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

Operational Resilience in Industrial Systems: Latency-Aware and Adaptive Architectures

Abhiraj Malhotra[1], Yu Jun[2], Dr. Vikram V. Patel[3], Malarvizhi S[4], Suganya S[5], Suraj Bhan[6], Kaustubh Milind Utpat[7], Boymirzayeva Khurshida Sobirovna[8]

[1] Centre for Research Impact & Outcome, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab 140401, India. Email: abhiraj.malhotra.orp@chitkara.edu.in ORCID: 0009-0005-1871-4807
[2] Research Scholar, School of Business and Management, Lincoln University College, Malaysia. Email: jun.phdscholar@lincoln.edu.my
[3] Associate Professor, Faculty of Engineering, Gokul Global University, Sidhpur, Gujarat, India. Email: vikrampatel@gokuluniversity.ac.in ORCID: 0009-0006-4834-6766
[4] Assistant Professor, Department of Commerce, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: malarvizhicom@maher.ac.in
[5] Assistant Professor, Department of Management Studies, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, India. Email: ssuganyamba@maher.ac.in
[6] School of Engineering & Technology, Noida International University, Greater Noida, Uttar Pradesh 203201, India. Email: suraj.bhan@niu.edu.in
[7] Department of Mechanical Engineering, Vishwakarma University, Pune, Maharashtra 411048, India. Email: kaustubh.utpat@vupune.ac.in
[8] Assistant, Department of Social Sciences and Humanities, Samarkand State Medical University, Samarkand 140100, Uzbekistan. Email: xurshida_boymirzayeva@sammu.uz ORCID: 0009-0006-9263-5856

Abstract
Operational resilience in industrial systems requires continuous adaptation to dynamic loads, uncertain disturbances, and strict timing constraints. Existing intelligent automation approaches emphasize accuracy but often overlook latency-aware decision-making and resilience under fluctuating industrial conditions, creating limitations in time-critical environments. This research aims to design Latency-Aware Deep Learning (DL) Architectures for Operational Resilience in Industrial Systems using a Cat Swarm Algorithm-driven Dynamic Deep Neural Network (CSA-DDNN). The purpose of the proposed method is to enable low-latency, fault-tolerant, and reliable industrial decision-making under dynamic operating conditions. Data pre-processing employs normalization and noise filtering using Min–Max normalisation and adaptive Kalman filtering to ensure temporal consistency. Feature extraction integrates Principal Component Analysis (PCA) to derive compact and discriminative representations. The model systematically acquires data, performs preprocessing, extracts salient features, and feeds them into a latency-aware learning module for resilient decision-making. The CSA-DDNN model utilizes CSA for optimal hyperparameter tuning and adaptive weight selection, enhancing exploration–exploitation balance, while the DDNN adjusts its structure in response to latency variations and system uncertainties for robust inference. This architecture explicitly models latency constraints, enabling timely responses and sustained operational stability. The Experimental result demonstrates an accuracy of 98.4%, an F1 score of 98.1%, parameters of 3.8 M, FLOPs of 250 M, Latency of 17 ms, and Energy of 6.6 mJ, which is implemented through Python. The approach enables scalable, low-latency, and resilient industrial intelligence with efficient resource utilization and consistent operational continuity.

Keywords: Operational resilience, Latency-aware systems, Industrial IoT, Fault tolerance, Adaptive learning, Intelligent automation.

1. Introduction

The exponential rise in Industry 4.0 has led to increased integration of automated industrial environments by adopting cloud, edge, and fog computing, along with wireless communications. With the advent of Optical Internet of Things (O-IoT), the data communication process and efficiency in a large-scale deployment of Internet of Things have become increasingly effective [1]. The field of medicine has seen the development of the Internet of Medical Things (IoMT), enabling intelligent monitoring and providing customized medical services through the use of interconnected medical equipment. Likewise, the advancements in sixth-generation network technology are fostering ultra-reliable low-latency communication (URLLC) solutions [2]. In addition, Cloud-Fog Automation and Autonomous Industrial Cyber-Physical Systems have attracted considerable interest because of their capability of providing effective integration of communication, computing, and control in the industrial domain [3]. Moreover, Multi-Robot Systems (MRS) are now becoming dependent upon Internet-of-Things connectivity and distributed intelligence.

Various Artificial Intelligence (AI), Machine learning (ML), and DL techniques have been established as key enablers towards intelligent decision-making and resource optimization in all these application scenarios. Latency-aware and energy-efficient task offloading in fog-enabled IoT environment has been accomplished using Deep Q-Network (DQN)-based learning schemes [4]. Security and adaptability in IoMT systems, along with maintaining data privacy in the process, has been achieved by combining reinforcement learning and federated learning techniques [5]. AI-enabled orchestration techniques have also been designed for the Network Function Virtualization (NFV) environment in 6G systems that allow for the prediction of latency and efficient resource management. Robotic applications have also used DL-based models with pose estimation methodologies in order to increase accuracy in decision-making in diverse network environments [6]. Additionally, ML-based multiclass storage tiering strategies have been introduced in cloud storage systems to enhance the trade-off between cost and latency.

