Predictive Maintenance of Distributed Processing Unit (DPU) Failures in DCS and PLC Systems Using Artificial Neural Networks (ANN)

NitinTrivedi*

Carbon Capture & Utilization Group, NTPC Energy Technology Research Alliance, NTPC Ltd., Greater Noida, Uttar Pradesh, India.
nitintrivedi@ntpc.co.in

Abstract. In modern industrial automation, DCS (Distributed control System) and PLC (Programmable logic Controller) are widely used for controlling processes and ensuring the safe operation of complex systems. The Distributed Processing Unit (DPU), a very important component in both DCS and PLC architectures, is responsible for real-time data acquisition and control. However, due to its continuous operation in challenging environments, DPUs are susceptible to failures, leading to unplanned downtimes and substantial financial losses. This paper proposes a predictive maintenance framework using Artificial Neural Networks (ANN) to predict DPU failures by monitoring key operational parameters. The model got trained & validated using the "Application failure prediction " dataset from Kaggle, which includes sensor readings and other self-diagnostic parameters and failure logs that align well with the operational characteristics of DPUs. The ANN model demonstrated an accuracy of 92.8% on the validation dataset, providing a reliable solution for early failure prediction. By implementing this ANN-based predictive maintenance framework, industries can proactively predict and address DPU failures, reducing unplanned downtime and minimizing maintenance costs, thereby enhancing overall operational efficiency.

Keywords: DPU, DCS, PLC, Artificial Neural Networks, Predictive Maintenance, Failure Prediction, Kaggle, ANN

1 Introduction

1.1 Predictive maintenance

An AI-enabled maintenance strategy that utilizes real-time data, historical patterns, and machine learning models to forecast equipment failures before they occur**.** Before AI came into the picture, predictive maintenance was based on: Statistical trend analysis, Threshold-based alarms, Rule-based expert systems, Time-series pattern recognition and Feed-forward control logic**.** These methods relied on engineering rules, manual tuning, and fixed algorithms to predict or anticipate equipment issues. For instance, vibration

S. Shaswattam et al. (eds.), *NTPC Transactions on Energy Research (NTER 2025)*,
https://doi.org/10.2991/978-94-6463-849-3_14

analysis and thermography have long been used in rotating equipment health monitoring, without involving AI.

AI-Driven Predictive Maintenance (Modern Approach): Today, AI/ML techniques such as Artificial Neural Networks (ANN), Random Forests, Support Vector Machines (SVM), Long short term memory (LSTM) for time-series allow systems to learn from data and improve predictions dynamically, which adds accuracy, adaptability, and scalability over traditional methods.

While the term "predictive maintenance" may seem contemporary, its conceptual roots run deep into history. The philosophy emerged during World War II, when wartime aviation and military operations demanded unprecedented reliability from machinery. Engineers realized that by monitoring signs of wear and tear, they could proactively service critical components before catastrophic failures occurred. This laid the foundation for a proactive rather than reactive approach to system maintenance.

Interestingly, predictive intelligence in control systems predates the AI boom. In the context of thermal power plants, for instance, the three-element drum level control system—used to regulate water levels in boiler drums—employs a feed-forward control mechanism. This component anticipates load changes and adjusts feed-water flow pre-emptively, effectively exhibiting predictive behaviour. In essence, feed-forward control can be viewed as an early form of artificial intelligence, implemented through analogue control logic decades before the rise of neural networks.

With the evolution of control strategies and the introduction of Distributed Control Systems (DCS), the scope and capability of predictive logic expanded. What was once mechanical intuition encoded into control loops has now evolved into sophisticated ANN-based models, which can process massive amounts of real-time data, identify hidden patterns, and forecast failures with high precision. These models act as digital sentinels, tirelessly watching over assets, and ensuring timely human or automated responses.

1.2 In industrial automation, Distributed Processing Units (DPUs) are critical components in DCS (Distributed control System) and PLC (Programmable logic Controller), responsible for real-time data acquisition and control loop execution (see Fig. 1). However, due to continuous operation in harsh environments, DPUs are prone to failure, which can lead to unplanned downtimes and substantial financial losses.

Failure of active DPU leads to tripping of Unit/ halt of running processes. In a 660 MW power plant, for example, unplanned downtime can cost as much as INR 2.112 million (21.12 lakhs) per hour, considering the electricity generation rate of 660,000 kWh per hour at INR 3.20 per unit besides loss of DC charges and light up oil cost to bring back the unit on bar. the Total cost per unit tripping comes out INR 25 to 30 million (2.5 to 3 Cr). Additional risks include not only financial losses but also potential safety hazards and regulatory compliance issues.

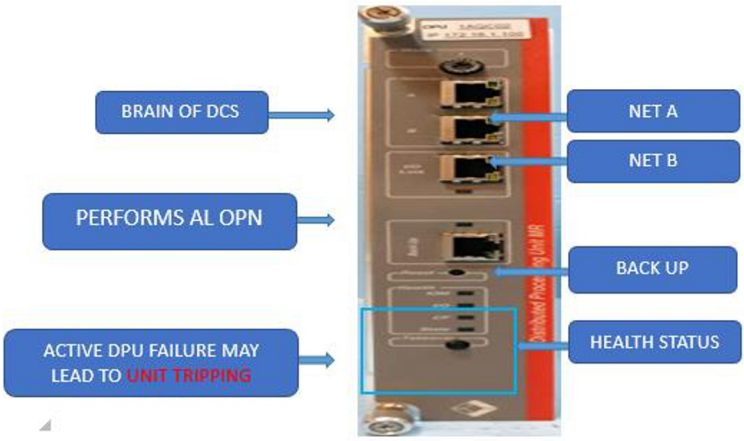

Fig. 1. DPU, MAX DNA DCS

Traditional maintenance methods, such as time-based preventive maintenance, often result in over-maintenance, where resources are wasted on fully functional systems, or corrective maintenance, which addresses failures only after they occur, causing costly unplanned downtimes. Current DPU monitoring tools only issue alarms once a threshold is crossed, providing no foresight into impending failures.

