An Innovative Framework for Next Activity Prediction Using Process Entropy and Dynamic Attribute-Wise Transformer for Business Process Monitoring

by

Hadi Zare
Bachelor of Science, Sharif University of Technology, 2017
Master of Business Administration, University of Tehran, 2022

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

We acknowledge and respect the Lək'ʷəŋən (Songhees and Xʷsepsəm/Esquimalt) Peoples on whose territory the university stands, and the Lək'ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

An Innovative Framework for Next Activity Prediction Using Process Entropy and Dynamic Attribute-Wise Transformer for Business Process Monitoring

by

Hadi Zare
Bachelor of Science, Sharif University of Technology, 2017
Master of Business Administration, University of Tehran, 2022

Supervisory Committee

Dr. H. Najjaran, Supervisor
(Department of Electrical and Computer Engineering)

Dr. A. Thomo, Departmental Member
(Department of Computer Science)

ABSTRACT

In the field of Business Process Management (BPM), accurately predicting the next activity in an ongoing process is critical for improving operational efficiency, optimizing resource allocation, and enabling proactive decision-making. Although recent advances in machine learning (ML) and artificial intelligence (AI) have significantly improved predictive performance, several key challenges remain. These include: (i) the limited transparency of deep learning models when applied to datasets that may not require such complexity; (ii) the reliance on trial-and-error methods for selecting suitable models across diverse event logs; and (iii) the underutilization of attributes that can carry valuable information for prediction.

Building on these identified gaps, this thesis is motivated by two central research questions: (1) How can predictive models be designed to effectively capture the complexity and variability inherent in modern event logs? and (2) How can organizations systematically determine the most suitable predictive models for their specific process characteristics?

To address the first question, this thesis introduces a novel predictive architecture called the Dynamic Attribute-Wise Transformer (DAW-Transformer). The model enhances predictive capability by extending the standard transformer architecture through the integration of multi-head attention and a dynamic windowing mechanism tailored to each dataset. This design captures long-range dependencies across multiple event attributes, offering a richer and more detailed representation of process behavior and improving the model's ability to generalize across heterogeneous logs.

To address the second question, an Entropy-Driven Model Selection Framework is proposed. This framework employs process entropy as a quantitative indicator of event log complexity, enabling adaptive model selection that balances predictive accuracy and interpretability. By aligning model choice with dataset variability, it overcomes the limitations of existing approaches that apply a single predictive model indiscriminately across all process types, regardless of their structural diversity.

The effectiveness of the proposed methods is validated through comprehensive experiments on six publicly available event logs across domains such as healthcare, logistics, and public administration. Results demonstrate that the DAW-Transformer

achieves superior performance, particularly on high-entropy processes, where activities exhibit greater variability, while interpretable models such as Decision Trees perform competitively on low-entropy, structured processes. These findings highlight the value of aligning model complexity with process entropy and underscore entropy's role as a guiding principle for model selection in predictive business process monitoring.

In summary, this thesis contributes to: (1) improving interpretability and reducing trial-and-error model selection via an Entropy-Driven Model Selection Framework; (2) mitigating attribute underutilization through the DAW-Transformer; and (3) conducting comprehensive evaluations across diverse datasets. Together, these contributions enhance predictive accuracy, strengthen interpretability, and establish a data-driven foundation for adaptive model choice.

Contents

List of Tables

List of Figures

Acronyms

ACIS Advanced Control and Intelligent Systems

AI Artificial Intelligence

Bi-LSTM Bidirectional Long Short-Term Memory

BPI Business Process Intelligence

BPM Business Process Management

CNN Convolutional Neural Network

DAW Dynamic Attribute-Wise

DENAP Data-Aware Explainable Next Activity Prediction

DL Deep Learning

ERP Enterprise Resource Planning

K Key

LRP Layer-Wise Relevance Propagation

LSTM Long Short-Term Memory

MiFTMA Multi-View Information Fusion Method

ML Machine Learning

MTLFormer Multi-Task Learning Guided Transformer

NLP Natural Language Processing

NSERC Natural Sciences and Engineering Research Council of Canada

Q Query

RNN Recurrent Neural Network

ReLU Rectified Linear Unit

SLA Service Level Agreement

SVM Support Vector Machine

V Value

XGBoost Extreme Gradient Boosting

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Najjaran for his invaluable mentoring, continuous support, thoughtful encouragement, and remarkable patience throughout the course of my research and graduate studies.

My wife, Bahare, for her unwavering love, immense patience, and constant encouragement. She has been by my side throughout this entire journey, and this achievement would not have been possible without her incredible support. I am also deeply grateful to my family for their lifelong support and for always believing in me.

I am also deeply grateful to Mostafa Abbasi and Maryam Ahang. As teammates and friends, they stood beside me throughout this journey, and I sincerely appreciate their help, collaboration, and support.

The Natural Sciences and Engineering Research Council of Canada (NSERC) for generously supporting my studies through a scholarship.

Finally, I extend my thanks to all members of the Advanced Control and Intelligent Systems (ACIS) Laboratory for fostering a collaborative, inspiring, and supportive research environment.

Chapter 1

Introduction

Business Process Management (BPM) is a systematic approach to identifying, analyzing, and improving business processes to enhance efficiency, consistency, and customer satisfaction [65]. It focuses on mapping key activities, establishing clear metrics, and continuously optimizing workflows to align with organizational goals. BPM integrates people, technology, and processes to create a structured framework that ensures accountability, reduces inefficiencies, and fosters a culture of continuous improvement. By emphasizing cross-functional collaboration and data-driven decision-making, BPM helps organizations streamline operations, improve quality, and adapt to changing business environments, ultimately driving long-term success [38, 17].

Within BPM, Process Mining has emerged as a crucial technique that extracts information from event logs to help organizations monitor, analyze, and optimize their operations [12, 53]. Among its many applications, next activity prediction plays a critical role by forecasting the continuation of ongoing or incomplete cases. Accurate predictions enable proactive resource allocation, workflow optimization, and early identification of deviations, which are particularly valuable in domains such as healthcare, manufacturing, supply chain management, and customer service [16, 50].

1.1 Motivation

In today's competitive and dynamic environment, anticipating future actions within business processes is a critical success factor. Accurate predictions of upcoming activities enable proactive decision-making, reducing operational costs, improving efficiency, and preventing disruptions. For example, in supply chain management, customer support, healthcare, and warehousing, predicting the next likely event supports optimized resource allocation, minimizes delays, and facilitates early identification of potential risks.

Business Process Management (BPM) provides the foundation for such predictive capabilities by analyzing historical event logs to monitor, evaluate, and improve process performance. Among the predictive tasks in BPM, next activity prediction is particularly important, as it enables workflow automation, supports data-driven decisions, and improves organizational responsiveness to real-time changes.

Machine learning (ML) and deep learning (DL) models have significantly advanced next activity prediction. However, many existing approaches struggle to capture the full range of attributes and activities across an entire process trace. A second fundamental challenge is the trade-off between interpretability and performance: simple models provide transparency but often lack accuracy on complex datasets, whereas deep models achieve higher accuracy but at the cost of higher computational overhead and reduced interpretability. Without systematic model selection, organizations risk deploying either underpowered or unnecessarily complex models, resulting in inefficiencies.

This thesis is motivated by the need for a flexible prediction framework that adapts model complexity to process complexity. To achieve this, process entropy is introduced as a quantitative measure of variability and uncertainty in event logs. Entropy serves as a guiding principle for dynamic model selection, ensuring that interpretable models are applied to structured, low-entropy processes, while more advanced architectures are reserved for high-entropy, variable processes. In addition, this thesis contributes a novel architecture, the DAW-Transformer, which extends the standard Transformer model by incorporating dynamic windowing and multi-

attribute embeddings.

Finally, in high-stakes domains such as healthcare, finance, and public administration, explainability is as important as accuracy. Stakeholders require models that not only predict reliably but also justify their outcomes. By proposing both a novel deep learning model (DAW-Transformer) and an entropy-driven selection framework, this research bridges the gap between predictive performance and interpretability, offering a principled approach to trustworthy predictive business process monitoring.

While the above challenges emphasize the general need to balance accuracy and interpretability, they also reveal specific shortcomings in how current models handle event logs. Two issues are particularly prominent: first, the limited use of process attributes and inadequate consideration of the full temporal dimension of event traces; and second, the declining transparency of increasingly complex algorithms. These concerns are elaborated in the following subsections.

1.1.1 Considering Incomplete Event Log Attributes

Most existing approaches to next activity prediction in Business Process Management (BPM) primarily rely on a limited subset of event log attributes typically the case ID, activity label, and timestamp. While these attributes capture the core structure of process traces, they often omit contextual and domain-specific variables that could significantly influence the course of a business process. For instance, in healthcare datasets such as Sepsis, attributes like patient age, gender, test results, or the resource (e.g., medical personnel) involved in an activity provide crucial insights that can affect both process flow and outcomes.

In addition to ignoring these rich contextual features, many models utilize a fixed-length sliding window, usually comprising only the last few event logs. This restricts the model's ability to learn from the complete historical sequence, which is particularly important in processes where earlier events heavily influence future decisions. Ignoring these long-range dependencies limits the capacity of predictive models to fully understand process dynamics and make informed, accurate predictions.

Therefore, one of the key motivations of this research is to create a more com-

prehensive modeling approach that incorporates all available attributes and considers the full event history using a dynamic windowing mechanism.

1.1.2 Lack of Transparency in Some ML Algorithms

As machine learning and deep learning models become more powerful, their interpretability often decreases. In particular, complex models like deep neural networks and Transformers, although highly accurate, act as black boxes, making it difficult for stakeholders to understand the rationale behind predictions. This lack of transparency can hinder trust and adoption, especially in critical domains like healthcare, finance, and legal processes, where decision-makers need clear justifications to act on predictive insights.

In contrast, traditional models such as decision trees offer intuitive explanations and visualizations, enhancing their usability and acceptability. However, these simpler models often fail to capture non-linear and high-dimensional patterns, limiting their performance on complex datasets.

This trade-off between accuracy and interpretability poses a significant challenge. In this research, we address this issue by introducing an entropy-based model selection framework that dynamically recommends simpler or more complex models based on the dataset's complexity. This ensures that interpretability is prioritized where possible without sacrificing predictive power when necessary.

1.2 Objective

To address the primary concerns identified in this research, several objectives are recognized and systematically explored throughout this thesis. These objectives are designed to advance the field of next activity prediction in Business Process Management (BPM) by addressing key methodological limitations and practical challenges encountered in real-world applications. Together, they form a cohesive roadmap toward the development of a more adaptive, interpretable, and high-performing predictive framework.

To achieve this goal, the research is divided into five main objectives. Each objective focuses on a specific part of the study, starting from understanding process complexity, then designing and testing the new model, and finally comparing results and improving interpretability. The following sections describe each objective in detail.

1.2.1 Develop an entropy-based framework to quantify the complexity of business process event logs and guide the selection of appropriate predictive models

Current model selection in predictive process monitoring relies heavily on trial-and-error and lacks a principled way to account for process complexity. To address this limitation, this research develops an entropy-driven framework that quantifies the complexity of event logs and uses this measure to guide model selection. This approach enables systematic differentiation between high- and low-entropy processes and results in the selection of models that better balance predictive accuracy, computational efficiency, and interpretability.

1.2.2 Design and implement the new Transformer architecture to improve next activity prediction by capturing long-range dependencies and enabling full trace utilization

Conventional Transformer-based models for next activity prediction are limited by fixed sliding windows and incomplete utilization of event traces. To overcome these limitations, this research proposes and implements the DAW-Transformer, which incorporates dynamic windowing and multi-head self-attention to capture long-range dependencies across full traces and multiple attributes. This design improves representation learning and leads to enhanced predictive performance on complex business process datasets.

1.2.3 Evaluate the relationship between process entropy and predictive model performance across multiple datasets

There is limited empirical understanding of how process complexity affects the performance of different predictive models. To address this gap, this research systematically evaluates multiple models across datasets with varying entropy levels. The analysis identifies relationships between entropy and model effectiveness, providing evidence-based guidelines for selecting appropriate models under different process conditions.

1.2.4 Compare the performance of the DAW-Transformer with state-of-the-art models using accuracy, precision, recall, and F1-score

Existing approaches lack comprehensive validation of new architectures against diverse baseline models under varying data complexities. To ensure rigorous evaluation, this research compares the DAW-Transformer with state-of-the-art models, including CNN-LSTM, CNN-BiLSTM, Vanilla Transformer, Random Forest, Decision Tree, and XGBoost, using metrics such as accuracy, precision, recall, and F1-score. The results demonstrate the conditions under which the proposed model outperforms existing methods, establishing its practical effectiveness.

1.2.5 Develop an adaptive model selection strategy that balances predictive performance and interpretability based on process entropy

High-performing predictive models are often complex and difficult to interpret, limiting their practical adoption. To address this challenge, this research investigates an adaptive model selection strategy based on entropy, where simpler models are applied to low-entropy processes and more complex models to high-entropy processes. This approach improves interpretability without sacrificing performance, enabling better alignment between model complexity and process characteristics and increasing stake-

holder trust.

1.3 Contributions

To address the discussed objectives, this thesis provides several key contributions in the context of next activity prediction in BPM. These contributions aim to improve prediction accuracy, model adaptability, and practical applicability in real-world process settings. The contributions can be categorized as follows:

1. **Entropy-Driven Model Selection Framework**
   This thesis introduces a novel model selection approach based on process entropy, which quantifies the complexity and variability of event logs. By calculating Shannon entropy for each dataset, the method dynamically recommends either interpretable models for low-entropy datasets or more complex deep learning models for high-entropy datasets. This adaptive strategy enhances the trade-off between accuracy and interpretability, optimizing model usage for different business scenarios.

2. **Development of the DAW-Transformer Architecture**
   A new Transformer-based model, named DAW-Transformer (Dynamic Attribute-Wise Transformer), is proposed to improve next activity prediction. Unlike traditional models that rely on fixed sliding windows and overlook some of attributes, the DAW-Transformer integrates multi-head self-attention and a dynamic windowing mechanism to capture long-range dependencies and leverage the full trace history along with all available features.

3. **Comprehensive Evaluation Across Diverse Datasets**
   The proposed methods are evaluated using six publicly available datasets representing both high- and low-entropy business processes. The DAW-Transformer is benchmarked against state-of-the-art models such as CNN-LSTM, CNN-BiLSTM, Vanilla Transformer, XGBoost, Random Forest, and Decision Trees. The results demonstrate the effectiveness of the proposed model in handling complex sequences.

1.4 Agenda

This thesis is structured into several chapters. An outline of these chapters is provided below:

Chapter 2: Background and Preliminaries

Provides an overview of the existing work in the fields of process mining, next activity prediction, process entropy, deep learning in business process management, and model interpretability. It critically examines related methods and highlights the gaps that this thesis aims to address.

Chapter 3: DAW-Transformer Methodology

Details the proposed entropy-based model selection framework and introduces the novel DAW-Transformer. The chapter elaborates on the theoretical foundations, including entropy computation, dynamic windowing, and model selection logic.

Chapter 4: Entropy-Driven Model Selection Framework Methodology

Describes the entropy-based evaluation framework used to quantify process complexity. This chapter explains the computation of conditional entropy from activity transitions and outlines how this metric informs the selection of predictive models. It also details the data preprocessing pipeline, including feature encoding, and trace reconstruction.

Chapter 5: Conclusion and Future Work

Summarizes the research contributions, reflects on the limitations, and suggests directions for future research in adaptive and interpretable predictive process monitoring.

Chapter 2

Background and Literature Review

In this chapter, the basic concepts in next activity prediction and entropy are explained, and the related literature in these areas is discussed.

2.1 Background

2.1.1 Event Logs

Event logs serve as structured repositories that systematically capture and store information about the execution of events within a process. Each record in an event log generally contains details about the type of event, the corresponding timestamp, and other contextual attributes collected during the execution of modern industrial systems, information systems, and machines. These logs provide an essential foundation for process mining, as they allow researchers and practitioners to trace the exact sequence of activities carried out in a process, thereby offering insights into how processes are performed in practice as opposed to how they are designed.

The significance of event logs lies in their ability to support predictive analysis, performance optimization, and proactive interventions. By analyzing the historical data embedded in event logs, organizations can identify patterns, deviations, and bottlenecks, ultimately enabling them to enhance operational efficiency, allocate resources more effectively, and ensure higher reliability of system performance [28].

Table 2.1 provides an illustrative example drawn from the BPIC_2012_A event

logs, which have been widely used in process mining research. This sample highlights the essential components of an event record. Specifically, the CaseID uniquely identifies an individual process instance, allowing the tracking of end-to-end execution flows. The Activities column denotes the actions or process steps executed as part of the workflow. Finally, the Timestamps attribute specifies the exact time at which each event occurs, thereby preserving the temporal order and enabling chronological analysis of process behavior. Together, these elements form the fundamental building blocks of event log data and demonstrate how raw execution data can be transformed into a valuable source of knowledge for predictive business process monitoring.

Table 2.1: A sample event log of the BPIC_2012_A event logs.

