified insights can then be provided to system engineers, enabling corrective actions at the system or control-design level.

1.1.3 Current industry practices for productivity analysis

In industrial practice, productivity analysis relies heavily on expert knowledge and predefined abstractions of scanner behavior.

Dependence on customer-side data sources. Some analyses rely on data that originates from the customer side, such as BLT data, which captures lot-level scheduling and processing context and must be explicitly provided by the customer rather than being available on ASML's own infrastructure. While such data can enrich analysis, it is not always accessible and introduces a dependency on customer side data. In contrast, scanner event codes are continuously captured and stored in the PMA cloud infrastructure, making them persistently available regardless of customer configuration. A purely event-code-based analysis therefore offers a more reliable and scalable foundation for productivity diagnostics.

Event code analysis based on known patterns. Scanner logs typically encode low-level actions as event codes, which are analyzed using rule-based methods to detect known inefficiencies or abnormal behaviors. These analyses are effective for diagnosing previously observed issues but are inherently limited to patterns that are already understood and explicitly modeled.

Manual root cause analysis by domain experts. When unexplained productivity losses occur, domain experts manually correlate event code sequences with timing traces, wafer-level histories, and additional contextual data such as machine settings and process parameters to infer potential root causes. This analysis is difficult to scale and highly dependent on expert experience.

Time-difference methods for pattern impact estimation. To quantify the impact of certain behaviors, direct statistical methods are commonly applied at an aggregate level. By comparing average WET, WST, or LOH between populations with and without a specific condition, practitioners estimate the contribution of that condition to productivity loss. While useful for high-level reporting, these methods cannot attribute impact at the level of individual or a certain collection of wafer sequences and struggle when multiple overlapping patterns are present.

Gaps in current practice. While current industry practices are effective for monitoring known issues, they offer limited support for systematically diagnosing unknown or weakly understood sources of productivity loss. In particular, existing approaches lack the ability to (i) operate solely on event logs without requiring manual feature engineering and other data sources, (ii) quantify the timing impact of overlapping or interacting event patterns at the individual wafer level, and (iii) provide consistent population-level rankings of pattern importance. Addressing these limitations is essential for moving from reactive analysis toward proactive, system-level productivity optimization, which constitutes the ultimate goal of this research.

1.1.4 Current research

Recent academic work has explored several data-driven paradigms relevant to event log analysis and manufacturing productivity, though important gaps remain. A comprehensive review is provided in Chapter 2; this section summarises the key parts.

Pattern mining in event logs. Sequential pattern mining and process mining have been widely studied for discovering recurring structures in event logs [14]. These methods focus on identifying frequent or discriminative patterns, often with constraints on gaps or ordering, or on reconstructing process models for conformance checking. However, they are typically disconnected from quantitative timing outcomes and therefore cannot directly explain productivity losses.

Bridging pattern mining and machine learning. Recent work has sought to connect mined patterns with supervised learning by encoding discovered patterns as input features for downstream models [18]. While this brings pattern information into a predictive framework, the pipeline remains not end-to-end: pattern discovery, feature engineering, and model training are decoupled, and predictive performance is sensitive to manual feature selection choices.

Sequence-to-target machine learning. End-to-end sequence models offer a more direct path from raw event sequences to timing predictions. Recurrent architectures such as LSTMs [29] and transformer-based models [26] have been applied to sequential prediction tasks in manufacturing and time-series domains. Transformers are particularly relevant because their self-attention mechanism explicitly captures long-range dependencies and positional relationships between events.

Model interpretability and quantification. Once a predictive model is trained, a separate post-hoc step can be applied to quantify the contribution of individual patterns. SHAP [11] estimates feature contributions via game-theoretic Shapley values but can be computationally expensive and unstable for high-dimensional token embeddings. Attention-weight analysis [24] is lightweight but attention weights are not always faithful indicators of feature importance [17]. Replacing specific input tokens and measuring the resulting prediction change offers a conceptually simpler alternative that directly measures causal impact, though it also requires multiple forward passes per pattern.

Research motivation. Together, these limitations motivate the need for an end-to-end approach that directly consumes event sequences, predicts timing metrics, and quantitatively attributes losses to event patterns without intermediate manual abstraction.

1.1.5 Research opportunities

The gaps identified in Sections 1.1.3 and 1.1.4 are two sides of the same coin. On the industrial side, current practice is manual, rule-based, and reactive: it depends on expert intuition, cannot scale to unknown patterns, and provides no quantitative attribution at the individual-sequence level. On the research side, transformer-based sequence models and post-hoc attribution methods have matured significantly, yet they have not been combined into a pipeline specifically designed for industrial event log analysis and timing quantification. In other words, the tools that could address the industrial gaps exist in the research literature, but the bridge between them has not been built.

This creates a concrete opportunity: by applying modern machine learning methods to scanner event logs, it becomes possible to more automatically, quickly, and accurately discover the causes of productivity loss than current industrial practice allows. Three research objectives motivate this thesis:

End-to-end timing prediction from event sequences. Build a model that accurately predicts timing metrics directly from raw event sequences, without manual feature engineering, providing a learned foundation for downstream pattern quantification.

Accurate and stable pattern contribution quantification. Combine a trained model with a post-hoc attribution method to quantify the timing impact of discovered event patterns at both the individual sequence level and the global population level, and evaluate how well different transformer architectures support this task.

Industrial validation and practical utility. Apply the pipeline to real industrial scanner event logs and validate its ability to identify productivity-relevant patterns that align with domain expert knowledge, as well as patterns that are new to experts.

Addressing these objectives would enable a new class of productivity analytics tools that combine the scalability of data-driven methods with the transparency required for industrial adoption, directly supporting faster root cause identification and more targeted productivity improvements.

1.2 Research Questions

As discussed in the background, current industrial productivity analysis lacks scalable methods for quantifying the timing impact of event sequence patterns, particularly at the level of individual sequences. While pattern discovery itself has been studied extensively, this thesis focuses primarily on the problem of pattern contribution quantification, assuming that candidate patterns are either known or obtained through simple, heuristic mining procedures.

Main Research Question. Can a transformer-based model, trained directly on machine event log sequences, accurately predict scalar process outcomes and support reliable quantification of event sequence pattern contributions at both individual sequence and population levels?

RQ1: Prediction Performance. Can a transformer-based model, trained on raw event code sequences from machine event logs, accurately predict scalar process outcomes? The methodology is designed to be applicable to any machine that produces discrete event logs paired with a measurable process outcome. As a primary application and evaluation context, we use lithography scanner timing metrics (FLOH, BLOH, WET, and WST), which serve as a representative and industrially relevant instantiation of this broader problem.

RQ2: Transformer Architecture and Pattern Contribution Quantification. When a transformer-based timing prediction model is combined with a post-hoc token replacement method (where the tokens of a target pattern are replaced with non-causal filler tokens and the resulting drop in predicted timing is used as the pattern's contribution estimate [19]), can this pipeline achieve accurate and stable quantification of event sequence pattern contributions at both individual sequence and global levels?

RQ3: Practical Pattern Validation. When applied to real industrial scanner data, can the proposed approach identify pattern contributions that align with known productivity issues, and highlight additional patterns considered relevant by domain experts?

1.3 Approach

The proposed approach operates in two parallel branches on a dataset of manufacturing event logs.

The first branch trains a transformer encoder model to predict timing metrics (FLOH, BLOH, WET, WST) directly from raw event sequences. Two architectures are compared: an Attention Pooling Based model [12], which uses a learned attention layer to aggregate token representations, and a CLS Token Based model [8], which prepends a special `[CLS]` token that accumulates global sequence information across all encoder layers. Both are compared against a naive time-difference baseline that represents the current industrial practice.

The second branch applies sequential pattern mining to the same event sequences to discover frequent and non-redundant behavioral patterns.

Once both branches are complete, a token replacement quantification procedure is applied: for each pattern occurrence in a sequence, the corresponding event tokens are replaced with non-causal filler tokens (event codes drawn from the vocabulary that belong to no known pattern and are therefore assumed not to contribute to timing), and the change in the model's prediction directly measures the pattern's causal contribution to the timing metric. This procedure enables quantification both at the individual sequence level (per-sequence root cause diagnosis) and at the global population level (ranking patterns by their average contribution to the timing metric across

all sequences in which they occur).

Controlled experiments on synthetic data with known ground-truth pattern impacts are used to benchmark the methods, followed by validation on real industrial scanner data with domain expert review.

1.4 Contributions

The main scientific contributions of this thesis are:

1. **Transformer-based prediction of scalar process outcomes from machine event sequences (RQ1).** We demonstrate that transformer encoder models, trained directly on raw machine event code sequences, can accurately predict scalar process outcomes without manual feature engineering, providing the predictive foundation required for downstream pattern quantification. The methodology is general: it applies to any machine that produces discrete event logs paired with a measurable process outcome. We evaluate it on lithography scanner timing metrics (FLOH, BLOH, WET, WST) as a concrete, industrially relevant example, but the approach is not limited to timing or to scanners.

2. **End-to-end quantification pipeline and empirical architecture comparison (RQ2).** We combine the trained prediction model with a post-hoc token replacement procedure to form a complete pattern quantification pipeline. Using a synthetic dataset constructed to mirror real scanner logs—sequences of discrete event codes paired with a scalar timing label, with target times generated as a controlled additive combination of pattern contributions and noise, so that exact ground-truth impacts are known—we systematically compare Attention Pooling Based and CLS Token Based aggregation. Results show that the CLS Token Based method consistently achieves more accurate pattern quantification as data complexity increases.

3. **Industrial validation and deployment (RQ3).** We demonstrate that the pipeline identifies pattern contributions consistent with domain expert knowledge on real scanner data, and deploy it as an interactive Spotfire dashboard supporting engineers in productivity root cause analysis.

1.5 Thesis Overview

The remainder of this thesis is organised as follows. Chapter 2 reviews related work on pattern mining, sequence-to-target machine learning, and model interpretability. Chapter 3 formalises the problem and defines the quantification objectives. Chapter 4 presents the Naive baseline and the two transformer-based methods. Chapter 5 evaluates all methods on a synthetic dataset with controlled ground-truth. Chapter 6 presents validation on real industrial scanner data. Chapter 7 concludes with a summary of findings and directions for future work.

Chapter 2

Related Work

This chapter reviews prior work related to event log analysis, pattern discovery, sequential machine learning, and model interpretability. The discussion is organised along four themes: pattern mining in event logs (Section 2.1), bridging mined patterns with machine learning (Section 2.2), end-to-end sequence-to-target models (Section 2.3), and quantification and interpretability of sequence models (Section 2.4).

2.1 Pattern Mining in Event Logs

Pattern mining is the process of discovering interesting, useful, or unexpected structures within symbolic databases. In manufacturing and system logs, this is commonly applied to event sequences generated by machines to identify regularities or root causes of malfunctions [21][10].

2.1.1 Sequential pattern mining

Sequential pattern mining (SPM) focuses on finding frequent subsequences in a database where the ordering of items is preserved [14]. Traditional algorithms like GSP [25], SPADE [28], and PrefixSpan [16] were designed for market basket analysis but have been adapted for complex industrial event logs.

A key challenge in SPM is the exponential growth of the pattern search space: as the minimum support threshold decreases, the number of discovered patterns can grow exponentially, making both mining and interpretation intractable [14]. To address this, modern algorithms focus on discovering concise representations rather than enumerating all patterns. Closed sequential patterns (e.g., CM-ClaSP [13]) retain only patterns not subsumed by a longer pattern with identical support, yielding a lossless but compact result set. Maximal sequential patterns (e.g., VMSP [15]) go further by retaining only patterns with no frequent super-sequence, producing the smallest result set at the cost of losing support information for sub-patterns.

A further important advancement is the integration of gap constraints, which restrict how far apart consecutive events in a pattern may occur within a sequence [25]. This directly reduces the search space by pruning candidate patterns whose item occurrences are too distant, lowering both runtime and the number of reported patterns to a practically useful set [14].

2.1.2 Process mining

Process mining bridges the gap between data science and business process management by extracting models from event logs [27][30]. It involves four primary types: discovery (creating a model from logs), conformance (comparing logs to an existing model), enhancement (improving a model), and operational support (real-time prediction) [4][27]. Algorithms like the Inductive Miner [20] are favored for their ability to guarantee sound models that are robust against noise. However, when applied to the highly flexible and unstructured logs of lithography scanners, these methods often result in models that are too complex to be interpreted or used for precise timing quantification [30].

2.1.3 Summary

Overall, pattern mining techniques are effective for discovering candidate patterns in event logs. In this thesis, these methods are treated as upstream tools for pattern discovery. The research focus lies not in proposing new mining algorithms, but in quantifying the timing impact of the identified patterns.

2.2 Pattern Mining for Feature Selection

The methods reviewed in Section 2.1 discover patterns based on their frequency of occurrence, without reference to any target outcome. Frequency alone, however, does not indicate whether a pattern is relevant to timing: a pattern may occur often yet have no measurable effect on the metric of interest. A natural extension is therefore to select patterns based on their association with a target variable.

2.2.1 Target-guided pattern discovery

Several classical approaches adopt this perspective. Contrast set mining [3] searches for attribute-value combinations whose support differs significantly between predefined conditions or groups. Emerging pattern mining [9] focuses on patterns whose frequency changes substantially from one class to another, making them strong discriminators between outcomes. These and related methods form a broader family of supervised descriptive pattern mining whose common goal is to find interpretable structures with different distributions across target classes [22].

A shared limitation of these approaches is that they are designed for classification settings: the target variable must be categorical, and continuous targets such as timing metrics typically require discretization before discovery can proceed. This constraint is significant for manufacturing applications where timing is inherently continuous and suitable thresholds are difficult to define.

2.2.2 Using patterns as features for machine learning

Once patterns are identified, a common next step is to encode them as features for a supervised model. Mined patterns are typically represented as binary presence-or-absence vectors and passed to classifiers or regressors such as Random Forests or Gradient Boosting. Kadıoğlu et al. [18] provide a recent instance of this bridge through Seq2Pat, which combines sequential pattern

mining with supervised machine learning via Dichotomic Pattern Mining (DPM): patterns that uniquely distinguish between outcome classes are automatically selected and encoded as features for downstream models.

2.2.3 Summary

This decoupled pattern mining and model training pipeline has two key weaknesses. First, pattern selection is not informed by the downstream model: a pattern may appear discriminative in isolation but become redundant when other patterns are also present. Second, because discovery and learning are separate stages, the pipeline cannot be jointly optimised for predictive performance. These limitations motivate the end-to-end sequence-to-target approaches reviewed in Section 2.3.

2.3 Sequence-to-Target Machine Learning

The limitations of decoupled mine-then-classify pipelines motivate end-to-end sequence-to-target learning, in which a model directly maps a raw event sequence to a scalar or vector output. This avoids intermediate feature engineering and allows the model to learn its own internal representations of pattern structure. Two families of architectures dominate this space: recurrent models and transformer-based models.

2.3.1 Recurrent models

Recurrent Neural Networks (RNNs) and their variants, such as Long Short-Term Memory (LSTM), have been the traditional standard for sequential data [29]. LSTMs utilize specialized gate units to regulate memory cell states, allowing them to handle long-range dependencies better than vanilla RNNs. Despite their success, these models suffer from vanishing and exploding gradient problems during backpropagation through time (BPTT), which limits their ability to learn dependencies in long sequences [1]. Furthermore, their sequential nature prevents them from utilizing the parallel processing capabilities of modern hardware like GPUs.

2.3.2 Transformer-based models

The Transformer architecture has become a dominant paradigm for sequence modeling by replacing recurrence with self-attention mechanisms [26]. This allows the model to correlate information from all elements in a sequence in parallel, significantly improving training efficiency and long-range alignment [29]. In a transformer, the term *token* refers to a single discrete input unit—in the context of this thesis, one event code—that is embedded into a dense vector and processed as one element of the sequence. To handle the quadratic time complexity of standard self-attention, specialised variants like Informer utilize a ProbSparse self-attention mechanism, making it suitable for long-sequence time-series forecasting [31]. Transformers have shown potential in capturing complex dependencies in lithography scanner logs, though they require sophisticated pooling or decoding strategies to map sequence-level information to timing targets [27].

2.3.3 Summary

Transformers are the dominant and most widely adopted architecture for sequence modeling, with successful applications across natural language processing [26], time-series forecasting [31], and process mining [27]. This thesis therefore adopts the transformer as the sequence encoder. The transformer offers two natural aggregation strategies: attention-based pooling and CLS-token aggregation, that form the basis of the two methods compared in the experiments.

2.4 Quantifying and Interpreting Sequence Models

Accurate prediction alone is insufficient for industrial adoption; models must also support quantification and explanation of their behavior. This section reviews four families of approaches: post-hoc explanation methods, attention-based interpretability, mechanistic interpretability, and unified prediction-and-explanation frameworks.

2.4.1 Post-hoc explanation methods

Post-hoc methods interpret black-box models after training by perturbing inputs and observing output changes [19]. Shapley Additive Explanations (SHAP), based on cooperative game theory, is a widely used framework that assigns relevance scores to individual features based on their influence on a model's prediction [11]. For textual or sequential data, variants like TransSHAP and TimeSHAP have been developed to handle subword tokens and multidimensional perturbations [19]. Post-hoc methods are often computationally expensive as they require multiple forward passes for each instance [6].

2.4.2 Attention-based interpretability

Attention weights are often intuitively considered as indicators of feature importance, as they show which tokens a model focuses on when generating a representation [24] [7]. However, there is a debate in the literature, with some researchers arguing that "attention is not explanation" [17]. Studies have shown that learned attention weights are frequently uncorrelated with gradient-based importance measures and can be permuted without altering model outputs. Conversely, newer work suggests that with proper training or when interpreted through a causal lens, attention can indeed provide meaningful insights into a model's decision-making process [24]. In our initial experiments on both synthetic and real manufacturing data, raw attention weights yielded unreliable attribution results; therefore, this approach was not adopted.

2.4.3 Mechanistic interpretability

Mechanistic interpretability (MI) is an emerging field that seeks to reverse-engineer neural networks by decomposing their computations into human-understandable algorithms and circuits [23]. Instead of relying on correlations, MI focuses on the internal computational pathways, such as residual streams and specific attention heads, that transform input sequences into final predictions. Techniques like Layer-wise Relevance Propagation (LRP) attempt to decompose decisions back through

the layers to the input, though standard LRP often faces numerical instabilities in Transformer architectures [5].

2.4.4 Unified prediction and explanation

To address the inefficiencies of post-hoc methods, newer frameworks like ShapTST attempt to unify prediction and explanation [6]. These models incorporate Shapley-based pre-training that trains the model to predict using partial information while simultaneously estimating the contributions of missing elements. This enables the model to generate quantitative explanations alongside its predictions in a single forward pass, offering a highly efficient solution for real-time industrial diagnostics where both speed and reliability are critical.

2.4.5 Summary

Each of the above approaches trades off between faithfulness, computational cost, and implementation complexity. Attention-based methods are lightweight but their faithfulness as explanations is disputed [17], and in our initial experiments on both synthetic and real data they yielded unreliable attributions. SHAP-based methods are theoretically grounded [11] but computationally expensive and can be unstable on high-dimensional token embeddings [19]. Mechanistic interpretability methods offer fine-grained insight into internal model computations [23], but techniques such as LRP face numerical instabilities in Transformer architectures [5] and require substantial implementation effort to apply reliably to a new domain. Unified prediction-and-explanation frameworks such as ShapTST [6] are promising but require specialised training objectives that would prevent reuse of standard pre-trained models. In this thesis, we adopt a token replacement strategy: the tokens of a target pattern are replaced with non-causal filler tokens in the original sequence, and the resulting change in the model's prediction directly quantifies that pattern's contribution. This approach is post-hoc, requires no modification to the training procedure, and directly measures causal impact in units of the timing metric, making it straightforward to interpret for industrial practitioners.

2.5 Summary and Positioning

The review above reveals that while substantial work exists on pattern mining, sequence modeling, and model interpretability, no prior work combines all three into an end-to-end pipeline specifically designed for quantifying timing impact in manufacturing event logs. Existing approaches either focus on pattern discovery without connecting to quantitative timing outcomes, or use interpretability methods not well-suited to sequential pattern attribution.

