# InferNet: Next Likely Action Prediction in Business Processes

Srinivasan Kidambi[1][0009−0001−7517−2080],
Gautam Vashishtha[1][0009−0008−8699−8348], and
R. P. Jagadeesh Chandra Bose[1][0000−0001−9441−1414]

Skan AI Labs Pvt Ltd, Bengaluru, India
https://www.skan.ai
kidambisrivatsan5@outlook.com
{gautam.v,jc.bose}@skan.ai

**Abstract.** Predictive business process monitoring aims to forecast future characteristics of an ongoing process by analyzing event data. Gaining foresight into process execution holds significant potential for enhancing operational efficiency, optimizing resource management, and delivering more effective customer services. Next activity (action) prediction is one of the fundamental problems in predictive process monitoring. Most research on this problem involved the use of deep-learning models. However, these *black-box* models often fall short in terms of explainability and temporal reasoning. In this paper, we explore the utility of Dynamic Bayesian Networks (DBN) as a *white-box* alternative to bridge the gap between performance and interpretability. A key challenge in using DBNs has been their struggle in handling unseen data combinations. In this paper, we introduce *InferNet*, a Dynamic Bayesian Network based approach that tackles unseen data combinations alongside providing the benefits of explainability of DBNs. We introduce a novel estimation approach called *score-based marginalization* to handle unseen data that leverages the network structure and evaluates parent contributions to improve the accuracy of posterior distribution estimation. We demonstrate that our approach significantly surpasses existing white-box methods and either outperforms or competes closely with more complex deep-learning-based black-box models while having the potential to provide explainable insights into process execution.

**Keywords:** Bayesian Network · Unseen Evidence · Activity Prediction · Posterior Estimation

## 1 Introduction

Process execution data, also known as Business Process (BP) logs, is growing exponentially due to the trend towards digital transformation and the accessibility of comparatively less expensive storage options. These logs have been analyzed using process mining techniques to discover, monitor, and enhance business processes. Organizations are aggressively investing in predictive analytics solutions in addition to post-hoc analysis to obtain performance insights. With an emphasis on projecting future features of an ongoing business process, predictive

business process monitoring, or PBPM, has become a key field in process mining. Effective resource management, increased operational effectiveness, process automation, and the ability to prevent impasse by forecasting the anticipated next action, its duration, and its remaining time to completion are just a few of the many beneficial business uses of PBPM. In this work, we focus on the next (most likely) action prediction for business processes. Fig. 1 illustrates the idea. For a given case (process instance), the problem is to be able to predict the event likely to happen at position $e_i$ based on the history of the case. One may use a historical context of a certain length ($k$) and the characteristics (e.g., activity name, resource, data attributes etc) of these events to assist in the prediction. A robust next activity prediction model is essential for process automation, as it allows systems to automatically carry out tasks with minimal human input.

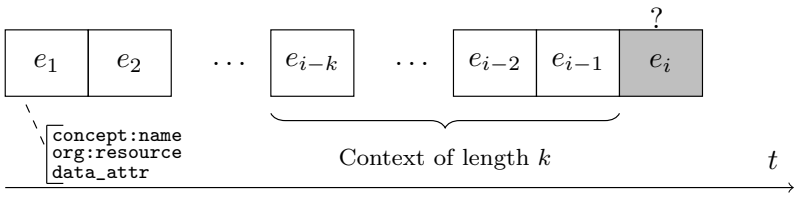


**Fig. 1.** Illustration of the next likely action prediction problem.

Deep neural networks have been extensively used for business process monitoring activities. Recurrent neural networks (RNNs) were introduced by Evermann et al. [7] to anticipate process behavior at runtime. Subsequently, numerous prediction models based on RNNs and other variants, like long-short term memory (LSTM) networks, have been presented for the tasks of output prediction [14], case remaining time prediction [10,14], suffix creation [5,4], and future activity prediction [6,5,10].

While deep learning based PBPM techniques are well-established for event sequence modeling, they have a significant drawback when it comes to achieving transparency/explainability in how they produce their outputs and visualizing the learnt causation and correlations particularly when it comes to the challenge of predicting the next activity. This is primarily due to the extensive latent parametrization in the existing DNN frameworks, causing them to function as black-box models.

To address the issue of lacking transparency and explainability, we introduce InferNet, based on Dynamic Bayesian Networks (DBNs), to model the temporal and conditional dependencies of events. A key challenge in using DBN lies in the unseen estimation where the chosen parent combination has not been seen *as-is* by the model during the training phase. To handle this we introduce a novel unseen estimation approach called *score-based marginalization*. This method leverages the network structure and evaluates parent contributions to significantly improve the accuracy of posterior distribution estimation. Through extensive experiments on standard event logs like BPIC-15 and BPIC-20 as well as large-scale event logs like BPIC-18, we demonstrate that our score-based marginalization method significantly outperforms traditional unseen estimation techniques such as Marginalization and Naive Bayes. Additionally, our approach surpasses

state of the art white-box models and competes closely/outperforms various black-box neural network based methods, all while offering a more compact and explainable model for predicting the next activity.

The remainder of the paper is organized as follows. Related work is presented in Section 2. Modeling next activity prediction using dynamic bayesian network is described in Section 3. Section 4 introduces the concept of score based marginalization for handling unseen data. Experimental results are discussed in Section 5. Finally, Section 6 concludes the paper with pointers to some future directions.

