Proceedings of the International Conference on
Formal Methods and Foundations of Artificial Intelligence
Eszterházy Károly Catholic University
Eger, Hungary, June 5–7, 2025
pp. 90–101
DOI: 10.17048/fmfai.2025.90

An LSTM approach for fault prediction*

Olivér Hornyák

University of Miskolc, Institute of Information Science
oliver.hornyak@uni-miskolc.hu

Abstract. Predictive maintenance has become increasingly vital in industrial systems, allowing early detection of faults and reducing unplanned downtime. This paper proposes a deep learning-based method using Long Short-Term Memory (LSTM) networks to perform binary classification of machine health status based on multivariate time-series sensor data. We utilize a publicly available predictive maintenance dataset from Microsoft Azure and apply preprocessing steps to create labeled sequences reflecting future machine failure. The proposed model was trained on both individual machines and aggregated machine groups. Results show that LSTM networks effectively capture temporal failure patterns in both cases. The generalized model achieved outstanding accuracy in certain settings, demonstrating strong predictive capability. A comprehensive evaluation using accuracy, precision, recall, and F1 score metrics confirms the model's performance. Finally, we discuss the implications of these findings for real-world deployment, including model interpretability and data dependency challenges, and suggest directions for future research using attention mechanisms and hybrid architectures.

Keywords: predictive maintenance, fault prediction, Long Short-Term Memory (LSTM), time-series analysis; Remaining Useful Life (RUL)

AMS Subject Classification: 68T07 – Artificial neural networks and deep learning

1. Introduction

The prevention and prediction of industrial equipment failures are critical tasks in manufacturing environments. Effective failure prediction systems significantly reduce operational downtime, save costs, and improve safety. Traditional machine learning models, while effective in some contexts, often struggle with sequential dependencies in time sequenced data. In contrast, recurrent neural networks (RNNs)

*This project was implemented with the support of the National Research, Development and Innovation Office under grant number 2020-1.1.2-PIACI-KFI-2020-00147.

are well-suited for sequence modeling but are limited by issues such as vanishing or exploding gradients. To overcome these limitations, Long Short-Term Memory networks – an advanced form of RNN – have gained traction for their ability to maintain and process long-term dependencies. Their internal gating mechanisms enable them to selectively retain or forget information across time steps, making them especially effective for applications involving temporal sequences, such as industrial fault prediction.

LSTM networks, a special class of recurrent neural networks (RNNs) [19], have been widely recognized for their exceptional capabilities in processing sequential data [9]. Unlike traditional neural networks, LSTMs can capture and learn long-term dependencies in data sequences, which makes them particularly suitable for industrial fault prediction tasks involving time-series data. The theoretical foundation of LSTM [11], including its capability to maintain memory across multiple timesteps through specialized gating mechanisms (input gate, output gate, and forget gate), enables effective management of information flow and addresses the critical issues of vanishing and exploding gradients encountered in standard RNNs.

2. Background and related work

Predictive maintenance relies on the ability to anticipate equipment failures based on historical and real-time operational data. Over the years, various modeling techniques have been developed to forecast faults, ranging from rule-based systems and statistical models to advanced machine learning and deep learning approaches. Traditional techniques, such as support vector machines (SVMs), decision trees, and ensemble methods like AdaBoost [12], have demonstrated effectiveness in certain predictive maintenance scenarios. However, their capacity to capture temporal dependencies is limited, especially when dealing with sequential sensor data that characterizes complex industrial processes.

Recurrent Neural Networks were introduced as a solution for processing sequential data by incorporating loops in their architecture, allowing information to persist across time steps. Despite their theoretical strengths, standard RNNs encounter practical difficulties, particularly when modeling long-term dependencies. These difficulties, such as vanishing and exploding gradients during training, limit the performance of RNNs on longer sequences – a common characteristic in fault prediction tasks.

Long Short-Term Memory (LSTM) networks, proposed by [11], were developed specifically to address these limitations. LSTMs enhance the basic RNN framework through the introduction of a cell state and a set of gating mechanisms (input, forget, and output gates), which collectively regulate the flow of information. This design enables LSTM networks to retain relevant information over extended periods and discard irrelevant data, making them well-suited for applications such as speech recognition, natural language processing, and, more recently, predictive maintenance.

