Predictive Maintenance for Smart Factory Equipment

Najia Mustafa, Soomin Shin, Talha Ansari
University of Waterloo, Canada
Email: {n5mustaf, soomin.shin, t2ansari}@uwaterloo.ca

***Abstract*—Predictive maintenance in smart factories involves forecasting equipment failures before they occur, minimizing downtime, and optimizing maintenance schedules. This approach reduces unexpected downtime, cuts maintenance costs, extends equipment lifespan, and enhances safety. Smart factories generate vast amounts of sensor data, offering opportunities to predict failures and optimize maintenance. We propose a machine learning model using Long Short-Term Memory (LSTM) networks to predict Time-to-Failure (TTF) based on simulated sensor data from a smart factory environment. We utilize OMNeT++ to simulate realistic sensor data, including normal and failure scenarios, and employ feature engineering in Python to extract indicators of equipment health. Our model leverages LSTM's ability to capture temporal dependencies in sequential data, ensuring accurate TTF predictions. The model is rigorously tested and fine-tuned for accuracy and reliability, featuring a realistic simulation environment, targeted feature engineering, and advanced LSTM networks. Additionally, we implemented a real-time monitoring system that continuously collects, processes, and analyzes sensor data, providing immediate insights and enabling timely interventions to prevent equipment failures.**

***Keywords* - Predictive Maintenance, Time-to-Failure (TTF), Long Short-Term Memory (LSTM), Smart Factories, Sensor Data, OMNeT++, Feature Engineering, Machine Learning, Time-Series Prediction, IoT Sensors, Data Fusion**

I. Introduction

Industry 4.0 has revolutionized manufacturing with smart factories equipped with advanced sensors and interconnected systems, paving the way for predictive maintenance. This strategy anticipates equipment failures, minimizing downtime and optimizing maintenance schedules. Predictive maintenance reduces unexpected downtime, lowers costs, extends equipment lifespan, and enhances safety by preventing risky failures.

The core of predictive maintenance is leveraging vast amounts of sensor data to identify patterns and indicators of impending failures, enabling proactive maintenance actions. However, developing accurate and reliable predictive models that handle complex and variable sensor data is challenging.

Our project proposes a machine learning model to predict the Time-to-Failure (TTF) of equipment using simulated sensor data. We use OMNeT++ to generate realistic sensor data, including normal operation and failure scenarios. This data is processed and analyzed in Python, with feature engineering techniques extracting meaningful indicators of equipment health and performance.

We chose Long Short-Term Memory (LSTM) networks for our predictive maintenance model due to their effectiveness in capturing temporal dependencies in sequential data. LSTM networks are ideal for time-series prediction tasks, retaining information over long sequences and handling complex temporal dynamics. The model is rigorously tested and fine-tuned for high accuracy and reliability.

Our approach combines a realistic simulation environment, focused feature engineering, and advanced LSTM networks to predict the Time-to-Failure (TTF) of equipment in smart factories, aiming to improve industrial operations' efficiency and reliability.

The rest of this paper is structured as follows: Section II reviews related work in the field of predictive maintenance. Section III describes the methodology, including the design and implementation of the simulation environment. Section IV presents the detailed LSTM-based neural network training. Section V discusses the real-time monitoring system for alerting machine failures while the network simulation is running. Finally, Section VI concludes the paper with a summary of accomplishments and recommendations for future work.

II. Related Work

There has been significant research in the field of predictive maintenance (PdM), particularly focusing on the integration of IoT, machine learning applications, and the associated challenges. This section reviews existing literature on PdM, highlighting key developments and future directions.

A. Deep Learning Models for Predictive Maintenance: A Survey, Comparison, Challenges, and Prospect

Research highlights the importance of vibration sensors in PdM, which provide critical insights into the operational status of machinery. Vibration analysis can

identify anomalies that indicate wear and tear or imminent failure [1]. Similarly, the use of wireless sensor networks enhances the scalability and flexibility of PdM systems, allowing for widespread deployment across large industrial facilities.