## 2 Related Work

Probabilistic models for predictive process monitoring have been widely studied due to their interpretability and explainability. Becker et al. [2] and Breuker et al. [3] use probabilistic finite automatons and hidden Markov models for state prediction. Lakshmanan et al. [9] proposed an instance-specific Probabilistic Process Model convertible to a Markov Chain. While these models are understandable, they often struggle with long-term dependencies due to limited event sequence data. Deep learning techniques, inspired by NLP successes, have been applied to Business Process Management.

Many state-of-the-art methods use LSTM cells [5,8,7,1,11], which can retain relevant information over time in terms of long and short term memory. To improve efficiency, Di Mauro et al. [6] proposed using Convolutional Neural Networks leveraging parallel processing abilities from kernel convolutions. Camargo et al. [5] explored lower-dimensional vectors combining activity and resource information, while Lin et al. [10] introduced the Modulator layer for effective attribute prediction. Hinkka et al. [8] addressed complexity by clustering events based on attribute values. However, neural network approaches sacrifice explainability, which is crucial in process management. To address this, Pauwels et al. [12] introduced extended Dynamic Bayesian Networks (eDBN) for anomaly detection in process logs. They later extended this work for next activity prediction [13], but their method struggles with unseen combinations of parent elements. Our work addresses this issue by introducing a score-based marginalization approach to improve posterior distribution estimation.

## 3 Dynamic Bayesian Network Modeling

In this section, we introduce the concept of Bayesian networks and Dynamic Bayesian Networks (DBNs), which serve as the foundation of our approach. Bayesian network is a probabilistic graphical model that represents a set of variables, $\mathcal{X} = \{X_1, X_2, \dots X_n\}$, and their conditional dependencies through a directed acyclic graph (DAG). Each node in the graph corresponds to a variable, while the edges signify direct dependencies between these variables. An edge $X_i \rightarrow X_j$ indicate that the variable $X_j$ is conditionally dependent on the variable $X_i$. The joint probability distribution of the entire network can then be expressed as:

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Parents}(X_i)) \tag{1}$$

where Parents($X_i$) represents the set of parent nodes (those with edges pointing to $X_i$). The factorization of the joint probability distribution into product of conditional probabilities of each variable $X_i$ is possible as the Bayesian network captures the corresponding dependence and independence among the variables, thereby making the computation and inference efficient.

DBNs are a class of probabilistic graphical models that extend traditional Bayesian networks by incorporating temporal dynamics, making them particularly well-suited for modeling time-dependent processes. The temporality is typically expressed as a context of length $k$, i.e., consider a window of events of length $k$. The next activity prediction problem can be modeled using the DBNs as follows: the activity at event $e_t$ will potentially depend on the previous events and their characteristics (attributes). For example, the `activity` name of an event at timestamp $t$ may depend on the `activity` name, `resource`, and `title` attribute values of the previous events. We model these attributes as nodes of the network. Fig. 2 represents a DBN with a context length of 2, i.e., the attributes

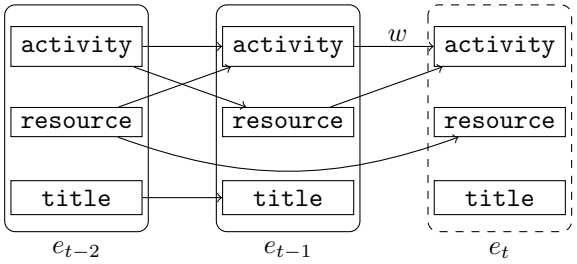


**Fig. 2.** Example of a Dynamic Bayesian Network for a context length of 2. Slices at $t-i$ show the $i^{th}$ event in history from event $t$.

of events (such as `activity`, `resource` and `title`) from events at timestamps $t-2$ and $t-1$ are used to predict the event attributes at timestamp $t$. The edges depict the dependencies between the temporal event attributes and carry a weight $w$, signifying the strength of the dependency, e.g., the model says that the `activity` at event $e_t$ is dependent on the `resource` and `activity` at $e_{t-1}$.

### 3.1 Conditional Probability Tables (CPTs)

Conditional Probability Tables (CPTs) are critical components of Bayesian networks that specify the probability distributions of each node given its parent nodes. Each CPT defines the likelihood of a particular state of a node based on the states of its parent nodes, capturing the conditional dependencies within the network. Here the posteriors are found from the data using frequency of occurrence as a proxy for the actual posterior distribution. In DBNs, CPTs are particularly important because they allow the model to represent the temporal evolution of the system by encoding how the probability of a node's state at the current time step is influenced by the states of its parents at the current and previous time steps. Table 1 illustrates a Conditional Probability Table (CPT) for predicting the next `activity` given the $\texttt{activity}_{t-1}$ and $\texttt{resource}_{t-1}$ as parents, i.e., the `activity` name at step $t$ is dependent on the `resource` at step $t-1$

and `activity` at step $t-1$. The table depicts the different combinations of these parent nodes and the associated probabilities of various activities. As given in Table 1, given the evidence of ($\texttt{activity}_{t-1}, \texttt{resource}_{t-1}$) as `(a, resource2)`, we observe the third row and conclude that the most likely next activity prediction is **b** given its maximum likelihood (**0.6**).

