information

IMPACT FACTOR 2.9

CITESCORE 6.5

Article

Neural Network-Based Optimization of Repair Rate Estimation in Performance-Based Logistics Systems

Milan Dejanović, Stefan Panić, Nataša Kontrec, Danijel Đošić and Saša Milojević

MDPI

https://doi.org/10.3390/info16121031

information

MDPI

Article

Neural Network-Based Optimization of Repair Rate Estimation in Performance-Based Logistics Systems

Milan Dejanović [1], Stefan Panić [1,2], Nataša Kontrec [1,*], Danijel Đošić [1] and Saša Milojević [3,*]

[1] Faculty of Sciences and Mathematics, University of Pristina, Lole Ribara 29, 32820 Kosovska Mitrovica, Serbia; milan.dejanovic@pr.ac.rs (M.D.); stefan.panic@pr.ac.rs (S.P.); danijel.djosic@pr.ac.rs (D.Đ.)
[2] Academy of Technical and Art Applied Studies, School of Electrical and Computing Engineering, Vojvode Stepe 283, 11000 Belgrade, Serbia
[3] Faculty of Engineering, University of Kragujevac, Sestre Janjic 6, 34000 Kragujevac, Serbia
* Correspondence: natasa.kontrec@pr.ac.rs (N.K.); sasa.milojevic@kg.ac.rs (S.M.)

Abstract

Performance-Based Logistics (PBL) frameworks prioritize system availability by optimizing maintenance strategies, with repair rate estimation playing a critical role in predictive maintenance planning. This study proposes a machine learning-based approach for repair rate prediction, leveraging fully connected neural networks (FCNNs) and Long Short-Term Memory (LSTM) networks trained on repair rate samples generated from a stochastic model. The FCNN estimates maximum repair rates, while the LSTM predicts minimum repair rates, capturing both steady-state and sequential dependencies in repair rate variations. By eliminating the need for complex mathematical formulations, the proposed methodology provides a scalable and computationally efficient alternative to traditional stochastic models. Extensive performance evaluations demonstrate that the neural networks achieve higher accuracy and lower computational costs compared to stochastic approaches, making them well-suited for real-time predictive maintenance applications. This research enhances decision-making in maintenance planning, optimizes resource allocation, and improves overall system reliability within PBL frameworks.

Keywords: neural networks; repair rate estimation; Performance-Based Logistics (PBL); predictive maintenance; repair rate

Academic Editor: Xinming Zhang

Received: 21 October 2025
Revised: 19 November 2025
Accepted: 22 November 2025
Published: 26 November 2025

Citation: Dejanović, M.; Panić, S.; Kontrec, N.; Đošić, D.; Milojević, S. Neural Network-Based Optimization of Repair Rate Estimation in Performance-Based Logistics Systems. *Information* **2025**, *16*, 1031. https://doi.org/10.3390/info16121031

1. Introduction

Modern industrial and defense systems demand high availability and reliability, requiring efficient maintenance strategies to minimize downtime and optimize resource allocation. Performance-Based Logistics (PBL) frameworks have emerged as a preferred approach, shifting the focus from managing spare part inventories to ensuring operational readiness. A critical aspect of PBL-driven maintenance planning is the accurate estimation of repair rates, which directly influences system availability and maintenance scheduling.

Traditional stochastic models for repair rate estimation rely on complex probability density functions (PDFs) and cumulative distribution functions (CDFs) to characterize repair time distributions. While these models provide mathematically rigorous formulations, they suffer from key limitations: (1) computational complexity: analytical solutions often require iterative numerical computations, making them impractical for real-time decision-making; (2) limited adaptability: stochastic models assume fixed probabilistic structures, which may not generalize well to dynamic maintenance environments with varying operational conditions; (3) difficult parameter estimation: accurate parameter

selection requires historical failure data and expert domain knowledge, posing challenges in data-limited scenarios.

A recent paper [1] introduces a structured methodology for generating repair rate samples, which serve as the basis for this research. Instead of directly applying stochastic models in future repair rate estimations, we propose training machine learning models on these generated samples to facilitate rapid and adaptable predictions. This approach bridges the gap between theoretical stochastic modeling and practical predictive maintenance applications, enabling computational efficiency and flexibility. In the literature, recent advancements in reliability-based optimization models have been explored in the context of railway vehicle maintenance. A study by Milković et al. (2024) [2] presents a reliability-based model for optimizing resource allocation in railway vehicle maintenance, demonstrating the applicability of predictive modeling in enhancing operational efficiency and reducing maintenance costs. Their approach aligns with the objectives of this research, reinforcing the potential of machine learning techniques in predictive maintenance applications.

