SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Resilient IT Systems: Engineering Approaches for Fault Tolerance and Recovery

Srinivasa Rao Jalluri[1], Dr. Sarbeswar Hota[2], Dr. Kamal Sutaria[3], Harshini R[4], Dr. R. Salini[5], Jaswinder Singh[6], Dr. R. Rajalakshmi[7], Jeyanthi P[8], Rahul Bhatt[9]

[1] Associate Professor, Department of Electrical and Electronics Engineering, VNR Vignana Jyothi Institute of Engineering & Technology, India. Email: srinivasarao_j@vnrvjiet.in ORCID: 0000-0002-8642-5012
[2] Associate Professor, Department of Computer Applications, Institute of Technical Education and Research, Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar, Odisha, India. Email: sarbeswarahota@soa.ac.in ORCID: 0009-0006-0921-8323
[3] Professor, Department of Computer Science and Engineering, Faculty of Engineering and Technology, Parul Institute of Technology, Parul University, Vadodara, Gujarat, India. Email: kamal.sutaria24554@paruluniversity.ac.in ORCID: 0000-0001-7557-7507
[4] Assistant Professor, Department of Computer Science, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: harshinir@maher.ac.in
[5] Associate Professor, Department of Computer Science and Engineering, Panimalar Engineering College, Tamil Nadu, India. Email: shalinirajendran@gmail.com ORCID: 0000-0001-7266-3389
[6] Assistant Professor, Department of Computer Science & Engineering, Noida International University, India. Email: jaswinder.singh@niu.edu.in
[7] Professor, Department of Electronics and Communication Engineering, Panimalar Engineering College, Chennai, Tamil Nadu, India. Email: rajeeramanathan@gmail.com ORCID: 0000-0002-4399-4071
[8] Assistant Professor, Department of Commerce, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: jeyanthicom@maher.ac.in
[9] Assistant Professor, Department of Computer Science & Engineering, Dev Bhoomi Uttarakhand University, Dehradun, Uttarakhand, India. Email: socse.rahul@dbuu.ac.in ORCID: 0009-0004-0486-9414

Abstract
Reliable banking systems demand uninterrupted service despite failures in infrastructure, network, or software components. Conventional fault tolerance approaches rely on static rules and basic machine learning models, which often fail to capture temporal dependencies, resulting in delayed fault detection and inefficient recovery. The aim is to design a robust, fault-tolerant, and recovery-oriented framework capable of identifying early failure patterns and ensuring rapid system restoration. Novelty is introduced by the Dumbo Octopus algorithm, self-attention with Stacked long short-term memory (DOA-SA-Stacked LSTM), enabling precise temporal learning and adaptive parameter tuning. A generalized dataset comprising system performance metrics, transaction logs, and failure records is utilized. Preprocessing involves the interquartile range (IQR) for data cleaning and the Z-score for normalization to ensure consistency and reliability. Feature extraction is performed using Independent Component Analysis (ICA) to capture temporal variations and system dynamics. The proposed model operates by first learning sequential dependencies through stacked LSTM layers, while the self-attention mechanism emphasizes critical time steps associated with anomalies. The DOA optimizes model parameters to improve convergence and prediction accuracy. Implementation is carried out using a Python-based deep learning (DL) model. Experimental results demonstrate an accuracy of 98.3%, a precision of 98.1%, a recall of 98.6%, an F1 score of 98.4%, and a recovery time of 6.8s, stimulated in Python. The method effectively models failure progression and recovery patterns, ensuring reliable operation in banking environments. The proposed approach provides an efficient and scalable solution for resilient Information Technology (IT) systems, enabling proactive fault management and rapid recovery.

Keywords— Fault Tolerance, Resilient Systems, Stacked LSTM, Self-Attention, Dumbo Octopus Optimization, Time-Series Analysis, Banking Systems.

