[1]Ms.Nupur Jain,
[2]Dr.Mahaveer Kumar Sain
[3]Ms.Deepti Shrimal
[4]Dr.Mohammed Ismail B

Enhancing Data Reliability in Cloud Storage Management: Proactive Fault Prevention with Advanced Machine Learning Strategies

Journal of Electrical Systems

***Abstract*—**The increasing reliance on cloud storage systems to manage massive volumes of data has highlighted the need for robust, efficient, and intelligent fault-prevention mechanisms. Traditional fault tolerance methods rely on reactive measures, leading to increased downtime and inefficiencies. This study introduces a proactive fault prevention approach using advanced machine learning techniques to enhance data resilience in cloud storage management. Utilizing the NSL-KDD dataset, undergoes preprocessing, including feature-label separation, one-hot encoding of categorical variables, and feature scaling to standardize input values before being split into training and testing sets. A Long Short-Term Memory (LSTM) model is implemented for failure prediction for binary classification. The model is trained using the Adam optimizer and evaluated with accuracy, loss, confusion matrix, and ROC curve, achieving high accuracy (93.87% training, 92.92% testing). To further enhance reliability, a reinforcement learning (RL)-based replica placement strategy is introduced, dynamically adjusting the number of replicas based on network topology, access patterns, node reliability, and resource constraints. Furthermore, failure to failover provisions allows continuous latency measuring for the system to perform optimally. The integration of resources and the use of predictive analytics in this work greatly improve cloud storage reliability and efficiency.

***Keywords*—**Cloud Computing, Data Reliability, Cloud Storage, Fault Prevention, Cloud Data Management, Machine Learning, NSL-KDD, Reinforcement Learning, LSTM.

I. INTRODUCTION

Cloud computing, through its on-demand scalable services along with cost savings and efficiency with flexibility, transformed the way consumers and organizations handle IT resource management [1][2]. Cloud computing enables users to store and access enormous amounts of data through the Internet through its key storage component [3][4] Organizations now recognize cloud storage as essential for data management because it provides secure remote data storage that can grow while making the data easily accessible [5][6]. The advantages of cloud storage exist while the systems experience failures that generate substantial service interruptions combined with data loss and deteriorated service quality [7][8]. The complexity of cloud infrastructure includes coordination over a huge amount of physical machines (PMs), virtual machines (VMs), and data centers dispersed across different locations, driving these risks[9][10].

Cloud storage management has become a critical area of focus where data reliability in cloud storage management is enhanced [11][12][13]. Whichever, cloud system faults due to software or hardware failures, human error results in system downtime, data loss performance degradation [14]. However, as cloud computing is

[1] Research Scholar,Department of Computer Science & Informatics,
Maharishi Arvind University, Jaipur, India
Email: jain n3@yahoo.co.in
Professor,Department of Computer Science & Informatics,
Maharishi Arvind University, Jaipur , India
Email: mahaveersain@gmail.com
Assistant Professor, Department of Computer Science,
MLSU, Udaipur, India
Email: deepti_na@mlsu.ac.in
Professor, Department of Artificial Intelligence & Machine Learning
P.A. College of Engineering Mangalore,
Karnataka, India
Email: aboutismail@gmail.com

developing, traditional fault tolerance (FT) methods, such as replication and checkpointing are widely used to guarantee data reliability [15][16]. Replication ensures data redundancy across several systems to avoid loss, and on the other hand, checkpointing saves the system state to facilitate recovery after failure [17][18]. These mechanisms, nevertheless, have their pros which include; high implementation costs and low resource demands. Hence, there is an increasing demand for proactive fault prevention measures that are aimed at preventing failure before it happens[19][20]. This way, using real-time monitoring and analyzing the signs of system decline, in terms of performance and hardware problems, cloud systems can prevent potential failures [21][22].

However, there are other more sophisticated replication approaches, such as adaptive and dynamic replication schedules, that will guarantee the availability of data to meet high requirements or when systems fail [23]. These approaches of new generation envisage to achieve maximum utilization of the resources and at the same time ensure high availability of data to the users and business organizations [24][25]. Some of the research challenges in cloud computing include; enhancing the cloud system fault tolerance and accurately predicting the cloud performance [26]. This study tackles important issues in cloud computing by improving data dependability in cloud storage management through the use of ML models to enhance fault prevention in cloud performance. A better option for efficiently monitoring automation systems is provided by current technologies, which include LSTM based on ML and reinforcement learning.

A. Motivation /Novelty and Contribution

The proposed work aims at enhancing the efficiency parameters and stability of cloud storage systems, as the current systems face numerous difficulties due to vast quantities of data and the increasing sensitivity and specificity of certain applications. Traditional methods for failure prediction and replica placement are often static and inefficient, motivating integration of advanced ML techniques, such as LSTM for failure prediction and reinforcement learning for dynamic replica optimization, to improve data availability and system performance. This paper contribution is listed below:

- Develop a fault prevention cloud storage system integrating advanced machine learning and reinforcement learning techniques to enhance data reliability and optimize replica placement dynamically.
- Utilized the NSL-KDD dataset to gather relevant data for failure prediction and replica placement optimization in cloud storage systems.
- Applied robust data cleaning, one-hot encoding for categorical features, and feature scaling to prepare the dataset, ensuring consistency and enhancing the performance of models.
- Trained an LSTMmodel for failure prediction and implemented a reinforcement learning agent to optimize replica placement based on system dynamics.
- The LSTM model's performance was evaluated using metrics such as recall, accuracy, precision, and F1-score; the reinforcement learning framework's performance was monitored via latency and failover responses.
- Applied reinforcement learning for dynamic optimization of replica placement based on network conditions, data criticality, and resource availability.
- Implemented real-time latency monitoring to detect high-latency events and trigger failover mechanisms for improved system resilience.

B. Structure of the paper

This is the proposed structure of the paper: Previous research on data reliability in cloud storage management utilizing various ways is presented in Section II, while the recommended material and methods are presented in Section III. The experimental data are discussed and examined in Section IV. Section V presents a conclusion and further research.

II. LITERATURE REVIEW

Literature review based on data reliability in cloud storage management using multiple methods and techniques for fault prevention is present in this section. Also summary of related work is in Table I.

Wen et al. (2024) the data security and reliability of cloud storage are increasingly threatened by unreliable factors, resulting in major data security problems. As the data security of cloud platforms becomes increasingly prominent, this study starts with the current cybersecurity situation and the challenges of cloud data storage and emphasizes the urgency of developing an efficient and reliable data verification mechanism. Subsequently, they propose a novel type of machine learning model that leverages advanced algorithms to identify and defend against potential threats of data breaches and corruption. In addition, they have verified an effectiveness of a model through a series of rigorous experiments, and an experimental outcomes show that the model can not only accurately identify various security vulnerabilities, but also adapt to different cloud platform environments, and provide a strong guarantee in terms of data integrity and confidentiality[27].