Several studies have explored the application of machine learning in predictive maintenance and repair rate estimation. Numsong et al. (2023) [3] introduced an artificial neural network (ANN)-based model for estimating repair costs in agricultural machinery, demonstrating low error rates. Ouadah (2022) [4] discussed predictive maintenance approaches using machine learning, emphasizing the advantages over traditional maintenance models. Dhada et al. (2022) [5] investigated failure rate modeling using recurrent neural networks (RNNs), demonstrating the effectiveness of RNNs in capturing temporal dependencies. Predictive maintenance strategies have also evolved through a combination of advanced modeling techniques, including fuzzy logic, machine learning, and mathematical optimization [6–11].

In line with these developments, recent studies have further advanced predictive maintenance through the integration of deep learning and data driven methods. Li et al. (2024) [12] presented a comprehensive survey of deep learning driven architectures for predictive maintenance, highlighting their scalability, adaptability, and efficiency in complex industrial environments. Benhanifia et al. (2025) [13] conducted a systematic review of predictive maintenance applications in manufacturing, demonstrating the effectiveness of artificial intelligence (AI) and Internet of Things (IoT) integration in improving reliability and operational sustainability. Kim et al. (2024) [14] proposed an LSTM-based approach for predicting maintenance costs in infrastructure management, achieving substantial improvements in forecast accuracy. Aminzadeh et al. (2025) [15] developed an IoT-enabled predictive maintenance framework using machine learning and cloud analytics for industrial air-compressor systems, confirming the benefits of real-time sensor data analytics for fault prediction. In [16] compared various deep learning models for predictive maintenance in manufacturing, concluding that hybrid neural architectures provide superior performance in fault diagnosis and maintenance optimization.

These findings support the motivation of the our research, which applies NN architectures to improve repair rate estimation accuracy and computational efficiency within PBL systems.

Previous studies have also addressed availability-based maintenance analysis for systems operating under repair time thresholds, offering additional insights into optimizing maintenance planning within PBL frameworks [17]. These studies provide theoretical support for linking availability targets with repair rate estimation, which aligns with the objectives of the present research.

To address the limitations of traditional stochastic approaches, this study proposes a data driven alternative using deep neural networks (DNNs) trained on repair rate samples

derived from a stochastic model. The objective is to enable rapid and accurate repair rate predictions while maintaining adaptability to different system configurations and operational environments. The proposed methodology focuses on two NN architectures: (1) a Fully Connected Neural Network (FCNN) that predicts maximum repair rates, capturing steady-state maintenance behaviors, and (2) a Long Short-Term Memory (LSTM) network that estimates minimum repair rates by leveraging sequential learning to model temporal dependencies in repair processes. This separation is also practically motivated: maximum repair rates reflect long-term maintenance capacity, whereas minimum repair rates fluctuate with short-term operational variability, making the two indicators fundamentally different in behavior.

The main contributions of this research are as follows:

1. An NN-based framework for repair rate estimation within PBL systems, trained on samples generated from a stochastic model.
2. A detailed comparative analysis of FCNN and LSTM architectures across different dataset sizes and training configurations.
3. Evidence that the proposed approach provides high predictive accuracy and scalability while significantly reducing computational cost.
4. A discussion on the practical implications and potential integration of the proposed framework into real-time predictive maintenance systems.

This paper contributes to advancing predictive maintenance research by demonstrating the effectiveness of deep learning for accurate and efficient repair rate estimation within PBL systems. Unlike existing predictive maintenance studies, our work focuses specifically on the joint estimation of maximum and minimum repair rates within PBL frameworks, which to the best of our knowledge has not been addressed using deep learning techniques. For clarity and ease of reference, Table 1 provides a summary of the main symbols and abbreviations used throughout the paper.

Table 1. Summary of symbols and abbreviations used in the paper.