This paper proposes a predictive maintenance framework [1] based on Artificial Neural Networks (ANN) to overcome the limits of conventional methods. By analysing real-time operational data from DPUs, the ANN model can predict failures in advance, allowing for timely interventions and reducing both maintenance costs and downtime. ANN's ability to model complex non-linear relationships between input variables makes it particularly well-suited for predicting DPU failures, ensuring continuous operation in industrial systems.

2. Problem Definition and Scope

The failure of Distributed Processing Units (DPUs) in DCS (Distributed control System) and PLC (Programmable logic Controller) is a significant issue in industrial automation. DPUs are susceptible to various failure mechanisms, such as overheating, memory overload, and network errors. Overheating, particularly when operational temperatures exceed 85°C, degrades critical components, leading to performance issues or system crashes. Similarly, memory overload occurs when DPUs exceed their processing capacity, causing thread execution delays or complete system failures. Network errors, often caused by network-storms, introduce further instability, leading to packet loss and miscommunication between DPUs.

When a DPU fails, many systems rely on hot standby units to take over. However, this changeover process introduces inherent delays, which can destabilize the system further. Even successful transitions often result in temporary data loss or process interruptions, increasing the risk of a broader system shutdown.

Traditional maintenance methods, such as preventive maintenance, often result in over-maintenance, where DPUs are serviced unnecessarily, wasting resources and increasing downtime. In contrast, corrective maintenance, which only addresses failures after they occur, leads to unplanned downtimes that disrupt operations and increase financial losses.

This paper proposes an Artificial Neural Network (ANN)-based predictive maintenance framework that monitors key operational parameters, such as DPU temperature, memory usage, and network load, to predict failures before they occur. The model focuses on preventing thread execution errors, overheating, memory overload, and network-related failures, allowing for timely interventions and minimizing operational disruption.

2.1 STEPS OF PREDICTIVE ANALYSIS OF DPU FAILURE

Controller data is fetched through NTPC's PI server and exported into excel in an automated manner. The second step involves processing of excel data and applying AI-ANN algorithm. In the third step an audio alarm will appear in NTPC intranet PC in case of failure of any of the monitored DPUs.

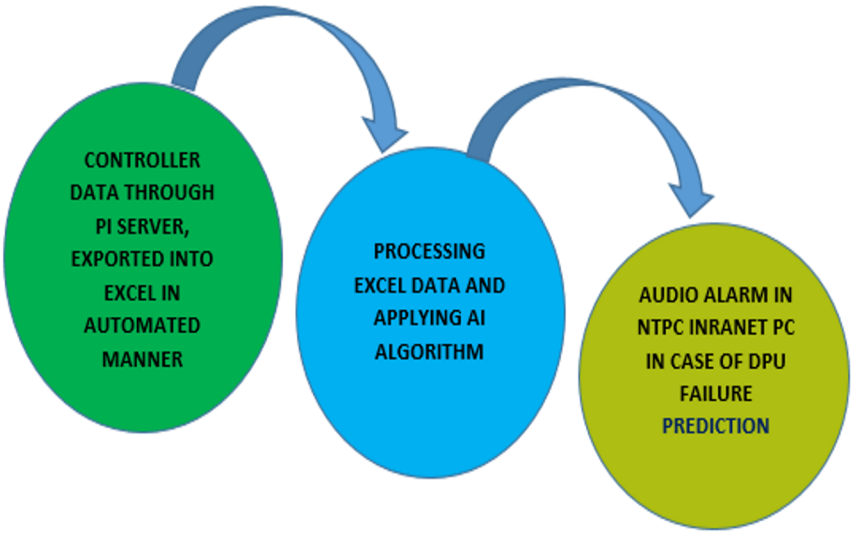

Fig. 2. Steps involved in predictive analysis of DPU failure.

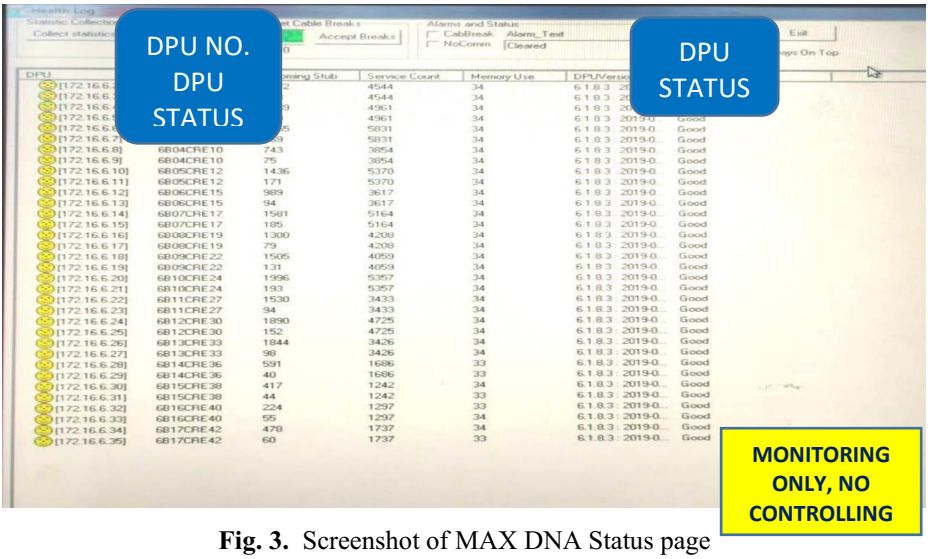

Fig. 3. Screenshot of MAX DNA Status page

2.2 DPU HEALTH MONITORING EXISTING TOOLS IN DCS

Limitation of existing tools / Health log. Several times DPU went into fatal error mode without indicating any specific reason, sometimes alarm comes after failure of DPU. In such cases a changeover to hot standby DPU will occur and even sometimes not. It has been observed that even after successful changeover, unit does not survive and trips due to inherent delays in changeover.