Case ID	Activity	Timestamp	Lifecycle	Amount Requested	org:resource	REG_DATE
173688	A_SUBMITTED	2011-10-01 00:38:44	COMPLETE	20,000	112	2011-10-01
173688	A_PARTLYSUBMITTED	2011-10-01 00:38:44	COMPLETE	20,000	112	2011-10-01
173688	A_PREACCEPTED	2011-10-01 00:39:37	COMPLETE	20,000	112	2011-10-01
173688	A_ACCEPTED	2011-10-01 11:42:43	COMPLETE	20,000	10862	2011-10-01
173688	A_FINALIZED	2011-10-01 11:45:09	COMPLETE	20,000	10862	2011-10-01
173688	A_REGISTERED	2011-10-13 10:37:29	COMPLETE	20,000	10629	2011-10-01
173688	A_APPROVED	2011-10-13 10:37:29	COMPLETE	20,000	10629	2011-10-01
173688	A_ACTIVATED	2011-10-13 10:37:29	COMPLETE	20,000	10629	2011-10-01

2.1.2 Trace

Within event logs, each process is composed of multiple independent cases. A case is defined as a complete instance of a process from initiation to completion, consisting of an ordered sequence of recorded events. In the literature, this sequence is frequently referred to as a trace. A trace thus captures the exact execution path that a particular case followed within the process. Formally, a trace T can be denoted as:

$$T = \langle e_1, e_2, e_3, \ldots, e_n \rangle, \tag{2.1}$$

where each element e_i represents a recorded event at position i in the sequence.

Each event e_i is typically associated with several attributes, the nature and number of which may vary depending on the event log under consideration [7]. At a minimum, events often include attributes such as the activity name (specifying the task performed), timestamp (capturing the moment of execution), and case identifier

(linking the event to a particular process instance). In more complex logs, additional attributes may be present, including the resource responsible for executing the task, organizational roles, costs, or lifecycle transitions (e.g., start, complete).

Traces are central to process mining because they provide the basis for reconstructing how processes unfold in reality. By comparing and analyzing multiple traces, analysts can discover common process models, identify variations and deviations, and gain insights into structural patterns or anomalies. Furthermore, traces allow for the examination of process behavior at different levels of granularity—ranging from the analysis of individual paths to aggregated statistical measures of process performance. In this study, traces extracted from event logs form the primary input for predictive modeling, serving as sequential data on which entropy, next-activity prediction, and other analyses are performed.

2.1.3 Process Entropy

Process entropy is a measure that quantifies the level of uncertainty or unpredictability inherent in the execution of a business process [31]. In practical terms, processes with high entropy are characterized by a greater degree of variability in the order and type of activities performed. This unpredictability poses significant challenges for workflow management tasks such as scheduling, resource allocation, and performance optimization, as the lack of stability often results in inefficient utilization of human and computational resources. On the other hand, processes with low entropy exhibit more deterministic behavior, allowing managers and decision-makers to plan with greater confidence and efficiency.

The concept of entropy has its roots in information theory, where it was formally introduced by Claude Shannon to represent the average amount of uncertainty or information content in a random variable [31]. This measure, commonly known as Shannon's entropy, is mathematically expressed as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log(p(x_i)). \tag{2.2}$$

In this formulation, X is a discrete random variable that can take on possible val-

ues $x_1, x_2, \ldots, x_n$, each occurring with probabilities $p(x_1), p(x_2), \ldots, p(x_n)$. For each i, the probabilities satisfy $p(x_i) \geq 0$ and the normalization condition $\sum_{i=1}^{n} p(x_i) = 1$. Entropy therefore provides a quantitative way of assessing the unpredictability of outcomes in a probabilistic system.

In the context of process mining, we are particularly interested in the conditional entropy, which refines this idea by measuring the uncertainty of one random variable given knowledge of another. More specifically, when predicting the next activity in a business process, conditional entropy quantifies the degree of uncertainty that remains about the next step once the current activity is known. It is defined as:

$$H(Y \mid X) = -\sum_{x \in X} p(x) \sum_{y \in Y} p(y \mid x) \log_2 p(y \mid x). \tag{2.3}$$

Here, X represents the current activity and Y the subsequent activity. The conditional probability $p(y \mid x)$ denotes the likelihood that activity y follows activity x. When the conditional entropy is low, the next activity can be predicted with high confidence, reflecting a structured and relatively rigid process. Conversely, high conditional entropy suggests that many different activities could follow a given step, indicating a flexible, less predictable, or even disordered process.

In this study, we compute conditional entropy values directly from the event log data by extracting transition frequencies between successive activities. This approach provides a data-driven way of characterizing the degree of uncertainty and variability in business processes, and the results are further utilized in the methodology section to compare processes of different entropy levels.

2.1.4 Machine Learning and Deep Learning Models

Transformer Architecture

The Transformer, introduced by Vaswani et al. [57], represents a fundamental shift in sequence modeling and transduction tasks. By replacing recurrence and convolution entirely with attention mechanisms, it achieves superior scalability, parallelization, and the ability to model long-range dependencies more effectively than Recurrent

Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). This innovation has become the foundation for many state-of-the-art models in natural language processing, time series forecasting, and other sequence-related domains.

Motivation for the Transformer

Prior to the Transformer, most sequence-to-sequence models were built on RNNs (including LSTMs and GRUs), which process tokens sequentially. This sequential nature limits parallelization and makes modeling long-term dependencies difficult due to vanishing gradients and long information paths. CNN-based approaches introduced some parallelism but still suffered from fixed receptive fields and hierarchical constraints. The Transformer addresses these limitations by:

- Removing recurrence, enabling full parallel processing of sequences.
- Using self-attention to provide direct paths between all input positions, reducing the path length between any two tokens.
- Allowing flexible modeling of dependencies at arbitrary distances.

Overall Architecture

The Transformer follows the encoder–decoder paradigm (Figure 2.1). The encoder takes an input sequence $x = (x_1, x_2, \ldots, x_n)$ and outputs contextualized representations $z = (z_1, z_2, \ldots, z_n)$.

The **decoder** generates the output sequence $y = (y_1, y_2, \ldots, y_m)$ autoregressively, attending both to previously generated outputs and to encoder outputs. Both encoder and decoder are stacks of N identical layers (6 in the base model).

Encoder Layers. Each encoder layer has two main components:

1. Multi-head self-attention mechanism.
2. Position-wise feed-forward network.

Residual connections and layer normalization are applied after each sub-layer, stabilizing gradients and improving convergence.

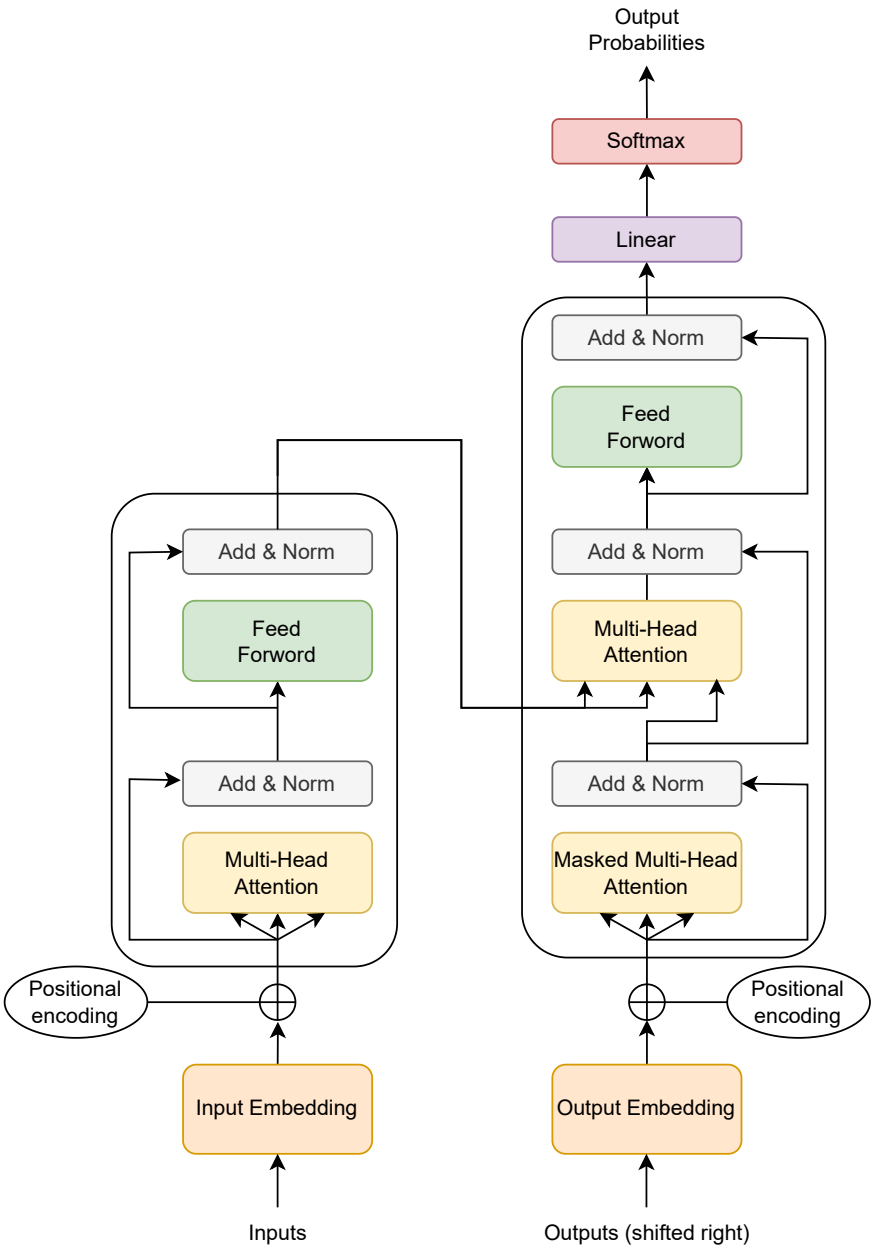


Figure 2.1: Transformer Architecture, reproduced by [57]

Decoder Layers. Each decoder layer contains three components:

1. Masked multi-head self-attention (prevents attending to future positions).
2. Encoder–decoder attention (attends to encoder outputs).
3. Position-wise feed-forward network.

The masking mechanism enforces autoregressive decoding, ensuring each position attends only to known previous outputs.

Scaled Dot-Product Attention

The core operation of the Transformer is the scaled dot-product attention, which maps a query Q, keys K, and values V to an output:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.4}$$

where:

- $Q \in \mathbb{R}^{n_q \times d_k}$: queries,
- $K \in \mathbb{R}^{n_k \times d_k}$: keys,
- $V \in \mathbb{R}^{n_k \times d_v}$: values,
- d_k: key dimensionality.

The scaling factor $\sqrt{d_k}$ prevents large dot products from saturating the softmax, which would otherwise yield vanishing gradients. This operation lets each position attend to all positions in the input, weighting them adaptively.

Multi-Head Attention

A single attention head may not capture all relevant relationships. The Transformer therefore uses *multi-head attention*, which projects Q, K, and V into h different subspaces via learned linear projections:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \tag{2.5}$$

and concatenates their outputs:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O. \tag{2.6}$$

Here W_i^Q, W_i^K, W_i^V, W^O are learned matrices. In the base Transformer, $h = 8$, $d_k = d_v = 64$, and $d_{\text{model}} = 512$. Each head can specialize in different types of relationships (e.g., syntactic, positional, or contextual), providing a richer representation than a single attention mechanism.

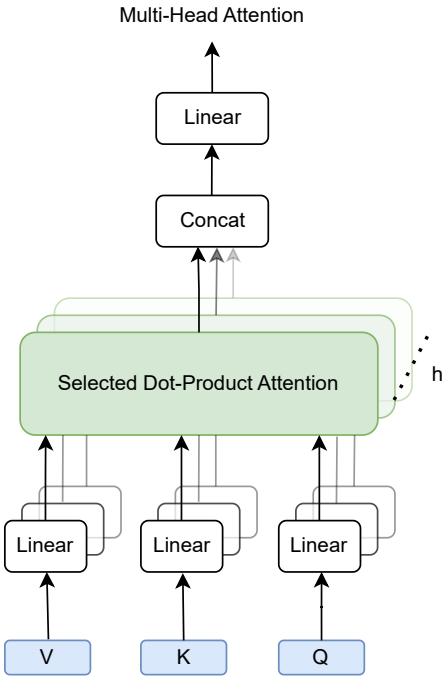


Figure 2.2: Overview of the Multi-Head Attention mechanism, reproduced by [57]

Position-Wise Feed-Forward Networks

Following attention, each encoder and decoder layer contains a fully connected feed-forward network (applied identically and independently to each position):

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2, \tag{2.7}$$

with an inner dimensionality $d_{\mathrm{ff}} = 2048$ in the base model. These layers expand model capacity and introduce nonlinearity between attention operations.

Positional Encoding

Since self-attention is permutation-invariant and lacks any notion of order, the Transformer adds *positional encodings* to input embeddings to inject sequence order information. Vaswani et al. [57] proposed sinusoidal positional encodings:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\mathrm{model}}}}\right), \tag{2.8}$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\mathrm{model}}}}\right). \tag{2.9}$$

These encodings allow the model to extrapolate to sequence lengths not seen during training and represent relative positions as linear functions of absolute positions.

Layer Normalization and Residual Connections

Each sub-layer is wrapped with residual connections followed by layer normalization:

$$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x)). \tag{2.10}$$

This stabilizes training, mitigates exploding/vanishing gradients, and allows for deeper networks without degradation.

Advantages over RNN and CNN Architectures

Vaswani et al. [57] highlight several benefits of self-attention:

- **Shorter paths:** Self-attention connects all positions directly, reducing the path length between any two tokens to $O(1)$ compared to $O(n)$ in RNNs.
- **Parallelization:** All tokens can be processed simultaneously, unlike the inherently sequential nature of RNNs.
- **Reduced computational complexity for typical NLP sizes:** With sequence length $n < d_{\text{model}}$, self-attention is more efficient than recurrent layers.

CNN–LSTM Models

Convolutional Neural Network–Long Short-Term Memory (CNN–LSTM) models represent a hybrid deep learning architecture that integrates convolutional and recurrent layers to capture both spatial and temporal dependencies in sequential data. The CNN component is designed to automatically extract high-level spatial or local features from raw input sequences. Through convolution and pooling operations, CNN layers identify relevant correlations and reduce noise, producing compact feature maps that emphasize important patterns in the data. These extracted features are subsequently fed into an LSTM network, which excels at modeling long-range temporal dependencies by mitigating the vanishing gradient problem inherent in traditional recurrent neural networks (RNNs) [27].

The integration of CNNs and LSTMs is particularly effective for multivariate time-series data, where the CNN layers act as a feature extractor across multiple input dimensions, and the LSTM layers model the dynamic evolution of these features over time. This dual-stage learning process allows the CNN–LSTM to capture both short-term local variations and long-term sequential patterns, which are often critical in applications. In energy demand forecasting, for example, CNN–LSTM models have been applied to extract interdependencies among variables [36].

Formally, the CNN layers transform the input sequence $X \in \mathbb{R}^{n \times d}$, where n denotes the sequence length and d the number of features, into a set of higher-level feature maps F. These feature maps are then provided as sequential inputs to the LSTM, which computes hidden states h_t by recursively updating its cell state through

input, forget, and output gates:

$$h_t, c_t = \text{LSTM}(F_t, h_{t-1}, c_{t-1}), \tag{2.11}$$

where c_t is the memory cell that preserves long-term dependencies. This combined architecture thus unifies CNN's spatial learning with LSTM's temporal modeling capacity.

Overall, CNN–LSTM models leverage the complementary strengths of CNNs and LSTMs, making them a powerful tool for predictive modeling in domains where both spatial correlations and sequential dependencies are significant. They have been widely applied in natural language processing, speech recognition, computer vision, and time-series forecasting, consistently showing superior performance over traditional methods.

CNN–BiLSTM Models

An important extension of the CNN–LSTM architecture is the Convolutional Neural Network– Bidirectional Long Short-Term Memory (CNN–BiLSTM) model. While CNN–LSTM models capture temporal dependencies in a unidirectional manner—using only past information to predict future values—the BiLSTM component enhances this capability by processing the input sequence in both forward and backward directions [25]. This allows the network to leverage both past and future context, resulting in richer temporal representations.

In the CNN–BiLSTM framework, convolutional layers first extract spatial features and local patterns from the input data, reducing dimensionality and filtering noise. These features are then fed into a BiLSTM, which consists of two LSTMs: one moving from $t = 1$ to $t = n$ (forward pass) and the other from $t = n$ to $t = 1$ (backward pass). The hidden states from both directions are concatenated:

$$h_t = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right], \tag{2.12}$$

where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ represent the forward and backward hidden states, respectively.

By combining these states, the BiLSTM captures dependencies that may not be observable when considering only one temporal direction.

This bidirectional modeling is particularly valuable in tasks where future context influences present predictions. For instance, in natural language processing, the meaning of a word depends on both preceding and succeeding words [25]. Similarly, in time-series prediction problems such as energy forecasting, healthcare monitoring, or traffic analysis, patterns may be influenced by both earlier and later events within the sequence. CNN–BiLSTM models therefore often achieve higher accuracy than standard CNN–LSTM models when full sequence information is available [26, 49].

