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

Adaptive Engineering Systems: Redundancy-Driven Optimization And System Reliability

Arif Ali[1], Antoni Biya[2], Amit Gaurav[3], Jyotsna Suryavanshi[4], Leena Bharat Chaudhari[5], A. A. R. Senthil Kumaar[6], Harshini R[7], Pooja Sharma[8]

[1] Assistant Professor, Department of Computer Application, Dev Bhoomi Uttarakhand University, Dehradun, Uttarakhand, India. Email: socse.arif@dbuu.ac.in ORCID: 0009-0000-9025-7368
[2] Assistant Professor, Department of Mathematics, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: antonibiya@maher.ac.in
[3] School of Engineering & Technology, Noida International University, Greater Noida, Uttar Pradesh 203201, India. Email: coe@niu.edu.in
[4] Department of Engineering, Science and Humanities, Vishwakarma Institute of Technology, Pune, Maharashtra 411037, India. Email: jyotsna.suryavanshi@vit.edu
[5] Department of Electronics and Telecommunication Engineering, Bharati Vidyapeeth's College of Engineering, Lavale, Pune, Maharashtra, India. Email: leenapc23@gmail.com
[6] Assistant Professor, Department of Computer Science Engineering, Koneru Lakshmaiah Education Foundation, Andhra Pradesh, India. Email: aarsenthil@kluniversity.in
[7] 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
[8] Centre for Research Impact & Outcome, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab 140401, India. Email: pooja.sharma.orp@chitkara.edu.in ORCID: 0009-0009-7750-9896

Abstract
Deep learning (DL) models are widely used in applications such as image classification, speech recognition, and autonomous systems, but their deployment is limited by high computational cost and sensitivity to hardware faults that reduce system dependability. Traditional fault tolerance methods rely on hardware redundancy, increasing system complexity and resource usage, making scalability difficult. This research proposes an adaptive engineering framework based on redundancy optimization to address these limitations. Data is preprocessed using Z-score normalization for standardization and autoencoder-based feature extraction to obtain compact representations. The proposed FAT-DRNN (Fault-Aware Training-tuned Dynamic Recurrent Neural Network) enhances robustness under simulated hardware faults. It integrates Fault-Aware Training to improve reliability through redundancy-based learning, while DRNN enables temporal modeling and adaptive system dynamics. Fault injection is applied within convolutional layers to make the network resilient without relying on external redundancy. The Python implementation optimizes learning-based redundancy instead of hardware redundancy using adaptive techniques. Results show improved fault tolerance and worst-case accuracy compared to conventional networks, with reduced redundancy. The model achieves 9.6 ms execution time, 97.1% accuracy, 98.2% precision, 97.8% recall, and 98.0% F1-score. Overall, the proposed FAT-DRNN framework demonstrates that integrating fault-aware training with recurrent architectures can significantly improve robustness and efficiency in deep learning systems deployed in unreliable hardware environments while minimizing dependence on external redundancy The approach provides a scalable solution for resilient intelligent systems balancing computational efficiency accuracy and fault tolerance across critical applications requiring dependable performance under real-world operational conditions in practical industrial and autonomous system deployments scenarios use.

Keywords: Adaptive engineering systems, System reliability, Redundancy optimization, Fault-aware training, Fault tolerance, Hardware fault resilience

1. Introduction

In today's advanced technological environment, adaptive engineering systems have become indispensable because of their capability to guarantee high levels of performance in changing and unpredictable situations. Adaptive engineering systems are used in different applications ranging from autonomous vehicles, industrial automation, aerospace systems, smart grids, to intelligent manufacturing. The primary feature of adaptive engineering systems is their ability to adapt to environmental changes, part failure, and workload change while continuing to operate properly [1]. With the increasing complexity of engineering systems, keeping high levels of reliability becomes difficult. Failure of engineering systems can result in negative effects in the form of safety hazards and economic losses. Redundancy optimization emerges as one of the solutions to enhance reliability. Redundancy optimization refers to adding extra components or other alternatives that ensure continued operation of a system despite system faults or disturbances [2]. Traditional methods of implementing redundancy are associated with significant limitations, especially regarding hardware approaches, which include higher energy consumption, computational cost, latency time, and high costs [3]. However, the problems become more difficult when it comes to implementing large-scale systems and/or in resource-constrained systems. Recent progress in the realm of artificial intelligence and DL techniques has allowed the creation of intelligent redundancy strategies. DL algorithms have shown great promise in fault detection, prediction, analysis, and system optimization [4]. Despite all these strengths, one of the major barriers to the utilization of these methods lies in their high computational costs and vulnerability to hardware malfunctions, which lead to the deterioration of prediction and system performance [5]. Thus, the combination of learning techniques with intelligent redundancy management can help in solving these problems [6]. This way, the system able to optimize its redundancy level, adapt to uncertainty, achieve higher reliability and lower operating costs.