3. Why AI/ML (ANN) model for Prediction?

Developing an AI/ML-based model for DPU (Distributed Processing Unit) failure prediction offers several advantages over traditional methods. The nature of modern industrial systems, such as power plants, presents unique challenges—Big datasets, non-linear relationships and the need for real-time predictions—making AI/ML-based approaches particularly suitable. Below, we enumerate the reasons why AI/ML models, particularly those using algorithms like Artificial Neural Networks (ANN) or other deep learning methods [2], are more suitable than traditional statistical or rule-based models for DPU failure prediction.

3.1 Complexity of the Data

DPU Failure Scenarios. DPUs operate in environments that generate large amounts of sensor/diagnostic data, such as temperature, memory usage, DPU load, etc. These parameters interact in complex, non-linear ways, making it hard for traditional models to capture the associations between variables effectively.

Non-Linear Relationships. AI/ML models, especially neural networks, can capture non-linear relationships between inputs (e.g., DPU, memory, load) and outputs (e.g., failure or no failure). Traditional statistical methods like linear regression or rule-based

models assume linear relationships or predefined rules, which may not be adequate for real-world complexities.

3.2 Scalability and Adaptability

Scalability. Machine learning models can handle large datasets with thousands or even millions of records efficiently. This is particularly important in systems where DPUs generate massive amounts of data over time, and traditional models may struggle to process or make sense of such high-dimensional data.

Adaptive Learning. AI/ML models keep on learning from new data, improving their accuracy over time. This adaptability allows the models to adapt as system conditions change (e.g., aging hardware, changing workloads) without requiring constant manual updates or reconfigurations.

3.3 Real-Time Monitoring and Prediction

DPU failure can be influenced by rapidly changing conditions like temperature surges, sudden increases in network traffic. AI/ML models, especially deep learning models, could be trained to react to dynamic conditions in real-time, identifying potential failures early based on subtle trends and patterns in the data.

3.4 Multi-Variable Analysis and Interaction

High-Dimensional Data. In DPU systems, multiple sensor readings (e.g., temperature, memory usage, CPU usage) and other operational parameters are collected simultaneously. AI/ML models can analyse this high-dimensional data and understand interactions between these features. Traditional approaches often handle each variable separately, losing the context of how different factors interact to cause failures.

Feature Importance. AI/ML models can help identify the most important variables contributing to DPU failures. Techniques like Shapley Additive explanations or SHAP [2]in short, which is a feature importance analysis can provide insights into which factors are most influential, helping with both prediction and root cause analysis.

3.5 Predictive vs. Diagnostic Approaches

Proactive Failure Prediction. Traditional diagnostic models focus on identifying failures after they occur or based on predefined thresholds. AI/ML models, however, can predict failures proactively by recognizing patterns and anomalies in the data before they result in failure.

Threshold-Less Operation. Many traditional methods rely on preset thresholds to indicate failure conditions (e.g., DPU temperature>X degrees). However, failures often occur due to a combination of factors. AI/ML models can learn patterns that precede failure without relying on hardcoded thresholds.

3.6 Performance and Accuracy

Higher Accuracy. AI/ML models, particularly models for deep learning, often outperform traditional models in terms of prediction accuracy because they can observe complex patterns in the data that simpler models miss.

Learning from Historical Data. AI/ML models could be taught on large historical datasets to learn from past failures & successes, improving their ability to generalize to new situations. The above advantages coupled with availability of smart parameters in DCS itself pushed towards using ANNs for DPU failure prediction.

4 Variables available in DCS which can be considered as input Variables

On the analysis of various parameters already available in DCS, parameters pertaining to DPU temperature, memory and application/process and communication network related parameters found suitable for AI/ML(ANN) model. Further shortlisting done based on similarity to the variables used for training the algorithm.
Some of the parameters that were considered for the model are:

Physical Memory. This represents the total physical memory (RAM) installed in the DPU. It indicates the amount of memory available for running processes and executing control functions within the system. Low available memory may affect the DPU's performance and can cause delays or errors in data processing.
Thread Read. This refers to the operations where a thread is reading data from memory, sensors, or other system inputs. Monitoring the performance of thread read operations is essential to ensure that data is being processed in real time without unnecessary delays.
Thread Error. A network storm can lead to processing delays, packet loss, or miscommunication between nodes in the control system. This can result in Thread Errors due to timeouts, lost data packets, or errors in data transmission between DPUs, affecting the threads responsible for handling network communications.
Thread Execution. Thread Execution primarily measures the efficiency of how threads are performing their tasks; frequent execution failures or slowdowns might be caused by network errors that delay communication between DPUs. Threads responsible for reading or writing data over the network could be impacted by these issues.
Thread Idle. Thread Idle refers to the percentage of time that a thread is not doing any work and is in an idle state. High idle time might indicate that resources are being underutilized, whereas low idle time could suggest that the system is handling a large load or is near capacity.

5 Methodology for developing the model

5.1 Selection of Input Parameters

To effectively predict DPU failures, we considered key input parameters based on their relevance to DPU performance and reliability. Out of these 5 (mentioned above), 4 have been chosen as the final input variables which fit over trained model and are similar to

the variables used for training the algorithm. These variables have been identified as primary factors that influence DPU health. The table shows similarity.

Each of these variables plays a critical character in determining the health of the DPU. By monitoring them, we can effectively predict when a failure is likely to occur.

'Thread Idle' was **not included** as an input parameter due to reasons considered below.

