An Empirical Study of a PCA-Based Multivariate Framework for Interpretable Log Anomaly Detection

1st Jiří Setinský
Brno University of Technology
Brno, Czech Republic
ORCID: 0009-0008-6085-3642

2nd Martin Žádník
CESNET
Prague, Czech Republic
ORCID: 0000-0002-2099-2348

***Abstract*—Effective anomaly detection is crucial for increasingly complex system logs, yet current methods often face challenges with labeled data reliance, high computational costs, or limited interpretability. This paper empirically applies an established Multivariate Statistical Network Monitoring (MSNM) framework, which leverages Principal Component Analysis (PCA) with D and Q statistics, to the log anomaly detection domain. We evaluate its performance on three benchmark datasets (HDFS, BGL, Thunderbird), focusing on its semi-supervised nature (requiring only normal operational data), computational efficiency, interpretability via count vector feature contributions, and ease of deployment. Our results demonstrate competitive F1 scores comparable to some supervised and deep learning methods, maintaining low computational overhead without GPU dependency. Furthermore, its strong interpretability is showcased through case studies, identifying specific log event patterns causing anomalies. This study highlights the MSNM framework's potential as a practical, efficient, and interpretable solution for log anomaly detection.**

***Index Terms*—Log Anomaly Detection, PCA, Interpretability**

I. Introduction

Robust automated anomaly detection is crucial for managing the increasing volume and complexity of system logs and ensuring system reliability and security. However, many existing techniques face significant challenges.

Firstly, supervised methods typically require extensively labeled datasets, which is often time-consuming, costly, and impractical for dynamic real-world systems [1]. The Multivariate Statistical Network Monitoring (MSNM) framework, evaluated in this study, addresses this by primarily requiring only normal operational data for training. Minor anomalies within this training data can often be identified and managed during a careful calibration phase.

Secondly, although powerful, large language models (LLMs) and other advanced deep learning techniques frequently incur significant computational demands for both training and inference [2]. In contrast, this paper investigates an MSNM framework leveraging Principal Component Analysis (PCA), known for its computational efficiency, offering a more sustainable option for continuous log monitoring in data-intensive environments.

Thirdly, a considerable barrier to adopting many complex anomaly detection systems is their "black-box" nature, offering little insight into anomaly causes. This paper's framework is designed for interpretability. By using intuitive event count vectors (ECVs) as features and employing contribution analysis, operators can identify the specific log patterns that contributed to an anomaly score. This transparency is essential for understanding anomaly causes and building trust in the detection system.

Finally, deployability and ease of use are crucial for practical applications. The PCA-based MSNM method offers a straightforward hyperparameter setup. For example, anomaly detection thresholds for the D and Q statistics can often be robustly determined using percentiles (e.g., the 99th percentile) of their distributions derived from normal data.

This paper empirically evaluates this PCA-based MSNM framework for log anomaly detection, focusing on its effectiveness in addressing these challenges. Our main contributions are:

- Empirical evaluation of a PCA-based MSNM framework for log anomaly detection, primarily utilizing normal operational data.
- Comprehensive analysis of its performance, computational efficiency, ease of deployment, and interpretability.
- Detailed assessment across HDFS, BGL, and Thunderbird benchmark datasets, including a case study on interpretability via contribution analysis.

Section II reviews related work. Section III details the MSNM framework and its application to logs. Section IV describes the experimental setup. Section V presents the empirical results, including performance, computational analysis, and interpretability. Section VI and VII discuss the findings, limitations, and future work.

II. Related Work

Log-based anomaly detection employs unsupervised, semi-supervised, and supervised learning paradigms. This section contextualizes the evaluated semi-supervised Multivariate Statistical Network Monitoring (MSNM) framework [3] to benchmarked methods from each category.