Research Aim: The proposed research develops an adaptive engineering system which increases reliability through fault tolerance via redundancy optimization based on FAT-DRNN model. As opposed to conventional redundancy schemes, this model integrates the concept of fault tolerance within the learning algorithm itself. This model learns from failures through the creation of faulty conditions and finding ways to solve them. By injecting faults, it improves its capabilities of performing well under different conditions of failure.

Research Organization: The structure of the paper includes the following: Section 1 discusses adaptive engineering systems, literature background, importance and difficulties associated with hardware faults. Section 2 provides the review of fault tolerance techniques and their shortcomings. Section 3 discusses the development of the proposed FAT-DRNN-based technique, with its architecture, fault injection approach and fault-tolerant training. Section 4 provides information on experiments results and conclusion presented in Section 5.

2. Related Works

Engineers' latest efforts to increase reliability are displayed in Table 1. They employ techniques such as data fusion, Markov models, Monte Carlo, MILP, and MLOps, to increase dependability, reduce costs, and correct errors. Big problems like very difficult computations, demanding optimizations, and not enough real-world tests, as well as being highly susceptible to model assumptions and data quality, still exists.

Table 1: Summary of Reliability Research Contributions

Ref	Objective	Method	Result	Limitations
[9]	Improve reliability in smart energy systems	Bi-level Mixed Integer Linear Programming (MILP) decentralized energy hub model	Improved reliability and reduced energy losses	Nonconvexity and computational complexity in optimization
[10]	Self-healing, reliable enterprise AI infrastructure	Monitoring diagnosis, an automated Machine Learning Operations (MLOps) framework	Improved reliability reduced recovery time	Complex failures are not fully handled

[11]	Reduce the cost of the drone swarm maintenance strategy.	Multi-layer network reliability optimization model	Reduced cost improved mission reliability	Limited real-world swarm validation scenarios
[12]	Improve planetary gear system reliability design	Hierarchical finite element fatigue model	Hierarchical finite element fatigue model	High computational complexity and test dependency
[13]	Improve multi-voltage distribution reliability evaluation	Monte Carlo and Failure Mode and Effects Analysis (FMEA) based evaluation	Improved reliability assessment accuracy demonstrated.	High computational time and complexity
[14]	Optimize the reliability of distributed heating systems	Markov process and nodal probability model	Improved heating reliability with cost optimization	Complex modeling and parameter estimation issues
[15]	Improve robust information fusion topology design.	Data-driven possibilistic fusion with redundancy analysis	Improved robustness under epistemic uncertainty	Performance is sensitive to missing training data
[16]	Assess the fault resilience of Deep Neural Network (DNN) models on hardware accelerators.	Architecture-Level Pre-Register-Transfer-Level Implementation Fault Injection (ALPRI-FI) framework	Enables early-stage evaluation of fault resilience, reducing design cost and improving reliability insights	not capture all low-level hardware faults, and can involve high evaluation complexity

2.1 Research Gap: Current solutions like MILP-based optimization schemes [9] face the challenge of complexity and nonconvexity, while MLOps-based self-healing frameworks [10] cannot cope with complex and hidden faults. Furthermore, ALPRI-FI [16] does not necessarily incorporate real-world fault dynamics and imposes computational overhead on the assessment of DNN accelerators during the initial stages. All these solutions lack scalability, adaptivity, and fault-awareness in their learning algorithms. Therefore, the need arises for an adaptive redundancy scheme that can integrate learning with fault awareness.

3. Methodology

The whole process of the Fault-Aware Training-tuned Dynamic Recurrent Neural Network (FAT-DRNN) for predicting system reliability, were explained in this research methodology section. The methodology starts with the Adaptive System Reliability Dataset, then applies Z-score normalization to prepare the data. An autoencoder cuts down dimensions and gets the essential features. Subsequently, these are used to train the FAT-DRNN algorithm to aid learning through erroneous scenarios, adaptive modeling, and increase robustness in different control systems, which is shown in Figure 1. They evaluate their efficiency based on accuracy, precision, recall, F1 score, and prediction time.