1. Introduction

Distributed and multi-agent systems are made up of physically dispersed entities that use networks for communications in order to meet their common goals. In MASs, consensus tracking plays a key role; however, issues arise due to faulty behaviors and cyber-attacks, such as denial-of-service attacks and deception attacks, that affect agent-to-agent communication [1]. Likewise, distributed systems, including blockchains as a type of decentralized ledger technology (DLT) system, have been used in many domains, including finance, health care, Internet of Things (IoT), and supply chain management, by using consensus mechanisms such as Byzantine Fault Tolerance algorithms [2]. In traditional network systems, where both control and data planes are integrated, there are delays in failure conditions, while SDNs allow for centralized control [3, 4]. To enhance the performance of fault detection, reliability, and recovery systems, a great deal of research has been done on artificial intelligence (AI), machine learning (ML), and deep learning (DL) techniques. To counteract the impact of sensor loss and actuator problems, an observer-based event-triggered control method and an adaptive estimator have been developed in MAS [5].ML classifiers have been proposed in IoT and SDN to mitigate cyber-attacks and ensure quality of service in the presence of fault and attack conditions. These techniques illustrate the utility of AI, ML, and DL techniques in the modeling, prediction and mitigation of faults in networked systems. However, there still exist a number of challenges in the current state-of-the-art fault tolerant and consensus tracking algorithms [6]. Traditional graph-theoretic and event-triggered techniques do not effectively tackle the issue of cyber-attacks and emerging actuator faults in MAS. IoT and SDN systems face limitations in ensuring data completeness in the presence of attacks, and compensating sensor faults [7]. There are no adequate fault tolerance mechanisms in traditional machine learning techniques that could be deployed at different phases. In addition, most artificial intelligence algorithms and deep learning models cannot address faults by incorporating mechanisms to detect faults, recover from them, and optimize recovery efforts, thus resulting in delays, high recovery costs, and low adaptability [8]. This calls for an innovative framework that can address these challenges.

Research aim: The purpose of this research work is to develop an advanced and intelligent fault-tolerant and recoverable architecture for resilient IT systems, especially for banking applications through the utilization of the DOA-SA-Stacked LSTM. The technique emphasizes early fault detection, modeling of system behavior using temporal dependencies, and fast recovery for higher availability and better system reliability.

Research Organization: A research paper consists of five sections. The section 1 discusses resilient IT system concepts. The section 2 highlights the existing body of knowledge regarding fault-tolerant systems and intelligent recovery approaches. The section 3 presents the methodology suggested to achieve efficient fault detection and recovery. Section 4 presents an analysis of the performance of the proposed solution. Section 5 draws conclusions of the research and outlines areas of further investigation.

2. Related work

Table 1 presents a comparative overview of recent research works focused on fault-tolerant and resilient systems across different domains such as distributed control systems, multi-agent networks, cloud computing, and high-availability databases.

Table 1: Prior research on Fault Tolerance and Recovery in Distributed and High-Availability Systems

Ref.	Research Aim	Techniques	Experimental Results	Limitations
[9]	For networked systems, distributed fault estimation and fault-tolerant control provide stability and disturbance rejection.	Distributed Fault Estimation Observer (DFEO), Distributed Fault-Tolerant Control (DFTC), Linear Matrix Inequalities (LMIs), plug-and-play design	Achieves asymptotic fault estimation, maintains stability, and ensures strong disturbance rejection, validated via power network simulations	Simulation-based validation only; it needs extension to real large-scale nonlinear industrial systems
[10]	Address sensor drift in distributed parameter systems and maintain system stability	Time-varying spatiotemporal model, adaptive observer-based fault detection, joint state-and-fault estimation, feedback control	Accurate fault detection and estimation with real-time compensation and stable closed-loop performance	Limited to controlled experiments; needs extension to nonlinear and large-scale uncertain environments.

[11]	Achieve bipartite output synchronization in multi-agent systems under faults	Distributed fault-tolerant control, output regulation theory, state & fault estimator, and Riccati equation-based design.	Successful bipartite synchronization under faults validated via simulations	Needs scalability improvement and handling of more complex, nonlinear, and uncertain communication systems
[12]	Ensure consensus tracking in MAS under actuator faults and disturbances	The system incorporates a dual-layer architecture (cyber-physical), virtual actuator-based fault compensator	Maintains stable consensus tracking and robustness under faults in simulations	The approaches may struggle in highly uncertain and large-scale systems due to its lack of learning-based adaptability
[13]	Improve fault-tolerant task scheduling in cloud computing systems	Priority-Based Task Scheduling, replication + checkpointing hybrid model	60%+ recovery improvement and better resource utilization and QoS metrics	Needs real-time adaptive and large-scale heterogeneous cloud optimization
[14]	Develop robust distributed state estimation under sensor faults	Distributed observer network with local and neighbor information sharing	State convergence achieved with improved resilience against sensor faults	Limited handling of severe faults, delays, and large-scale dynamic networks
[15]	Reduce computational burden in fault-tolerant control of multi-controller systems	Data-based distributed policy iteration, fault compensators	Achieves fault tolerance with reduced computational cost and stable performance	Needs extension to nonlinear and large-scale uncertain systems
[16]	Enhance high-availability databases' fault detection and recovery	AI-based ML framework, predictive analytics, and real-time fault recovery	The model achieved a 95% detection rate and 40% reduction in recovery time	Needs scalability for large distributed systems and deeper DL integration