This thesis positions itself at this intersection: adopting established transformer architectures (attention-based pooling and CLS-token aggregation) for timing prediction, and a post-hoc token replacement strategy for quantification, as described in Chapter 4. The remainder of this thesis focuses on their empirical evaluation.

Chapter 3

Problem Formulation

This chapter formalizes the research problem addressed in this thesis. Section 3.1 defines the task at a high level, and Section 3.2 gives the formal mathematical definitions of the key concepts and quantification objectives.

3.1 Task Definition

Given a dataset of event sequences from a manufacturing process, each associated with a timing metric (e.g., FLOH, BLOH, WET, WST), we aim to:

1. **Predict** the timing metric from the event sequence alone.
2. **Quantify** the contribution of each known sequential event pattern at two levels:
    - Global level: the average impact of each pattern across all sequences in which it occurs, enabling pattern ranking and prioritisation.
    - Individual level: the contribution of each pattern occurrence within a specific sequence, enabling per-sequence root cause diagnosis.

The set of candidate patterns is assumed to be provided from an upstream mining procedure. This thesis focuses on quantification rather than pattern discovery.

3.2 Formal Problem Definition

Dataset and event sequences. Let $\mathcal{D} = \{(S_i, t_i)\}_{i=1}^{N}$ be a dataset of N pairs, where $S_i = (e_{i,1}, e_{i,2}, \ldots, e_{i,L_i})$ is an ordered sequence of L_i discrete events drawn from a finite vocabulary $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$ (i.e., $e_{i,x} \in \mathcal{V}$ for all $x = 1, \ldots, L_i$), and $t_i \in R^+$ is the associated timing metric (e.g., wafer swap time).

Sequential patterns. A sequential pattern $P = (p_1, p_2, \ldots, p_k)$ with $p_m \in \mathcal{V}$ is a short ordered sequence of events. We say that P occurs in S_i at positions $(j_1, j_2, \ldots, j_k)$ if $e_{i,j_m} = p_m$ for $m = 1, \ldots, k$, with $j_1 < j_2 < \cdots < j_k$. To accommodate minor interruptions in real event logs, matching may be gap-tolerant: occurrences are accepted as long as the number of gaps (positions

where $j_{m+1} > j_m + 1$) does not exceed a maximum gap count G_c, and each individual gap length ($j_{m+1} - j_m - 1$) does not exceed a maximum gap length G_l. When $G_c = 0$ and $G_l = 0$, patterns must occur as contiguous sub-sequences (i.e., $j_{m+1} = j_m + 1$ for all m). In the real-data experiments (Chapter 6), candidate patterns are discovered automatically using PrefixSpan [16] followed by an overlap-ratio consolidation step: for each pair of patterns where the shorter pattern is a prefix of the longer one, the shorter is removed if the fraction of its occurrences that are subsumed by the longer exceeds a threshold τ. This is preferred over strict closed-pattern pruning, which retains near-duplicate sub-patterns whenever their support counts differ by even one sequence—a common situation in real manufacturing logs. In the synthetic experiments (Chapter 5), the ground-truth causal patterns are predefined and provided directly to the quantification stage.

Prediction objective. Given the dataset $\mathcal{D}$, the prediction task is to train a model $\mathcal{M}$ that maps an event sequence S_i to a predicted timing metric $\hat{t}_i = \mathcal{M}(S_i)$, minimizing the expected prediction error $E[|\, \mathcal{M}(S_i) - t_i \,|]$ over unseen sequences.

Quantification objectives. Given the dataset $\mathcal{D}$ and a set of candidate patterns $\{P_1, \ldots, P_M\}$, the task has two levels:

1. **Global-level quantification:** For each pattern P, estimate the average impact $\hat{\alpha}_P$ on the timing metric across all sequences in which P occurs.

2. **Sequence-level quantification:** For a single sequence S_i containing pattern P at specific positions, estimate the contribution $\hat{\alpha}_{P,i}$ of that occurrence to the predicted time $\hat{t}_i$.

Chapter 4

Methodology

This chapter presents the end-to-end pipeline and the individual methods used to predict timing metrics and quantify the impact of sequential event patterns. Section 4.1 provides a high-level overview of the pipeline. Section 4.2 describes the pattern mining and consolidation procedure. Section 4.3 describes the Naive baseline. Section 4.4 gives a high-level description of the two transformer-based methods and introduces their architectures. Section 4.5 describes the shared transformer backbone common to both. Section 4.6 presents the Attention Pooling Based Method, covering architecture and quantification. Section 4.7 presents the CLS Token Based Method, covering architecture and quantification.

4.1 End-to-End Pipeline Overview

The proposed pipeline takes a dataset of manufacturing event logs as input and produces two outputs: a ranking of event sequence patterns by their population-level timing impact, and per-sequence pattern contribution estimates that support individual root-cause diagnosis. As illustrated in Figure 4.1, the workflow proceeds along two parallel branches that converge at the pattern quantification stage.

Data preprocessing. Manufacturing event logs are first processed into structured event sequences suitable for modeling and analysis. Each sequence represents the ordered events associated with a production instance and is paired with a corresponding timing metric that serves as the prediction target.

Branch 1: Model training and prediction. The event sequences and their associated timing metrics are used to train a transformer-based prediction model. The model learns to map an event sequence to its expected timing outcome. Once trained, the model can generate predictions for unseen sequences, providing a learned representation of how different event configurations influence process timing.

Branch 2: Pattern mining. In parallel, the same event sequences are analysed using a sequential pattern mining procedure to discover frequent and non-redundant event subsequences. A

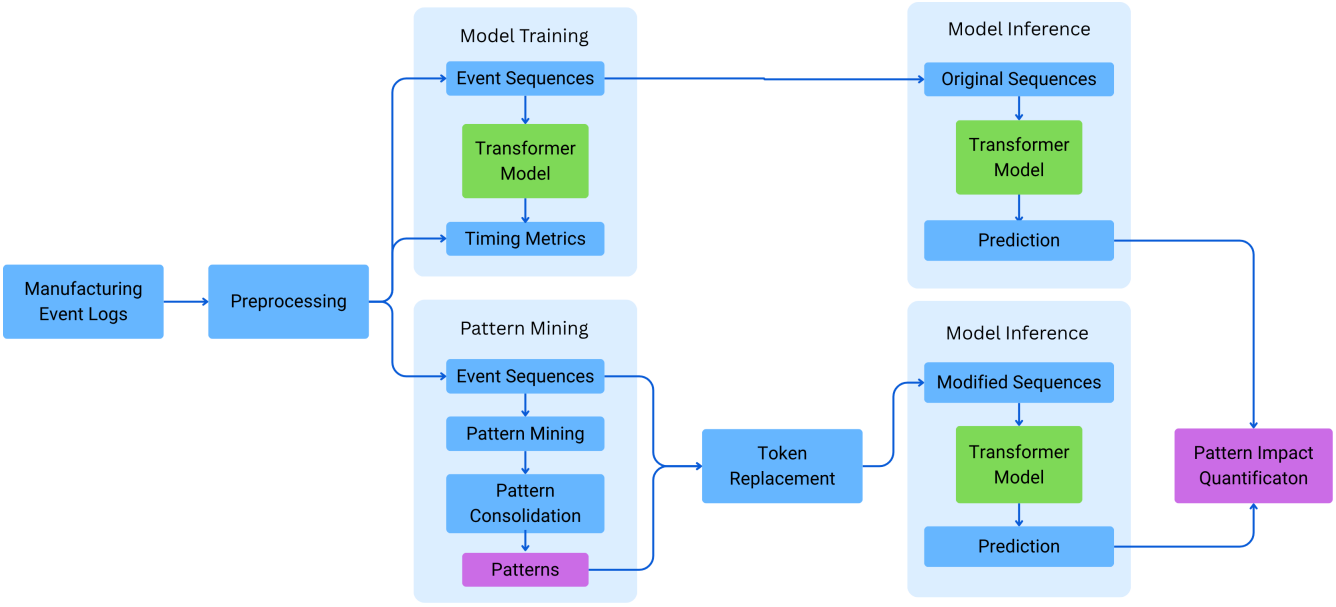


Figure 4.1: End-to-end workflow of the proposed pattern quantification framework. Manufacturing event logs are first preprocessed into event sequences. The upper branch trains a transformer-based prediction model, while the lower branch performs sequential pattern mining to extract relevant event patterns. During quantification, original and token-modified sequences are evaluated by the trained model, and the difference in predictions is used to estimate the impact of each pattern on the timing metric.

consolidation step is then applied to reduce redundancy and produce a compact set of representative patterns that capture common behavioral structures in the process.

Convergence: Pattern impact quantification. The trained prediction model and the mined pattern set are combined through a token replacement procedure. For each occurrence of a pattern within a sequence, the corresponding event tokens are replaced with non-causal filler tokens to generate a modified sequence. Both the original and modified sequences are evaluated by the trained transformer model. The change in the predicted timing metric reflects the contribution of that pattern occurrence.

This process produces two complementary outputs:

- **Sequence-level pattern contributions:** for each sequence, the method estimates how much each detected pattern contributes to the predicted timing, enabling fine-grained root cause analysis at the individual instance level.
- **Global-level pattern contributions:** contributions are aggregated across all occurrences of each pattern to obtain a population-level estimate of its impact, allowing patterns to be ranked according to their overall influence on the timing metric.

4.2 Pattern Mining and Consolidation

The pipeline requires a set of candidate patterns as input to the quantification stage. A variety of pattern mining methods from the literature can serve this role, including GSP [25], SPADE [28], CM-ClaSP [13], and VMSP [15], each offering different trade-offs between result compactness, support counting, and runtime (see Section 2.1). In this thesis we adopt PrefixSpan [16] as the mining algorithm. It grows patterns incrementally by projecting the database onto suffix sub-databases, avoiding the costly candidate generation step of earlier algorithms.

4.2.1 PrefixSpan Mining

PrefixSpan is applied with a minimum support threshold, i.e., the number of distinct sequences in which the pattern must occur, and configurable minimum and maximum pattern lengths. Gap-tolerant matching is used: a pattern $P = (p_1, \ldots, p_k)$ is present in a sequence if its events appear in order with other events permitted between them. This is appropriate for scanner logs where key operational events are interleaved with auxiliary events and relative ordering carries more meaning than strict adjacency.

4.2.2 Overlap-Ratio Consolidation

PrefixSpan can return many overlapping candidates, including short patterns that are near-prefixes of longer, more specific ones. Strict closed-pattern pruning (which removes a pattern only if a super-pattern with *identical* support exists) is too conservative in practice: if pattern A→B occurs in 1,001 sequences and A→B→C in 1,000, both are retained as closed, despite being functionally near-identical.

Instead, a threshold-based consolidation step is applied after mining. Let $\mathcal{S}_P$ denote the set of sequences containing pattern P. For each pair of patterns where the shorter is a prefix of the longer, the overlap ratio is computed:

$$r(\text{shorter}, \text{longer}) = \frac{|\mathcal{S}_{\text{shorter}} \cap \mathcal{S}_{\text{longer}}|}{|\mathcal{S}_{\text{shorter}}|}, \tag{4.1}$$

i.e., the fraction of the shorter pattern's occurrences that are subsumed by the longer. If $r \geq \tau$ (with $\tau = 0.98$ in this thesis), the shorter pattern is merged into the longer and removed. Patterns are processed in order of increasing length so that consolidation proceeds greedily from shortest candidates upward.

This ensures the attribution stage operates on a compact, non-redundant library, and prevents shorter prefix patterns from absorbing contribution that belongs to the longer, operationally more specific pattern.

4.3 Baseline: Naive Time Difference Method

The naive baseline is a simple, model-free estimator that reflects the time-difference approach commonly used in practice (see Section 1.1.3). For each pattern P, the dataset $\mathcal{D}$ is partitioned

into sequences that contain P and those that do not. The estimated global impact is the difference in mean target times:

$$\hat{\alpha}_P^{\text{naive}} = \frac{1}{|\mathcal{D}_P^+|} \sum_{i \in \mathcal{D}_P^+} t_i \; - \; \frac{1}{|\mathcal{D}_P^-|} \sum_{i \in \mathcal{D}_P^-} t_i, \tag{4.2}$$

where $\mathcal{D}_P^+ = \{i \in \{1, \ldots, N\} : P \subseteq S_i\}$ and $\mathcal{D}_P^- = \{i \in \{1, \ldots, N\} : P \not\subseteq S_i\}$ are index sets.

The method requires no model training, is fast to compute, and is easily interpretable. But it provides no sequence-level decomposition: the method produces a single global estimate per pattern and cannot estimate per-sequence pattern contributions. And it also provides no modeling of interactions, positional effects, or pattern overlap: the impact of pattern P is estimated independently of other patterns and its temporal position.

4.4 Overview of Transformer-Based Methods

Both the Attention Pooling Based and CLS Token Based methods are transformer encoder models that take a raw event sequence as input and produce a scalar timing prediction. They differ in how token-level representations are aggregated into a single sequence-level vector, and consequently in how pattern contributions are extracted during quantification.

Figure 4.2 illustrates both architectures side by side. Both methods share the same encoder backbone and prediction head; they differ only in the aggregation block between the encoder output and the head, highlighted in grey. The following sections first describe the shared backbone, then detail each method's aggregation strategy and quantification procedure.

4.5 Shared Transformer Architecture

Both the Attention Pooling Based and CLS Token Based methods share a common transformer encoder backbone. Given an input sequence $S = (e_1, \ldots, e_L)$:

Token embedding. Each event e_j is mapped to a dense vector $\mathbf{x}_j = \mathbf{E}[e_j] \in R^d$ via a learned embedding table $\mathbf{E} \in R^{|\mathcal{V}| \times d}$, where d is the embedding dimension and the padding token (index 0) maps to the zero vector.

Positional encoding. A learned positional embedding $\mathbf{p}_j \in R^d$ is added to each token embedding:

$$\mathbf{h}_j^{(0)} = \mathbf{x}_j + \mathbf{p}_j, \quad j = 1, \ldots, L. \tag{4.3}$$

Transformer encoder. The sequence of embeddings $\mathbf{H}^{(0)} = (\mathbf{h}_1^{(0)}, \ldots, \mathbf{h}_L^{(0)})$ is processed by a stack of N_L standard transformer encoder layers. Each layer applies multi-head self-attention followed by a position-wise feed-forward network, both with residual connections and layer normalisation:

$$\mathbf{A}^{(\ell)} = \text{MultiHeadAttn}(\mathbf{H}^{(\ell-1)}, \mathbf{H}^{(\ell-1)}, \mathbf{H}^{(\ell-1)}; \mathbf{M}), \tag{4.4}$$

$$\tilde{\mathbf{H}}^{(\ell)} = \text{LayerNorm}(\mathbf{H}^{(\ell-1)} + \mathbf{A}^{(\ell)}), \tag{4.5}$$

$$\mathbf{H}^{(\ell)} = \text{LayerNorm}\Big(\text{FFN}(\tilde{\mathbf{H}}^{(\ell)}) + \tilde{\mathbf{H}}^{(\ell)}\Big), \tag{4.6}$$

where $\ell = 1, \ldots, N_L$ indexes the layer, and the three equations correspond to the three sub-operations within each layer:

1. **Multi-head self-attention.** $\text{MultiHeadAttn}(\mathbf{H}^{(\ell-1)}, \mathbf{H}^{(\ell-1)}, \mathbf{H}^{(\ell-1)}; \mathbf{M})$ computes self-attention by using the previous layer's output $\mathbf{H}^{(\ell-1)}$ as queries, keys, and values. Each token can attend to every other token in the sequence, allowing the model to capture contextual dependencies at varying ranges. Multiple attention heads learn different relational patterns in parallel, and their outputs are concatenated and projected to form $\mathbf{A}^{(\ell)}$. The binary mask $\mathbf{M}$ prevents attention to padding positions by setting their logits to $-\infty$ before the softmax, so they contribute zero weight.

2. **First residual connection and layer normalisation.** The attention output $\mathbf{A}^{(\ell)}$ is added back to the input $\mathbf{H}^{(\ell-1)}$ via a residual connection, and the sum is normalised to produce the intermediate representation $\tilde{\mathbf{H}}^{(\ell)}$. The residual connection preserves the original token information and mitigates vanishing gradients during training, while layer normalisation stabilises the magnitude of activations across the sequence dimension.

3. **Feed-forward network with second residual connection and layer normalisation.** A two-layer position-wise feed-forward network $\text{FFN}(\mathbf{h}) = \mathbf{W}_2\,\text{ReLU}(\mathbf{W}_1\mathbf{h} + \mathbf{b}_1) + \mathbf{b}_2$ is applied independently to each token representation in $\tilde{\mathbf{H}}^{(\ell)}$. This introduces non-linear transformations that allow the model to learn more complex, token-level feature interactions beyond what self-attention alone captures. A second residual connection and layer normalisation are applied to yield $\mathbf{H}^{(\ell)}$, the output of layer ℓ.

Prediction head. Both methods aggregate the encoder output into a single vector $\mathbf{z} \in R^d$ (the aggregation strategy differs per method, as described below) and pass it through a two-layer MLP with dropout:

$$\hat{t} = \mathbf{W}_2\,\text{ReLU}(\text{Dropout}(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1)) + b_2, \tag{4.7}$$

where $\mathbf{W}_1 \in R^{(d/2)\times d}$ and $\mathbf{b}_1 \in R^{d/2}$ are the weight matrix and bias of the first linear layer, and $\mathbf{W}_2 \in R^{1\times(d/2)}$ and $b_2 \in R$ are the weight and bias of the output layer.

All models are trained by minimizing the mean squared error (MSE) loss with early stopping on a validation set. Method-specific training hyperparameters are detailed in Chapter 5.

4.6 Attention Pooling Based Method

4.6.1 Architecture

The Attention Pooling Based method uses the shared transformer encoder with a learned attention pooling mechanism to aggregate token-level representations into a single sequence-level vector. The choice of attention pooling is motivated by recent work showing that the pooling strategy has a significant impact on the downstream performance of transformer-based models [12].

Given the encoder output $\mathbf{H}^{(N_L)} = (\mathbf{h}_1, \ldots, \mathbf{h}_L)$ with $\mathbf{h}_j \in R^d$, the attention pooling layer computes a scalar logit u_j, a normalised attention weight α_j, and a pooled representation $\mathbf{z}$:

$$u_j = \mathbf{w}_u^\top \tanh(\mathbf{W}_a \mathbf{h}_j + \mathbf{b}_a) + b_u, \quad j = 1, \ldots, L, \tag{4.8}$$

$$\alpha_j = \frac{\exp(u_j)}{\sum_{k=1}^{L} \exp(u_k)}, \tag{4.9}$$

$$\mathbf{z} = \sum_{j=1}^{L} \alpha_j \, \mathbf{h}_j \in R^d, \tag{4.10}$$

where $\mathbf{W}_a \in R^{(d/2)\times d}$, $\mathbf{b}_a \in R^{d/2}$, $\mathbf{w}_u \in R^{d/2}$, and $b_u \in R$ are learnable parameters, and d is the embedding dimension (Section 4.5). Padding positions are masked before the softmax so that $\sum_{j=1}^{L} \alpha_j = 1$ over non-padding tokens. The pooled vector $\mathbf{z}$ is passed through the shared prediction head to produce $\hat{t}$.

The key property is that the attention weights α_j are directly interpretable: they indicate the relative importance of each token to the prediction. Tokens belonging to high-impact patterns are expected to receive higher attention.

4.6.2 Quantification via Token Replacement

The Attention Pooling Based method uses token replacement as the primary quantification method at both sequence and global levels. Let $f : \mathcal{V}^* \to R^+$ denote the trained transformer model that maps an event sequence to its predicted timing. The core idea is to replace the tokens belonging to a pattern with non-causal filler tokens and measure the resulting change in the model's prediction. For each occurrence of pattern $P = (p_1, \ldots, p_k)$ in sequence S_i at positions $\mathcal{P} = \{j_1, j_2, \ldots, j_k\}$ (with $j_1 < j_2 < \cdots < j_k$, as defined in Section 3.2):

1. Compute $\hat{t}_i = f(S_i)$, the model's prediction on the original sequence.
2. Construct a modified sequence S_i' by replacing tokens at positions $\mathcal{P}$ with non-causal filler tokens drawn randomly from the vocabulary, excluding all known pattern tokens. When the overall filler pool is insufficient (i.e., fewer non-causal tokens exist than needed), the method falls back to drawing from the full vocabulary.
3. Compute $\hat{t}_i' = f(S_i')$, the prediction on the modified sequence.
4. The raw per-sequence contribution is $\Delta_P^{(i)} = \hat{t}_i - \hat{t}_i'$.