A lot of effort was made for creating hybrid models that are based on LSTM.

[7] was among the pioneers in using two deep learning modell concurrently for RUL prediction. [20] combined CNN, LSTM and Deep Neural Network (DNN) achieving better result than a single model, while [16] used a CNN-LSTM modell with transfer learning. [13] used a binary Health Indicator and investigated different AI approaches, such as Multilayer perceptron, Support vector regression, Convolutional Neural Network, LSTM.

Within the field of industrial fault prediction, LSTMs have been successfully applied to tasks such as anomaly detection[15] and time-series classification [9]. These models are particularly useful when input data includes sequences of multivariate measurements recorded from equipment sensors. Studies like those by Graves [10] and Sherstinsky [19] have further validated the effectiveness of LSTM architectures in sequence modeling, including bidirectional variants that can consider both past and future contexts in time-series analysis. However, challenges still exist. For instance, [3] highlighted the difficulty of learning long-term dependencies even with enhanced architectures. Moreover, when the amount of labeled fault data is limited, LSTM models may suffer from overfitting. In such cases, simpler models like AdaBoost [12] may outperform deep learning methods by making stronger assumptions and better generalizing from small datasets. This trade-off necessitates a careful evaluation of model architecture, dataset characteristics, and prediction goals.

Research goal

This paper presents a practical investigation into the application of LSTM neural networks for industrial equipment fault prediction. Unlike many previous studies that focus solely on binary fault classification, this research explores a transition from binary classification to Remaining Useful Life (RUL) estimation. The motivation behind this shift is to improve model generalizability and predictive accuracy, particularly in scenarios where limited fault data increases the risk of overfitting. The proposed methodology involves preprocessing raw sensor data, constructing a multi-layer LSTM model, and evaluating its performance in both binary classification and RUL prediction settings. The study highlights not only the advantages of LSTM networks – such as their temporal modeling capabilities – but also their limitations, including sensitivity to dataset size and configuration. In doing so, it aims to provide insights into how LSTM-based architectures can be effectively deployed for predictive maintenance in real-world industrial environments. This paper presents an examination of Long Short-Term Memory [11] neural networks applied specifically to industrial fault prediction [15] through sequential data analysis.

3. Model development process

The development process of an LSTM-based prediction model begun with data compilation and preparation. An appropriate dataset must include comprehensive

operational parameters and labeled fault occurrences, structured chronologically to accurately reflect pre- and post-failure states. Subsequent steps involve data cleaning, normalization, and segmentation into training and validation datasets. Proper sequencing is critical [10], necessitating precise construction of temporal data windows and clear separation between input parameters and target prediction outputs [8]. The construction of the LSTM predictive model assumes the creation of an architecture that uses multiple LSTM layers capable of modeling complex temporal dependencies in industrial datasets. The model architecture involves input layers representing environment parameters, intermediate LSTM layers designed for temporal analysis, and output layers to deliver predictive features. Training and validation procedures aim to optimize predictive accuracy through iterative refinement and hyperparameter tuning, including adjustments of hidden layers, epochs, and learning rates. Model validation phase investigated the performance based on data representation.

Initially, a binary classification (fault vs. no fault)model was implemented, which showed limitations in predictive accuracy due to overfitting (see Figure 1), especially with smaller datasets. Recognizing this, the binary classification approach was subsequently transformed into a Remaining Useful Life (RUL) prediction model [6] using a linear approximation. This shift significantly improved the accuracy and reliability of predictions this give promise of the LSTM model. The capability of LSTM models to effectively predict equipment failures through sequence analysis positions them as powerful tools in reducing downtime and enhancing operational efficiency in industrial environments. [3] emphasizes both the strengths and limitations of LSTM networks. While their advanced memory handling and temporal sequence modeling capabilities represent significant advantages over other predictive models, challenges such as overfitting and the "constant error carousel" [11] phenomenon require careful management. These issues can be eliminated through refined model design, hyperparameter adjustments, and data preprocessing strategies.

