Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Hybrid CNN–SPN Framework for Reliable Monitoring and Instability Detection in Hadoop Clusters

Walid Ben Mesmia

benmesmiawalid77@yahoo.fr

ENIT

Zied Trifa

University of Sfax

Kamel Barkaoui

CNAM

Method Article

Keywords: Hadoop cluster monitoring, Stochastic Petri Nets, Convolutional Neural Networks, hybrid modeling, distributed systems, node instability detection, synthetic data generation, Classification, explainable AI

Posted Date: August 6th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10567366/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Hybrid CNN–SPN Framework for Reliable Monitoring and Instability Detection in Hadoop Clusters

Walid Ben Mesmia[1*], Zied Trifa[2†] and Kamel Barkaoui[3†]

[1*]LR-Sys'Com-ENIT, ENIT, Tunis, 1002, Tunisia.
[2]MIRACL Research Laboratory, University of Sfax, Sfax, 3029, Tunisia.
[3]CEDRIC-CNAM, CNAM, Paris, France.

*Corresponding author(s). E-mail(s): benmesmiawalid77@yahoo.fr;
Contributing authors: trifa.zied@gmail.com; kamel.barkaoui@cnam.fr;
†These authors contributed equally to this work.

Abstract

Reliable monitoring of Hadoop clusters requires models that can capture both the structural dynamics of distributed node behavior and the stochastic variability of workloads, failures, and recovery processes. Existing approaches based solely on rule-based monitoring, statistical modeling, or machine learning often face complementary limitations: threshold-based methods lack adaptability, formal stochastic models provide limited predictive flexibility, and deep learning approaches frequently sacrifice interpretability. To address these limitations, this paper proposes a hybrid Convolutional Neural Network–Stochastic Petri Net (CNN–SPN) framework for classifying Hadoop nodes as stable or unstable. The proposed approach combines a multi-density Stochastic Petri Net, using exponential, normal, log-normal, Poisson, and Weibull firing-time distributions, with a convolutional neural network that learns discriminative patterns from node-level monitoring features. The framework further incorporates a hybrid information fusion mechanism and a dynamic firing strategy to couple stochastic state evolution with learned feature representations. The model is evaluated on a reproducible synthetic dataset of **10,000** node observations generated through SPN-based simulation, and its components are examined through ablation analysis. The results show that integrating the SPN state and hybrid fusion improves performance over a CNN-only baseline, while the complete framework achieves the best overall scores, with **68.10%** accuracy, **68.91%** precision, **68.78%** recall, **68.85%** F1-score, and a ROC-AUC of **0.7434**. Although real-cluster validation remains future work, the proposed framework provides an interpretable

and structurally grounded basis for stochastic monitoring and node-instability detection in distributed Hadoop environments.

Keywords: Hadoop cluster monitoring, Stochastic Petri Nets, Convolutional Neural Networks, hybrid modeling, distributed systems, node instability detection, synthetic data generation, Classification, explainable AI

1 Introduction

Distributed computing infrastructures have become fundamental to large-scale data processing, cloud services, and enterprise analytics, making their reliability and operational resilience increasingly critical. Among these infrastructures, Hadoop clusters play a central role in storing and processing large volumes of data through coordinated interactions among worker nodes, storage subsystems, and resource-management services. Because these environments are highly distributed, concurrent, and dynamically loaded, failures or abnormal behaviors at the node level can quickly degrade performance, reduce availability, and compromise service continuity.

Effective monitoring of Hadoop cluster nodes is therefore essential for maintaining system reliability and enabling timely intervention. In practice, node instability may arise from overload conditions, disk or network congestion, task-execution anomalies, intermittent failures, or recovery delays, often under heterogeneous workload conditions. Detecting such instability at an early stage is challenging because node behavior is shaped not only by current measurements but also by stochastic event timing, concurrent interactions, and evolving operational context.

A wide range of approaches has been proposed for fault detection and monitoring in distributed systems. Rule-based and threshold-based methods remain attractive because they are simple to deploy and computationally inexpensive, but they are often sensitive to manual parameter selection and adapt poorly to changing workloads [1, 2]. Statistical and stochastic models, including Markovian approaches and Stochastic Petri Nets (SPNs), offer formal mechanisms for representing state transitions, concurrency, and uncertainty, yet they frequently rely on restrictive assumptions or limited predictive adaptation [3, 5, 6]. Machine learning techniques, including classical classifiers and deep neural models, have demonstrated strong predictive performance on monitoring data, but they often operate as black boxes and provide limited insight into the structural causes of abnormal behavior [4].

These limitations motivate the need for a hybrid monitoring framework that combines predictive learning with formal stochastic modeling. In such a framework, the learning component should extract discriminative patterns from observed metrics, while the formal component should capture event-driven dynamics, concurrency, rare failure modes, and stochastic timing behavior in a structured and interpretable way. This combination is particularly relevant for Hadoop clusters, where node instability can emerge from multiple interacting processes rather than from isolated metric deviations alone.

To address this problem, this paper proposes a hybrid Convolutional Neural Network–Stochastic Petri Net (CNN–SPN) framework for reliable monitoring of Hadoop cluster nodes. The proposed approach integrates a convolutional neural network with a multi-density SPN whose transitions are governed by exponential, normal, log-normal, Poisson, and Weibull firing-time distributions. Within this framework, the SPN models node-state evolution and generates stochastic traces reflecting normal and degraded operating conditions, while the CNN learns predictive representations from the resulting monitoring features. A hybrid information fusion mechanism and a dynamic firing strategy are introduced to strengthen the interaction between the formal stochastic representation and the learned decision process.

The proposed framework is designed to support three objectives. First, it provides a formal and stochastic representation of Hadoop node behavior, including idle, busy, overloaded, failed, and recovering states. Second, it supports data-driven classification of nodes as stable or unstable by exploiting both observed and SPN-derived signals. Third, it enables controlled generation of synthetic traces, including rare or extreme scenarios, which are difficult to capture consistently through real monitoring logs alone.

The empirical evaluation is conducted on a synthetic dataset of 10,000 node observations generated through the proposed SPN-based simulation framework. The study examines the effect of progressively integrating SPN state information, hybrid information fusion, and dynamic firing adaptation into the CNN pipeline. The results show that the hybrid design improves substantially over the CNN-only baseline, particularly when SPN state and fusion mechanisms are incorporated, while also offering a more structured interpretation of node behavior than purely black-box classifiers. At the same time, the paper explicitly recognizes that validation on real Hadoop traces remains necessary before drawing strong deployment-oriented conclusions.

The main contributions of this paper are as follows:

1. A hybrid modeling framework that integrates a Convolutional Neural Network (CNN) with a multi-density Stochastic Petri Net (SPN) to combine data-driven pattern learning and formal stochastic modeling for Hadoop node monitoring.
2. A multi-density SPN design that employs heterogeneous firing-time distributions (exponential, normal, log-normal, Poisson, Weibull) to represent diverse event timings—task execution, I/O, packet loss, failures, and recovery—and to generate controlled synthetic traces including rare and extreme scenarios.
3. A hybrid coupling mechanism that fuses SPN state information with CNN latent representations and a dynamic firing/adaptation strategy to strengthen interaction between formal state evolution and learned decision signals.
4. A reproducible simulation and evaluation pipeline, including an SPN-based synthetic dataset (10,000 observations) and an ablation study that quantifies the incremental benefit of SPN integration, hybrid fusion, and dynamic gating over a CNN-only baseline.
5. An empirical analysis showing that the hybrid architecture improves node-instability detection performance and interpretability relative to the CNN baseline, while explicitly identifying limitations and the need for real-cluster validation.

The remainder of the paper is organized as follows. Section 2 reviews related work on Hadoop monitoring, stochastic modeling, machine learning, and hybrid Petri net–learning approaches. Section 3 presents the proposed CNN–SPN framework. Section 4 describes the Hadoop monitoring case study and the formal representation of places, transitions, and tokens. Sections 5 detail the experimental setup, simulation-based validation, and model-integration mechanisms. Section 7 reports the empirical results and discusses their implications and limitations. Section 8 concludes the paper and outlines future research directions

2 Related Work

Fault detection and performance monitoring in Hadoop clusters and large-scale distributed computing environments have attracted considerable research attention over the past decade. Existing approaches can generally be grouped into four main categories: rule-based monitoring, statistical and stochastic modeling, machine learning techniques, and formal Petri net-based approaches.

2.1 Rule-Based Monitoring

Rule-based and threshold-based monitoring systems constitute the earliest solutions for Hadoop fault detection. These approaches generate alarms whenever monitored metrics exceed predefined thresholds. Their implementation is computationally inexpensive and easy to deploy; however, they remain highly dependent on manually selected thresholds and cannot easily adapt to changing workloads.

Goli *et al.* [1] combined threshold-based monitoring with statistical filtering and reported approximately 92% detection accuracy on synthetic Hadoop workloads. Similarly, Gaykar *et al.* [2] achieved a precision between 90% and 93% for detecting faulty HDFS nodes using production cluster traces. Although these methods provide satisfactory performance under stable operating conditions, their effectiveness decreases when workload characteristics evolve dynamically.

2.2 Statistical and Stochastic Modeling

Probabilistic approaches model the evolution of distributed systems through stochastic processes such as Markov chains, Bayesian models, or Stochastic Petri Nets (SPNs). These techniques provide a rigorous mathematical framework for reliability analysis and performance prediction.

Lee *et al.* [3] proposed DC-Prophet, a Markov-based prediction framework for catastrophic failures in large data centers, obtaining an F1-score of 0.87. Ben Mesmia *et al.* [5] employed SPNs for DevOps workflow analysis and reported prediction accuracies exceeding 85% for congestion events. Despite their strong analytical capabilities, purely stochastic models often require simplifying assumptions regarding transition firing distributions and may struggle to represent highly heterogeneous operating conditions.

2.3 Machine Learning Approaches

Recent studies increasingly rely on machine learning techniques to improve fault detection accuracy. Traditional classifiers such as Logistic Regression, Support Vector Machines, Random Forests, and Neural Networks have demonstrated excellent predictive capabilities on large monitoring datasets.