Symbol/Abbreviation	Description
A	Target system availability
x	Rayleigh distribution parameter
x_0	Expected value of the Rayleigh parameter x
y	Random variable uniformly distributed on [0, 1]
μ_r	Repair rate (1/MTTR)
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
CDF	Cumulative Distribution Function
DNN	Deep Neural Network
FCNN	Fully Connected Neural Network
LSTM	Long Short-Term Memory Network
MAE	Mean Absolute Error
MSE	Mean Squared Error
NN	Neural Network
PBL	Performance-Based Logistics
PDF	Probability Density Function
RMSE	Root Mean Squared Error
R-squared	Coefficient of Determination

2. System Model

2.1. Overview of the Used Stochastic Model

Building directly on the context established in the previous sections, the stochastic model proposed by Kontrec et al. (2018) [1] serves as a foundational step toward improving repair rate estimations within performance-based logistics frameworks. In this study, the authors specifically consider a system composed of several distinct components, each characterized by its own failure rate. This modeling approach assumes that failure times follow a Rayleigh distribution, commonly utilized in reliability engineering. The model introduces key probabilistic measures, such as the repair rate PDF and CDF, to determine the likelihood of different repair rates occurring.

The Rayleigh model describes the stochastic nature of failure processes, where the repair rate is treated as a random variable dependent on system availability. The failure-time variable x is modeled using the Rayleigh distribution with scale parameter x_0, whose probability density function is given by [1]:

$$f_X(x \mid x_0) = \frac{2x}{x_0} \exp\left(-\frac{x^2}{x_0}\right), \qquad x > 0. \tag{1}$$

The following equations summarize the analytical form of the model and its probabilistic interpretation. This distribution is widely used in reliability engineering because its linearly increasing hazard rate effectively captures degradation driven failure mechanisms. Moreover, it is a special case of the Weibull family with generalized value of shape parameter, making it a standard benchmark for lifetime modeling. Its analytical tractability enables transparent comparison between classical stochastic estimators and the proposed NN based approach. For the underlying failure-time model, we assume a Rayleigh distribution with scale parameter x_0. This choice leads to a closed-form expression for the repair-rate density $p_{\mu_r}(\mu_r)$ in Equation (4) via a change of variables transformation. In practical terms, the only parameter that needs to be estimated from data is the Rayleigh scale x_0. Given a sample $\{x_1, \ldots, x_n\}$ of failure times, the log-likelihood of x_0 is:

$$\ell(x_0) = n \ln 2 + \sum_{i=1}^{n} \ln x_i - n \ln x_0 - \frac{1}{x_0} \sum_{i=1}^{n} x_i^2, \tag{2}$$

which yields the well-known maximum likelihood estimator:

$$\hat{x}_0 = \frac{1}{n} \sum_{i=1}^{n} x_i^2. \tag{3}$$

In practical applications, Rayleigh distributed failure-time observations may be used to estimate the scale parameter x_0 through the MLE expression given above. However, in this study we follow the standard stochastic formulation adopted in the literature: the values of and the availability parameter x_0 and A are taken directly from validated analytical models in the literature. These fixed parameters are then used to generate repair rate samples by applying inverse transform sampling to the closed-form CDF in Equation (6), ensuring full consistency with the theoretical repair-rate distribution. The repair rate distribution in Equation (4) is then obtained analytically by substituting this estimate into the closed-form expression for $p_{\mu_r}(\mu_r)$; since μ_r is derived through a change of variables transformation, no separate likelihood for μ_r is required.

The PDF of the repair rate μ_r is determined as in [1]:

$$p_{\mu_r}(\mu_r) = \frac{8A^2}{(1-A)^3 \mu_r^3 \, \pi x_0} \exp\left(-\frac{4A^2}{(1-A)^2 \, \mu_r^2 \, \pi x_0}\right). \tag{4}$$

This equation expresses the PDF of the repair rate μ_r, showing how the likelihood of a certain repair rate depends on the system availability A, the Rayleigh distribution parameter x_0, and the shape of the exponential term. Higher availability values increase the concentration of the PDF around larger repair rates, meaning that systems with stricter reliability requirements require faster or more frequent repairs.

The corresponding CDF integrates the PDF and represents the probability that the repair rate is less than or equal to a given value. It therefore provides a convenient way to determine cumulative probabilities and to generate random samples. The CDF formula is given as in [1]:

$$F_{\mu_r}(\mu_r) = 1 - \exp\left(\frac{-4A^2}{(1-A)^2\mu_r^2\pi x_0}\right) \tag{5}$$

From this expression, repair rate samples can be generated using the inverse transform sampling method, where a random variable y uniformly distributed in the range [0, 1] is substituted into the inverse of the CDF. This allows the generation of realistic stochastic repair rate data for numerical simulations.