Overall, CNN–BiLSTM models combine the feature extraction capability of CNNs with the context-aware sequence modeling of BiLSTMs. By integrating information from both past and future contexts, they provide a more comprehensive representation of sequential data and have shown superior performance in many real-world applications where bidirectional dependencies are crucial.

Decision Trees

Decision trees are one of the most fundamental and widely applied supervised learning algorithms in both statistics and machine learning [48]. A decision tree partitions the input feature space into a set of disjoint regions by recursively splitting the data according to feature values. Each internal node represents a test on a feature, branches correspond to possible outcomes of the test, and terminal nodes (leaves) represent a predicted outcome, a class label in classification or a numerical value in regression. This hierarchical structure allows decision trees to approximate complex, nonlinear decision boundaries while remaining interpretable.

The construction of a decision tree generally follows three main steps:

Splitting – At each node, the data is divided into subsets according to a selected attribute and a splitting criterion. Popular measures for evaluating splits include the Gini index, entropy/information gain, gain ratio, and classification error. These measures assess the "purity" of resulting nodes, aiming to maximize homogeneity with respect to the target variable. The splitting process is repeated recursively, creating branches until certain conditions are met.

Stopping – To prevent the tree from growing indefinitely and capturing noise, stopping rules are introduced. These may specify the minimum number of samples required in a node before a split is attempted, the minimum number of samples in a leaf node, or the maximum depth of the tree. Balancing these constraints is crucial, as overly complex trees risk overfitting, while overly simplistic ones may underfit and miss important patterns.

Pruning – Even with stopping rules, decision trees can still become excessively complex. Pruning addresses this issue by removing branches that add little predictive value. This can be done in two ways: pre-pruning, where statistical tests (e.g., chi-square) prevent unnecessary splits during tree growth, or post-pruning, where a fully grown tree is simplified by eliminating weak branches using a validation set or cross-validation. The goal is to enhance the generalizability of the model.

The foundation of modern decision trees is the Classification and Regression Trees (CART) framework introduced by Breiman et al. [10]. CART defines splitting criteria for both classification and regression. For classification, the Gini impurity is commonly used to measure the heterogeneity of a node:

$$Gini(t) = 1 - \sum_{k=1}^{K} p_k^2, \tag{2.13}$$

where p_k is the proportion of samples of class k in node t, and K is the total number of classes. A split is chosen to maximize the reduction in impurity. For regression, splits are selected to minimize the variance of the target variable within child nodes. The reduction in variance at a node t is defined as:

$$\Delta Var = Var(t) - \left(\frac{N_L}{N} Var(t_L) + \frac{N_R}{N} Var(t_R) \right), \tag{2.14}$$

where $Var(t)$ is the variance of the target variable in node t, t_L and t_R are the left and right child nodes, and N_L, N_R, and N denote the number of samples in the left child, right child, and parent node, respectively.

A tree is built recursively: starting from the root node containing the entire dataset, the algorithm selects the feature and threshold that yield the best split ac-

cording to the chosen criterion. This process continues until a stopping condition is reached, such as a maximum tree depth, a minimum number of samples per node, or the absence of further improvement in the splitting criterion. The resulting tree provides a piecewise constant approximation of the underlying function mapping inputs to outputs.

Decision trees offer several key advantages. First, they are highly interpretable, as each path from root to leaf can be translated into a simple set of if–then rules. This transparency makes them suitable for domains requiring explainable decision-making, such as healthcare, law, and finance. Second, they can naturally handle both numerical and categorical features, missing values, and nonlinear relationships without requiring complex preprocessing.

Despite these strengths, decision trees have notable limitations. They are prone to high variance, meaning that small perturbations in the training data may result in very different trees. They also tend to overfit when grown to full depth, capturing noise rather than general patterns. To mitigate these issues, pruning techniques can be applied to remove branches that do not contribute substantially to predictive performance. Furthermore, ensemble methods such as bagging and random forests have been developed to aggregate many trees, thereby improving predictive accuracy and robustness. Breiman [8, 9] demonstrated that random forests—ensembles of randomized decision trees—consistently outperform single trees while retaining much of their interpretability.

From a theoretical perspective, decision trees can be viewed as adaptive partitioning estimators. They divide the feature space into hyperrectangles, within which predictions are made by averaging the outputs (in regression) or majority voting (in classification). This connects them to nonparametric methods such as nearest-neighbor and kernel estimators [18, 5]. However, their recursive and data-dependent construction makes formal statistical analysis challenging [6].

In summary, decision trees provide an interpretable and flexible approach to classification and regression. Their recursive structure allows them to capture nonlinear patterns effectively, but their instability and tendency to overfit limit their standalone performance. Nevertheless, as the building blocks of powerful ensemble methods, de-

cision trees remain central to modern machine learning.

Random Forests

Random forests, introduced by Breiman [9], are one of the most influential ensemble learning methods in modern machine learning. They are built upon the principle that aggregating multiple weak or moderately strong learners can significantly improve predictive performance and robustness. A random forest constructs an ensemble of decision trees, each trained on a bootstrap sample of the data, and combines their predictions through majority voting (for classification) or averaging (for regression).

The construction of a random forest involves two key sources of randomness:

1. **Bootstrap sampling (bagging):** Each tree is trained on a bootstrap sample, i.e., a dataset obtained by sampling with replacement from the original training data. This introduces variability among the trees and reduces variance through aggregation.

2. **Random feature selection:** At each split in a tree, instead of considering all available features, the algorithm selects a random subset of m_{try} features and determines the best split only among them. This decorrelates the trees and ensures that individual strong predictors do not dominate every split.

Random forests possess several advantages over individual decision trees:

Reduced variance: By averaging across many trees, random forests significantly reduce the instability of single decision trees.

Strong generalization: They achieve high predictive accuracy across a wide variety of domains without extensive parameter tuning.

Robustness: Random forests are resilient to noise, outliers, and overfitting, particularly in high-dimensional datasets.

Variable importance: They provide measures of feature importance, such as mean decrease in impurity and permutation importance, which are widely used for interpretability.

Despite their empirical success, understanding the theoretical properties of random forests remains challenging due to their complex, data-dependent structure. Biau

and Scornet [6] review statistical results showing that random forests can be interpreted as adaptive partitioning estimators, where the feature space is recursively split into hyperrectangles and predictions are made locally within them. Under certain conditions, consistency results have been established, suggesting that random forests can asymptotically approximate the Bayes predictor. Nevertheless, providing a full mathematical characterization of their performance remains an open problem.

Since their introduction, random forests have become a standard tool across scientific and industrial domains. They have been applied in bioinformatics, remote sensing, finance, ecology, and medical diagnostics, among others. Their combination of high accuracy, robustness, and interpretability has made them one of the most widely adopted machine learning algorithms.

In summary, random forests extend decision trees through ensemble learning, combining bagging and random feature selection to create a robust, accurate, and interpretable model. Their empirical success and widespread applicability, together with ongoing theoretical research, confirm their central role in both the practice and study of modern machine learning.

XGBoost

XGBoost (eXtreme Gradient Boosting) is a scalable and regularized implementation of gradient boosted decision trees [14]. It builds an ensemble of regression trees in a stagewise manner to minimize a differentiable loss while incorporating regularization to control model complexity.

The regularized objective is:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k), \qquad \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2, \tag{2.15}$$

where ℓ is the training loss, $\hat{y}_i$ the prediction, T the number of leaves in tree f, and w the leaf weights.

At boosting round t, a second-order Taylor expansion of the loss yields:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \tfrac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t), \tag{2.16}$$

with first and second derivatives $g_i = \partial_{\hat{y}} \ell(y_i, \hat{y}_i)$, $h_i = \partial^2_{\hat{y}} \ell(y_i, \hat{y}_i)$. For a leaf j containing instances I_j, the optimal weight is:

$$w_j^\star = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}, \tag{2.17}$$

and the corresponding score:

$$\tilde{\mathcal{L}}^{(t)}(q) = -\tfrac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T. \tag{2.18}$$

XGBoost employs greedy split finding, where the gain of splitting a node into left (L) and right (R) children is:

$$\text{Gain} = \tfrac{1}{2} \left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right) - \gamma. \tag{2.19}$$

In addition to its algorithmic design, XGBoost integrates system-level optimizations such as histogram-based split search, sparsity-aware learning, out-of-core computation, and parallel/GPU training, making it widely used in practice for large-scale machine learning tasks.

2.1.5 Evaluation Metrics

Accuracy

Accuracy measures the overall proportion of correctly predicted events across all classes [23, 34]. It reflects the general ability of the model to classify events correctly but may overestimate performance in the presence of class imbalance. For instance, if one activity dominates the dataset, a model biased toward predicting that activity may achieve high accuracy while failing to capture less frequent but critical classes.

Let tp_i, fp_i, tn_i, and fn_i denote the true positives, false positives, true negatives, and false negatives for class i, respectively. Let s_i be the number of events in class i and $n = \sum_{i=1}^{l} s_i$ the total number of events in the dataset. The weighted average

accuracy is computed as:

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i + tn_i}{tp_i + tn_i + fp_i + fn_i}. \tag{2.20}$$

Precision

Precision quantifies the proportion of correctly predicted events for a given class among all events predicted as belonging to that class [24]. High precision indicates that when the model predicts an activity, it is usually correct. This metric is especially important in domains where false positives are costly, such as clinical decision support or fraud detection. It is defined for class i as:

$$\text{Precision}_i = \frac{tp_i}{tp_i + fp_i}, \tag{2.21}$$

The overall precision can be reported as a weighted average across all classes:

$$\text{Precision} = \frac{1}{n}\sum_{i=1}^{l} s_i \cdot \text{Precision}_i. \tag{2.22}$$

Recall

Recall measures the proportion of correctly predicted events for a class among all actual events of that class [24]. High recall indicates that the model can successfully identify most true instances of an activity. This is particularly critical in process-aware applications where missing rare but important events could lead to incomplete predictions or missed risk signals. For class i, it is defined as:

$$\text{Recall}_i = \frac{tp_i}{tp_i + fn_i}, \tag{2.23}$$

The weighted average recall across all classes is:

$$\text{Recall} = \frac{1}{n}\sum_{i=1}^{l} s_i \cdot \text{Recall}_i. \tag{2.24}$$

F1-score

The F1-score is the harmonic mean of precision and recall, providing a balanced metric that penalizes extreme disparities between them [11]. A high F1-score reflects that the model not only predicts activities accurately but also captures most of their true occurrences. This makes it particularly suitable for imbalanced datasets, where either precision or recall alone may give an incomplete picture of performance. For class i, it is computed as:

$$\text{F1}_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}, \tag{2.25}$$

The weighted F1-score across all classes is:

$$\text{F1-score} = \frac{1}{n} \sum_{i=1}^{l} s_i \cdot \text{F1}_i. \tag{2.26}$$

2.2 Related Work

Over the past decade, predicting the next activity in BPM has attracted attention for improving organizational efficiency and supporting better decision-making. Numerous studies have focused on predicting the next activity in ongoing cases [50, 29, 62, 3, 13]. Early approaches relied on traditional machine learning techniques. Models like Decision Trees were applied first due to their simplicity and interpretability [9, 48]. Decision trees have been widely adopted in predictive process monitoring scenarios, being used for tasks such as predicting Service Level Agreement (SLA) violations, Linear Temporal Logic (LTL) constraint violations, and process risks [15, 19, 40, 51]. They have often been combined with other techniques like clustering [19] or neural networks [40] to improve accuracy. Similarly, Support Vector Machines (SVM) have also been popular for predicting process performance, risk, and unnecessary checks [33, 59], while evolutionary computing methods [42] have shown potential for outperforming other ML algorithms in certain runtime prediction tasks.

However, as event logs grew more complex, researchers increasingly turned to deep learning techniques to address these complexities [43, 11, 1, 20]. As the field

has evolved, research in predictive process monitoring has focused not only on developing novel deep learning models for next-activity prediction but also on identifying effective methods for selecting the best model for each dataset. The next subsections first review the progress of deep learning in BPM and then discuss model selection methods, pointing out the gaps that inspire our work.

Initially, deep learning in BPM relied on RNNs for next-activity forecasting [2]. Although RNNs showed promising performance, they failed to remember the earlier context in lengthy sequences, and thus their performance on sequence prediction tasks was limited [61]. To overcome this, LSTMs were applied, demonstrating improvements in sequence modeling [63, 37, 30, 52]. However, LSTMs still faced challenges with long-range dependencies. However, LSTMs continued to face difficulties with long-range dependencies and are computationally expensive, often requiring significant time to capture input relationships [11]. These limitations prompted researchers to explore other deep learning architectures.

In response to these challenges, convolutional neural networks (CNNs) were explored for next-activity prediction [20, 45]. This CNN-based approach outperformed LSTMs in both accuracy and computational efficiency. As a result, CNNs emerged as a strong alternative for modeling sequential data. Nevertheless, LSTM-based methods remained popular, with models such as Data-Aware Explainable Next Activity Prediction (DENAP) [4], which combines LSTMs with Layer-Wise Relevance Propagation (LRP), achieving high accuracy while enhancing interpretability.

Building on these developments, Transformers have emerged as a powerful solution to the limitations of RNNs and LSTMs by leveraging self-attention to model long-range dependencies. Introduced by [57], for neural machine translation, the Transformer architecture replaces the sequential recurrence of earlier models with a parallel attention-based design, enabling more efficient and scalable sequence processing. This shift enables the modeling of long-range dependencies and the extraction of generalized relationships between inputs and outputs. In self-attention, the importance of each token in a sequence is computed relative to a reference token, allowing the model to prioritize the most relevant elements when forming representations. For example, given a sequence of events, each is encoded as a query, key, and value vec-

tors; the attention representation for a specific event is obtained by computing dot products between its query and all keys, producing attention weights that are applied to the corresponding values to form a weighted sum. Because each token's attention is calculated independently, all positions can be processed in parallel, removing the need for recurrence [11].

To capture multiple types of relationships, such as semantic or temporal dependencies, the self-attention mechanism is executed in parallel across several projections, resulting in multi-head self-attention. Although initially developed for natural language processing, the Transformer is a domain-agnostic architecture that has proven effective for diverse sequence modeling tasks, including predictive business process monitoring. Its ability to capture complex, multi-view dependencies makes it a strong candidate for advancing next-activity prediction in BPM. Models such as the Multi-View Information Fusion Method (MiFTMA) and the Multi-Task Learning Guided Transformer Network (MTLFormer) have shown notable improvements in capturing long-term dependencies, reducing complexity, and improving prediction accuracy [61, 60]. However, these Transformer-based approaches rely on a sliding window mechanism, which restricts their ability to capture broader process behaviors because they only consider fixed-length segments of event traces. Similarly, the ProcessTransformer [11] demonstrated the potential of applying Transformer architectures to predictive business process monitoring, it primarily focused on modeling activity sequences using fixed-size context windows and did not fully exploit all available event attributes, such as resources or case-level features.

While traditional machine learning methods, such as decision trees, offered simplicity and interpretability, they struggled with complex event logs and failed to capture temporal dependencies. Deep learning methods addressed some of these limitations, yet RNNs and LSTMs suffered from vanishing gradients, long-range dependency issues, and high computational costs, while CNNs were limited to modeling local patterns. Transformer-based models significantly advanced next-activity prediction by leveraging self-attention for long-range dependencies, but they rely on fixed sliding windows and focus primarily on activity sequences, overlooking valuable contextual attributes such as resources, time, and case-level features. To address these limita-

tions, our work introduces a dynamic, context-aware approach, DAW-Transformer, that moves beyond fixed-window representations and integrates multiple event attributes to more effectively capture complex process behaviors for next-activity prediction.

Building on this, a further challenge lies in determining when such complex models are essential, as not all event logs exhibit the same degree of variability [46]. Prior research highlights the importance of aligning model complexity with dataset variability to balance accuracy, efficiency, and interpretability [21, 35, 22]. In particular, measures of entropy have long been used to quantify uncertainty in information systems [31, 32] and have been applied in domains such as computer vision, natural language processing [44] to characterize data complexity and guide model selection.

Event logs in process mining differ significantly in their structure: some exhibit high variability and complexity (high entropy), whereas others follow more stable and predictable patterns (low entropy) [47]. Despite this diversity, existing studies rarely apply a systematic approach to model selection, often defaulting to deep learning methods regardless of dataset complexity. This practice introduces unnecessary computational overhead and reduces interpretability.

To address this gap, we propose an **Entropy-Driven Model Selection** strategy that quantifies dataset complexity and ensures optimal model selection. Unlike previous works that rely on trial and error [46], our approach uses normalized Shannon entropy. The framework recommends complex models such as the DAW-Transformer for high-entropy datasets, while suggesting simpler, interpretable models such as Decision Trees for low-entropy datasets. This approach ensures deep learning is applied only when necessary, providing a systematic balance between accuracy, interpretability, and efficiency.