Although the proposed model can handle complex temporal linkages, it overcomes the limitations of other existing models based on simpler mechanisms, such as standard LSTM, to improve the fault tolerance and resilience of the IT system. The use of a DOA-SA-Stacked LSTM enables learning of short- and long-term dependencies, while the DOA mechanism contributes to dynamic parameter tuning that results in faster fault identification and reduced recovery time.

3. Methodology

It uses DOA-SA-Stacked LSTM to increase the efficiency of detecting and recovering from faults in resilient computing systems in a banking scenario. This involves collecting performance measurements, transaction logs, and fault logs, and then carrying out preprocessing that includes dealing with issues such as missing data, duplicate values, outliers, and normalization. ICA is used to extract features. As seen in Figure 1, hyperparameter adjustment has been carried out to improve convergence and performance.

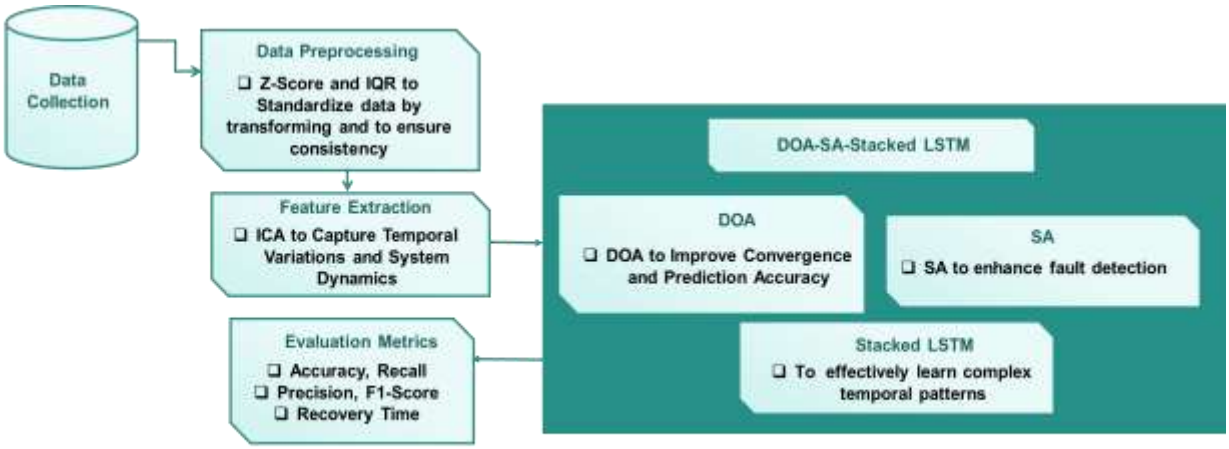

Figure 1: Methodology of the proposed method

3.1 Dataset description: The data used in this research project for building fault-tolerant and recoverable framework in resilient IT was sourced from open source website (https://www.kaggle.com/datasets/colabsss/banking-it-resilience-dataset). This dataset comprises of the transaction activity logs, infrastructure performance data, network conditions data, availability status of services provided, failures, recovery actions and final resilience classes of the system. This dataset is made up of 12000 entries and includes normal transactions as well as degraded and failed transaction cases. To train and test the model, 70% of the dataset is used for training while the rest of 30% is used for testing.

3.2 Preprocessing using Z-score and IQR: By using numerical data for standardization, Z-score normalization improves the effectiveness of machine learning algorithms by achieving a mean value of zero and a standard deviation of one.The IQR uses statistical measures to determine the degree of data spread within the middle range data set.