The repair rate samples can be represented as [1]:

$$\mu_r = \sqrt{-\frac{(1-A)^2\mu_r^2\pi x_0}{4A^2}\ln(1 - F_{\mu_r}^{-1}(\mu_r))} = \sqrt{-\frac{(1-A)^2\mu_r^2\pi x_0}{4A^2}\ln(1-y)} \tag{6}$$

where y is a random number from a uniformly distribution. This sampling method provides a straightforward way to create numerical datasets that reflect the stochastic variability of the repair process, which can later be used to train and validate predictive machine learning models.

To validate the stochastic model described in [1], numerical simulations were conducted on a system comprising three distinct components with failure rates: 0.29, 0.58, and 0.84. Each component was analyzed under three different system availability levels: 0.7, 0.8, and 0.9. The PDF and CDF were computed for each case to examine the behavior of repair rates under varying reliability constraints.

The following discussion summarizes the main simulation results to illustrate how the model behaves under different conditions.

The PDF plots illustrate the probability distribution of repair rates, showing how system availability affects the required maintenance efforts. As can be noticed from Figures 1–3, higher availability levels ($A = 0.9$) lead to a shift in the PDF toward higher repair rates, indicating a need for more frequent and efficient maintenance interventions. Lower availability levels ($A = 0.7$ and $A = 0.8$) allow for broader distributions, where lower repair rates remain more probable. Components with higher failure rates (0.84) exhibit a more narrow and peaked PDF, meaning that their repair rates are more predictable and concentrated within a certain range.

The CDF plots provide insight into the cumulative probability of the repair rate being below a certain threshold. Steeper CDF curves for $A = 0.9$ suggest that high repair rates are required with greater certainty, as maintaining high availability imposes strict maintenance requirements. More gradual CDF curves for $A = 0.7$ and $A = 0.8$ indicate that lower repair rates are more common, reducing the intensity of required maintenance interventions. Across all cases, higher failure rates result in steeper CDFs, confirming that more frequent repairs are essential for components with high failure probabilities. These are presented in Figures 4–6.

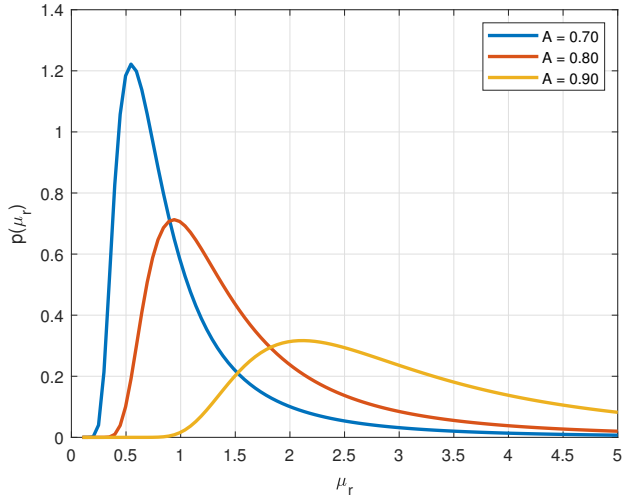

Figure 1. PDF of the repair rate for the first part (failure rate = 0.29).

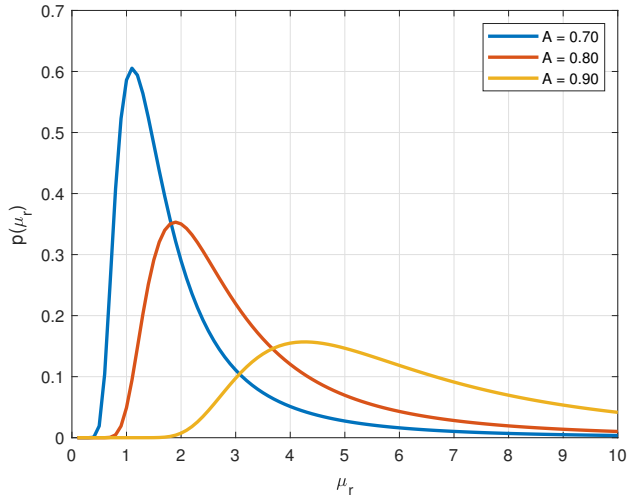

Figure 2. PDF of the repair rate for the second part (failure rate = 0.58).