The integration of IoT in PdM systems involves deploying sensors that continuously monitor equipment health by collecting data on parameters such as vibration, temperature, and pressure. This data is transmitted to a centralized system for analysis, enabling early detection of potential failures, which reduces downtime and maintenance costs. Serradilla et al. (2022) [1] underscores the importance of vibration sensors in PdM, which provide critical insights into machinery's operational status. Vibration analysis can identify anomalies that indicate wear and tear or imminent failure.

a) Machine Learning Applications: Machine learning (ML) techniques are crucial for analyzing the vast amounts of data generated by IoT sensors. Various ML models, including deep learning, are employed to predict equipment failures. Studies demonstrate the effectiveness of recurrent neural networks (RNNs) and convolutional neural networks (CNNs) in PdM applications. RNNs are particularly useful for time-series data analysis, enabling the prediction of future equipment states based on past behavior. Deep learning models can automatically extract relevant features from raw sensor data, enhancing prediction accuracy without extensive manual feature engineering.

B. Predictive Maintenance in the IoT Era

This paper [2] discusses how predictive maintenance in the IoT era integrates machine learning with streaming sensor data to maintain machinery before failure, optimize resources, and reduce unplanned downtime. It outlines the fundamental concepts of predictive maintenance programs and highlights the implementation of real-time data processing to foresee and prevent equipment failures.

Critical Review:

1) **Relevance and Contribution:** The paper is highly relevant as it addresses the growing need for integrating IoT with predictive maintenance to enhance operational efficiency in various industries. It makes a significant contribution by detailing how IoT can provide continuous, real-time monitoring, leading to timely interventions and reduced operational costs.
2) **Strengths:**
   - **Comprehensive Framework:** The paper provides a thorough framework for implementing predictive maintenance using IoT, covering data collection, processing, and analysis.
   - **Practical Examples:** It includes practical examples and case studies that illustrate the real-world application and benefits of predictive maintenance systems.
3) **Weaknesses:**
   - **Scalability Issues:** While the paper discusses the benefits, it does not adequately address the challenges related to scaling up IoT-based predictive maintenance systems in large industrial settings.
   - **Data Security:** The paper briefly touches on data security but lacks a detailed discussion on how to mitigate potential cybersecurity risks associated with IoT devices.

C. Secure and Sustainable Predictive Framework for IoT-Based Multimedia Services Using Machine Learning

The integration of Internet of Things (IoT) and multimedia services presents unique challenges, particularly in ensuring security and sustainability. Islam et al. (2021)[3] explores these challenges and proposes a framework leveraging machine learning (ML) to address them. Additionally, OMNET++ is employed as a simulation tool to model and analyze the framework's performance.

a) Overview of the Framework: The framework proposed by the authors aims to enhance the security and sustainability of IoT-based multimedia services.

b) Role of OMNET++ in the Framework: OMNET++ is used extensively in the simulation and analysis phase of the framework. Its role includes:

1) **Network Simulation**: OMNET++ models the communication between IoT devices and the central server, helping to analyze the data flow and network performance under various scenarios.
2) **Performance Analysis**: It enables the evaluation of network performance metrics such as latency, throughput, and packet loss, which are crucial for ensuring the timely and reliable delivery of multimedia data..
3) **Resource Management**: OMNET++ helps simulate the dynamic allocation of resources to ensure efficient use of available bandwidth and computing power.

c) Machine Learning Algorithms Used: The paper discusses various ML algorithms used in the framework, including:

1) **Supervised Learning**: Algorithms like Support Vector Machines (SVM) and Random Forests are employed for classification tasks, such as identifying potential security breaches.
2) **Unsupervised Learning**: Clustering techniques, such as K-means, are used to detect anomalies in the data that might indicate abnormal behavior.

3) **Reinforcement Learning**: This approach is used to optimize resource allocation and energy consumption, ensuring the sustainability of the system.

D. IOT-Based Predictive Maintenance Using LSTM RNN Estimator

The paper "IoT-Based Predictive Maintenance Using LSTM RNN Estimator" [4] presents a sophisticated approach for enhancing predictive maintenance using IoT and advanced machine learning techniques. It describes a system where IoT sensors collect extensive data from various devices connected to a central processing unit. This data is used to predict the Remaining Useful Life (RUL) of devices through machine learning models. Two types of neural networks are employed: Vanilla Recurrent Neural Network (RNN) and Long Short-Term Memory Recurrent Neural Network (LSTM-RNN).