978-3-903176-75-1

a) Unsupervised methods: Unsupervised methods identify anomalies by learning patterns directly from unlabeled log data. Principal Component Analysis (PCA) is a widely used technique applied to the entire dataset to capture variance. Anomalies are then detected as outliers with high squared prediction errors (Q-statistic) or extreme scores in the PC space. Xu et al. [4] explored this with Event Count Vectors (ECVs). Yang et al. [5] introduced SemPCA, enhancing PCA with semantic embeddings (TF-IDF weighted GloVe) to better separate anomalies.

b) Semi-supervised Methods: Semi-supervised approaches train a model of normal behavior using only explicitly designated normal log data, flagging deviations as anomalies. The applied MSNM framework, a representative of this category, trains its PCA model exclusively on normal ECVs to establish a precise baseline, using D and Q statistics for anomaly identification. LogCluster [6] groups normal log sequences into clusters, flagging new sequences that do not fit as anomalies. Deep learning methods include DeepLog [7], which trains an LSTM on normal log event IDs to predict the next event, signaling an anomaly upon unexpected predictions. LogAnomaly [8] similarly uses LSTMs for next-event prediction on normal data, incorporating Template2Vec embeddings. LogBERT [9] adapts BERT, often trained on normal logs with Masked Language Modeling (MLM); anomalies are detected via high prediction errors for masked tokens or low-density sequence embeddings.

c) Supervised Methods: Supervised methods require datasets labeled with both normal and anomalous instances to train a classifier. Traditional algorithms such as Support Vector Machines (SVMs) [10], often operating on ECVs, are common benchmarks despite the challenges of obtaining comprehensive anomaly labels. Deep learning approaches include LogRobust [11], which uses a Bi-LSTM with attention to predict anomaly scores from log sequences (trained on labeled data with FastText embeddings), and NeuralLog [12], which tokenizes raw logs with WordPiece, uses pre-trained BERT for contextual embeddings, and employs a Transformer encoder for binary classification.

While supervised and advanced semi-supervised deep learning methods often achieve high accuracy, they may incur drawbacks regarding labeling requirements, computational cost, or interpretability.

III. Framework Overview and Application to Logs

This paper investigates the application of an interpretable anomaly detection framework to log data, building upon the principles of Multivariate Statistical Network Monitoring (MSNM) detailed in [3].

A. Methodology Overview

Applying the MSNM framework [3] to log anomaly detection involves the following key stages:

1) Feature Engineering: Log Parsing and Event Count Vector Generation: The initial step is log parsing, transforming raw log messages into structured event templates using the Drain algorithm [13] for efficient online template extraction. Subsequently, Event Count Vectors (ECVs) are generated, where each element quantifies occurrences of a specific log event template (identified by Drain) within a predefined time window or log sequence. These interpretable ECVs form the input for subsequent modeling.

2) Optional Feature Filtering: To reduce ECV dimensionality and computational cost, an optional feature filtering step can be applied before PCA [14]. This process selects significant templates using two prevalence thresholds: Local (T_L) for event bursts within sequences, and Global (T_G) based on total dataset frequency. Counts of discarded templates are combined into a single *default* counter, ensuring no information loss. The final ECV thus contains features for frequent or bursty events along with this aggregate column.

3) PCA Modeling of Normal Log Behavior: PCA is applied to the (typically normalized) ECVs to reduce dimensionality, preserve significant variance, enhance computational efficiency, and mitigate noise. It transforms the original feature matrix $\mathbf{X}$ (observations as rows, features as columns) into orthogonal principal components (PCs) through the decomposition $\mathbf{X} = \mathbf{T_A}\mathbf{P_A^T} + \mathbf{E_A}$, where $\mathbf{T_A}$ is the *scores* matrix, $\mathbf{P_A}$ the *loadings* matrix, and $\mathbf{E_A}$ the matrix of *residuals*. The number of retained PCs is a key parameter, indicating the variance in normal data, balancing model complexity with capturing normal system structure.

4) Anomaly Detection using D and Q Statistics: Normal system behavior is modeled using selected PCs from training data (representing normal operational conditions in Phase I of MSNM). Deviations are measured using the D-statistic (Hotelling's T-squared) and the Q-statistic (Squared Prediction Error - SPE):

$$D_n = \mathbf{t_n}(\mathbf{\Sigma_T})^{-1}\mathbf{t_n^T} \qquad Q_n = \mathbf{e_n}\mathbf{e_n^T}$$

where $\mathbf{t_n}$ is the score vector for observation n, $\mathbf{e_n}$ the residual vector, and $\mathbf{\Sigma_T}$ the covariance matrix of training data scores. The Q-statistic measures conformity to the PC model, while the D-statistic measures deviation within the PC space. Anomalies are identified when D or Q exceeds thresholds derived from their distributions on normal training data (e.g., specific percentiles).