In spite of the progress achieved in this field, there are still quite a number of issues that need to be addressed. First, wireless communication networks are associated with certain latencies and instabilities that can undermine the performance of real-time industrial control systems regardless of using the most sophisticated architecture of automation [7]. Second, there are still considerable knowledge gaps within Cloud-Fog Automation regarding the joint optimization of communication, computation, and control operations. Third, despite being an important performance factor, latency is still not thoroughly considered within many cybersecurity mechanisms, especially IDS, as late attack identification can cause serious consequences for Cyber-Physical Systems (CPS) and IoT environments [8]. Apart from this, there are a number of other limitations connected to the use of current technologies, including communication overhead, limitations on resources, scalability, security issues, dynamic nature of the network environment, and others.

Research aim: The objective of this research is to design a latency-aware DL framework by applying CSA-DDNN for effective decision-making in dynamic environments while ensuring accuracy, minimal latency, and fault tolerance.

Research organisation: In Section 1, the importance of latency-aware DL methods in industrial operations impacted by Industry 4.0 is discussed, and explains how AI, ML, and DL enable processing. In Section 2, related research on latency-aware approaches to ML is discussed. These include approaches that utilize cloud and edge and DL-based multi-robot perception, among others. In Section 3, the proposed model, referred to as CSA-DDNN, is explained. This includes the Min-Max Normalization process, Adaptive Kalman Filtering process, PCA, CSA for hyperparameter tuning, and DDNN. In Section 4, various evaluation experiments for the CSA-DDNN model are conducted, comparing them to other baseline models such as TabNet. Research's conclusion is displayed in section 5.

2. Related work

Recent research on latency-aware architectures, resilient IoT systems, edge computing, federated learning, and multi-cloud platforms is summarized in Table 1, highlighting their research objectives, methodologies, experimental outcomes, and limitations. To enhance system resilience, scalability, and real-time decision-making in dynamic situations, the evaluated works show the increasing use of AI, edge intelligence, distributed computing, and latency-aware optimization techniques.

Table 1: Recent research on Latency-Aware architectures

Ref	Research Aim	Technique Used	Experimental Results	Limitation

[9]	Achieve High Resilience and Low Latency in multi-cloud data platforms.	Multi-Cloud Data Engineering (MCDE), Data Replication (DR), Policy-as-Code (PaC).	Improved fault tolerance and service continuity during cloud outages.	High synchronization and consistency overhead.
[10]	Enable real-time injury risk prediction with low latency.	Cloud–Edge Architecture (CEA), Edge Inference (EI).	Reduced End-to-End Latency (E2EL) and improved scalability.	Evaluated mainly using simulated workloads.
[11]	Improve wireless industrial control under communication delays.	Internal Model Control-Latency Aware (IMC-LA), Wi-Fi 6, 5G.	Reduced position error and improved control stability.	Tested only on Ball-and-Beam (BB) system.
[12]	Develop a resilient decentralized IoT consensus.	Sequenced Probabilistic Double Echo, Randomized Asynchronous Consensus with Effective Real-time Sampling, and Peer-assisted Latency-Aware Traffic Optimization.	Achieved throughput >600 MB/s in a 100-node network.	Limited large-scale real-world validation.
[13]	Optimize TinyML for 6G edge environments.	DRL-TinyEdge, Deep Reinforcement Learning (DRL), Dynamic Offloading.	28% lower latency and 43% lower energy consumption.	Evaluated in simulated 6G environments.
[14]	Improve Software defined IoT (SD-IoT) scalability and fault tolerance.	Multi-Controller Load Balancing and Traffic Optimization (MC-LBTO), which consists of Secure Trusted Adaptive Multi-Control, P4-based Adaptive Load Balancer, and P4-enabled Dynamic State Monitoring	36% lower request latency and 25% higher throughput.	Focused mainly on SD-IoT scenarios.
[15]	Minimize latency during Mobile Edge Computing (MEC) cluster upgrades.	Reinforcement Learning (RL), Self-Attention Scheduling.	Reduced task latency by ~30%.	Limited to cluster upgrade operations.
[16]	Improve FL fault tolerance and reduce communication latency.	Hierarchical FL, EfficientNet-B0, Tabular Neural Network (TabNet), Federated Proxima (FedProx).	Accuracy >93% under 40% server failures; 25% lower latency.	Evaluated primarily through simulations.

Research gap: Recent research in cloud-fog computing and AI in industry has primarily aimed at improving accuracy and reducing latencies. However, they lack integrated solutions for optimizing latency, energy efficiency, and model adaptability in dynamic industrial environments. Existing algorithms typically use single-objective optimization with fixed neural networks, resulting in inefficiencies. The proposed CSA-DDNN approach tackles these issues by integrating a DDNN that can modify its architecture for better decision-making and utilizing Cat Swarm Optimization for tuning hyperparameters.

3. Methodology

The designed methodology creates a latency-aware industrial intelligence system for operations resiliency by using the CSA-DDNN model. Data processing through Min-Max normalization and Adaptive Kalman Filtering is performed at the beginning, followed by data transformation into a compressed form using PCA. Hyperparameters are optimized using CSO, whereas learning is done dynamically using a DDNN to make decisions quickly, accurately, and fault- tolerance in industries, as seen in Figure 1.