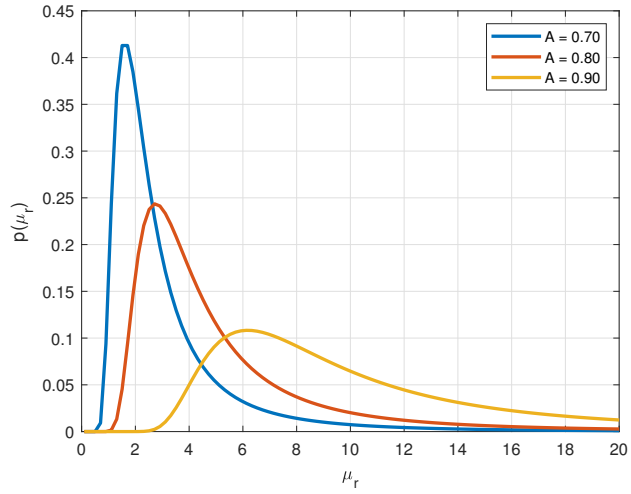

Figure 3. PDF of the repair rate for the third part (failure rate = 0.84).

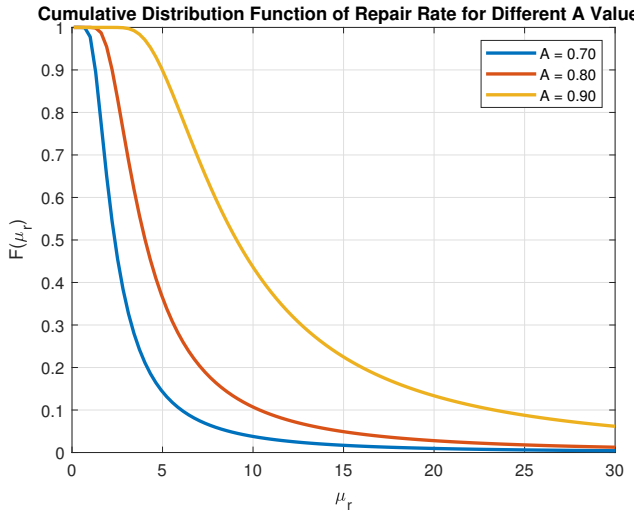

Figure 4. CDF of the repair rate for the first part (failure rate = 0.29).

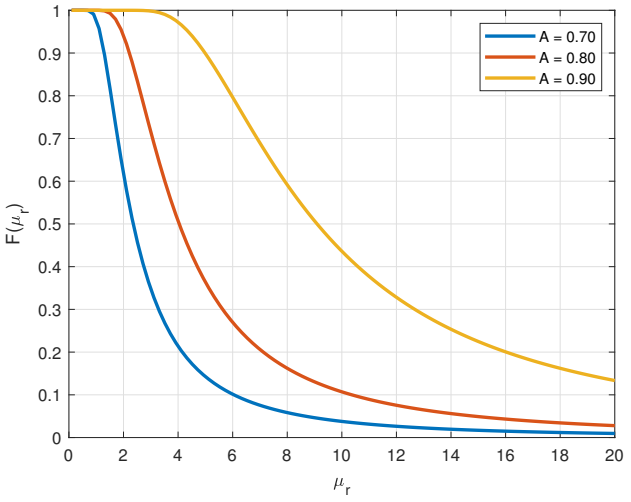

Figure 5. CDF of the repair rate for the second part (failure rate = 0.58).

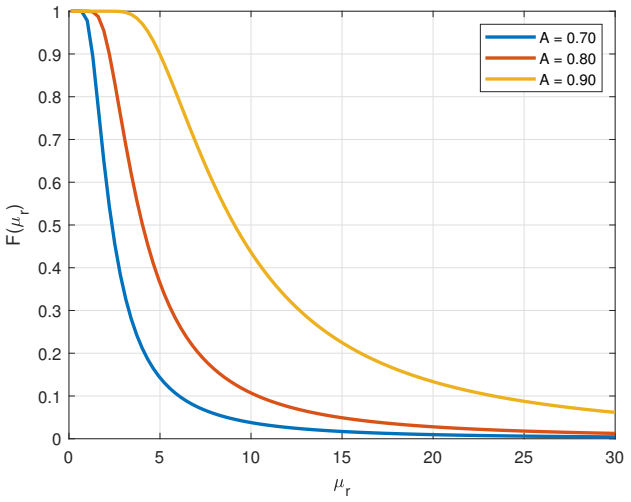

Figure 6. CDF of the repair rate for the third part (failure rate = 0.84).