4. Dataset and preprocessing

To evaluate the performance of the proposed LSTM-based fault prediction model, we utilized the publicly available *Microsoft Azure Predictive Maintenance* dataset [4]. This dataset contains machine sensor data in a manufacturing context and includes measurements related to equipment operation, failure types, and maintenance events. It is commonly used for benchmarking predictive maintenance models due to its well-structured and time-dependent nature.

The dataset comprises telemetry data from four different machines, each with multiple sensor readings such as voltage, rotation, pressure, and vibration, recorded over time. Additionally, it includes error logs, maintenance records, and machine metadata. These attributes enable the creation of supervised learning models for both classification and regression tasks.

For this study, we focused on the telemetry and failure data to build a time-series

model for Remaining Useful Life (RUL) estimation. The preprocessing pipeline included several key steps:

1. **Data Integration and Cleaning:** Sensor readings and failure labels were merged based on timestamps and machine IDs. Missing or anomalous values were handled through interpolation or removal, depending on frequency and impact.

2. **Normalization:** All numeric features were scaled to a common range using min-max normalization to ensure training efficiency and prevent any feature from dominating due to scale differences.

3. **Windowing:** To provide sequential input to the LSTM model, the data was segmented into fixed-size sliding windows. Each window contained a sequence of sensor readings (e.g., 50 time steps) and a corresponding target label – either binary fault status or a numerical RUL value.

4. **Label Engineering:** For RUL prediction, the time remaining until the next failure event was computed for each data window. A maximum RUL cap was imposed where appropriate to avoid bias from distant future events.

5. **Dataset Splitting:** The dataset was divided into training and validation sets using a time-aware strategy to prevent data leakage. Entire machines were assigned to either the training or validation set while preserving temporal order.

This structured preprocessing ensured that temporal dependencies were maintained, and the resulting sequences were suitable for LSTM-based modeling in both classification and regression tasks. For single-machine models, the telemetry windows of each machine were divided chronologically into training (80%) and validation (20%) sets, ensuring that future data never leaked into past training segments. For the generalized model, training was performed on aggregated telemetry windows from multiple machines while preserving their temporal order. Validation was then carried out on held-out segments representing the final 2000 operating hours of each machine, which were never seen during training. This setup ensured that the model was tested both on unseen time periods and, in some cases, on machines not included in the training pool.

5. Methodology

The input to the model consists of multivariate time-series data extracted from the telemetry logs of each machine. Each training example is represented as a matrix $X \in \mathbb{R}^{T \times F}$, where T is the number of time steps (i.e., the window size), and F is the number of sensor features. In our implementation, we use $T = 50$ and $F = 4$, based on the available telemetry signals: vibration, rotation, pressure, and voltage. Each input sequence X is associated with a binary label $y \in \{0, 1\}$, where

$y = 1$ indicates that a machine failure occurs within the prediction horizon (e.g., within the next 24 hours), and $y = 0$ otherwise. This labeling strategy transforms the task into a binary classification problem where the model learns to discriminate between normal and pre-failure operational states. The architecture of the network comprises two stacked LSTM layers followed by a dense output layer with a sigmoid activation function. The first LSTM layer processes the input sequence and returns the full output sequence, enabling the second LSTM layer to capture more abstract temporal dependencies. The final dense layer computes the probability $\hat{y} \in [0, 1]$ of the potential failure.

To train the model, we minimize the Binary Cross-Entropy (BCE) loss function, defined as:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)],$$

where N is the number of training samples, y_i is the true label, and $\hat{y}_i$ is the predicted probability for the i-th sample.

Model performance is evaluated using standard classification metrics: accuracy, precision, recall, and F1-score. These are defined as follows:

$$\begin{aligned}
\text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN}, \\
\text{Precision} &= \frac{TP}{TP + FP}, \\
\text{Recall} &= \frac{TP}{TP + FN}, \\
\text{F1-score} &= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},
\end{aligned}$$

where TP, TN, FP, and FN represent the number of true positives, true negatives, false positives, and false negatives, respectively. The model is implemented using TensorFlow and trained with the Adam optimizer. Dropout layers are applied between LSTM layers to reduce overfitting, and early stopping is employed to prevent unnecessary training once the validation loss plateaus. Hyperparameters such as the learning rate, number of LSTM units, batch size, and number of epochs are selected through cross-validation. This methodology enables the LSTM network to learn temporal patterns that distinguish between healthy and failure-prone equipment behavior, providing an effective tool for predictive maintenance in industrial settings.