Kumar (2024) study a lot of research on AI-driven strategies, like how to improve security and privacy, get data faster, use advanced compression techniques, get rid of duplicate data, and manage data automatically throughout its lifecycle. Aside from that, they also look at how AI-powered storage options might help lower costs and meet the needs of specific industries. AI and ML can make cloud storage systems much better, according to their research. They can also lead to new, cheaper, and more environmentally friendly storage options. This study offers ideas for future research that could better use AI technologies to meet the growing need for cloud storage services by putting together existing research and finding gaps in the field[28].

Shahid et al. (2023) study enhances cloud fault detection using ML algorithms—Naïve Bayes, LibSVM, MLR, SMO, KNN, and RF—with delta-checkpointing (D-CP). Data from ETH Zurich's HPC system and virtual machines were analyzed using 5-fold cross-validation. Results showed NB excelled on CPU-Mem, SMO on HDD, and RF had high accuracy but higher time complexity. Modified SMO (MSMO) with D-CP is proposed to enhance fault prediction, accuracy, and reliability[29].

Tengku et al. (2022) have put up a thorough assessment method to forecast task failure. They constructed five TML models and three DL models to evaluate their ability to predict work and task failure using the GCT dataset. They identified two top-performing models, one based on DT and the other on RF, for task-level prediction. The accuracy score for the models is 89.75% and the F-value is 0.9154. When it comes to determining whether a work or task has been terminated, the TML models outperform the DL models by a small margin[30].

Uppal et al. (2022) proposed defect prediction model was evaluated using the RF, GNB, KNN, and DT algorithms; on the dataset that was provided, random forest performed better than the others. With an accuracy of 91.25 percent, random forest was determined to be the most effective of the ML approaches used to IoT-based sensors for monitoring this hospital automation process. The suggested model has the potential to aid the user in deciding on the suggested solution and in controlling unforeseen losses caused by automation process errors [31].

Alzahrani et al. (2022) offer a hybrid method that combines erasure coding with replication to achieve the best storage solution, with an emphasis on recovery and dependability. To facilitate future dynamic structure building and data model testing, learning and training methods were established. Various experimental setups are formulated using RAID architecture. The suggested hybrid framework has a substantial effect on cloud storage performance, as shown by the total results. It was determined that the ideal setup for optimal performance was RAID-6c on the server side [11].

TABLE I. CONCISELY SUMMARIZES EACH STUDY'S METHODOLOGY, KEY RESULTS, BENEFITS, DRAWBACKS, AND POTENTIAL FUTURE IMPROVEMENTS

Reference	Methodology	Results Analysis	Advantages	Limitations	Future Work
Wen et al. (2024)	Proposed an ML-based data reliability mechanism to detect security vulnerabilities in cloud storage.	The model effectively identifies security threats and adapts to different cloud environments.	Ensures data integrity and confidentiality in cloud storage.	May require continuous updates to adapt to emerging threats.	Enhance model efficiency and scalability for real-world cloud platforms.
Kumar (2024)	Investigated AI-driven strategies for improving cloud	AI and ML significantly enhance cloud storage	Reduces costs and enhances	Challenges in integrating AI with existing	Explore AI-based solutions for industry-specific

	storage security, compression, deduplication, and automation.	performance and efficiency.	environmental sustainability.	cloud infrastructure.	cloud storage needs.
M. Shahid et al. (2023)	Applied ML algorithms (NB, LibSVM, MLR, SMO, KNN, RF) with delta-checkpointing for cloud fault detection.	RF achieved high accuracy, while SMO provided better time efficiency. Modified SMO (MSMO) was proposed for improvement.	Improves fault prediction and reliability in cloud computing.	RF had higher time complexity.	Optimize MSMO further for better fault tolerance in cloud computing.
Tengku et al. (2022)	Evaluated five TML and three DL models for predicting task/job failures using the GCT dataset.	Decision Tree (DT) and RF performed best with 89.75% accuracy and 0.9154 F-score.	TML models outperformed DL models in classifying task failures.	Limited dataset scope; may not generalize to all cloud environments.	Extend analysis with diverse datasets and improve model generalization.
Uppal et al. (2022)	Used ML models (DT, KNN, GNB, RF) for cloud-based fault prediction	RF achieved the highest accuracy of 91.25%.	Efficient monitoring of IoT-based automation processes.	Model performance depends on the dataset quality and variability.	Expand model applications to other IoT-based industries.
Alzahrani et al. (2022)	Developed a hybrid model combining replication and erasure coding for cloud storage optimization.	RAID-6c at the server side provided optimal storage performance.	Enhances reliability and recovery in cloud storage.	Requires high computational resources for real-time deployment.	Improve hybrid model efficiency for large-scale cloud applications.

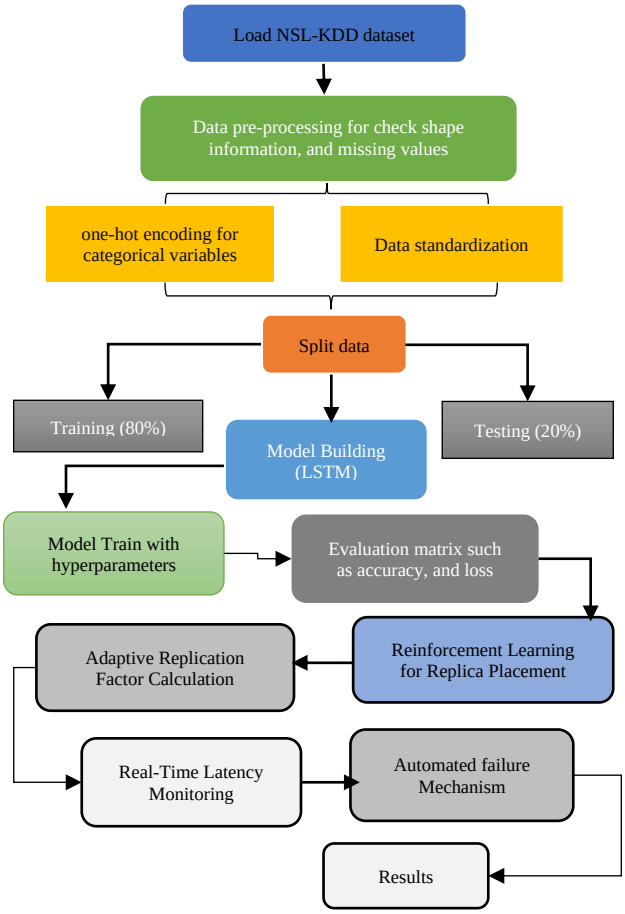

Fig. 1. Flowchart of cloud storage management for proactive fault prevention with improve data reliability