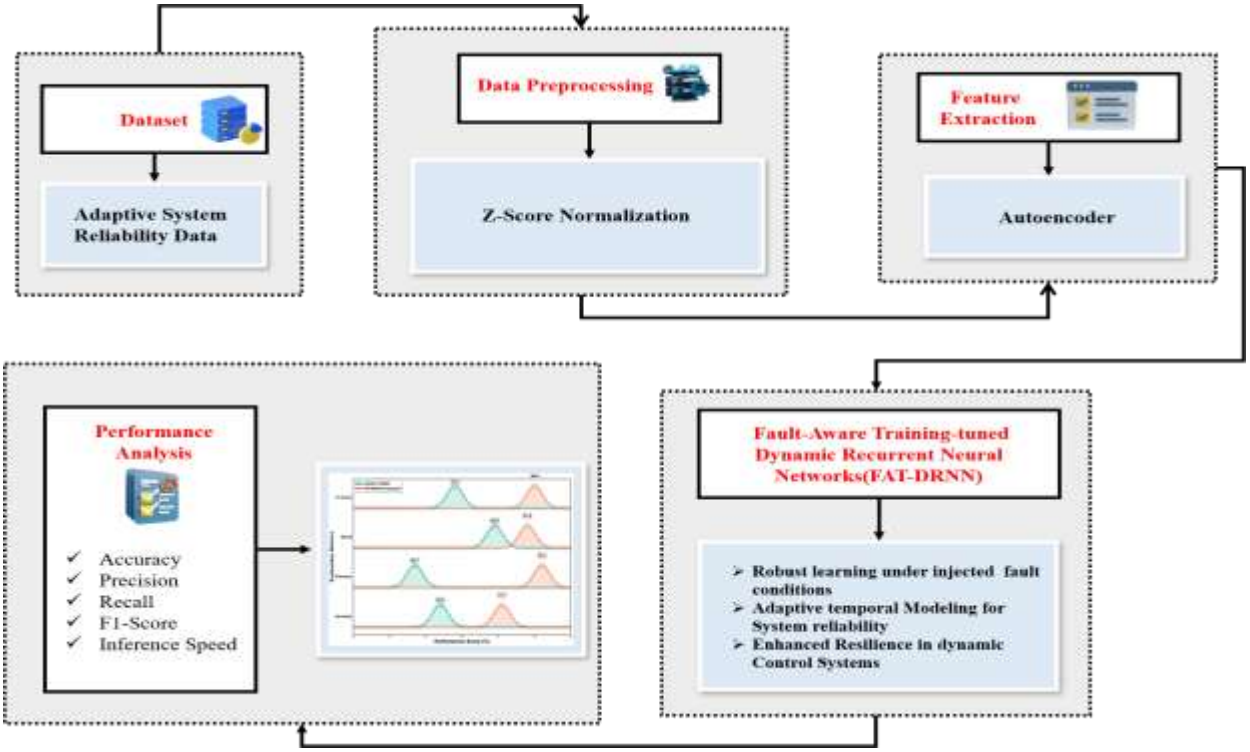

Figure 1: Fault-Aware Training Enhanced DRNN Architecture for Robust System Reliability Modeling

3.1 Data collection

The Adaptive System Reliability Dataset is useful for studies involving the application of redundancy and intelligent system error-handling capabilities in varying contexts. The database contains 10,000 samples corresponding to realistic autonomous and industrial situations. Each entry covers fault behavior, energy use, delay times, memory space, fix times, stability indexes, and overall reliability. This dataset is provided in CSV format for easy processing and analysis. Researchers analyze performance, fault tolerance, and reliability predictions across conditions, with 80% of data for training and 20% for evaluation.

Kaggle Source: https://www.kaggle.com/datasets/colabsss/adaptive-system-reliability-data

3.2 Data preprocessing using z-score normalization

The Z-score normalization scales data such that its mean is zero and standard deviation equals one, a technique quite useful in leveraging statistics during training. For adaptive systems, normalization ensures consistent scaling for different sensors and signals, thereby increasing accuracy and effectiveness in comparison and fusion. Normalization eliminates any impact of high-magnitude or noisy values that make the optimization process hard for the learning algorithms. This is particularly useful in maintaining reliable features in the presence of redundancy under varying conditions.

$$Z - Scored\ EMG_{s,c,t} = (EMG_{s,c,t} - \mu_{train,t})/\sigma_{train,t} \quad (1)$$

In Equation (1), $Z - Scored\ EMG_{s,c,t}$ is the normalized electromyography signal after applying a $Z - Scored$ transformation for subject s, channel c, at time t. $EMG_{s,c,t}$ show the raw EMG signal value. $\mu_{train,t}$ is the mean of the EMG signal in the training dataset at time t, acting as a reference point. $\sigma_{train,t}$ indicates how much the training EMG signals vary at time t.