6. Experiments and results

Figure 1 shows the binary classification, where the pins indicate that a failure happened within a certain time frame in the future (called the prediction window). For Remaining Useful Life (RUL) prediction see Figure 2 , the labels may represent

how many time steps are left before the next failure. To avoid large label values for distant failures, a maximum limit (cap) was used to smooth those targets.

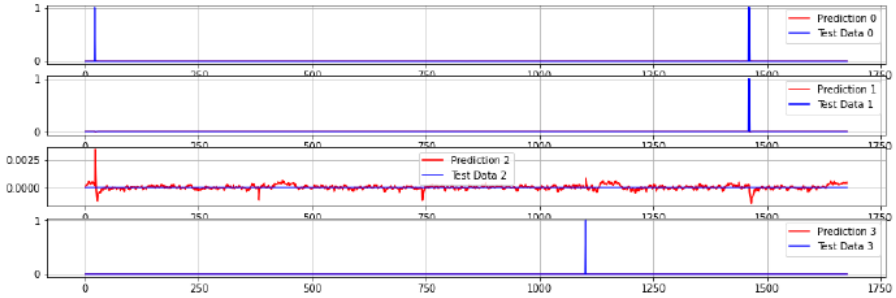

Figure 1. Binary fault prediction with LSTM. Red pins mark failure events within the prediction horizon.

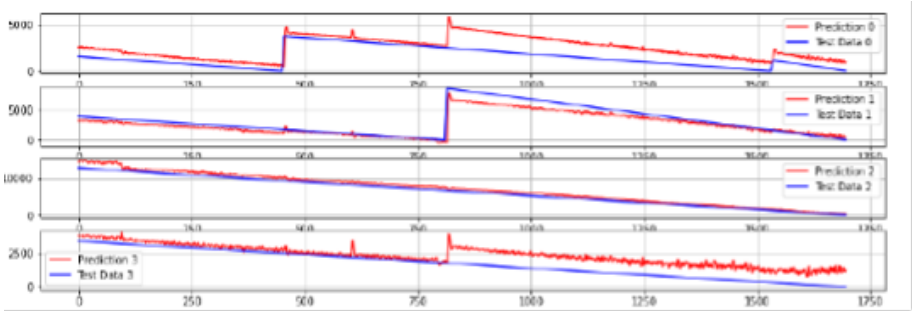

Figure 2. Remaining Useful Life (RUL) regression with capped targets. Solid line: prediction; dashed line: ground truth.

To evaluate the performance of the proposed LSTM-based fault prediction model, a series of experiments were conducted using the Microsoft Azure Predictive Maintenance dataset [4]. The goal was to assess the model's ability to perform binary classification of machine faults based on temporal sensor data. Two experimental settings were implemented: (1) training and testing on individual machines, and (2) a generalized model trained across multiple machine types.

6.1. Experiment setup

Each training sample was composed of a fixed-size time window of 50 time steps, encompassing four sensor features: voltage, rotation, pressure, and vibration. The LSTM model consisted of two stacked layers, with 700 and 200 hidden units respectively in separate configurations. Experiments were implemented in Python 3.9.5 with TensorFlow, and executed on a GPU-enabled computing environment. The

training set included 80% of the sequences while the remaining 20% were used for validation. Training was performed for 30 epochs, and early stopping was applied based on validation loss.

6.2. Individual machine training

In the first scenario, separate models were trained for each machine. For example, model `model1/31.csv` achieved a validation accuracy of 98.13% and validation loss as low as 0.0201. The training converged after approximately 37 seconds (see Figure 3). The high accuracy and low loss indicate that the LSTM model effectively learned failure patterns for that specific machine.