III. METHODOLOGY

The proposed methodology integrates advanced machine learning techniques to improve data reliability in cloud storage management with proactive fault prevention, this propose system shows in Figure 1. Initially, the NSL-KDD dataset undergoes exploratory data analysis and pre-processing, including feature-label separation, one-hot encoding, and feature scaling, ensuring compatibility with an LSTM-based fault prediction model. The LSTM model, trained with Adam optimization and categorical cross-entropy loss, achieves high accuracy, precision, recall, f1-score and ROC predicting failures, enabling proactive data replication. Additionally, reinforcement learning optimizes replica placement by dynamically adjusting replication factors based on criticality, failure probability, and resource constraints. A latency monitoring mechanism further enhances system resilience by identifying performance anomalies and triggering failover mechanisms in real time. This combined approach ensures robust, adaptive, and efficient fault prevention in cloud storage systems. These whole processes are discussed below:

A. Data Collection and Visualization

The NSL-KDD dataset consists of 25,192 samples and 42 features, including categorical and numerical variables. An initial EDA reveals that there are no missing values in the dataset. Among the categorical features, protocol type has 66 unique values, service has 11 unique values, dst_host_srv_rerror_rate has 22 unique values, and duration has 3 unique values shown in Figures 2, 3 and 4. The dataset's numerical columns exhibit varying distributions, requiring standardization for better model performance. The target column Label originally contained 21 unique values, which were mapped into a binary classification format: 0 (12,697 instances) representing normal traffic and 1 (12,495 instances) representing attack traffic shown in Figure 5. This binary conversion balances the dataset for enhanced model learning and ensures effective classification.

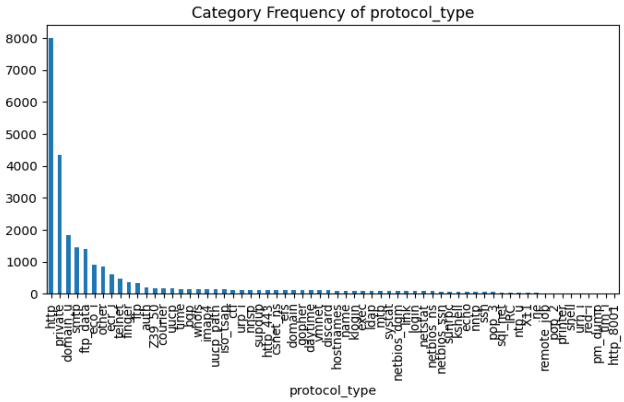

Fig. 2. Category frequency of protocol_type

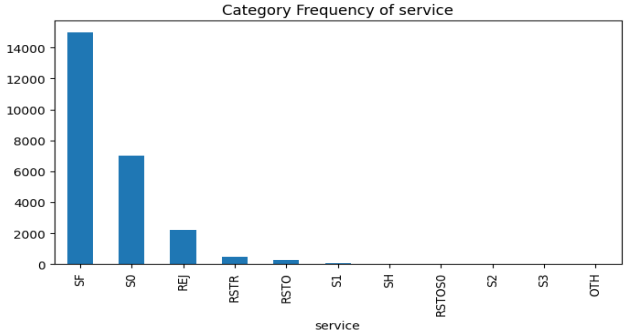

Fig. 3. Category frequency of service

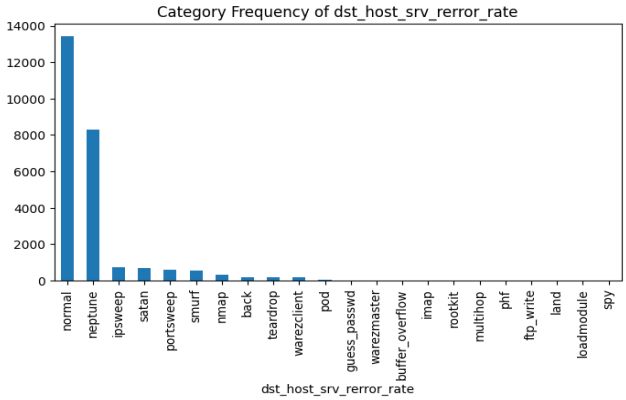

Fig. 4. Category frequency of dst_host_srv_rerror_rate

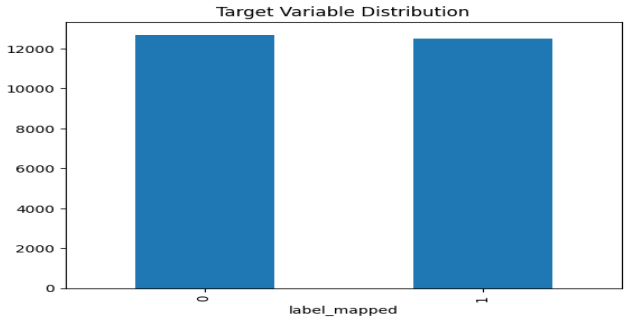

Fig. 5. Bar graph for target variable distribution

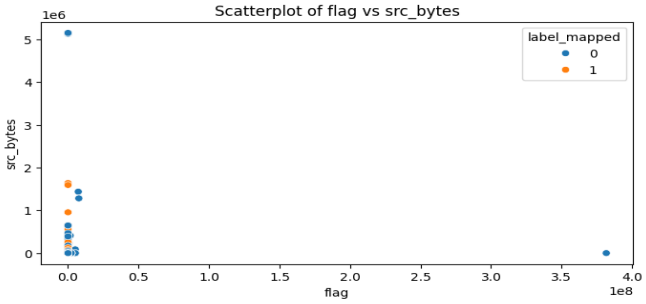

Fig. 6. Scatter plot for label mapped

Figure 6 presents a scatter plot visualizing the relationship between the 'flag' and 'src_bytes' variables, differentiated by the 'label mapped' categories (0 and 1). The plot reveals a potential correlation between 'flag' and 'src_bytes', suggesting that as 'flag' values increase, 'src_bytes' also tend to increase.

B. *Data Preprocessing*

The first step in putting ML defect avoidance into practice was data preparation. The quality of the dataset is improved in this step, which impacts the performance of fault prevention. The dataset was pre-processed as:

1) Feature and Label Separation

Splitting the dataset into features-containing independent variables (X) and an attack-label-containing target variable (Y) allowed for classification of traffic as either normal or malicious.

2) One-Hot Encoding for Categorical variables

One of the most popular ways to handle the neutralization of categorical characteristics is via one-hot encoding. Converts categorical columns in X into numeric form using one-hot encoding to ensure compatibility with the model.