3.2.1 Z-score normalization: Normalization using Z-score method is employed during the preprocessing phase for converting the system performance parameters to be fed as input into the model to detect and recover from any faults in resilient computing infrastructures. This process makes sure that the features like system log information, resource consumption data, and transaction signals are scaled to the same range.

$$X_{score} = \frac{rss(n)-\mu}{\sigma} \quad (1)$$

$$rss_{j,i} = [rss(1), rss(2), \dots rss(N)] \qquad for\ 1 \le n \le N \quad (2)$$

Equations (1) and (2) X_{score} define the Z-score for the nth observation, where $rss(n)$ is the residual sum of squares for that point. Here, μ represents the mean of all RSS values, and σ is their standard deviation. The vector notation $rss_{j,i} = [rss(1), rss(2), \dots rss(N)]$ lists all RSS values for indices j and i, where N is the total number of observations, allowing for the standardization of errors.

3.2.2 Data cleaning using IQR: The use of IQR as a statistical method is done in order to detect and discard the outliers present in system performance measures, transactional data, and failure data. The main goal in the application of IQR is to improve preprocessing stability and performance through the elimination of outliers.

$$f(x) \leftarrow d_i = 0, x_i \le 0 \quad (3)$$

In equation (3), $f(x)$ refers to the function used during pre-processing or transformation of the data inputs in the resilient IT systems model. The parameter x_i refers to the value of the i^{th} data input in the input dataset that can include system data, transactions records, or performance measures. The variable d_i denotes the value of the data input after it is transformed; any values that satisfy the constraint of $x_i \le 0$ are set to zero.

$$IQR = q_3 - q_1 \quad (4)$$

In Equation (4), IQR represents the interquartile range used for outlier detection in the preprocessing stage of the proposed model. The variable $q_1$denotes the first quartile of the dataset, which marks the lower boundary of the central distribution of system performance or log data. The variable q_3 represents the third quartile, which defines the upper boundary of the central data distribution.

3.3 Feature extraction using ICA: ICA is a method of feature extraction in which a transformation of multivariate data to independent components takes place. With respect to the proposed architecture that is

fault tolerant and has recovery capability, ICA is used to uncover hidden patterns in a variety of data pertaining to transactions, problems, and system performance. In order to discover fault patterns, the ICA algorithm's main objective is to reduce redundancy and extract independent components from the data.

$$X = AS \quad (5)$$

$$S = WX \quad (6)$$

The equation (5) represents the observed data matrix X is a linear mixture of unknown source signals S through a mixing matrix A. Here, X denotes the collected input data from system logs and performance metrics, A represents the unknown mixing process that combines independent underlying sources, and S represents the original independent source components that the aim to recover. Equation (6) is used for source separation, where W is the unmixing matrix learned by the ICA algorithm. This matrix W is used to transform the observed data X is the back into statistically independent components S, which correspond to the extracted fault-relevant and normal behavior features in the proposed system.

3.4 DOA-SA-Stacked LSTM for Fault Detection and Recovery Optimization in Resilient IT Systems

The state-of-the-art Deep Learning architecture known as stacked LSTM with self-attention is composed of multiple layers of LSTMs that assist in identifying both short-term and long-term dependencies in a sequence. This makes it perfect for learning intricate temporal patterns from system logs and fault-related data. In order to improve parameters and guarantee a better trade-off between exploration and exploitation, DOA is a bio-inspired optimization method that mimics animal hunting behavior.Stacked LSTM would be used to increase fault detection accuracy through deep temporal dependencies, whereas DOA will help to tune the model's parameters to maximize convergence and recovery time.

3.4.1Self-Attention for Enhancing Critical Feature Extraction in Fault Detection and Recovery Systems

One DL strategy is self-attention, which allows the DL model to concentrate on particular elements of a sequence by giving different timestamps in the input dataset different degrees of importance. Compared to standard sequence-based methods, self-attention facilitates the identification of patterns in system logs, transaction records, and failure cases. The purpose of employing self-attention here is to increase fault detection efficiency by focusing on relevant temporal aspects, making it easier for the DL model to recognize early failure indications, and aiding quicker decision-making regarding system recovery.

3.4.2 Stacked LSTM for sequential data arrangement:

A DL model called stacked LSTM uses a series of LSTM layers to identify both short and long dependencies in a sequential dataset. In order to facilitate fault detection and increase recovery efficiency, the objective is to be able to identify and learn complex patterns from the system's performance data.

$$h_t^{(l)} = LSTM_{(l)}(h_t^{(l-1)}, x_t) \quad (7)$$

Equation (7) represents the operation of a stacked LSTM network, where each layer processes sequential data to learn temporal dependencies. Here, x_t denotes the input data at time step t, such as system logs, performance metrics, or failure indicators in the IT environment. The term $h_t^{(l-1)}$ represents the output (hidden state) from the previous LSTM layer, which serves as the input to the current layer l, allowing information to flow through multiple layers. The function $\text{LSTM}^{(l)}$ performs nonlinear transformations to capture complex temporal relationships, and the resulting output $h_t^{(l)}$ encodes higher-level feature representations in resilient IT systems.