To reduce variance from random filler token selection, the replacement is repeated with K different filler token assignments and the predictions are averaged (K=5 by default). The choice of K=5 is empirically validated in Chapter 5.

Attention redistribution correction. The raw difference $\Delta_P^{(i)} = \hat{t}_i - \hat{t}_i'$ is not the exact pattern impact in the Attention Pooling Based method, because the softmax pooling weights must always sum to 1. When the replaced filler tokens receive near-zero pooling attention, the remaining tokens' weights inflate to compensate, making the non-pattern tokens contribute more to $\hat{t}_i'$ than they did to

$\hat{t}_i$. This pushes $\hat{t}'_i$ upward relative to a true "pattern-absent" counterfactual, so $\Delta_P^{(i)}$ systematically underestimates the true impact. In the approximation that filler positions receive negligible pooling attention, the remaining weights scale by $1/(1-\beta_P)$, where $\beta_P = \sum_{j \in \mathcal{P}} \alpha_j$ is the pooling attention share of the replaced positions on the original sequence. We correct for this effect:

$$\hat{\Delta}_P^{(i)} = \frac{\Delta_P^{(i)}}{1-\beta_P}, \quad \text{if } 0 < \beta_P < 0.95. \tag{4.11}$$

The corrected sequence-level estimate is:

$$\hat{\alpha}_{P,i}^{\text{M1}} = \hat{\Delta}_P^{(i)}, \tag{4.12}$$

and the global impact is the mean across all occurrences:

$$\hat{\alpha}_P^{\text{M1}} = \frac{1}{|\mathcal{D}_P^+|} \sum_{i \in \mathcal{D}_P^+} \hat{\Delta}_P^{(i)}. \tag{4.13}$$

4.7 CLS Token Based Method

4.7.1 Architecture

The CLS Token Based method follows a BERT-style approach [8]. A blank `[CLS]` token is prepended to every input sequence, producing $S' = (\texttt{[CLS]}, e_1, \ldots, e_L)$. The `[CLS]` token participates in self-attention across all transformer layers, allowing it to aggregate information from the entire sequence. After N_L layers of encoding, the representation of `[CLS]` at the final layer serves as the sequence-level summary:

$$\mathbf{z} = \mathbf{h}_{\texttt{[CLS]}}^{(N_L)}. \tag{4.14}$$

This vector is passed through the prediction head to produce $\hat{t}$.

Unlike the Attention Pooling Based method, there is no explicit attention pooling layer; the `[CLS]` token learns to aggregate relevant information through the multi-head self-attention mechanism over multiple layers.

4.7.2 Quantification via Token Replacement

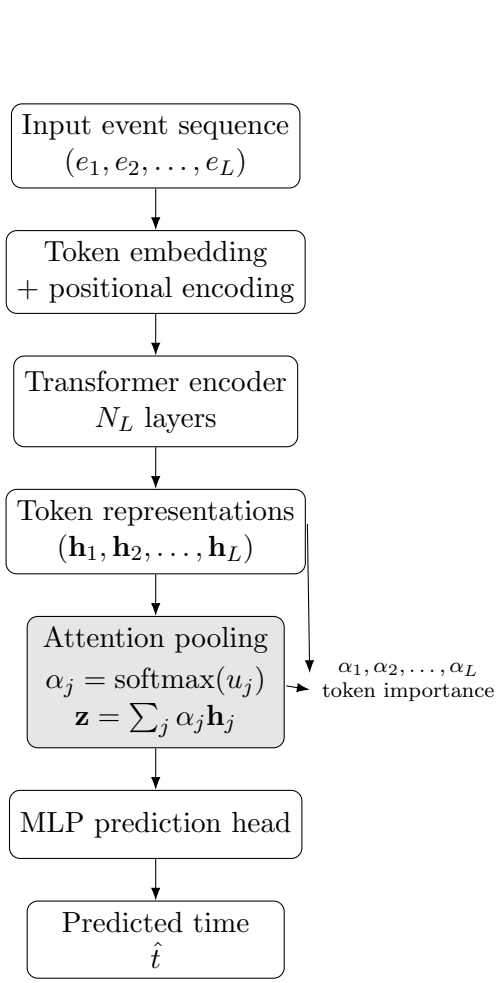
The CLS Token Based method uses the same token replacement procedure as the Attention Pooling Based method for pattern quantification (see Section 4.6). For each pattern occurrence in sequence S_i, the pattern tokens are replaced with non-causal filler tokens and the sequence-level contribution is:

$$\hat{\alpha}_{P,i}^{\text{M2}} = \Delta_P^{(i)} = \hat{t}_i - \hat{t}'_i, \tag{4.15}$$

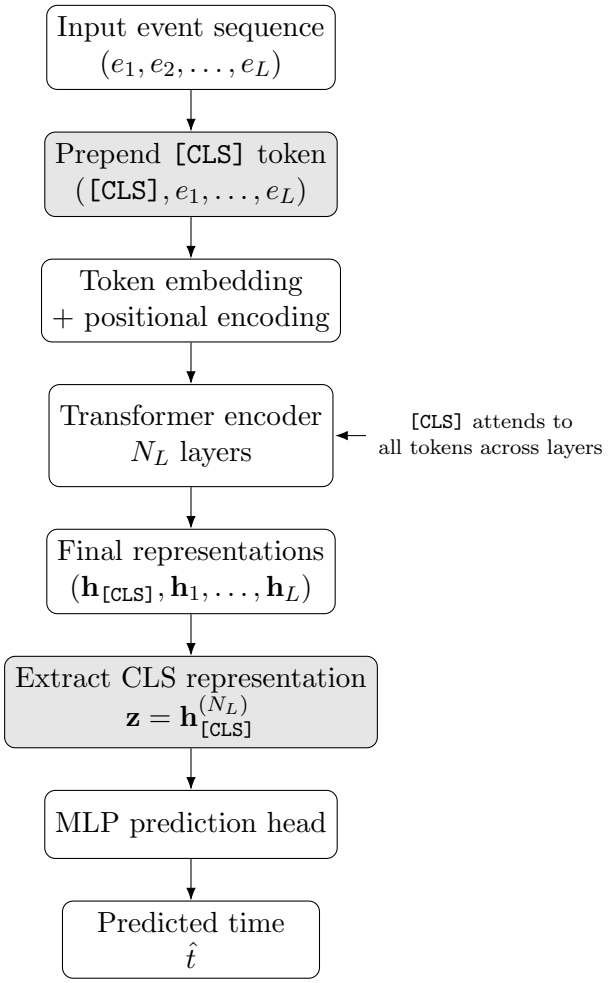
and the global impact is the mean over all occurrences:

$$\hat{\alpha}_P^{\text{M2}} = \frac{1}{|\mathcal{D}_P^+|} \sum_{i \in \mathcal{D}_P^+} \Delta_P^{(i)}. \tag{4.16}$$

Because the CLS architecture has no explicit softmax pooling layer over event tokens, the attention redistribution correction (Equation 4.11) is not needed.



(a) Attention Pooling Based method. After the shared encoder, a learned attention layer produces per-token scalar weights α_j and aggregates them into a single vector $\mathbf{z} = \sum_j \alpha_j \mathbf{h}_j$. During quantification, the same weights inform the attention redistribution correction.



(b) CLS Token Based method. A prepended `[CLS]` token accumulates global sequence information through multi-head self-attention across all N_L encoder layers. Its final-layer representation $\mathbf{h}^{(N_L)}_{\texttt{[CLS]}}$ is passed directly to the prediction head; no explicit pooling layer is required.

Figure 4.2: Architectures of the two transformer-based methods. Both share the same encoder backbone (token embedding, positional encoding, and N_L transformer encoder layers) and MLP prediction head. They differ only in the aggregation strategy between the encoder output and the prediction head, highlighted in grey.

Chapter 5

Synthetic Dataset Experiments

In this chapter, we evaluate the three methods—Naive baseline, Attention Pooling Based, and CLS Token Based—on fully synthetic data where ground-truth pattern impacts are known exactly. Evaluating pattern impact quantification methods requires ground-truth knowledge of which patterns are causal and what their true impacts are. Real manufacturing event logs lack such ground truth: pattern impacts are unknown and can only be approximated via expert knowledge. To resolve this, we construct a synthetic dataset with 15 known causal patterns, each assigned a true base impact. The target time for each sequence is computed as an additive combination of pattern contributions plus controllable perturbations, providing exact ground truth for every sequence and every pattern. This controlled setting allows us to measure quantification accuracy directly and to isolate the effect of individual data-complexity factors through a systematic controlled factor experiment.

5.1 Evaluation Metrics

The three methods are compared along three dimensions.

Prediction performance. For the Attention Pooling Based and CLS Token Based methods, regression accuracy is measured by R^2, MAE, and RMSE on a held-out test set. The naive baseline does not produce per-sequence predictions and is excluded from this evaluation.

Global-level quantification accuracy. For each pattern, the estimated global impact $\hat{\alpha}_P$ is compared against the ground-truth mean impact α_P^*. Metrics include: mean absolute error (MAE) across all patterns and mean absolute rank error (the average displacement of each pattern's estimated rank relative to its true rank).

Computational complexity. End-to-end wall-clock runtime is reported, including model training and pattern quantification.

5.2 Experimental Setup

5.2.1 Factors to Mirror Real Event Logs

The synthetic data generation is designed to mirror key challenges observed in real semiconductor event log analysis. Real semiconductor event logs are inherently noisy, high-dimensional, and structurally complex: individual sequences can span hundreds of events, event patterns that causally contribute to the timing metric (hereafter *causal patterns*) are interspersed with large volumes of irrelevant filler activity, and the relationship between event occurrences and cycle time is rarely additive or position-independent. Furthermore, real manufacturing processes exhibit interaction effects between co-occurring fault patterns and contain overlapping sub-sequences that confound attribution when patterns are mined and scored independently.

A purely empirical evaluation on real data would make it impossible to verify attribution correctness, since ground-truth per-pattern contributions are never directly observable in production logs. Synthetic data with known causal structure provides the only principled way to measure how accurately a method recovers true pattern impacts. To ensure that conclusions drawn from synthetic experiments transfer meaningfully to the real setting, we designed several experimental factors that reproduce the statistical properties most commonly encountered in real-world conditions. Each experimental factor corresponds to a specific real-world phenomenon:

- **Target noise (random noise on the target time):** Real event logs contain timing variability from system jitter, measurement error, and background processes not captured in the event log. We model this as additive Gaussian noise on the target time, controlled by the standard deviation σ.
- **Interaction:** In real manufacturing, co-occurring patterns can have non-additive effects. For example, a calibration event followed by a specific error under a particular machine configuration may cause amplified overhead beyond the sum of individual impacts.
- **Position:** The same event pattern occurring early vs. late in a process sequence may have different impacts. An error at the start of wafer processing typically causes more cumulative overhead than the same error near the end.
- **Overlap:** In real scanner logs, distinct operational routines often share common event subsequences. For example, two different calibration procedures may begin with the same initialization steps. When both a short pattern (e.g., A→B) and a longer pattern containing it (e.g., A→B→C) are mined and quantified separately, the shorter pattern can absorb contribution that should belong to the longer one, distorting impact estimates for both.

5.2.2 Data Generation

Vocabulary and patterns. The synthetic vocabulary contains 100 event types (`E001` through `E100`). Events `E001`–`E060` form the causal event pool from which 15 non-overlapping causal patterns are constructed. Events `E061`–`E100` serve as filler (non-causal) events. Pattern lengths range from 2 to 5 events, with a mix of lengths: three 2-event, four 3-event, three 4-event, and three 5-event patterns. Causal events are exclusive to one pattern.

Sequence construction. Each sequence is generated by: (1) sampling a target length uniformly from 20 to 60 events; (2) for each of the 15 patterns, independently deciding inclusion based on the pattern's inclusion probability (a Bernoulli draw); (3) placing included patterns at random non-overlapping positions using a greedy longest-first strategy; and (4) filling remaining positions with random filler events.

Target time generation. The target time for each sequence follows:

$$t = t_{\text{base}} + \sum_{p \in \mathcal{P}_{\text{present}}} f(p, \text{position}, \text{interaction}, \text{overlap}) + \epsilon \tag{5.1}$$

where $t_{\text{base}} \sim \mathcal{N}(100, 10)$ is a stochastic base time; $\mathcal{P}_{\text{present}} \subseteq \{1, \ldots, 15\}$ is the set of causal patterns included in the sequence (sampled independently per pattern from its base frequency); $f(p, \text{position}, \text{interaction}, \text{overlap})$ is the contribution of pattern p, defined as its base impact α_p scaled by the active experimental modifiers (position weight w, interaction boost b, and overlap co-occurrence bias); and $\epsilon \sim \mathcal{N}(0, \sigma)$ is Gaussian noise. Pattern contributions are attributed using greedy longest-first assignment, consistent with the analysis procedure.

Core configuration. Table 5.1 summarizes the baseline configuration shared by all tiers.

Table 5.1: Baseline (reference) configuration for synthetic data generation. All experimental tiers share this configuration unless explicitly overridden by one factor.

Parameter	Default Value
Number of event types	100
Causal event pool	60 events (`E001`–`E060`)
Filler event pool	40 events (`E061`–`E100`)
Number of causal patterns	15
Sequences per tier	2,000
Sequence length range	20–60 events
Base time (s)	$\mathcal{N}(100, 10)$
Pattern impact range	10–80 s
Pattern inclusion probability range	0.10–0.25
Target noise σ	5.0 s
Train / test split	1,600 / 400 (80%/20%)

5.2.3 Ground-Truth Causal Patterns

Table 5.2 lists all 15 causal patterns used across all tiers. Each pattern's base impact α_p is drawn independently and uniformly from the impact range $[10, 80]$ s, and its inclusion probability is drawn uniformly from the frequency range $[0.10, 0.25]$; both are fixed by the random seed and shared across all tiers, enabling direct cross-tier comparison.

Table 5.2: Ground-truth causal patterns used in all synthetic tiers, ranked by base impact. The pattern index (P1–P15) is used as shorthand in subsequent tables. Inclusion probability is the per-sequence Bernoulli probability with which the pattern is placed, drawn uniformly from the frequency range [0.10, 0.25] at dataset construction time; the expected number of sequences containing the pattern is inclusion probability × 2,000.

Index	Rank	Pattern	Length	Base Impact (s)	Inclusion Prob.
P1	1	E022 → E051 → E005 → E030	4	85.67	0.133
P2	2	E055 → E012	2	76.65	0.182
P3	3	E013 → E036	2	61.52	0.123
P4	4	E010 → E037 → E032	3	60.45	0.211
P5	5	E044 → E007 → E057	3	59.22	0.115
P6	6	E014 → E047 → E058 → E003	4	54.57	0.214
P7	7	E019 → E029 → E040 → E027 → E049	5	45.91	0.143
P8	8	E039 → E024	2	43.31	0.251
P9	9	E043 → E033 → E054	3	40.02	0.157
P10	10	E025 → E001 → E045 → E026 → E053	5	35.65	0.171
P11	11	E011 → E060 → E023 → E042	4	25.80	0.128
P12	12	E017 → E021 → E056	3	25.14	0.118
P13	13	E028 → E038 → E006 → E035 → E052	5	23.98	0.181
P14	14	E059 → E034 → E020	3	16.90	0.228
P15	15	E046 → E031 → E050 → E004	4	14.34	0.254

5.2.4 Factor Controlled Experiment

We adopt a single-factor controlled experiment with two selected tiers per factor (target noise, position, interaction, and overlap). Each tier modifies exactly one factor from the default configuration. Together with the baseline and two combined tiers (combined-medium and combined-hard), this produces a total of 11 experimental conditions.

Factor definitions.

- **Target noise** (σ): Standard deviation of Gaussian noise added to the target time. Selected tier settings: 15, 25 (default: 5).
- **Position** (w): Position-dependent impact scaling. Patterns appearing earlier in a sequence receive higher impact: impact $\times$ $[1 + w \cdot (1 - \text{relative_position})]$. Selected tier settings: $w = 0.4, 0.8$ (default: $w = 0$).
- **Interaction** (b): Non-additive co-occurrence boost. When two interacting patterns co-occur, each pattern's impact is multiplied by $(1 + b)$. Selected tier settings: 4 pairs / b=0.6, 7 pairs / b=1.2 (default: none).
- **Overlap**: Fraction of pattern pairs that share a common event prefix, creating nested-like structures (e.g., patterns A→B and A→B→C→D share the prefix A→B). When one partner is already included in a sequence, the other's inclusion probability is boosted. Selected tier settings: 40%, 80% (default: 0%).

Table 5.3 summarizes the selected factor-tier design used in this chapter.

Table 5.3: Multi-factor controlled experiment: 4 factors with selected tiers. Each tier modifies exactly one factor from the default configuration.

Factor	Parameter	Default	Tier 1	Tier 2
Target noise	σ	5	15	25
Position	w	0	0.4	0.8
Interaction	pairs / boost	0 / 0	4 / 0.6	7 / 1.2
Overlap	shared ratio / boost	0% / 0	40% / 0.4	80% / 0.8

The combined-medium tier activates all factors at medium settings simultaneously (σ=15, w=0.4, 4 interaction pairs with b=0.6, 40% overlap), serving as an intermediate stress setting between single-factor tiers and the hardest case.

The combined-hard tier activates all factors at their maximum tier settings simultaneously (σ=25, w=0.8, 7 interaction pairs with b=1.2, 80% overlap), serving as an end-to-end stress test.

5.2.5 Implementation Details

The model architecture and training settings for both methods are summarized in Table 5.4. Key hyperparameter choices and the ranges explored during development are reported in Table 5.5.

Table 5.4: Model architecture and training configuration for synthetic experiments, showing settings for both methods.

Parameter	CLS Token Based	Attention Pooling Based
Embedding dimension	128	128
Transformer layers	2	2
Attention heads	4	4
Feed-forward dimension	256	256
Dropout	0.1	0.1
Max sequence length	100	100
Batch size	32	32
Learning rate	$\mathbf{5 \times 10^{-4}}$	$\mathbf{1 \times 10^{-3}}$
Optimizer	**AdamW** (λ=10^{-4})	**Adam** (λ=10^{-5})
Epochs (max)	50	50
Warmup epochs	**5**	**0**
Early stopping patience	10	10
Gradient clipping	**1.0**	**None**
Pattern contribution mode	replacement	replacement

The embedding dimension (128) and transformer depth (2 layers, 4 heads) were chosen to provide sufficient representational capacity for the 100-event vocabulary and sequences of length 20–60, while remaining small enough to train quickly on synthetic data. Smaller values (64-dim, 1 layer) underfit the pattern structure, while larger values (256-dim, 4 layers) show no improvement and increased training time. The feed-forward dimension is fixed at twice the embedding dimension (256), following standard practice. The two methods differ in their optimizer settings: The CLS Token Based method uses AdamW with a warmup schedule and gradient clipping, as the CLS token aggregation benefits from more stable early training; the Attention Pooling Based method

uses Adam without warmup, as the explicit attention pooling layer provides a strong learning signal from the start. The filler token count is fixed at K=5 for all experiments, balancing quantification stability against runtime cost.

Table 5.5: Justification of key hyperparameter choices for synthetic experiments. Each row reports the explored range and the rationale for the selected value.

Hyperparameter	Explored Range	Selected	Rationale
Embedding dim	64, 128, 256	128	64 underfits the 100-token vocabulary; 256 gives no accuracy gain and doubles training time.
Transformer layers	1, 2, 4	2	1 and 2 layers show similar performance on the simple dataset; on the more complex dataset, 2 layers perform slightly better; 4 layers do not improve performance.
Attention heads	4, 8	4	4 heads (32 dims/head) balances expressiveness and parameter count; 8 heads show no benefit.
Learning rate (Attention Pooling Based)	10^{-4}, 5×10^{-4}, 10^{-3}, 5×10^{-3}	10^{-3}	Balances training speed and stability.
Learning rate (CLS Token Based)	10^{-4}, 5×10^{-4}, 10^{-3}, 5×10^{-3}	5×10^{-4}	Balances training speed and stability.
Batch size	16, 32, 64	32	Balances gradient noise and memory; 64 provides no accuracy improvement.
Filler tokens K	1, 3, 5, 10	5	MAE is stable across all tested values; K=5 selected as the minimal value providing a safety margin for real-data correlations.