3) Feature Scaler

In ML, a feature scaler is a tool for uniformly distributing features or independent variables across NSL KDD datasets [32]. Standardizes X to have a mean of 0 and a standard deviation of 1. Using Eq. (1), the normalization procedure was carried out. This procedure reduced all the numerical values to the integers from 0 to 1.

$$X' = X - \frac{\mu}{\sigma} \qquad (1)$$

The original feature value, X, and the normalized value, X′, are both represented in this equation. The standard deviation is denoted by σ, whereas the mean is represented by μ.

4) Label Conversion

Convert Labels into categorical format. The original multi-class attack labels were mapped into a binary classification (0: Normal, 1: Attack) to simplify detection, making it suitable for a binary classification task.

C. *Data Splitting*

An 80:20 split between the data sets should be used for training and testing purposes. A considerable percentage of the data is used to train the model, and its performance is suitably assessed on a separate test set.

D. *Proposed Long-Short-Term Memory (LSTM) Model*

An enhanced LSTMmodel is used in this research for cloud storage management for proactive fault prevention with improve data reliability on NSL-KDD dataset. LSTM, initially proposed by Hoch Reiter and Schmid Huber in 1997 and later refined by Alex Graves, has emerged as a powerful and widely adopted neural network architecture[33]. LSTM effectively addresses the challenge of long-term dependencies in sequential data by design, enabling it to retain and utilize information over extended time spans effortlessly[34]. There are two steps to a single neuron's computation in the LSTM repeat module: changing the network state and calculating the output

value [33]. The input gate, the forgetting gate, and the output gate are three circuits that make up a neurone. The gate function regulates the values of input, memory, and output.

The quantity of data that is now being deleted by the state of the neural network may be controlled by the forgetting gate. Here is the calculating method for the forgetting gate (2):

$$f_t = \sigma(W_f.[h_{t-1}\ x_t] + b_f \qquad (2)$$

where f_t represents the forgetting gate's output, h_{t-1} is the last-minute hidden condition, and the information fusion controls the fraction of forgotten information W_f , b_f , and sigmoid function of σ.

There are two components to the input gate: the original input value and the new input (3,4):

$$i_t = \sigma(W_i.[h_t, x_t] + \ b_i) \qquad (3)$$

$$\tilde{c}_t = \tanh\ (W_c.[h_{t-1}, x_t] + b_c) \qquad (4)$$

The input gate excludes data from the input layer when the tanh function's output falls within the range of -1 to 1. The calculation technique for the input gate is as follows (5):

$$C_t = f_t.C_{t-1} + i_t.C_t \qquad (5)$$

where the current state of the NN is represented [35], by C_t. Here is the procedure for calculating the hidden state and output gate (6,7):

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \qquad (6)$$

$$h_t = o_t.\tanh\ (C_t) \qquad (7)$$

where σ is a sigmoid activation function, $[h_{t-1}, x_t]$ concatenates the previous hidden state and the current input. Each gate has its own weight matrix (W_o) and bias vector (b_o). C_t represent the cell state of each LSTM cell unit.

E. Model Training

The LSTM model for the NSL-KDD dataset consists of two LSTMlayers: the first with 64units returning sequences for stacking, followed by a dropout layer (20%) to prevent overfitting. The second LSTMlayer has 32 units and outputs a single vector, followed by another dropout layer (30%). The softmax activation function is used for binary classification in a fully linked dense layer that has two output neurones. For adaptive learning rate control, the Adam optimizer is used to optimize the model, with categorical cross-entropy serving as the loss function. The ROC curve, confusion matrices, loss trends, and accuracy are used to assess performance.

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 64)	52,224
dropout (Dropout)	(None, 1, 64)	0
lstm_1 (LSTM)	(None, 32)	12,416
dropout_1 (Dropout)	(None, 32)	0
dense (Dense)	(None, 2)	66

Total params: 64,706 (252.76 KB)
Trainable params: 64,706 (252.76 KB)
Non-trainable params: 0 (0.00 B)

Fig. 7. Summary of LSTM Model

Figure 7 displayed the overview of a multi-layer sequential neural network model that includes LSTM, Dropout, and Dense layers. An LSTM layer is the first part of the model. It has 52,224 trainable parameters and an output shape of (None, 1, 64). Next, there is a Dropout layer that lacks parameters but has the same output form. Next, another LSTM layer is present, which reduces the output shape to (None, 32) and has 12,416 parameters. The output shape is preserved by an additional parameter-free Dropout layer that follows. An output shape of (None, 2) is achieved via a Dense layer, which contains 66 trainable parameters. With the exception of a few non-trainable parameters, all 64,706 of the model's parameters are trainable. The model's total size is 252.76 KB.

F. Model Evaluation

The experimental outcomes that were achieved by using the LSTM architecture are the main emphasis of this section. Five criteria for classification evaluation—accuracy/loss, recall, precision, and F1 score—along with ROC curve, classification report, and conclusion matrix—were chosen via fault prevention studies. The confusion matrix has the following values:

- **True Positive (TP):** Count of samples when the expected and actual labels are both positive.
- **True Negative (TN):** The sum of all samples for which the expected and real labels were negative.
- **False Positive (FP):** Count of samples when the anticipated value is positive and the actual value is negative.
- **False Negative (FN):** The total number of samples that had positive labels but negative projected labels.

Accuracy (Acc): To measure accuracy, take a ratio of a number of occurrences in the dataset to the number of properly predicted values (TP or TN). The model is doing a good job if its accuracy is high. It is measured as (8):

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (8)$$

Precision (PRE): The percentage of samples that were really positive out of all the samples that were projected to be positive by the model is represented by PRE. It is measured as (9):

$$Precision = \frac{TP}{TP+FP} \quad (9)$$

Recall (RE): REC is a measure of how well the model predicted the real positive samples relative to all positive samples. It is measure as (10):

$$Recall = \frac{TP}{TP+FN} \quad (10)$$

F1-score (F1): F1 is an all-encompassing statistic that finds the weighted harmonic average of the two metrics, Precision and Recall. It is measure as (11):

$$F1 - Score = 2 \times \frac{Precision*Recall}{Precision+Recall} \quad (11)$$

Loss: Measures error between predicted and actual values (model-specific).

Receiver operating characteristic (ROC) curve: The curve of acceptance. Distinct judgement criteria lead to distinct points on the curve, which indicate reactions to the same signal source. The X-axis points and the Y-axis points of the curve represent FPR and TPR, respectively. A better model's performance is often indicated by a ROC curve that is closer to the top-left corner.

G. Reinforce learning

The reinforcement learning (RL) approach learns replica placement from system feedback dynamically to optimize them in a cloud storage network. Then an RL agent is trained to place replicas such that placed replicas result in key metrics being shown, e.g. network topology, historical access patterns, node reliability, and resource utilization. This allows making the reward function that penalizes undesirable states to produce optimal replica distribution. Replication factor starts at value of 3 and is adjusted depending on data criticality, failure probability, and available resources to achieve reliability balance between reliability and system constraint. It provides efficient fault tolerance minimizes the data loss, and improves in general cloud storage reliability.