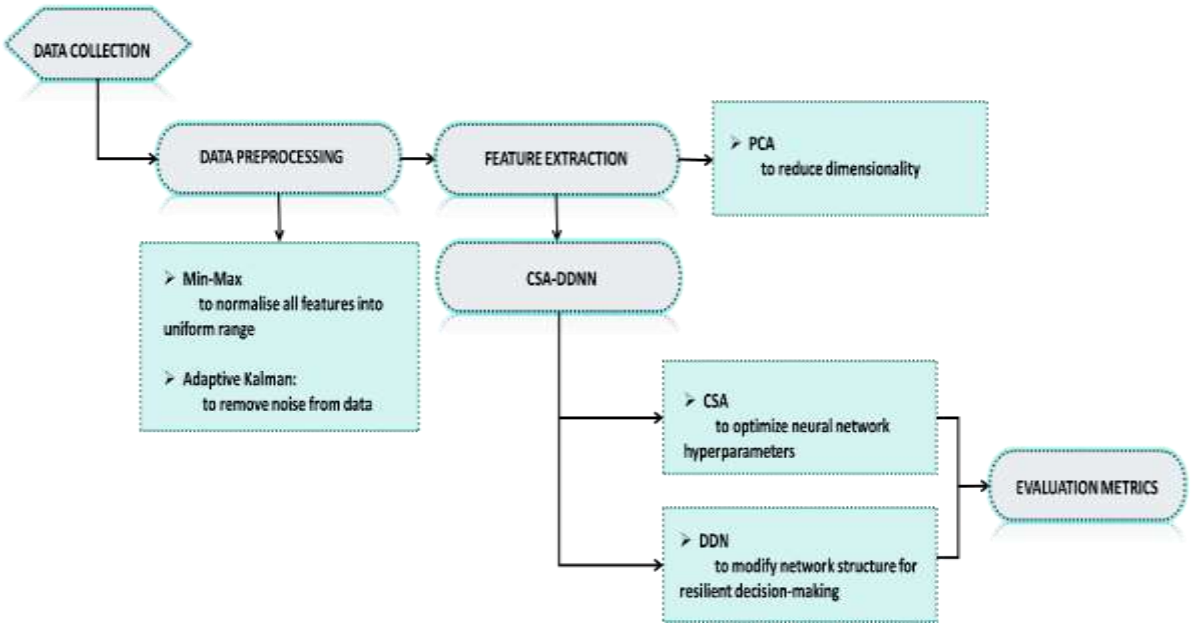

Figure 1: Methodology of the proposed method

3.1 Dataset description An open-source dataset (https://www.kaggle.com/datasets/colabsss/latency-aware-industrial-resilience-data) comprises 13,000 records, which is used in the research of operational resilience in an industrial setting, taking into account multiple scenarios that include latencies, loading, disturbances, and faults. The included variables involve industrial sensors' readings, delays in communication, loading metrics, disturbance metrics, fault metrics, and operational resilience status for the environment where manufacturing or power control units operate. This dataset is meant to be analysed using operational resilience, whereby 80% of the total records are used for training while 20% is used for testing.

3.2 Preprocessing of data

To improve the quality and consistency of industrial sensor data, the data processing stage uses Adaptive Kalman Filtering and Min-Max normalization. Min–Max normalization transforms all input features into a unified range to ensure uniform contribution during model training, while Adaptive Kalman Filtering removes noise and reduces uncertainty in sensor signals while preserving temporal dynamics. Together, these methods aim to improve data reliability, stability, and suitability for accurate latency-aware DL-based industrial decision-making.

3.2.1 Min-Max normalisation

A method for rescaling numerical features into a fixed range, usually [0, 1], is called min-max normalization while, maintaining the relationships among data points, and the goal is to enhance numerical stability, accelerate convergence in DL models, and prevent larger range features from overshadowing others, thus improving the performance of latency-aware industrial prediction systems.

$$Y_{normalized} = \frac{Y - Y_{min}}{Y_{max} - Y_{min}} \quad (1)$$

Equation (1) represents the min-max normalisation equation, where $Y_{normalized}$ is the scaled value of the original feature after transformation. Y is the original input feature value, Y_{max} and Y_{min} are the maximum and minimum values of the features.

3.2.2 Noise reduction using adaptive Kalman filtering

In an industrial setting, an adaptive Kalman filtering methodology is utilized as a recursive technique to filter noisy and uncertain sensor data to determine the state of a system by forecasting future values using measurements. Depending on how the system's environment changes, the method adjusts its own parameters, such as the noise covariance matrix. The goal is to make correct DL choices by removing noise while maintaining temporal dynamism.

$$C_{v_k^{(-)}} = \frac{1}{N} \sum_{j=k-N+1}^{k} v_j^{(-)} \left(v_j^{(-)}\right)^T \quad (2)$$

$$\hat{R}_k = C_{v_k^{(-)}} H_K P_k^{(-)} H_k^T \quad (3)$$

$$\hat{Q}_k = \frac{1}{N} \sum_{j=k-N+1}^{k} \Delta x_j \, \Delta x_j^T + P_k^{(+)} - \Phi P_{k-1}^{(+)} \Phi^T \quad (4)$$

$$\hat{C}_{\Delta x_k} = \frac{1}{N} \sum_{j=k-N+1}^{k} \Delta x_j \ \Delta x_j^T \approx \ K_k \hat{C}_{v_k^{(-)}} K_k^T \quad (5)$$