**Table 1.** Illustration of Conditional Probability Table

| **Parents** activity$_{t-1}$, resource$_{t-1}$ \ **activity**$_t$ | a | b | c | end |
|---|---|---|---|---|
| `(start, resource1)` | 1.0 | 0.0 | 0.0 | 0.0 |
| `(start, resource2)` | 0.8 | 0.1 | 0.1 | 0.0 |
| `(a, resource2)` | 0.0 | 0.6 | 0.3 | 0.1 |
| `(a, resource3)` | 0.0 | 0.4 | 0.4 | 0.2 |
| `(b, resource1)` | 0.1 | 0.0 | 0.7 | 0.2 |
| `(b, resource4)` | 0.2 | 0.0 | 0.8 | 0.0 |
| `(c, resource4)` | 0.0 | 0.2 | 0.0 | 0.8 |

### 3.2 Training the Dynamic Bayesian Network

Learning a DBN encompasses both structure learning and parameter estimation. Parameter estimation in DBNs refers to the process of calculating the numerical values that quantify the relationships between nodes in the network. The structure learning process typically employs a greedy search strategy, utilizing local scoring metrics such as the Bayesian Information Criterion (BIC) or Categorical Scoring. At each iteration, the algorithm evaluates potential edge additions, deletions, or reversals, selecting the modification that maximizes the chosen scoring function. This process continues until the incremental improvement in score falls below a predefined threshold, indicating convergence. The scoring metrics balance model fit against complexity, mitigating overfitting and promoting parsimony in the learned network structure. For training our model, we use Categorical Scoring as the scoring function with the score improvement threshold set as 0. The preference for Categorical Scoring over regularizing functions like BIC stems from the relatively modest size of BPIC logs. Given the data-intensive nature of DBNs, regularizing scores tend to induce underfitting in these datasets, compromising model performance. Once the DAG structure is established, the Conditional Probability Tables (CPTs) are populated to complete the DBN specification.

## 4 Prediction and Handling Unseen Data

Once a DBN has been trained using historical data to model the relationships and build the CPTs for inputs with context length $k$, this model is used for predicting the next activity for test data with the same context length. Although CPTs encapsulate all parent node value combinations observed in the training dataset,

unseen combinations frequently emerge during inference which require robust estimation techniques for probability calculation. Existing methods [13] utilize marginalization and naive-Bayes approaches, which often yield suboptimal predictions for next activities. These approaches struggle when individual parent values are observed but their specific combination is missing, largely due to their assumption of node independence. This assumption is particularly problematic in the context of business process logs, where inter-attribute correlations are both common and significant. To address this, we introduce a novel *score-based marginalization* (SM) approach to leverage the network structure and evaluate parent contributions to improve the posterior distribution estimation.

### 4.1 Marginalization

Marginalization is an estimation technique for scenarios where one or more parent values are unseen in the training data. The process involves:

1. Iterating over all observed values in the Conditional Probability Table (CPT) for parent nodes with unseen values.
2. Maintaining fixed values for seen parent nodes.
3. Calculating the activity probability as a weighted sum over all possible combinations of unseen parent values.

The marginalization equation for estimating the probability of an activity $A$ given partially unseen evidence is:

$$P(A|\mathcal{X}(\text{Parents}_s), \text{Parents}_u) = \frac{1}{N} \sum_{v_u \in V(\text{Parents}_u)} P(A|\mathcal{X}(\text{Parents}_s), v_u) \quad (2)$$

where:

- $A$ is the activity being predicted
- $\text{Parents}_s$ represent the set of seen (observed) parent nodes
- $\mathcal{X}(\text{Parents}_s)$ represents the value combination of the seen parent nodes in the given context.
- $\text{Parents}_u$ represent the set of unseen parent nodes
- $V(\text{Parents}_u)$ is the set of all possible value combinations for the unseen nodes
- $v_u$ is a specific value combination for the unseen nodes
- N is the number of rows where $\mathcal{X}(\text{Parents}_s)$ is seen in the CPT

This equation computes the probability of activity $A$ by summing over all possible combinations of unseen parent values ($v_u$). The term $P(A|\mathcal{X}(\text{Parents}_s), v_u)$ represents the probability of an activity given both seen parent values and a specific value combination for the unseen parent nodes. These probabilities are estimated from the CPTs generated from the training data.

We illustrate this with an example. Consider the CPT in Table 1. Assume we are given the evidence, `(b, resource5)`. We note that we have seen the value `b` for $\texttt{activity}_{t-1}$ but have not seen `resource5` for $\texttt{resource}_{t-1}$. Hence, we choose to marginalize over the $\texttt{resource}_{t-1}$ field as:

$$P(A \mid (\texttt{b}, -)) = \frac{1}{N} \sum_{\texttt{r} \in \texttt{resource}_{t-1}} (P(A \mid (\texttt{b}, \texttt{r})))$$

where $N$ is the number of rows where $\texttt{activity}_{t-1}$ is `b`. In our CPT, we have two rows where $\texttt{activity}_{t-1}$ is `b`, viz., `(b, resource1)` and `(b, resource4)`. Therefore, our calculation becomes:

$$P(A \mid (\texttt{b}, -)) = \frac{1}{2}\left(P(A \mid (\texttt{b}, \texttt{resource1})) + P(A \mid (\texttt{b}, \texttt{resource4})\right)$$

We compute this for all possible values of activity, $A$ (i.e., `a`, `b`, `c` and `end`) and return the activity with the maximum probability. $P(\texttt{a} \mid (\texttt{b}, -)) = \frac{1}{2}(0.1 + 0.2) = 0.15$. Similarly, we get $P(\texttt{b} \mid (\texttt{b}, -)) = 0.0$, $P(\texttt{c} \mid (\texttt{b}, -)) = 0.75$ and $P(\texttt{end} \mid (\texttt{b}, -)) = 0.1$. From this we conclude that our prediction of the next activity given the unseen evidence (`b`, `resource5`) is `c`, with a likelihood **0.75**.