IV. Result Analysis And Discussion

In this section provide the experimental results of LSTM model and Reinforce learning for enhancing data reliability in cloud storage management with proactive fault prevention on NSL-KDD dataset. The tests conducted in this paper were coded in Python 3. The experiment was carried out using a desktop computer that has a 64-bit version of Windows 10, 4GB of RAM, and an Intel core i3 - 2.00 GHz processor. The following Table II provides the efficiency of the LSTM model Performance.

TABLE II. RESULTS OF LSTM MODEL FOR DATA RELIABILITY IN CLOUD STORAGE MANAGEMENT ON NSL-KDD DATA

Performance Matrix	**Long-Short-term Memory (LSTM)**	
	Train	***Test***
Accuracy	93.87	92.92
Loss	17.14	21.19

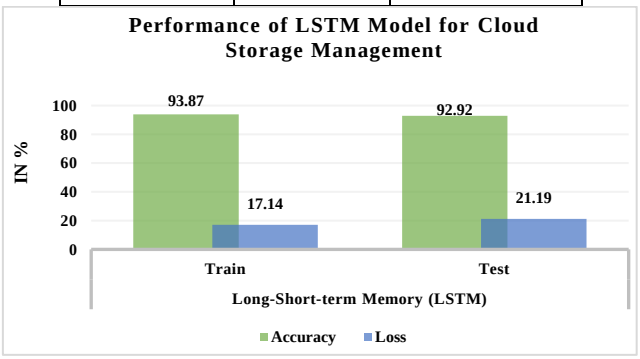

Fig. 8. Bar graph for LSTM Model Performance

Results from an LSTM model training on the NSL-KDD dataset for data dependability in cloud storage management are shown in Table II and Figure 8. The model's training accuracy was 93.87% and its testing accuracy was 92.92%, as shown in the table. There is a loss of 17.14 in training and a loss of 21.19 in testing, as shown in the table respectively. The LSTM model demonstrates effectiveness in predicting data reliability in cloud storage management because it achieves high accuracy in both training and testing data.

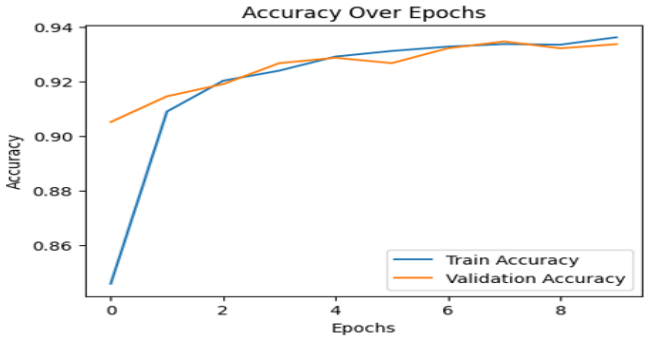

Fig. 9. Accuracy Curve for LSTM Model

Figure 9 displays an accuracy curve for an LSTM model in proactive fault prevention. The training (blue) and validation (orange) accuracies steadily rise, plateauing at 93.87% and 92.92%. Their close convergence suggests good generalization without overfitting, highlighting the model's effectiveness.

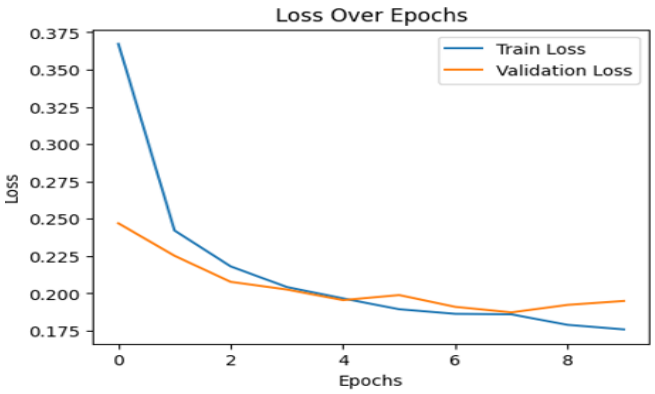

Fig. 10. Loss Curve for LSTM Model

Figure 10 displayed a loss curve for an LSTM model in proactive fault prevention. Training (blue) and validation (orange) losses drop rapidly, stabilizing at 17.14% and 21.19%. Their close alignment suggests effective learning, minimal overfitting, and strong generalization to unseen data.

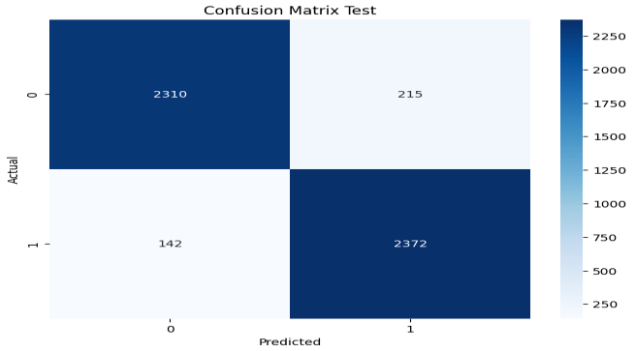

Fig. 11. Confusion matrix for LSTM Model

Figure 11 displays the confusion matrix of the LSTMmodel on the NSL-KDDdatasets. With 215 FP and 142 FN, the model managed 2372 TP and 2310 TN. High true values and low misclassifications indicate strong performance, allowing further evaluation through precision, recall, and F1-score.

Classification Report (Test Data):

	precision	recall	f1-score	support
0	0.94	0.91	0.93	2525
1	0.92	0.94	0.93	2514
accuracy			0.93	5039
macro avg	0.93	0.93	0.93	5039
weighted avg	0.93	0.93	0.93	5039

Fig. 12. Classification matrix for LSTM Model

Figure 12 presents the classification report for an LSTM model's performance on an NSL-KDD dataset, providing key evaluation metrics for proactive fault prevention. There is support for both classes 0 and 1 as well as precision, recall, and F1-score in the report. For class 0, Pre is 0.94, the Re is 0.91, and the F1 is 0.93. For class 1, the Pre is 0.92, Re is 0.94, and F1 is 0.93. The overall Acc of the model is 0.93. Both macro and weighted averages for Pre, Re, and F1 are also 0.93 with support 5039, reflecting the balanced performance across both classes.