3.3 Feature extraction using autoencoders

The optimization of adaptive engineering systems using an unsupervised DL model that involves the use of autoencoders. System-level observations are acquired from various cloud-based environments, including latency, bandwidth, and fault metrics. Heterogeneous observations are encoded in the hidden layer of the algorithm to minimize noise and redundancy. This leads to higher quality data, greater understanding of the

system at hand, and more reliable decision making, which translates to improved reliability and efficient utilization of system resources in actual application.

$$W = e_\theta(W) = t(XW + a_W) \quad (2)$$

In Equation (2), W stands for the output or learned feature representation post-transformation. The encoder $e_\theta(W)$ takes input and maps it to a new feature space. The sigmoid adds non-linearity through t. Here, X represents the input data with system metrics or features. W inside XW is a weight matrix that linearly transforms inputs, while $a_W$acts as a bias term adjusting those transformed values. So, $XW + a_W$performs the linear transformation of input, before activation.

$$W' = h'_\theta(Z) = t(X'Z + a_Z) \quad (3)$$

In Equation (3), the W' shows the decoded features or reconstructed output. The decoder function $h'_\theta(Z)$ with params θ' rebuilds data from the latent space of Z. t adds non-linearity. X' is what gets fed into the decoder, like compressed features. a condensed feature vector from the encoder $a_Z$is bias for the decoding transformation.

$$\theta = min\, K\,(W, W') = min\, K\,(W, h(\big(e(W)\big)) \quad (4)$$

Equation (4), θ, which includes weights and biases, gets optimized during training. min is to minimize the loss function, K, which measures how different the original data W is from the reconstructed data W'. This happens through an encoder function $e(W)$that transforms the input into a compressed form, and a decoder function $h(\big(e(W)\big)$ that tries to rebuild the data. When all together $min\, K\,(W, h(\big(e(W)\big)$ the complete autoencoder that does encoding then decoding.

$$K_1(\theta) = \sum_{j=1}^{m} \left\| w_j - w'_j \right\|^2 \quad (5)$$

In Equation (5), $K_1(\theta)$is the loss function based on parameters θ, showing reconstruction error. $\sum$ uses a sum to add up errors across all data samples. The index goes from j = 1 to m for each sample in the dataset. Here, w_j is the original input, while w'_j is what's reconstructed. They compare them using squared Euclidean distance, $\left\| w_j - w'_j \right\|^2$, which measures how well the data was reconstructed.

$$K_2(\theta) - \sum_{j=1}^{m} \left[w_j\, log\big(z_j\big) + \big(1 - w_j\big)\, log\big(1 - z_j\big) \right] \quad (6)$$

Equation (6) denotes the loss function, $K_2(\theta)$uses binary cross-entropy to measure how well the model's predictions match the actual inputs. Here, θ stands for the weights and biases that get tweaked during training. $\sum$ sums up the losses for all m samples. For each sample j, from 1 to m, w_j is the input value, and z_j, which is the model's guess, usually after passing through a sigmoid function. Taking the log of each case helps to make inaccurate predictions more effectively. If the true value is zero, $\big(1 - w_j\big)$ comes into play; similarly, $\big(1 - z_j\big)$shows the model's estimated chance for class 0. Finally, the negative sign ensures everything works together nicely for minimizing overall loss.

3.4 Improve reliability under system faults using FAT-DRNN

The use of FAT-DRNN provides increased system reliability through the combination of FAT and DRNN. FAT injects artificial failures into the training process, thus ensuring that the model learns to extract robust features in situations where failures, noise, and uncertainty exist in the system. The system continues operating reliably during runtime when faults arise in the hardware. DRNN is able to learn temporal patterns that be present in system dynamics, and which can be utilized for adaptive control using sequential datasets. As such, FAT and DRNN provide an innovative combination that improves fault tolerance, adaptability, and prediction capabilities.

- **FAT for Fault tolerance and reliability improvement**

FAT is the initial phase of FAT-DRNN, aimed at improving system reliability by training neural networks under simulated hardware faults. It deals with problems such as temporary faults, memory corruption, bit-flips, and processing interference. To make the predictions more accurate and reliable in such situations, FAT incorporates faults within the training stage of the algorithm.