**Limitation:** The key limitation of the marginalization approach lies in its treatment of unseen parent value combinations. Assume an unseen evidence `(a1,b1,c1)` for parents `(a,b,c)` respectively. If our CPT contains partial evidences such as `(a1,b2,c1)` and `(a2,b1,c2)`, this approach would marginalize `b` in `(a1,b2,c1)` thus losing information coming from the evidence `(a2,b1,c2)` that includes `b1`. More generally, marginalization causes loss of potentially informative evidences due to not considering the importance of each partial evidence in the prediction. Our empirical analysis reveals that these partial match scenarios are prevalent, significantly reducing the method's effectiveness in real-world applications.

### 4.2 Naïve Bayes

To address the limitation of marginalization, Naïve Bayes assumes independence among parent nodes. Individual posteriors are computed for each observed parent value, then combined to estimate the final posterior distribution:

$$P(A|v_1, v_2, ..., v_n) = \prod_{i=1}^{n} P(A|v_i) \tag{3}$$

Where $A$ is the activity being predicted, and $v_i$ are the values of the parent nodes for the current context.

Let us look at an example based on the CPT provided in Table 1. Assume we are given the evidence `(b, resource2)`. We note that we have seen the values `b` for $\texttt{activity}_{t-1}$ and `resource2` for $\texttt{resource}_{t-1}$ in different combinations but not together. Hence, we choose to estimate independent posteriors over each parent using marginalization and later combine them. Considering $\texttt{activity}_{t-1}$ field alone, using marginalization equation (2)

$$P(A \mid (b, -)) = \frac{1}{N} \sum_{\texttt{r} \in \texttt{resource}_{t-1}} (P(A \mid (\texttt{b, r})))$$

where $N$ is the number of rows given $\texttt{activity}_{t-1}$ is `b`. We compute this for different values of $A \in \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{end}\}$. $P(\texttt{a} \mid (\texttt{b, -})) = \frac{1}{2} \cdot (0.1 + 0.2) = 0.15$. On similar lines, $P(\texttt{b} \mid (\texttt{b, -})) = 0.0$, $P(\texttt{c} \mid (\texttt{b, -})) = 0.75$, and $P(\texttt{end} \mid (\texttt{b, -})) = 0.1$.

Now we consider the second parent $\texttt{resource}_{t-1}$, independently and assess the probability of activity $A$ as

$$P(A \mid \texttt{(-, resource2)}) = \frac{1}{N} \sum_{\alpha \in \texttt{activity}_{t-1}} (P(A \mid (\alpha, \texttt{resource2})))$$

where $N$ is the number of rows given $\texttt{resource}_{t-1}$ is resource2.

$P(\texttt{a} \mid \texttt{(-,resource2)}) = \frac{1}{2} \cdot (0.8 + 0.0) = 0.4$. Similarly, $P(\texttt{b} \mid \texttt{(-, resource2)}) = 0.35$, $P(\texttt{c} \mid \texttt{(-, resource2)}) = 0.2$ and $P(\texttt{end} \mid \texttt{(-, resource2)}) = 0.05$.

We finally estimate the posterior assuming the two variables to be independent:

$$P(A \mid \texttt{(b, resource2)}) = P(A \mid \texttt{(b, -)}) \cdot P(A \mid \texttt{(-, resource2)})$$

This leads us to the following estimates for the four activities $P(\texttt{a} \mid \texttt{(b, resource2)}) = 0.15 \cdot 0.4 = 0.06$, $P(\texttt{b} \mid \texttt{(b, resource2)}) = 0.0$, $P(\texttt{c} \mid \texttt{(b, resource2)}) = 0.15$ and $P(\texttt{end} \mid \texttt{(b, resource2)}) = 0.005$. Based on these estimates, we predict that the next activity given the unseen evidence (b, resource2) is c, with a likelihood of **0.15**.

**Limitation:** The core assumption of independence between parent nodes is most often incorrect. In our experiments we observed the graph having links between various pairs of parent variables, directly invalidating this assumption of no correlation.