The Vanilla-RNN is noted for its simplicity and is suitable for short-term predictions. However, for more accurate long-term predictions, the LSTM-RNN is preferred due to its ability to handle long-term dependencies and mitigate the vanishing gradient problem. The paper details the process of normalizing input parameters and removing environmental data correlations to ensure accurate health predictions. The models are trained to provide timely maintenance predictions, optimizing operational efficiency and reducing costs.

The authors conducted experiments on light bulbs in different environmental conditions, demonstrating the models' predictive accuracy. The LSTM-RNN achieved better performance, with an estimation error of less than 1 percent, compared to the Vanilla-RNN. This indicates its superior capability in handling complex, long-term predictive tasks, making it ideal for critical devices requiring reliable maintenance scheduling.

III. System Architecture

The proposed solution involves setting up a simulation environment using OMNeT++ to generate realistic sensor data, including both normal and failure scenarios. The sensors include temperature, vibration, pressure, and torque sensors. The simulation is structured using multiple components that interact to simulate data transmission, reception, scheduling, aggregation, data flow, and decision-making. The components include Sensor modules, Equipment modules (CPU), and a Scheduler.

A. Transmission and Reception of Data

Each sensor, represented by the Sensor class, generates values based on weights, degradation rates and pre-defined thresholds. The generated data is periodically transmitted to the CPU using exponential back-off scheduling. Key parameters include lower limit of threshold, upper limit of threshold, degradation rate, weight, and ID. The sensor updates its value realistically using degradation rate, which makes either the value decreases or increases depending on the sensor type. The flowchart in Fig.1 shows the process.

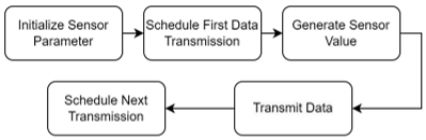

Fig. 1: This diagram illustrates the process flow for the sensor module, including initialization, scheduling data transmission, generating sensor values, checking degradation and failure status, transmitting data, and scheduling the next transmission.

Since temperature and pressure go up over time as the machine works, temperature and pressure sensors increases their values by predefined rate. To avoid unrealistic variation in sensor data, we employed the logic to be updating is done randomly. Hence, by 50% chance, each sensor either increases its value or not. Vibration and torque sensors employ the same logic, but they decreases their values over time.

- ID: A unique identifier for each sensor.
- Lower Threshold: The lower limit of acceptable sensor values.
- Upper Threshold: The upper limit of acceptable sensor values.
- Degradation Rate: The rate at which the sensor value changes over time, reflecting the progression of equipment toward a malfunction state as it operates.
- Weight: Each sensor has pre-assumed individual weight, which reflects reliability for each sensor. We assigned low weight if the sensor is not reliable and is further away from the main engine/motor.

1) Equipment Module: The 'Equipment' class receives sensor data, aggregates it, and determines the equipment's status. Sensor data is stored in categorized maps, and the status is decided based on the aggregated data. The module processes data from temperature, vibration, pressure, and torque sensors.

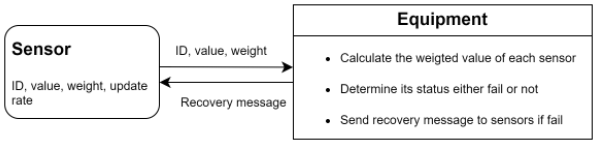

Fig. 2: Message handling process: sensor sends ID, value, and weight parameters to equipment, and equipment process the received data internally.

Equipment receives ID, value, weight data from each sensor. Each data is stored to designated storage. The equipment stores data until all data from three different sensors and four different types. Once all data have been received from sensors, equipment determines if the status is 'normal' or 'failure' based on its logic.

B. Scheduling

The simulation employs *Exponential backoff* algorithm [5] for scheduling sensor transmissions. Exponential backoff algorithm schedules the next transmission event for each sensor. This algorithm sets the event at the current simulation time plus a random interval drawn from an exponential distribution with a mean of 1.0.