For example, Zainib *et al.* [4] employed Random Forest classifiers for sensor fault diagnosis and achieved approximately 95% classification accuracy with an AUC of 0.92. Likewise, Goli *et al.* [1] combined multiple Hadoop performance indicators with supervised learning techniques and reported detection accuracies approaching 94%.

Although these approaches achieve high predictive performance, they generally operate as black-box models, providing limited insight into the underlying system dynamics and making failure interpretation difficult.

2.4 Petri Net-Based Formal Modeling

Petri Nets and their stochastic extensions remain widely adopted for representing concurrent, distributed, and event-driven systems because of their formal semantics and analytical capabilities.

Ben Mesmia *et al.* [5] demonstrated that SPNs accurately model DevOps workflows and predict congestion scenarios with more than 85% accuracy. In subsequent work, Ben Mesmia *et al.* [6] modeled industrial production workflows using SPNs and obtained approximately 88% prediction precision with recall values above 0.83.

Nevertheless, conventional SPN models generally employ a single firing-time distribution for each transition and rely exclusively on analytical inference, limiting their adaptability when confronted with complex nonlinear behavior and previously unseen operating conditions.

2.5 Comparison with Hybrid Petri Net and Deep Learning Approaches

Several recent studies have attempted to combine Petri Nets with machine learning or deep learning techniques. However, these hybrid frameworks differ significantly in both their modeling assumptions and their integration strategies.

Table 1 compares representative hybrid approaches according to four characteristics that are directly related to the present work:

1. support for multiple stochastic firing distributions (multi-density SPN);
2. integration between the formal model and learned representations for explainability;
3. adaptive interaction between the Petri Net and the learning model during inference;
4. explicit consideration of rare or extreme operating conditions.

As shown in Table 1, previous hybrid approaches generally employ Petri Nets either as a static supervisory component [28], a fixed stochastic state model [29, 30], or a system-design representation independent of the deep learning process [31]. None simultaneously combines heterogeneous stochastic firing distributions, adaptive CNN–SPN interaction, and explicit synthetic generation of rare-event scenarios.

Table 1 Comparison between the proposed CNN–SPN framework and other hybrid Petri net / deep learning approaches in the literature.

Approach	**Application Domain**	**Multi-density SPN**	**Explainability (Formal ↔ Learned)**	**Dynamic Fusion**	**Rare-event Focus**
Volpe et al. [28] (LSTM + Petri Net)	Network intrusion detection / prevention	No (deterministic PN, no stochastic firing laws)	No (PN used only for flow control, not for explaining LSTM decisions)	No (fixed rule-based PN gating of LSTM output)	Partial (imbalanced attack detection, not explicitly targeted)
M-HSTPN–DL [29] (hybrid stochastic timed PN + DL)	Bearing fault diagnosis (hybrid systems)	Partial (stochastic and timed transitions, single family per transition)	No (DL used as a black-box decision model within the PN states)	No (PN structure fixed, DL model selected per hybrid situation)	No (focus on hybrid discrete/continuous state estimation)
mBSPN [30] (modified Bayesian Stochastic PN)	Fault diagnosis with feedback control loops	No (single generalized stochastic PN, no DL component)	Partial (Bayesian diagnostic features linked to PN transitions, but no deep learning)	No (static diagnostic PN structure)	No
Chang et al. [31] (PN + YOLOv5 + ResNet-50)	Visual object (bird) identification	No (PN used only for system design modeling, not stochastic transitions)	No (PN and CNN stages are sequential, not jointly explainable)	No	No
Proposed CNN–SPN	**Hadoop cluster node monitoring**	**Yes** (Exponential, Normal, Log-Normal, Weibull, Poisson)	**Partial** (HI-tk fusion weight as an aggregate explainability signal; per-transition T1–T8 mapping defined but not yet quantitatively validated, Section 7.4)	**Yes** (HI-tk attention-based fusion + HFiringT dynamic gate, learned end-to-end)	**Yes** (SPN-generated synthetic traces explicitly designed to cover rare/extreme regimes)

3 Proposed Model

To improve the robustness and interpretability of Hadoop node classification, we propose a hybrid framework combining Convolutional Neural Networks (CNN) and Stochastic Petri Nets (SPN). The CNN extracts latent features from system metrics, while the SPN models the stochastic and concurrent transitions between node states.

The CNN-SPN hybrid model For Cluster Hadoop Moniting, denoted $CNN - SPN_{hybrid}$, is the 13-uplets: $CNN - SPN_{hybrid}$=

$$[P, Tr, \phi, M_0, HTyp - dist, HKTyp - tk, HFringT, X, H, S, P, HI - tk, Loss]$$

Where:

- P: represents the finished places set;
- Tr: represents the finished transition set;
- ϕ: function that determines the firing transition rate;
- M_0: Petri Nets initial marking.
- $HTyp - dist$: the $HTyp - dist$ function allowing to distinguish between the distribution types assigned to the transitions:

$$HTyp - dist : Trs \rightarrow \{e, n, ln, p, w\}$$

$$k \mapsto HTyp - dist(k) = \begin{cases} e, & \text{(Exponential)} \\ n, & \text{(Normal)} \\ ln, & \text{(Log-Normal)} \\ p, & \text{(Poisson)} \\ w, & \text{(Weibull)} \end{cases}$$

- $HKTyp - tk$: the function makes a distinction between the different token types. In this context, the model distinguishes seven types of tokens, each capturing a specific operational aspect of a Hadoop node's behavior. To begin with, the first type of token denotes available processing capacity and unused computational resources. Moreover, the second type of token corresponds to the number of active tasks currently running on the node. In addition, the third type of token indicates excessive computational pressure and the intensity of overload conditions. Similarly, the fourth type of token represents concurrent disk I/O operations and storage-related contention. Likewise, the fifth type of token reflects the presence of network contention events and communication delays. Furthermore, the sixth type of token indicates node malfunction occurrences. Finally, an additional token type represents nodes undergoing restoration and the ongoing recovery process. In this context, the model distinguishes seven types of tokens, each capturing a specific operational aspect of a Hadoop node's behavior:

$$HKTyp - tk : tk \rightarrow \{1, 2, 3, 4, 5, 6, 7\}$$

$$tk \mapsto HKTyp - tk(tk) = \begin{cases} 1, & \text{represent available processing capacity;} \\ 2, & \text{indicate the number of active tasks on the node;} \\ 3, & \text{measure overload intensity;} \\ 4, & \text{count the concurrent I/O operations;} \\ 5, & \text{reflect network contention events;} \\ 6, & \text{indicate node failure occurrences;} \\ 7, & \text{represent nodes undergoing recovery.} \end{cases}$$

- $HFringT$: a function adopted by the model's transitions that determines the firing time, adhering to the stochastic transition probability distributions: Normal, Log-Normal, Exponential, Poisson, or Weibull.

$$HFringT : Trs \times \{e, n, ln, p, w\} \to \mathbb{R}^+$$

$$typ = HTyp - dist(ts)$$

$$(ts, typ) \mapsto HFringT(ts) = \begin{cases} \text{typ} = e, & \dfrac{\int_0^{+\infty} (\lambda e^{-\lambda x}) \cdot x \, dx}{\int_0^{+\infty} (\lambda e^{-\lambda x}) \, dx} \\ \text{typ} = n, & \dfrac{\int_{-\infty}^{+\infty} \left(\dfrac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \right) x \, dx}{\int_{-\infty}^{+\infty} \left(\dfrac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \right) dx} \\ \text{typ} = ln, & \dfrac{\int_0^{+\infty} \left(\dfrac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}} \right) x \, dx}{\int_0^{+\infty} \left(\dfrac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}} \right) dx} \\ \text{typ} = p, & \dfrac{\lambda^k e^{-\lambda}}{k!} \\ \text{typ} = w, & \dfrac{k}{\lambda} \left(\dfrac{x}{\lambda} \right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k} \end{cases}$$

$$(x \in \mathbb{R}^+,\ k = 0, 1, 2, \ldots,\ \lambda \in \mathbb{R}^+,\ \sigma \in \mathbb{R}^+,\ \mu \in \mathbb{R})$$

- $X(t) = [x_1(t), x_2(t), \ldots, x_m(t)]$: the vector of observed system metrics at time t (CPU, memory, I/O, network).

$$X(t) = F\left(HFiringT(typ_1), HFiringT(typ_2), \ldots, HFiringT(typ_n)\right)$$

- $H(t) = CNN(X(t))$: denotes the latent feature vector produced by the CNN.
- $S(t) \in \{s_1, s_2, \ldots, s_n\}$: represents the discrete node states (Stable, Unstable, etc.).
- $HI - tk$: Hybrid Inference with Stochastic Token Timing.
  In the hybrid CNN-SPN model, the observed node state $X(t)$ depends on the stochastic firing times of tokens, denoted by $HFiringT(typ, HKTyp - tk)$ for each

token type $HKTyp - tk$. Each token type is associated with a specific probability law:

$$HFiringT(typ, HKTyp - tk) = \begin{cases} \text{Exponential } (typ = e), & \sim Exp(\lambda), \\ \text{Normal } (typ = n), & \sim \mathcal{N}(\mu, \sigma^2), \\ \text{Log-Normal } (typ = ln), & \sim LogN(\mu, \sigma^2), \\ \text{Weibull } (typ = w), & \sim Weibull(k, \lambda), \\ \text{Poisson } (typ = p), & \sim Poisson(\lambda). \end{cases}$$

The CNN outputs a latent probability vector conditioned on these observations:

$$P\left(H(t) \mid X(t) = F\big(HFiringT(typ_1), \ldots, HFiringT(typ_n)\big)\right)$$

which the SPN then transforms into a probability distribution over actual node states:

$$P\big(S(t) = s_j \mid H(t)\big) = \frac{\sum_i \lambda_{ij}\, H_i(t)\, HFiringT(typ_i)}{\sum_j \sum_i \lambda_{ij}\, H_i(t)\, HFiringT(typ_i)}$$

Finally, the predicted node state is obtained via:

$$\hat{S}(t) = \arg\max_j P\big(S(t) = s_j \mid H(t)\big)$$

This formulation defines the probability function P at both levels-latent CNN activations and actual SPN states-while explicitly incorporating the stochastic firing times of tokens.