5) Interpreting Anomalies: A key strength of the MSNM framework is its interpretability, achieved through *contribution analysis*. Anomaly scores (D or Q) for a flagged log sequence can be decomposed to quantify each original input feature's influence (e.g., individual log event template counts). For the Q-statistic, large contributions indicate features whose values significantly deviate from the PCA model's prediction (large residuals); for the D-statistic, they stem from features pushing the sequence far from the normal data distribution's center in the learned subspace. By identifying and visualizing these contributions (e.g., via bar plots, Section V-C), specific event counters responsible for an anomaly are highlighted. This empowers system administrators to move beyond basic alerts, understand unusual system activities or errors from log event frequencies, and facilitate diagnostics and root cause analysis.

This principle is essential to exploratory techniques such as oMEDA [15].

IV. Experimental Setup

This section details the datasets, evaluation metrics, and implementation specifics used to empirically evaluate the applied MSNM framework.

A. Datasets

We utilized three benchmark datasets from the Loghub collection [16]: HDFS, BGL, and Thunderbird. These datasets, used in their complete or sampled forms, provide ground truth labels for log templates and anomalies. Table I summarizes their key characteristics.

HDFS (Hadoop Distributed File System) contains logs from a Hadoop cluster, with anomalies typically resulting from block corruptions or data node failures. It provides sequence-level anomaly labels, with sequences constructed by grouping related log entries using the *BlockID* field. No feature filtering was applied.

BGL (Blue Gene/L) logs are from a supercomputing system, reflecting hardware or software faults. While original labels are event-level, we adopted the aggregation strategy from [5] in which logs were grouped by component and split into sequences every 120 lines. A sequence was labeled anomalous if it contained at least one anomalous event. No feature filtering was applied.

Thunderbird is another supercomputer dataset, capturing diverse system faults. Following prior work [9], we used the first 20 million lines from the full dataset. Similar to BGL, original event-level labels were converted to sequence-level; sequences were generated using a hybrid rule of up to 40 lines or a maximum of 1 minute (whichever came first), and labeled anomalous if they contained at least one anomalous event. Due to the high dimensionality originating from the high number of unique log templates, feature filtering was applied using local (T_L = 0.01) and global (T_G = 0.001) prevalence thresholds, reducing the feature space from 1874 to 506 event counters.

TABLE I
Characteristics of Selected Benchmark Log Datasets

Characteristic	HDFS	BGL	Thunderbird
Total Messages	11,175,629	4,747,963	20,000,000
Parsed Templates	46	298	1,874
Total Sequences	575,061	85,577	500,965
Normal Sequences	558,223	49,094	344,948
Anomaly Sequences	16,838	36,483	156,017

B. Evaluation Metrics

To assess the performance and efficiency of all evaluated anomaly detection methods, we use:

F1-Score, the primary metric for detection accuracy. It is the harmonic mean of Precision (correctly identified anomalies among all flagged) and Recall (actual anomalies correctly identified). The F1-score is particularly suitable for imbalanced datasets, effectively balancing both metrics.

Training Time is the wall-clock time required to train each model on its designated data (e.g., normal for semi-supervised, mixed for supervised), reflecting model-building cost.

Prediction (Inference) Time is the total wall-clock time for a trained model to process the entire test dataset and assign anomaly scores, reflecting its overall throughput.

C. Implementation Details and Hyperparameters

The MSNM framework was implemented in Python, leveraging *Numpy* for core data manipulation and *Scikit-learn PCA* and *StandardScaler* for principal component analysis and data autoscaling (centering and scaling to unit variance), exclusively based on normal training data.

A key hyperparameter, the number of principal components (PCs) to retain, was initially guided by the column-wise k-fold (ckf) cross-validation algorithm for PCA [17], with a subsequent search conducted within ± 5 PCs of the ckf-suggested value. Thresholds for the D (Hotelling's T-squared) and Q (Squared Prediction Error) statistics were established between the 99th and 100th percentiles of their respective distributions from normal training data, with optimal thresholds corresponding to the $(1 - p\_value)$ percentile.

Baseline implementations for SVM and LogCluster were sourced from Loglizer [18]. LogAnomaly, DeepLog, LogRobust, SemPCA, and a standard PCA variant were adapted from the SemPCA repository [5]. LogBERT [9] and NeuralLog [12] originated from their dedicated repositories. Necessary modifications ensured consistent data handling and evaluation across all methods.

All experiments were conducted on the MetaCentrum[1] grid computing service, an HPC environment within the e-INFRA CZ infrastructure utilizing the PBS scheduler.