Simulated Hardware Fault Injection: As the name suggests, FAT starts with fault injections into the neural networks in the training stage. The approach is unlike the traditional ones that always take a fault-free environment. Here, fault injections are done for certain parameters and activations to simulate faulty behavior during execution. This can be through memory faults, bit flips, computational faults, and communication faults, among others.

Reliability-Aware Weight Optimization: Once the faults are injected, the optimization is then done to discover the configurations that are highly accurate and fault tolerant. Traditional training techniques tend to

optimize prediction error, and hence they produce highly accurate yet brittle models. The suggested FAT algorithm, however, considers reliability issues in its optimization stage. The overall objective function is expressed as:

$$L_{total} = L_{prediction} + \lambda L_{prediction} \quad (7)$$

In Equation (7), L represents the loss function, L_{total} is the final loss used for training the model, $L_{prediction}$ measures how well the model predicts outputs (classification or regression loss). λ is a weighting factor that controls the trade-off between accuracy and reliability.

Robust Feature Learning Under Fault Conditions: In the last step of FAT, the goal is to learn resilient representations of features that are not affected by hardware problems. This is accomplished through training on both regular and faulty computation examples so that noise can be identified from real patterns. Invariant features can thus be learned where information is retained even if there is a fault.

Dynamic system control using DRNN: In complex engineering settings, Dynamic Recurrent Neural Networks (DRNN) make quick context-aware decision-making in evolving scenarios. They predict and handle nonlinear processes really well because they can track how things change over time and deal with varying system dynamics. Due to the ability of DRNNs to update their state all the time, they maintain stability amidst changes or disruption in the input. Fault-tolerant training of DRNNs assists in retaining control even with the presence of hardware problems. So, this approach makes systems more reliable, cuts down on the need for static controls, and keeps operations smooth in tough, safety-critical situations.

$$j_s = \sigma\left(X_j . w_s + V_j . out_{s-1} + a_j\right) \quad (8)$$

j_s is the gate value that controls how info flows for updates. a keeps neuron activations steady. j controls how much info gets updated. s monitors the system's progress over time. σ is the sigmoid function that maps numbers into a probability range. X_j represents current system measurements; it's the input feature vector. w_s determines the input's impact over time, V_j captures what happened before in the sequence, out_{s-1} the last output saves the previous system state, b is the bias term, keeps everything stable, and shifts are presented in Equation (8).

$$state_s = b_s \Theta j_s + e_s \Theta state_{s-1} \quad (9)$$

Equation (9) denotes the current system state, $state_s$ stores dynamic memory. b_s provides a candidate state with new info. The update gate, j_s controls how much new stuff comes in. e_s acts as the forget gate, saving past system knowledge. $state_{s-1}$is the last system state with historical data. Θ shows element-wise multiplication that merges gated info efficiently.

$$out_s = tanh(state_s)\ \Theta p_s \quad (10)$$

In this Equation (10), out_s final system output shows controlled response behavior, $state_s$ internal memory capturing current system dynamics, $tanh$ smooth activation keeping output stable and bounded, h (bias/hidden adjustment), a small correction term improving output accuracy and stability in a dynamic control system, p_s $g$ate deciding how much information is released.

Algorithm 1: FAT-DRNN for Adaptive System Reliability

```
Input:
D → System dataset (signals, performance metrics)
E → Training epochs, ρ → Fault rate, λ → Reliability weight
Output:
→ Reliable predictions, M → Trained model
Begin
Load dataset D
Preprocessing (Z-score normalization):
    Normalize all features in D using the training mean and standard deviation
Feature Extraction (Autoencoder):
    Initialize encoder–decoder parameters
    For epoch = 1 to E:
        Encode input data into compact features
        Reconstruct input from encoded features
        Compute reconstruction loss and update parameters
    End For
    Extract compressed feature set F
```

```
Fault-Aware Training (FAT):
    Initialize the DRNN model
    For epoch = 1 to E:
        For each feature vector in F:
            Inject random faults with probability ρ
            Predict output using DRNN
            Compute prediction + reliability loss
            Update model parameters using combined loss (λ-weighted)
        End For
    End For
Dynamic Control using DRNN:
    For each time step:
        Update internal state using current input and past output
        Generate controlled system output
    End For
Return O and trained model M
End
```

The suggested FAT-DRNN model comprises the components of Z-Score normalization, feature learning through an autoencoder, fault awareness learning, and dynamic control using DRNNs for improved system robustness. Firstly, robust features are extracted from the data after normalization and compression. To improve robustness, artificial faults are added to the training phase. Finally, time dependency is achieved through DRNN. The whole approach is summarized in Algorithm 1.