3.4.3 DOA-Driven Optimization: The performance of the proposed fault-tolerant and recovery-based system can be improved by using DOA as an optimization technique. The stacked LSTM with self-attention system used for fault detection and recovery in the banking IT system incorporates the DOA stages into its tuning and optimization process.

i. Initialization Phase: Early in the algorithm, DOA starts the candidates for solutions by initializing model parameters such as learning rate, number of LSTM layers, number of hidden nodes, and attention coefficients.

ii. Capture Stage (Exploration and Exploitation Balance): Based on performance metrics like accuracy, DOA adjusts model hyperparameters, recovery period, and availability rate. The use of a dynamic switch allows the system to achieve balanced exploitation of good solutions and exploration of other configurations, resulting in efficient optimization of the stacked LSTM–attention model.

iii. Seduction, Scare, and Escape Stage (Exploitation of Locals): In subsequent stages, DOA refines the best-performing parameter settings to guarantee fault prediction accuracy and reduce recovery time. Switching based on iterations avoids local optimums and helps in faster convergence of parameters towards the optimum resilient system configuration. Using this method in optimizing stacked LSTM with attention model increases the accuracy of fault detection, reduces recovery time, and increases system availability, thus ensuring reliable and resilient operation of the IT system.

Algorithm 1: DOA-SA-Stacked LSTM Algorithm for Fault Detection and Recovery

Input:
D → Banking IT system dataset, Epochs → Training epoch, N → Population size, lb, ub → Parameter bounds
Output:
O → Optimized fault detection and recovery model
Begin
1. Load dataset D
2. Preprocess data: - Z-score normalization:
$X_{score} = \frac{rss(n)-\mu}{\sigma}$
- Data cleaning using IQR:
IQR = q3 - q1
Remove outliers and invalid entries
3. ICA feature extraction:
Compute:
$X = AS; S = WX$
Extract independent fault-related features
4. Build SA-Stacked LSTM:
Apply stacked LSTM layers
$h_t^{(l)} = LSTM_{(l)}(h_t^{(l-1)}, x_t)$
Apply self-attention mechanism; Capture critical temporal fault patterns
5. DOA initialization:
Initialize candidate solutions for:
Randomly generate solutions within lb and ub
6. DOA optimization:
If rand < 0.1
Explore new parameter combinations
Else
Update solutions using random differences
End If
Perform exploration and exploitation; Update best parameter set
7. Fault classification and recovery:
8. Evaluate performance
9. Return O
End

Firstly, Algorithm 1 pre-processes the data related to the IT system of banking using Z-score normalization and outlier removal based on IQR values. Then, Independent Component Analysis is performed for extracting fault-related independent components. The stacked LSTM along with SA learns fault-related temporal patterns, whereas DOA optimizes the parameters of the learning model.

4. Result and discussion

The results from the experimental evaluation of the DOA-SA-Stacked LSTM model will be discussed in this section. An experiment comparison will also be conducted between the stacked LSTM proposed and the DOA models by implementing machine learning algorithms such as RF and LSTM [16]. In all experiments, the use of Python programming was employed.

4.1 Multidimensional Temporal and Harmonic Feature Analysis: Figure 2 represents segmented acoustic rail and multi-dimensional timbre features. Figure 2(a) shows that there is a strong positive correlation between average response time (200-1900 ms) and the packet loss rate (0-900 packets). Clustering was seen in the range of 600-900 ms response time and 250-450 packets loss, indicating stable transmissions. The CPU Utilization (50-60%) and Memory Consumption (40-50%) density distribution in Figure 2(b) reflects an efficient multimedia processing with balanced resource utilization. Figure 2(c) shows the timbre representation by three clusters with low latency areas (95-100 stability) compared to high latency areas (65-80 stability). Figure 2(d) shows that the spectral amplitude distributions are varying intensities in the ranges

of 0 and 100. In addition, the most dominating layers have ranges of 40-80 with some instances more than 90 units. Finally, the synchronized signal variations were shown in Figure 2(e) with transmission delays of 0.2-1.0 and packet variations of 0.6-1.0 over 300 samples. The harmonic frequency responses for different timbre types are shown in Figure 2(f), where the maximum amplitude values are seen to be almost equal to 1.5 up to the frequency value of 5-6 and minimum amplitude values less than 0.3 units at frequencies of 2-3 and 7-8.