In Equation (2), the term $C_{v_k^{(-)}}$ is the covariance matrix of the measurement innovation over a sliding window of size N, where $v_j^{(-)}$ is the innovation at time j and $(v_j^{(-)})^T$ is its transpose and Σ indicates summation of over all time steps j from $k-N+1$ to k. Equation (3) uses this innovation covariance to estimate the measurement noise covariance $\hat{R}_k$, with H_K as the observation matrix, and H_k^T is its transpose, where T is the transpose matrix and $P_k^{(-)}$ as the predicted error covariance. In Equation (4), the process noise covariance $\hat{Q}_k$ is the calculated using the state estimation error Δx_j and Δx_j^T is its transpose, the posterior covariances $P_k^{(+)}$ and $P_{k-1}^{(+)}$, and the state transition matrix Φ, reflecting changes in system dynamics. Equation (5) links the state error covariance $\hat{C}_{\Delta x_k}$ with innovation covariance using Kalman Gain operator K_k, while $\hat{C}_{v_k^{(-)}}$ is the covariance matrix. These equations allow for the adaptive Kalman filter to adapt to changes in the measurement and process noises, thus increasing its robustness, efficiency, and performance in industry.

3.3 Feature extraction using PCA

PCA is a dimensionality reduction method that preserves the greatest amount of variance in high-dimensional correlated data while reducing it to a smaller set of uncorrelated components. Within industrial systems, it helps extract the most informative features from large sensor and operational datasets, reducing computational complexity and improving model efficiency. Its objective is to reduce data dimensionality, remove redundancy, and retain essential information to enhance the performance of latency-aware DL models.

$$N^{m*m} = n_{j,i}, (n_{j,i} = Cov(B_j, B_i)) \quad (6)$$

$$Cov(B_j, B_i) = \frac{\sum_{l=1}^{m} (B_{jl} - B_j')(B_{il} - B_i')}{m-1} \quad (7)$$

$$Nv = \delta v \quad (8)$$

From the above equation (6-8), N^{m*m} represents the size of the covariance matrix, where m is the total number of samples. $n_{j,i}$ is the Covariance value between feature B_j and feature B_i and B_j and B_i are individual features. B_{jl} and B_{il} are the value of feature B_j and B_i at sample l and l is the index of each sample observation. B_j' and B_i' are the mean values of feature B_j and B_i. Cov denotes the covariance between two variables, B_j and B_i, $\sum_{l=1}^{m}$ represents the summation over all m samples in the dataset used to compute covariance. N is the covariance matrix of the dataset, where v is the original value, and δ is a scalar value.

3.4 CSA and DDNN for adaptive Optimization

The CSA is a metaheuristic optimization method inspired by cat behavior, combining seeking and tracing modes to optimize neural network hyperparameters by balancing exploration and exploitation. A DDNN is an adaptive architecture that dynamically modifies its structure based on input conditions, system load, and latency variations to ensure efficient learning. In combination, their purpose is to improve the performance of models through optimal tuning and adaptation that will facilitate quick and efficient decision making in changing industrial settings.

3.4.1 DDNN operation using a loss function

The DDNN can be described as an adaptive deep learning model that changes its internal configuration in terms of the number of activated neurons, layers, or computational pathways according to the data and runtime conditions of the system. As such, the ability to adjust the model makes it capable of effectively responding to changes in industrial situations and demands for computation, and its primary aim is to attain robust decision-making with low latency and limited resources.

$$L = (R(t,b) - u)^2 \quad (9)$$

$$R = \sum_{i=1}^{m} R(t_i, b_i) \quad (10)$$

$$L = (u - R(t,b))^2 = (u - \sum_{i=1}^{m} R(t_i, b_i))^2 \quad (11)$$

In Equation (9), L denotes the loss function, $R(t,b)$ is the estimated action-value for taking action b in state t, and u represents the reward received after taking action b. Equation (10) expresses the total Q-value as the sum of individual Q-values across m state-action pairs, where t_i and b_i are the i^{th} state and corresponding action. Finally, Equation (11) combines the previous two concepts, showing that the loss is calculated as the squared difference between the actual reward u and the aggregated Q-value over multiple state-action pairs.

3.4.2 CSA for position velocity update

The CSA is a nature-inspired optimization technique used to find optimal solutions by simulating the movement behavior of cats through seeking and tracing modes and its objective is to efficiently explore the search space and exploit promising regions to obtain optimal hyperparameters, thereby improving model accuracy, convergence, and overall performance in complex optimization problems.

$$ve_d^q(t+1) = s * ve_d^q(t) + b * u * (x_{best}^d - x_q^d) \quad (12)$$

$$x_q^d(t+1) = x_q^d + ve_d^q(t+1) \quad (13)$$

From equations (12-13) where $ve_d^q(t)$ is the velocity of the q^{th} cat in the d^{th} dimension at time t, s is the weight of inertia that regulates the impact of prior velocity,b is the coefficient of acceleration, u is a random value in $[0, 1]$introducing stochastic behavior, x_{best}^d is the best solution found so far in dimension d, and x_q^d is the current position of the cat. The updated velocity $ve_d^q(t+1)$ guides the search direction, while $x_q^d(t+1)$ represents the new position of the solution candidate.