Chapter 3

DAW-Transformer Methodology and Experimental Evaluation

3.1 Introduction

This chapter introduces the DAW-Transformer, a novel deep learning architecture developed to advance next-activity prediction in BPM. While prior approaches such as recurrent and convolutional neural networks have shown potential, they struggle to capture long-range dependencies and to fully utilize the diverse attributes contained in event logs. Traditional Transformer architectures partially address these limitations through self-attention but still depend on fixed-length sliding windows and typically focus only on activity sequences, neglecting valuable contextual information such as resources, timestamps, or numerical case-level attributes.

The DAW-Transformer addresses these challenges by extending the standard Transformer design in two key ways. First, it introduces a dynamic windowing mechanism that automatically adapts the sequence length to each dataset, ensuring that full process traces can be analyzed without shorting critical historical information. Second, it performs attribute-wise embedding, integrating categorical, numerical, and temporal features into a unified representation. This enables the model to learn interdependencies across multiple dimensions of process behavior rather than relying solely on activity labels.

Together, these innovations allow the DAW-Transformer to capture both global and local contextual patterns within event logs, improving predictive accuracy in complex and variable processes while maintaining scalability across datasets of different sizes and structures. The following sections details the proposed methodology, including data preprocessing, model architecture, training configuration and results.

3.2 Model Development and Implementation

3.2.1 Multi-Transformer

The Multi-Transformer architecture is specifically developed to capture intricate dependencies that exist in sequential data while simultaneously incorporating diverse feature types. Unlike traditional sequence modeling methods, which often struggle to represent long-range interactions or handle heterogeneous inputs, the Multi-Transformer leverages attention mechanisms to focus on the most relevant elements across a sequence selectively. This enables it to model both short- and long-term relationships with higher accuracy. Furthermore, by integrating multiple features, such as categorical, numerical, and temporal attributes, into its attention layers, the Multi-Transformer provides a more comprehensive representation of the data, leading to a richer understanding of context and improved predictive performance across a wide range of sequence learning tasks.

3.2.2 Multi-Feature Embedding and Position Encoding

This component is designed to construct a rich and comprehensive representation of each sequence by jointly embedding both categorical and numerical features, while also incorporating positional encodings to preserve the temporal order of events. The embedding process is fundamental, as it enables the model to project heterogeneous features into a shared latent space where their interdependencies and temporal dynamics can be more effectively captured. Without this step, the model would struggle to discern meaningful relationships between categorical and continuous attributes or to recognize how these relationships evolve. Since the attention mechanism itself

is inherently permutation-invariant and does not natively encode sequential order, positional encoding plays a critical role. By explicitly assigning positional information to each event, the model gains the ability to interpret not only which features co-occur but also when they occur within the sequence. This facilitates a deeper understanding of temporal dependencies and progression patterns, ultimately improving the model's ability to capture long-term dependencies and predict future events with greater accuracy [57].

3.2.3 DAW-Transformer

To tackle the challenge of predicting the next activity in complex business processes, we propose the DAW-Transformer. This model builds on the baseline Multi-Transformer architecture presented in Section 2.1.4. While the Multi-Transformer typically relies on a fixed-size sliding context window (e.g., a limited number of recent events), the DAW-Transformer dynamically determines the input window based on the characteristics of each dataset. Specifically, it identifies the maximum sequence length (i.e., the length of the longest case trace) and uses this as the window size for that dataset. All other case traces are padded accordingly. This approach allows the model to adjust to the complexity and structure of each dataset, ensuring that sufficient historical context is preserved without truncating valuable information. Additionally, the DAW-Transformer incorporates multi-modal features, such as categorical (activities, resources), temporal, and numerical (for example in Sepsis event logs: Age, CRP, LacticAcid, Leucocytes), which allows for richer contextual modeling. This comprehensive, full-history approach provides the model with a more holistic view of each case's progression, enabling superior performance in high-entropy, complex business processes.

To clearly highlight the differences between the DAW-Transformer and the baseline vanilla Transformer, Table 3.1 summarizes the key distinctions in sequence handling, attribute usage, and suitability for complex, high-entropy logs. While the Vanilla Transformer relies on a fixed sliding window and limited event attributes, the DAW-Transformer processes full-length traces, pads shorter sequences, and incorpo-

rates all available event attributes, enabling improved predictive performance. This comparison emphasizes the novelty and advantages of proposed approach [57].

Table 3.1: Comparative overview of the Vanilla Transformer and DAW-Transformer architectures.

Dimension	**Vanilla Transformer**	**DAW-Transformer**
Sequence Handling	Fixed sliding window	Full-length sequence (longest trace); shorter traces padded
Event Attributes	Limited (e.g., activity only)	Comprehensive: categorical, numerical, and temporal attributes

3.2.4 Data Preparation

To illustrate the preprocessing procedure, we refer to the *Sepsis* event logs, which combine control-flow information with clinical data. Categorical variables, including activities and resources, are first label-encoded and subsequently transformed into dense representations through embedding. Temporal features are derived from event timestamps, expressed as fractional hours relative to the case start, and normalized to the $[0, 1]$ range using min–max scaling. Clinical attributes such as *Age*, *CRP*, *LacticAcid*, and *Leucocytes* are z-score normalized, while missing values are imputed using a combination of forward- and backward-filling within each case.

3.2.5 Sequence Preparation and Padding

The data are first grouped by case, yielding sequences of tuples in which each event is represented together with its corresponding feature values. From these sequences, input–output pairs are constructed by generating sub-sequences for each case: the inputs comprise all features up to the current event, while the outputs correspond to the subsequent activity. To facilitate learning across cases, separate lists are maintained for each input feature as well as for the output.

Since cases vary in length, raw sequences differ in size, making them incompatible with batch training. To ensure consistent input dimensions for the model, all sequences are subsequently padded to a uniform length. Padding is typically applied at

the end of shorter sequences with a special token that does not carry semantic meaning. This allows the model to distinguish between real data and artificial padding. In practice, masks are also generated alongside padded sequences to prevent the attention mechanism from attending to padded positions, thereby preserving the integrity of learning.

3.2.6 Model Architecture

The DAW-Transformer architecture integrates categorical, temporal, and numerical inputs into a unified predictive framework. Categorical features are mapped into dense vectors through embedding layers, where masking is employed to handle padded tokens. Trainable positional embeddings are then added to these vectors to encode event order, enabling the model to preserve sequence information while maintaining parallelizability.

Sequence dependencies are modeled using stacked Transformer encoder blocks, each composed of a multi-head self-attention mechanism, position-wise feed-forward layers, residual connections, and layer normalization. The self-attention mechanism allows the model to capture dependencies across all events in the sequence, while residual pathways and normalization stabilize optimization and enable deeper stacking without degradation. The feed-forward layers expand the representational capacity of each encoder block by applying non-linear transformations before projecting back into the embedding space.

Numerical features, comprising both time-based and other attributes, bypass the embedding layers. Instead, these features are processed directly through dense transformations and flattened into fixed-length representations. This design reflects the heterogeneous nature of the data: while categorical variables benefit from semantic embeddings and sequence modeling, numerical variables carry direct scalar information that can be integrated without additional abstraction.

The transformed categorical embeddings and flattened numerical features are concatenated into a single joint representation vector. This fusion step ensures that both symbolic and continuous information contribute to the predictive signal. The com-

bined representation is subsequently passed through fully connected layers with non-linear activations and dropout regularization, mitigating overfitting while enhancing generalization. The final layer employs a softmax activation, producing a probability distribution over the possible next activities in the sequence.

The architecture reflects a hybrid design that leverages the strengths of Transformer encoders for modeling sequential dependencies, while simultaneously incorporating dense layers for temporal covariates. This approach enables the DAW-Transformer to capture both the contextual dynamics of process traces and the predictive value of auxiliary features, thereby improving accuracy in next-activity prediction tasks.

3.2.7 Training

The dataset was split into training (64%), validation (16%), and testing (20%) sets. The training set was used to fit the model parameters, while the validation set guided the tuning of hyperparameters and provided an unbiased signal for early stopping and model checkpointing. Early stopping was applied to prevent overfitting by monitoring validation loss and halting training when performance plateaued. Model checkpointing ensured that the best-performing weights on the validation set were preserved and later restored for evaluation. The held-out test set was reserved exclusively for the final unbiased assessment of generalization performance.

The model was trained using the Adam optimizer, chosen for its adaptive learning rate capabilities, with categorical cross-entropy employed as the loss function to capture the multi-class nature of the prediction task. Batch training was adopted to accelerate convergence and efficiently utilize GPU resources. To mitigate overfitting, dropout regularization was applied in the fully connected layers, and masking was used to exclude padded positions from influencing the attention mechanism. Hyperparameters such as learning rate, embedding dimension, number of attention heads, and feed-forward size were tuned empirically through iterative experimentation.

Dataset

The experiments were carried out on six publicly available datasets commonly used in process mining. These event logs were chosen not only because of their availability but also because they represent a diverse range of real-world processes, ensuring that conclusions drawn from the evaluation are broadly applicable. Some relevant statistics from these logs are shown in Table 3.2. These include the number of cases, the total number of recorded events, the number of distinct activities, as well as the average and maximum case lengths. Such characteristics highlight the variability in dataset complexity: logs with longer cases and higher activity diversity generally pose greater challenges for prediction.

The datasets selected span domains such as healthcare, public administration, and service management, allowing for the evaluation of model performance across heterogeneous contexts. This diversity ensures that the DAW-Transformer is not tailored to a narrow application but instead demonstrates generalizable performance. In particular, the inclusion of datasets with both balanced and imbalanced activity distributions enables assessment of how well the model handles rare events, which are often critical in practical applications.

Table 3.2: General properties of each dataset.

Event log	**Num. cases**	**Num. events**	**Num. activities**	**Avg. case length**	**Max. case length**
Sepsis	1,049	15,214	16	14.48	185
Filtered_Hospital log	1,142	136,023	64	119.11	1,611
NASA	2,566	73,638	47	28.70	50
BPI_2012_A	13,087	60,849	10	4.65	8
Helpdesk	4,580	21,348	14	4.66	15
BPI_2020_Prepaid travel cost	2,099	18,246	29	8.69	21

The following provides a brief description of each event log, emphasizing its domain of origin, process characteristics, and challenges for predictive modeling:

Sepsis: This real-world event log includes events of sepsis cases from a hospital, documented by the ERP (Enterprise Resource Planning) system. Each case in the log represents a patient's journey through the hospital [41]. Table 3.3 shows a sample

of this event logs.

Table 3.3: A sample event log of the Sepsis Cases dataset.

Case ID	org:group	Activity	Age	Timestamp	CRP	Lactic Acid	Leucocytes
XJ	A	ER Registration	90	2013-11-07 08:18:29	–	–	–
XJ	C	ER Triage	–	2013-11-07 08:29:18	–	–	–
XJ	A	ER Sepsis Triage	–	2013-11-07 08:37:32	–	–	–
XJ	B	LacticAcid	–	2013-11-07 08:51:00	–	1.4	–
XJ	B	Leucocytes	–	2013-11-07 08:51:00	–	–	296.2
XJ	B	CRP	–	2013-11-07 08:51:00	16	–	–
XJ	A	IV Liquid	–	2013-11-07 09:05:57	–	–	–
XJ	A	IV Antibiotics	–	2013-11-07 10:05:58	–	–	–
XJ	I	Admission NC	–	2013-11-07 11:11:34	–	–	–
XJ	B	Leucocytes	–	2013-11-08 08:00:00	–	–	297.6
XJ	B	Leucocytes	–	2013-11-12 08:00:00	–	–	381.3
XJ	E	Release A	–	2013-11-13 12:30:00	–	–	–
XJ	B	Return ER	–	2013-12-11 11:02:20	–	–	–
I	L	ER Registration	80	2013-11-09 09:21:03	–	–	–

Filtered Hospital logs: Real-life event log from a Dutch academic hospital, originally published for the first Business Process Intelligence Contest (BPIC 2011) [54]. The original event log contains 624 different activities with a normalized entropy of 0.236. In this study, the log is filtered to include only activities that occurred more than 400 times. After filtering, the number of activities is reduced to 64, and the normalized entropy increases to 0.313, indicating higher entropy and making it suitable for our work. Table 3.4 shows a sample of this event logs.

Table 3.4: A sample event log of the Hospital dataset.

org:group	Activity	Timestamp	Age	Treatment code	concept:name
Radiotherapy	1e consult poliklinisch	2005-01-03 00:00:00	33	23	0
Radiotherapy	administratief tarief - eerste pol	2005-01-03 00:00:00	33	23	0
Nursing ward	1e consult poliklinisch	2005-01-05 00:00:00	33	23	0
Nursing ward	administratief tarief - eerste pol	2005-01-05 00:00:00	33	23	0
General Lab Clinical Chemistry	aanname laboratoriumonderzoek	2005-02-15 00:00:00	33	23	0
General Lab Clinical Chemistry	aanname laboratoriumonderzoek	2005-02-15 00:00:00	33	23	0
General Lab Clinical Chemistry	ureum	2005-02-15 00:00:00	33	23	0
General Lab Clinical Chemistry	hemoglobine foto-elektrisch	2005-02-15 00:00:00	33	23	0
General Lab Clinical Chemistry	creatinine	2005-02-15 00:00:00	33	23	0
General Lab Clinical Chemistry	natrium vlamfotometrisch	2005-02-15 00:00:00	33	23	0
General Lab Clinical Chemistry	kalium potentiometrisch	2005-02-15 00:00:00	33	23	0

NASA: The event log is obtained by instrumenting the NASA Crew Exploration

Vehicle (CEV) class using the XPort tool. It records method-call level events describing a single run of an exhaustive unit test suite for the CEV example. The life-cycle information corresponds to method calls (start) and returns (complete), thus capturing a method-call hierarchy [39]. Table 3.5 shows a sample of this event logs.

Table 3.5: A sample event log of the NASA dataset.

apploc:linenr	apploc:etype	time:timestamp	concept:name	lifecycle:transition	case:concept:name
20	call_new	2017-02-13 15:50:51.610000+00:00	`cev.CEV()`	start	`cev.TestCEV.test_1()`
29	call_new	2017-02-13 15:50:51.613000+00:00	`cev.ErrorLog()`	start	`cev.TestCEV.test_1()`
29	return_new	2017-02-13 15:50:51.613000+00:00	`cev.ErrorLog()`	complete	`cev.TestCEV.test_1()`
31	call_new	2017-02-13 15:50:51.617000+00:00	`cev.Failures(cev.ErrorLog)`	start	`cev.TestCEV.test_1()`
24	call_new	2017-02-13 15:50:51.621000+00:00	`cev.Failures$Type(java.lang.String,int)`	start	`cev.TestCEV.test_1()`
24	return_new	2017-02-13 15:50:51.622000+00:00	`cev.Failures$Type(java.lang.String,int)`	complete	`cev.TestCEV.test_1()`

BPIC_2012_A: This is an event log recording a real-world loan application process [55]. Table 3.6 shows a sample of this event logs.

Table 3.6: A sample event log of BPIC_2012_A.

resource	lifecycle:transition	concept:name	time:timestamp	case:concept:name	case:AMOUNT_REQ
112	COMPLETE	A_SUBMITTED	2011-10-01 00:38:44.546000	173688	20000
112	COMPLETE	A_PARTLYSUBMITTED	2011-10-01 00:38:44.880000	173688	20000
112	COMPLETE	A_PREACCEPTED	2011-10-01 00:39:37.906000	173688	20000
10862	COMPLETE	A_ACCEPTED	2011-10-01 11:42:43.308000	173688	20000
10862	COMPLETE	A_FINALIZED	2011-10-01 11:45:09.243000	173688	20000
10629	COMPLETE	A_REGISTERED	2011-10-13 10:37:29.226000	173688	20000
10629	COMPLETE	A_APPROVED	2011-10-13 10:37:29.226000	173688	20000
10629	COMPLETE	A_ACTIVATED	2011-10-13 10:37:29.226000	173688	20000
112	COMPLETE	A_SUBMITTED	2011-10-01 08:08:58.256000	173691	5000
112	COMPLETE	A_PARTLYSUBMITTED	2011-10-01 08:09:02.195000	173691	5000
112	COMPLETE	A_PREACCEPTED	2011-10-01 08:09:56.648000	173691	5000

Helpdesk: This dataset comprises events from the ticket management process of the help desk of an Italian software company. Each case in the log begins with a new ticket entry into the ticket management system and concludes with the resolution of the issue and the closing of the ticket [58]. Table 3.7 shows a sample of this event logs.