### 4.3 Score-Based Marginalization

In this section, we propose Score-Based Marginalization (SM), a novel technique that addresses the challenge faced by Marginalization and Naive-Bayes estimation. SM leverages network structure to estimate posterior distributions more accurately. The key insight of SM is to estimate the impact of each partial evidence when complete evidence is unavailable. This is achieved through a systematic evaluation of potential partial evidences based on their structural importance in the network. The steps involved in this estimation process are as follows:

1. Given an unseen evidence configuration, we generate all possible partial evidences (subsets of unseen evidence) from it which are present in the Conditional Probability Table (CPT)
2. Given an evidence $(a1, b1, c1, \ldots, m1)$ having parent attributes $(a, b, c, \ldots, m)$, a partial evidence is defined as all those evidences that have at least one matching attribute value i.e same parent attribute having same parent attribute value. We define $O$ as the set of observed variables For example : $(a1, b2, c2, d2, \ldots, m2)$ is a partial evidence sharing $a1$ as a common value for parent attribute $a$. Thus $O = \{a\}$ Similarly, $(a1, b2, c2, d2, \ldots, m1)$ is a partial evidence sharing $(a1, m1)$. Thus $O = \{a, m\}$
3. For each compatible partial evidence $E$, we compute a score $S(E)$:

$$S(E) = \sum_{\hat{p} \in O} \text{edge\_score}(\hat{p} \to c) \tag{4}$$

where $c$ is the target child node (the predictor variable), and edge_score($\hat{p} \rightarrow c$) is the weight of the edge from parent $\hat{p}$ to child $c$ in the network structure.

4. The surrogate prediction for the unseen evidence $E^u$ is made using partial evidences such that:

$$P(A|E^u) = \sum_{E \in \mathcal{E}} \overline{S(E)} P(A|E) \tag{5}$$

where $A$ is the next activity, $P(A|E)$ is the surrogate posterior prediction for evidence $E$, $\mathcal{E}$ is the set of all partial evidences found in the CPT corresponding to $E^u$, $\overline{S(E)}$ is the normalized score for partial evidence $E$.

This formulation ensures that SM prioritizes evidence from the most influential parents, as determined by the network's overall structure and parameters.

Let us illustrate this with an example based on the CPT in Table 1. Assume we are given the evidence $E^u =$ `(b, resource2)`. We have observed the values `resource2` for $\texttt{resource}_{t-1}$ and `b` for $\texttt{activity}_{t-1}$ in different combinations, but not together. From the CPT, we can say that:

$$\mathcal{E} = \{(\texttt{b}, \texttt{resource1}), (\texttt{b}, \texttt{resource4}), (\texttt{a}, \texttt{resource2})\}$$

and the set of observed variables $O$ will be $\{\texttt{activity}_{t-1}\}, \{\texttt{activity}_{t-1}\}$, and $\{\texttt{resource}_{t-1}\}$ respectively. Assume that the score of the edge $\texttt{activity}_{t-1} \rightarrow \texttt{activity}_t$ is 40 and $\texttt{resource}_{t-1} \rightarrow \texttt{activity}_t$ is 20. Then we can say that,

$$\begin{aligned} P(A|\texttt{(b,resource2)}) = {} & 0.4 * P(A|\texttt{(b,resource1)}) \\ & + 0.4 * P(A|\texttt{(b,resource4)}) \\ & + 0.2 * P(A|\texttt{(a,resource2)}) \end{aligned}$$

Substituting with the base probabilities from the CPT in Table 1 for every activity of $A \in \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{end}\}$, we get

$$\begin{aligned} P(\texttt{a}|\texttt{(b,resource2)}) &= 0.4 * 0.1 + 0.4 * 0.2 + 0.2 * 0.0 = 0.12, \\ P(\texttt{b}|\texttt{(b,resource2)}) &= 0.4 * 0.0 + 0.4 * 0.0 + 0.2 * 0.6 = 0.12, \\ P(\texttt{c}|\texttt{(b,resource2)}) &= 0.4 * 0.7 + 0.4 * 0.8 + 0.2 * 0.3 = 0.66, \\ P(\texttt{end}|\texttt{(b,resource2)}) &= 0.4 * 0.2 + 0.4 * 0.0 + 0.2 * 0.1 = 0.10, \end{aligned}$$

From this we conclude that our prediction of the next activity given the unseen evidence `(b,resource2)` is `c`, with a likelihood **0.66**.

## 5 Experiments and Discussion

In this section, we present our experimental study and discuss the results. We assessed the efficacy of our proposed approach for next activity prediction against several real-life event logs. We report the results from three sets of event logs viz., the bpi challenge (BPIC) 2015,2018 and 2020. The details of the logs are presented in Appendix A.1.

We assess the effectiveness of our approach against state-of-the-art next activity prediction models. It is important to note that apart from our model, [13] is the only other **white-box** (non parametrized) predictive model. All the other models are state-of-the-art **black-box** models for next activity prediction.

Table 2 depicts the test accuracy for the different BPIC_15 logs for context lengths of $k = 5$ and $k = 10$. As can be seen, our approach outperforms all the other models for context length 10 while our model is the second best approach for all datasets for context length 5 with an average difference of 2.2% from the best model by [6]. Furthermore, we notice that the next best activity prediction accuracy slightly reduces for a context length of 10 when compared to 5. This indicates that the process executions typically depend on shorter contexts. It is also important to note that white-box approaches perform significantly better than all black box approaches barring [6]. This could be attributed to the fact that the BPIC_15 logs are complex with the number of distinct activities being in the order of a few hundreds. We hypothesise that the most complex deep learning models with the exception of [6] couldn't converge well owing to the limited number of contexts vis-a-vis the number of distinct activities (the number of contexts are not sufficient enough for learning the interplay between the activities).