Hyperparameter optimization for all methods, aiming to maximize the F1 score, was performed using the Optuna framework [19]. During this phase, models were trained on 50% of each dataset's sequences and validated on an additional 10% (containing both normal and anomalous instances). The final selected and optimized hyperparameters for all evaluated methods and datasets are listed in Table II.

V. Experiments and Results

A. Performance Evaluation on Benchmark Datasets

This subsection evaluates the anomaly detection performance of the MSNM framework on the HDFS, BGL, and Thunderbird datasets, focusing on the F1-score. A 10-fold cross-validation (CV) procedure was employed to assess model robustness and to obtain F1-score distributions, which are presented in Figure 1. Optimal hyperparameters, determined as described in Section IV-C, were used for each method.

For the CV, each dataset was first shuffled and then partitioned into 10 equal-sized segments. In each fold, the training set comprised 50% of the data, including only normal sequences for the MSNM method. A 10% validation set was

[1] https://www.metacentrum.cz

TABLE II
HYPERPARAMETERS PER METHOD AND DATASET

Method	Parameter	BGL	HDFS	TBird
MSNM	pcs_to_use	80	11	178
	p_valueD	5.65e-4	2.73e-3	9.62e-2
	p_valueQ	4.06e-2	4.78e-6	1.53e-5
DeepLog	hidden_size	64	64	64
	layers	2	2	2
	epochs	10	10	10
	batch_size	512	512	512
	lr	0.002	0.002	0.002
LogAnomaly	hidden_size	128	128	128
	layers	2	2	2
	epochs	10	10	10
	batch_size	2048	1024	512
	lr	0.002	0.002	0.001
	lr_decay	0.740	0.810	0.790
LogRobust	hidden_size	128	128	128
	layers	2	2	2
	epochs	20	20	20
	batch_size	128	256	512
	lr	0.002	0.001	0.002
	lr_decay	0.990	0.920	0.930
LogCluster	max_dist	0.01	0.01	0.33
	anomaly_threshold	0.01	0.01	0.38
	bootstrap_samples	3500	3500	4500
SemPCA	threshold_mult	0.791	0.131	0.476
	n_components	5	10	4
	c_alpha	3.891	3.891	3.891
PCA	threshold_mult	0.100	0.127	0.105
	n_components	3	4	123
	c_alpha	3.891	3.891	3.891
SVM	penalty	l1	l1	l1
	tol	0.001	0.000	0.008
	C	3.231	0.031	6.365
	max_iter	300	700	100

reserved and not used for re-tuning during this evaluation phase, while the test set consisted of the remaining 40% of the data, used for evaluating the trained model. Segments rotated across folds, ensuring robust evaluation and yielding 10 F1 scores per method, thereby providing a robust estimate of generalization performance.

1) Overall Detection Performance: Figure 1 shows F1 score distributions from the 10-fold cross-validation for the MSNM framework and baseline methods. The framework consistently delivers strong and stable F1 scores. Unlike other PCA-based methods (e.g., standard PCA, SemPCA), MSNM benefits from straightforward hyperparameter tuning, and its combined D and Q statistics enable more comprehensive anomaly detection than relying solely on reconstruction error.

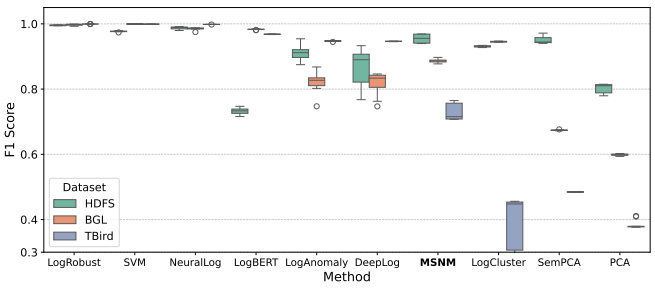

Fig. 1. Distribution of F1 scores from 10-fold cross-validation for the applied MSNM framework and comparative methods.

Table III presents average F1 scores across datasets, grouped by supervision level. MSNM achieves the best semi-supervised performance on HDFS (0.955) and is competitive on BGL (0.887), outperforming LogAnomaly and DeepLog. Its overall average F1 (0.857) is slightly lower than LogBERT (0.894) and LogAnomaly (0.892), primarily due to reduced performance on Thunderbird (0.730).

TABLE III
AVERAGE F1 SCORES PER DATASET AND OVERALL.