The numerical results confirm that the stochastic model accurately predicts repair rate distributions under different operating conditions. Specifically, higher failure rates necessitate higher repair rates, validating the theoretical assumptions of the model. Increased availability constraints shift the repair rate distributions, aligning with expectations

that greater availability requires more proactive maintenance. PDF and CDF trends follow standard probabilistic behavior, demonstrating the reliability of the model in estimating repair rates. The numerical analysis supports the effectiveness of the stochastic model in predicting repair rate distributions for different system configurations. The results confirm that both failure rates and availability levels significantly impact repair rate requirements, reinforcing the applicability of the model in predictive maintenance planning and logistics optimization. This validation further strengthens the case for integrating machine learning techniques to enhance repair rate estimations and automate maintenance decision-making.

To further support predictive maintenance strategies, extreme values such as maximum and minimum repair rates are also computed. The maximum repair rate $x_{\max} = \max(\mu_{r_1}, \mu_{r_2}, \ldots, \mu_{r_n})$ and minimum repair rate $x_{\min} = \min(\mu_{r_1}, \mu_{r_2}, \ldots, \mu_{r_n})$ play a crucial role in evaluating the robustness and efficiency of repair processes in PBL systems. These extreme values provide insights into the best-case and worst-case repair scenarios, which are essential for risk assessment, maintenance scheduling, and predictive modeling. The minimum repair rate $x_{\min}$ represents the longest expected repair duration, corresponding to situations where system recovery is slow due to operational constraints, resource limitations, or unexpected failures. This value helps in defining worst case maintenance scenarios and ensuring that systems remain operational even under adverse conditions. Conversely, the maximum repair rate $x_{\max}$ represents the shortest expected repair time, which is important for identifying optimal maintenance conditions and achievable system recovery performance. By analyzing $x_{\max}$, maintenance planners can assess whether a given system meets contractual availability targets and optimize maintenance resource allocation. By considering both $x_{\max}$ and $x_{\min}$, decision-makers can establish realistic repair time expectations, optimize maintenance cycles, and design predictive maintenance strategies that account for variability in repair durations, ultimately enhancing system reliability and operational efficiency. In [9] a novel method for the determination of the maximum and minimum repair rates of the entity comprising the observed units is presented. The obtained generalized PDF expressions can be used to predict total repair time. The presented method provides two new measures that comprehensively define the total repair time and have not been studied in this way before. In the numerical section, the proposed model was applied to the before mention system consisting of three key components. PDFs of repair rate for each component, as well as the PDF and CDF maximum and minimum repair rates for the entire system. In this case selection of three random variables $(\mu_{r_1}, \mu_{r_2}, \mu_{r_3})$ for computing the maximum and minimum repair rates is based on the practical constraints of system redundancy and maintenance decision-making. Many real-world systems operate with a limited number of redundant components or subsystems (typically three or fewer), where each component undergoes independent repair processes under varying operational conditions. Observing a higher number of random variables would increase computational complexity without significantly improving the accuracy of repair rate estimation, as beyond a certain threshold, additional components contribute diminishing returns in predictive maintenance optimization. To accurately model the stochastic nature of the system repair rate, we implement a sampling-based approach that generates random repair rate samples for different system availabilities A_1, A_2, A_3. This process enables us to analyze the variability in repair rate estimations under different operational conditions. Given the system parameters representing different component characteristics, and availability levels A_1, A_2, A_3, the repair rate samples are computed separately for different components of the system, resulting in three vectors of repair rate realizations: $\mu_{r_1} = [\mu_{11}, \mu_{12}, \mu_{13}]$, $\mu_{r_2} = [\mu_{21}, \mu_{22}, \mu_{23}]$, $\mu_{r_3} = [\mu_{31}, \mu_{32}, \mu_{33}]$, where each vector represents repair rates for different configurations of system availability. To analyze

the best-case and worst-case scenarios, we compute the minimum and maximum repair rates across the three sampled vectors:

$$x_{\min} = \min(\mu_{r_1}, \mu_{r_2}, \mu_{r_3}) \tag{7}$$

and

$$x_{\max} = \max(\mu_{r_1}, \mu_{r_2}, \mu_{r_3}) \tag{8}$$

These computations provide insight into the lower and upper bounds of repair rate estimates under varying operational conditions, ensuring robust predictive maintenance planning.