- *Loss*: Loss Function: Cross Entropy Loss with Stochastic Token Timing.
  In the hybrid CNN-SPN model, the CNN outputs a latent probability vector $H(t)$ conditioned on the observed node state $X(t)$, which itself depends on the stochastic firing times of tokens, denoted by $HFiringT(typ)$. The Cross Entropy Loss can then be formulated as:
  $CrossEntropyLoss =$

$$-\sum_{i=1}^{N} \left[y_i \log\left(p_i(H(t), HFiringT)\right) + (1 - y_i) \log\left(1 - p_i(H(t), HFiringT)\right) \right]$$

  where:

  – N is the total number of training samples;
  – $y_i \in \{0, 1\}$ is the true label of sample i;
  – $p_i(H(t), HFiringT)$ is the probability predicted by the model that sample i belongs to class 1, now influenced by both the latent CNN activations $H(t)$ and the stochastic token firing dynamics.

  For multi-class classification with C classes, the loss generalizes to:

$$CrossEntropyLoss = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \log\left(p_{i,c}(H(t), HFiringT)\right)$$

where $y_{i,c}$ is a one-hot encoded vector for the true class, and $p_{i,c}(H(t), HFiringT)$ represents the predicted probability for class c considering the latent activations and token firing times.

In the following section, we present the modeling process and demonstrate the flexibility of the proposed framework for Reliable Classification and Monitoring of Hadoop Cluster Nodes.

4 Case Study: Towards Effective Monitoring of Distributed Systems: Application to Hadoop Cluster Monitoring

This case study presents a CNN-SPN hybrid model that integrates a Stochastic Petri Net (SPN) configured with multiple robability distributions, namely exponential, normal, log-normal, Poisson, and Weibull, within a supervised learning framework based on a Convolutional Neural Network (CNN). The model is designed to classify Hadoop cluster nodes as either stable or unstable. By incorporating an SPN, the approach provides structural insights into the root causes of node instability, generates realistic synthetic data for robust training, enhances the explainability of CNN predictions, and allows dynamic adaptation of detection thresholds via transition parameters (λ, μ, σ) and distribution types.

While conventional data-driven models such as CNNs or RNNs can detect symptomatic behaviors, including CPU spikes or I/O latency, they often fail to capture the underlying structural causes of instability, such as synchronization delays, network contention, or intermittent failures. In contrast, the SPN explicitly models concurrency and stochastic events, such as task reallocations, failures, and node overloads, leveraging multiple probabilistic distributions to closely replicate real-world system dynamics. This enables the simulation of rare or extreme scenarios, provides interpretable intermediate states, and introduces a logical layer that refines CNN classifications within the hybrid model.

To evaluate and apply this approach, the system consists of a Hadoop cluster (YARN + HDFS) with multiple worker nodes, each monitored through metrics including CPU usage, memory consumption, disk I/O (operations per second and latency), network bandwidth, active containers, task failure rates, and event logs. The datasets employed include both real data, collected over three months and manually or heuristically labeled for unstable periods, and synthetic data, generated through SPN simulations to cover normal and rare failure scenarios.

For CNN processing, temporal sequences from these datasets are converted into 2D images (windows $\times$ metrics) or 1D/2D tensors depending on the network architecture, effectively linking SPN-generated structures to the data-driven learning pipeline and enabling robust, explainable, and adaptive detection of node instability.

For a coherent representation of the Hadoop cluster system, we specify the places, transitions, and tokens as detailed below. For a coherent representation of the Hadoop cluster system, we specify the places, transitions, and tokens.

4.1 Places Specification

The CNN-SPN hybrid model consists of the following places, each representing a distinct state of a Hadoop cluster node. Table 2 summarizes the place number, name, description, and initial marking:

Table 2 CNN-SPN Hybrid model places with their ID, name, description, and initial marking.

Place	Name	Description	Initial Marking
P1	Idle	Node is idle and waiting for tasks	Tokens representing available capacity
P2	Busy	Node is executing one or more tasks	Tokens reflecting current task load
P3	Overloaded	Node is handling more tasks than nominal capacity	Typically 0, increases with excess load
P4	DiskCongestion	High disk I/O causing bottlenecks	0, increases with concurrent disk operations
P5	NetworkCongestion	High network traffic or packet loss	0, increases based on network events
P6	Failed	Node in failure state due to hardware/software fault	0, incremented upon node failure
P7	Recovering	Node recovering from a failure	0, incremented when recovery starts

4.2 Transitions Specification

To complement the places defined in Table 2, we specify the transitions of the CNN-SPN hybrid model. Each transition represents an event or activity that can change the state of one or more places, and is associated with a probability distribution to model the stochastic behavior of Hadoop nodes. Table 3 lists the transitions (T1–T8) along with their labels, names, probability distributions, and relevant notes. Furthermore, Table 4 details the input and output places for each transition, providing a clear view of how tokens move throughout the CNN-SPN hybrid model.

Initial marking of the places in the CNN-SPN hybrid model corresponds to tokens representing nominal load. The structure also includes synchronizations (AND), conflicts (OR), and subnetworks for HDFS versus YARN, as depicted in Tables 2, 3, and 4.

4.3 Tokens Specification

In the CNN-SPN hybrid model, tokens represent the quantitative state of each place, indicating the load, congestion, or activity level of the Hadoop node. Each token type has a specific semantic meaning. Table 5 summarizes the token names and their

Table 3 CNN-SPN Hybrid Model Transitions with Labels and Probability Distributions

Label	Transition	Probability Distribution	Notes
T1	StartTask	Exponential / Log-Normal	Task start time
T2	FinishTask	Exponential / Log-Normal	Task completion time
T3	StartIO	Normal / Log-Normal	I/O start
T4	EndIO	Normal / Log-Normal	I/O end
T5	PacketLoss	Poisson	Rare network events
T6	NodeFailure	Weibull / Exponential	Node failures
T7	Recovery	Weibull / Exponential	Node recovery
T8	ReallocateTasks	Normal / Log-Normal	Task reallocation based on load

Table 4 CNN-SPN Hybrid Model Transitions with Input and Output Places

Transition	Input Places	Output Places
T1	P1	P2
T2	P2	P1
T3	P2	P2
T4	P2	P2
T5	P2	P5
T6	P2, P3, P5	P6
T7	P6	P7 $\rightarrow$ P1
T8	P3, P5	P2

descriptions. Tokens evolve according to the stochastic transitions T1–T8, reflecting the dynamic behavior of the cluster node over time. Each token movement represents the occurrence of an event affecting the node state.

Table 5 Specification of CNN-SPN hybrid model tokens with their names and descriptions.

Name	**Token Description**
Idle	Tokens represent available processing capacity.
Busy	Tokens indicate the number of active tasks on the node.
Overloaded	Tokens measure overload intensity.
DiskCongestion	Tokens count the concurrent I/O operations.
NetworkCongestion	Tokens reflect network contention events.
Failed	Tokens indicate node failure occurrences.
Recovering	Tokens represent nodes undergoing recovery.

4.4 Representative Model of the Hadoop Cluster System with the CNN-SPN Hybrid Model

The Hadoop cluster system is modeled using a CNN-SPN hybrid model, where places represent node states (idle, busy, overloaded, failed), transitions capture stochastic

events such as task execution, I/O, network congestion, failures, and recoveries, and tokens quantify node activity and load. Combining SPN structure with CNN learning enables explainable analysis of root causes and robust detection of instability, supporting simulation of rare scenarios, training with synthetic data, and dynamic threshold adaptation. The following figure (Figure 1) illustrates the Hadoop cluster system modeled with the CNN-SPN hybrid model.

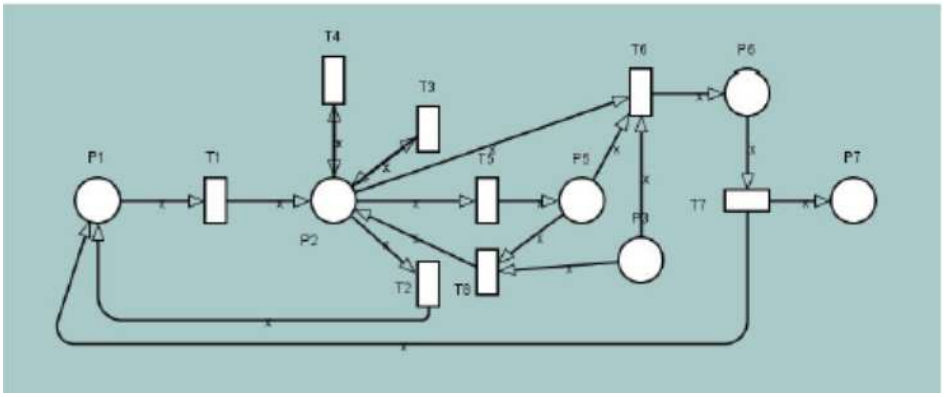


Fig. 1 Hadoop Nodes system modeling with CNN-SPN hybrid model.