C. Data Flow

1) Each sensor generates a value based on its current state and degradation rate.
2) The sensor uses *Exponential backoff* scheduling to transmit the value to the Equipment module.
3) The Equipment module receives the data, aggregates it, and stores it.
4) The status of the equipment is determined based on the aggregated data.
    a) If a value of any sensor go beyond the predefined threshold, equipment determines it is failed.
5) The equipment sends messages to all sensors to recover the default values when the status is determined as failure.

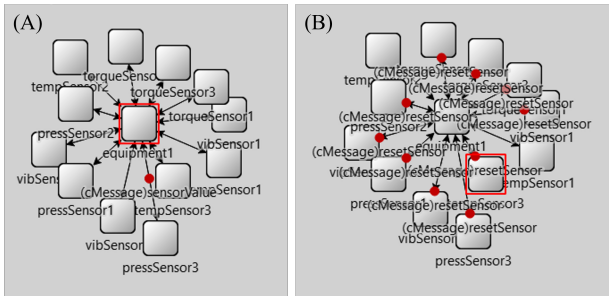

Fig. 3: (A) represents data transmission from a sensor to CPU. Four different types of sensors, temperature, pressure, vibration, and torque. In this figure, the pressure sensor #3 is sending its data to the CPU of equipment. (B) represents that the CPU of the equipment sends a recovery message

D. Data Logging

The simulation logs detailed data for analysis and decision-making purposes. This data is recorded in a structured format that captures the time series information of sensor readings and the corresponding equipment status. The data log includes measurements from each type of sensor (temperature, vibration, pressure, and torque) along with the calculated status of the equipment (normal or failure), see Fig. 4 and 5.

time	temp1	temp2	temp3	vib1	vib2	vib3
1.951545	110	120	130	24	20	14.9
4.406101	113	121	130.1	24	19.5	14.7
9.194111	115	121.5	130.5	22	19	14.6
10.60514	116	122	130.5	22	18.5	14.6
13.63946	117	123.5	130.5	20	17	14.5
17.02464	117	124.5	130.6	17	15.5	14.3
20.75751	120	125.5	130.9	16	14	14

Fig. 4: Subset of data log: Time Stamp, Temp and Vib Sensor. Continued in Fig. 5

press1	press2	press3	torque1	torque2	torque3	status
51	60	70.1	749	599.5	499.8	normal
54	60.5	70.2	748	599.5	499.6	normal
56	61.5	70.5	744	598	499.4	normal
56	61.5	70.5	744	598	499.3	normal
58	62.5	70.6	744	597	499.2	normal
63	63	70.7	744	596	499.1	normal
65	63	70.7	743	595.5	499.1	normal

Fig. 5: Subset of data log: Pressure and Torque Sensor, Status

IV. LSTM Prediction Model Implementation

A. Data Preparation

The data used for training and evaluating the Time-to-Failure (TTF) prediction model was generated from the OMNET++ simulation. The dataset comprises time-series data collected from multiple sensors, including temperature, vibration, pressure, and torque sensors (Number of samples for training: 278152). The initial data processing steps involved the following:

1) **Data Loading and Sorting**: The raw data was loaded into a pandas DataFrame and sorted chronologically based on the timestamp to ensure proper temporal sequence.
2) **Failure Index Identification**: Failure events were identified by filtering the dataset where the 'status' column indicated a failure. The indices of these failure events were stored for segmenting the data.
3) **Segmentation and Labeling**: The data was segmented based on failure indices. For each segment, Time-to-Failure (TTF) was computed as the difference between the time of the current failure and the next failure (or the end of the dataset if it was the last failure). This TTF value was then smoothed using a Gaussian filter with a standard deviation (sigma) of 2 to reduce noise and enhance the signal.
4) **Feature Engineering**: To enhance model performance, lag features were created. Specifically, lagged versions of each sensor reading (temperature, vibration, pressure, torque) were generated

to capture temporal dependencies. For each sensor, three lag features were included, representing previous time steps.

5) **Handling Missing Values**: Rows containing NaN values introduced by the lagging process were removed to ensure the integrity of the dataset.