4. Result and Discussion

The proposed FAT-DRNN-based redundancy-driven method makes broken DL systems more reliable when hardware fails. It shows better accuracy and fault tolerance, too. They built it using Python 3.10 and tested it on a Windows system with an Intel® Core™ i7 processor. Not only is it simpler and needs less hardware, but its performance stays consistent at different fault levels as well.

Data feature exploration: These combinations show how different system conditions affect the interplay of reliability components. As follows from the legend, patterns of reliability, fault tolerance, and stability in cumulative form have been presented on the orderly set of data shown in Figure 2(a). Correlation of reliability values within the range of 0.65-0.98 with fault tolerance of 0.6-0.95 and stability of 0.7-0.97 has been detected. System stability for low, moderate, and high values of reliability is shown in Figure 2(b). Low variability means stable system operation: low reliability is equal to 0.5-0.7, moderate – 0.7-0.85, and high – 0.85-0.98.

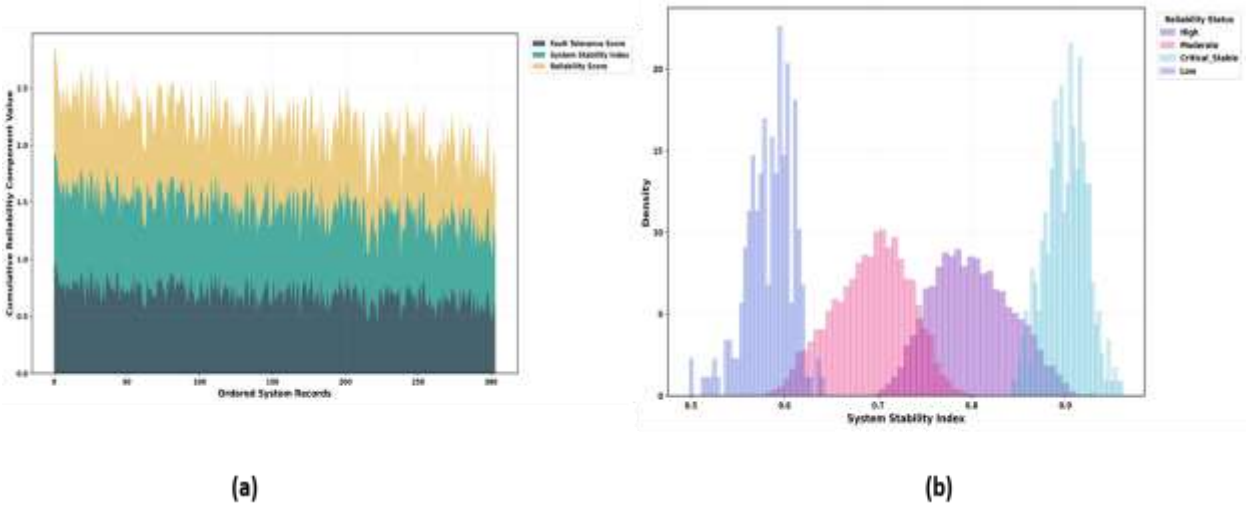

Figure 2: Distribution of (a) Cumulative Reliability Components Across Ordered System Records (b) System Stability Index by Reliability Status

Inference Speed (ms): The speed at which a model can process a single input sample is known as inference speed. For systems that must adjust and resolve problems. **Accuracy (%):** Accuracy refers to the frequency at which reliability is correctly determined. The measure refers to the degree to which the predictions are in agreement with the true value of the data. **Precision (%):** Precision refers to the correct identification of reliable cases relative to all reliable predictions and minimizes false positives**. Recall (%):** This evaluates the model's ability to identify all of the genuinely reliable cases. A high recall score is essential for systems where errors are not an option. **F1 Score (%):** The F1 score strikes a balance between recall and precision. When dependability classes aren't equally represented in the data, this statistic provides a crucial compromise.