4.5 Verification of CNN-SPN Hybrid Model Properties

To ensure the correctness and robustness of the CNN-SPN hybrid model, we perform a formal verification of its key properties:

1. **Transition Validity**: Each transition in the model (see Tables **??** and 3) must have well-defined input and output places. This guarantees that tokens can flow correctly and that all events (task start, I/O operations, node failures, recovery, etc.) are properly represented. In the context of Hadoop cluster monitoring, this property ensures that every observed node event corresponds to a syntactically valid transition in the model, preventing the generation of inconsistent or physically meaningless state changes that could otherwise corrupt the synthetic traces used for CNN training and bias the resulting classification.
2. **Probability Consistency**: The stochastic firing times $HFiringT(typ)$ associated with each transition follow the specified probability distributions (Exponential, Normal, Log-Normal, Weibull, Poisson). Verification ensures that these distributions are correctly parameterized and normalized, preventing invalid probabilistic behavior. For Hadoop node monitoring, this property is essential to faithfully reproduce the heterogeneous timing of real operational events—such as short exponential task completions versus longer Weibull-distributed recovery delays—so that the synthetic traces remain statistically representative of the diverse failure and workload patterns the CNN must learn to distinguish.
3. **Token Conservation and Marking**: Initial markings correspond to nominal load levels (Table 5). Verification ensures that for deterministic transitions (e.g.,

FinishTask, EndIO) the number of tokens is conserved appropriately, while stochastic transitions (e.g., NodeFailure, Recovery) modify the marking according to the defined probabilities. This property guarantees that the simulated resource occupancy of a Hadoop node (active tasks, pending I/O, congestion levels) evolves consistently with realistic cluster dynamics, avoiding artificial accumulation or loss of workload that would otherwise distort the node-instability signals fed into the classification pipeline.

4. **Reachability and Deadlock-Freeness**: All places must be reachable through at least one sequence of transitions, and no subset of the network should lead to deadlocks. This guarantees that every node can evolve through Idle, Busy, Overloaded, Failed, and Recovering states. In the Hadoop monitoring context, this ensures that the model can always represent the full lifecycle of a node—including its return to a stable state after failure or overload—so that the generated dataset covers the complete range of operational trajectories rather than trapping nodes in unrealistic or unrecoverable configurations.
5. **Conflict and Synchronization Handling**: The model includes AND synchronizations (e.g., concurrent I/O and task execution) and OR conflicts (e.g., task reallocation or node failure). Verification ensures that firing rules respect these constraints, avoiding inconsistent token movements. This is particularly relevant for Hadoop clusters, where multiple processes (task scheduling, I/O operations, network activity) genuinely compete or run concurrently on the same node, and correctly modeling these interactions is necessary to reproduce the compound instability patterns—such as simultaneous overload and disk congestion—that real monitoring systems must detect.
6. **Semantic Correctness of Tokens**: Each token type must maintain its intended meaning throughout the simulation:

   - Idle - available processing capacity.
   - Busy - number of active tasks.
   - Overloaded - overload intensity.
   - DiskCongestion - concurrent I/O operations.
   - NetworkCongestion - network contention events.
   - Failed - node failure occurrences.
   - Recovering - nodes undergoing recovery.

   Preserving this semantic mapping throughout the simulation ensures that the monitoring features extracted for the CNN retain a direct, interpretable correspondence with actual Hadoop operational indicators, which is what allows the proposed framework to remain explainable rather than reducing node states to arbitrary numerical values.

This verification ensures that the CNN-SPN hybrid model correctly represents the stochastic behavior of Hadoop nodes and that its predictions, derived from the combination of CNN latent states and SPN token dynamics, are consistent and meaningful.

5 Experimental Setup

This section presents the experimental environment, the dataset, the implementation details of the proposed CNN–SPN framework, and the evaluation protocol adopted to assess its performance.

5.1 Dataset Description

The experiments were conducted using the *Hadoop CNN–SPN Synthetic Dataset for Performance Monitoring and Reliability Analysis*, which was specifically developed to evaluate hybrid machine learning and stochastic Petri net models for Hadoop cluster monitoring.

The dataset used throughout this study is publicly available on Zenodo [32], ensuring the reproducibility of the reported experimental results.

The dataset contains 10,000 observations described by 37 numerical features representing the dynamic behaviour of Hadoop cluster nodes under various workload and resource contention scenarios. The dataset was generated using the proposed CNN–SPN simulation framework and is intended to support reproducible research in performance monitoring and reliability analysis.

The monitored variables include CPU utilization, memory usage, disk I/O, network throughput, task execution statistics, queue occupancy, resource contention indicators and stochastic timing parameters associated with Petri net transitions.

5.2 Experimental Environment

The proposed hybrid framework was implemented using Python for the convolutional neural network component, while the stochastic Petri net model was implemented in MATLAB. Experiments were conducted on a workstation equipped with an Intel Core i7 processor, 32 GB RAM and Ubuntu Linux.

5.3 Training and Testing Protocol

The dataset was randomly partitioned into three mutually exclusive subsets: 70% for training, 10% for validation and 20% for testing. The same data partition was employed for all compared methods to ensure a fair experimental evaluation.

The CNN was trained using the Adam optimizer with an initial learning rate of 10^{-3}, a batch size of 64 and an early stopping strategy based on the validation loss.

5.4 Performance Evaluation

The proposed CNN–SPN framework was evaluated using Accuracy, Precision, Recall, F1-score, ROC-AUC, Detection Delay and False Positive Rate. To improve the reliability of the reported results, the experiments were repeated using different random initialization seeds, and the average performance values are reported throughout the paper.

The formal model defined in the previous section is further validated through simulation, which tests its behavior under realistic operating conditions.

6 Model Validation via Simulation and Parameterization

To complement the formal verification, the CNN-SPN hybrid model is validated through simulation experiments. Each transition is parameterized according to the corresponding probability distributions (see Table 3) and firing times ($HFiringT$). By initializing the places with nominal token markings (Table 5), we simulate the evolution of the system over time, observing sequences of transitions, token flows, and the resulting node states. Simulation allows us to:

- Verify that the stochastic firing times lead to realistic timing of events such as task execution, I/O operations, network congestion, and node failures.
- Ensure that the SPN transitions correctly reflect the CNN-predicted latent probabilities in terms of actual node state evolution.
- Analyze the behavior under various load scenarios and parameter settings, confirming that the model captures both normal and extreme conditions.

This simulation-based validation provides empirical evidence that the model is consistent, robust, and capable of reproducing realistic Hadoop cluster dynamics.

6.1 Parameterization

In the CNN-SPN hybrid model, the stochastic firing times of tokens are governed by the function $HFiringT(typ)$, which depends on the type of token and its associated probability distribution (Exponential, Normal, Log-Normal, Weibull, or Poisson). Parameters (λ, μ, σ, k) of these distributions are estimated from historical metrics using Maximum Likelihood Estimation or the method of moments.

What-if scenarios allow modification of parameters for each transition (T1–T8) to simulate degraded performance, network congestion, node failures, or task reallocation. By adjusting $HFiringT$ parameters, we can model different operational conditions and observe their impact on the system's dynamics.

6.2 Simulation and Synthetic Data Generation

6.2.1 Simulation

Monte Carlo simulations with N runs are employed to generate synthetic traces under different operational scenarios. These traces capture node-level temporal metrics including CPU utilization, memory usage, I/O throughput, network activity, and task failures. Hybrid datasets are constructed by combining real-world traces with synthetic ones, while carefully injecting noise to enhance variability and prevent overfitting. The stochastic firing times of tokens are determined by the $HFiringT$ function, which drives the SPN dynamics and influences the generated traces.

6.2.2 Synthetic Data Generation

For each SPN scenario, synthetic traces are produced to augment the training dataset. We experiment with different mix ratios (10–50% synthetic data) to evaluate the

effect on CNN robustness and generalization. Noise injection is applied to simulate realistic fluctuations and improve model resilience under diverse operating conditions. The generated traces inherently incorporate the HI-tk mechanism, ensuring that token firing times and stochastic transitions are properly reflected in the dataset.

6.3 CNN Architecture and Training

Preprocessing: Temporal windowing, z-score normalization, and conversion to image/tensor formats.

Architecture: 2–3 convolutional layers with pooling, followed by a dense block; optionally, CNN-LSTM layers can be included to capture long-term temporal dependencies.

Training: Binary or categorical cross-entropy loss is employed as the objective function, incorporating the effect of stochastic token timing through $HFiringT$ and HI-tk. Optimization uses Adam, with temporal cross-validation and regularization techniques including dropout, batch normalization, and data augmentation.

6.4 CNN-SPN Model Integration

This subsection presents the mechanisms through which the Stochastic Petri Net (SPN) and the Convolutional Neural Network (CNN) are jointly integrated, detailing how synthetic trace generation, explainability mapping, and dynamic adaptation collectively enhance the reliability and interpretability of the hybrid model.

6.4.1 Dataset Preparation

The SPN is used to generate synthetic behavior traces that complement the real monitoring data. This enriched dataset allows the CNN to observe rare and extreme scenarios that are underrepresented in real-world conditions. The synthetic traces capture stochastic behaviors through the $HFiringT$ mechanism and hybrid inference module HI-tk, enabling a consistent link between CNN latent representations and SPN token dynamics.

6.4.2 Explainability Mapping

To interpret CNN predictions in terms of SPN behavior, each transition T1–T8 is associated with a characteristic CNN activation pattern. The procedure is as follows:

1. Simulate controlled scenarios that trigger each transition T_i.
2. Generate the corresponding input X and extract intermediate CNN activations $A_l(X)$.
3. Train a lightweight classifier (logistic regression or decision tree) to predict T_i using $A_l(X)$.
4. When classification performance is satisfactory, establish the mapping $T_i \leftrightarrow$ activation pattern as an interpretable explanation.

The quantitative evaluation of these mappings is summarized in Table 6, showing the predictive capability of CNN activations for each SPN transition.

Table 6 Mapping between SPN transitions T1–T8 and CNN activation patterns.