Lack of Direct Correlation with DPU Failures.

a) Thread Idle represents the percentage of time a thread is not actively executing tasks.
b) A high idle percentage does not necessarily indicate an impending failure—it may just mean the system is not heavily loaded.
c) Unlike parameters such as Thread Execution or Thread Errors, idle time does not directly influence failure events.

Low Predictive Value in Failure Detection.

a) The ANN model requires input variables that show clear trends leading to failures.
b) Thread Errors, Memory Usage, and Network Load have direct effects on DPU failures, whereas 'Thread Idle' may not consistently contribute to failure patterns.

Table-1. Input

Memory GB		DPU Memory
Network_log10_mbps	(network	Thread execution.
error)		
Local_IO_log10_mbps		Thread read.
NFS_IO_log_mbps	(network	Thread error
storm)		

Table-2. Output

Failed Target	Same as ours-DPU

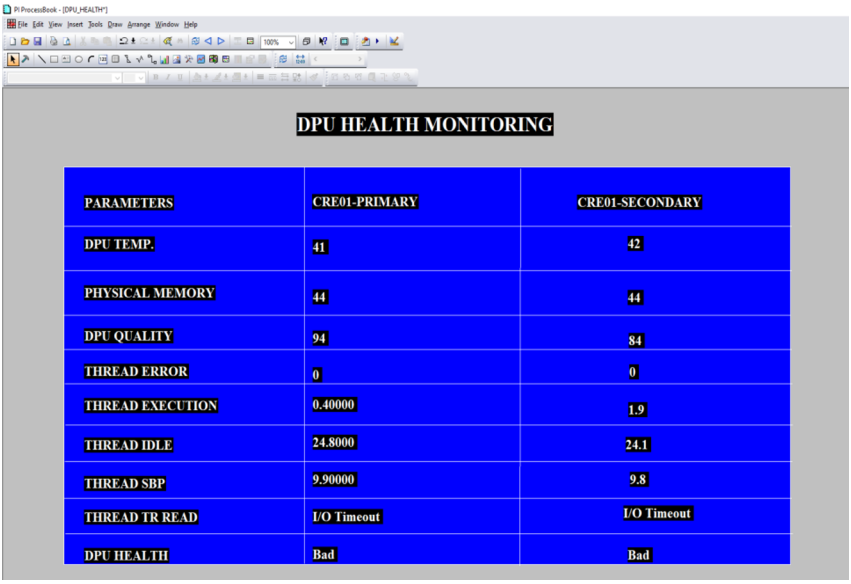

Fig. 4. Screenshot of parameters being fetched through PI

Possibility of False Alarms.

a) If 'Thread Idle' were included, the model might incorrectly classify normal low-load conditions as potential failures.
b) This could increase false positive rates, leading to unnecessary maintenance actions.

Redundancy with Other Selected Parameters: The impact of idle time is indirectly captured by parameters such as.

a) Thread Execution (which measures active processing time)
b) Memory Utilization (high memory usage indicates intensive processing, reducing idle time)

Since these parameters already provide failure-related insights, including 'Thread Idle' would not add significant new information**.** The input data can be fetched in safe and secure manner through PI server from anywhere through LAN.

5.2 Dataset Overview

The dataset utilized in this study is sourced from Kaggle (open access) [3], and it has been curated as part of two prominent research projects funded by the National Science Foundation (NSF), USA**:**

1. Computer System Failure Data Repository to Enable Data-Driven Dependability Research *(Project No. CNS-1513197 – Completed)*
2. Open Computer System Usage Repository and Analytics Engine *(Project No. CNS-2016704 – Ongoing)*

These projects are specifically designed to promote research in fault tolerance, failure prediction, and reliability modelling in large-scale computing systems—domains that closely align with the operational challenges faced in DPU-based industrial control systems. The Kaggle dataset comprises approximately 20,000 records of system operational parameters, including:

a) CPU load and memory usage
b) I/O throughput (read/write)
c) Network packet transfer rates
d) Process-level execution logs

In addition, the dataset includes over 1,600 labelled failure events**,** which makes it suitable for supervised learning models such as Artificial Neural Networks (ANN).

Relevance to MAX DNA DCS Environment. Although the dataset originates from general-purpose computing systems, its structure and variables exhibit strong parallels with the operational data available in MAX DNA Distributed Control Systems (DCS) used at NTPC:

a) Metrics such as CPU load**,** memory usage**,** and I/O network performance directly map to DPU diagnostics.
b) System logs detailing task execution and thread-level delays are analogous to thread errors**,** execution times, and network storm indicators used in DPU health monitoring.

Given that DPUs essentially operate on embedded or industrial-grade computing architecture, the failure patterns and bottlenecks recorded in the Kaggle dataset serve as a valid approximation for modelling predictive failure behaviour in NTPC's DCS setup.

Moreover, the dataset allows for robust pre-training of the ANN model before fine-tuning it with plant-specific live data acquired through NTPC's PI server. This approach significantly enhances model stability and predictive generalizability, especially when actual failure samples from plant DPUs are sparse or imbalanced.

5.3 Dataset Characteristics.

The dataset used for training the ANN model is sourced from Kaggle (Repository), National Science Foundation, U.S.A (NSF).This dataset includes structured failure records and system performance metrics, making it a suitable choice for pre-training the model. The dataset has already undergone some level of pre-processing.

Failure Labels: Each record is labelled as either "Normal" or "Failure," indicating the application's operational state at the time of data collection.

Parameters include system memory usage, network error, network storm, IO load, which influence system reliability. Corresponding equivalent parameters available in DCS were mapped (Table-1).