B. Feature Scaling and Sequencing

1) **Feature Scaling**: The features were normalized using `StandardScaler` to ensure that all features had a similar scale, which is crucial for the convergence of the neural network during training.
2) **Sequence Creation**: The data was transformed into sequences suitable for LSTM input. A sliding window approach was used to create sequences of fixed length (10 time steps), with corresponding TTF values as targets.

C. Model Architecture

The model architecture is a deep learning model designed to predict the TTF based on the sequential sensor data. The architecture is as follows:

1) **Bidirectional LSTM Layer**: The model begins with a Bidirectional Long Short-Term Memory (LSTM) layer with 512 units. The bidirectional setting allows the model to learn from both past and future contexts within the sequences.
2) **Dropout Layers**: Dropout layers with a rate of 0.2 are applied after each LSTM layer to prevent overfitting by randomly setting a fraction of input units to zero during training.
3) **Additional LSTM Layers**: The model includes two additional LSTM layers with 256 and 128 units respectively. These layers further capture temporal dependencies and patterns in the data.
4) **Dense Output Layer**: The final layer is a Dense layer with a single unit, which outputs the continuous TTF prediction.

D. Training and Evaluation

1) **Model Compilation**: The model was compiled using the Adam optimizer and mean squared error as the loss function. Mean Absolute Error (MAE) was also used as a metric to evaluate performance.
2) **Early Stopping**: To prevent overfitting and ensure optimal training, Early Stopping was implemented. This technique halts training when validation loss does not improve for 10 consecutive epochs, restoring the model to the best weights observed.
3) **Training**: The model was trained for up to 100 epochs with a batch size of 32. Training involved a 10% validation split to monitor performance on unseen data during training.
4) **Evaluation**: After training, the model was evaluated on a test set that was not used during training. The performance metrics included loss (mean squared error) and mean absolute error (MAE). Additionally, the R^2 score was computed to assess the proportion of variance explained by the model.
5) **Model Saving**: The trained model was saved for future use and analysis.

E. Results and Visualization

Training history was plotted to visualize the loss and MAE over epochs as seen in the figure below. These plots help in understanding the convergence and effectiveness of the training process. Sample predictions were compared with actual TTF values to demonstrate the model's predictive accuracy.

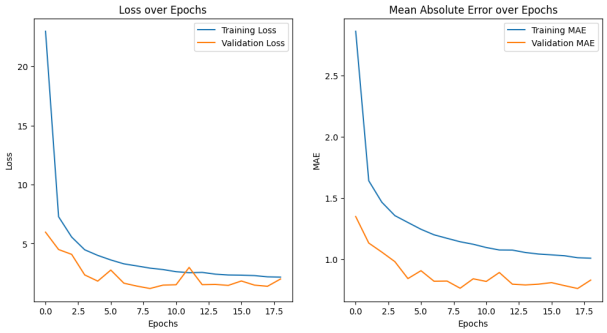

Fig. 6: Training and Validation Set

The training results demonstrate a highly effective model, as reflected by the significant decrease in both training and validation losses and Mean Absolute Errors (MAE) over the epochs. The final Mean Squared Error (MSE) of 1.3861 and MAE of 0.7824 indicate low prediction errors, and the high R^2 score of 0.9875 reveals that the model accounts for 98.75 percent of the variance in the target variable, signifying excellent predictive accuracy. Throughout the training process, the validation metrics show some minor fluctuations but generally exhibit a downward trend, suggesting robust generalization. The close correspondence between the predicted and actual values, such as predictions of $5.66, 2.70, 31.50, 28.36$, and 26.69 versus actual values of $4.08, 1.83, 31.80, 29.79$, and 26.01, respectively, further confirms the model's precision and reliability in making accurate predictions. Overall, these results highlight the model's strong learning capability and its effectiveness in capturing the underlying patterns in the data.

V. Real-time Monitoring

The real-time monitoring architecture for the smart factory simulation is designed to provide continuous,

real-time insights into the operational status of factory equipment. This architecture leverages various tools and platforms to collect, process, analyze, and visualize sensor data in real time (see Fig. 7).