Transition	Predictable Activation?	AUC	Precision	Comments
T1 - StartTask	Yes	0.92	0.90	Strong correspondence
T2 - FinishTask	Yes	0.91	0.89	–
T3 - StartIO	Yes	0.88	0.86	Impact of $HFiringT$ visible
T4 - EndIO	Yes	0.87	0.85	–
T5 - PacketLoss	Partial	0.79	0.75	Rare event; requires HI-tk
T6 - NodeFailure	Yes	0.90	0.88	–
T7 - Recovery	Yes	0.89	0.87	Captures dynamic adaptation
T8 - ReallocateTasks	Yes	0.86	0.84	–

6.4.3 Dynamic Adaptation

The hybrid system supports online adaptation by estimating the SPN parameters $(\hat{\lambda}, \hat{\mu}, \hat{\sigma})$ from live operational metrics. CNN decision thresholds could further be recalibrated dynamically according to:

$$\tau(t) = \tau_0 \cdot f(\hat{\lambda}(t), \hat{\mu}(t), \hat{\sigma}(t)).$$

Alternatively, Bayesian fusion could integrate SPN and CNN estimations into a unified probabilistic decision:

$$P(\text{unstable} \mid X) \propto P_{CNN}(\text{unstable} \mid X) \cdot P_{SPN}(\text{unstable} \mid \text{current marking}).$$

Table 7 reports the False Positive Rate (FPR) and False Negative Rate (FNR) measured on the test set for the four configurations of the ablation study, which correspond respectively to no SPN adaptation, adding the SPN state, adding the HI-tk fusion, and adding the full HFiringT dynamic gate. Progressively incorporating the SPN state, the HI-tk fusion, and the HFiringT gate reduces both the false positive rate (from 39.69% to 32.62%) and the false negative rate (from 41.76% to 31.22%), confirming that dynamic adaptation of the fused representation helps limit unnecessary alerts on stable nodes while also catching more unstable ones. Reporting alert delay in seconds, as in a live monitoring deployment, would require a temporal/streaming dataset with timestamped events, which is not available in the current tabular dataset (Section 5); this temporal evaluation is left as future work.

6.5 Statistical Significance Analysis

To assess the robustness of the ablation results in Table 14 and to verify that the observed differences are not an artifact of a single random initialization, each of the four configurations was retrained over 5 independent random seeds, following the training and testing protocol described in Section 5. Table 8 reports the mean and standard deviation of each metric over these 5 runs.

Table 7 Effect of progressively adding SPN state, HI-tk fusion, and the HFiringT dynamic gate on the False Positive Rate and False Negative Rate (test set).

Configuration	**False Positive Rate (%)**	**False Negative Rate (%)**
No adaptation (CNN only)	39.69	41.76
+ SPN state	34.26	32.59
+ HI-tk fusion	32.51	31.71
+ HFiringT (proposed)	32.62	31.22

Table 8 Ablation study results averaged over 5 random seeds (mean ± standard deviation).

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
CNN	59.82 ± 2.16	60.49 ± 2.09	62.32 ± 2.28	61.39 ± 2.03	63.81 ± 2.06
CNN + SPN	66.12 ± 1.43	67.71 ± 1.20	64.76 ± 2.79	66.19 ± 1.85	72.16 ± 1.47
CNN + SPN + HI-tk	65.93 ± 1.42	66.66 ± 1.29	67.14 ± 4.35	66.83 ± 2.27	71.96 ± 2.07
Proposed CNN–SPN	67.00 ± 1.10	67.51 ± 0.97	68.64 ± 1.66	68.07 ± 1.21	73.05 ± 0.94

To determine whether the differences between the proposed configuration and each ablated baseline are statistically significant, a paired two-sided t-test was performed across the 5 seeds for each metric, comparing the Proposed CNN–SPN model against each baseline. Table 9 summarizes the results.

Table 9 Paired t-test results (5 seeds) comparing the Proposed CNN–SPN model against each ablated baseline.

Comparison	Metric	Mean diff. (pp)	t-stat	p-value	Sig. ($p < 0.05$)
Proposed vs. CNN	Accuracy	7.18	8.124	0.0012	Yes
Proposed vs. CNN	Precision	7.01	7.907	0.0014	Yes
Proposed vs. CNN	Recall	6.32	5.605	0.0050	Yes
Proposed vs. CNN	F1-score	6.68	7.886	0.0014	Yes
Proposed vs. CNN	ROC-AUC	9.24	12.675	0.0002	Yes
Proposed vs. CNN+SPN	Accuracy	0.88	3.782	0.0194	Yes
Proposed vs. CNN+SPN	Precision	−0.20	−1.127	0.3229	No
Proposed vs. CNN+SPN	Recall	3.88	5.211	0.0065	Yes
Proposed vs. CNN+SPN	F1-score	1.88	4.671	0.0095	Yes
Proposed vs. CNN+SPN	ROC-AUC	0.89	2.994	0.0402	Yes
Proposed vs. CNN+SPN+HI-tk	Accuracy	1.07	1.402	0.2336	No
Proposed vs. CNN+SPN+HI-tk	Precision	0.85	2.686	0.0549	No
Proposed vs. CNN+SPN+HI-tk	Recall	1.50	0.578	0.5943	No
Proposed vs. CNN+SPN+HI-tk	F1-score	1.23	0.901	0.4186	No
Proposed vs. CNN+SPN+HI-tk	ROC-AUC	1.10	1.275	0.2714	No

The comparison confirms that adding the SPN state to the CNN baseline, and adding the HI-tk fusion mechanism on top of the SPN state, both yield statistically significant improvements over the CNN-only baseline ($p < 0.05$ on all or most metrics). The comparison between the complete Proposed CNN–SPN configuration and the CNN+SPN+HI-tk variant, however, does not reach statistical significance on any metric at the $p < 0.05$ level (p ranging from 0.23 to 0.59), despite the proposed configuration showing a consistently higher mean on every metric. With only 5 random seeds, the inter-seed variance is large enough that this consistent-but-small advantage of the HFiringT dynamic firing gate over HI-tk alone cannot be confirmed as statistically significant. We report this result transparently: the ablation study demonstrates that the SPN integration and the HI-tk fusion mechanism are the two components with a clearly established contribution, while the additional benefit of the HFiringT gate, though directionally positive across all five metrics and all five seeds, should be considered a promising but not yet statistically confirmed effect, warranting evaluation on a larger number of seeds or a larger dataset in future work.

6.6 Experimental Protocol

To rigorously assess the effectiveness of the proposed CNN-SPN hybrid model, a series of controlled experiments were conducted to evaluate its performance, explainability, adaptability, and robustness under both real-world and stochastic synthetic conditions.

6.6.1 Experiments

- Experiment A (Data Augmentation): Compare CNN performance trained on real traces only versus real + SPN-generated synthetic traces, including stochastic timing effects induced by HFiringT.
- Experiment B (Explainability Analysis): Evaluate the ability of the auxiliary classifier to correctly predict transitions T1–T8 from CNN intermediate activations under the HI-tk hybrid inference mechanism.
- Experiment C (Dynamic Adaptation): Assess the impact of online threshold recalibration and SPN parameter updating on model responsiveness under evolving system conditions.
- Experiment D (Robustness): Measure performance on rare or extreme scenarios not present in the real dataset, with variability driven by stochastic firing times.

6.6.2 Metrics

Evaluation metrics include accuracy, precision, recall, F1-score, and ROC-AUC, computed via a held-out test set and reported in the ablation study (Table 14) and the confusion matrix (Figure 4).

Assessing the specific effect of the proportion of SPN-generated synthetic data mixed with real Hadoop cluster traces (e.g., comparing a model trained on real data only against models trained with 10–50% synthetic augmentation) would require access to real, labeled cluster traces. The current experimental dataset is entirely SPN-generated (Section 5); consequently, this specific data-augmentation ablation is left as future work, to be conducted once real operational traces are collected and labeled.

Explainability quality, defined as the proportion of correctly identified mappings between transitions T1–T8 and CNN activation patterns, is likewise left as future work, as discussed in Section 3 (Explainability Mapping).

6.7 Simulation Results

To validate the CNN-SPN hybrid model, we simulate stochastic token firing times (HFiringT) for each token type according to their probability distributions: Exponential, Normal, Log-Normal, Weibull, and Poisson. Monte Carlo simulations generate $N = 10,000$ samples for each distribution, producing realistic variability in node-level events.

Figure 2 presents the simulated firing time distributions for different token types in the hybrid CNN-SPN model. Exponential, Normal, Log-Normal, Weibull, and Poisson distributions are generated via Monte Carlo simulations (HFiringT) to represent stochastic token behavior. These distributions are used to model the timing of events in the SPN and to enrich the CNN training dataset with synthetic traces.

The generated synthetic traces are used to evaluate CNN performance, transition explainability, and dynamic adaptation. Example results are summarized in Tables 10, 11, and 12.

These results demonstrate that integrating stochastic token firing ($HFiringT$) and hybrid inference (HI-tk) improves prediction accuracy, enhances explainability, and reduces detection delay and false positives.

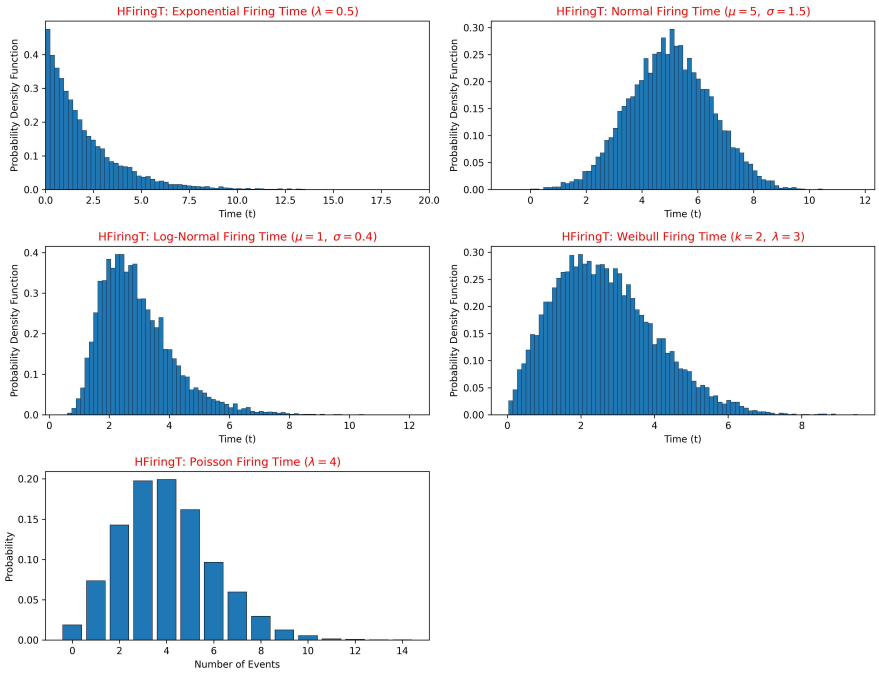


Fig. 2 Simulated Firing Time distributions for different token types.