Algorithm 1: CSA-DDNN-Based Industrial Resilience Framework

$Input: D \rightarrow Dataset, MaxIter \rightarrow Iterations, N \rightarrow Population\ size, lb, ub \rightarrow Bounds$
$Output: M \rightarrow Trained\ DDNN\ model$
$Begin$
1. $Load\ dataset\ D$
2. $Preprocess\ data$:
   $Apply\ Min-Max\ normalization\ and\ Adaptive\ Kalman\ Filtering.$
3. $Feature\ extraction\ using\ PCA$:
   $N^{m*m} = n_{j,i}, (n_{j,i} = Cov(B_j, B_i))$
   $Select\ top\ eigenvectors \rightarrow Feature\ set\ F$
4. $Train\ DDNN\ with\ optimized\ hyperparameters\ and\ dynamic\ structure$
5. $Initialize\ CSA\ population\ to\ 0$
6. $For\ t = 1\ to\ MaxIter$:
   $Update\ velocity: ve_d^q(t+1) = s * ve_d^q(t) + b * u * (x_{best}^d - x_q^d)$
   $Update\ position: x_q^d(t+1) = x_q^d + ve_d^q(t+1)$
   $Evaluate\ DDNN\ fitness\ and\ update\ x_{best}^d$
7. $Return\ M$
End

The algorithm 1 first preprocesses industrial sensor data using Min-Max normalization and Adaptive Kalman Filtering, then reduces dimensionality via PCA. CSA optimizes the DDNN hyperparameters by updating cat positions and velocities, balancing exploration and exploitation. Finally, the DDNN is trained dynamically for low-latency, accurate, and resilient industrial decision-making.

4. Results and discussion

Experimental results for the proposed CSA-DDNN model are presented in this section. The performance of the CSA-DDNN method is evaluated and compared with baseline models such as TabNet [16]. The complete setup for experiment has been created using the Python programming language, and the output obtained has been evaluated using the following parameters: accuracy, F1 score, latency, Parameters, Flops, and energy consumption.

4.1 Performance Correlation, State, and Latency Analysis

Figure 2 illustrates pairwise interactions between processing time, anomaly score, and stability for two categories of operation. Process time falls between 200 and 800 ms, and majority samples fall between 450 and 650 ms. The anomaly score varies between 0.20 and 0.80; most samples have values falling between 0.40 and 0.65, whereas the stability range lies between 0.60 and 1.00, mostly above 0.75. There is a positive relationship between processing time and anomaly score, although stability becomes smaller with increases in processing time and anomaly score. There are clear separations between the two different categories, with one having small anomalies and large stability and the other large anomalies and small stability.

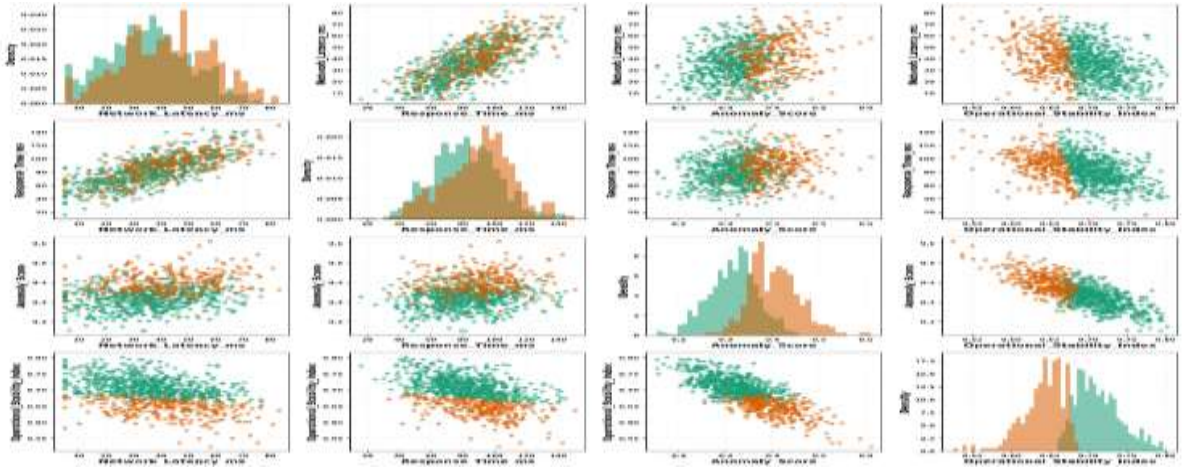

Figure 2: Multivariate analysis of processing time, anomaly behavior, stability variation, and operational state discrimination.

Figure 3 depicts the performance and latency features in terms of response time, throughput, and request distribution. In Figure 3(a), latency levels range from 5 to 90 ms, with throughput values from 100 to 300 requests per second, exhibiting positive correlation for almost all observations in the range of 20-60 ms latency and 140-240 requests/second throughput level. Figure 3(b) presents system stability during approximately 1,000 observation intervals, in which response time stays in the range of 150-170 ms, CPU utilization changes from 20 to 40%, and memory utilization maintains the values from 100 to 110 units. Finally, in Figure 3(c), request distributions are plotted according to their latency intervals, which demonstrate their maximum concentrations at 40-80 ms; successful requests exceed the value of 200, while warnings reach about 120 requests.