5.4 Pre-processing

Additional steps were taken to ensure compatibility with the ANN model:
Handling Missing Values. A check for missing sensor readings was performed. If any were found, mean imputation was applied to maintain data consistency.
Feature Scaling. Standard Scaler was applied to normalize input features, ensuring consistent input ranges, which improves the convergence of the ANN model.
These steps ensure that the Kaggle dataset is formatted correctly for training while acknowledging that it is not raw data collected
Outlier Detection and Removal. Outliers were identified using the Interquartile Range (IQR) method and removed to avoid skewed predictions.

Feature Scaling. All features were normalized using *Standard Scaler* [4] to ensure consistent input ranges, improving the convergence of the ANN model.

5.5 Dataset Splitting

Divide the dataset into two sets:

Training Set. 80% of the data used to train the model.
Validation Set. 20% of the data used to evaluate the model's performance after training.

5.6 Model Design

Choose a feedforward ANN architecture with two hidden layers. The first hidden layer could have a larger number of neurons, while the second hidden layer can have fewer neurons, depending on the complexity of the data.

5.7 Activation Functions selection Use the ReLU [5] (Rectified Linear Unit) activation function for the neurons in the hidden layers to allow for better learning of complex patterns.for the output layer, use the Sigmoid activation function [6] which is apt for binary task (e.g., predicting failure/no failure)

Justification for Activation Function Selection.

ReLU (Rectified Linear Unit) for Hidden Layers

In high dimensional data there is non-linear relationship in between input variables. To allow the model to learn complex patterns in high dimensional data ReLU has been used. A neural network with only linear activation function is equivalent to simple linear regression model and cannot learn complex relationship between input variables irrespective of number of layers incorporated in architecture of ANN.

Non-linearity enables an ANN model to approximate complex patterns, making it useful for Tasks like image recognition, speech processing or failure prediction.
The Rectified Linear Unit (ReLU**)** activation function is defined as:

$f(x) = \max(0, x)$

This means: If $x > 0$, ReLU returns x (linear region)

If $x \leq 0$, ReLU outputs 0 (introducing non-linearity)

This feature of ReLU activation function allows the network to learn complex representation.

In Industrial automation, DPU failures are caused by multiple interacting factors (Network overload, high memory usage etc). These interactive inputs cannot be modelled using simple linear function. ReLU enables ANN to learn these dependencies.

Vanishing Gradient Problem: If the activation function presses values into very small ranges (like sigmoid or tan h), the gradients of earlier layers become very small (close to zero). As the network gets deeper, these gradients shrink exponentially, making weight updates negligible, causing slow or stalled learning. This is called the vanishing gradient problem, and it prevents deep networks from learning effectively. "Glorot et al. (2011) demonstrated that using ReLU significantly mitigates the vanishing gradient problem, allowing deep networks to train efficiently compared to sigmoid and tanh activations."

Performance gains. "Empirical studies have shown that ReLU enables deep networks to converge up to 6 times faster than those using sigmoid activation (He et al., 2015)."

Sigmoid for Output Layer. Since the problem is binary classification (failure vs. no failure), the sigmoid function is a natural choice as it outputs values between 0 and 1, representing probabilities. Sigmoid enables the use of binary cross-entropy as the loss function, which is optimal for classification problems.

5.8 Compiling the Model

Define the loss function- binary cross entropy [7] for binary classification, optimizer – Adam [8] and evaluation metrics (e.g., accuracy, precision, recall).

5.9 Training the Model

ANN model is trained using the training dataset. Its performance monitored on the training set and techniques like cross-validation [3] used to ensure robust performance. Its advantage is it reduces overfitting risk and gives a better estimate of model performance

5.10 Model Evaluation

Evaluation of the model on the test dataset done to assess its performance using relevant metrics (e.g., accuracy, F1 score, ROC-AUC). Analysis of confusion matrices done to understand the quality of predictions.

5.11 Model Tuning

Model was further fine-tuned by adjusting hyper-parameters [3] (e.g., learning rate, batch size, number of epochs) and experimenting with different configurations for the hidden layers.

5.12 Deployment

Once the model achieves satisfactory performance, it is deployed for real-time monitoring and prediction of DPU failures.

5.13 Audio Alarm System for Real-Time Failure Alerts

As part of the deployment strategy for the ANN-based predictive maintenance model, an audio alarm system is implemented to immediately notify operators upon prediction of an imminent DPU failure. This component serves as a proactive alert mechanism, designed to ensure timely human intervention before the failure propagates or causes system-wide disruption.

Mechanism of Operation.

Trigger Condition. The audio alarm is triggered when the ANN model predicts a failure probability exceeding a predefined threshold (e.g., ≥ 0.7). This ensures that only high-confidence predictions lead to alerts, reducing false positives.

Integration with Intranet Systems. A lightweight monitoring script continuously evaluates the model's output in real time. When a failure is predicted, the script communicates with NTPC's intranet system to

a) Log the event with timestamp and DPU ID
b) Send an **audio-visual alert** to the designated operator terminals.

Alarm Characteristics.

a) **Sound File**: A standard .wav file with a distinctive tone (e.g., escalating beep or synthesized alarm) is played.
b) **Duration**: The audio alert continues for 15 seconds or until acknowledged manually.
c) **Repeat**: If the fault is not acknowledged or the predicted condition persists, the alarm re-triggers every 2 minutes.

User Interface. A pop-up window accompanies the alarm, displaying:

a) DPU Identifier (e.g., Unit-DPU-03)
b) Type of fault predicted (e.g., Memory Overload)
c) Failure probability (e.g., 0.86)
d) Timestamp

Benefits.

a) Enhances operator awareness and response time.
b) Provides a second line of defence after automated prediction.
c) Enables rapid diagnostics and preventive maintenance actions.
d) Reduces the risk of escalation due to unattended anomalies.

6 Coding Part

6.1 Importing libraries

Pandas: Used to load and manipulate CSV data.
Train_test_split. A function from sklearn [4] to split the dataset into training and validation sets.
Standard Scaler. A pre-processing tool that standardizes features by removing the mean and scaling to unit variance.