Table 3.7: A sample event log from the Helpdesk dataset.

concept:name	lifecycle:transition	org:resource	time:timestamp	Activity	Resource	case:concept:name	case:variant	case:variant-index
`Assign seriousness`	complete	Value 2	2010-01-13 08:40:25+00:00	`Assign seriousness`	Value 2	Case3608	Variant 33	33
`Take in charge ticket`	complete	Value 2	2010-01-29 08:52:27+00:00	`Take in charge ticket`	Value 2	Case3608	Variant 33	33
`Resolve ticket`	complete	Value 2	2010-01-29 08:52:34+00:00	`Resolve ticket`	Value 2	Case3608	Variant 33	33
`Closed`	complete	Value 5	2010-02-13 08:52:48+00:00	`Closed`	Value 5	Case3608	Variant 33	33
`Closed`	complete	Value 5	2010-02-13 08:52:48+00:00	`Closed`	Value 5	Case3608	Variant 33	33
`Assign seriousness`	complete	Value 2	2010-01-13 12:26:04+00:00	`Assign seriousness`	Value 2	Case2748	Variant 1	1
`Take in charge ticket`	complete	Value 2	2010-01-19 09:26:05+00:00	`Take in charge ticket`	Value 2	Case2748	Variant 1	1
`Resolve ticket`	complete	Value 2	2010-01-19 09:28:50+00:00	`Resolve ticket`	Value 2	Case2748	Variant 1	1
`Closed`	complete	Value 5	2010-02-13 12:00:28+00:00	`Closed`	Value 5	Case2748	Variant 1	1

BPIC_2020_Prepaid Travel Costs: This file includes events associated with

prepaid travel expenses for the parent item [56]. Table 3.8 shows a sample of this event logs.

Table 3.8: A sample event log from the Travel Permit dataset.

id	org:resource	concept:name	time:timestamp	org:role	case:Task	case:Activity	case:concept:name
st_step 73555_0	STAFF MEMBER	Permit SUBMITTED by EMPLOYEE	2017-01-09 14:48:43+00:00	EMPLOYEE	task 71977	activity 505	request for payment 73550
st_step 73554_0	STAFF MEMBER	Permit FINAL_APPROVED by SUPERVISOR	2017-01-09 14:48:55+00:00	SUPERVISOR	task 71977	activity 505	request for payment 73550
st_step 73558_0	STAFF MEMBER	Request For Payment SUBMITTED by EMPLOYEE	2017-01-12 11:40:27+00:00	EMPLOYEE	task 71977	activity 505	request for payment 73550
st_step 73559_0	STAFF MEMBER	Request For Payment FINAL_APPROVED by SUPERVISOR	2017-01-12 11:41:59+00:00	SUPERVISOR	task 71977	activity 505	request for payment 73550
st_step 73557_0	STAFF MEMBER	Request For Payment REJECTED by MISSING	2017-01-12 11:53:07+00:00	MISSING	task 71977	activity 505	request for payment 73550
st_step 73556_0	STAFF MEMBER	Permit REJECTED by MISSING	2017-07-28 10:10:18+00:00	MISSING	task 71977	activity 505	request for payment 73550
st_step 73555_0	STAFF MEMBER	Permit SUBMITTED by EMPLOYEE	2017-01-09 14:48:43+00:00	EMPLOYEE	task 71977	activity 505	request for payment 73552
st_step 73554_0	STAFF MEMBER	Permit FINAL_APPROVED by SUPERVISOR	2017-01-09 14:48:55+00:00	SUPERVISOR	task 71977	activity 505	request for payment 73552
st_step 73561_0	STAFF MEMBER	Request For Payment SUBMITTED by EMPLOYEE	2017-01-19 18:47:18+00:00	EMPLOYEE	task 71977	activity 505	request for payment 73552

Hyperparameter Setup

Table 3.9 presents the hyperparameters used for the DAW-Transformer model on the Sepsis dataset. The model employs an embedding dimension of 256 to effectively capture feature representations, with 8 attention heads to enhance the learning of sequential dependencies. A feed-forward dimension of 256 ensures sufficient model capacity for processing complex patterns in activity sequences. The Adam optimizer is utilized for efficient gradient-based optimization, while a batch size of 16 is chosen to accommodate the variable-length sequences in the dataset. The model is trained for 50 epochs with a validation split of 0.2, ensuring a balanced evaluation of its generalization ability. These hyperparameter choices are designed to optimize performance while maintaining stability during training. Additionally, early stopping with a patience of 10 and model checkpointing based on validation loss were employed to prevent overfitting and retain the best model. Layer normalization, ReLU activation in the feed-forward network, and no dropout were applied within the Transformer blocks.

For the Sepsis dataset, a CNN-BiLSTM model was used, with carefully tuned hyperparameters to enhance performance, as shown in Table 3.10. Key parameters included an initial filter size of 64 for the first convolutional layer, which progressively increased to 256 in subsequent layers, facilitating the extraction of complex features. To counteract overfitting, dropout values of 0.4 and 0.5 were added. To extract temporal relations in the sequence, a 128-unit bidirectional LSTM layer was added. The Adam optimizer with a learning rate of 0.001 was employed, and the model was

Table 3.9: Hyperparameters for DAW-Transformer Model on the Sepsis dataset.

Hyperparameter	**Value**
Embedding dimension (embed_dim)	256
Number of Transformer heads (num_heads)	8
Feed-forward dimension (ff_dim)	256
Activation function	ReLU
Normalization	LayerNorm ($\epsilon = 10^{-6}$)
Positional encoding	Learned embeddings
Optimizer	Adam
Loss function	Sparse categorical cross-entropy
Batch size	16
Number of epochs	50 (with early stopping)
Early stopping patience	10
Validation split	0.2
Model checkpointing	Best model (val_loss) saved

trained over 300 epochs with a batch size of 32; early stopping and a learning rate scheduler were used to refine the training process and enhance generalization.

Table 3.10: Hyperparameters for CNN-LSTM model on sepsis dataset.

Hyperparameter	**Value**	**Hyperparameter**	**Value**	**Hyperparameter**	**Value**
Filters in 1st Conv Layer	64	Optimizer	Adam	Dropout rate after 1st Conv	0.4
Kernel size in 1st Conv Layer	3	Learning rate	0.001	Filters in 2nd Conv Layer	128
Pool size in 1st Max Pooling	1	Batch size	32	Kernel size in 2nd Conv Layer	3
Dropout rate after 1st Conv	0.4	Number of epochs	300	Pool size in 2nd Max Pooling	1
Filters in 2nd Conv Layer	128	Validation split	0.2	Dropout rate after 2nd Conv	0.5
Kernel size in 2nd Conv Layer	3	Early stopping patience	30	Filters in 3rd Conv Layer	256
Pool size in 2nd Max Pooling	1	Learning rate scheduler patience	10	Kernel size in 3rd Conv Layer	3
Dropout rate after 2nd Conv	0.5	Learning rate scheduler factor	0.5	Pool size in 3rd Max Pooling	1
Filters in 3rd Conv Layer	256	Learning rate scheduler min lr	1e-6	Dropout rate after 3rd Conv	0.5
Units in LSTM Layer	128	Output Layer activation	Softmax	Units in Dense Layer	100
L2 Regularization in Dense	0.02	Dropout rate after Dense Layer	0.6		

The hyperparameter configurations in Tables 3.9 and 3.10 were selected through iterative experimentation and cross-validation. For instance, an embedding dimension of 256 was chosen to provide sufficient representational capacity for capturing both categorical and numerical features without leading to overfitting or excessive computational overhead. Similarly, the patience value of 10 in early stopping reflects a balance between allowing the model to converge and preventing unnecessary training epochs once performance plateaued. These values were not arbitrarily selected but were informed by preliminary experiments, common practices in sequence modeling,

and the characteristics of the datasets used.

Evaluation Metrics

To assess the performance of the proposed model in predicting the next activity, several widely used evaluation metrics are employed. These metrics provide complementary perspectives on classification quality, especially in datasets with imbalanced classes, where relying on a single measure such as accuracy may be misleading. Specifically, **accuracy**, **precision**, **recall**, and **F1-score** are considered. Together, these metrics enable a balanced evaluation of both global correctness and class-specific sensitivity, which is essential in process mining where rare activities often carry high practical significance. They also provide a comprehensive evaluation of the model's predictive performance. Accuracy offers a global view of correctness, precision emphasizes reliability of positive predictions, recall highlights completeness in capturing true events, and the F1-score balances both aspects. In the context of next-activity prediction, this combination ensures that the evaluation accounts for not only the overall success rate but also the capacity to handle minority classes, which are often critical in real-world process execution.

3.3 Results and Discussion

This section presents an experimental evaluation of the DAW-Transformer model, comparing its performance with that of several established baselines: CNN-LSTM, CNN-BiLSTM, Vanilla Transformer, XGBoost, Decision Trees, and Random Forest. These models were selected to provide a diverse set of benchmarks covering both deep learning and traditional machine learning paradigms. The CNN-LSTM and CNN-BiLSTM architectures represent widely used sequence learning methods that combine convolutional layers for local feature extraction with recurrent layers for temporal modeling. The Vanilla Transformer variant demonstrates the performance of an attention-based model with a constrained temporal horizon, while XGBoost, Decision Trees, and Random Forest exemplify non-neural approaches that rely on feature engineering and ensemble learning. By evaluating against this broad spectrum

of models, the relative strengths and weaknesses of the DAW-Transformer can be highlighted more comprehensively.

In this study, a wide range of models were implemented and evaluated across six event logs, with performance measured using accuracy, precision, recall, and F1-score. The comparative analysis reveals several important findings.

First, the proposed DAW-Transformer consistently outperformed baseline models in challenging datasets such as the Sepsis and Filtered Hospital Logs. Its ability to integrate multiple attributes and apply a dynamic windowing mechanism enabled it to capture long-range dependencies and contextual information more effectively than conventional Transformer-based approaches or recurrent models. In these datasets, the DAW-Transformer achieved the highest F1-scores, reflecting a strong balance between precision and recall, particularly for minority classes that are often difficult to predict.

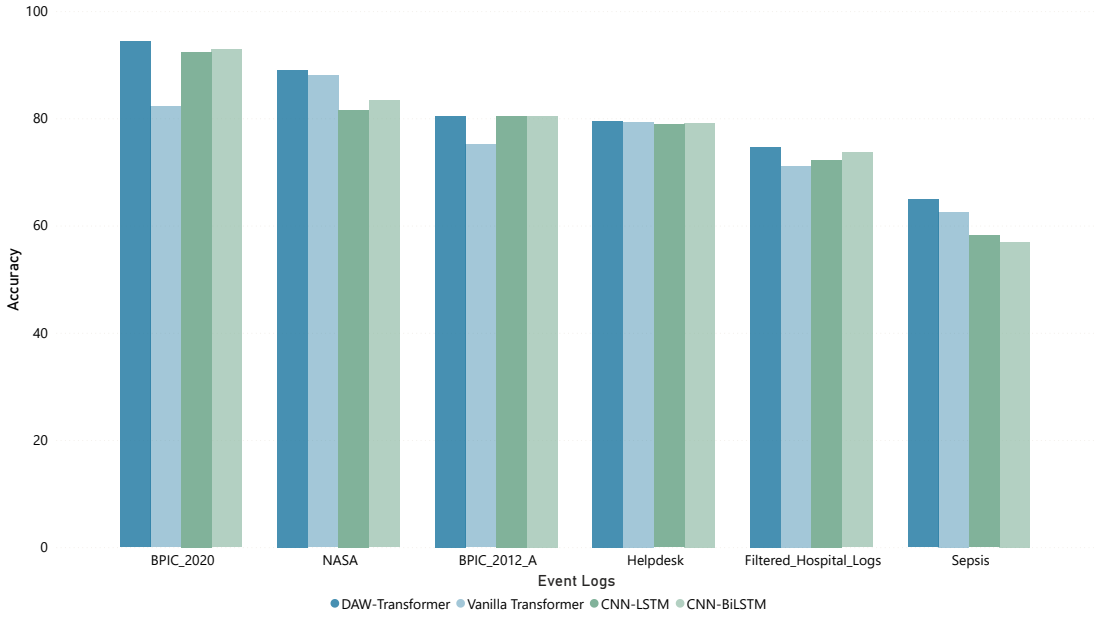
Second, on some of datasets such as the BPIC_2012_A and Helpdesk logs, traditional machine learning models, particularly Decision Trees and Random Forests, achieved competitive results with lower computational cost and higher interpretability. This demonstrates that simpler models remain highly effective in predictable process contexts.

Third, hybrid deep learning baselines such as CNN-LSTM and CNN-BiLSTM showed moderate improvements over classical models on several datasets, but they generally fell short of the DAW-Transformer in capturing complex contextual dependencies.

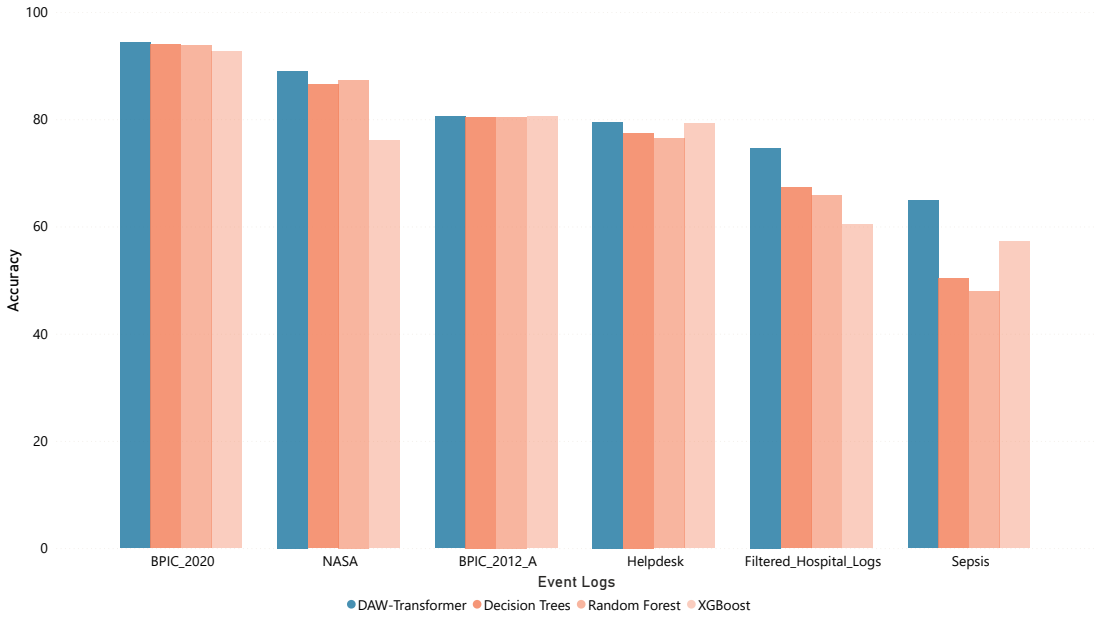
The detailed outcomes of these evaluations are summarized in Table 3.11, which reports the performance of each model across all datasets. Figure 3.1 shows that the DAW-Transformer outperforms the other models in both the deep learning and traditional machine learning groups.

3.4 Conclusions

The experimental results confirm the effectiveness of the proposed DAW-Transformer in advancing next-activity prediction. Across multiple event logs, it consistently



(a) DAW-Transformer vs. deep learning models.



(b) DAW-Transformer vs. traditional ML models.

Figure 3.1: DAW-Transformer shows strong performance across all event logs compared to deep learning and traditional ML baselines.

achieved superior performance compared to both deep learning baselines and traditional machine learning models, particularly in scenarios requiring the capture of long-range dependencies and multiple attributes. At the same time, the experiments also demonstrate that simpler models, such as Decision Trees and Random Forests, remain valuable alternatives in some of datasets where efficiency is prioritized. These results motivate the need for entropy-driven model selection, which we elaborate in the next chapter.

Table 3.11: Performance Comparison across Models and Event Logs (in %)

Event Logs	Metric	Normalized Entropy	Decision Trees	Random Forest	XGBoost	CNN-LSTM	CNN-BiLSTM	Vanilla Transformer	DAW-Transformer
Sepsis	Accuracy	0.5021	50.30	47.93	57.16	58.18	57.48	62.58	**64.88**
	F1-score		50.47	47.25	55.94	55.16	54.96	61.84	64.40
	Recall		50.30	47.93	57.16	58.18	57.48	62.58	64.88
	Precision		50.74	47.54	59.75	56.73	55.26	63.19	65.31
Filtered_Hospital Logs	Accuracy	0.3132	67.32	65.80	60.40	72.26	73.66	71.08	**74.63**
	F1-score		66.56	65.08	55.67	68.36	71.71	70.02	73.99
	Recall		67.32	65.80	60.40	72.26	73.66	71.08	74.63
	Precision		65.95	64.69	59.25	67.41	72.21	71.17	74.61
NASA	Accuracy	0.2843	86.46	87.37	76.00	81.55	83.30	88.04	**89.02**
	F1-score		86.66	87.47	75.51	80.08	82.77	88.22	89.25
	Recall		86.46	87.37	76.00	81.55	83.30	88.04	89.02
	Precision		89.13	88.97	78.68	81.22	84.91	90.86	92.27
BPIC_2012_A	Accuracy	0.2339	80.46	80.46	80.47	80.47	80.47	75.10	**80.49**
	F1-score		75.25	75.25	75.25	75.25	75.25	69.50	72.25
	Recall		80.47	80.47	80.47	80.47	80.47	75.10	80.47
	Precision		72.24	72.24	72.24	72.24	72.24	67.66	72.24
Helpdesk	Accuracy	0.2322	77.43	76.39	79.19	78.86	79.16	79.28	**79.48**
	F1-score		73.61	74.21	74.40	73.50	73.85	75.38	75.14
	Recall		77.43	76.39	79.19	78.86	79.16	79.28	79.48
	Precision		73.33	73.26	72.03	69.13	69.66	81.53	81.18
BPIC_2020_ Prepaid Travel Cost	Accuracy	0.1922	93.93	93.76	92.65	92.39	92.83	82.23	**94.35**
	F1-score		92.86	92.69	91.35	90.93	91.36	79.71	93.11
	Recall		93.93	93.76	92.65	92.39	92.83	82.23	94.35
	Precision		93.19	92.98	91.74	90.10	90.54	82.21	93.44

Chapter 4

Entropy-Driven Model Selection Framework

4.1 Introduction

The effectiveness of predictive process monitoring largely depends on selecting models. In practice, however, organizations often rely on trial-and-error experimentation to identify suitable models, leading to inefficiencies and inconsistent outcomes. While the previous chapter focused on developing and evaluating the DAW-Transformer as a powerful predictive architecture capable of handling multi-attribute event logs, not all processes require such high-capacity models. Many real-world business processes exhibit structured and repetitive patterns where simpler, interpretable models such as Decision Trees or Random Forests can achieve comparable accuracy with significantly lower computational cost. This observation motivates the need for a principled framework that guides model selection based on data characteristics rather than intuition alone.