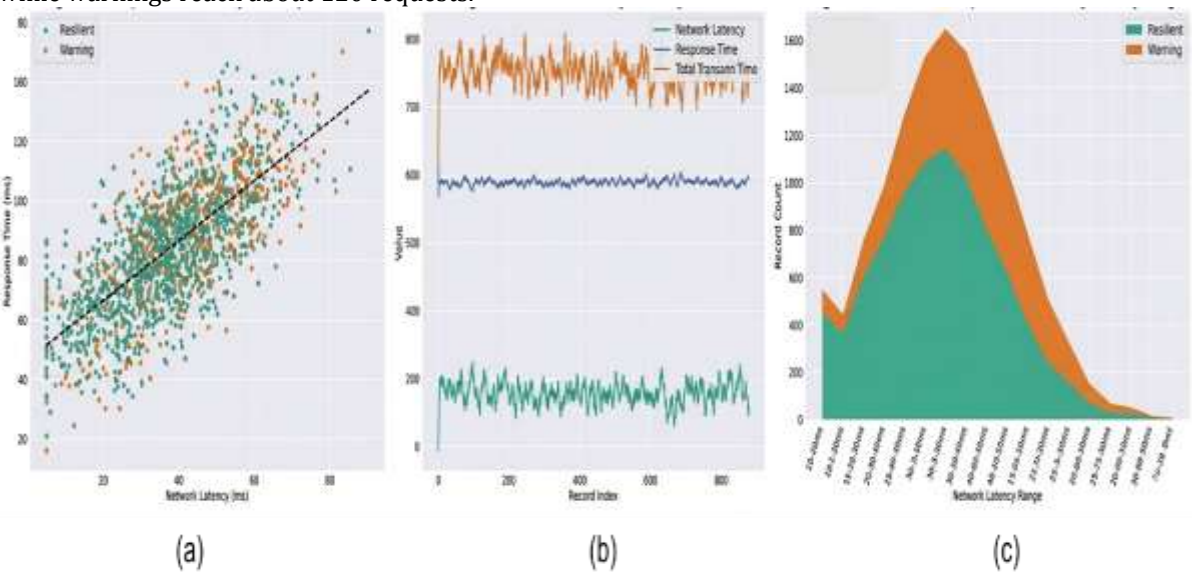

Figure 3: Analysis of (a): Latency and throughput relationship analysis, (b) Temporal performance monitoring analysis, (c) Request distribution across latency ranges.

4.2 Metrics evaluation

Accuracy (%): This measure assesses the proportion of accurately identified data samples among all anticipated data samples. This enables us to assess the effectiveness of the CSA-DDNN model in forecasting the several operating states of the industries, including Critical, Warning, and Resilient. **F1 (%):** The F1-Score measure aids in evaluating how successfully the classification algorithm strikes a balance between false positives and false negatives.This metric is especially important when dealing with an imbalanced dataset for the purpose of industry resilience prediction. **Params (M):** Parameters (in Millions) represent the total number of trainable weights in the neural network model. This helps us in determining the complexity of the model. **FLOPs (M):** Floating Point Operations (FLOPS) in Millions evaluate the number of floating-point operations required by the model during one forward pass. It is an effective measure of

computational efficiency of the model. **Latency (ms):** Latency helps to determine the time required by the model to predict any instance. **Energy (mJ):** This is the total energy consumed by the model while performing its prediction process.

4.3 Result and discussion

This proposed model is trained using the NASA CMAPSS Turbofan Dataset [16], comprising of multivariate time-series from 247 synthetic turbofan engines, with four types of faults generated, having 14 sensor measurements, such as temperatures, pressures, and flow rates collected for each engine cycle. Table 2 illustrates the comparative analysis of the CSA-DDNN model against the TabNet [16], showing how the CSA-DDNN model attains better results in terms of higher accuracy and F1-score, as well as lower resource usage, with a total of 3.2 million Params, 210 million FLOPs, lower latency (14ms), and energy usage (5.8mJ). Figure 4 further elaborates on these results.

Table 2: Evaluation of the suggested model's performance using the existing dataset

Model	Accuracy (%)	F1 (%)	Params (M)	FLOPs (M)	Latency (ms)	Energy (mJ)
TabNet (NASACMAPSS) [16]	95.1	95	3.6	240	18	6.7
CSA-DDNN [Proposed]	97.2	97.0	3.2	210	14	5.8