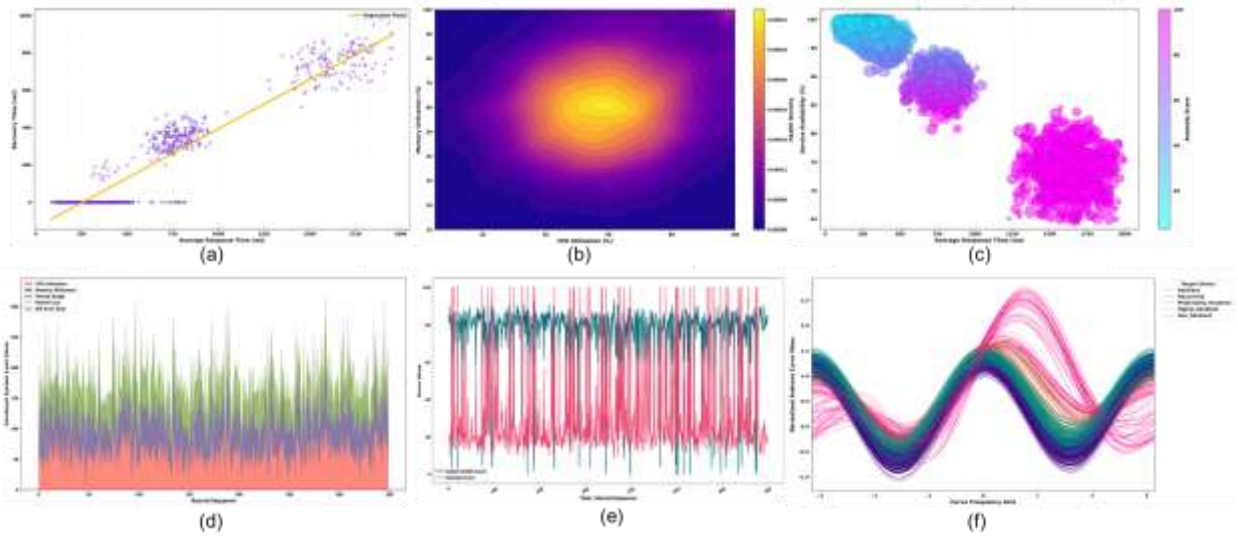

Figure 2: Analysis of (a) Delay growth and packet variation analysis, (b) Resource utilization distribution during multimedia processing operations, (c) Cluster-based separation of latency and stability feature groups (d) Temporal energy variation across sequential acoustic segments, (e) Dynamic synchronization behavior of transmission-related parameters, (f) Frequency variations across multiple acoustic categories

4.2 Metrics explanation

Accuracy: The percentage of correct predictions between the situations where the problem occurred and the cases where it did not is the statistic used to assess the model's accuracy. **Precision:** This score assesses how well the fault's occurrences were predicted. Stated differently, it computes the ratio of correctly predicted fault cases to those where the fault was projected to occur. **Recall:** This measure assesses how well the model can anticipate the fault's existence in actual situations. **F1-Score:** The F1-Score is a single indicator of how well a model detects faults by taking the harmonic mean of precision and recall. **Recovery Time:** The average time taken by the system to restore normal operation after a fault is detected, reflecting the efficiency of the recovery process.

4.3 Performance evaluation: The proposed DOA-SA-Stacked LSTM is trained using an experimental dataset from a high availability database environment [16]. These datasets consist of transaction database logs, performance statistics, and actual failures reported in real-life systems from various domains such as financial services, health care, and e-commerce industries. Table 2 shows the comparative analysis of performance metrics between (RF, and LSTM) [16] and the proposed DOA-SA-Stacked LSTM. As evident from the table, the proposed DOA-SA-Stacked LSTM gives the maximum accuracy score of 95.6% and recall score of 96.3%, thereby proving its fault detection capabilities. It also gives the maximum F1-score of 97.1% and minimum recovery time of 9.5 seconds.