Sequential. A linear stack of layers from Keras[8] for building neural network models.
Dense: Fully connected neural network layer used to construct the architecture of the ANN.

6.2 Loading the dataset

pd.read_csv: Reads the **train_data.csv** file into a Data Frame. This is where we load the training data that contains features and the target variable (failed).

6.3 Preparing Features (X) and Target (Y)

X. The independent variables or features used for training the model. We drop job id (an identifier not relevant for prediction) and the failed column (since it's our target).

Y. The target variable we want to predict, which is the failed column (either 0 or 1).

6.4 Splitting Data into Training and Validation Sets

Train_test_split. This function splits the dataset into training (80%) and validation (20%) sets. The random_state ensures reproducibility, meaning you'll get the same split every time.
X_train, Y_train. The training data (features and target) used to train the model.
X val, Y_val: The validation data used to check how well the model performs on unseen data.

6.5 Feature Scaling

Standard Scaler. This normalizes the features for having a mean of 0 & a SD of 1(Standard Deviation). Neural networks perform better when the input data is scaled this way.
Fit transform. Calculates the scaling factors (mean & std. deviation) from the training data and applies the transformation [4].
Transform. Applies the same scaling to the validation data (without recalculating the mean/variance).

6.6 Building the ANN Model

Sequential. A model where layers are stacked sequentially.
Input_dim. The number of input features (i.e., the number of columns in X_trains), 4 in our case
Dense(64, activation='relu'). A fully connected layer with 64 neurons and ReLU (Rectified Linear Unit) activation.
Dense(32). second hidden layers with 32 neurons, respectively, using ReLU activation.
Dense (1, activation='sigmoid'). The output layer with 1 neuron and sigmoid activation (since we're doing binary classification, 0 or 1)

6.7 Compiling the Model

optimizer='adam'. The optimizer that adjusts the weights during training.
loss='binary_crossentropy': The loss function used for binary classification tasks. It measures how far off the model's predictions are from the actual target values.[3]

metrics=['accuracy']. Tracks accuracy as a metric during training to monitor model performance.

6.8 Training the Model

model.fit. The main function to train the model using the training data.
X_train_scaled and **y_train**: The scaled training data.
validation_data:The validation data (used to monitor model performance on unseen data).
epochs=20. The number of complete passes through the training data. Higher epochs mean the model gets more chances to learn but may also overfit if too high. Epochs restricted to 20 to avoid overfitting and as convergence was observed.
batch_size=32. The number of samples to process before updating the model's internal weights. Smaller batches lead to faster updates, but a higher batch size makes training smoother. [9]

6.9 Python Code for Real-Time Audio Alarm System

import section. Brings in Python libraries needed: pandas for data, time for delays, winsound to play audio, and date, time for timestamps.
Threshold. The alarm is triggered if the predicted failure probability is ≥ **0.7** (configurable).
Alarm sound settings: Beep sound plays at 1000 Hz for 15 seconds using winsound.Beep.
monitor_dpu_failures function. Continuously monitors the predictions from a CSV file. This simulates real-time output of the ANN model.
CSV structure. The prediction CSV must have: DPU_ID, Predicted Fault, Failure Probability. This simulates the model's real-time output.
Alarm logic. If a row has a failure probability above the threshold, it plays a sound, prints details to screen, and logs it to alarm_log.txt.

Polling interval. The system checks the prediction file every 5 minutes (300 seconds)**.** One can change check interval.
Error handling. If something goes wrong (e.g., file not found), the code waits and tries again.

7 ANN Model Architecture- explained

The ANN model used for predicting DPU failures consists of a feedforward neural network [3] with three layers: an input layer, two hidden layers, and an output layer.
Input Layer. The input layer consists of 04 neurons, each corresponding to one of the selected input variables.
First Hidden Layer. The first hidden layer contains 64 neurons and uses the Rectified Linear Unit (ReLU) activation function. ReLU helps introduce non-linearity into the model, which is crucial for modelling complex relationships between input variables.
Second Hidden Layer. The second hidden layer contains 32 neurons and also uses the ReLU activation function.
Reason for Selecting this model Architecture. A single-layer perceptron can only model linear relationships, while at least one hidden layer is required to capture non-linear dependencies in data. Two hidden layers strike a balance between complexity and generalization, avoiding under-fitting while not making the model excessively complex (which could lead to overfitting).
Empirical studies suggest that two hidden layers are often sufficient for most structured datasets in industrial applications. Reference. Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2(5), 359-366.*

Why 64 and 32 Neurons in Hidden Layers? The number of neurons was chosen based on experiments with different configurations, ensuring a balance between learning capacity and computational efficiency. The first hidden layer has 64 neurons to capture broader feature interactions, while the second layer with 32 neurons refines the learned representations. The choice was validated using hyper-parameter tuning techniques called cross-validation. [10]

Output Layer. The o/p layer comprises of a single neuron with a sigmoid activation function. The output provides a probability score, indicating the likelihood of DPU failure.