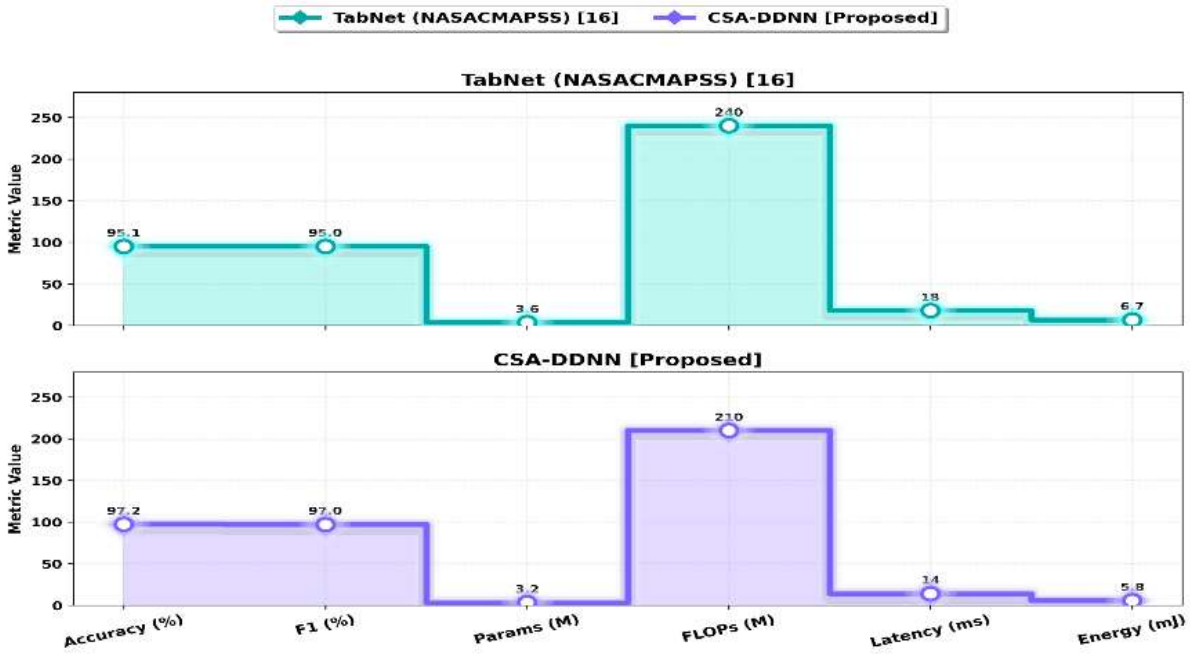

Figure 4: Graphical presentation of the proposed training in the existing dataset

Table 3 presents a comparative performance analysis between TabNet (NASA CMAPSS) [16] and the proposed CSA-DDNN model, which was trained on a latency-aware industrial resilience dataset. The proposed CSA-DDNN model delivers enhanced prediction accuracy, which is 98.4%, as well as an F1-score of 98.1%, but at the same time reduces computational costs (with parameters of 3.8M and 250M FLOPs), as opposed to TabNet [16]. Moreover, the CSA-DDNN model improves efficiency due to decreased latency (17 ms) and energy consumption (6.6 mJ). This claim is supported by Fig. 5 below that depicts the advantages of the CSA-DDNN model with respect to TabNet regarding all assessed criteria.

Table 3: Comparison of existing and proposed models trained on the latency-aware industrial resilience data

Model	Accuracy (%)	F1 (%)	Params (M)	FLOPs (M)	Latency (ms)	Energy (mJ)

TabNet (NASA CMAPSS)	96.8	96.5	4.2	280	22	7.9
CSA-DDNN [Proposed]	98.4	98.1	3.8	250	17	6.6

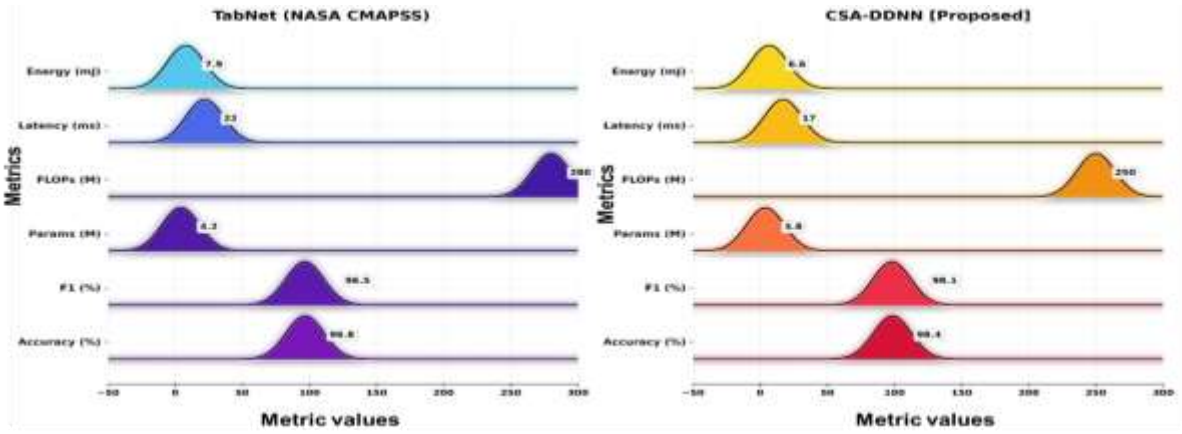

Figure 5: Representation of the existing and proposed models trained on the latency-aware industrial resilience dataset

Discussion

Existing methods for industrial prediction and latency-aware systems primarily rely on static DL architectures or single-objective optimization, which limits their adaptability under dynamic industrial conditions and fails to jointly optimize latency, accuracy, and energy efficiency. These techniques also face challenges with sensitivity to noise and redundant features in sensor data in multidimensional spaces. In contrast, the newly designed CSA-DDNN overcomes these challenges due to the integration of CSA in the process of tuning hyperparameters and structural adaptation through DDNNs.

5. Conclusion

The suggested model for CSA-DDNN achieves a remarkable improvement in the resilience of operations within industrial applications by incorporating Min-Max Normalization, adaptive kalman filtering, PCA-based feature extraction, optimization based on CSA, and a DDNN for effective decision-making with low latency and high performance. The experimental result gives 98.4% Accuracy, F1 score at 98.1%, Parameters at 3.8M, FLOPS at 250M, latency at 17ms, Energy at 6.6mJ, which has been carried out using the Python platform. However, the scope of this paper is constrained in the sense that it depends on datasets and simulations, making it impossible to cover all aspects of uncertainty involved in actual industrial applications. Future work may include implementing the model in various heterogeneous environments and distributed Edge-Cloud infrastructures with the use of meta-heuristic/hybrid optimization methods.