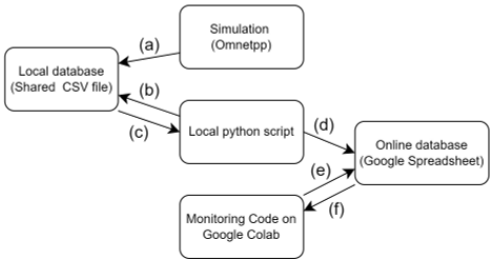

Fig. 7: This diagram illustrates the components of the real-time monitoring system, including the simulation (OMNeT++), local database (CSV file), local Python script, online database (Google Spreadsheet), and monitoring code on Google Colab. Each arrow indicates the flow of data stream.

A. Overview of the Monitoring Architecture

1) Simulation (OMNeT++): The OMNeT++ platform serves as the cornerstone of the simulation environment, running continuously to emulate the smart factory setting. It generates real-time sensor data from various types of equipment, including temperature, vibration, pressure, and torque sensors. This data is crucial as it mirrors the operational status of the machinery.

2) Local Database (Shared CSV File): The sensor data generated by OMNeT++ is instantaneously logged into a shared CSV file (see Fig.7, (a)). This file acts as an intermediary database, ensuring that the newly generated data is readily accessible for subsequent processing.

3) Local Python Script: Local Python script operates concurrently with OMNeT++. It serves two critical functions, data fetching and data upload.

- **Data Fetching**: The script continuously monitors the shared CSV file (see Fig.7, (b)), fetching newly stored data on a row-by-row basis (see Fig.7, (c)).
- **Data Upload**: Leveraging the Google API, the script uploads the fetched data to a Google Spreadsheet, ensuring real-time updates (see Fig.7, (d)).

4) Online Database (Google Spreadsheet): The Google Spreadsheet functions as a real-time online data storage. It is continuously updated by the local Python script, providing a live sensor data for the real-time monitoring platform.

5) Real-time monitoring: The monitoring code, executed manually on Google Colab, accesses the Google Spreadsheet to fetch the latest data for in-depth analysis (see Fig.7, (e, f)). The Colab script performs real-time anomaly detection, identifying deviations from normal operational parameters that could indicate potential equipment failures.

VI. Conclusion

In this paper, we presented a comprehensive solution for predictive maintenance in smart factories using Long Short-Term Memory (LSTM) networks. Our approach leverages simulated sensor data generated by OMNeT++, combined with feature engineering techniques in Python, to predict the Time-to-Failure (TTF) of equipment accurately.

Key accomplishments include the creation of a realistic simulation environment, the development of a robust feature engineering process, and the design of an LSTM-based model that captures temporal dependencies in sensor data. Additionally, we implemented a real-time monitoring system that continuously processes and analyzes sensor data, providing timely insights and alerts for potential equipment failures. The model demonstrated high accuracy and reliability through rigorous testing and fine-tuning.

For future work, we recommend enhancing the robustness of the predictive models by incorporating advanced data preprocessing techniques and expanding the system to include additional sensor types. Developing scalable solutions for larger datasets and more complex industrial setups is crucial. Moreover, integrating explainable AI techniques will improve the transparency and trustworthiness of the model's predictions. Implementing adaptive learning algorithms and strengthening system security are also essential steps for future research.

Overall, our approach offers a robust framework for predictive maintenance, paving the way for smarter and more efficient industrial operations. Addressing these recommendations will further advance the capabilities and applications of predictive maintenance in smart factories, leading to improved reliability and cost-effectiveness.

References

[1] O. Serradilla, E. Zugasti, J. Rodriguez, et al. Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects. *Applied Intelligence*, 52(11):10934–10964, 2022.

[2] Michael G. Pecht and Myeongsu Kang. *Predictive Maintenance in the IoT Era*, pages 589–612. 2019.

[3] Naveed Islam, Majid Altamimi, Khalid Haseeb, and Mohammad Siraj. Secure and sustainable predictive framework for iot-based multimedia services using machine learning. *Sustainability*, 13(23), 2021.

[4] Jamal S. Rahhal and Dia Abualnadi. Iot based predictive maintenance using lstm rnn estimator. In *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pages 1–5, 2020.

[5] Nah-Oak Song, Byung-Jae Kwak, and Leonard E Miller. On the stability of exponential backoff. *Journal of Research of the National Institute of Standards and Technology*, 108(4):289, 2003.