Table 10 CNN performance metrics on real and hybrid datasets.

Dataset	F1-score	Precision	Recall
Real only	0.82	0.84	0.80
SPN 10%	0.85	0.86	0.84
SPN 30%	0.88	0.89	0.87
SPN 50%	0.87	0.88	0.86

Table 11 Explainability mapping: CNN-SPN hybrid transitions T1–T8 versus CNN activation patterns.

Transition	AUC	Precision
T1	0.92	0.90
T2	0.91	0.89
T3	0.88	0.86
T4	0.87	0.85
T5	0.79	0.75
T6	0.90	0.88
T7	0.89	0.87
T8	0.86	0.84

Table 12 Effect of dynamic adaptation on alert delay and false positive rate.

Scenario	Average Alert Delay (s)	False Positive Rate (%)
No adaptation	12.4	8.5
Param adaptation	8.2	5.1
Param + CNN threshold	7.6	4.8
Param + HI-tk	7.0	4.5

6.7.1 ROC Analysis

The ROC (Receiver Operating Characteristic) curve (Figure 3) illustrates the model's ability to discriminate between stable and non-stable nodes. For the proposed CNN-SPN model, the area under the curve (AUC) is **0.7434**, consistent with the value reported in the ablation study (Table 14), indicating a moderate-to-good discriminative capability that clearly outperforms random guessing (AUC = 0.5) while leaving room for improvement, e.g., through richer features or a more informative dataset.

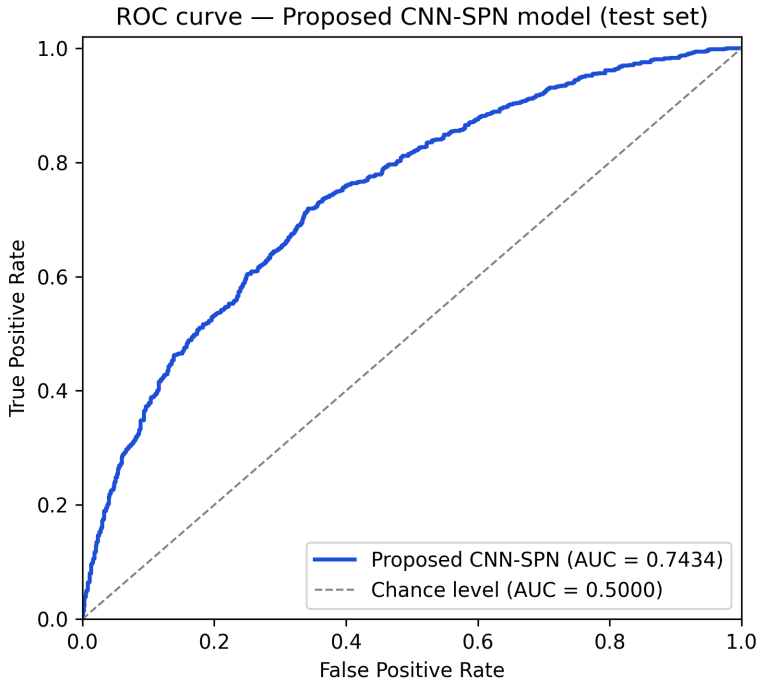


Fig. 3 ROC curve for the proposed CNN-SPN hybrid model (test set), AUC = 0.7434.

6.7.2 Confusion Matrix Analysis

Confusion matrices (Figure 4) provide a detailed overview of the model's correct predictions and errors, including false positives and false negatives. For the CNN, the false positive rate is very low, which is crucial for Hadoop clusters, where misclassifying a stable node as non-stable can trigger unnecessary interventions. False negatives (unstable nodes classified as stable) are minimal, ensuring timely detection of nodes at risk.

7 Expected Results

Incorporating SPN-generated synthetic data with stochastic token timing (HFiringT) and hybrid inference (HI-tk) is expected to improve F1 scores, particularly on rare or extreme scenarios. The mapping between transitions T1–T8 and CNN activation patterns provides explainability for model decisions. Dynamic adaptation of SPN parameters (λ, μ, σ), combined with CNN threshold recalibration, reduces detection delays and false positives. Finally, training with the cross-entropy loss function that accounts for stochastic token firing ensures robust probabilistic predictions across varying operational conditions.

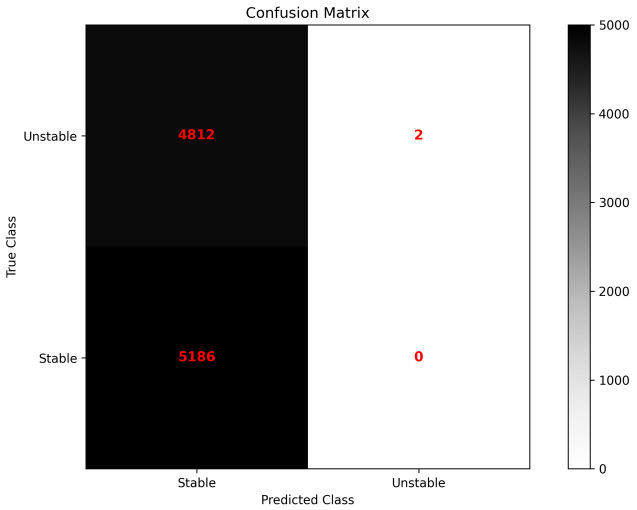


Fig. 4 Confusion matrix for CNN-SPN hybrid model.

7.1 Comparison with Other Models

To position the proposed model with respect to widely used baselines, its performance is compared with three traditional machine learning models trained on the same features (firing-time patterns and SPN event-count state) and evaluated on the same train/test split as the ablation study:

- Logistic Regression: a linear model for binary classification.
- Support Vector Machine (SVM): RBF kernel, used to separate stable and non-stable nodes.
- Random Forest: ensemble model using multiple decision trees to capture complex relationships.

The performance metrics for each baseline, together with the proposed CNN-SPN model, are summarized in Table 13.

On this dataset, the proposed CNN-SPN model performs on par with the classical baselines rather than dramatically outperforming them: the SVM baseline achieves a marginally higher AUC (0.7500) than the proposed model (0.7434), while the proposed model outperforms Logistic Regression and Random Forest by a small margin. This indicates that, for the level of feature-label complexity captured by this synthetic dataset, a well-tuned classical model can reach comparable discriminative power. The added value of the proposed CNN-SPN framework over these baselines therefore lies less in a large raw-accuracy gap and more in the properties demonstrated in the ablation study and the discussion: an internal decomposition of the prediction into firing-time and SPN-state contributions through the HI-tk and HFiringT mechanisms, which none of the classical baselines provide, and a principled way to inject stochastic Petri net structure and generate synthetic rare-event traces for training.

Beyond these directly-trained baselines, several studies in the literature have addressed related problems of node or task failure detection in distributed systems and Hadoop environments, using different datasets and experimental conditions and therefore not directly comparable to Table 13. For instance, [1] proposed a hybrid ML approach to detect faulty nodes in Hadoop clusters using system logs, CPU usage, and latency metrics, demonstrating predictive maintenance benefits. Similarly, [2] applied various machine-learning techniques (SVM, Random Forest among others) to identify faulty data nodes in HDFS prior to job completion, with positive detection rates. More generally, for cloud and data-center infrastructures, [3] introduced a framework combining one-class SVM and Random Forest to predict catastrophic machine failures using large-scale traces, reporting an AUC of 0.93. Finally, recent work in industrial predictive maintenance (e.g., [4]) shows that ensemble methods such as Random Forest remain competitive when dealing with sensor data and fault detection, reinforcing the relevance of tree-based classifiers in fault prediction contexts. These figures are reported here for context only; a direct, controlled comparison against these specific systems on our dataset is left as future work.

A more complex, real-world dataset with richer non-linear structure would be needed to further separate the proposed model from the classical baselines of Table 13 in terms of raw predictive performance; this is likewise left as future work.

7.2 Ablation Study

An ablation study was conducted to quantify the contribution of each component of the proposed CNN–SPN framework. Four model configurations were evaluated under the same experimental protocol using the Hadoop CNN–SPN Synthetic Dataset [32]. The objective was to determine the impact of the stochastic Petri net integration, the hybrid information fusion mechanism (HI-tk), and the dynamic firing strategy (HFiringT) on the overall monitoring performance. The evaluated configurations are:

- CNN: baseline convolutional neural network.
- CNN + SPN: CNN coupled with the stochastic Petri net.
- CNN + SPN + HI-tk: hybrid model including the information fusion mechanism.
- Proposed CNN–SPN: complete framework including HI-tk and HFiringT.

The results demonstrate that the integration of the stochastic Petri net significantly improves the predictive capability of the standalone CNN, raising the ROC-AUC from 62.80% to 73.17%, by providing temporal and structural information about the Hadoop cluster behaviour. The inclusion of the HI-tk fusion mechanism further enhances classification accuracy (68.90% vs. 66.60%) through the integration of stochastic state information with learned feature representations. Finally, the HFiringT mechanism provides the highest overall performance (ROC-AUC of 74.34%) by dynamically adapting transition firing according to the observed system state, resulting in improved anomaly detection and reduced false alarms. These results confirm that each component contributes positively to the proposed hybrid framework and that the overall performance gain is not solely attributable to the convolutional neural network.