Reference

1. Saqib, N., Abdullah, N.F., Abu-Samah, A. and Nordin, R., 2025. Delay Optimized VNF Placement in 5G Enabled Industry 4.0 Networks Using DRL with Wireless Reliability and Cyberattack Resilience. IEEE Sensors Journal,25(20), pp.39064-39081. https://doi.org/10.1109/JSEN.2025.3604001
2. Benaboura, A., Bechar, R., Kadri, W., Ho, T.D., Pan, Z. and Sahmoud, S., 2025. Latency-aware and energy-efficient task offloading in IoT and cloud systems with DQN learning. Electronics, 14(15), p.3090. https://doi.org/10.3390/electronics14153090
3. Almuseelem, W., 2025. Secure latency-aware task offloading using federated learning and zero trust in edge computing for IoMT. IEEE Access,13, pp.117808-117830. https://doi.org/10.1109/ACCESS.2025.3586730
4. Alnaim, A.K. and Albarrak, K.M., 2025. Latency-Aware NFV Slicing Orchestration for Time-Sensitive 6G Applications. Systems, 13(11), p.957. https://doi.org/10.3390/systems13110957
5. Jin, J., Pang, Z., Kua, J., Zhu, Q., Johansson, K.H., Marchenko, N., and Cavalcanti, D., 2025. Cloud-fog automation: The new paradigm towards autonomous industrial cyber-physical systems. IEEE Journal on Selected Areas in Communications,43(9), pp. 2917 - 2937. https://doi.org/10.1109/JSAC.2025.3574587

6. Dwedar, M. and Jesser, A., 2025. Hybrid Decision Making Via Deep Learning for Integrated Visual Object Detection, Pose Estimation, and Energy Control in IoT Connected Multi-Robot Systems Using 5G Network. IEEE Access, 13, pp.194832-194849. https://doi.org/10.1109/ACCESS.2025.3630823
7. Dash, S. and Acken, J.M., 2026. Intrusion detection latency: the neglected metric. Cybersecurity, 9(1), p.144. https://doi.org/10.1186/s42400-026-00574-7
8. Motta, F.A., Villela, S.M., Bernardino, H.S., Gonçalves, G.D., and Vieira, A.B., 2026. A multiclass cost-latency aware framework for multi-tiered cloud storage optimization via access pattern forecasting. Computer Communications, 249,p.108437. https://doi.org/10.1016/j.comcom.2026.108437
9. Parepalli, S., 2024. Architecting multi-cloud data engineering models for high resilience and low latency patterns for active pipelines, consistent governance, and operational automation. International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), 10(7), pp.309-328. https://doi.org/10.32628/CSEIT24107452
10. Guo, X., Sun, Y., Wang, H., Qin, M., Jahangir, M., Guo, Y., and Yang, X., 2026. A latency-aware cloud–edge architecture for real-time injury risk early warning using wearable and training-load data. Journal of Cloud Computing,15, p.72. https://doi.org/10.1186/s13677-026-00887-5
11. Lyu, H., Pang, Z., Bengtsson, A., Nilsson, S., Isaksson, A.J. and Yang, G., 2024. Latency-aware control for wireless cloud-fog automation: Framework and case study. IEEE Transactions on Automation Science and Engineering, 22, pp.5400-5410. https://doi.org/10.1109/TASE.2024.3420770
12. Auhl, Z., Moraliyage, H., Chilamkurti, N. and Alahakoon, D., 2025. RACER: A Lightweight Distributed Consensus Algorithm for the IoT with Peer-Assisted Latency-Aware Traffic Optimisation. Technologies, 13(4), p.151. https://doi.org/10.3390/technologies13040151
13. Alaklabi, S. and Alharbi, S., 2026. DRL-TinyEdge: Energy-and Latency-Aware Deep Reinforcement Learning for Adaptive TinyML at the 6G Edge. Future Internet, 18(1), p.31. https://doi.org/10.3390/fi18010031
14. Alyanbaawi, A., El-Sayed, A., Salah, N., Said, W., Elmezain, M., and Elkomy, O., 2025. MC-LBTO: secure and resilient state-aware multi-controller framework with adaptive load balancing for SD-IoT performance optimization. Scientific Reports,15,p.44660. https://doi.org/10.1038/s41598-025-31216-6
15. Cui, H., Tang, Z., Lou, J., Jia, W., and Zhao, W., 2024. Latency-aware container scheduling in edge cluster upgrades: A deep reinforcement learning approach. IEEE Transactions on Services Computing, 17(5), pp.2530-2543. https://doi.org/10.1109/TSC.2024.3394689
16. Haripriya, R., Kumar, A., Pandey, M., Khare, N., Choudhary, J., Singh, D.P., Solanki, S., and Haripriya, A., 2025. Optimising Fault tolerance and Latency of Federated learning using Edge Servers and pre-trained models. IEEE Access,13, pp. 158737 - 158750. https://doi.org/10.1109/ACCESS.2025.3607738