**Table 2.** Prediction accuracy of the different approaches on BPIC 15 event logs. The best score is highlighted in **bold** and the second best is underlined.

| **Method** | **Context Length** = 5 | | | | | **Context Length** = 10 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 15_1 | 15_2 | 15_3 | 15_4 | 15_5 | 15_1 | 15_2 | 15_3 | 15_4 | 15_5 |
| Our Approach | 57.52 | 54.56 | 61.11 | 61.25 | 58.03 | **57.12** | **52.23** | **59.30** | **59.11** | **56.12** |
| Pauwels et. al. [13] | 53.19 | 47.05 | 56.29 | 55.84 | 52.96 | 52.08 | 47.08 | 54.59 | 53.00 | 50.49 |
| Tax et. al. [14] | 55.21 | 50.63 | 58.21 | 57.99 | 54.51 | 54.61 | 49.11 | 57.71 | 56.50 | 53.68 |
| Lin et. al. [10] | 53.40 | 43.81 | 53.31 | 51.46 | 51.40 | 54.58 | 47.90 | 57.51 | 54.16 | 52.95 |
| Dimauro et. al. [6] | **59.73** | **54.85** | **64.86** | **63.84** | **61.68** | 55.85 | 50.24 | 55.31 | 54.25 | 51.12 |
| Proc.Transformer et. al [4] | 54.87 | 51.45 | 57.91 | 57.06 | 53.17 | 52.08 | 47.08 | 54.59 | 53.00 | 50.49 |

Table 3 depicts the prediction accuracy for BPIC_20 logs. As can be seen, our approach outperforms all other models for a context length of 5. For a context length of 10, our model performs the best for BPIC 20_5 logs and is the second best in most other datasets with the average accuracy difference being 1.5% as compared to the best model. Tax et al. [14] performs better for the majority of logs. This can be attributed to the fact that the number of distinct activities for BPIC_20 logs are very few (of the order of tens) and the contexts capture almost all of the variants (this is also evident from the percentage of unseen contexts being low from Table 4) (in Appendix A.1), thereby making the deep learning models overfit. The fact that a context length of 5 exhibits better prediction accuracy also indicates that these processes are relatively simpler with short dependencies.

To consider the efficacy of our approach over long term dependencies, Fig. 3. shows the comparative performance of different approaches on BPIC_18 logs which capture long term dependencies and larger logs, as is evident from the

data characteristics as seen in appendix Table 4. Fig. 3. shows that our approach performs at par with complex black-box approaches even when trained on limited proportion of data where it is evident that the other white-box approach [13] fails to keep up due to the inherent data-hungry nature of Bayesian models which is alleviated by SM inferencing. We further investigate the effect of inferencing mechanisms in DBNs with varying percentage of unseen contexts for BPIC 18 log. As seen from Fig. 4, as the percentage of unseen contexts increases, the performance of marginalization as used by [13] and naive-bayes fails drastically as opposed to our approach (SM) that is able to withstand the unseen contexts by leveraging the learnt network structure to enhance posterior distribution estimation.

To illustrate the explainable nature of our approach, Fig. 5 showcases the temporal relation between the predicted activity 'revoke approval' and its parent activities. Due to the explicit graphical structure, one can easily identify the correlating activities as well as their effect on the predicted node, which is otherwise not possible in black-box modeling frameworks due to their latent modeling paradigm. Here we can see that the activity 'refuse' and 'insert document' contributes maximally to the decision of revoking approval. Interestingly, we can observe the gradual decline of contribution as the temporal gap increases. This demonstrates how the model captures the diminishing influence of earlier activities on later outcomes. This representation allows for clear interpretation of process flow and activity importance in predicting the final decision, enhancing model explainability compared to black-box approaches.

**Table 3.** Prediction accuracy of the different approaches on BPIC 20 event logs. The best score is highlighted in **bold** and the second best is underlined.

| **Method** | **Context Length** = 5 | | | | | **Context Length** = 10 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 20_1 | 20_2 | 20_3 | 20_4 | 20_5 | 20_1 | 20_2 | 20_3 | 20_4 | 20_5 |
| Our Approach | **93.95** | **86.90** | **81.98** | **94.10** | **93.30** | 87.65 | 91.11 | 81.42 | 88.88 | **92.44** |
| Pauwels et. al. [13] | 93.79 | 86.60 | 81.04 | 93.84 | 92.10 | 86.42 | 90.98 | 80.68 | 80.68 | 88.00 |
| Tax et. al. [14] | 93.16 | 83.76 | 78.17 | 91.82 | 92.20 | **90.12** | **91.99** | **83.66** | 82.22 | 90.22 |
| Lin et. al. [10] | 83.35 | 83.00 | 78.17 | 93.52 | 89.72 | 90.12 | 90.46 | 82.78 | 84.44 | 85.33 |
| Dimauro et. al. [6] | 85.97 | 86.39 | 80.38 | 86.76 | 86.11 | 87.22 | 89.91 | 80.47 | 83.22 | 86.16 |
| Proc.Transformer et. al [4] | 91.10 | 81.75 | 76.82 | 91.20 | 84.57 | 87.31 | 86.31 | 78.69 | **89.41** | 87.91 |

## 6 Conclusions and Future Directions

In this paper, we extended the dynamic Bayesian network framework for next activity prediction by incorporating a score-based marginalization approach to handle unseen evidence. Our empirical results demonstrate significant potential with this enhancement. Accurate next action prediction can pave the way for intelligent automation and process co-pilots, playing a crucial role in enterprise transformation. As future work, we aim to extend this framework from process mining event logs to task mining event data, where the granularity of events is much finer, capturing actions such as widget clicks. Task mining presents unique