All experiments use PyTorch with a fixed random seed (42) for reproducibility. The baseline requires no training and computes naive impact directly from the data.

5.3 Results and Analysis

This section presents results across three dimensions: prediction performance (Section 5.3.1), global-level quantification accuracy with per-factor gradient analysis (Sections 5.3.2–5.3.3), and computational complexity (Section 5.3.4). Note that the transformer methods additionally produce per-sequence contribution estimates enabling root-cause diagnosis at the individual sequence level

— a capability the Naive baseline fundamentally cannot provide — which is demonstrated on real data in Chapter 6.

5.3.1 Prediction Performance

Table 5.6 reports prediction performance for the Attention Pooling Based and CLS Token Based methods across all experimental tiers. Accurate prediction is a prerequisite for reliable quantification: if the model cannot predict the target time well, the token replacement cannot meaningfully measure pattern contributions.

Table 5.6: Prediction performance (R^2, MAE, RMSE) for Attention Pooling Based and CLS Token Based methods across all synthetic experimental tiers, evaluated on both the full dataset and the held-out test set.

Tier	Method	R^2 (All)	R^2 (Test)	MAE (s)	RMSE (s)
baseline	Attn Pooling Based	0.9445	0.8808	10.37	14.26
baseline	CLS Token Based	0.9750	0.9445	6.60	9.57
noise_15	Attn Pooling Based	0.9357	0.8129	10.90	15.78
noise_15	CLS Token Based	0.9509	0.8743	9.72	13.79
noise_25	Attn Pooling Based	0.8565	0.7115	18.51	24.79
noise_25	CLS Token Based	0.9152	0.7880	13.49	19.06
position_04	Attn Pooling Based	0.9452	0.8829	12.29	17.23
position_04	CLS Token Based	0.9735	0.9521	8.29	11.98
position_08	Attn Pooling Based	0.9568	0.8994	12.76	18.11
position_08	CLS Token Based	0.9786	0.9565	8.04	12.73
interaction_4_06	Attn Pooling Based	0.9558	0.8858	9.51	14.23
interaction_4_06	CLS Token Based	0.9684	0.9285	7.68	12.03
interaction_7_12	Attn Pooling Based	0.8791	0.6812	19.00	30.91
interaction_7_12	CLS Token Based	0.9465	0.8903	10.12	20.57
overlap_04	Attn Pooling Based	0.9685	0.9095	7.61	10.69
overlap_04	CLS Token Based	0.9771	0.9429	6.28	9.12
overlap_08	Attn Pooling Based	0.9597	0.8954	9.05	12.35
overlap_08	CLS Token Based	0.9756	0.9421	6.52	9.60
combined_medium	Attn Pooling Based	0.9187	0.7395	10.67	17.43
combined_medium	CLS Token Based	0.9652	0.9168	7.50	11.15
combined_hard	Attn Pooling Based	0.8898	0.7440	9.59	15.48
combined_hard	CLS Token Based	0.8810	0.7581	8.68	16.09

Analysis. The key message from prediction performance is that the CLS Token Based architecture is more robust to structural complexity, particularly non-linear interaction effects, while both methods remain adequate predictors across all tiers. The CLS advantage is most pronounced under strong interaction (test R^2 gap of 0.890 vs. 0.681 at interaction-7-12), where the CLS token's holistic aggregation captures joint pattern effects that weighted pooling cannot represent. Under position and overlap stress, the gap narrows because self-attention and positional embeddings are sufficient for both architectures to handle these structural variations.

The convergence at combined-hard ($R^2 \approx 0.75$ test for both) reflects a shared difficulty ceiling imposed by simultaneous maximum-intensity confounders, not a method-specific failure. The widening gap between all-data and test-set R^2 under these conditions signals mild overfitting when the signal-to-noise ratio is low. Crucially, even in the hardest settings both methods retain enough predictive signal to support downstream token-replacement quantification, as confirmed in the sections that follow.

5.3.2 Per-Pattern Global Quantification

Having established prediction quality, we now evaluate how accurately each method estimates the global impact of individual patterns. Figures 5.1, 5.2, and 5.3 show per-pattern quantification on the baseline, combined-medium, and combined-hard tiers, respectively: each point represents one of the 15 causal patterns, and proximity to the identity diagonal indicates accurate quantification. Table 5.7 summarises the mean absolute quantification error across all 15 patterns for each method-tier combination, providing an aggregate view.

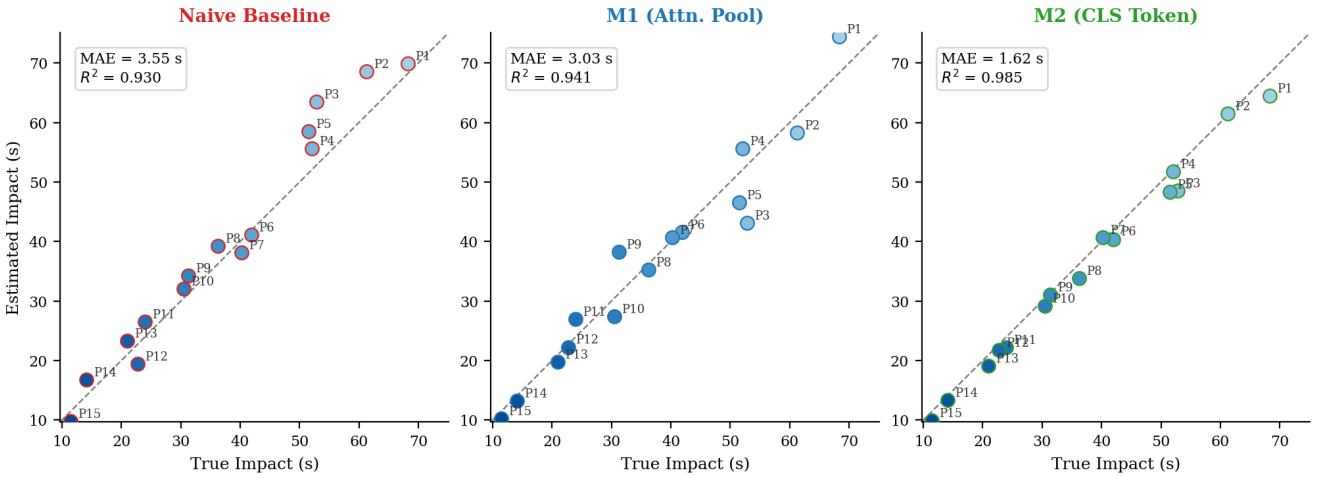


Figure 5.1: Estimated versus ground-truth global pattern impact on the baseline tier. Each point is one of the 15 causal patterns; proximity to the identity diagonal indicates accurate quantification. MAE and R^2 are annotated per method panel.

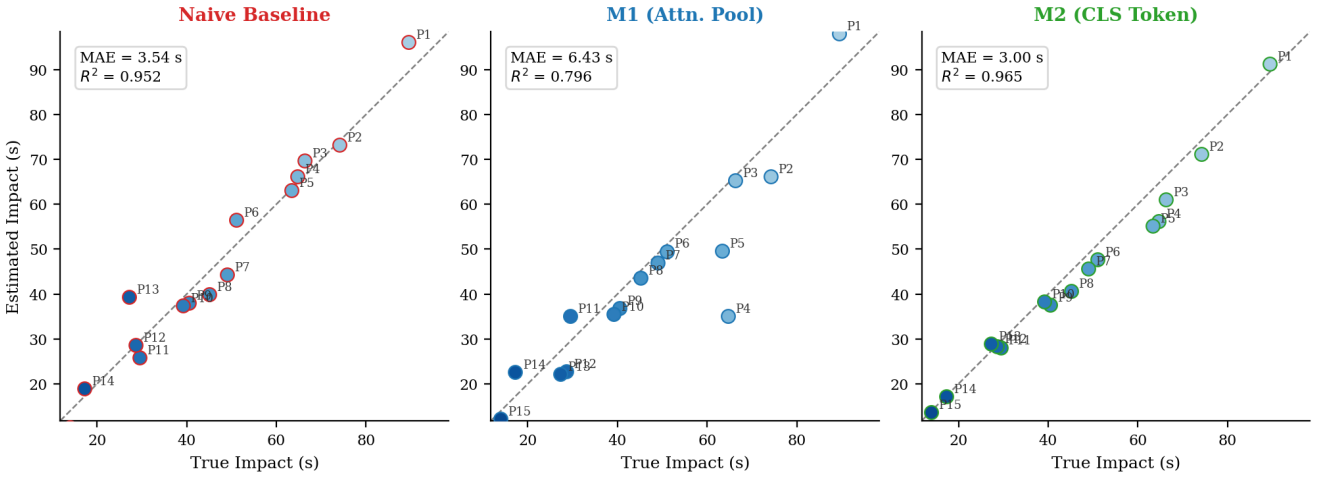


Figure 5.2: Estimated versus ground-truth global pattern impact on the combined-medium tier. Each point is one of the 15 causal patterns. Compared to the baseline (Figure 5.1), increased scatter reflects the combined effect of all four stress factors at medium intensity.

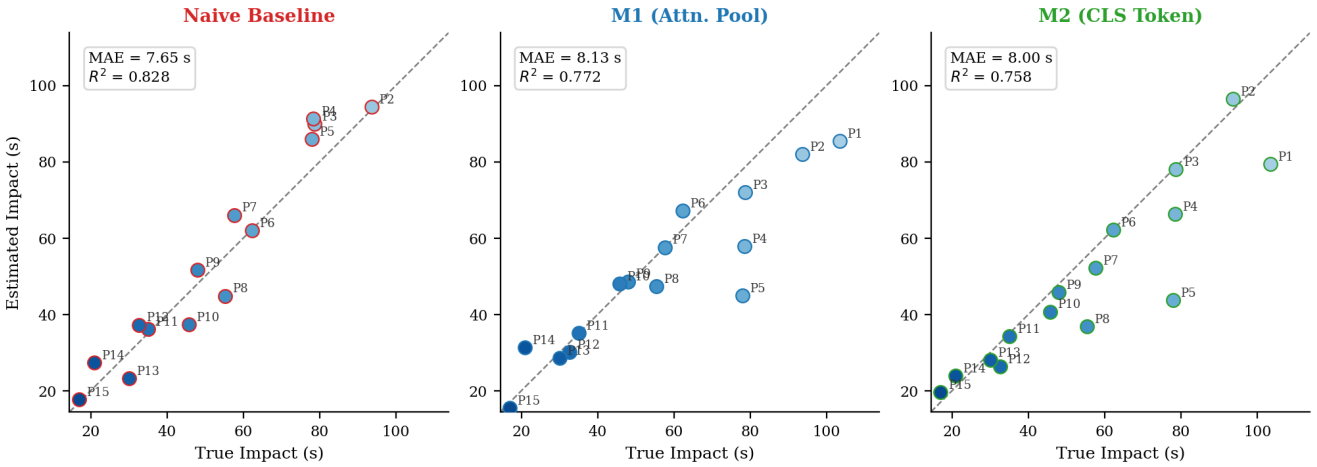


Figure 5.3: Estimated versus ground-truth global pattern impact on the combined-hard tier (all factors at maximum). Increased deviation from the diagonal compared to baseline and combined-medium reflects the cumulative difficulty of simultaneous noise, position, interaction, and overlap effects.

Table 5.7: Mean absolute quantification error (MAE) across all 15 patterns for each method and tier, summarizing the accuracy of global-level impact estimation.

Tier	Method	Mean Quantification Error (s)
baseline	Naive	3.55
baseline	Attn Pooling Based	3.03
baseline	CLS Token Based	**1.62**
noise_15	Naive	3.51
noise_15	Attn Pooling Based	**2.10**
noise_15	CLS Token Based	2.92
noise_25	Naive	**3.46**
noise_25	Attn Pooling Based	3.80
noise_25	CLS Token Based	3.65
position_04	Naive	4.39
position_04	Attn Pooling Based	2.47
position_04	CLS Token Based	**1.72**
position_08	Naive	5.22
position_08	Attn Pooling Based	2.66
position_08	CLS Token Based	**2.19**
interaction_4_06	Naive	4.54
interaction_4_06	Attn Pooling Based	3.31
interaction_4_06	CLS Token Based	**1.75**
interaction_7_12	Naive	10.01
interaction_7_12	Attn Pooling Based	4.23
interaction_7_12	CLS Token Based	**4.20**
overlap_04	Naive	2.23
overlap_04	Attn Pooling Based	3.57
overlap_04	CLS Token Based	**2.08**
overlap_08	Naive	4.77
overlap_08	Attn Pooling Based	5.31
overlap_08	CLS Token Based	**1.87**
combined_medium	Naive	3.54
combined_medium	Attn Pooling Based	6.43
combined_medium	CLS Token Based	**3.00**
combined_hard	Naive	**7.65**
combined_hard	Attn Pooling Based	8.13
combined_hard	CLS Token Based	8.00

Analysis. The fundamental reason the CLS Token Based method outperforms the Naive baseline is that it controls for co-occurrence bias: the Naive estimator cannot subtract the contributions of other patterns that happen to co-occur in the same sequence, so its error is structural and persistent regardless of data cleanliness. The transformer eliminates this bias by learning to isolate each pattern's contribution from its context. This advantage erodes only when heavy confounders corrupt the learned representations.

On the clean baseline (Figure 5.1), this advantage is clear: CLS achieves R^2=0.985 and more than halves the Naive error, while the Attention Pooling Based method lies between the two. As confounders are stacked at medium intensity (Figure 5.2), the Naive error is stable — confirming its bias is structural, not confounder-sensitive — while the Attention Pooling Based method degrades sharply to worse than Naive, driven by a few outlier patterns whose contributions its pooling mechanism cannot isolate under simultaneous stress.

At combined-hard (Figure 5.3), all three methods converge to similar error levels. The Naive method is marginally the most accurate, because maximally stacked confounders corrupt the transformer's learned representations to the point where its co-occurrence advantage is lost. This convergence is not a shared difficulty ceiling: it reflects the Naive method's structural bias and the transformer's confounder sensitivity cancelling each other out under extreme conditions.

5.3.3 Gradient Analysis

To understand how each data-complexity factor affects the three methods, we conduct a gradient analysis: each factor is varied independently from the default configuration while all other factors remain at their default values. This isolates the sensitivity of each method to a specific type of data complexity. For each factor, we report both prediction and quantification metrics. Where informative, per-pattern breakdowns at the strongest tier are included to diagnose which patterns are most affected.

Target Noise Factor

The target noise factor increases the standard deviation (σ) of the Gaussian noise added to the target time, reducing the signal-to-noise ratio while keeping the causal structure of events unchanged. The aggregate trend is reported in Table 5.8, with pattern-level errors at the strongest tier shown in Figure 5.4.

Table 5.8: Gradient results for the target noise factor: prediction and quantification metrics as target noise σ increases from the default (5) to 15 and 25.

Tier	Method	R^2	MAE_{pred} (s)	$\text{MAE}_{\text{impact}}$ (s)	Rank MAE
baseline	Naive			3.55	0.40
baseline	Attn Pooling Based	0.9445	10.37	3.03	0.40
baseline	CLS Token Based	0.9750	6.60	1.62	0.27
noise_15	Naive			3.51	0.40
noise_15	Attn Pooling Based	0.9357	10.90	2.10	0.40
noise_15	CLS Token Based	0.9509	9.72	2.92	0.13
noise_25	Naive			3.46	0.40
noise_25	Attn Pooling Based	0.8565	18.51	3.80	0.53
noise_25	CLS Token Based	0.9152	13.49	3.65	0.53

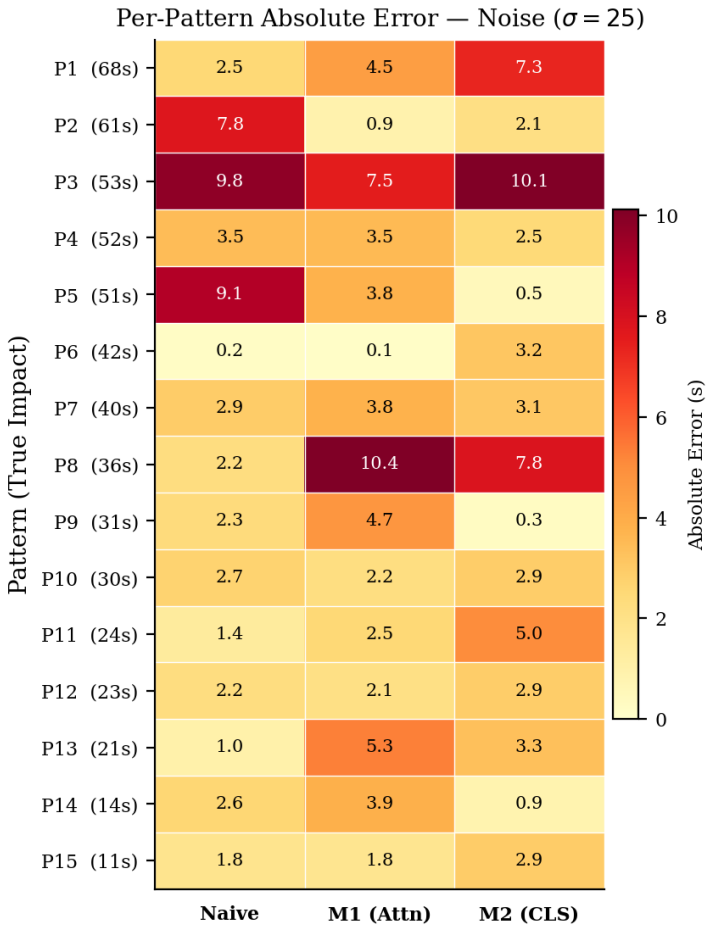


Figure 5.4: Per-pattern absolute quantification error on the noise-25 tier (σ=25). Cell values show $|\hat{\alpha}_P - \alpha_P^*|$ in seconds; column MAE is shown below each method. P3 (inclusion prob. 0.123) shows high errors across all three methods; P5 (inclusion prob. 0.115) shows a high Naive error (9.1 s) but low CLS error (0.5 s), indicating that the low-frequency penalty is partly method-dependent rather than universal.

Analysis. Target noise is the one factor where the Naive method is not disadvantaged relative to the transformer methods, because Gaussian noise averages out in population-level mean comparisons but corrupts the model weights that transformers fit to. The practical implication is that when timing variability is dominated by random jitter rather than structural confounders, the simpler Naive estimator is sufficient and no transformer is needed.

The per-pattern heatmap (Figure 5.4) reveals a nuance: not all low-frequency patterns are equally affected. P3 shows uniformly high errors across all three methods, consistent with the general difficulty of estimating impacts from few observations under high noise. P5, however, shows a high Naive error but near-zero CLS error — suggesting the transformer can extract reliable signal from the few available occurrences of P5, whereas the population-mean estimator cannot. This indicates that low inclusion probability is a sufficient but not necessary condition for high error under noise; the method's ability to use contextual information also matters.

Interaction Factor

The interaction factor introduces non-additive co-occurrence effects between pattern pairs. As the number of interacting pairs and boost strength increase, the naive baseline's additive assumption is violated, while the Attention Pooling Based and CLS Token Based methods can capture non-linear effects. The aggregate trend is shown in Table 5.9, and pattern-level errors at the strongest tier are shown in Figure 5.5.

Table 5.9: Gradient results for the interaction factor: prediction and quantification metrics as the number of interacting pattern pairs and boost strength increase from baseline (none) to 4 pairs/b=0.6 and 7 pairs/b=1.2.