To address this challenge, this chapter introduces an entropy-driven model selection approach that leverages process entropy as a quantitative indicator of behavioral complexity and variability in event logs. Entropy, a concept rooted in information theory, measures the degree of uncertainty within a process, that is, how unpredictable the next activity is given the current one. In structured, low-entropy processes,

transitions between activities follow well-defined patterns, whereas in unstructured, high-entropy processes, numerous possible next steps create substantial uncertainty. By quantifying this uncertainty, process entropy provides an objective basis for distinguishing between processes that favor interpretable models and those that demand more expressive architectures such as the DAW-Transformer.

The aim of this chapter is therefore twofold: (1) to formalize the computation of conditional and normalized entropy as indicators of process variability, and (2) to demonstrate how these measures can systematically guide the selection of predictive models in Business Process Management (BPM). Through this entropy-based perspective, we move beyond model-centric experimentation toward a data-centric decision framework, ensuring that model complexity is introduced only when justified by process complexity. The subsequent sections describe the methodology for computing entropy from event logs, outline the experimental setup for validating the approach, and present empirical results demonstrating how entropy correlates with model performance across diverse datasets.

4.2 Methodology

The methodology presented in this section describes how process entropy is calculated and applied to support Entropy-Driven Model Selection Framework . The approach begins with data preparation and transition extraction. Using these transitions, conditional entropy is computed to quantify the uncertainty of predicting the next activity given the current one. To enable consistent comparison among datasets with varying activity counts, entropy values are then normalized to a range between 0 and 1. These normalized entropy scores are subsequently used to classify processes as either structured or unstructured, guiding the choice between interpretable models and more complex architectures such as the DAW-Transformer.

4.2.1 Data Preparation

Reliable entropy estimation begins with a well-structured and consistent dataset. To avoid data leakage, the entropy used for model selection is calculated only on

the training and validation data, which together constitute 80% of the dataset. By excluding the test set, we ensure that information from unseen cases does not influence model selection. This allows the entropy to reflect the inherent uncertainty of the process based solely on the data used for training, while keeping the test set completely independent for unbiased evaluation.

Each event log consists of several attributes such as activity, timestamp, case ID, and context-specific features (*e.g.,* resource). During preprocessing, categorical attributes are encoded, and all data is standardized to ensure compatibility with machine learning algorithms.

Event logs stored in a CSV file are used, where each row represents an event and contains at least the following columns:

- `case:concept:name`: the unique identifier for each process instance (case)
- `concept:name`: the activity name

The event logs are loaded using the pandas library and group activities by their corresponding case ID to reconstruct *traces*, i.e., sequences of activities that occurred within each process instance.

4.2.2 Transition Extraction

To capture the sequential relationships between activities, all *bigrams* are extracted, i.e., ordered pairs of consecutive activities (a, b), from each trace. These bigrams represent observed transitions between steps in the process.

```
bigrams = []
for trace in traces:
    bigrams.extend([(trace[i], trace[i+1]) for i in range(len(trace) - 1)])
```

We count:

- The frequency of each bigram (i.e., transition): `Counter((a, b))`
- The frequency of each activity as the current activity a

4.2.3 Conditional Entropy Calculation

The uncertainty of predicting the next activity given the current one can be formally expressed as the conditional entropy.

In our implementation, $p(x)$ corresponds to the empirical probability of observing activity x as the current activity, and $p(y \mid x)$ denotes the probability of transitioning from x to y, estimated as:

$$p(y \mid x) = \frac{\text{count}(x, y)}{\text{count}(x)}. \tag{4.1}$$

Thus, while the transition probabilities $p(y \mid x)$ are directly computed from the event log, the entropy formulation aggregates these probabilities into a single metric of conditional uncertainty, which we refer to as conditional entropy.

4.2.4 Normalization

To compare entropy values across datasets with varying numbers of unique activities, the conditional entropy is normalized using the base-2 logarithm of the total number of unique activities [64]. This normalization accounts for the fact that event logs may have high conditional entropy even with a small number of distinct activities, while other logs with the same entropy may involve many more activities, indicating that some activities occur rarely. Normalization ensures that entropy values from different event logs can be compared on the same scale, as they are adjusted based on the number of activities in each log. Figure 4.1 illustrates the values of conditional entropy and normalized entropy across the selected event logs. A noteworthy observation is that datasets with a larger number of activities, such as BPIC_2020, exhibit relatively higher conditional entropy. However, when entropy is normalized with respect to the number of activities, the resulting normalized entropy may in fact be lower than that of datasets with fewer activities. This phenomenon is evident in the comparison between BPIC_2020 and Helpdesk: although BPIC_2020 records a higher conditional entropy, its normalized entropy falls below that of Helpdesk, underscoring the importance of normalization when comparing datasets with varying activity cardinalities. This ensures that the normalized entropy value lies within the

interval $[0, 1]$, facilitating consistent interpretation regardless of the dataset size.

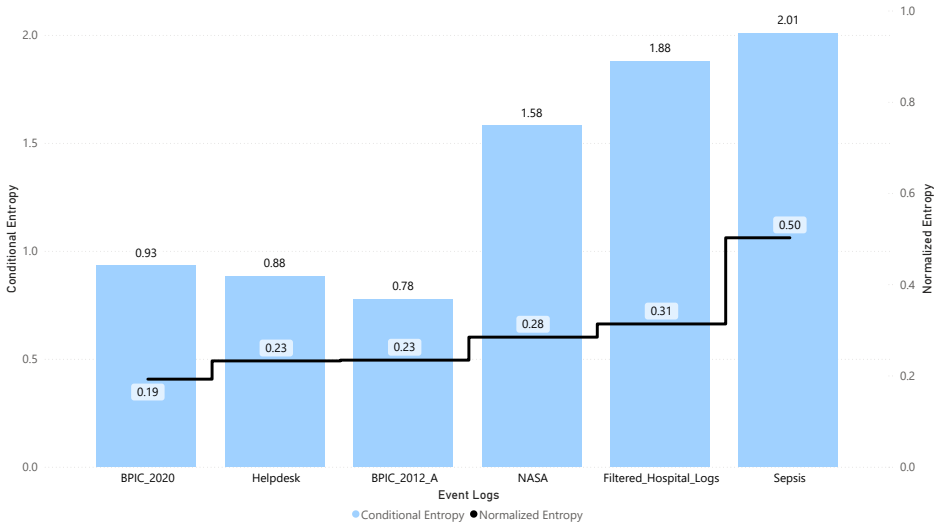


Figure 4.1: Conditional and normalized entropy across event logs.

4.2.5 Entropy-Driven Model Selection Framework

Building on the entropy computation described in Section 4.1, we introduce an Entropy-Driven Model Selection Framework guided by the level of process entropy. The objective is to align model complexity with the behavioral variability observed in the dataset. Based on the normalized entropy values calculated earlier, we distinguish between structured and unstructured process behaviors (as interpreted in Section 4.2.4). This distinction forms the basis of our model selection approach:

- **Low Entropy** suggests structured and predictable process behavior. In such cases, simpler models like Decision Trees are typically sufficient to achieve high accuracy.
- **High Entropy** indicates unstructured, complex, and less predictable process behavior. For these datasets, more complex models such as the DAW-Transformer are preferred, due to their ability to capture intricate sequential patterns.

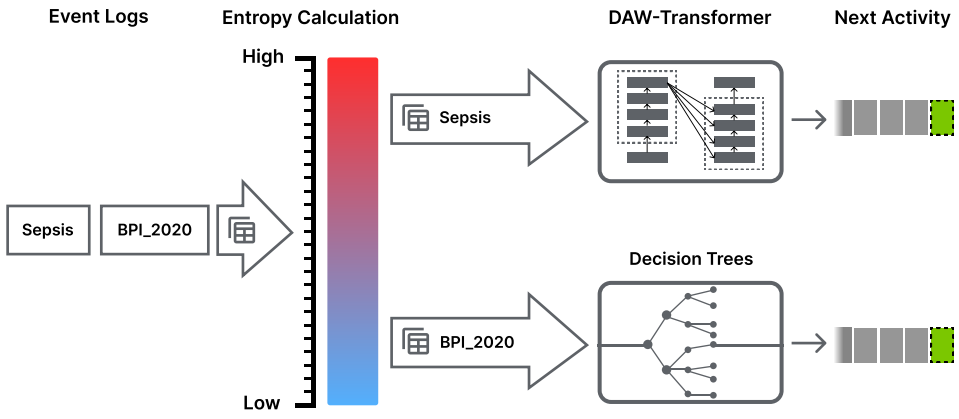


Figure 4.2: Entropy-Driven Model Selection Framework: high-entropy datasets utilize DAW-Transformer for greater accuracy, while low-entropy datasets rely on Decision Trees for interpretability with comparable performance.

This entropy-driven selection framework enables an informed and adaptive modeling process, ensuring that the predictive model aligns with the underlying complexity of the data. Figure 4.2 illustrates the overall methodology, showing how entropy estimation supports model selection in next activity prediction tasks.

4.3 Experiments

The experimental analysis in this section is designed to validate the entropy-driven model selection framework introduced in Section 4.2. Whereas Chapter 3 primarily focused on comparing the proposed DAW-Transformer with a range of benchmark models on raw predictive accuracy and other evaluation metrics, the present chapter builds on those results by explicitly examining the role of process entropy as a guiding factor in model choice. In other words, while the experiments of Chapter 3 established that different models achieve different levels of performance across event logs, the experiments here aim to explain how entropy can be used systematically to interpret and predict such outcomes. This shift from pure performance benchmarking to entropy-based analysis ensures that the thesis not only proposes a new model but also offers a principled framework for adaptive model selection.

The same six event logs described in Section 3.2.7, are used in this study. Each

event log exhibits different structural properties: some, such as Sepsis and Filtered Hospital Logs, contain highly variable activity sequences with numerous transitions, while others, such as BPI 2012 A or Helpdesk, are far more structured and repetitive. By quantifying this variability through conditional entropy, we obtain a numerical indicator of the uncertainty inherent in each log. The normalized entropy values make it possible to compare logs of very different sizes and activity counts on a unified scale between 0 and 1.

The evaluation process follows a structured pipeline. First, entropy values are computed for each log by analyzing transition probabilities between consecutive activities, as explained in Section 4.2. These values are then normalized to enable cross-dataset comparison. Next, the performance of a broad set of predictive models is assessed using the same classification metrics already introduced. These metrics (accuracy, precision, recall, and F1-score) provide a comprehensive view of predictive quality, ensuring that performance is not judged solely by overall correctness but also by the ability to capture rare or minority activities that are often critical in real-world processes.

The central goal of this experiment is to investigate the relationship between entropy and predictive performance. More specifically, we aim to determine whether entropy can serve as a reliable criterion for choosing between simple interpretable models and more complex deep learning architectures.

4.4 Results

The resulting values of conditional entropy provided a numerical indicator of process predictability. A low entropy value reflects structured behavior where the current activity largely determines the next activity. In contrast, high entropy values indicate unstructured or flexible processes with less predictable activity sequences. Based on our experimental observations and analysis of the dataset (Figure 4.3), we observed a substantial difference in accuracy between deep learning models and simpler approaches.

The resulting values of conditional entropy provided a numerical indicator of pro-

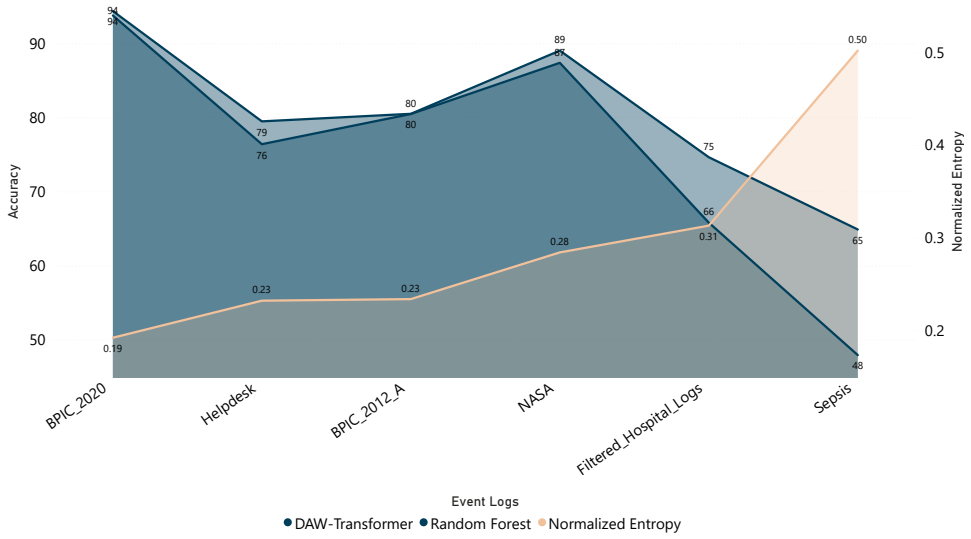


Figure 4.3: Model accuracy across event logs: Random Forest and DAW-Transformer perform similarly in low-entropy logs, while DAW-Transformer shows clear advantages in high-entropy logs.

cess predictability. A low entropy value reflects structured behavior where the current activity largely determines the next activity, while high entropy values indicate unstructured or flexible processes with less predictable activity sequences.

Our experimental observations (Figure 4.4) revealed a consistent pattern which in datasets with relatively lower entropy values (e.g., NASA, BPIC 2012 A, Helpdesk, BPIC 2020), simpler models such as Decision Trees and Random Forests achieved competitive performance compared to deep learning models. By contrast, in datasets exhibiting higher entropy values (e.g., Sepsis, Filtered Hospital Logs), the DAW-Transformer consistently outperformed classical methods, highlighting its ability to capture complex dependencies in unstructured processes.

At the lower end of the entropy spectrum, processes are highly structured and well served by interpretable models. At the higher end, where event logs display significant variability and uncertainty, more expressive deep learning architectures such as the DAW-Transformer become necessary. Table 4.1 summarizes the entropy values of the datasets, illustrating how they align with observed model performance.

Among our datasets, Sepsis (0.50) and Filtered Hospital Logs (0.31) exhibited rela-

Table 4.1: Conditional and Normalized Entropy of Event Logs

Event Log	**Conditional Entropy**	**Normalized Entropy**
Sepsis	2.0084	0.5021
Filtered_Hospital_Logs	1.8792	0.3132
NASA	1.5794	0.2843
BPIC_2012_A	0.7771	0.2339
Help desk	0.8839	0.2322
BPIC_2020_Prepaid Travel Cost	0.9335	0.1922

tively higher entropy values, making them more challenging for interpretable models. In both cases, the DAW-Transformer consistently outperformed the baselines. For the Sepsis dataset, it achieved an accuracy of 64.88%, while Random Forest and Decision Tree reached 47.93% and 50.30%, respectively. Similarly, in the Filtered Hospital Logs, the DAW-Transformer achieved 74.63%, outperforming Random Forest (65.80%) and Decision Tree (67.32%). These differences highlight the advantage of advanced architectures in capturing long-range dependencies in high-entropy processes.

In contrast, performance differences across models were minimal in low-entropy datasets. For example, in the BPIC_2020_Prepaid Travel Cost dataset (normalized entropy: 0.19), accuracies were nearly identical: 93.93% for Decision Tree, 93.76% for Random Forest, and 94.35% for the DAW-Transformer. A similar trend was observed in the Helpdesk dataset (normalized entropy: 0.23), where the DAW-Transformer achieved 79.48%, only slightly higher than XGBoost (79.19%) and Decision Trees (77.43%). These results suggest that in highly structured, low-entropy processes, simpler models are sufficient, as they provide competitive accuracy without the computational overhead of deep learning architectures.