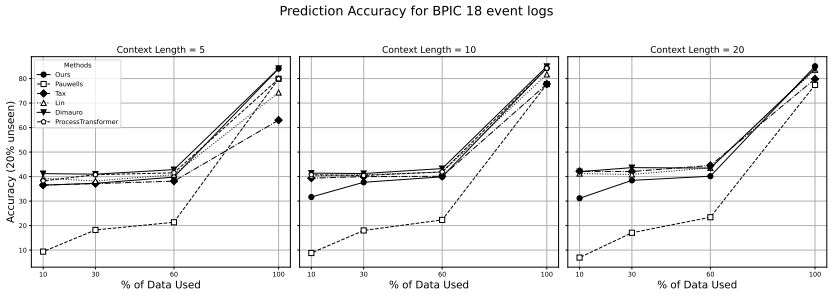


**Fig. 3.** Comparision of Prediction Accuracy on BPIC 18 log.

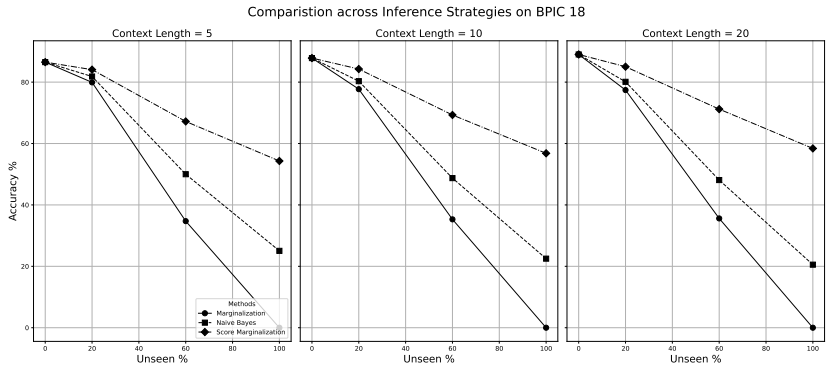


**Fig. 4.** Comparision against existing inference methods across varying percentage of unseen contexts for BPIC 18 log.

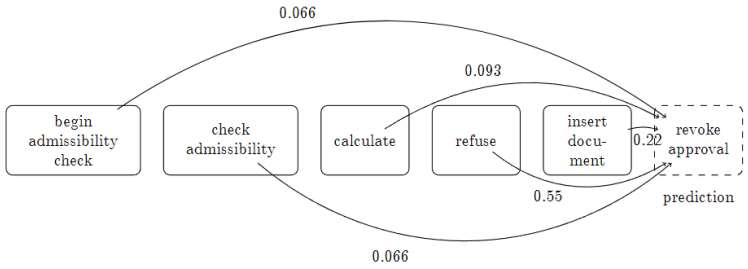


**Fig. 5.** Illustration of Explainable capabilities of Our Approach.

challenges, including handling repetitive widget actions and managing an event space that is significantly larger than the activity space in process mining logs. Additionally, in this paper, we primarily focused on modeling with only the activity name and resource attributes of events. Moving forward, we plan to incorporate all available data attributes, which will necessitate efficient feature selection techniques to reduce the complexity of training and inference while also improving accuracy.

## References


1. Aversano, L., Bernardi, M.L., Cimitile, M., Iammarino, M., Verdone, C.: A data-aware explainable deep learning approach for next activity prediction. Engineering Applications of Artificial Intelligence **126**, 106758 (2023)
2. Becker, J., Breuker, D., Delfmann, P., Matzner, M.: Designing and implementing a framework for event-based predictive modelling of business processes (2014)
3. Breuker, D., Matzner, M., Delfmann, P., Becker, J.: Comprehensible predictive models for business processes. Mis Quarterly **40**(4), 1009–1034 (2016)
4. Bukhsh, Z.A., Saeed, A., Dijkman, R.M.: Processtransformer: Predictive business process monitoring with transformer network. arXiv preprint arXiv:2104.00721 (2021)
5. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate lstm models of business processes. In: Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17. pp. 286–302. Springer (2019)
6. Di Mauro, N., Appice, A., Basile, T.M.: 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. pp. 348–361. Springer (2019)
7. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decision Support Systems **100**, 129–140 (2017)
8. Hinkka, M., Lehto, T., Heljanko, K.: Exploiting event log event attributes in rnn based prediction. In: International Symposium on Data-Driven Process Discovery and Analysis. pp. 67–85. Springer (2018)
9. Lakshmanan, G.T., Shamsi, D., Doganata, Y.N., Unuvar, M., Khalaf, R.: A markov prediction model for data-driven semi-structured business processes. Knowledge and information systems **42**, 97–126 (2015)
10. Lin, L., Wen, L., Wang, J.: Mm-pred: A deep predictive model for multi-attribute event sequence. In: Proceedings of the 2019 SIAM international conference on data mining. pp. 118–126. SIAM (2019)
11. Pasquadibisceglie, V., Appice, A., Castellano, G., Malerba, D.: A multi-view deep learning approach for predictive business process monitoring. IEEE Transactions on Services Computing **15**(4), 2382–2395 (2021)
12. Pauwels, S., Calders, T.: An anomaly detection technique for business processes based on extended dynamic bayesian networks. In: Proceedings of the 34th ACM/SIGAPP symposium on applied computing. pp. 494–501 (2019)
13. Pauwels, S., Calders, T.: Bayesian network based predictions of business processes. In: Business Process Management Forum: BPM Forum 2020, Seville, Spain, September 13–18, 2020, Proceedings 18. pp. 159–175. Springer (2020)
14. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with lstm neural networks. In: Advanced Information Systems Engineering: 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings 29. pp. 477–492. Springer (2017)