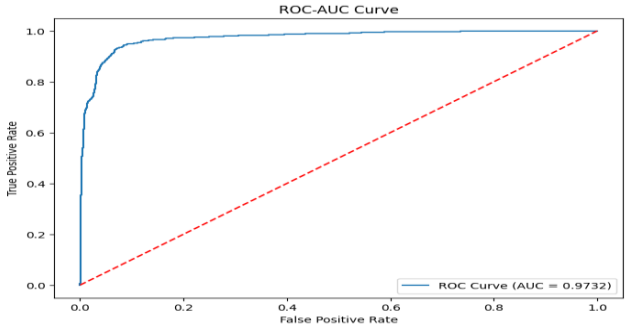

Fig. 13. ROC Curve for LSTM Model

Figure 13 displays the ROCcurve for the LSTMmodel on the NSL-KDD dataset, showing its ability to distinguish between classes. With an AUC of 0.9732, the model demonstrates high separability, indicating strong performance in proactive fault detection.

A. Results of Reinforce Learning

A reinforcement learning (RL) environment was developed to optimize replica placement within a storage network.

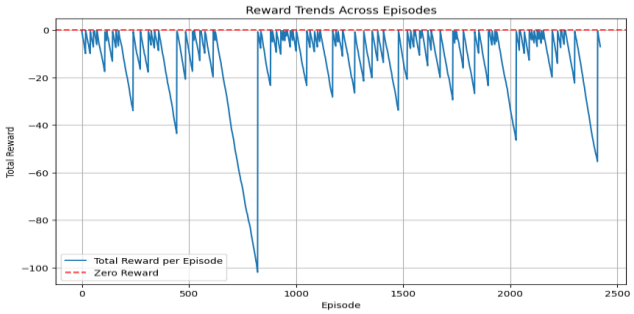

Fig. 14. Total Rewards across episodes

Figure 14 illustrates total reward trends across 2500 episodes, showing initial fluctuations with frequent negative rewards due to the agent's early exploration phase. As training progresses, rewards improve, frequently approaching the zero-reward line, indicating better decision-making. However, some episodes still show steep drops, with rewards occasionally falling below -100, reflecting the impact of unpredictable state transitions and learning variability. The wide range of total rewards across episodes (from values close to 0 to significantly negative ones) reflects the randomness in the initial states, transitions, and episode terminations.

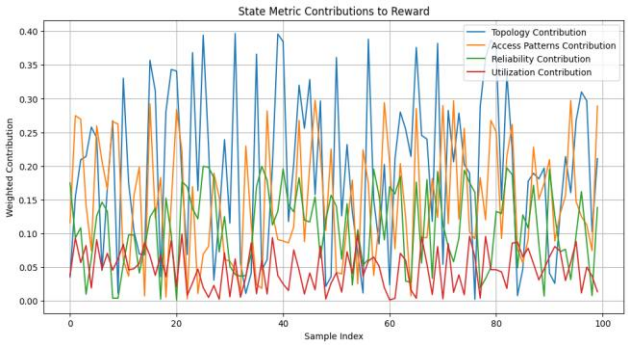

Fig. 15. Static Metric contribution

Figure 15 illustrates the contributions of state metrics—Topology, Access Patterns, Reliability, and Utilization—to the reward function across 100 samples. Topology and Access Patterns show the highest fluctuations, often exceeding 0.35 in weighted contribution, indicating their dominant influence on decision-making. Reliability and Utilization contribute less, generally remaining below 0.15, suggesting secondary

importance. The variability highlights the dynamic nature of the system, requiring adaptive strategies for optimal performance.

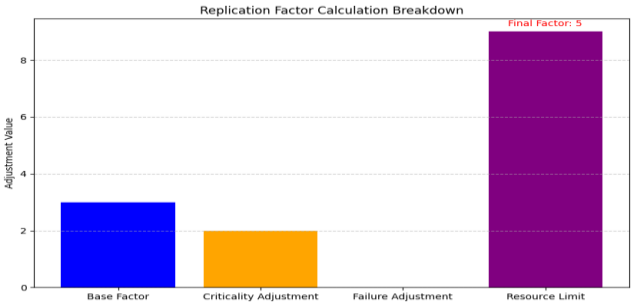

Fig. 16. Calculation of replication factor

The Replication Factor Calculation Breakdown is in the Figure 16 chart. The number of replicas for a resource based on its base factor, criticality, failure probability, and resource availability. Starting with a base of 3 replicas, it adds or subtracts replicas depending on criticality (+2 for high, 0 for medium, and -1 for low). It then adjusts downward by twice the failure probability and ensures the factor is at least 1. Finally, it limits the factor to a maximum value proportional to resource availability (resource availability * 10). For the input provided (criticality='high', health=0.8, failure_prob=0.1, resource availability=0.9), the function calculates a replication factor of 5 by starting with 5 (3 + 2 for high criticality), adjusting minimally for failure probability (-0), and keeping it within the resource limit (9).

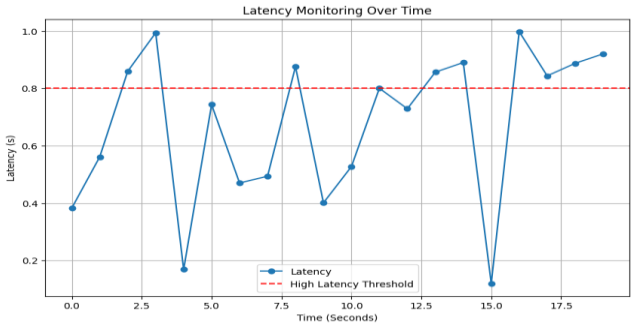

Fig. 17. Latency Monitoring over time

Figure 17 illustrates latency monitoring over time, capturing fluctuations in response times over 20 seconds. Real-time latency monitoring to support performance evaluation in a system with failover mechanisms. A list of 20 random latency values is generated to represent system response times, and a high latency threshold is set at 0.8 seconds to flag potential performance issues. A line plot is created to show latency trends over time, with a dashed red line marking the threshold for high latency. The latency varies between approximately 0.1s and 1.0s, with multiple spikes exceeding the high-latency threshold of 0.8s, particularly around 2.6s, 7.6s, 13s, and 17s. Periodic dips below 0.2s indicate instances of optimal performance, but overall, the system experiences unstable latency, requiring optimization strategies for improved reliability.

B. Discussion

A comparison of traditional and ML techniques for proactive fault prevention using the NSL-KDD dataset highlights the superiority of deep learning models in ensuring data reliability shown in Table III. LSTM achieves the highest accuracy of 92.92%, with precision, recall, and F1-score of 93%, demonstrating its ability to effectively capture sequential patterns in network traffic. KNN follows with 91% accuracy and a recall of 93%, but its lower precision of 81% suggests susceptibility to false positives. AdaBoost achieves 89% across all metrics, indicating balanced performance but slightly lower reliability than LSTM. Convolutional Neural Networks (CNN) exhibit moderate performance with 79.75% accuracy and 86.46% precision but a significantly low recall of 56.96%, suggesting difficulty in identifying all attack patterns. Artificial Neural Networks (ANN) attain 78% accuracy with a high 96% precision, yet its recall of 62.05% indicates a trade-off favouring fewer false alarms at the cost of

missing threats. Deep Neural Networks (DNN) display the lowest accuracy of 75.75%, with a precision of 83% and recall of 76%, reflecting moderate but inconsistent results. Overall, LSTM stands out as the most effective model for proactive fault prevention, ensuring superior data reliability in cloud storage management.