Tier	Method	R^2	MAE_{pred} (s)	$\text{MAE}_{\text{impact}}$ (s)	Rank MAE
baseline	Naive			3.55	0.40
baseline	Attn Pooling Based	0.9445	10.37	3.03	0.40
baseline	CLS Token Based	0.9750	6.60	1.62	0.27
interaction_4_06	Naive			4.54	1.07
interaction_4_06	Attn Pooling Based	0.9558	9.51	3.31	0.53
interaction_4_06	CLS Token Based	0.9684	7.68	1.75	0.67
interaction_7_12	Naive			10.01	0.80
interaction_7_12	Attn Pooling Based	0.8791	19.00	4.23	1.07
interaction_7_12	CLS Token Based	0.9465	10.12	4.20	0.80

Analysis. Interaction effects expose the fundamental weakness of the Naive estimator: its additive assumption is directly violated when co-occurring patterns produce non-linear boosts, and no amount of data can correct for this structural mismatch. Both transformer methods, which can capture joint effects through self-attention, are far more robust. The practical takeaway is that when interaction effects are present, transformer outperforms the naive method: the Naive method's error triples while both transformer methods remain within 2× of their baseline accuracy.

Importantly, the CLS advantage over the Attention Pooling Based method disappears at strong interaction: both converge to similar errors at interaction-7-12. This means CLS's edge lies in handling overlap and combined confounders (shown in subsequent subsections), not in modeling interaction per se. The heatmap (Figure 5.5) shows that residual errors for both transformers are concentrated on P9 and P15 — patterns in the most heavily interacting pairs — where the individual contributions are intrinsically entangled and cannot be isolated by any token-replacement method. P14 is a noteworthy exception: despite being a member of an interacting pair, M1 achieves near-zero error (1.9 s) while CLS does not, suggesting that the specific interaction structure for P14 is better captured by attention pooling than by CLS aggregation.

Position Factor

The position factor introduces position-dependent impact scaling, where patterns appearing earlier in a sequence have proportionally higher impact. This violates the naive baseline's assumption of position-invariant impact. Table 5.10 summarizes the tier-wise trend, while Figure 5.6 provides the strongest-tier pattern-wise error breakdown.

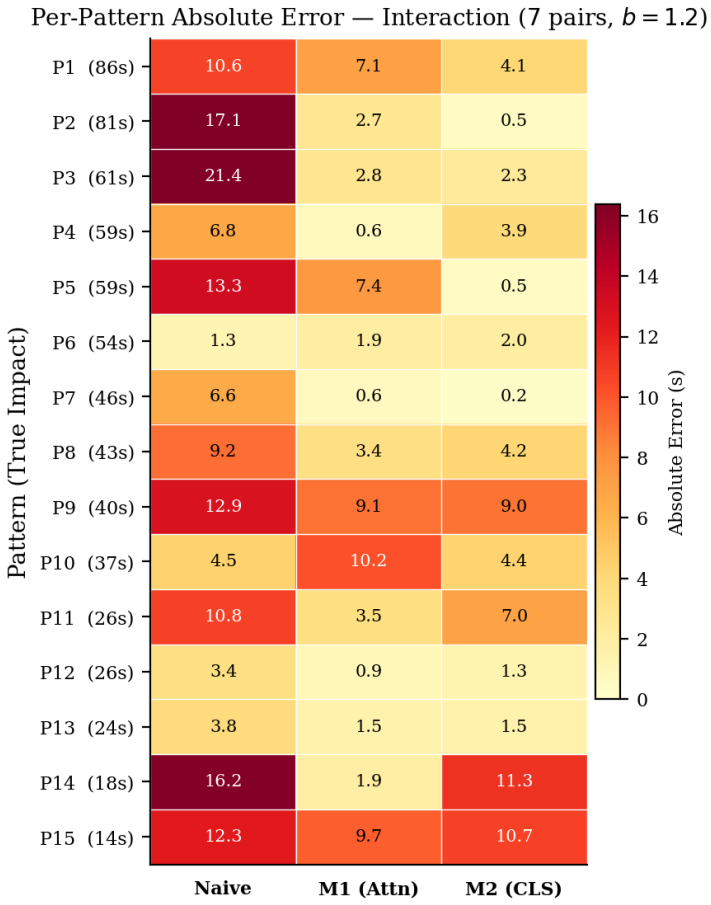


Figure 5.5: Per-pattern absolute quantification error on the interaction-7-12 tier (7 interacting pairs, boost b=1.2). The Naive column is broadly elevated across most patterns, reflecting the additive assumption violation; Attention Pooling Based and CLS Token Based achieve lower errors for most patterns. P9 and P15 remain difficult for all three methods; P14 is high for both Naive (16.2 s) and CLS (11.3 s) but low for M1 (1.9 s), reflecting asymmetric sensitivity to interaction structure.

Table 5.10: Gradient results for the position factor: prediction and quantification metrics as position-dependent scaling weight w increases from 0 (default) to 0.4 and 0.8.

Tier	Method	R^2	MAE_{pred} (s)	$\text{MAE}_{\text{impact}}$ (s)	Rank MAE
baseline	Naive			3.55	0.40
baseline	Attn Pooling Based	0.9445	10.37	3.03	0.40
baseline	CLS Token Based	0.9750	6.60	1.62	0.27
position_04	Naive			4.39	0.27
position_04	Attn Pooling Based	0.9452	12.29	2.47	0.53
position_04	CLS Token Based	0.9735	8.29	1.72	0.13
position_08	Naive			5.22	0.27
position_08	Attn Pooling Based	0.9568	12.76	2.66	0.40
position_08	CLS Token Based	0.9786	8.04	2.19	0.13

Analysis. Position effects demonstrate that self-attention provides implicit positional context sufficient for transformers to handle position-dependent impacts, while the Naive estimator — which treats all pattern occurrences as position-invariant — is structurally unable to do so. Both

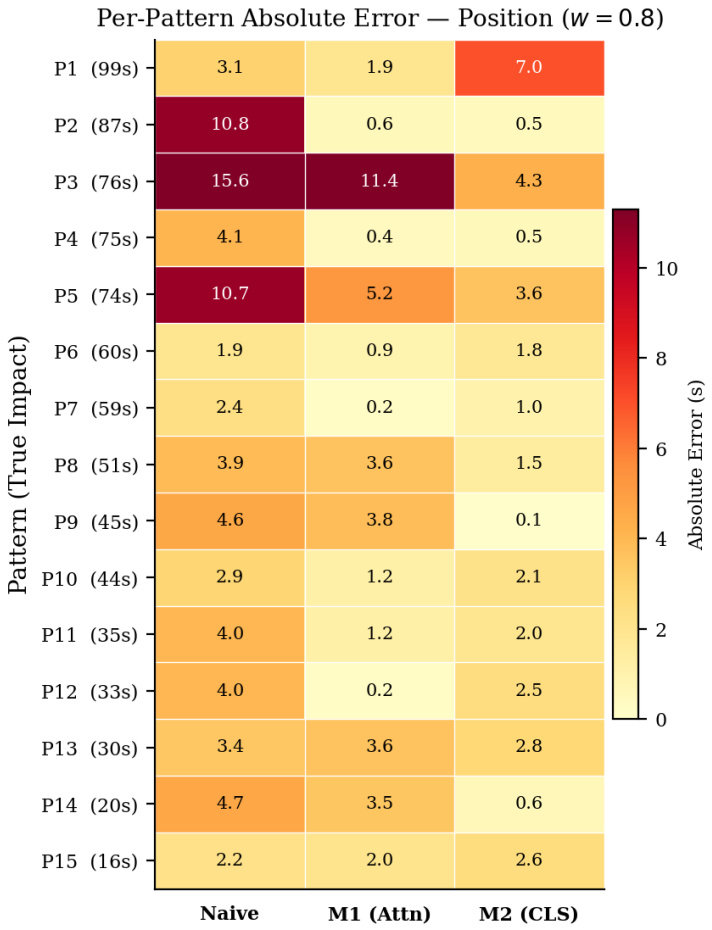


Figure 5.6: Per-pattern absolute quantification error on the position-08 tier (w=0.8 position scaling). The Naive column shows elevated errors for P2 (10.8 s), P3 (15.6 s), and P5 (10.7 s). The CLS Token Based method achieves lower errors for most patterns, though P1 is an exception (7.0 s). The Attention Pooling Based method also struggles on P3 (11.4 s), the highest-impact pattern under position scaling, indicating that attention pooling has difficulty attributing position-amplified contributions for the largest-impact patterns.

transformer methods are robust to position stress at the aggregate level, in contrast to the strong Naive degradation (47% MAE increase to position-08).

However, the aggregate result conceals a pattern-level difference revealed in Figure 5.6: the Attention Pooling Based method fails specifically on P3, the highest-impact pattern in this tier (error: 11.4 s), while CLS Token Based achieves 4.3 s. This suggests that attention pooling struggles to attribute position-amplified contributions for the most impactful patterns, even when its aggregate error appears acceptable. The CLS Token Based method is therefore the safer choice when position-dependent attribution matters, though it too shows a residual exception at P1 (7.0 s).

Overlap Factor

The overlap factor introduces shared-event dependencies between pattern pairs: a fraction of patterns are forced to share a common event prefix, creating nested-like structures (e.g., patterns A→B and A→B→C→D share the A→B prefix). This violates the naive method's independent presence vs. absence assumption, since the presence or absence of one pattern is no longer independent of its overlapping partner. The overall trend is reported in Table 5.11, with pattern-level errors at the strongest tier shown in Figure 5.7.

Table 5.11: Gradient results for the overlap factor: prediction and quantification metrics as the fraction of pattern pairs sharing a common event prefix increases from 0% (default) to 40% and 80%.

Tier	Method	R^2	MAE_{pred} (s)	$\text{MAE}_{\text{impact}}$ (s)	Rank MAE
baseline	Naive			3.55	0.40
baseline	Attn Pooling Based	0.9445	10.37	3.03	0.40
baseline	CLS Token Based	0.9750	6.60	1.62	0.27
overlap_04	Naive			2.23	0.67
overlap_04	Attn Pooling Based	0.9685	7.61	3.57	0.53
overlap_04	CLS Token Based	0.9771	6.28	2.08	0.13
overlap_08	Naive			4.77	1.20
overlap_08	Attn Pooling Based	0.9597	9.05	5.31	0.40
overlap_08	CLS Token Based	0.9756	6.52	1.87	0.13

Analysis. Overlap is the factor that most clearly differentiates the two transformer methods, and the result has a clear mechanistic explanation. The Attention Pooling Based method attributes contributions by weighting individual tokens; when two patterns share a common prefix, those shared tokens are attributed to both patterns simultaneously, causing systematic over- and under-estimation for overlapping pairs. The CLS Token Based method is immune to this: token replacement measures the global prediction drop when an entire pattern is masked, so the attribution is determined by the holistic sequence context rather than by token-level weight sharing. The practical consequence is that CLS Token Based retains near-baseline accuracy at overlap-08 while Attention Pooling Based deteriorates to worse than Naive — making CLS the only viable choice when pattern overlap is present.

The apparent Naive improvement at overlap-04 should not be interpreted as robustness: the rank MAE simultaneously worsens, indicating that the MAE improvement is coincidental (prefix sharing happens to bias mean comparisons favourably for some patterns) rather than a genuine improvement in attribution quality. The per-pattern heatmap (Figure 5.7) provides the mechanism-level evidence: large Attention Pooling Based errors are concentrated exclusively on members of overlapping pairs (P1, P4, P6, P7), while CLS Token Based errors remain low across all patterns.

Summary of Gradient Analysis Findings

Taken together, the gradient analysis yields two clear take-aways. Both transformer methods outperform the Naive baseline when the data contains structural confounders: under interaction effects the Naive error triples (10.0 s) while transformers remain below 4.5 s, and under position-dependent impacts transformers reduce error by roughly 50% relative to Naive. Pure target noise is the exception—it affects all methods equally, leaving no room for a transformer advantage. Between the two transformer architectures, the CLS Token Based method's decisive advantage

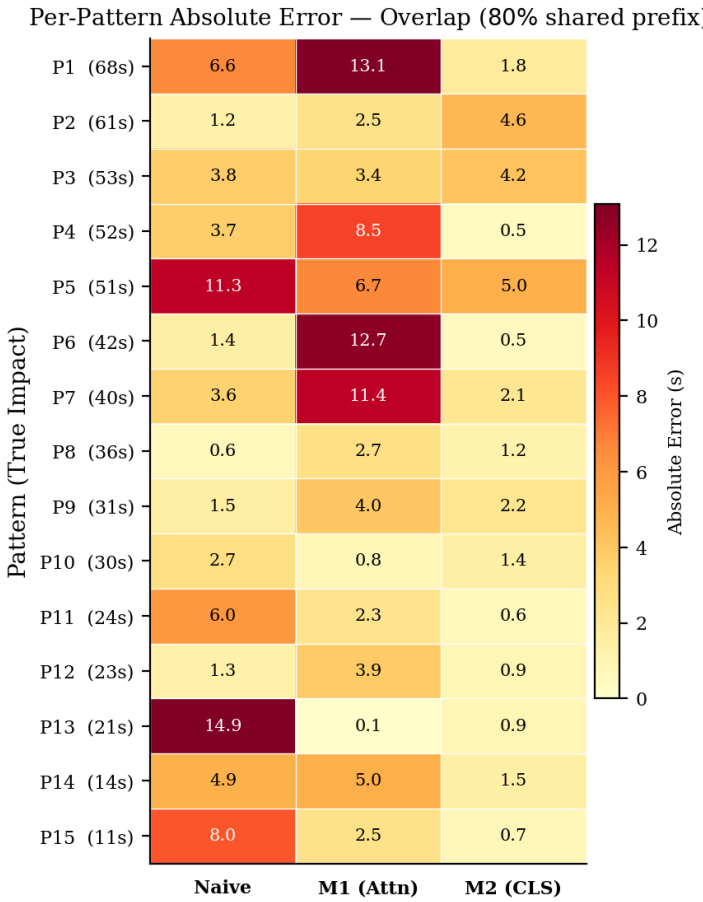


Figure 5.7: Per-pattern absolute quantification error on the overlap-08 tier (80% shared-prefix pairs). The Attention Pooling Based column shows large errors on P1 (13.1 s), P4 (8.5 s), P6 (12.7 s), and P7 (11.4 s), which are members of overlapping pairs. The CLS Token Based method maintains low errors across most patterns. The Naive column shows large errors on P5 (11.3 s), P13 (14.9 s), and P15 (8.0 s), reflecting the prefix-sharing bias that distorts population-level mean comparisons.

arises specifically under pattern overlap (1.87 s vs. 5.31 s at overlap-08, where Attention Pooling degrades below Naive). The mechanism is direct: shared-prefix tokens are double-counted by the pooling weights, while CLS token replacement measures the global prediction drop for a fully masked pattern and is immune to this artefact. Under pure noise or interaction without overlap, the two architectures converge to similar errors, confirming that the CLS advantage is driven by structural overlap, not by interaction non-linearity.

5.3.4 Computational Complexity

Runtime comparison across methods is summarised in Table 5.12. The dominant cost for both methods is the pattern quantification stage, which requires multiple forward passes per sequence (one per pattern occurrence, repeated $K = 5$ times for filler variance reduction). The CLS Token Based method is slower than the Attention Pooling Based method because the CLS token architecture

requires processing the full sequence for each replacement, whereas the Attention Pooling Based method's attention pooling allows more efficient computation.

Table 5.12: End-to-end wall-clock runtime (seconds) per method on 2,000 sequences, including model training and pattern quantification for all 15 patterns.

Method	baseline	combined_hard
Naive	1.91	1.93
Attention Pooling Based	459.28	366.32
CLS Token Based	2498.99	1900.40

The figures in Table 5.12 are measured on 2,000 sequences. To give practitioners a sense of how runtime scales with dataset dimensions, we additionally ran the CLS Token Based method on six configurations crossing three sequence lengths ($\bar{\ell} \approx 50$, 100, 200 events) with two dataset sizes ($N \in \{500, 2{,}000\}$). Increasing N from 500 to 2,000 gives a 4.0× increase at $\bar{\ell}\approx 50$ but only 2.9× at $\bar{\ell}\approx 100$, indicating that training cost (which grows with sequence length faster than linearly) becomes a larger fraction of total runtime for longer sequences and dilutes the linear scaling with N. Increasing sequence length has a super-linear effect: doubling from ≈50 to ≈100 events raises runtime by 3.3× at N=500 and 2.4× at N=2,000.

Table 5.13: End-to-end wall-clock runtime (seconds) of the CLS Token Based method as a function of sequence length and dataset size, including model training and pattern quantification for all 15 patterns.

Sequence Length	N Sequences	Runtime (s)
≈ 50	500	873
≈ 50	2,000	3514
≈ 100	500	2910
≈ 100	2,000	8567
≈ 200	500	7438
≈ 200	2,000	21754

Chapter 6

Real Data Experiments

The synthetic experiments in Chapter 5 established that the CLS Token Based method achieves superior quantification accuracy and prediction performance compared to the Attention Pooling Based method across all difficulty tiers. In this chapter, we therefore apply only the CLS Token Based method to real industrial scanner event logs, alongside the Naive baseline, to evaluate whether these advantages transfer to the challenges of real manufacturing data. A key goal is to validate that the discovered patterns are recognized by domain experts as genuine productivity-relevant behaviors, since no ground truth is available for real data. Section 6.1 describes the full experimental setup, covering data preprocessing, pattern mining, and model training configuration. Section 6.2 presents dataset statistics across the four timing buckets. Section 6.3 presents prediction performance and pattern quantification results with expert review across different timing buckets. Section 6.4 describes the interactive Spotfire dashboard developed to support engineers in exploring the mined patterns in production.

6.1 Experimental Setup

6.1.1 Data Source and Preprocessing

Data Source and Raw Event Log Schema

In this section, we conduct experiments on real industrial data. During production, scanner event logs are reported and stored in Databricks catalogs. Each event record contains the equipment ID, an event timestamp, an event identifier (whose prefix indicates the originating scanner component, followed by the event series and a specific event ID), and developer text that provides additional context for the event.

Table 6.1 shows a representative dummy dataset illustrating the raw event log schema. Each row corresponds to a single event emitted by a scanner, identified by its equipment number and timestamp. The `event_code` field encodes the event type (prefix indicates subsystem), while `developer_text` carries machine-generated context that is later used for event-code refinement via text classification.

Table 6.1: Dummy Event Code Sequence Dataset

equipment_nr	event_dt_clt	event_code	developer_text
AAA	2025-10-14 08:23:45.321	ZZ-0012	–
AAA	2025-10-14 08:25:17.842	ZZ-0013	–
AAA	2025-10-14 08:26:02.113	WZ-2201	–
AAA	2025-10-14 08:28:10.765	ZZ-0012	–
AAA	2025-10-14 08:30:44.589	ZZ-0013	–
BBB	2025-10-14 09:01:11.052	ZZ-0012	–
BBB	2025-10-14 09:02:38.604	ZZ-0013	–
BBB	2025-10-14 09:03:17.983	WZ-2201	–
BBB	2025-10-14 09:04:55.200	WZ-2205	–
BBB	2025-10-14 09:05:22.671	ZZ-0012	–
BBB	2025-10-14 09:07:40.258	ZZ-0013	–
CCC	2025-10-14 09:30:00.424	ZZ-0012	–
CCC	2025-10-14 09:31:45.876	ZZ-0013	–
CCC	2025-10-14 09:32:21.905	WZ-3301	–
CCC	2025-10-14 09:33:10.452	WZ-3304	–

Construction of Lot- and Wafer-Level Timing Labels (FLOH/BLOH/WET/WST)

The operational meaning and productivity relevance of the four timing metrics (FLOH, BLOH, WET, WST) are introduced in detail in Section 1.1.2. Here we describe how they are computed from the raw event log. Because raw event logs do not directly provide the performance targets used in modeling, we calculate the timing labels from specific lot/wafer boundary events. First, the event codes that delimit lot and wafer processing are isolated. Second, the four timing metrics for each complete lot or wafer are calculated from the aligned boundaries:

- **FLOH (Front Lot Overhead):** time from the effective lot start to the start of the first wafer exposure.
- **BLOH (Back Lot Overhead):** time from the end of the last wafer exposure to the effective lot end.
- **WET (Wafer Expose Time):** for wafer n, time from wafer n expose start to wafer n expose end.
- **WST (Wafer Swap Time):** for wafer n ($n \geq 2$), the wafer $n-1$ expose end to wafer n expose start, with the first wafer defined as 0 swap time.