Performance evaluation using ImageNet datasets [16]: The proposed FAT-DRNN was trained on the ImageNet dataset; the ALPRI-FI [16] based DNN approach achieves an inference speed of 29, demonstrating excellent fault-aware performance. The FAT-DRNN model achieves a maximum inference speed of 25 after training on ImageNet. FAT-DRNN has a better mix of performance and fault tolerance in adaptive systems since it is more resilient and reliable. That makes it fantastic for safety-critical applications since it handles redundancy well and boosts performance with less inference time.

Performance evaluation using the adaptive system reliability dataset: Both the proposed and existing methods were trained on the adaptive system reliability dataset; the FAT-DRNN model is compared to the ALPRI-FI (DNN). FAT-DRNN performs way better at 9.6 ms for inference; Table 2 and Figure 3 show that the ALPRI-FI takes 12.8ms. Also, FAT-DRNN recall up to 97.8% from 96.9%, precision to 98.2% from 94.7%, accuracy to 97.1% from 95.4%, and the F1 score to 98.0% from 95.8%. As can be seen, these improvements prove the effectiveness and efficiency of FAT-DRNN. Hence, this approach can be used in all tasks that require fault-tolerant computing.

Table 2: Performance Comparison of Reliability Models on Adaptive System Reliability Dataset

Methods	Inference Speed (ms)	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
ALPRI-FI (DNN)	12.8	95.4	94.7	96.9	95.8
FAT-DRNN [Proposed]	9.6	97.1	98.2	97.8	98.0

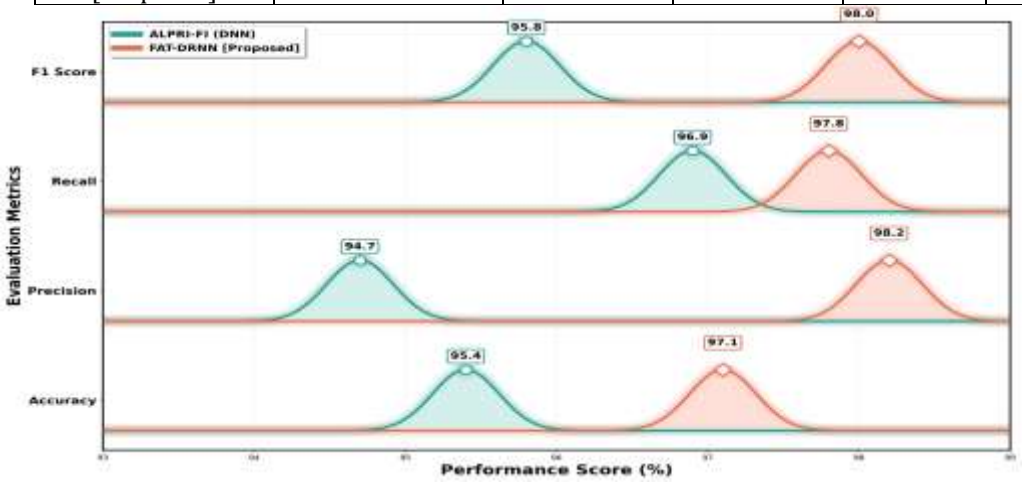

Figure 3: Evaluation Metrics of FAT-DRNN and ALPRI-FI (DNN)

Discussion: For example, uses a MILP-based [9] model for energy hubs, however, its scalability is hindered by the extremely high computational complexity. In addition, the author proposes the ALPRI-FI [16] approach for evaluating fault resilience; nevertheless, it is unable to evaluate complex processes and cannot recognize all kinds of hardware faults. On the contrary, the FAT-DRNN method solves these problems. Namely, by integrating redundancy-driven optimization and fault-awareness, it is able to provide efficient and scalable fault tolerance.

5. Conclusion

The introduced FAT-DRNN model improves the system reliability owing to its fault-aware learning and optimization by means of redundancy. Tests prove that it performs fantastically, as its inference takes only 9.6 ms and its accuracy is 97.1%. The precision, recall, and F1-score were estimated at 98.2%, 97.8%, and 98.0%, correspondingly. Given the fact that it decreases hardware dependencies and preserves output reliability at different fault levels, the model can be used for critical, safe, and energy-efficient large-scale engineering processes. It is worth noting that the method used for simulating the faults in FAT-DRNN is unlikely to reflect the real faults in hardware. What is more, fault-aware models create additional difficulties with training, and they have a different performance for different datasets; therefore, their use is limited by this factor. Further research needs to validate the introduced model with the use of real hardware and a wide range of datasets. Moreover, its scalability and adaptability may be enhanced through its combination with other DL architectures.