TABLE III. COMPARING TRADITIONAL AND MACHINE LEARNING TECHNIQUES FOR PROACTIVE FAULT PREVENTION ON NSL-KDD

Models	**Accuracy**	**Precision**	**Recall**	**F1-score**
LSTM	92.92	93	93	93
KNN[36]	91	81	93	91
AdaBoost [37]	89	89	89	89
CNN [38]	79.75	86.46	56.96	59.71
ANN[39]	78	96	62.05	75.57
DNN[40]	75.75	83	76	75

This study presents a significant advancement in cloud storage reliability by integrating LSTM-based failure prediction with reinforcement learning-driven replica placement. The LSTM model effectively captures sequential patterns in network traffic, achieving high accuracy (92.92%) while minimizing false positives and overfitting in compare to other existing CNN, KNN, ANN, and DNN. The RL-based approach dynamically optimizes replica distribution based on system conditions, enhancing fault tolerance and resource efficiency. Additionally, real-time latency monitoring ensures proactive performance management, reducing downtime and improving data availability. Compared to traditional static replication methods, this intelligent framework offers adaptive, scalable, and efficient cloud storage management, making it highly suitable for modern data-driven applications.

V. CONCLUSION AND FUTURE WORK

This study demonstrates the superiority of the LSTM model in proactive fault prevention for cloud storage management using the NSL-KDD dataset. The LSTM model outperforms KNN alongside AdaBoost, CNN, ANN and DNN through its highest accuracy of 92.92% while maintaining equivalent precision, recall and F1-score at 93%. Unlike static fault-tolerance techniques, this approach integrates reinforcement learning (RL)-based replica placement to optimize redundancy dynamically, reducing resource wastage while ensuring data reliability. Real-time latency monitoring functions together with this method to preserve system performance by minimizing disruptions, which proves the method is a reliable adaptive solution for cloud storage reliability. Their LSTM system delivers excellent results but needs more training time with extensive datasets, plus requires substantial technical resources for its operation. Real-time deployment needs specific modifications when operating with changing cloud settings. The future direction of their work will be model pruning and quantization to improve efficiency, federated learning for decentralizing the training, as well as to enhance adaptability to changing attack patterns using the self-learning AI capabilities. Moreover, expanding the dataset and incorporating additional cloud storage parameters will further refine the model's reliability and robustness.

REFERENCES

[1] M. S. Rajeev Arora, “Applications of Cloud Based ERP Application and how to address Security and Data Privacy Issues in Cloud application,” *Himal. Univ.*, 2022.

[2] N. Patel, “Secure Access Service Edge(Sase): Evaluating The Impact Of Convereged Network Security Architectures In Cloud Computing,” *J. Emerg. Technol. Innov. Res.*, vol. 11, no. 3, pp. e703–e714, 2024.

[3] A. and P. Khare, “Cloud Security Challenges : Implementing Best Practices for Secure SaaS Application Development,” *Int. J. Curr. Eng. Technol.*, vol. 11, no. 6, pp. 669–676, 2021, doi: https://doi.org/10.14741/ijcet/v.11.6.11.

[4] T. K. K. and S. Rongala, “Implementing AI-Driven Secure Cloud Data Pipelines in Azure with Databricks,” *Nanotechnol. Perceptions*, vol. 20, no. 15, pp. 3063–3075, 2024, doi: https://doi.org/10.62441/nano-ntp.vi.4439.

[5] P. Kumari and P. Kaur, "A survey of fault tolerance in cloud computing," *Journal of King Saud University - Computer and Information Sciences*. 2021. doi: 10.1016/j.jksuci.2018.09.021.

[6] K. Rajchandar, M. Ramesh, A. Tyagi, S. Prabhu, D. S. Babu, and A. Roniboss, "Edge Computing in Network-based Systems: Enhancing Latency-Sensitive Applications," in *2024 7th International Conference on Contemporary Computing and Informatics (IC3I)*, 2024, pp. 462–467. doi: 10.1109/IC3I61595.2024.10828607.

[7] L. He, Z. Qian, and F. Shang, "A novel predicted replication strategy in cloud storage," *J. Supercomput.*, 2020, doi: 10.1007/s11227-018-2647-4.

[8] R. Arora, S. Gera, and M. Saxena, "Mitigating security risks on privacy of sensitive data used in cloud-based ERP applications," in *Proceedings of the 2021 8th International Conference on Computing for Sustainable Global Development, INDIACom 2021*, 2021. doi: 10.1109/INDIACom51348.2021.00081.

[9] S. Murri, S. Chinta, S. Jain, and T. Adimulam, "Advancing Cloud Data Architectures: A Deep Dive into Scalability, Security, and Intelligent Data Management for Next-Generation Applications," *Well Test. J.*, vol. 33, no. 2, pp. 619–644, 2024.

[10] B. Boddu, "Serverless Databases Are the Future of Database Management," *https://jsaer.com/download/vol-6-iss-1-2019/JSAER2019-6-1-277-282.pdf*, vol. 6, no. 1, p. 5, 2020.

[11] A. Alzahrani, T. Alyas, K. Alissa, Q. Abbas, Y. Alsaawy, and N. Tabassum, "Hybrid Approach for Improving the Performance of Data Reliability in Cloud Storage Management," *Sensors*, vol. 22, no. 16, 2022, doi: 10.3390/s22165966.

[12] M. Shah, P. Shah, and S. Patil, "Secure and Efficient Fraud Detection Using Federated Learning and Distributed Search Databases," in *2025 IEEE 4th International Conference on AI in Cybersecurity (ICAIC)*, 2025, pp. 1–6. doi: 10.1109/ICAIC63015.2025.10849280.

[13] Suhag Pandya, "Advanced Blockchain-Based Framework for Enhancing Security, Transparency, and Integrity in Decentralised Voting System," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 2, no. 1, pp. 865–876, Aug. 2022, doi: 10.48175/IJARSCT-12467H.

[14] S. R. Thota, S. Arora, and S. Gupta, "Hybrid Machine Learning Models for Predictive Maintenance in Cloud-Based Infrastructure for SaaS Applications," 2024, pp. 1–6. doi: 10.1109/ICDSNS62112.2024.10691295.