Event Filtering and Sequence Extraction

After loading, raw events can be filtered to remove known irrelevant or noisy event codes. The pipeline supports: (1) removing explicit event codes, (2) optionally removing all event codes with specified prefixes, (3) optionally keeping only events with specific prefixes to focus on a subsystem of interest. These filters are applied prior to sequence generation so that both model training and interpretation operate on the same cleaned vocabulary.

Next, we construct event sequences that correspond to the selected analysis target FLOH, BLOH, WET or WST (as introduced in Section 1.1.2). For each labeled instance, an event sequence is extracted by slicing the event code data between a start and end timestamp that matches the metric definition:

- **FLOH sequence:** events between effective lot start time and first wafer exposure start time.
- **BLOH sequence:** events between last wafer exposure end time and effective lot end time.
- **WET sequence:** for wafer n, events between wafer n start time and wafer n end time.
- **WST sequence:** for wafer n ($n \geq 2$), events between the wafer $n-1$ start time and the wafer n start time, intentionally including the prior wafer's exposure period to better capture preswap context.

Each extracted sequence is stored with its metadata (lot name/id, machine ID, wafer index) and its numeric label (target time).

Event-Code Refinement via Developer Text Classification

To improve event interpretability, another step in preprocessing is to further classify event codes using developer text. For a small set of high-impact event codes, the meaning of the event depends on the developer text. For these events, the pipeline maps recognized phrases in the developer text to a suffix (e.g., `SY-123D-03`), so the original event is split into multiple distinct variants. The vocabulary is rebuilt after classification.

Outlier Handling and Length Normalization

To stabilize training and avoid extreme cases dominating the regression loss, the pipeline optionally removes outlier sequences using two criteria:

1. Target-time threshold: sequences with target time $>$ outlier threshold (seconds) are dropped.
2. Sequence-length threshold: sequences longer than max seq length are removed.

For model input, sequences are also normalized structurally: tokens are mapped to integer IDs using a vocabulary. During batching, sequences are padded to the maximum length in the batch, and an attention mask is generated to distinguish real tokens from padding.

6.1.2 Pattern Mining and Consolidation

The PrefixSpan mining and overlap-ratio consolidation procedure applied here follows the methodology described in Section 4.2. Because exhaustive mining can be expensive on large datasets, mining is applied to a sampled subset of sequences, providing a controllable trade-off between coverage and runtime while still yielding representative high-support patterns. The minimum support threshold and pattern length bounds are configured per bucket based on dataset size and the desired result set cardinality. The consolidation threshold is set to $\tau = 0.98$. The full set of mining and consolidation hyperparameters used in the real-data experiments is listed in Table 6.2.

Table 6.2: Pattern mining and consolidation configuration for real-data experiments.

Parameter	Value
Sequential pattern mining (PrefixSpan)	
Min. pattern length	2 events
Max. pattern length	12 events
Min. support (occurrences)	20
Max. allowed gaps	1
Max. gap length	2
Mining sample size	5,000 sequences
Overlap-ratio consolidation	
Consolidation threshold τ	0.98
Max. length difference for consolidation	10

6.1.3 Model Training and Pattern Quantification

Once the event sequences and their timing labels have been prepared, the CLS Token Based model is trained independently for each bucket using an 80%/20% random train/test split, with early stopping monitored on the held-out test set. After training, the best model checkpoint is applied to the full sequence set (training and test sequences combined) to generate predictions; since no further parameter updates occur at this stage, applying the frozen model to training sequences does not constitute data leakage. The resulting predicted-versus-actual scatter and reported metrics (R^2, MAE, RMSE) are computed over all sequences and are used to assess whether the learned signal is reliable enough to support downstream pattern attribution. For each bucket, sequential pattern mining is then applied to the sequence collection to enumerate candidate patterns, followed by the overlap-ratio consolidation step. The consolidated pattern set is passed to the token replacement procedure, which quantifies each pattern's contribution by measuring the change in predicted timing when its tokens are replaced with non-causal fillers. The resulting Naive and CLS Token Based impact estimates are aggregated across all sequences to produce the global-level pattern rankings reported in Section 6.3.

The real-data experiments apply the CLS Token Based method with the same architecture and training configuration used in the synthetic experiments (Table 5.4), following the motivation that a consistent setup enables a fair comparison of method behavior across controlled and real-world conditions. The only parameters adjusted for the real data are the maximum sequence length (reduced to 60 to match observed real log lengths) and the early stopping patience (increased to 10 to accommodate the noisier training signal). The full configuration is summarized in Table 6.3.

Table 6.3: Implementation configuration for real-data experiments (CLS Token Based method).

Parameter	Value
Compute environment	
Platform	Databricks notebook
Device	CUDA GPU if available, else CPU
Software stack	
Framework	PyTorch
Pattern mining	PrefixSpan
Random seed	42
Model hyperparameters	
Embedding dimension	128
Transformer layers	2
Attention heads	4
Feed-forward dimension	256
Dropout	0.1
Max sequence length	60
Training setup	
Optimizer	AdamW (λ=10^{-4})
Learning rate	5×10^{-4}
Batch size	32
Epochs (max)	50
Early stopping patience	10
Loss function	MSE
Train / test split	80% / 20%
Quantification	
Filler token repetitions K	5

6.1.4 Evaluation Metrics

Two dimensions are evaluated in this chapter:

- **Prediction performance:** R^2, MAE, and RMSE on the hold-out test set, reported per bucket.
- **Pattern quantification and expert review:** for each retained bucket (FLOH and WST), the top-ranked patterns discovered by sequential pattern mining are listed together with their Naive and CLS Token Based quantified impact estimates. The pattern list is presented to domain experts, who assess whether each pattern corresponds to a known productivity-relevant behavior.

6.2 Dataset

We conducted experiments across four buckets (FLOH, BLOH, WST, WET) on two scanners: Scanner A and Scanner B. FLOH and BLOH are lot-level metrics, while WET and WST are

wafer-level metrics. Since one lot typically contains between 1 and 25 wafers, wafer-level buckets naturally yield many more sequences than lot-level buckets for the same period.

6.2.1 Scanner A

For Scanner A, event log data for WET and WST are collected over one month, while FLOH and BLOH data are collected over ten days to obtain a sufficient number of lot-level sequences. After data preprocessing and filtering, dataset-scale statistics are reported in Table 6.4, and the target-time distribution for each bucket is shown in Figure 6.1.

Table 6.4: Real-data sequence statistics by bucket for Scanner A

Bucket	Level	Sequences	Unique Event Types	Avg Len	Max Len
FLOH	Lot	1233	287	28.7	158
BLOH	Lot	1235	71	2.3	147
WST	Wafer	20834	241	19.6	168
WET	Wafer	20887	236	15.8	162

6.2.2 Scanner B

For Scanner B, data for all four buckets are collected over a three-week period from the same scanner. After data preprocessing and filtering, dataset-scale statistics are reported in Table 6.5, and the target-time distribution for each bucket is shown in Figure 6.2.

Table 6.5: Real-data sequence statistics by bucket for Scanner B

Bucket	Level	Sequences	Unique Event Types	Avg Len	Max Len
FLOH	Lot	4474	294	30.4	166
BLOH	Lot	4494	75	2.4	155
WST	Wafer	91344	249	20.8	176
WET	Wafer	91537	243	16.4	169

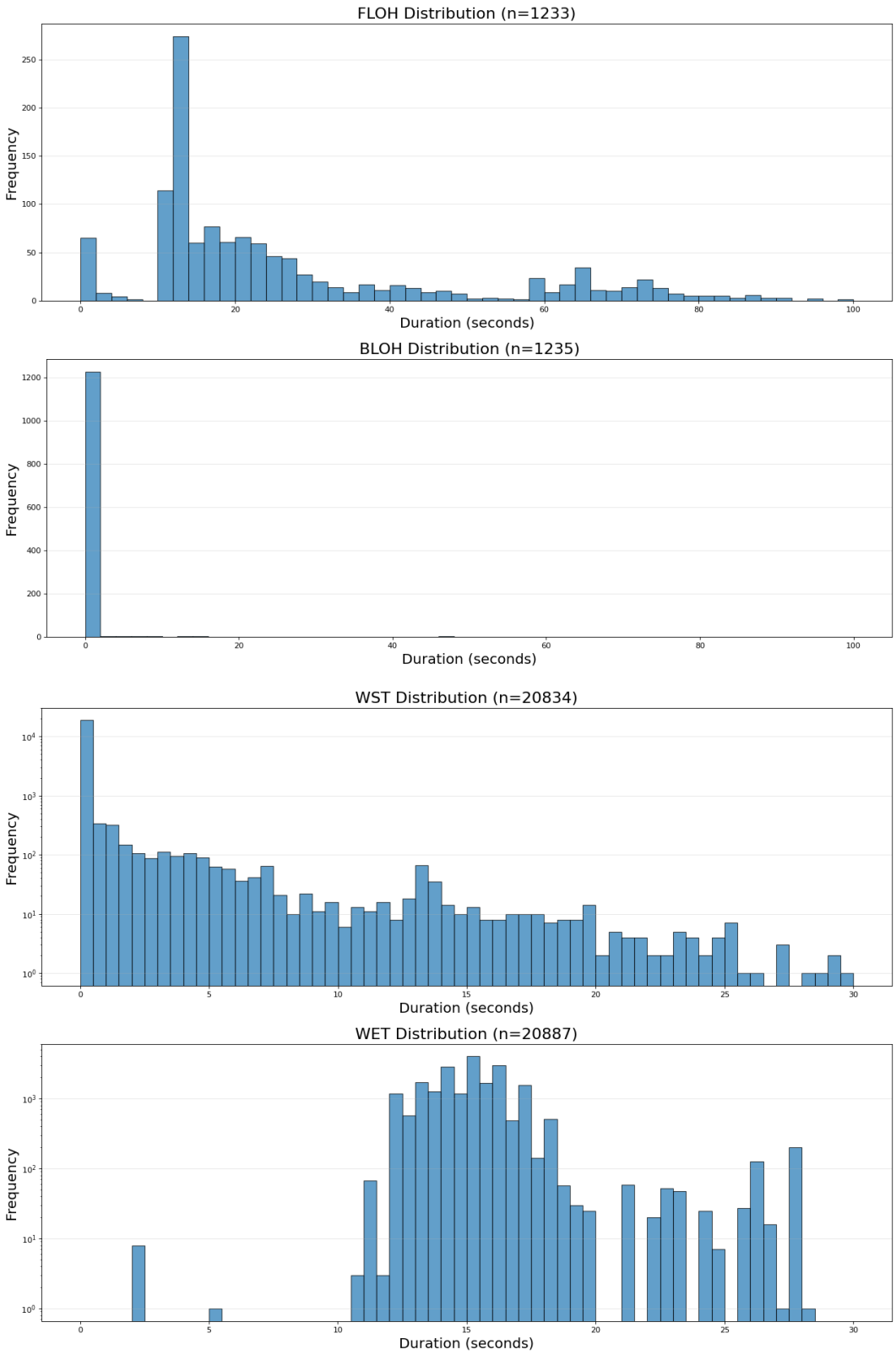


Figure 6.1: Real-data timing distributions by bucket for Scanner A

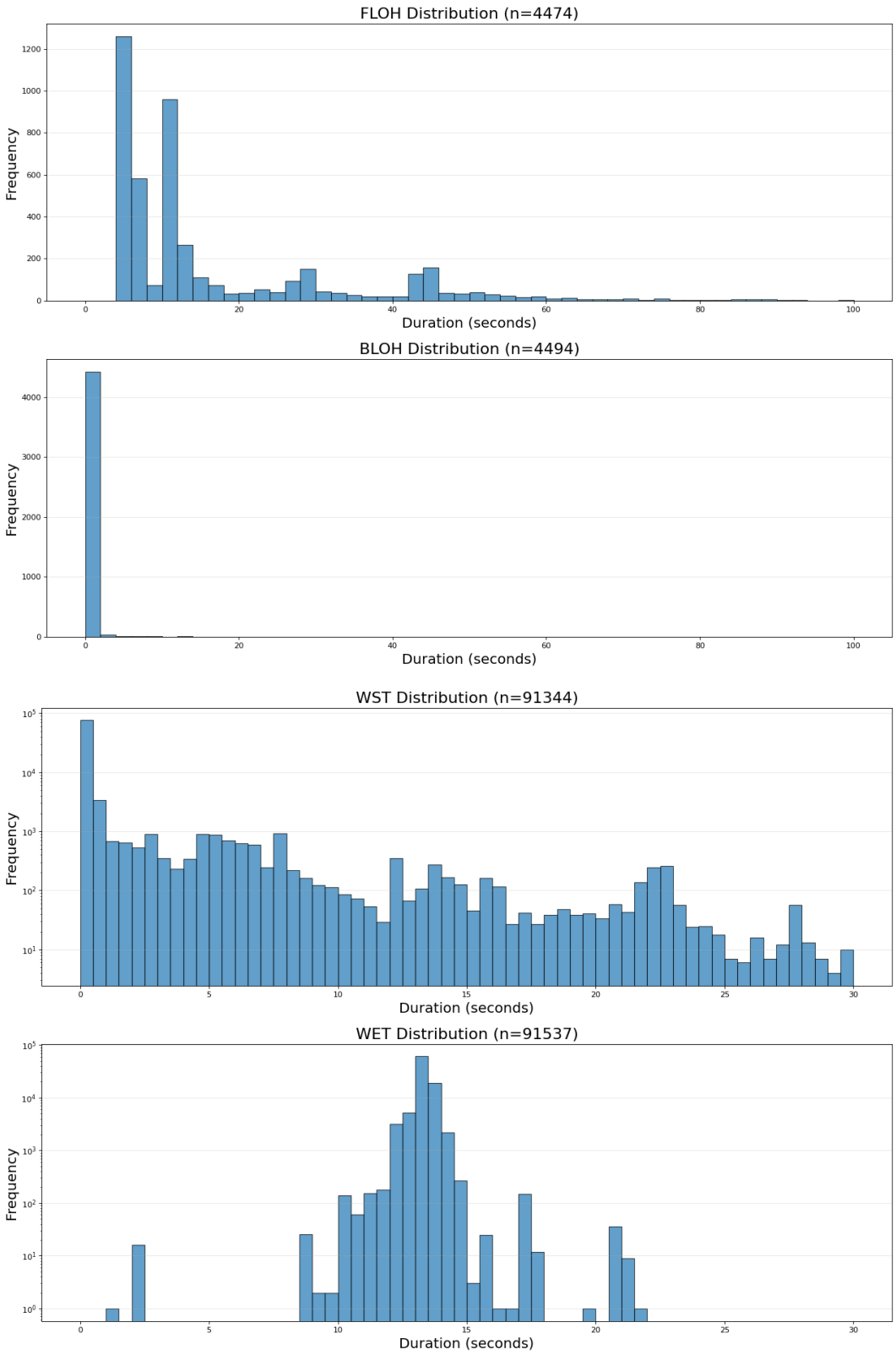


Figure 6.2: Real-data timing distributions by bucket for Scanner B

6.3 Results and Analysis

6.3.1 Prediction Performance Across Targets

Before examining individual pattern contributions, it is important to assess how well the model predicts the timing metric for each target, since reliable quantification depends on a well-calibrated predictor.

Scanner A

Table 6.6 reports R^2, MAE, and RMSE for the CLS Token Based method on Scanner A across all four targets.

Table 6.6: Prediction performance per target for Scanner A

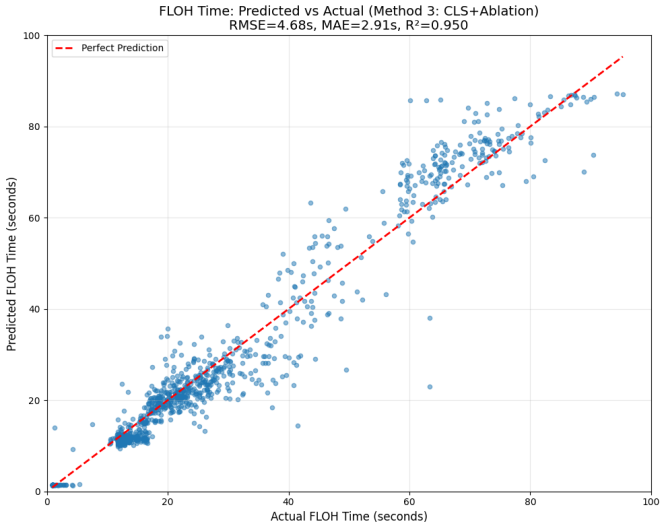
Target	Method	R^2	MAE (s)	RMSE (s)
FLOH	CLS Token Based	0.950	2.91	4.68
BLOH	CLS Token Based	-0.063	0.03	0.39
WST	CLS Token Based	0.755	0.33	1.56
WET	CLS Token Based	0.207	1.31	2.06

FLOH achieves strong predictive accuracy, with $R^2 = 0.950$, MAE of 2.91 s, and RMSE of 4.68 s. The scatter plot shows predictions clustering tightly around the identity diagonal, confirming that the model reliably captures the dominant sources of variance in front lot overhead. This high explained variance makes FLOH well-suited for pattern quantification and expert review. WST also achieves good prediction accuracy with $R^2 = 0.755$, MAE of 0.33 s, and RMSE of 1.56 s. The scatter plot shows that the model performs well for sequences with low swap times, but prediction accuracy decreases for high-overhead cases approaching 30 s; this suggests that the dominant causes of unusually long swap times are not fully represented by the event patterns available to the model, and the model did not find patterns closely correlated with near-30 s timing values. BLOH yields a negative R^2 (-0.063), indicating that the model performs worse than a constant-mean predictor. Inspection of the BLOH target-time distribution (Figure 6.1) reveals that the vast majority of BLOH values are at or near zero, with only a small fraction of sequences exhibiting non-trivial back-overhead. This extreme imbalance leaves the model unable to learn a reliable predictive signal, making pattern attribution for BLOH unreliable. BLOH is therefore excluded from all subsequent pattern quantification analysis. WET yields an R^2 of 0.207, with MAE 1.31 s and RMSE 2.06 s. Wafer expose time is primarily determined by the process configuration, in particular scan speed, defectivity-driven adjustments, and other related factors, which cannot be captured by the event sequence. The low explained variance confirms that the event sequence carries limited information about WET; WET patterns are therefore not submitted for expert review.

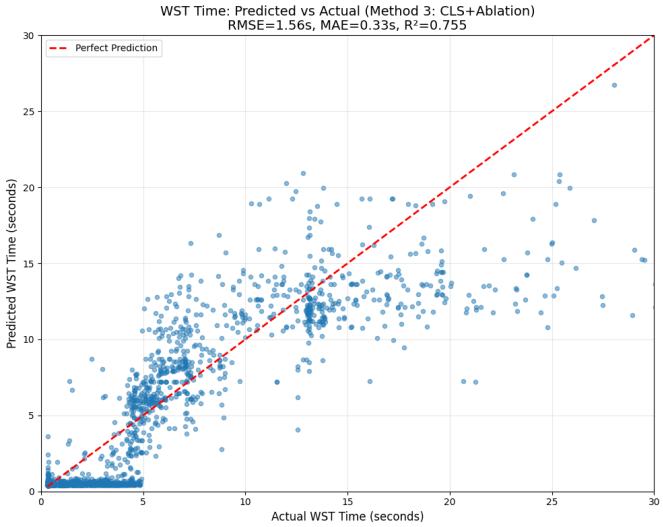
Predicted versus actual timing scatter plots for the two retained buckets are shown in Figure 6.3.

Scanner B

Table 6.7 reports R^2, MAE, and RMSE for the CLS Token Based method on Scanner B across all four targets.



(a) FLOH ($R^2 = 0.950$)



(b) WST ($R^2 = 0.755$)

Figure 6.3: Predicted versus actual timing for Scanner A across the two retained targets (FLOH and WST). Points on the diagonal indicate perfect prediction; spread around the diagonal reflects prediction error.