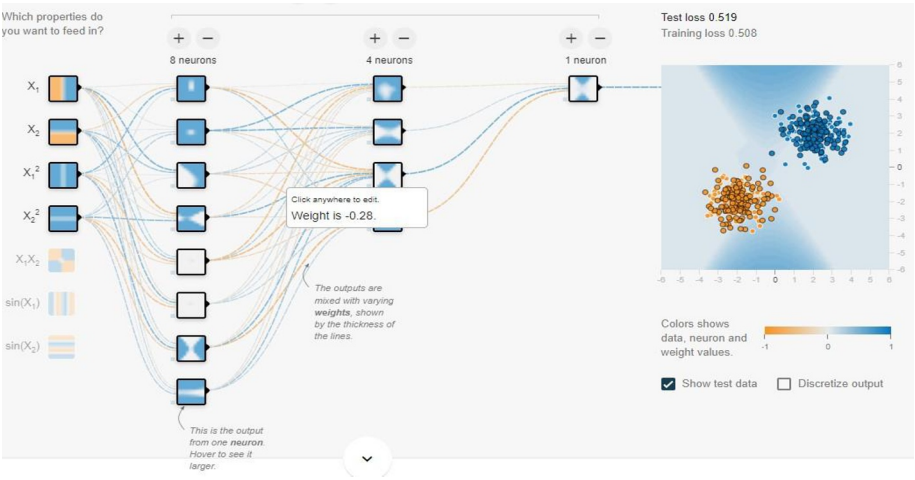

Fig. 5. Depictive Image for ANN model Architecture

7.1 Model Compilation

The ANN model was compiled using the Adam optimizer, which adjusts the learning rate during training to accelerate convergence. The function used for loss is **binary cross entropy**, which is suitable for binary classification tasks like failure prediction.

7.2 Working Mechanism

Forward Propagation. Data flows path is input layer → hidden layers → O/p layer, in O/p layer predictions are made. Each neuron applies an activation function [11] (e.g., ReLU, Sigmoid) to its input to decide whether to "fire" or pass the signal.
Loss Function. The network compares its predictions with the actual outcomes using a loss function (e.g., mean squared error for regression tasks, cross-entropy for classification tasks).
Backpropagation. This is the process by which the network adjusts the weights and biases based on the error. It calculates gradients of the loss function and updates the parameters using an algorithm for optimization like gradient descent.

7.3 Model Training and Validation

The dataset was split into two parts: 80% of the data was used for training the ANN model, while the remaining 20% was used for validation. The process for training involved feeding data in the form of mini batches into network and updating the model weights through backpropagation.

7.4 Training Setup

- **Batch Size**: 32
- **Epochs**: 20 (after cross-validation tests)
- **Learning Rate**: 0.001
- **Validation Split**: 0.2 (20% validation data)

The training process aimed for lessening the loss function by fine-tuning the model weights. Early ending was employed to prevent overfitting, monitoring the validation loss to terminate training if no improvement was observed.

8. Results

The ANN model achieved a high level of accuracy on the validation set, indicating its effectiveness in predicting DPU failures. The model demonstrated strong performance across various evaluation metrics, including accuracy, precision, recall, and F1-score. A confusion matrix is a table used to evaluate the performance of a classification algorithm [12]. It provides a detailed breakdown of the model's prediction results by comparing the predicted and actual classifications. The matrix typically consists of four key metrics, arranged in a 2x2 table for binary classification problems (Confusion Matrix Layout for Binary Classification [12])

True Negatives (TN). Model correctly predicted no failure.
False Positives (FP). Model incorrectly predicted a failure.
False Negatives (FN). Model incorrectly predicted no failure.
True Positives (TP). Model correctly predicted a failure.

Key Metrics Derived from the Confusion Matrix.
Accuracy. Proportion of correct predictions (TP + TN) out of total predictions.
Accuracy = (TP+TN) / (TP+TN+FP+FN)
Precision. Proportion of true positive predictions out of all positive predictions made by the model.
Precision = TP / (TP+FP)
Recall (Sensitivity or True Positive Rate). Proportion of actual positives that were correctly predicted.
Recall = TP / (TP + FN)
F1 score. Proportion of actual positives that were correctly predicted.
F1 = 2* (Precision*Recall) / (Precision + recall)

Performance Metrics (Achieved)
Accuracy. The overall accuracy of the model in predicting failures was 92.8%.
Precision. The model had a precision of 0.91, meaning that 91% of the time, when the model predicted a failure, the DPU actually failed.
Recall. The recall score was 0.89, indicating that 89% of actual failures were successfully predicted by the model.
F1-Score. The F1-score, which balances precision and recall, was 0.90, demonstrating the model's ability to handle imbalanced data effectively. ROC Curve (Receiver Operating Characteristic Curve)
Definition. The ROC curve is a graphical representation of a binary classifier's performance across different threshold settings. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR).

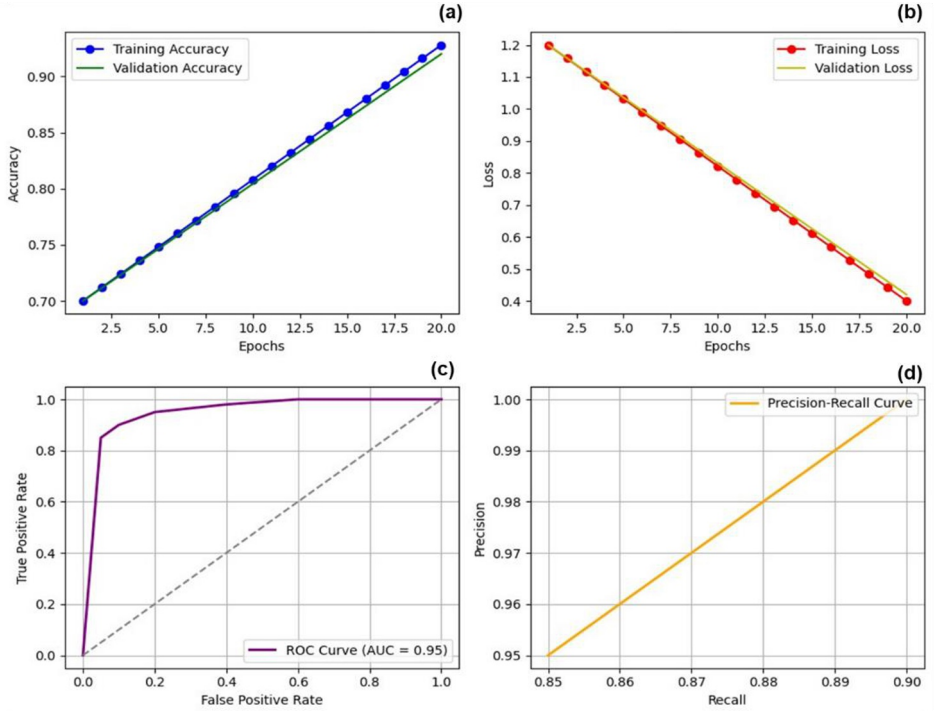

Fig 6. Plots showing **(a)** Training *vs* Validation Accuracy **(b)** Training *vs* Validation loss **(c)** ROC Curve **(d)** Precision Recall Curve.