References

1 Zhang, Y., Wang, H., Pan, X. and Wang, Q., 2026. Hierarchical Redundancy-Driven Real-Time Replanning for Manipulators Under Dynamic Environments and Task Constraints. Electronics, 15(8), p.1577. https://doi.org/10.3390/electronics15081577

2 Wu, S., Mao, B., Jiang, H., Luan, H. and Zhou, J., 2019. PFP: Improving the reliability of deduplication-based storage systems with per-file parity. IEEE Transactions on Parallel and Distributed Systems, 30(9), pp.2117-2129. https://doi.org/10.1109/TPDS.2019.2898942

3 Liu, K.Y., Hung, T.K., Lin, C.H. and Liu, Y.C., 2024. Redundancy-Driven Multi-Task Adaptive Backstepping Tracking Control for Aerial Manipulators. IEEE Access, 12, pp.47134-47145. https://doi.org/10.1109/ACCESS.2024.3382983

4 Cámara, J., Troya, J., Vallecillo, A., Bencomo, N., Calinescu, R., Cheng, B.H., Garlan, D. and Schmerl, B., 2022. The uncertainty interaction problem in self-adaptive systems. Software and systems modeling, 21(4), pp.1277-1294. https://doi.org/10.1007/s10270-022-01037-6

5 Patil, S.S., Bewoor, A.K., Kumar, R., Ahmadi, M.H., Sharifpur, M. and PraveenKumar, S., 2022. Development of optimized maintenance program for a steam boiler system using reliability-centered maintenance approach. Sustainability, 14(16), p.10073. https://doi.org/10.3390/su141610073

6 Soundappan, S.J., 2022. AI-based fault detection and isolation for reliability in modern power systems. International Journal of Research Publications in Engineering, Technology and Management (IJRPETM), 5(4), pp.7106-7110. https://doi.org/10.15662/IJRPETM.2022.0504002

7 Li, W., Guan, Y., Huang, D. and Trisovic, N., 2023. Two methods for studying the response and the reliability of a fractional stochastic dynamical system. Communications in Nonlinear Science and Numerical Simulation, 120, p.107144. https://doi.org/10.1016/j.cnsns.2023.107144

8 Xu, Y., Kohtz, S., Boakye, J., Gardoni, P. and Wang, P., 2023. Physics-informed machine learning for reliability and systems safety applications: State of the art and challenges. Reliability Engineering & System Safety, 230, p.108900. https://doi.org/10.1016/j.ress.2022.108900

9 Nozarian, M., Fereidunian, A. and Barati, M., 2024. A system of systems approach to reliability-oriented planning of people-centric smart city energy infrastructure: A bilevel MILP formulation. IEEE Systems Journal, 18(4), pp.2085-2096. https://doi.org/10.1109/JSYST.2024.3474090

10 Tarafdar, R., 2025. Self-healing Ai model infrastructure: an automated approach to model deployment maintenance and reliability. International Journal of Information Technology and Management Information Systems (IJITMIS), 16(1), pp.992-1004. https://doi.org/10.34218/IJITMIS_16_01_071

11 Guo, J., Wang, L. and Wang, X., 2022. A group maintenance method of drone swarm considering system mission reliability. Drones, 6(10), p.269. https://doi.org/10.3390/drones6100269

12 Li, M., Luo, Y. and Xie, L., 2022. Fatigue reliability prediction method of large aviation planetary system based on hierarchical finite element. Metals, 12(11), p.1785. https://doi.org/10.3390/met12111785

13 Zuo, J., Zhou, S., Xu, Z., Gao, C., Zheng, S. and Chen, P., 2023. Reliability Evaluation Method of Multi-Voltage Levels Distribution System Considering the Influence of Sense-Control Terminal Faults. Applied Sciences, 13(21), p.11761. https://doi.org/10.3390/app132111761

14 Postnikov, I., 2023. Methods for the reliability optimization of district-distributed heating systems with prosumers. Energy Reports, 9, pp.584-593. https://doi.org/10.1016/j.egyr.2022.11.085

15 Holst, C.A. and Lohweg, V., 2022. Designing possibilistic information fusion—The importance of associativity, consistency, and redundancy. Metrology, 2(2), pp.180-215. https://doi.org/10.3390/metrology2020012
16 Mahmoud, K. and Nicolici, N., 2024. ALPRI-FI: A Framework for Early Assessment of Hardware Fault Resiliency of DNN Accelerators. Electronics, 13(16), p.3243. https://doi.org/10.3390/electronics13163243