Table 2: Performance evaluation of the proposed model is trained using the existing dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1- Score (%)	Recovery Time (s)
RF [16]	92.5	91.0	93.2	92.1	12.5
LSTM [16]	95.0	94.5	95.5	95.0	10.0
DOA-SA-Stacked LSTM [Proposed]	95.6	95.3	96.3	97.1	9.5

With the highest accuracy of 98.3% and recall rate of 98.6%, the suggested model outperformed the other models, including the DOA-SA-Stacked LSTM. As shown in Table 3 and Figure 3 below, the suggested model also

outperforms the RF and LSTM models with the highest precision of 98.1% and the highest F1-score of 98.4%. Additionally, the suggested model obtains the lowest recovery time of 6.8 seconds.

Table 3: Comparison of existing and proposed models trained on the banking IT resilience dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1- Score (%)	Recovery Time (s)
RF	94.2	93.8	94.5	94.1	11.0
LSTM	96.8	96.4	97.0	96.7	8.5
DOA-SA-Stacked LSTM [Proposed]	98.3	98.1	98.6	98.4	6.8

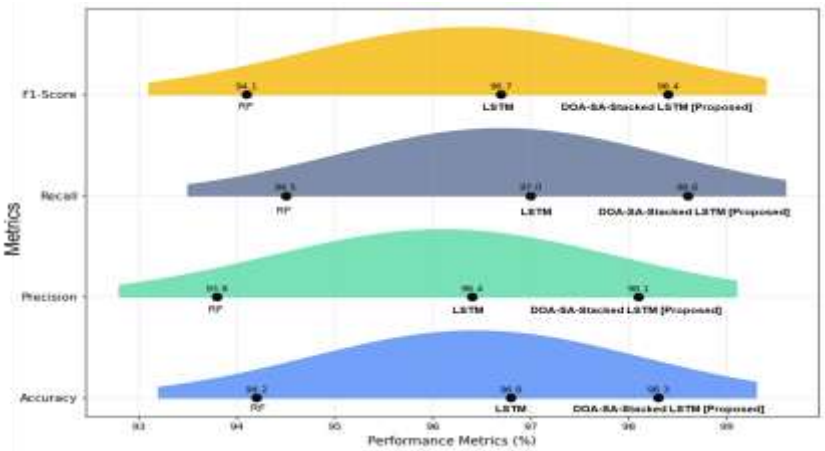

Figure 3: Graphical presentation of existing trained in the proposed model

4.4 K-Fold validation: The results of the five-fold validation process demonstrate the stability and dependability of the proposed DOA-SA-Stacked LSTM architecture in terms of detecting and fixing errors in banking IT systems. The great generalization ability and minimal chance of overfitting in the case of the suggested approach are demonstrated by the consistent performance in terms of accuracy, precision, recall, F1-score, and recovery time across all folds. These average outcomes demonstrate the effectiveness and efficiency of the proposed framework in making accurate predictions and speeding up system recovery (Table 4).

Table 4: 5-Fold validation results of proposed DOA-SA-Stacked LSTM model

Folds	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Recovery Time (s)
Fold 1	97.9	97.5	97.8	97.6	7.2
Fold 2	98.1	97.8	98.0	97.9	7.0
Fold 3	98.3	98.0	98.2	98.1	6.9
Fold 4	98.5	98.2	98.4	98.3	6.7
Fold 5	98.4	98.1	98.3	98.2	6.8
Average	98.2	97.9	98.1	98.0	6.92

5. Conclusion

In this regard, the proposed DOA-SA-Stacked LSTM model is quite useful in improving fault tolerance and recovery in resilient information technology systems due to its ability to exploit complex temporal dependencies and optimize model parameters for better accuracy. According to experimental findings, the accuracy of the proposed model was found to be 98.3%, the precision level was 98.1%, recall stood at 98.6%, F1 score was 98.4%, while the recovery time was recorded at 6.8s, stimulated in Python. On the downside, the proposed solution has its limitations associated with its reliance on simulated and past data that might not necessarily reflect real-time dynamics within the banking systems. Moreover, computational complexity is relatively higher in the case of the proposed model due to its ML and DL architecture optimization.