Table 6.7: Prediction performance per target for Scanner B

Target	Method	R^2	MAE (s)	RMSE (s)
FLOH	CLS Token Based	0.861	2.67	5.71
BLOH	CLS Token Based	-0.083	0.08	0.42
WST	CLS Token Based	0.848	0.41	1.56
WET	CLS Token Based	0.401	0.18	0.39

FLOH achieves good predictive accuracy with $R^2 = 0.861$, MAE of 2.67 s, and RMSE of 5.71 s. The model captures the dominant patterns well; however, scatter plots reveal that sequences in the high FLOH range are systematically under-predicted, suggesting that the rarest and longest overhead cases are not fully explained by the available event sequences alone. Despite this tail limitation, the overall prediction quality is sufficient to support reliable pattern quantification. WST achieves strong predictive accuracy with $R^2 = 0.848$, MAE of 0.41 s, and RMSE of 1.56 s, with prediction errors remaining relatively uniform across both low and high timing regions. BLOH yields a negative R^2 (-0.083), which, as for Scanner A, results from the near-degenerate distribution of back-overhead times (Figure 6.2). BLOH is therefore excluded from all subsequent pattern quantification analysis for Scanner B. WET yields an R^2 of 0.401, higher than for Scanner A. Nevertheless, given that wafer expose time is primarily determined by process configuration, WET patterns are not submitted for expert review.

Predicted versus actual timing scatter plots for the two retained buckets are shown in Figure 6.4.

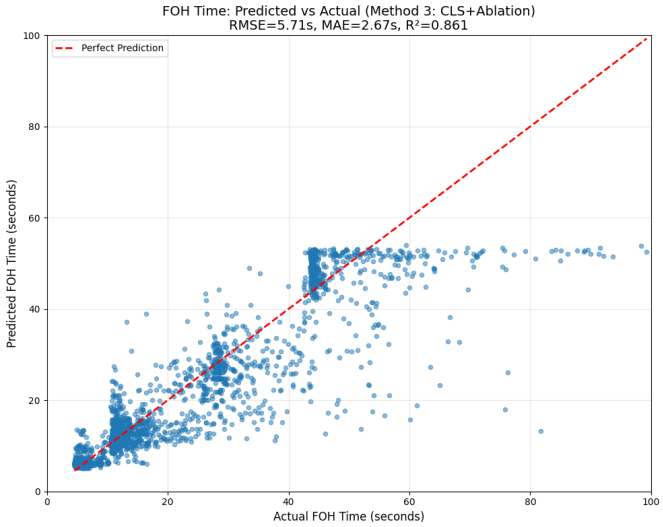
The Scanner B scatter plots display visible horizontal stripes: multiple sequences receive identical predicted values while their actual times differ. This occurs because the model is deterministic—sequences with identical event content produce the same prediction. The within-stripe vertical spread reflects timing variability driven by factors absent from the event log (e.g., unlogged system states or scheduling context), and marks the information ceiling of the event-sequence representation.

6.3.2 Pattern Quantification Results and Expert Review

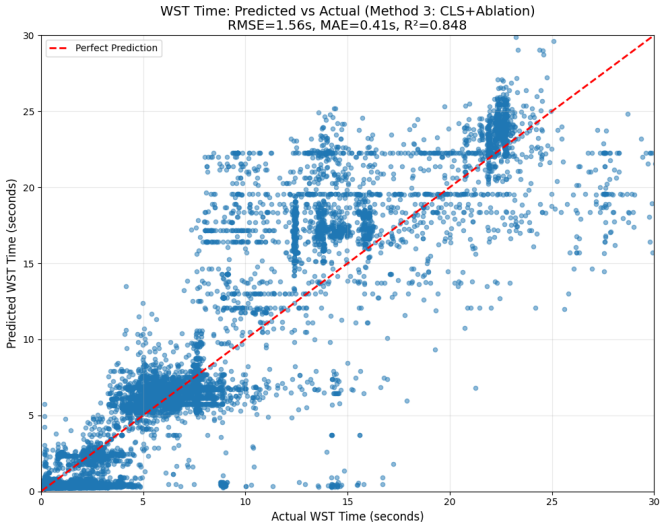
For each scanner and each retained bucket (FLOH and WST), the mined patterns are listed with their quantified impact estimates from both the Naive baseline and the CLS Token Based method. As discussed above, WET and BLOH are excluded from this analysis. Because real manufacturing event codes cannot be disclosed, each pattern is referenced by an anonymous index (Pattern 1, 2, ...). The tables are ordered by descending CLS impact within each bucket. The top-ranked patterns were presented to domain experts for review; their recognition status is recorded in the Expert column.

Due to the large number of mined patterns in the final results, only patterns with a CLS quantification impact greater than 5 seconds for LOH buckets and greater than 1 s for WST were selected for expert review. This threshold-based filtering ensures that expert attention is focused on patterns with practically meaningful impact on timing metrics.

Each table reports the following columns: Pattern is the anonymous pattern index; Naive (s) is the naive time-difference impact estimate; CLS (s) is the token replacement estimate from the CLS Token Based method; Prevalence (%) is the fraction of sequences in the bucket that contain the pattern; Avg With (s) and Avg Without (s) are the mean target times for sequences containing and not containing the pattern, respectively; Expert indicates whether a domain expert recognized



(a) FLOH ($R^2 = 0.861$)



(b) WST ($R^2 = 0.848$)

Figure 6.4: Predicted versus actual timing for Scanner B across the two retained targets (FLOH and WST). Points on the diagonal indicate perfect prediction; spread around the diagonal reflects prediction error.

the pattern as a known productivity-relevant behavior: *Fully Recognized* means the expert could identify the corresponding root cause and confirm that the quantified impact is consistent with their empirical experience of that behavior; *Partly Recognized* means the expert could identify only part of the event sequence within the pattern, and could not associate the full pattern with a single known behavior; *Not Recognized* means the pattern could not be associated with any known behavior. The total number of sequences per bucket is stated in the table caption.

Scanner A

Pattern quantification results for Scanner A are reported in Tables 6.8 (FLOH) and 6.9 (WST); only fully or partly recognized patterns are shown. Of the 71 mined FLOH patterns, 17 received expert recognition (15 fully, 2 partly). The top-ranked recognized patterns carry CLS impacts up to approximately 45 s. The CLS estimates are generally lower than the Naive estimates for high-impact patterns. This is explained by co-occurrence bias in the Naive estimator: sequences containing a given high-impact FLOH pattern tend to simultaneously contain other overhead-generating patterns (e.g., multiple interdependent initialization or reticle-handling behaviors that cluster under certain operational conditions). The Naive estimator computes the mean timing difference between sequences with and without the target pattern, so it inadvertently attributes the contributions of all co-occurring patterns to each individual pattern, inflating every estimate. The CLS Token Based method avoids this by replacing only the target pattern's tokens and measuring the marginal drop in the model's prediction, isolating the causal contribution of that pattern from whatever else co-occurs in the same sequence. Of the 33 WST patterns, 21 were recognized (19 fully, 2 partly), a high recognition rate reflecting that the dominant swap-time contributors are well-captured by the model; CLS and Naive estimates are closely aligned for the top ranks.

Table 6.8: Pattern quantification results for FLOH bucket, Scanner A (1,222 sequences total). Only fully/partly recognized patterns shown (17 of 71). Event codes are masked for confidentiality. Sorted by descending CLS impact.

Pattern	Naive (s)	CLS (s)	Prevalence (%)	Avg With (s)	Avg Without (s)	Expert Review
Pattern 1	46.31	44.87	5.5	70.13	23.81	Fully Recognized
Pattern 3	48.48	37.57	6.6	71.62	23.14	Fully Recognized
Pattern 4	46.10	37.56	6.7	69.36	23.26	Fully Recognized
Pattern 6	47.18	22.84	14.7	66.58	19.40	Fully Recognized
Pattern 14	46.27	16.02	5.6	70.04	23.78	Fully Recognized
Pattern 18	22.74	15.45	26.3	43.12	20.38	Fully Recognized
Pattern 21	45.40	14.13	5.7	69.15	23.75	Fully Recognized
Pattern 26	43.47	11.95	7.8	66.44	22.97	Fully Recognized
Pattern 31	37.24	10.45	14.6	58.17	20.93	Fully Recognized
Pattern 34	27.80	9.77	33.6	44.83	17.03	Fully Recognized
Pattern 35	27.84	9.69	20.4	48.52	20.68	Fully Recognized
Pattern 42	29.75	7.72	2.3	55.42	25.67	Fully Recognized
Pattern 51	47.89	6.55	6.7	71.03	23.14	Fully Recognized
Pattern 60	38.94	5.49	5.8	63.03	24.09	Fully Recognized
Pattern 66	35.49	5.29	2.7	60.88	25.39	Partly Recognized
Pattern 67	37.69	5.16	9.5	60.46	22.78	Fully Recognized
Pattern 68	22.65	5.09	12.6	46.15	23.50	Partly Recognized

Table 6.9: Pattern quantification results for WST bucket, Scanner A (20,834 sequences total) with expert recognition. Only fully/partly recognized patterns shown (21 of 33). Event codes are masked for confidentiality. Sorted by descending CLS impact.

Pattern	Naive (s)	CLS (s)	Prevalence (%)	Avg With (s)	Avg Without (s)	Expert Review
Pattern 1	16.32	19.09	2.6	17.55	1.23	Fully Recognized
Pattern 2	19.42	16.30	2.0	20.69	1.27	Fully Recognized
Pattern 3	6.95	13.69	1.2	8.52	1.57	Fully Recognized
Pattern 4	18.80	13.40	1.8	20.11	1.31	Fully Recognized
Pattern 5	5.20	12.42	1.1	6.79	1.59	Partly Recognized
Pattern 6	17.14	11.93	2.8	18.32	1.18	Fully Recognized
Pattern 7	6.95	11.82	1.2	8.52	1.57	Fully Recognized
Pattern 8	17.14	11.72	2.8	18.32	1.18	Fully Recognized
Pattern 9	16.49	11.30	2.7	17.69	1.20	Fully Recognized
Pattern 10	5.20	10.93	1.1	6.79	1.59	Fully Recognized
Pattern 13	17.14	10.25	2.8	18.32	1.18	Fully Recognized
Pattern 16	13.18	7.93	3.5	14.38	1.19	Fully Recognized
Pattern 17	18.80	7.11	1.8	20.11	1.31	Fully Recognized
Pattern 20	13.83	4.05	3.9	14.94	1.11	Fully Recognized
Pattern 22	1.70	2.40	0.9	3.34	1.64	Partly Recognized
Pattern 23	5.12	2.37	1.1	6.72	1.59	Fully Recognized
Pattern 24	13.97	2.33	3.3	15.16	1.19	Fully Recognized
Pattern 27	13.83	1.52	3.9	14.94	1.11	Fully Recognized
Pattern 29	16.49	1.47	2.7	17.69	1.20	Fully Recognized
Pattern 31	3.45	1.17	2.1	5.03	1.58	Fully Recognized
Pattern 33	13.15	1.06	3.9	14.28	1.14	Fully Recognized

Scanner B

Pattern quantification results for Scanner B are reported in Tables 6.10 (FLOH) and 6.11 (WST); only fully or partly recognized patterns are shown. Of the 96 mined FLOH patterns, 39 received expert recognition (1 fully, 38 partly). The higher raw count compared to Scanner A reflects the larger pattern set, while the predominance of partial over full recognition indicates that Scanner B exhibits distinct event sequences whose physical meaning is partially but not definitively identified by domain experts. The top-ranked patterns carry CLS impacts up to approximately 46 s. Of the 32 WST patterns, 27 were recognized (7 fully, 20 partly), indicating that the dominant swap-time patterns are well-covered.

Table 6.10: Pattern quantification results for FLOH bucket, Scanner B (4,226 sequences total) with expert recognition. Only fully/partly recognized patterns shown (39 of 96). Event codes are masked for confidentiality. Sorted by descending CLS impact.

Pattern	Naive (s)	CLS (s)	Prevalence (%)	Avg With (s)	Avg Without (s)	Expert Review
Pattern 1	40.03	45.69	2.7	54.90	14.87	Partly Recognized
Pattern 2	38.20	41.30	5.9	51.91	13.71	Fully Recognized
Pattern 3	36.60	40.70	1.6	52.00	15.40	Partly Recognized
Pattern 4	52.19	40.23	1.2	67.55	15.36	Partly Recognized
Pattern 5	36.76	34.14	7.8	49.85	13.10	Partly Recognized
Pattern 7	39.39	33.09	1.2	54.90	15.51	Partly Recognized
Pattern 9	33.23	31.50	1.7	48.62	15.39	Partly Recognized
Pattern 15	36.66	29.07	7.7	49.80	13.14	Partly Recognized
Pattern 16	36.56	27.75	1.4	52.02	15.46	Partly Recognized
Pattern 17	35.20	26.75	0.9	50.87	15.67	Partly Recognized
Pattern 19	33.65	26.27	1.5	49.13	15.48	Partly Recognized
Pattern 22	36.08	24.77	4.7	50.34	14.26	Partly Recognized
Pattern 23	32.25	24.29	1.8	47.62	15.37	Partly Recognized
Pattern 27	35.76	20.73	6.3	49.47	13.72	Partly Recognized
Pattern 31	35.07	19.43	1.1	50.66	15.59	Partly Recognized
Pattern 33	33.98	18.84	1.5	49.44	15.46	Partly Recognized
Pattern 36	31.58	18.40	2.1	46.88	15.30	Partly Recognized
Pattern 38	34.15	17.62	0.9	49.81	15.66	Partly Recognized
Pattern 40	35.03	17.14	3.5	49.77	14.74	Partly Recognized
Pattern 43	47.42	15.95	1.0	62.92	15.50	Partly Recognized
Pattern 44	36.27	14.99	1.6	51.66	15.39	Partly Recognized
Pattern 46	41.64	12.97	0.6	57.36	15.72	Partly Recognized
Pattern 48	32.10	12.33	4.7	46.55	14.46	Partly Recognized
Pattern 51	38.66	11.54	2.3	53.74	15.08	Partly Recognized
Pattern 52	31.59	11.42	11.1	44.04	12.46	Partly Recognized
Pattern 55	31.52	10.74	1.4	47.06	15.54	Partly Recognized
Pattern 57	33.23	10.72	1.7	48.62	15.39	Partly Recognized
Pattern 59	41.13	10.45	1.0	56.67	15.54	Partly Recognized
Pattern 61	33.23	10.34	1.7	48.62	15.39	Partly Recognized
Pattern 63	36.87	9.87	4.4	51.24	14.36	Partly Recognized
Pattern 64	36.27	9.85	1.6	51.66	15.39	Partly Recognized
Pattern 65	37.37	9.84	2.0	52.58	15.21	Partly Recognized
Pattern 70	31.59	9.12	11.1	44.04	12.46	Partly Recognized
Pattern 72	31.53	7.73	2.2	46.82	15.28	Partly Recognized
Pattern 76	36.07	7.16	4.0	50.60	14.53	Partly Recognized
Pattern 82	35.76	6.31	6.3	49.47	13.72	Partly Recognized
Pattern 87	35.96	6.01	2.0	51.22	15.26	Partly Recognized
Pattern 89	35.47	5.91	1.9	50.75	15.28	Partly Recognized
Pattern 92	36.87	5.03	4.4	51.24	14.36	Partly Recognized

Table 6.11: Pattern quantification results for WST bucket, Scanner B (91,344 sequences total) with expert recognition. Only fully/partly recognized patterns shown (27 of 32). Event codes are masked for confidentiality. Sorted by descending CLS impact.

Pattern	Naive (s)	CLS (s)	Prevalence (%)	Avg With (s)	Avg Without (s)	Expert Review
Pattern 2	8.92	12.77	2.5	9.88	0.96	Partly Recognized
Pattern 3	15.67	10.92	1.1	16.69	1.01	Partly Recognized
Pattern 4	4.06	9.76	1.0	5.21	1.15	Partly Recognized
Pattern 5	18.61	9.46	1.2	19.56	0.95	Fully Recognized
Pattern 6	4.00	9.30	0.8	5.15	1.15	Partly Recognized
Pattern 7	4.96	8.12	1.6	6.06	1.10	Partly Recognized
Pattern 8	5.62	7.39	3.0	6.63	1.02	Partly Recognized
Pattern 9	5.75	7.31	2.2	6.81	1.06	Fully Recognized
Pattern 10	5.28	6.84	2.3	6.34	1.06	Partly Recognized
Pattern 11	5.62	6.69	3.1	6.63	1.01	Partly Recognized
Pattern 12	5.64	6.19	1.8	6.73	1.08	Fully Recognized
Pattern 13	4.96	6.10	1.6	6.06	1.10	Partly Recognized
Pattern 14	5.64	5.93	1.8	6.73	1.08	Fully Recognized
Pattern 15	4.95	5.50	1.7	6.05	1.10	Partly Recognized
Pattern 17	6.19	4.38	1.6	7.28	1.09	Fully Recognized
Pattern 18	6.67	4.13	1.0	7.79	1.12	Partly Recognized
Pattern 19	9.63	4.07	4.7	10.36	0.73	Partly Recognized
Pattern 20	18.37	4.00	0.6	19.45	1.08	Partly Recognized
Pattern 21	7.01	3.82	1.0	8.13	1.12	Fully Recognized
Pattern 22	4.60	3.67	3.4	5.63	1.03	Partly Recognized
Pattern 24	18.61	3.07	1.2	19.56	0.95	Fully Recognized
Pattern 25	4.67	2.94	0.7	5.82	1.15	Partly Recognized
Pattern 26	5.28	2.86	2.3	6.34	1.06	Partly Recognized
Pattern 28	17.14	1.75	0.6	18.23	1.09	Partly Recognized
Pattern 29	6.67	1.74	1.0	7.79	1.12	Partly Recognized
Pattern 30	17.14	1.04	0.6	18.23	1.09	Partly Recognized
Pattern 31	8.31	1.03	1.3	9.39	1.08	Partly Recognized

6.3.3 Qualitative Expert Feedback on Unrecognized Patterns

Beyond confirming known patterns, the expert review session provided qualitative insights about patterns that were not recognized. For both scanners and both buckets (FLOH and WST), the majority of unrecognized patterns were short two- or three-event sequences that experts could not unambiguously associate with a specific operational behavior. Experts suggested that these patterns may reflect legitimate but uncommon operational variants — for example, recovery sequences triggered under rare machine states — that are not well-documented in existing engineering knowledge. Several such patterns were flagged as worth investigating further, particularly those with high CLS-estimated impact but low prevalence, as they may represent infrequent but high-overhead root causes that have not yet been systematically catalogued.

In a small number of cases, experts expressed interest in patterns they had not previously encountered as a combined sequence, even though they recognized the individual event codes within them. These patterns were considered potentially new behavioral signatures worth deeper investigation through targeted data pulls. No quantitative follow-up results were available at the time of writing, but the experts' engagement with the unrecognized patterns indicates that the pipeline has the potential to surface genuinely novel findings beyond confirming known behaviors — which strengthens the case for deploying it as an exploratory tool in engineering practice.

6.4 Interactive Spotfire Dashboard for Pattern Exploration

After validation through experiments on real manufacturing datasets, domain experts confirmed the effectiveness of the prediction and quantification method. Consequently, we further developed an analysis pipeline and an interactive dashboard to support engineers in diagnosing and localizing previously unknown issues.

6.4.1 Analysis Pipeline

The analysis pipeline is designed to systematically uncover and quantify previously unknown rate efficiency loss patterns from manufacturing event logs. Starting from raw event code data stored in Databricks, rate efficiency loss metrics are computed at both lot and wafer levels, including FLOH, BLOH, WST, and WET. For each loss bucket, relevant event code sequences are extracted based on predefined temporal boundaries. A machine learning model is then trained to predict the duration of each sequence, enabling the quantification of time losses associated with specific event code patterns. Finally, sequence pattern mining is performed to identify recurrent and impactful patterns, whose quantified contributions are aggregated and provided as structured outputs for expert analysis and visualization.

6.4.2 Interactive Dashboard

To help exploration of the mined event code patterns and their quantified impact, an interactive dashboard was developed using Spotfire. The dashboard is tightly integrated with the analysis pipeline outputs and supports both global pattern inspection and detailed, pattern-specific investigation.