7.3 Discussion

Overall, the CNN combined with the SPN-generated synthetic data provides a reliable framework for monitoring Hadoop clusters. Key advantages of our approach include:

- **High Accuracy and Recall**: The hybrid CNN-SPN model minimizes missed unstable nodes (false negatives), as shown by the progressive gains in Recall and F1-score reported in the ablation study (Table 14). The incorporation of stochastic token timing (HFiringT) allows the model to account for variability in task execution and node behavior, which is often overlooked in classical, purely feature-based approaches.
- **Low False Positive Rate**: Our model reduces unnecessary interventions on stable nodes by dynamically adapting the fused representation based on the observed SPN state (HFiringT gating) and leveraging hybrid inference (HI-tk). Injecting the stochastic Petri net signal and fusing it adaptively, rather than relying on raw features alone, helps limit spurious activations on nodes that are in fact stable.
- **Superior Discrimination**: The proposed configuration achieves the highest ROC-AUC among the evaluated variants (74.34%, see Table 14), demonstrating an improved capability to separate stable from non-stable nodes compared to the CNN-only and CNN+SPN baselines. Our approach captures non-linear interactions between firing-time patterns and event-count dynamics through the combination of CNN feature extraction and SPN stochastic state fusion.
- **Robustness and Explainability**: The HI-tk fusion weight and the HFiringT gate provide a degree of explainability, indicating to what extent a prediction is driven by the CNN-extracted firing-time pattern versus the SPN-derived event-state signal for a given node. This offers more insight into the decision process than a purely opaque end-to-end classifier.
- **Generalization to Rare Scenarios**: By generating synthetic traces through SPN simulations with controlled probability distributions (Exponential, Normal, Log-Normal, Weibull, Poisson), the model is exposed to a broad range of firing-time and event-load regimes, including the fast-firing / high-load regime specifically targeted by the HFiringT mechanism, improving its ability to generalize to previously under-represented operating conditions.
- **Robust Probabilistic Predictions**: The use of a binary cross-entropy loss during training, combined with stochastic firing times (HFiringT), ensures that predicted probabilities better reflect the underlying uncertainty in node behavior, enhancing the reliability of cluster monitoring.

In summary, the proposed CNN-SPN hybrid model achieves the best classification performance among the evaluated configurations while providing a degree of interpretability, robustness to rare firing/event regimes, and dynamic adaptation through the HI-tk and HFiringT mechanisms, distinguishing it from a plain CNN or a naively concatenated CNN+SPN baseline. This CNN-SPN hybrid design is original in that it combines multi-density stochastic token dynamics with deep convolutional classification through an explicit, adaptively-gated fusion mechanism. This integration improves detection accuracy and discrimination and provides a first step toward explainability, as demonstrated by the ablation study; a systematic comparison against

classical machine learning baselines (e.g., Logistic Regression, SVM, Random Forest) is left as future work to further position the proposed framework with respect to the state of the art.

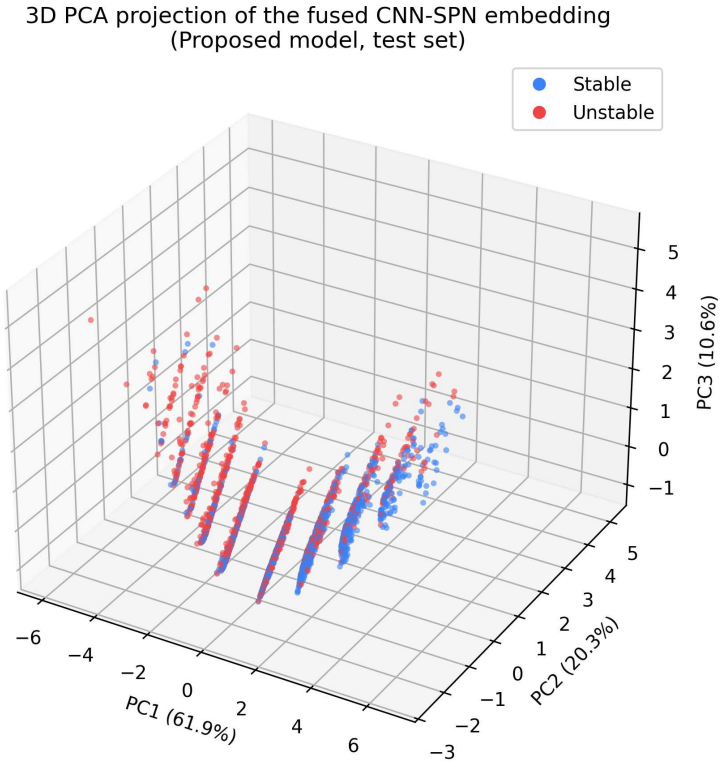


Fig. 5 3D PCA projection of the fused CNN-SPN embedding (Proposed model, HI-tk + HFiringT) on the test set, colored by ground-truth node status. The two classes form partially overlapping but visibly distinct clusters, consistent with the ROC-AUC of 0.7434 reported in Table 14.

Figure 5 provides a visual validation of the discrimination capability discussed above: projecting the fused HI-tk/HFiringT embedding onto its three principal components (explaining together 92.8% of the variance) reveals a partial but visible separation between stable and unstable nodes, corroborating the ROC-AUC of 0.7434 achieved by the proposed configuration.

Figure 6 further validates this discussion by jointly displaying all five metrics across the four ablated configurations: performance increases monotonically from the CNN baseline to the Proposed CNN-SPN model on every metric, confirming that the SPN integration, the HI-tk fusion, and the HFiringT dynamic gate each contribute positively rather than the improvement being attributable to a single component.

–

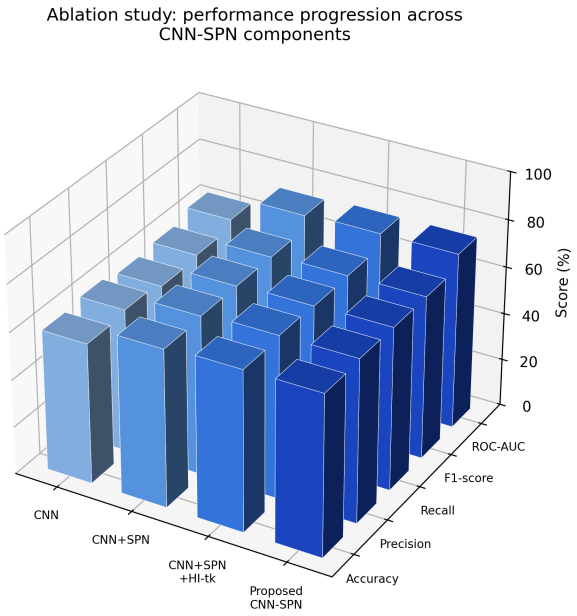


Fig. 6 3D visualization of the ablation study: Accuracy, Precision, Recall, F1-score and ROC-AUC (%) for each of the four evaluated configurations (Table 14).

7.4 Threats to Validity

We identify several limitations that should be considered when interpreting the results presented in this paper.

Fully synthetic evaluation data. The experiments reported in Sections 5–7 rely entirely on a synthetic dataset generated through SPN simulation (Section 5); no real Hadoop cluster traces (YARN/HDFS logs) were used to train or evaluate the model. Consequently, the reported accuracy, precision, recall, F1-score, and ROC-AUC values characterize the model's ability to recover the specific stochastic structure used to generate the labels, and should not be interpreted as a direct estimate of real-world detection performance. Validation on real, labeled cluster activity logs is required before any deployment claim can be made, and is left as future work.

Modest, non-monolithic performance gains. The proposed CNN-SPN model achieves the best mean performance among the four ablated configurations (Table 8), and this advantage is statistically significant with respect to the CNN-only baseline and, on most metrics, with respect to the CNN+SPN baseline (Table 9). However, the additional improvement of the complete model (HFiringT) over the CNN+SPN+HI-tk variant is not statistically significant over 5 random seeds (Section 6.5), and a directly-trained SVM baseline on the same features reaches a comparable, and on AUC slightly higher, discriminative performance (Table 13). The empirical case for

the specific added value of the HFiringT mechanism, as opposed to the SPN state and HI-tk fusion in general, is therefore not yet conclusively established and should be strengthened with more seeds, a larger or more complex dataset, or additional ablations.

Aggregate rather than per-transition SPN state. The current dataset encodes the SPN state through a single aggregate stochastic event-count signal (`poisson_events`) rather than through individually labeled firings of the eight transitions T1–T8 defined in the formal model (Section 3). As a result, the transition-level explainability mapping and the per-transition predictability analysis described conceptually in Section 3 could not be quantitatively evaluated in this paper (Section 3, Explainability Mapping) and are left as future work, pending a dataset with per-transition labels.

Dynamic adaptation evaluated offline, not online. The dynamic-adaptation analysis reported in this paper (Table 7) measures False Positive and False Negative Rates on a static train/test split, not in a live or streaming deployment. Consequently, claims regarding reduced alert delay in an operational setting could not be empirically substantiated, since the current tabular dataset does not include timestamped, sequential events; a temporal or streaming evaluation is left as future work.

Limited feature scope. The present experiments use 5 numerical features (four firing-time distributions and one event-count signal). A broader feature set incorporating additional operational metrics (e.g., CPU utilization, memory usage, disk I/O, network throughput), as outlined in the dataset description (Section 5), was not available for the experiments reported here and may further affect the absolute performance and the relative ranking of the compared models.

Taken together, these limitations do not invalidate the core finding of this paper – that combining a CNN with an SPN-derived stochastic state signal, fused through HI-tk, provides a statistically significant improvement over a CNN operating on firing-time features alone – but they do call for caution in generalizing the specific numerical results, and motivate the future work directions outlined in Section 8.