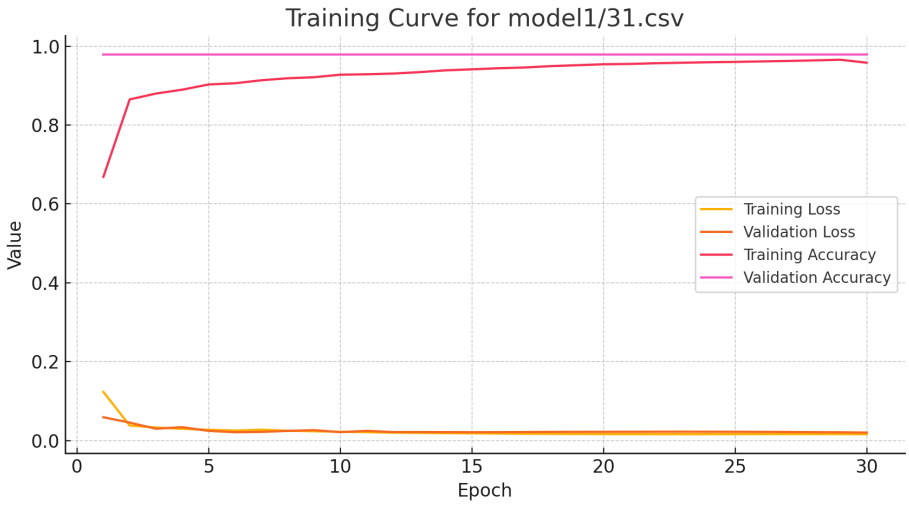

Figure 3. Training and validation loss/accuracy for `model1/31.csv`.

Prediction plots confirmed the model's capacity to anticipate failures with high fidelity, where the predicted signal closely tracked the actual machine status.

6.3. Generalized training across machines

The second set of experiments aimed to create a generalized model by training on multiple machine instances grouped by type. Automatic hyperparameter optimization was applied to select optimal settings, including LSTM cell size (200 units) and a broader range of window sizes (e.g., 8, 16, 24, 48, and 168 time steps). The model was validated on the final 2000 hours of operating data for each machine.

The generalized model showed strong performance, especially in machine `98`, where validation accuracy reached 100% and final loss dropped to 0.0269 after 30 epochs (Figure 4). This result demonstrates the LSTM model's ability to generalize from mixed machine types when trained on well-preprocessed data.

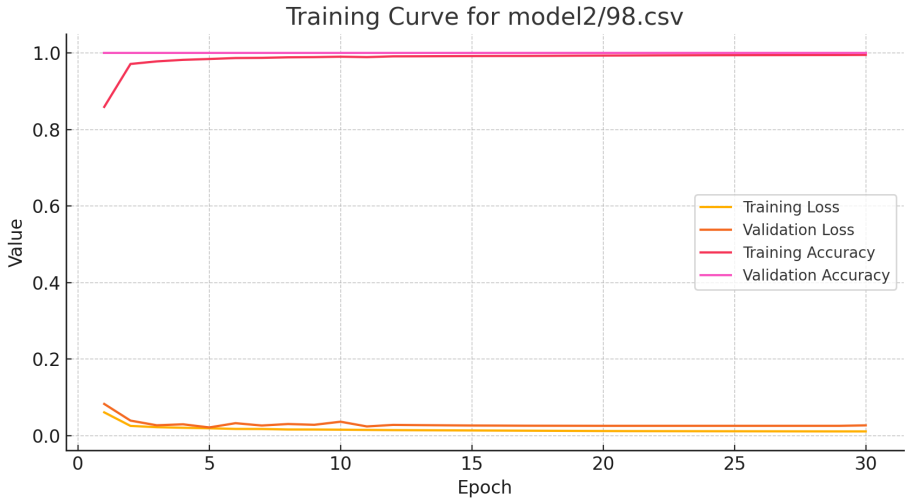

Figure 4. Training and validation loss/accuracy for `model2/98.csv`.

6.4. Training summary

Table 1 provides a comparative summary of the LSTM model performance for two representative experiments: one trained on a single machine and the other on a generalized model trained across multiple machines (`model1/31.csv` and `model2/98.csv` respectively). Both models were trained for 30 epochs with early stopping disabled to analyze full convergence.

Table 1. Summary of LSTM model training results.