In this study, in addition to accuracy, recall, precision, and F1-score are also calculated. Recall measures the proportion of true positives captured by the model, reflecting its ability to identify all relevant events in the dataset. High recall is essential in predictive process monitoring because missing rare but critical activities, such as infrequent patient treatments in healthcare, can lead to incomplete or misleading predictions. The results confirm this: in low-entropy logs such as BPIC 2020

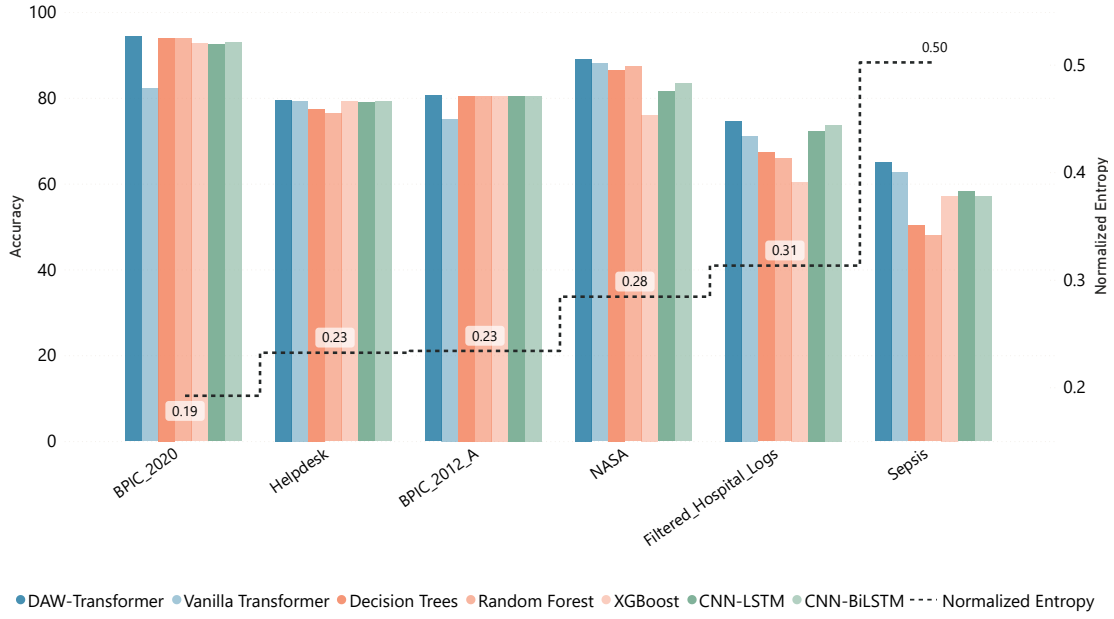


Figure 4.4: Model accuracy across event logs: Decision Tree, Random Forest, and DAW-Transformer perform similarly in low-entropy logs, while DAW-Transformer shows clear advantages in high-entropy logs.

and Helpdesk, recall is uniformly high across all models, reflecting that the structured nature of the process makes it relatively easy to capture most transitions. In contrast, in high-entropy datasets like Sepsis, classical models exhibit significantly reduced recall, meaning they fail to capture a substantial portion of rare events. DAW-Transformer, however, maintains higher recall because it models entire process traces and uses attribute-level embeddings to learn subtle variations in event sequences. Conceptually, recall here highlights the weakness of simple models in high-entropy environments: they tend to focus on frequent transitions and ignore outliers, while DAW-Transformer ensures broader coverage, thereby better capturing the variability inherent in unstructured processes.

Precision reflects the correctness of positive predictions (Figure 4.5). This metric is particularly important in contexts where false positives are costly, such as recommending a wrong treatment step or allocating the wrong resource. The results show that in low-entropy datasets, precision is consistently strong across models,

confirming that structured processes produce predictable transitions that are easy to classify correctly. However, in high-entropy datasets, the differences become more meaningful. Simpler models achieve lower precision because they misclassify rare activities as frequent ones, thereby increasing false positives. The DAW-Transformer, in contrast, achieves slightly higher precision while simultaneously improving recall. Conceptually, this demonstrates that DAW-Transformer not only broadens coverage of rare activities but also maintains the reliability of its predictions. Precision thus reinforces the observation that in complex processes, advanced architectures are capable of expanding recall without inflating error rates, a balance that simpler models struggle to achieve.

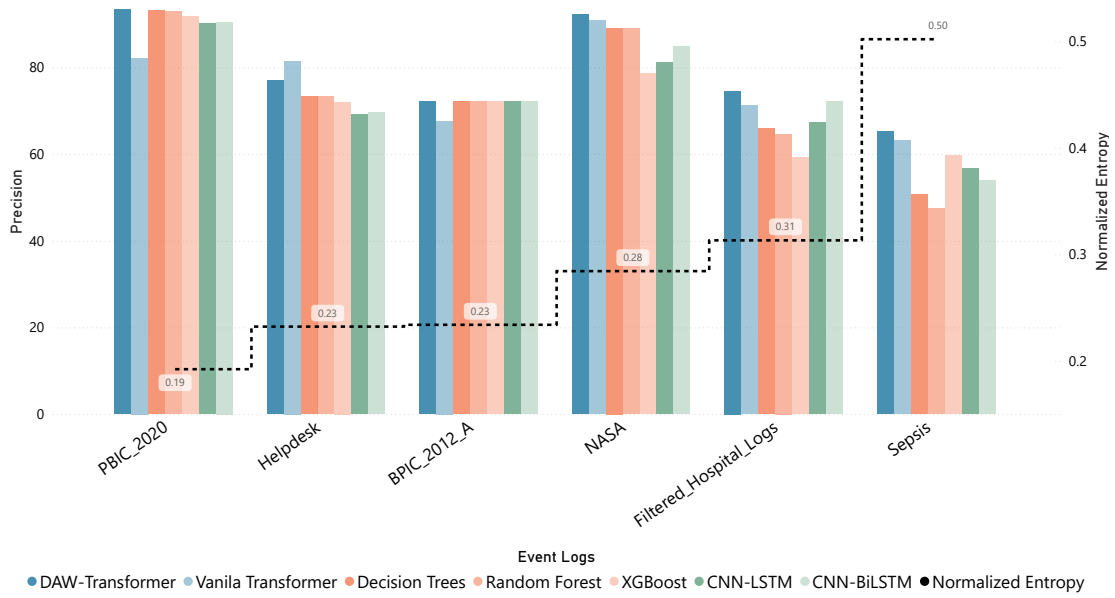


Figure 4.5: Model precision across event logs: Decision Tree, Random Forest, and DAW-Transformer perform similarly in low-entropy logs, while DAW-Transformer shows clear advantages in high-entropy logs.

The F1-score, defined as the harmonic mean of precision and recall, captures the balance between identifying as many true cases as possible (recall) and ensuring that predictions are correct (precision). This balance is particularly critical in event logs where class distributions are imbalanced, as overemphasizing one dimension can

misrepresent true performance. The results show that in low-entropy datasets such as BPIC_2020, Helpdesk, and BPIC_2012_A, most models achieve similar F1-scores, indicating that both precision and recall are consistently high and stable in structured processes. In these cases, even simpler models are able to maintain a good balance, which explains why their F1-scores converge with those of more advanced models. However, in high-entropy datasets like Sepsis and Filtered Hospital Logs, the gap between DAW-Transformer and classical models becomes pronounced. The DAW-Transformer achieves higher F1-scores because it avoids the trade-off where classical models achieve high precision by overpredicting frequent activities but suffer from low recall when missing rare transitions. By integrating multiple attributes and modeling long-range dependencies, the DAW-Transformer better balances recall and precision, which directly translates into superior F1-scores. Conceptually, this shows that F1 is a sensitive indicator of entropy-driven complexity: the more unstructured the process, the more difficult it becomes for simple models to preserve balance, while advanced architectures maintain stability.

Overall, the results demonstrate a consistent pattern: the DAW-Transformer yields substantial gains in high-entropy event logs, with improvements of up to 29% compared to classical models, while in low-entropy datasets the performance of simpler methods converges closely with that of advanced models. Figure 4.4 visualizes the relationship between event logs entropy and model performance. The results show that in datasets with relatively lower normalized entropy values, simpler interpretable models such as Decision Trees perform competitively with advanced architectures. However, as entropy increases and processes become more variable and less predictable, the DAW-Transformer achieves substantially higher accuracy than the interpretable baselines.

4.5 Discussion

The results confirm that process entropy is a reliable indicator for selecting predictive models in business process management (BPM). By aligning model complexity with dataset entropy, the proposed framework achieves a balance between predic-

tive accuracy and interpretability. In structured, low-entropy settings, models like Decision Trees provide transparent, rule-based logic that domain experts can readily validate. This interpretability enables users to trust predictions, justify decisions, and comply with regulatory requirements. By contrast, although the DAW-Transformer achieves superior accuracy in high-entropy scenarios, its internal mechanisms are less transparent. The entropy-driven framework thus ensures interpretability is preserved whenever possible, while complexity is introduced only when strictly necessary.

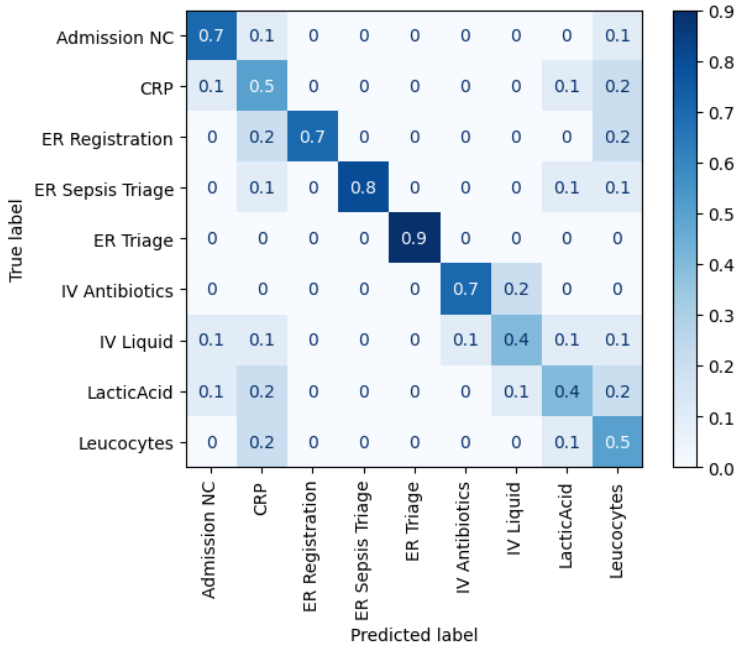
Based on results, in high-entropy event logs (Sepsis), the DAW-Transformer yielded a 29% accuracy improvement over simpler models. Confusion matrix analysis (Figure 4.6) further illustrates this effect: the DAW-Transformer produced a denser diagonal compared to Random Forest. These findings underscore the necessity of deep architectures in highly variable, unpredictable processes.

By contrast, in the BPIC_2020_Prepaid Travel Cost dataset (normalized entropy = 0.19), a Decision Tree achieved accuracy nearly identical to the DAW-Transformer while generating simple, interpretable decision rules. Figure 4.7 presents a simplified three-level Decision Tree for this dataset, illustrating how transparent, rule-based structures can represent next activity predictions in an easily understandable way. Such interpretable outputs allow auditors and managers to directly validate predictions against organizational policies.

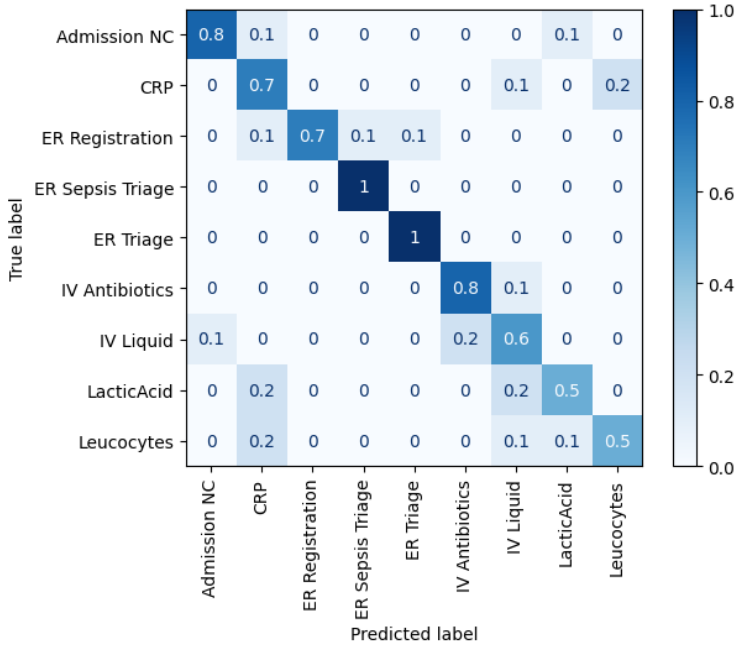
Similarly, in the Helpdesk dataset, all models achieved comparably high accuracy. In these structured, low-entropy settings, simpler models like Decision Trees and Random Forests are preferable, as they provide rule-based predictions that domain experts can readily interpret and trust.

Overall, these results demonstrate a complementary relationship between model types:

- For high-entropy datasets, DAW-Transformer captures complex, long-range dependencies that interpretable models cannot.
- For low-entropy datasets, Decision Trees provide competitive accuracy while offering clear, case-level decision rules that enhance stakeholder trust and usability.



(a) Random Forest



(b) DAW-Transformer

Figure 4.6: The Sepsis confusion matrix demonstrates improved performance

By presenting both types of results, the entropy-driven framework not only optimizes predictive accuracy but also ensures that interpretability is preserved whenever

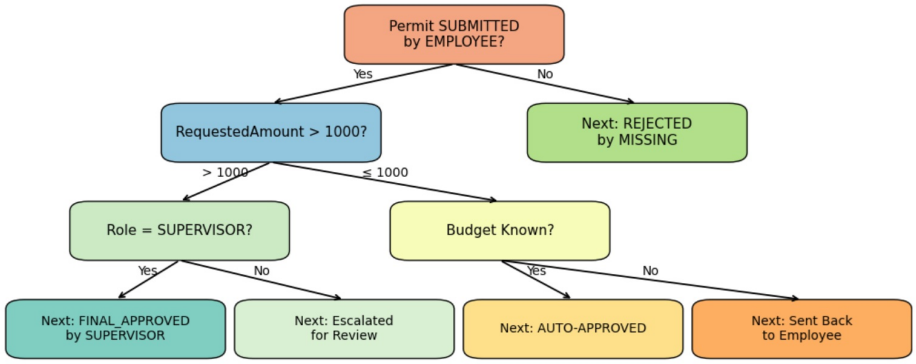


Figure 4.7: 3-level decision tree for BPIC_2020_Prepaid Travel Cost

possible.

Beyond accuracy, we assessed precision, recall, and F1-scores to provide a more complete evaluation (Table 3.11). These metrics also confirmed the entropy-driven selection model. On high-entropy datasets, the DAW-Transformer consistently maintained balanced precision and recall, achieving the strongest F1-scores. Simpler models, in contrast, exhibited weaker recall, reflecting frequent misclassification of valid activities. In low-entropy datasets, however, differences across models were negligible, with F1-scores varying by less than 2%.

Taken together, the analysis of precision, recall, and F1-scores reinforces the entropy-driven model selection framework. In structured, low-entropy processes, simpler models provide competitive accuracy and balanced classification performance. However, in unstructured, high-entropy settings, advanced models such as the DAW-Transformer are necessary to achieve both higher accuracy and improved reliability across all evaluation metrics.

The key implication of these findings is that entropy can guide adaptive model selection: employ interpretable models for structured processes, and apply more complex architectures only when required by process complexity.

4.6 Conclusions

The experimental results confirm that process entropy serves as a meaningful indicator for guiding model selection. As illustrated in Figure 4.4, event logs with lower normalized entropy values (e.g., BPIC 2020, Helpdesk, BPIC 2012 A, and NASA) demonstrate relatively structured and predictable behavior. In these cases, simpler and more interpretable models, such as Decision Trees and Random Forests, achieve predictive performance comparable to that of more complex architectures. In contrast, high-entropy event logs such as Sepsis (0.50) and Filtered Hospital Logs (0.31) exhibit greater variability and uncertainty, where the DAW-Transformer shows clear advantages, achieving substantially higher accuracy than classical models.

These findings support the central hypothesis of this research: entropy can be used to distinguish between structured and unstructured processes and to inform the choice of predictive models accordingly. Although the separation between high- and low-entropy processes exists on a spectrum rather than at a fixed boundary. This conclusion reinforces the utility of entropy as both a theoretical measure of process complexity and a practical tool for adaptive model selection in predictive business process monitoring.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis explored the challenge of next activity prediction in Business Process Management (BPM) by addressing two complementary research questions: (1) How can predictive models be improved to handle the complexity of modern event logs? and (2) How can organizations decide which predictive models are most appropriate for their processes?