[15] T. K. Bui, C. H. Tran, and T. V. Pham, "V2PFQL: A proactive fault tolerance approach for cloud-hosted applications in cloud computing environment," *IET Control Theory Appl.*, 2022, doi: 10.1049/cth2.12324.

[16] V. N. Boddapati *et al.*, "Data migration in the cloud database: A review of vendor solutions and challenges," *Int. J. Comput. Artif. Intell.*, vol. 3, no. 2, pp. 96–101, Jul. 2022, doi: 10.33545/27076571.2022.v3.i2a.110.

[17] L. Zhu, Q. Zhuang, H. Jiang, H. Liang, X. Gao, and W. Wang, "Reliability-aware failure recovery for cloud computing based automatic train supervision systems in urban rail transit using deep reinforcement learning," *J. Cloud Comput.*, 2023, doi: 10.1186/s13677-023-00502-x.

[18] V. S. Thokala, "A Comparative Study of Data Integrity and Redundancy in Distributed Databases for Web Applications," *Int. J. Res. Anal. Rev.*, vol. 8, no. 4, pp. 383–389, 2021.

[19] S. Singh, "Enhancing Observability and Reliability in Wireless Networks with Service Mesh Technologies," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 5, no. 1, 2025, doi: 10.48175/568.

[20] M. Gopalsamy, "Scalable Anomaly Detection Frameworks for Network Traffic Analysis in cybersecurity using Machine Learning Approaches," *Int. J. Curr. Eng. Technol.*, vol. 12, no. 06, pp. 549–556, 2022, doi: : https://doi.org/10.14741/ijcet/v.12.6.9.

[21] N. Abid, "Empowering Cybersecurity : Optimized Network Intrusion Detection Using Data Balancing and Advanced Machine Learning Models," *TIJER*, vol. 11, no. 12, 2024.

[22] Y. Tian, J. Tian, and N. Li, "Cloud reliability and efficiency improvement via failure risk based proactive actions," *J. Syst. Softw.*, vol. 163, p. 110524, 2020, doi: https://doi.org/10.1016/j.jss.2020.110524.

[23] R. Arora, A. Kumar, and A. Soni, "AI-Driven Self-Healing Cloud Systems: Enhancing Reliability and Reducing Downtime through Event- Driven Automation." 2024.

[24] M. S. Samarth Shah, "Deep Reinforcement Learning For Scalable Task Scheduling In Serverless Computing," *Int. Res. J. Mod. Eng. Technol. Sci.*, vol. 3, no. 12, pp. 1845–1852, 2021, doi: DOI : https://www.doi.org/10.56726/IRJMETS17782.

[25] R. Tarafdar, "Algorithms on Majority Problem," *Univ. Missouri-Kansas City*, 2017.

[26] P. S. N. Patel, D. Parikh, R. K. Eranna, J. Patel, and P. Siddhapura, "Machine Learning Based Security Device For Cloud Computing," 418563–001, 2024

[27] C. Wen, G. Liu, F. Ding, Z. Li, and H. Lin, "Research on Data Reliability and Storage Security Verification for Cloud Platform Based on Machine Learning," in *Third International Conference on Electronic Information Engineering, Big Data, and Computer Technology (EIBDCT 2024)*, J. Zhang and N. Sun, Eds., SPIE, Jul. 2024, p. 345. doi: 10.1117/12.3031396.

[28] S. Kumar, "Enhancing Cloud Storage Efficiency and Accessibility with Artificial Intelligence: A Comprehensive Review," *Int. J. Adv. Res. Eng. Technol.*, vol. 15, pp. 183–196, 2024, doi: 10.17605/OSF.IO/9DC2J.

[29] M. A. Shahid, M. M. Alam, and M. M. Su'ud, "Achieving Reliability in Cloud Computing by a Novel Hybrid Approach," *Sensors*, 2023, doi: 10.3390/s23041965.

[30] T. N. Tengku Asmawi, A. Ismail, and J. Shen, "Cloud failure prediction based on traditional machine learning and deep learning," *J. Cloud Comput.*, 2022, doi: 10.1186/s13677-022-00327-0.

[31] M. Uppal *et al.*, "Cloud-Based Fault Prediction for Real-Time Monitoring of Sensor Data in Hospital Environment Using Machine Learning," *Sustain.*, 2022, doi: 10.3390/su141811667.

[32] Mani Gopalsamy, "Enhanced Cybersecurity for Network Intrusion Detection System Based Artificial Intelligence (AI) Techniques," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 12, no. 01, pp. 671–681, Dec. 2021, doi: 10.48175/IJARSCT-2269M.

[33] K. Ullah *et al.*, "Short-Term Load Forecasting: A Comprehensive Review and Simulation Study with CNN-LSTM Hybrids Approach," *IEEE Access*, vol. 12, no. July, pp. 111858–111881, 2024, doi: 10.1109/ACCESS.2024.3440631.

[34] H. Sinha, "The Identification of Network Intrusions with Generative Artificial Intelligence Approach for Cybersecurity," *J. Web Appl. Cyber Secur.*, vol. 2, no. 2, pp. 20–29, Oct. 2024, doi: 10.48001/jowacs.2024.2220-29.

[35] M. K. A Arif, A Khan, "Role of AI in Predicting and Mitigating Threats: A Comprehensive Review," *JURIHUM J. Inov. dan Hum.*, vol. 2, no. 3, pp. 297–311, 2024.

[36] B. S. Sukhadeo, R. N. Patil, R. Atole, Y. D. Sinkar, U. C. Patkar, and R. Chopade, "MLIDS: A Machine Learning-Based Intrusion Detection System Using the NSLKDD Data," *Int. J. Intell. Syst. Appl. Eng.*, 2024.

[37] A. John, I. F. Bin Isnin, S. H. H. Madni, and F. B. Muchtar, "Enhanced intrusion detection model based on principal component analysis and variable ensemble machine learning algorithm," *Intell. Syst. with Appl.*, vol. 24, p. 200442, 2024, doi: https://doi.org/10.1016/j.iswa.2024.200442.

[38] Y. Yang, Y. Gu, and Y. Yan, "Machine Learning-Based Intrusion Detection for Rare-Class Network Attacks," *Electron.*, 2023, doi: 10.3390/electronics12183911.

[39] S. Sapre, P. Ahmadi, and K. Islam, "A Robust Comparison of the KDDCup99 and NSL-KDD IoT Network Intrusion Detection Datasets Through Various Machine Learning Algorithms," 2019.

[40] M. Eshak Magdy, A. M. MATTER, S. HUSSIN, D. HASSAN, and S. Elsaid, "A Comparative study of intrusion detection systems applied to NSL-KDD Dataset," *Egypt. Int. J. Eng. Sci. Technol.*, 2022, doi: 10.21608/eijest.2022.137441.1156.