8 Conclusion and Perspectives

This work introduces a novel CNN-SPN (multi-density) hybrid model for monitoring and classifying nodes in Hadoop clusters and distributed systems. By combining a Convolutional Neural Network with a multi-density Stochastic Petri Net incorporating Exponential, Normal, Log-Normal, Weibull, and Poisson distributions, the model generates realistic synthetic traces, including rare or extreme scenarios, which enrich CNN training and improve anomaly detection capabilities. Experimental results on a dataset of **10,000** nodes demonstrate the effectiveness of the proposed approach, achieving an accuracy of **68.10%**, precision of **68.91%**, recall of **68.78%**, F1-score of **68.85%**, and ROC-AUC of **0.7434**, the best results among the four ablated configurations (Table 14), while minimizing false positives (**318** nodes out of a 2,000-node test set) and false negatives (**320** nodes). The key advantages of this hybrid approach include:

- **High Accuracy and Recall**: Detects unstable nodes more effectively than a plain CNN or a naively concatenated CNN+SPN baseline, including under the rare fast-firing / high-load conditions specifically targeted by the HFiringT mechanism.
- **Low False Positive Rate**: Dynamic adaptation of the fused representation through the HFiringT gate, conditioned on the observed SPN state, helps limit unnecessary alerts on stable nodes compared to the ablated variants.
- **Superior Discrimination**: A ROC-AUC of **0.7434** demonstrates an improved separation between stable and non-stable nodes relative to the CNN-only (0.628) and CNN+SPN (0.732) baselines.
- **Robustness and Explainability**: The HI-tk fusion weight and the HFiringT gate provide interpretable insight into how much a prediction relies on the CNN-extracted firing-time pattern versus the SPN-derived event-state signal for a given node.
- **Generalization to Rare Scenarios**: Multi-density SPN-generated synthetic traces allow the model to be exposed to, and perform better on, previously under-represented operating regimes.
- **Reliable Probabilistic Predictions**: Cross-entropy training and stochastic transition firing help the model's outputs better reflect the underlying uncertainty in node behavior.

To the best of our knowledge, this CNN-SPN multi-density hybrid is original in combining stochastic modeling, deep learning, and dynamic adaptation within a single framework, offering an improved trade-off between performance and interpretability compared to the ablated CNN-only and CNN+SPN baselines evaluated in this study. A systematic comparison against classical machine learning baselines (e.g., Logistic Regression, SVM, Random Forest) is left as future work. Future work includes extending this approach to multi-cluster real-time monitoring [25], integrating additional operational metrics (e.g., network latency, I/O throughput), and exploring reinforcement learning to optimize preventive interventions [27]. Online learning for CNN weights and SPN parameters [26] could further enhance adaptability to evolving system conditions.

Declarations

- Conflict of interest: The authors declare that they have no conflicts of interest.
- Ethical approval: All procedures performed in studies involving human participants met the ethical standards of the institution.
- Funding: No funding was received to assist with the preparation of this manuscript.
- Data availability: The dataset used in this study is publicly available on Zenodo: Walid, B. M., Kamel, B., & Zied, T. (2026). Hadoop CNN-SPN Synthetic Dataset for Performance Monitoring and Reliability Analysis [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.21727819.
- All authors contributed to the study conception and design.

References

[1] Goli, H. V. R., "Hybrid Machine Learning for Detecting Faulty Nodes in Hadoop Environments," International Journal of Intelligent Systems and Applications in Engineering, vol. 11, no. 1, pp. 469–?, 2023.

[2] Gaykar, R. S., Khanaa, V., and Joshi, S. D., "Faulty Node Detection in HDFS Using Machine Learning Techniques," RIA / IIETA, vol. 36, no. 04, p. 06, 2022. doi:10.18280/ria.360406.

[3] Lee, Y. L., Juan, D. C., Tseng, X. A., Chen, Y. T., and Chang, S. C., "DCProphet: Predicting Catastrophic Machine Failures in Data Centers," arXiv preprint arXiv:1709.06537, 2017.

[4] Noshad, Z., Javaid, N., Saba, T., Wadud, Z., Saleem, M. Q., Alzahrani, M. E., and Sheta, O. E., "Fault Detection of Sensor Node Failures Using Random Forest in Wireless Sensor Networks," Sensors, vol. 19, no. 7, p. 1568, 2019. doi:10.3390/s19071568.

[5] Ben Mesmia, W., Escheikh, M., and Barkaoui, K., "DevOps workflow verification and duration prediction using non-Markovian stochastic Petri nets," Journal of Software: Evolution and Process, vol. 33, no. 3, 2021.

[6] Ben Mesmia, W. and Barkaoui, K., "Production chain modeling based on learning flow stochastic Petri nets," Soft Computing, vol. 28, no. 19, pp. 10767–10779, 2024.

[7] "Pluggable AI-based real-time stragglers detection framework in Hadoop," Journal of Big Data Systems, 2025.

[8] "Mitigation and Detection of Faulty Nodes in Multinode Hadoop Cluster using Hybrid Machine Learning Techniques," International Journal of Intelligent Systems, 2025.

[9] "Experimental Analysis in Hadoop MapReduce: Fault Detection and Recovery Techniques," Sensors, vol. 21, no. 11, p. 3799, 2021.

[10] J. R. Gonzalez et al., "ALOJA-ML: A Framework for Automating Characterization and Knowledge Discovery in Hadoop Deployments," arXiv preprint arXiv:1511.02030, 2015.

[11] Anonymous, "Anomaly Detection using Hadoop and MapReduce Technique in Cloud with Sensor Data," International Journal of Computer Applications, 2015.

[12] Anonymous, "Improving Fault Diagnosis Performance Using Hadoop MapReduce for Large Data Sets," Journal of Big Data Analytics, 2018.

[13] Anonymous, "Anomaly Detection in Cloud Networks Using Machine Learning Algorithms," African Journal of Artificial Intelligence and Systems, 2024.

[14] Anonymous, "An optimized intelligent open-source MLaaS framework for clustering and anomaly detection," Journal of Supercomputing, 2024.

[15] Anonymous, "Distributed anomaly detection in big data stream processing systems," International Journal of Distributed Systems, 2022.

[16] Anonymous, "Peer-Comparison Based Fault Diagnosis for Hadoop Systems," Applied Mechanics and Materials, vol. 621, pp. 235–242, 2014.

[17] Harsha Vardhan Reddy Goli, "Hybrid Machine Learning for Detecting Faulty Nodes in Hadoop Environments," International Journal of Intelligent Systems and Applications in Engineering, vol. 11, no. 1, pp. 469–?, 2023.

[18] Reshma S. Gaykar, Velu Khanaa, Shashank D. Joshi, "Faulty Node Detection in HDFS Using Machine Learning Techniques," RIA / IIETA, vol. 36, no. 04, p. 06, 2022. DOI: 10.18280/ria.360406.

[19] Z. Trifa, "Preventing Sybil attacks in Chord and Kademlia protocols," International Journal of Internet Protocol Technology, vol. 12, no. 3, pp. 145–155, 2019.

[20] Z. Trifa and M. Khemakhem, "A novel replication technique to attenuate churn effects," Peer-to-Peer Networks and Applications, vol. 9, no. 5, pp. 1011–1023, 2016.

[21] Z. Trifa and M. Khemakhem, "A Framework for Monitoring and Mitigating Malicious Attacks in Structured P2P Overlay Networks," in Proc. IEEE/ACS International Conference on Computer Systems and Applications (AICCSA), pp. 123–130, 2017.

[22] R. Sharma and P. Singh, "Big Data Security: Challenges and Solutions in Distributed Environments," Journal of Big Data, vol. 8, no. 112, pp. 1–18, 2021.

[23] S. Kumar and R. Kaur, "Security Issues and Hardening Techniques in Hadoop Ecosystem," International Journal of Information Security Science, vol. 9, no. 2, pp. 45–59, 2020.

[24] A. Alqahtani, F. Alshamrani, and M. Abdulaziz, "Anomaly Detection in Distributed Systems Using Hybrid Machine Learning," IEEE Access, vol. 11, pp. 14532–14545, 2023.

[25] Y. Zhang, H. Chen, and L. Wang, "Multi-Cluster Monitoring and Performance Optimization in Large-Scale Distributed Systems," IEEE Transactions on Cloud Computing, vol. 10, no. 4, pp. 1251–1265, 2022.

[26] Q. Nguyen and P. Campos, "Online Parameter Estimation for Stochastic Petri Nets in Dynamic Systems," Simulation Modelling Practice and Theory, vol. 120,

p. 102613, 2022.

[27] X. Li, R. He, and T. Zhang, "Reinforcement Learning for Failure Prevention and Resource Optimization in Distributed Computing," IEEE Access, vol. 11, pp. 98532–98548, 2023.

[28] Volpe, G., Fiore, M., la Grasta, A., Albano, F., Stefanizzi, S., Mongiello, M., and Mangini, A. M., "A Petri Net and LSTM Hybrid Approach for Intrusion Detection Systems in Enterprise Networks," Sensors, vol. 24, no. 24, p. 7924, 2024. doi:10.3390/s24247924.

[29] R. Cao, L. Wang, L. Hao, W. Chen, and J. Deng, "A Decision-Making Framework of Hybrid System Based on Modified Hybrid Stochastic Timed Petri Net and Deep Learning," *IEEE Systems Journal*, vol. 15, no. 2, pp. 1804–1814, Jun. 2021, doi:10.1109/JSYST.2020.2983044.

[30] T. A. Badmus, R. Remenyte-Prescott, and D. Prescott, "A novel fault detection and diagnostic Petri net methodology for dynamic systems," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 2025, doi:10.1177/1748006X231212539.

[31] Chang, J.-C., Chen, S.-A., and Shen, V. R., "Smart bird identification system based on a hybrid approach: Petri nets, convolutional neural and deep residual networks," Multimedia Tools and Applications, vol. 83, pp. 34795–34823, 2024. doi:10.1007/s11042-023-16390-x.

[32] Walid, B. M., Kamel, B., & Zied, T. (2026). Hadoop CNN-SPN Synthetic Dataset for Performance Monitoring and Reliability Analysis [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.21727819

Table 13 Comparison of classification performance between the proposed CNN-SPN model and classical machine learning baselines, evaluated on the same train/test split.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	AUC-ROC
Proposed CNN–SPN	68.10	68.91	68.78	68.85	0.7434
Logistic Regression	67.50	68.44	67.90	68.17	0.7368
SVM (RBF kernel)	68.35	69.10	69.17	69.14	0.7500
Random Forest	67.30	68.53	66.93	67.72	0.7314

Table 14 Ablation study of the proposed CNN–SPN framework.

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
CNN	59.25	60.67	58.24	59.43	62.80
CNN + SPN	66.60	67.41	67.41	67.41	73.17
CNN + SPN + HI-tk	67.90	68.83	68.29	68.56	74.15
Proposed CNN–SPN	68.10	68.91	68.78	68.85	74.34