Category	Method	HDFS	BGL	TBIRD	Avg.
Supervised	LogRobust	0.996	0.997	0.999	0.997
	SVM	0.977	1.000	0.999	0.992
	NeuralLog	0.987	0.985	0.998	0.990
Semi-supervised	LogBERT	0.732	0.983	0.968	0.894
	LogAnomaly	0.910	0.820	0.947	0.892
	DeepLog	0.866	0.817	0.946	0.876
	MSNM	0.955	0.887	0.730	**0.857**
	LogCluster	0.931	0.945	0.393	0.756
Unsupervised	SemPCA	0.951	0.674	0.485	0.703
	PCA	0.803	0.599	0.384	0.595

Thunderbird presents a significant challenge due to its numerous unique log events. Event vectors remain high-dimensional (509, reduced to 178 by PCA) even after filtering, hindering effective detection of subtle anomalies. Despite this, MSNM still outperforms LogCluster and other PCA-based methods on Thunderbird.

Focusing on HDFS and BGL datasets, which have more manageable dimensionality (Table I), MSNM achieves an average F1 score of 0.921, surpassing all other semi-supervised methods. This indicates the framework's strong performance in log environments where the feature space dimensionality is not excessively large.

2) Performance with Varying Training Data Size: Figure 2 illustrates the MSNM framework's F1 score on the HDFS dataset with varying training data proportions, assessing its robustness to training data volume using the same cross-validation process. The plot demonstrates that MSNM maintains high performance even with relatively small amounts of normal training data. This is significant for practical deployments, as it means a reliable model of normal system behavior can be built from a small, anomaly-free log set, minimizing the need for extensive pre-deployment data. This also facilitates quick model updates by retraining on smaller,

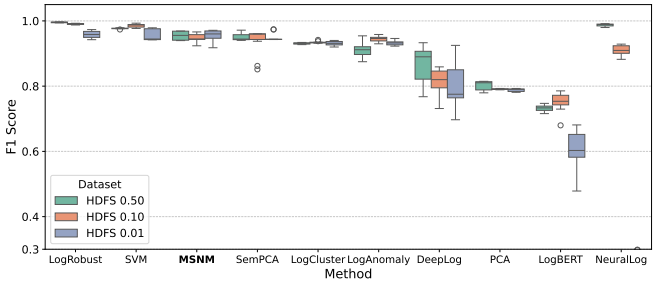

Fig. 2. F1 score of the benchmarked methods on the HDFS dataset with variable training data size.

recent normal traffic segments, enabling adaptation to evolving system behavior. Such robustness to training data size, combined with the efficiency of PCA-based modeling, enhances the framework's practical applicability and maintainability in dynamic operational environments.

B. Computational Resource Analysis

Computational efficiency is crucial for practical deployment. This subsection analyzes the resource consumption of the MSNM framework and baseline methods, focusing on model training time (measured for 50% of the dataset) and prediction (inference) time (for the entire dataset), as defined in Section IV-B. Figure 3 compares these averaged times over HDFS and BGL datasets.

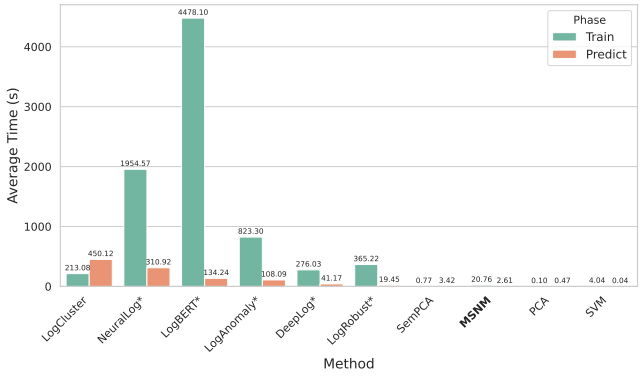

Fig. 3. Average model fitting and prediction times, averaged over HDFS and BGL datasets. Methods with * use GPU acceleration.

The MSNM framework demonstrates competitive computational efficiency, offering practical fitting times for periodic retraining and efficient batch prediction. Its modest resource consumption is comparable to other PCA-based methods, significantly outperforming deep learning models (with their substantial training requirements) and LogCluster (which has the slowest prediction time). Notably, MSNM achieves faster prediction times without GPU support, a practical advantage in resource-constrained environments compared to GPU-accelerated deep models. The framework's computational complexity stems primarily from PCA decomposition during training and matrix operations for predictions (e.g., D-statistic calculation), which adds some overhead. This overhead, however, is justified by enhanced detection and interpretability without hindering practicality.