# A Appendix

### A.1 Dataset Characteristics

The BPIC_15 comprises of a set of 5 logs pertaining to building permit process from 5 different Dutch municipalities while the BPIC_20 event logs pertain to

the travel expense claims. There are five variants of the travel expenses, viz., domestic (BPIC_20_1) and international declarations (BPIC_20_2), travel permits (BPIC_20_3), requests for payment (BPIC_20_4) and pre-paid travel costs (BPIC_20_5).The BPIC_18 dataset contains event logs from a loan application process in a Dutch financial institute, detailing various stages of application handling, including approvals, rejections, and intermediate steps. The characteristics of each of these logs are presented in Table 4. Contexts of different lengths were generated using an overlapping sliding window. For each event, we considered the `concept:name` attribute and `org:resource` attribute, i.e., the activity name and the resource as input characteristics for modeling the DBNs to predict the next `concept:name`. The training is done using a 80:20 ratio split for train/test data on BPIC_15 and BPIC_20 logs and the same data split has been used to train and evaluate all models. For BPIC_18, the training is done using a 90:10 ratio split for train/test data on all models. To report the results for context lengths 5 and 10 for [13], [10], [14], we have used the code provided by Pauwels et al. [13]. For [6] and [4], we have used the code provided by the respective authors.

**Table 4.** Characteristics of event logs used for our study.

| Characteristic | BPIC 15 | | | | | BPIC 20 | | | | | BPIC 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 15_1 | 15_2 | 15_3 | 15_4 | 15_5 | 20_1 | 20_2 | 20_3 | 20_4 | 20_5 | |
| events | 52217 | 44354 | 59681 | 47293 | 59083 | 56437 | 72151 | 86581 | 36796 | 18246 | 2514266 |
| cases | 1199 | 832 | 1409 | 1053 | 1156 | 10500 | 6449 | 7065 | 6886 | 2099 | 43809 |
| distinct activities | 398 | 410 | 383 | 356 | 389 | 17 | 34 | 51 | 19 | 29 | 41 |
| contexts ($k = 5$) | 46236 | 40202 | 52648 | 42041 | 53303 | 6384 | 39926 | 51285 | 5799 | 8264 | 2295221 |
| contexts ($k = 10$) | 40368 | 36084 | 45926 | 38829 | 47545 | 267 | 10270 | 21151 | 1880 | 1285 | 2076176 |
| contexts ($k = 20$) | - | - | - | - | - | - | - | - | - | - | 1638086 |
| unseen contexts (%) ($k = 5$) | 36.05 | 42.15 | 30.83 | 32.60 | 35.35 | 00.47 | 00.95 | 05.23 | 00.69 | 04.11 | 17482 |
| unseen contexts (%) ($k = 10$) | 39.33 | 49.95 | 40.71 | 35.24 | 41.52 | 01.86 | 01.07 | 16.23 | 00.53 | 12.84 | 23908 |
| unseen contexts (%) ($k = 20$) | - | - | - | - | - | - | - | - | - | - | 21364 |

### A.2 Training Algorithm for DBN

The psuedocode for the DBN graph building algorithm is shown below:

## B DBN inferencing with SM algorithm

The psuedocode for the inferencing from DBN using the novel proposed SM algorithm is shown below:

Some points of note in the testing algorithm are:

- SM refers to the novel score based marginalization, detailed in Section 4.3.
- The 'UNKNOWN' prediction for any feature is the value 0 in all experiments for this paper. Thus all string encodings start from 1 for each feature.

**Algorithm 1** DBN Graph Building

```
Input: all-edges
Output: graph
improvement ← true
graph ← ()
while improvement do
    add_best ← ()
    best_add_delta ← 0
    for edge in all-edges do
        if edge not in graph then
            if score(graph + edge) - score(graph) > best_add_delta then
                best_add_delta ← score(graph + edge) - score(graph)
                add_best ← edge
            end if
        end if
    end for
    del_best ← ()
    best_del_delta ← 0
    for edge in graph do
        if score(graph - edge) - score(graph) > best_del_delta then
            best_del_delta ← score(graph - edge) - score(graph)
            del_best ← edge
        end if
    end for
    if best_add_delta > best_del_delta then
        graph ← graph + add_best
    else if best_del_delta > 0 then
        graph ← graph - del_best
    else
        improvement ← false
    end if
end while
return graph
```

**Algorithm 2** DBN inferencing with SM

```
Input: test_context, prediction_fields, cpts, parents
Output: next_event
for field in prediction_fields do
    cpt ← cpts[field]
    parent_combination ← test_context(parents(field))
    if parent_combination in cpt then
        field_val ← max_w(cpt(parent_combination))
    else if at_least_one_seen(parent_combination) then
        probs ← SM(parent_combination, field)
        field_val ← max_w(probs)
    else
        field_val ← UNKNOWN
    end if
    next_event(field) ← field_val
end for
return next_event
```