Reference

1. Liu, C., Jiang, B., Wang, X., Yang, H., and Xie, S., 2022. Distributed fault-tolerant consensus tracking of multi-agent systems under cyber-attacks. IEEE/CAA Journal of Automatica Sinica, 9(6), pp.1037-1048. https://doi.org/10.1109/JAS.2022.105419
2. Marcozzi, M. and Mostarda, L., 2024. Analytical model for performability evaluation of Practical Byzantine Fault-Tolerant systems. Expert Systems with Applications, 238, p.121838. https://doi.org/10.1016/j.eswa.2023.121838
3. Menaceur, A., Drid, H., and Rahouti, M., 2023. Fault tolerance and failure recovery techniques in software-defined networking: A comprehensive approach. Journal of Network and Systems Management, 31(4), p.83. https://doi.org/10.1007/s10922-023-09772-x
4. Huang, S., Liao, F., and Teo, R.S.H., 2022. Fault-tolerant control of a quadrotor based on sensor fault diagnosis and recovery information. Machines, 10(11), p.1088. https://doi.org/10.3390/machines10111088
5. Myllyaho, L., Raatikainen, M., Männistö, T., Nurminen, J.K. and Mikkonen, T., 2022. On misbehaviour and fault tolerance in machine learning systems. Journal of Systems and Software, 183, p.111096. https://doi.org/10.1016/j.jss.2021.111096
6. Ramani, S. and Jhaveri, R.H., 2022. ML-based delay attack detection and isolation for fault-tolerant software-defined industrial networks. Sensors, 22(18), p.6958. https://doi.org/10.3390/s22186958
7. Jammalamadaka, S.K.R., Chokara, B., Jammalamadaka, S.B. and Duvvuri, B.K., 2024. Using DL Models in the Service Layer to Enhance the Fault Tolerance of IoT Networks. Electronics, 13(22), p.4334. https://doi.org/10.3390/electronics13224334
8. Rehman, A.U., Aguiar, R.L., and Barraca, J.P., 2022. Fault-tolerance in the scope of cloud computing. Ieee Access, 10, pp.63422-63441. https://doi.org/10.1109/ACCESS.2022.3182211
9. Liang, D., He, Z., Ding, S. and Yang, Y., 2023. Distributed fault estimation and fault-tolerant control of interconnected systems with plug-and-play features. IEEE Transactions on Circuits and Systems I: Regular Papers, 71(1), pp.431-442. https://doi.org/10.1109/TCSI.2023.3330839
10. Balasubramanian, V., Aloqaily, M. and Reisslein, M., 2023. Fed-TSN: Joint failure probability-based federated learning for fault-tolerant time-sensitive networks. IEEE Transactions on Network and Service Management, 20(2), pp.1470-1486. https://doi.org/10.1109/TNSM.2023.3273396
11. Zhang, J., Ding, D.W., Lu, Y., Deng, C. and Ren, Y., 2022. Distributed fault-tolerant bipartite output synchronization of discrete-time linear multiagent systems. IEEE Transactions on Cybernetics, 53(2), pp.1360-1373. https://doi.org/10.1109/TCYB.2021.3137346
12. Yin, Y., Wang, F., Liu, Z. and Chen, Z., 2022. Distributed adaptive fault-tolerant control for multiagent systems via virtual-actuator-based reconfiguration. IEEE Transactions on Cybernetics, 53(12), pp.7497-7508. https://doi.org/10.1109/TCYB.2022.3169692
13. Mushtaq, S.U., Sheikh, S. and Idrees, S.M., 2025. Enhanced priority based task scheduling with integrated fault tolerance in distributed systems. International Journal of Cognitive Computing in Engineering, 6, pp.152-169. https://doi.org/10.1016/j.ijcce.2024.12.006
14. Yang, G., Rezaee, H., Serrani, A. and Parisini, T., 2022. Sensor fault-tolerant state estimation by networks of distributed observers. IEEE Transactions on Automatic Control, 67(10), pp.5348-5360. https://doi.org/10.1109/TAC.2022.3190429
15. Wei, Q., Li, H., Li, T. and Wang, F.Y., 2022. A novel data-based fault-tolerant control method for multicontroller linear systems via distributed policy iteration. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 53(5), pp.3176-3186. https://doi.org/10.1109/TSMC.2022.3223910
16. Gadde, H., 2024. AI-Powered Fault Detection and Recovery in High-Availability Databases. International Journal of Machine Learning Research in Cybersecurity and Artificial Intelligence, 15(1), pp.500-529. https://ijmlrcai.com/index.php/Journal/index