Overall, the MSNM framework effectively balances detection performance, interpretability, and computational resource demands, presenting an excellent GPU-independent solution for log anomaly detection, particularly in resource-constrained environments.

C. Interpretability Case Study

A key advantage of the MSNM framework is its interpretability. This section demonstrates how contribution analysis (Section III-A5) aids in understanding detected anomalies by identifying contributing log event patterns, using the HDFS dataset as a case study.

Figure 4 presents the aggregated contributions of individual log event counts to anomaly scores across all detected HDFS test set anomalies. Systemic anomaly patterns include

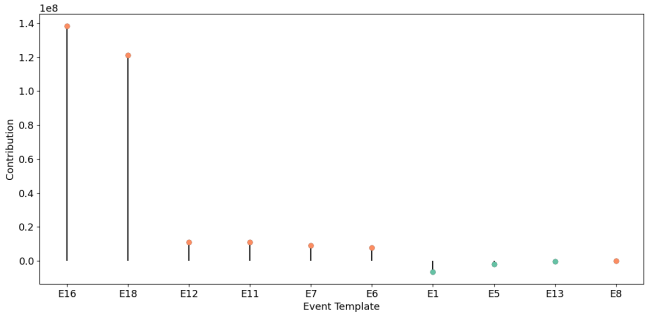

Fig. 4. Features with the highest aggregated positive (increased occurrence) and negative (decreased occurrence) contributions to HDFS anomalies.

increased occurrences of Event **16** (*Unexpected error trying to delete block...*), indicating metadata inconsistencies or I/O issues in block deletion (a common HDFS anomaly cause), and Event **18** (*AddStoredBlock request received, but it does not belong to any file...*), pointing to orphaned block registrations from earlier errors. Conversely, Event **1** (*Receiving block...*), a normally frequent event, appearing less often during anomalies, suggests data pipeline disruptions due to node unavailability or network partitioning; its absence signals an anomaly. These findings demonstrate the value of the contribution analysis for fault diagnosis.

Figure 5 illustrates contribution analysis for a specific and rare HDFS anomaly. Primary contributors are Event **12** (*Starting thread to transfer block...*), **11** (*Ask to replicate block...*), and **7** (*Transmitted block*), indicating the anomaly stemmed from numerous block transfers and replications. While Event **17** (*Got an exception while serving the block...*) contributed minimally, it likely triggered the issue, with the system's automatic response causing the unusual behavior. The contribution analysis explains the anomaly's event sequence and the system's reaction.

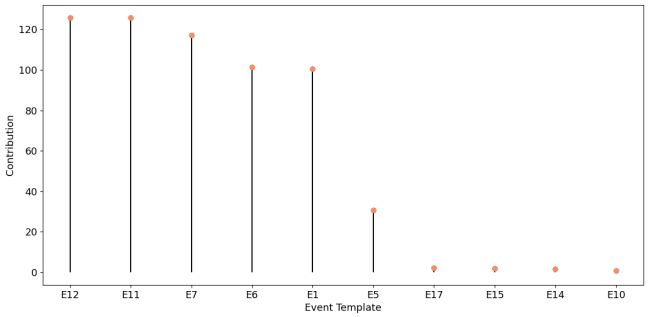

Fig. 5. Feature contributions for a single HDFS anomaly. Event 17 triggered the issue, but block transfers and replications caused the anomaly.