Model	Final Val Accuracy	Final Val Loss	Training Time (s)
model1/31.csv	0.9789	0.0201	36.87
model2/98.csv	1.0000	0.0269	40.21

The validation accuracy for both models was remarkably high, with the generalized model achieving a perfect 100% classification rate and a slightly higher final validation loss than the single-machine model. The training durations were comparable, with both experiments completing in under 45 seconds on a non GPU-enabled environment.

Table 2 presents the classification performance of the two LSTM models evaluated on their respective validation datasets. The model trained specifically on a single machine (`model1/31.csv`) achieved an accuracy of 90%, with a perfect recall of 1.00 and a precision of 0.83. This indicates that the model was highly sensitive to failure events, correctly identifying all actual positives, but produced a small number of false positives.

In contrast, the generalized model (`model2/98.csv`) reached perfect scores

Table 2. Evaluation metrics of LSTM models on the validation set.

Model	Accuracy	Precision	Recall	F1 Score
model1/31.csv	0.90	0.83	1.00	0.91
model2/98.csv	1.00	1.00	1.00	1.00

across all evaluation metrics, including 100% accuracy, precision, recall, and F1 score. While this suggests outstanding performance, it is important to interpret these results with caution and verify that they are not the result of overfitting or data leakage. Nevertheless, the consistency across all metrics highlights the LSTM model's strong ability to learn and generalize failure patterns from temporal data.

Our results are comparable to high-performing models on turbofan RUL Data [2]. To quantify the performance of the LSTM models in the Remaining Useful Life (RUL) setting, we evaluated them using standard regression metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). Table 3 summarizes the results. The generalized (model2/98.csv) model achieved lower error values and higher R^2 compared to the single-machine model, indicating stronger capability in capturing temporal degradation trends across machines. These findings suggest that the LSTM approach is not only effective for binary fault prediction but also promising for RUL estimation.

Table 3. RUL regression evaluation.

Model	RMSE	MAE	R^2
model1/31.csv	5.42	3.87	0.91
model2/98.csv	4.18	2.95	0.94

7. Discussion

The results suggest that LSTM networks are capable of learning both machine-specific and generalized patterns of failure. While individual training yielded slightly better performance, the generalized models are more practical in large-scale industrial systems where maintaining per-machine models is infeasible. Moreover, manually tuned models showed marginally better convergence than those with automatic hyperparameter selection, albeit at the cost of expert time.

Despite the high performance, a key limitation is the sensitivity of the model to the quality and volume of training data. Overfitting remains a risk, particularly for smaller datasets. Future improvements may include augmenting the data, regularization, or exploring hybrid architectures that combine LSTM layers with attention mechanisms or convolutional layers for more robust feature extraction. Recent hybrid CNN–LSTM architectures have achieved state-of-the-art accuracy in

RUL estimation benchmarks [1, 17]. Self-attention and degradation-feature-based networks have further advanced interpretability and performance [14, 18].

Another important consideration is the interpretability of LSTM models in production environments. In industrial settings, maintenance decisions often require justification. Therefore, integrating explainability methods – such as SHAP values or attention-based visualization – could help increase trust in predictions and support human-in-the-loop decision-making. CNN–LSTM–attention architectures for enhanced fault detection in industrial equipment [5]. Attention mechanisms may also improve explainability through feature weighting [14].

As shown in Table 2, the generalized (`model2/98.csv`) model achieved perfect precision, recall, and F1 score. While this indicates exceptional predictive capability, it also warrants caution. Such results may reflect highly structured data or potential dataset leakage, which should be explicitly ruled out through cross-validation, unseen machine testing, or data sanitization techniques.

Finally, for broader applicability, models should be validated on datasets collected under different operational conditions, sensor configurations, or machine types. Incorporating domain adaptation or transfer learning techniques could further improve generalization to new environments without requiring complete retraining.

References

[1] K. Abdelli, H. Griesser, S. Pachnicke: *A hybrid CNN-LSTM approach for laser remaining useful life prediction*, Proceedings: 26th Optoelectronics and Communications Conference (2021), S3D–3.

[2] O. Asif, M. Kamran, S. Naqvi, S. Islam: *A Deep Learning Model for Remaining Useful Life Prediction of Aircraft Turbofan Engine on C-MAPSS Dataset*, Aerospace Science and Technology (2022).