To address the first research question, Chapter 3 introduced the DAW-Transformer, an extension of the Transformer architecture specifically designed for process prediction. The model integrates multiple event attributes directly into the attention mechanism and employs a dynamic windowing strategy to adaptively capture dependencies across different temporal contexts. Experimental results across six benchmark event logs (Sepsis, Filtered_Hospital Logs, NASA, BPIC_2012_A, Helpdesk, and BPIC_2020_Prepaid Travel Cost) demonstrated that the DAW-Transformer consistently outperforms both traditional machine learning methods (Decision Trees, Random Forests, XGBoost) and deep learning baselines (CNN–LSTM, CNN–BiLSTM, Vanilla Transformer). The performance gains were especially notable in unstructured, high-variability processes such as Sepsis and Filtered Hospital Logs, where the DAW-Transformer achieved significant improvements in accuracy, precision, recall, and F1-score. These results confirm the value of attribute integration and dynamic

windowing as key innovations in predictive business process monitoring.

To address the second question, Chapter 4 proposed an Entropy-Driven Model Selection Framework. Conditional entropy was employed as a quantitative measure of process variability and uncertainty, and was normalized to allow comparison across datasets of varying sizes and activity counts. The analysis revealed a consistent relationship between entropy and predictive performance: low-entropy datasets (e.g., NASA, BPIC_2012_A, Helpdesk, BPIC_2020_Prepaid Travel Cost) exhibited structured and predictable patterns, where simpler interpretable models provided performance comparable to advanced architectures. In contrast, high-entropy datasets (e.g., Sepsis, Filtered Hospital Logs) required the representational power of deep learning models, with the DAW-Transformer emerging as the most effective. Rather than a fixed threshold, entropy should be understood as a continuous spectrum: lower values indicate more structured processes, for which simple, interpretable models suffice, whereas higher values correspond to more variable processes, for which advanced architectures demonstrate clear advantages. This relationship was visually reinforced in Figure 4.3, where the divergence in model performance aligned closely with increasing entropy levels.

By integrating the outcomes of Chapters 3 and 4, this thesis contributes both a novel predictive architecture and a principled framework for model selection. The DAW-Transformer demonstrates how architectural advances can improve performance in complex scenarios, while the entropy framework provides a systematic method to decide when such advanced methods are necessary. Model complexity is introduced only when justified by process entropy, thereby addressing the persistent tension between interpretability and predictive accuracy.

Beyond its technical contributions, this research carries broader implications for organizations and society. By enabling more accurate and adaptive process predictions, the proposed methods can help businesses and public institutions respond more rapidly to critical events, allocate resources more efficiently, and prevent costly delays or failures. In domains such as healthcare, manufacturing, and public administration, these capabilities translate into earlier detection of process bottlenecks, improved service quality, and data-driven decision-making grounded in transparency

and efficiency.

Ultimately, this thesis contributes to making predictive process analytics not only more powerful but also more interpretable and accessible, bridging the gap between advanced AI models and real-world operational needs.

5.2 Limitations

While the proposed framework presents an explainable and practical approach for guiding model selection in predictive business process monitoring, several limitations should be acknowledged.

First, its current implementation still relies on human expertise when selecting models. This dependency may limit scalability and consistency across different use cases.

Second, the DAW-Transformer and other Transformer-based architectures generally require large datasets to capture meaningful patterns effectively. In situations where event logs are small or incomplete, model performance may deteriorate due to limited representational capacity.

Ultimately, the Entropy-Driven Model Selection Framework primarily focuses on achieving both predictive accuracy and interpretability. Other important aspects, such as computational efficiency, fairness, and energy consumption, were not explicitly considered in this study.

These limitations highlight several avenues for future research and development aimed at further refining and enhancing the proposed framework. Addressing these challenges would not only improve the accuracy, robustness, and interpretability of predictive models in complex business processes but also expand their applicability across diverse real-world domains. Ultimately, overcoming these constraints will contribute to more adaptive, transparent, and scalable solutions for predictive business process monitoring.

5.3 Future Work

Future research can extend this work along several important directions.

First, the proposed framework can be enhanced by incorporating multi-criteria decision-making mechanisms. In addition to predictive performance, future studies should consider factors such as robustness, consistency, and resilience of models under varying process conditions. This includes defining criteria for both inclusion and exclusion of models based on stability across datasets and sensitivity to noise, enabling more reliable and generalizable model selection.

Second, further research is needed to advance the automation of the model selection process. This involves developing strategies that balance conservative and non-conservative decision-making. For instance, a conservative approach may prioritize stable and interpretable models, while a less conservative approach may favor high-performance but more complex models. Designing adaptive mechanisms that dynamically navigate this trade-off would improve both usability and trust in real-world deployments.

Third, future work can explore the integration of advanced Transformer-based architectures, including large language models (LLMs), into predictive process monitoring. Leveraging pre-trained models and transfer learning techniques may improve the representation of sequential and contextual information in event logs, particularly in complex or data-scarce environments.

Additionally, to address data limitations, future research could incorporate data augmentation techniques to improve model robustness when training data are limited.

Moreover, expanding the entropy-driven framework to include broader performance dimensions, such as computational efficiency, energy consumption, and fairness, would support its application in sustainable and responsible AI systems.

Finally, an important direction for future work is to enhance actionability. Integrating reinforcement learning or closed-loop control mechanisms could enable systems not only to predict future activities but also to recommend or automatically implement optimal process adaptations in real time.

Chapter 6

Additional Information

6.1 Preface

This thesis corresponds to work that has been submitted to the Journal of Information Systems and subsequently published as a pre-print on ArXiv. The paper, entitled "An Innovative Next Activity Prediction Using Process Entropy and Dynamic Attribute-Wise-Transformer in Predictive Business Process Monitoring" appears under the citation [66]. As the lead researcher, I led the work on conceptual development, algorithm design, programming, visualization, and manuscript preparation. Mostafa Abbasi, a PhD student in the ACIS lab at the University of Victoria, made major contributions to the algorithm development, visualization, and writing. Maryam Ahang offered essential support and was actively involved in generating ideas. My supervisor, Homayoun Najjaran, oversaw the work and provided critical feedback during the manuscript revision stages.

Bibliography

[1] Luka Abb, Peter Pfeiffer, Peter Fettke, and Jana-Rebecca Rehse. A discussion on generalization in next-activity prediction. In *International Conference on Business Process Management*, pages 18–30. Springer, 2023.

[2] Mostafa Abbasi, Rahnuma Islam Nishat, Corey Bond, John Brandon Graham-Knight, Patricia Lasserre, Yves Lucet, and Homayoun Najjaran. A review of ai and machine learning contribution in business process management (process enhancement and process improvement approaches). *Business Process Management Journal*, 2024.

[3] Khaled A Alaghbari, Mohamad Hanif Md Saad, Aini Hussain, and Muhammad Raisul Alam. Activities recognition, anomaly detection and next activity prediction based on neural networks in smart homes. *IEEE Access*, 10(1):28219–28232, 2022.

[4] Lerina Aversano, Mario Luca Bernardi, Marta Cimitile, Martina Iammarino, and Chiara Verdone. A data-aware explainable deep learning approach for next activity prediction. *Engineering Applications of Artificial Intelligence*, 126(1):106758, 2023.

[5] Gérard Biau and Luc Devroye. On the layered nearest neighbour estimate, the bagged nearest neighbour estimate and the random forest method in regression and classification. *Journal of Multivariate Analysis*, 101(10):2499–2518, 2010.

[6] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

[7] Alfredo Bolt, Wil MP van der Aalst, and Massimiliano De Leoni. Finding process variants in event logs: (short paper). In *On the Move to Meaningful Internet Systems. OTM 2017 Conferences: Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part I*, pages 45–52. Springer, 2017.

[8] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[9] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] Leo Breiman, Jerome Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Chapman and Hall/CRC, 2017.

[11] Zaharah A Bukhsh, Aaqib Saeed, and Remco M Dijkman. Processtransformer: Predictive business process monitoring with transformer network. *arXiv preprint arXiv:2104.00721*, 2021.

[12] Andrea Burattin. Process mining techniques in business environments. *Lecture Notes in Business Information Processing*, 207:220, 2015.

[13] Michelangelo Ceci, Pasqua Fabiana Lanotte, Fabio Fumarola, Dario Pietro Cavallo, and Donato Malerba. Completion time and next activity prediction of processes using sequential pattern mining. In *Discovery Science: 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings 17*, pages 49–61. Springer, 2014.

[14] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[15] Raffaele Conforti, Massimiliano De Leoni, Marcello La Rosa, Wil MP Van Der Aalst, and Arthur HM Ter Hofstede. A recommendation system for predicting risks across multiple business process instances. *Decision Support Systems*, 69:1–19, 2015.

[16] Vincenzo Dentamaro, Donato Impedovo, Giuseppe Pirlo, Gianfranco Semeraro, et al. Next activity prediction and elapsed time prediction on process dataset. In *Ital-IA*, pages 605–609, 2023.

[17] Irving DeToro and Thomas McCabe. How to stay flexible and elude fads. *Quality progress*, 30(3):55, 1997.

[18] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[19] Chiara Di Francescomarino, Chiara Ghidini, Fabrizio Maria Maggi, and Fredrik Milani. Predictive process monitoring methods: Which one suits me best? In *International conference on business process management*, pages 462–479. Springer, 2018.

[20] Nicola Di Mauro, Annalisa Appice, and Teresa MA Basile. Activity prediction of business process instances with inception cnn models. In *AI* IA 2019–Advances in Artificial Intelligence: XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19–22, 2019, Proceedings 18*, pages 348–361. Springer, 2019.

[21] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[22] Christoph Drodt, Sven Weinzierl, Martin Matzner, and Patrick Delfmann. Predictive recommining: Learning relations between event log characteristics and machine learning approaches for supporting predictive process monitoring. In *International Conference on Advanced Information Systems Engineering*, pages 69–76. Springer, 2023.

[23] Joerg Evermann, Jana-Rebecca Rehse, and Peter Fettke. A deep learning approach for predicting process behaviour at runtime. In *International Conference on Business Process Management*, pages 327–338. Springer, 2016.

[24] Giles M Foody. Challenges in the real world use of classification accuracy metrics: From recall and precision to the matthews correlation coefficient. *Plos one*, 18(10):e0291908, 2023.

[25] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005.

[26] Jingwei Guo, Wei Wang, Yinying Tang, Yongxiang Zhang, and Hengying Zhuge. A cnn-bi_lstm parallel network approach for train travel time prediction. *Knowledge-Based Systems*, 256:109796, 2022.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[28] Congfang Huang, Akash Deep, Shiyu Zhou, and Dharmaraj Veeramani. A deep learning approach for predicting critical events using event logs. *Quality and Reliability Engineering International*, 37(5):2214–2234, 2021.

[29] Donato Impedovo, Giuseppe Pirlo, and Gianfranco Semeraro. Next activity prediction: An application of shallow learning techniques against deep learning over the bpi challenge 2020. *IEEE Access*, 11(1):117947–117953, 2023.

[30] Mohammad Sabik Irbaz, Lutfun Nahar Lota, and Fardin Ahsan Sakib. Predicting user-specific future activities using lstm-based multi-label classification. In *Human Activity and Behavior Analysis*, pages 301–310. CRC Press, 2024.

[31] Jae-Yoon Jung. Measuring entropy in business process models. In *2008 3rd International Conference on Innovative Computing Information and Control*, pages 246–246. IEEE, 2008.

[32] Jae-Yoon Jung, Chang-Ho Chin, and Jorge Cardoso. An entropy-based uncertainty measure of process models. *Information Processing Letters*, 111(3):135–141, 2011.

[33] Bokyoung Kang, Dongsoo Kim, and Suk-Ho Kang. Periodic performance prediction for real-time business process monitoring. *Industrial Management & Data Systems*, 112(1):4–23, 2012.

[34] Asjad Khan, Hung Le, Kien Do, Truyen Tran, Aditya Ghose, Hoa Dam, and Renuka Sindhgatta. Deepprocess: supporting business process execution using a mann-based recommender system. In *International Conference on Service-Oriented Computing*, pages 19–33. Springer, 2021.

[35] Sungkyu Kim, Marco Comuzzi, and Chiara Di Francescomarino. Explaining the impact of design choices on model quality in predictive process monitoring. *Journal of Intelligent Information Systems*, pages 1–26, 2024.

[36] Tae-Young Kim and Sung-Bae Cho. Predicting residential energy consumption using cnn-lstm neural networks. *Energy*, 182:72–81, 2019.

[37] Kundan Krishna, Deepali Jain, Sanket V Mehta, and Sunav Choudhary. An lstm based system for prediction of human activities with durations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–31, 2018.

[38] Roy G Lee and Barrie G Dale. Business process management: a review and evaluation. *Business process management journal*, 4(3):214–225, 1998.

[39] Maikel Leemans. Nasa crew exploration vehicle (cev) software event log, 2017. Dataset.

[40] Philipp Leitner, Johannes Ferner, Waldemar Hummer, and Schahram Dustdar. Data-driven and automated prediction of service level agreement violations in service compositions. *Distributed and Parallel Databases*, 31(3):447–470, 2013.

[41] Felix Mannhardt. Sepsis cases - event log, 2016.

[42] Alfonso E Márquez-Chamorro, Manuel Resinas, Antonio Ruiz-Cortés, and Miguel Toro. Run-time prediction of business process indicators using evolutionary decision rules. *Expert Systems with Applications*, 87:1–14, 2017.

[43] Tahani Hussein Abu Musa and Abdelaziz Bouras. Prediction of next events in business processes: A deep learning approach. In *IFIP International Conference on Product Lifecycle Management*, pages 210–220. Springer, 2023.

[44] Ameet Annasaheb Rahane and Anbumani Subramanian. Measures of complexity for large scale image datasets. In *2020 international conference on artificial intelligence in information and communication (ICAIIC)*, pages 282–287. IEEE, 2020.

[45] Efrén Rama-Maneiro, Juan C Vidal, and Manuel Lama. Deep learning for predictive business process monitoring: Review and benchmark. *IEEE Transactions on Services Computing*, 16(1):739–756, 2021.

[46] Sebastian Raschka. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*, 2018.

[47] Aleksandra Revina and Ünal Aksu. An approach for analyzing business process execution complexity based on textual data and event log. *Information Systems*, 114:102184, 2023.

[48] Yan-Yan Song and LU Ying. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130, 2015.

[49] Alessio Staffini. A cnn–bilstm architecture for macroeconomic time series forecasting. *Engineering Proceedings*, 39(1):33, 2023.

[50] Xiaoxiao Sun, Siqing Yang, Yuke Ying, and Dongjin Yu. Next activity prediction of ongoing business processes based on deep learning. *Expert Systems*, 41(5):e13421, 2024.

[51] Bayu Adhi Tama and Marco Comuzzi. An empirical comparison of classification techniques for next event prediction using business process event logs. *Expert Systems with Applications*, 129:233–245, 2019.

[52] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with lstm neural networks. In *International conference on advanced information systems engineering*, pages 477–492. Springer, 2017.

[53] Chris J Turner, Ashutosh Tiwari, Richard Olaiya, and Yuchun Xu. Process mining: from theory to practice. *Business process management journal*, 18(3):493–512, 2012.

[54] Boudewijn van Dongen. Real-life event logs - hospital log, 2011. Dataset.

[55] Boudewijn van Dongen. Bpi challenge 2012, 2012. Dataset.

[56] Boudewijn van Dongen. Bpi challenge 2020: Prepaid travel costs, 2020.

[57] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[58] Ilya Verenich. Helpdesk, 2016. Available at: `https://doi.org/10.17632/39bp3vv62t.1`.

[59] Ilya Verenich, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, and Chiara Di Francescomarino. Minimizing overprocessing waste in business processes via predictive activity ordering. In *International conference on advanced information systems engineering*, pages 186–202. Springer, 2016.

[60] Jiaojiao Wang, Jiawei Huang, Xiaoyu Ma, Zhongjin Li, Yaqi Wang, and Dingguo Yu. Mtlformer: Multi-task learning guided transformer network for business process prediction. *IEEE Access*, 2023.

[61] Jiaxing Wang, Chengliang Lu, Bin Cao, and Jing Fan. Mitfm: A multi-view information fusion method based on transformer for next activity prediction of business processes. In *Proceedings of the 14th Asia-Pacific Symposium on Internetware*, pages 281–291, 2023.

[62] Sven Weinzierl, Sebastian Dunzer, Sandra Zilker, and Martin Matzner. Prescriptive business process monitoring for recommending next best actions. In *Inter-*

national conference on business process management, pages 193–209. Springer, 2020.

[63] Sven Weinzierl, Sandra Zilker, Jens Brunk, Kate Revoredo, Martin Matzner, and Jörg Becker. Xnap: making lstm-based next activity predictions explainable by using lrp. In *International Conference on Business Process Management*, pages 129–141. Springer, 2020.

[64] Allen R Wilcox. Indices of qualitative variation. Technical report, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 1967.

[65] Mohamed Zairi. Business process management: a boundaryless approach to modern competitiveness. *Business process management journal*, 3(1):64–80, 1997.

[66] Hadi Zare, Mostafa Abbasi, Maryam Ahang, and Homayoun Najjaran. An innovative next activity prediction approach using process entropy and daw-transformer. *arXiv preprint arXiv:2502.10573*, 2025.