The main view (Figure 6.5) of the dashboard presents a predicted-versus-actual scatter plot, where each point corresponds to an individual event code sequence and its associated process time. This view allows users to visually assess model performance and identify regions of interest, such as sequences associated with larger prediction residuals or extended durations. Users can interactively select and individual sequence or a subset of sequences directly from the scatter plot for further analysis.

Based on the selected sequence set, the dashboard highlights which mined event code patterns are present and summarizes their quantified contributions. Users can then mark a specific pattern of interest for deeper inspection. For the selected pattern, the dashboard provides detailed distribution views (Figure 6.6) of the sequences containing that pattern, including process time, machine number, and other relevant metadata. These views enable domain experts to investigate systematic behaviors, identify correlations with specific tools or process conditions, and support root cause analysis of previously unknown rate efficiency losses.

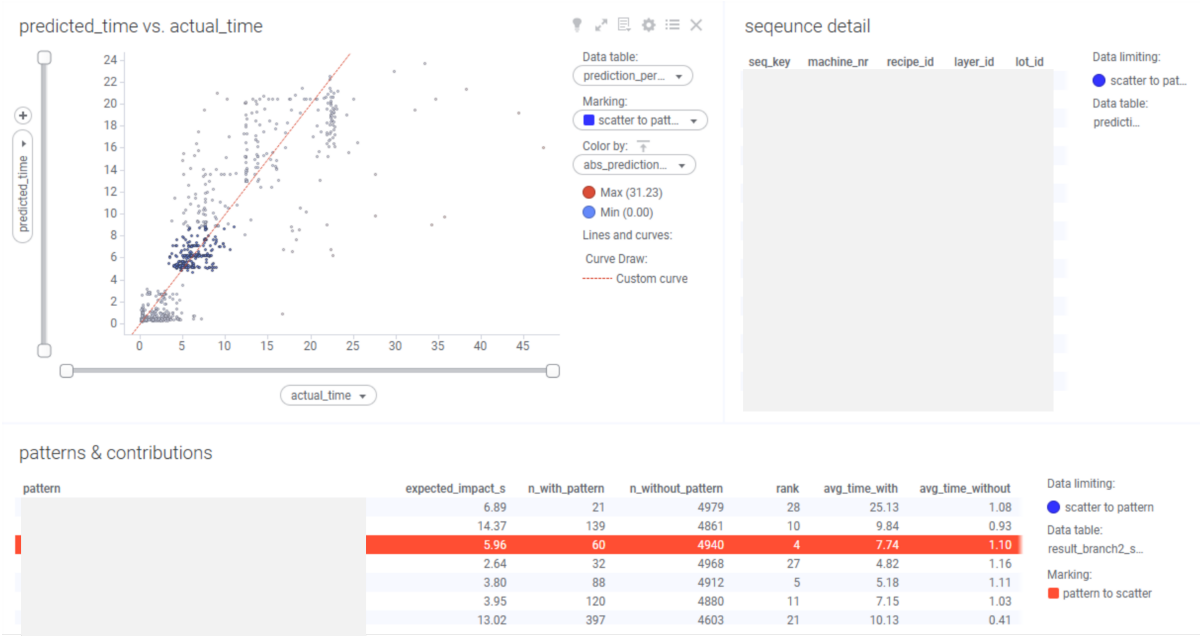


Figure 6.5: Spotfire dashboard main view, showing the predicted-versus-actual scatter plot. Users can interactively select subsets of event code sequences to inspect associated patterns and their quantified impact. Detailed sequence information is intentionally covered for confidentiality.

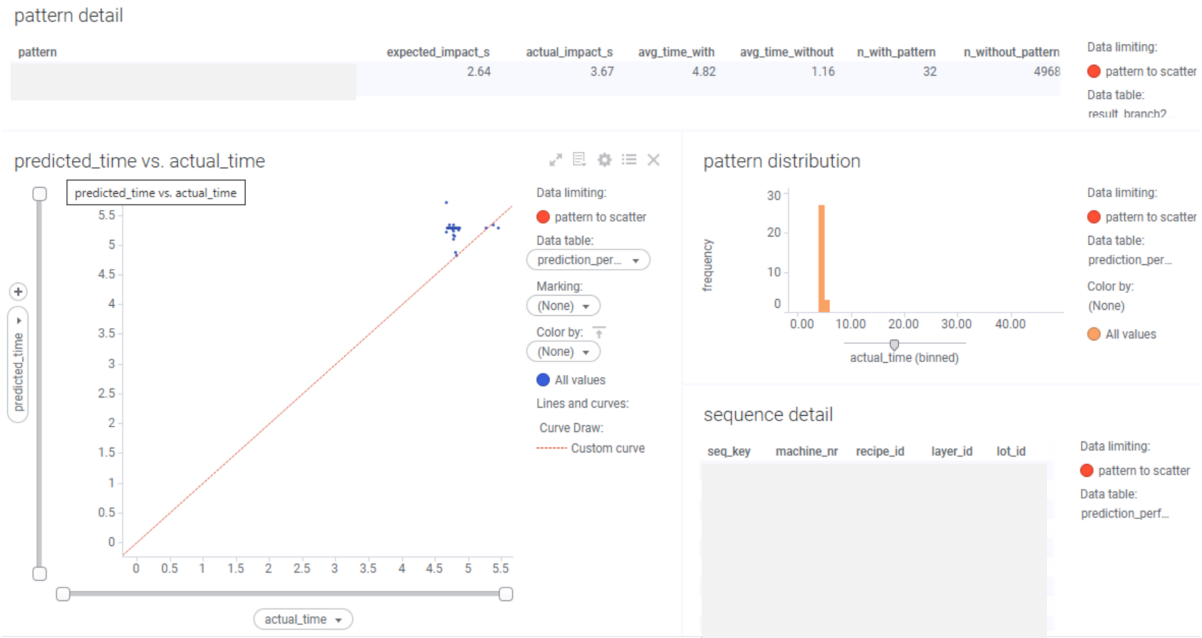


Figure 6.6: Deep-dive analysis view of the Spotfire dashboard. For a selected event code pattern, detailed distributions of associated sequences are shown to support root cause investigation. Sensitive sequence details are intentionally covered for confidentiality.

Chapter 7

Conclusion

This thesis addressed the problem of quantifying the timing impact of sequential event patterns in semiconductor manufacturing logs, with the goal of moving beyond aggregate time-difference statistics toward per-sequence, model-grounded attribution. The key question driving the work was whether a transformer-based model, trained directly on raw event code sequences, could both predict scalar process outcomes accurately and support reliable attribution of timing contributions to discovered event patterns. This chapter answers that question by drawing together findings from the synthetic and real-data experiments, reflecting on their broader significance and limitations in the context of prior work, and identifying directions for future research.

7.1 Conclusions

RQ1: Prediction Performance

Can a transformer-based model, trained on raw event code sequences from machine event logs, accurately predict scalar process outcomes?

The answer is yes, with important nuances depending on the target metric. The CLS Token Based method consistently achieved high prediction accuracy for FLOH and WST on both real scanners: R^2=0.950 and R^2=0.755 on Scanner A, and R^2=0.861 and R^2=0.848 on Scanner B. These results confirm that a transformer encoder trained directly on raw event code sequences, without any manual feature engineering, can capture the dominant sources of timing variance in front lot overhead and wafer swap time. The methodology is not inherently scanner-specific: the same architecture and pipeline can be applied to any machine event log paired with a scalar process metric, making this a general result.

Not all metrics are equally predictable from event sequences, however. BLOH was unpredictable on both scanners due to a near-constant target distribution, and WET showed limited predictability (R^2=0.207 and R^2=0.401), consistent with the expectation that wafer expose time is primarily governed by process configuration rather than event-sequence patterns. These results clarify the conditions under which event-sequence models are and are not the appropriate analytical tool: a transformer can be expected to predict a metric well when that metric is primarily determined by which operational procedures were triggered and in what order, i.e., when the event sequence itself encodes the causal signal. When the dominant variance is instead driven by process physics or configuration parameters that do not manifest as distinct event subsequences, the event log carries

insufficient information and the model is bounded by that informational ceiling. This distinction has practical value: it provides engineers with principled guidance on which timing metrics are amenable to event-sequence analysis and which are not, rather than leaving this as an empirical question to be answered by trial and error for each new target.

RQ2: Transformer Architecture and Pattern Contribution Quantification

Can a transformer-based timing prediction model, combined with a post-hoc token replacement procedure, achieve accurate and stable quantification of event sequence pattern contributions at both individual sequence and global levels?

The answer is yes for the CLS Token Based method, with the architecture choice mattering substantially. On the synthetic benchmark with known ground-truth impacts, the CLS Token Based method consistently outperformed both the Naive baseline and the Attention Pooling Based method across most data complexity conditions. On the clean baseline tier, CLS halved the Naive quantification error (1.62 vs. 3.55 seconds MAE). The advantage is most decisive under pattern overlap, where shared-prefix tokens cause the Attention Pooling Based method to systematically misattribute contributions across overlapping patterns, while the CLS token replacement is immune to this artefact by measuring the holistic prediction drop when an entire pattern is masked. Under strong interaction effects, both transformer methods substantially outperform the Naive baseline. Only under extremely high target noise do all methods converge to similar errors, because noise corrupts the learned model weights that transformers depend on.

The mechanistic explanation for the CLS advantage under overlap is worth emphasising, as it generalises beyond the specific architecture studied here. Any attribution method that apportions importance at the token level — including raw attention-weight analysis and similar approaches from the literature — will encounter the same failure mode when patterns share tokens, because a shared token's weight is simultaneously attributed to both patterns. The CLS token replacement sidesteps this entirely by measuring the causal effect of removing an entire pattern from the sequence context. This insight suggests that holistic sequence-level masking is the principled choice for pattern attribution whenever structural overlap cannot be ruled out in the data, and that token-level weighting schemes should be treated with caution in such settings.

A critical and exclusive capability of both transformer methods is sequence-level attribution: for any given sequence, the token replacement procedure yields a per-pattern contribution estimate, enabling individual root-cause diagnosis. The Naive baseline fundamentally cannot provide this capability, as it only produces population-level averages. This sequence-level breakdown was demonstrated on real industrial data and deployed in the Spotfire dashboard (Chapter 6).

RQ3: Practical Pattern Validation

When applied to real industrial scanner data, can the proposed approach identify pattern contributions that align with known productivity issues, and highlight additional patterns considered relevant by domain experts?

The answer is yes on both counts. On Scanner A, 17 of 71 FLOH patterns and 21 of 33 WST patterns were recognized by domain experts as known productivity-relevant behaviors, with CLS impact estimates consistent with engineers' empirical experience of those behaviors. On Scanner B, 39 of 96 FLOH patterns and 27 of 32 WST patterns were recognized. The high recognition

rate for WST patterns on both scanners confirms that the dominant swap-time contributors are reliably captured. For FLOH, the lower recognition rate partly reflects the greater diversity of lot-overhead sources, and partly that some unrecognized patterns may represent genuinely new behavioral signatures that have not yet been documented in engineering knowledge bases.

A notable difference between the two scanners is the ratio of fully to partly recognized patterns: Scanner A shows a higher rate of full recognition, while Scanner B exhibits predominantly partial recognition in the FLOH bucket. This likely reflects scanner-specific operational differences — Scanner B may activate more complex or less-documented initialization procedures — but it also reveals a practical boundary of the current approach: when the pipeline surfaces a pattern whose individual event codes engineers can identify but whose combined sequence cannot be associated with a single known behavior, the result is still actionable as a prompt for targeted investigation, even if immediate interpretation is unavailable. Expert review indicated that several such high-impact unrecognized patterns were flagged as worth further investigation, suggesting the pipeline has practical utility both for confirming known behaviors and for guiding discovery of new ones.

Main Research Question

Can a transformer-based model, trained directly on machine event log sequences, accurately predict scalar process outcomes and support reliable quantification of event sequence pattern contributions at both individual sequence and population levels?

Yes. The thesis demonstrates an end-to-end pipeline — transformer prediction, sequential pattern mining, and token replacement quantification — that is more accurate than the Naive time-difference method under the structural data complexities (interaction, position dependence, pattern overlap) that are characteristic of real manufacturing event logs. It also uniquely provides per-sequence attribution. The pipeline has been validated on controlled synthetic data with known ground truth and on real industrial scanner data with expert review, and has been deployed as an interactive productivity analysis tool.

7.2 Strengths and Limitations

Relative to prior research, the pipeline's central strength is its integration of capabilities that have previously been addressed in isolation: end-to-end prediction from raw event sequences, sequential pattern mining, and per-sequence attribution of pattern contributions. Process mining and sequence-to-target models each address only one of these; post-hoc attribution methods such as SHAP have not been combined with pattern mining in a manufacturing event log context. The CLS token replacement approach further improves on attention-weight attribution by measuring the holistic prediction drop when a complete pattern is masked, avoiding the token double-counting failure that affects any weight-based method when patterns share subsequences — a benefit confirmed empirically across all synthetic overlap tiers. The controlled synthetic benchmark is a second methodological contribution in its own right: by generating data with known additive causal structure, quantification error can be measured directly and structural failure modes diagnosed mechanistically, providing a reference framework that does not exist in prior work.

On the application side, the pipeline directly addresses the gaps in current industrial lithography practice identified at the outset of this thesis. It requires no manual feature engineering and

no customer-side data sources such as BLT data, operating entirely on event codes available in ASML's PMA infrastructure. It produces both a global pattern ranking — supporting systematic prioritization of productivity improvement efforts — and individual sequence breakdowns that enable targeted root-cause diagnosis for specific anomalous lots or wafers, a capability that aggregate time-difference methods fundamentally cannot provide. The unrecognized high-impact patterns flagged by experts as worth investigating further suggest the pipeline can function as an exploratory discovery tool, not only as a confirmation mechanism for known behaviors. The deployed Spotfire dashboard demonstrates that these capabilities are practically usable by process engineers without machine learning expertise.

The principal limitations concern computational cost, completeness, and architectural assumptions. Token replacement requires K forward passes per pattern occurrence, making the CLS method orders of magnitude slower than the Naive baseline; on the 91,344-sequence WST dataset for Scanner B this is a real deployment constraint. Attribution quality is also bounded by the mined pattern set: infrequent or structurally complex patterns below the PrefixSpan support threshold are silently excluded, so rare but high-impact root causes may be missed. The replacement procedure rests on an isolation assumption — that a pattern's contribution can be measured by replacing its tokens without altering the surrounding context — which may under- or over-estimate when effects are deeply context-dependent. Finally, the model discards inter-event timestamps, and models are trained per scanner without cross-machine transfer, limiting portability.

7.3 Future Work

The most pressing technical limitation is computational cost. Incorporating Shapley-based pretraining objectives, as in ShapTST [6], could yield attribution estimates in a single forward pass rather than requiring K replacement passes per pattern occurrence. Alternatively, a fast screening step using a lightweight proxy model could identify the small subset of high-impact patterns warranting full CLS replacement quantification, reducing total inference time substantially without sacrificing accuracy on the patterns that matter.

A second open problem is the treatment of rare patterns. The current pipeline silently excludes patterns below the PrefixSpan support threshold, yet infrequent but high-impact root causes are precisely those of greatest engineering interest. Moving toward an end-to-end architecture that jointly learns to discover and rank patterns — for example through differentiable pattern selectors trained with the prediction objective — could surface such patterns without requiring them to meet a frequency threshold. Separately, adding bootstrap confidence intervals to the token replacement estimates would help engineers distinguish reliable impacts from noisy ones, which is especially important for low-prevalence patterns where few sequences are available for averaging. A related architectural direction concerns the reliability of the event ordering itself. Currently, event logs from all scanner subsystems are merged into a single flat sequence, but because different components operate on independent clocks and logging latency varies across subsystems, the resulting token order may not faithfully reflect the true temporal order of operations. A natural extension would be to model each subsystem's event stream separately and fuse the resulting representations at a higher level, which could both reduce ordering noise and make cross-subsystem interaction patterns more explicitly learnable.

More broadly, the pipeline is applicable to any system producing discrete event logs paired with

a scalar outcome, and validating it on other manufacturing equipment. Cross-machine transfer, through fleet-level pre-training with machine-specific fine-tuning, could further reduce cold-start problems for newly installed tools and reveal behavioral patterns that are invisible when each machine is modeled independently.

Bibliography

[1] Sabeen Ahmed, Ian E Nielsen, Aakash Tripathi, Shamoon Siddiqui, Ravi P Ramachandran, and Ghulam Rasool. Transformers in time-series analysis: A tutorial. *Circuits, systems, and signal processing*, 42(12):7433–7466, 2023.

[2] ASML. TWINSCAN: 20 years of innovation. https://www.asml.com/en/news/stories/2021/twinscan-20-years-innovation, 2021. Accessed: 2026-06-20.

[3] Stephen D Bay and Michael J Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, 2001.

[4] Alessandro Berti, SJ Van Zelst, and W Van Der Aalst. Process mining for python (pm4py): bridging the gap between process-and data science (2019). *arXiv preprint arXiv:1905.06169*, 1905.

[5] Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 782–791, 2021.

[6] Qisen Cheng, Jinming Xing, Chang Xue, and Xiaoran Yang. Unifying prediction and explanation in time-series transformers via shapley-based pretraining. In *2025 21st IEEE International Colloquium on Signal Processing & Its Applications (CSPA)*, pages 325–330. IEEE, 2025.

[7] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL workshop BlackboxNLP: analyzing and interpreting neural networks for NLP*, pages 276–286, 2019.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186, 2019.

[9] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 43–52. ACM, 1999.

[10] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298, 2017.

[11] Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, et al. Explainable ai (xai): Core ideas, techniques, and solutions. *ACM computing surveys*, 55(9):1–33, 2023.

[12] Sofiane Ennadir, Levente Zólyomi, Oleg Smirnov, Tianze Wang, John Pertoft, Filip Cornell, and Lele Cao. Pool me wisely: On the effect of pooling in transformer-based models. *arXiv preprint arXiv:2510.03339*, 2025.

[13] Philippe Fournier-Viger, Antonio Gomariz, Manuel Campos, and Rincy Thomas. Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 40–52. Springer, 2014.

[14] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77, 2017.

[15] Philippe Fournier-Viger, Cheng-Wei Wu, Antonio Gomariz, and Vincent S Tseng. Vmsp: Efficient vertical mining of maximal sequential patterns. In *Canadian conference on artificial intelligence*, pages 83–94. Springer, 2014.

[16] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, Meichun Hsu, et al. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224. IEEE Piscataway, NJ, USA, 2001.

[17] Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 3543–3556, 2019.

[18] Serdar Kadıoğlu, Xin Wang, Amin Hosseininasab, and Willem-Jan van Hoeve. Seq2pat: Sequence-to-pattern generation to bridge pattern mining with machine learning. *AI Magazine*, 44(1):54–66, 2023.

[19] Enja Kokalj, Blaž Škrlj, Nada Lavrač, Senja Pollak, and Marko Robnik-Šikonja. Bert meets shapley: Extending shap explanations to transformer-based classifiers. In *Proceedings of the EACL hackashop on news media content analysis and automated report generation*, pages 16–21, 2021.

[20] Sander JJ Leemans et al. Robust process mining with guarantees. In *BPM (Dissertation/Demos/Industry)*, pages 46–50. Springer, 2018.

[21] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, et al. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *Ijcai*, volume 19, pages 4739–4745, 2019.

[22] Petra Kralj Novak, Nada Lavrač, and Geoffrey I Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–435, 2009.

[23] Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*, 2024.

[24] Mattia Rigotti, Christoph Miksovic, Ioana Giurgiu, Thomas Gschwind, and Paolo Scotton. Attention-based interpretability with concept transformers. In *International conference on learning representations*, 2021.

[25] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *International conference on extending database technology*, pages 1–17. Springer, 1996.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[27] Maurice Wimmer, Alexander Serebrenik, Anton Wijs, Marwan Hassani, Ignacio Alonso, Joost Gevers, and Stiven Schwanz Dias. Predicting disturbances in high-tech lithography machines. 2022.

[28] Mohammed J Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1):31–60, 2001.

[29] S Zargar. Introduction to sequence learning models: Rnn, lstm, gru. *Department of Mechanical and Aerospace Engineering, North Carolina State University*, 37988518, 2021.

[30] Przemyslaw Zbrozek. Using predictive process monitoring to identify failures of asml euv system parts.

[31] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.