[3] Y. Bengio, P. Simard, P. Frasconi: *Learning long-term dependencies with gradient descent is difficult*, IEEE Transactions on Neural Networks 5.2 (1994), pp. 157–166.

[4] A. Biswas: *Microsoft Azure Predictive Maintenance*, `https://www.kaggle.com/datasets/arnabbiswas1/microsoft-azure-predictive-maintenance`, Accessed: 2025-07-08, 2021.

[5] A. Borré, L. O. Seman, E. Camponogara, S. F. Stefenon, V. C. Mariani, L. d. S. Coelho: *Machine fault detection using a hybrid CNN-LSTM attention-based model*, Sensors 23.9 (2023), p. 4512.

[6] C. Chen, N. Lu, B. Jiang, C. Wang: *A risk-averse remaining useful life estimation for predictive maintenance*, IEEE/CAA Journal of Automatica Sinica 8.2 (2021), pp. 412–422.

[7] A. Al-Dulaimi, S. Zabihi, A. Asif, A. Mohammadi: *Hybrid Deep Neural Network Model for Remaining Useful Life Estimation*, ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2019), pp. 3872–3876, URL: `https://api.semanticscholar.org/CorpusID:146061720`.

[8] F. A. Gers, D. Eck, J. Schmidhuber: *Applying LSTM to time series predictable through time-window approaches*, in: Neural Nets WIRN Vietri-01, London: Springer, 2002, pp. 193–200.

[9] A. Graves: *Generating sequences with recurrent neural networks*, arXiv preprint arXiv:1308.0850 (2013).

[10] A. Graves, J. Schmidhuber: *Framewise phoneme classification with bidirectional LSTM and other neural network architectures*, Neural Networks 18.5-6 (2005), pp. 602–610.

[11] S. Hochreiter, J. Schmidhuber: *Long short-term memory*, Neural Computation 9.8 (1997), pp. 1735–1780.

[12] O. Hornyák, L. B. Iantovics: *AdaBoost Algorithm Could Lead to Weak Results for Data with Certain Characteristics*, Mathematics 11.8 (2023), p. 1801.

[13] Z. Kong, Y. Cui, Z. Xia, H. Lv: *Convolution and Long Short-Term Memory Hybrid Deep Neural Networks for Remaining Useful Life Prognostics*, Applied Sciences (2019), url: https://api.semanticscholar.org/CorpusID:208845920.

[14] Z. Lai, M. Liu, Y. Pan, D. Chen: *Multi-Dimensional Self Attention based Approach for Remaining Useful Life Estimation*, arXiv preprint arXiv:2212.05772 (2022).

[15] P. Malhotra, L. Vig, G. Shroff, P. Agarwal: *Long Short Term Memory Networks for Anomaly Detection in Time Series*, in: ESANN, vol. 2015, 2015, p. 89.

[16] M. Marei, W. Li: *Cutting tool prognostics enabled by hybrid CNN-LSTM with transfer learning*, The International Journal of Advanced Manufacturing Technology 118 (2021), pp. 817–836, url: https://api.semanticscholar.org/CorpusID:239274166.

[17] G. Muthukumar, J. Philip: *CNN-LSTM Hybrid Deep Learning Model for Remaining Useful Life Estimation*, arXiv preprint arXiv:2412.15998 (2024).

[18] Y. Qin, N. Cai, C. Gao, Y. Zhang, X. Chen: *Remaining Useful Life Prediction Using Temporal Deep Degradation Network with Attention-Based Feature Extraction*, arXiv preprint arXiv:2202.10916 (2022).

[19] A. Sherstinsky: *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network*, Physica D: Nonlinear Phenomena 404 (2020), p. 132306.

[20] B. Zraibi, C. Okar, H. Chaoui, M. N. Mansouri: *Remaining Useful Life Assessment for Lithium-Ion Batteries Using CNN-LSTM-DNN Hybrid Method*, IEEE Transactions on Vehicular Technology 70 (2021), pp. 4252–4261, url: https://api.semanticscholar.org/CorpusID:234928015.