VI. Discussion

The empirical findings from our study strongly underscore the MSNM framework's potential for log anomaly detection across HDFS, BGL, and Thunderbird datasets, particularly in terms of its effectiveness, efficiency, interpretability, and deployability. The framework achieved strong F1 scores, comparable to deep learning methods on HDFS and BGL, utilizing only normal training data. Its efficacy with small training sets enhances practical deployability and adaptability. Computationally, the PCA-based approach proved efficient with modest, GPU-independent training and prediction times, offering a sustainable alternative to resource-heavy models. The D-statistics inclusion, while adding slight overhead, is justified by improved detection capabilities. A key contribution is the framework's interpretability. An HDFS case study showed how contribution analysis traces anomalies to specific deviating log event counts (both presence of errors and absence of normal events), providing actionable insights for administrators. This transparent nature, combined with straightforward hyperparameter tuning (CKF-guided PC selection, percentile-based D/Q thresholds), enables easy deployment. The MSNM framework generally outperformed similar baselines (SemPCA, basic PCA) and offered a compelling balance compared to deep learning methods, particularly given its semi-supervised nature and inherent interpretability. **Limitations** include its reliance on high-quality log parsing and the representational capacity of ECVs. It assumes mostly normal training data, requiring careful curation. PCA on ECVs may not capture complex time-based patterns as effectively as deep learning models. Furthermore, the method performs less effectively and more slowly with high-dimensional data, such as the Thunderbird dataset's many log templates, complicating subtle anomaly detection. **Future work** will involve enhancing feature engineering (e.g., incorporating log parameter values or reducing highly dimensional ECVs), exploring adaptive thresholding, and developing methods for automated verification of normal training data.

VII. Conclusion

This paper empirically evaluated a PCA-based Multivariate Statistical Network Monitoring (MSNM) framework for log anomaly detection on HDFS, BGL, and Thunderbird benchmarks. The framework achieved competitive F1 scores, comparable to supervised and deep learning methods, while leveraging only normal training data. Key strengths include its robustness to varying training data sizes, low GPU-independent computational overhead, and strong interpretability via Event Count Vector (ECV) contribution analysis, which helps pinpoint specific log event patterns causing anomalies (demonstrated on HDFS). This effective balance of performance, efficiency, and interpretability positions MSNM as a practical and valuable solution for log anomaly detection, especially where resource constraints, limited labeled data, or explainability are critical concerns.

Acknowledgment

This work was supported by Brno University of Technology under project number FIT-S-23-8141.

References

[1] T. Wittkopp, A. Acker, and O. Kao, "Progressing from Anomaly Detection to Automated Log Labeling and Pioneering Root Cause Analysis," Dec. 2023. arXiv:2312.14748 [cs] version: 1.

[2] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. R. So, M. Texier, and J. Dean, "The carbon footprint of machine learning training will plateau, then shrink," *Computer*, vol. 55, no. 7, pp. 18–28, 2022.

[3] J. Camacho, A. Pérez-Villegas, P. García-Teodoro, and G. Maciá-Fernández, "PCA-based multivariate statistical network monitoring for anomaly detection," *Computers & Security*, vol. 59, pp. 118–137, June 2016.

[4] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pp. 117–132, 2009.

[5] L. Yang, J. Chen, S. Gao, Z. Gong, H. Zhang, Y. Kang, and H. Li, "Try with simpler-an evaluation of improved principal component analysis in log-based anomaly detection," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 5, pp. 1–27, 2024.

[6] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th international conference on software engineering companion*, pp. 102–111, 2016.

[7] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 1285–1298, 2017.

[8] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs.," in *IJCAI*, vol. 19, pp. 4739–4745, 2019.

[9] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via bert," in *2021 international joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, 2021.

[10] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in ibm bluegene/l event logs," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pp. 583–588, IEEE, 2007.

[11] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, *et al.*, "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, pp. 807–817, 2019.

[12] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 492–504, IEEE, 2021.

[13] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *2017 IEEE international conference on web services (ICWS)*, pp. 33–40, IEEE, 2017.

[14] J. Camacho, K. Wasielewska, R. Bro, and D. Kotz, "Interpretable Feature Learning in Multivariate Big Data Analysis for Network Monitoring," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2024. arXiv:1907.02677 [cs, stat].

[15] J. Camacho, "Observation-based missing data methods for exploratory data analysis to unveil the connection between observations and variables in latent subspace models," *Journal of Chemometrics*, vol. 25, no. 11, pp. 592–600, 2011.

[16] S. He, J. Zhu, P. He, and M. R. Lyu, "Loghub: A large collection of system log datasets towards automated log analytics," *CoRR*, vol. abs/2008.06448, 2020.

[17] E. Saccenti and J. Camacho, "On the use of the observation-wise k-fold operation in pca cross-validation," *Journal of Chemometrics*, vol. 29, no. 8, pp. 467–478, 2015.

[18] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *27th IEEE International Symposium on Software Reliability Engineering, ISSRE 2016, Ottawa, ON, Canada, October 23-27, 2016*, pp. 207–218, IEEE Computer Society, 2016.

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*, pp. 2623–2631, ACM, 2019.