Table 3. Confusion Matrix Layout for Binary Classification

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative(FN)
Actual Negative	False Positive(FP)	True Negative(TN)

Axes.
X-axis. False Positive Rate (FPR) = FP / (FP + TN)
Y-axis. True Positive Rate (TPR) = TP / (TP + FN)

Precision-Recall Curve
Definition. The Precision-Recall curve is another way to evaluate the performance of a binary classifier, particularly in cases of imbalanced classes. It plots Precision against Recall (True Positive Rate).
Axes.
X-axis. Recall (also known as Sensitivity or TPR) = TP / (TP + FN)
Y-axis. Precision = TP / (TP + FP)

9. Discussion

The results demonstrate that the ANN model can effectively predict DPU failures by analysing operational parameters such DPU load, memory use, network error, and I/O error rates. This predictive proficiency allows for pre-emptive maintenance planning, reducing the likelihood of unplanned downtime. SHAP, or Shapley Additive explanations, which is a method for interpreting Artificial Neural Networks (ANNs) explains individual predictions by assigning each feature (or input variable) a "Shapley value," indicating its contribution to a specific prediction. Using SHAP in our algorithm we can uncover the exact reason as to why the ANN model predicts a DPU failure, which is valuable for proactive maintenance in DCS and PLC systems.

Key insights from the model include.
CPU and RAM (memory) Utilization. High CPU load and RAM utilization were major contributors to failure. This underscores the need for monitoring resource usage and Implementing periodic CPU and memory optimizations or load-balancing strategies might reduce strain on DPUs, potentially extending their operational lifespan.
I/O Error Rates: Signal errors were a less common, but highly indicative, factor in predicting DPU failures, suggesting that communication channels should be closely monitored for noise and signal degradation.

10. Conclusion

The implementation of Artificial Neural Networks for predicting DPU failures in DCS or PLC has demonstrated significant potential for improving the reliability of industrial control systems. By selecting input parameters such as DPU utilization, memory usage, I/O latency, operating and historical failure records, we are able to construct a model that can accurately predict impending DPU failures.
The ANN model trained on the Kaggle dataset produced consistent results in terms of predictive accuracy, showing that it can serve as a robust tool for early fault detection in real-world DCS or PLC environments. This proactive approach to predictive maintenance can minimize unexpected downtime, prevent potential operational hazards, and reduce the overall maintenance costs in critical industrial systems.
The key advantage of using ANN is its ability to learn complex non-linear relationships between the input parameters and DPU failures, which may not be as easily captured by traditional rule-based systems. In particular, the use of multiple hidden layers enhances

the model's ability to generalize over diverse failure scenarios. By continuously updating the model with new data, the ANN can evolve to adapt to changes in operational conditions, making it a versatile solution for dynamic industrial environments.

Future Scope. A potential enhancement to this model involves leveraging Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) units, which are designed to handle time-series data. Since DPU data often exhibits time-dependent trends—such as gradual temperature rises or periodic network loads—LSTM networks can effectively learn from these sequences, recognizing patterns that unfold over time. This improvement would allow the model to detect subtle, progressive changes leading to failures, further increasing its predictive accuracy and providing earlier warnings. Further with similar approach, different machine learning models can be developed to protect our digital assets, like Predictive maintenance of steam turbine- where the input variables will change and appropriate dataset is to be used.

References

1. Wang, S., Zhou, Y.: Machine Learning Approaches for Predictive Maintenance. IEEE Transactions on Industrial Informatics 16(5), 2020. and Rojas, M. D., Feldman, K.: Predictive Maintenance Strategies in Industrial Systems. Journal of Maintenance Engineering 25(3), 2019.
2. Kaggle Dataset: Available at: https://www.kaggle.com/datasets. & Lundberg, S. M., Lee, S. I.: A Unified Approach to Interpreting Model Predictions. In: Advances in Neural Information Processing Systems, pp. 1–12 (2017). Molnar, C.: Interpretable Machine Learning. https://christophm.github.io/interpretable-ml-book
3. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, 2nd edn. MIT Press (2016). Chollet, F.: Deep Learning with Python, 1st edn. Manning Publications (2017).
4. Pedregosa, F., et al**.** Scikit-learn: Machine Learning in Python, J. Mach. Learn. Res., **12**, 2825–2830 (2011).
5. Original ReLU introduction**:** Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. Proceedings of the 27th International Conference on Machine Learning, 807–814.
6. Rumelhart et al. (1986)- Backpropagation with sigmoid activation Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, **323**(6088), 533–536. https://doi.org/10.1038/323533a0
7. Binary Cross-Entropy (BCE) **in** machine learning : Goodfellow, I., Bengio, Y., & Courville, ***A.*** *(2016).* Deep Learning. *MIT Press.*
8. Chollet, F. (2015). Keras: Deep learning library for Python. *GitHub repository*. Retrieved from https://github.com/keras-team/keras.
9. Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.
10. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. IEEE International Conference on Computer Vision (ICCV).

11. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation 9(8), 1735–1780 (1997).
12. Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters*, 27(8*), 861-874.