The obtained stochastic repair rate samples, including their minimum and maximum values, serve as the foundation for training data in the subsequent NN analysis. By connecting the analytical framework of the stochastic model with data-driven predictive modeling, this approach enables efficient estimation of repair rates under varying operational conditions. The following subsection introduces the NN architectures developed for this purpose.

2.2. Neural Network Architecture

An empirical inspection of the simulated repair rate samples confirms the rationale for employing two separate neural models. Maximum repair rate values tend to cluster around their upper stochastic limits and vary smoothly, without exhibiting sequential irregularities, therefore a feed-forward FCNN is sufficient for capturing their steady state behavior. Minimum repair-rate values, however, display pronounced local fluctuations and occasional abrupt drops, making their prediction more sensitive to short-term variations. Such fluctuation patterns are more effectively handled by an LSTM architecture, which is designed to model irregular or context dependent changes across adjacent samples even when the overall sequence is not strongly time dependent.

Building on the stochastic model results, two NN architectures were developed to predict repair rates more efficiently and adaptively. The generated repair rate samples were used as inputs, allowing the networks to learn complex nonlinear relationships between system availability and repair performance.

The first model is an FCNN, designed to predict the maximum repair rates $(x_{1max}, x_{2max}, x_{3max})$ based on nine input features. Its architecture, shown in Figure 7, consists of three fully connected (linear) layers. The first layer comprises 128 neurons, the second layer 64 neurons, while the output layer contains three neurons responsible for predicting the target values. Each hidden layer utilizes the Rectified Linear Unit (ReLU) activation function, which facilitates efficient learning of nonlinear relationships in the data. The ReLU activation function is defined as follows [18]:

$$\mathrm{ReLU}(z) = \max(0, z) \tag{9}$$

where z is the input to the neurons. This model is well-suited for regression tasks since the output layer does not employ an activation function, allowing for continuous value predictions. The network is trained using the Mean Squared Error (MSE) loss function, which minimizes the squared differences between the actual and predicted values. In addition to the MSE loss function, other metrics were used to further evaluate the model's performance: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R-squared). MAE measures the average absolute prediction error, while RMSE emphasizes larger errors by penalizing them through squaring. R-squared assesses how well the model explains the variability in the data, with values closer to 1 indicating better model fit.

To improve training stability, batch normalization was also tested after each fully connected layer. This technique normalizes activations across mini-batches, reducing internal covariant shift and allowing for higher learning rates [19]. However, the results did not improve as expected, and in some cases, performance slightly degraded. Therefore, batch normalization was ultimately not included in the final model configuration.

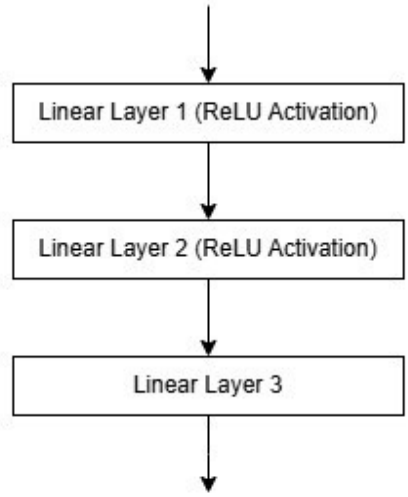

Figure 7. FCNN Architecture.

The optimization of parameters is performed using the Adaptive Moment Estimation (Adam) algorithm with an initial learning rate of 0.001 and L2 regularization (weight decay) of 10^{-4}, which helps mitigate overfitting. Adam combines the benefits of momentum and adaptive learning rates, enabling more stable and efficient convergence [20].

Before training, all input data were standardized using the Standard Scaler (SS) method to avoid feature imbalance and accelerate convergence. This transformation ensures that the data have a zero mean and unit standard deviation, facilitating faster and more efficient model learning. The transformation is performed using the formula:

$$x^* = \frac{x - \mu}{\sigma} \tag{10}$$

where μ and σ represent the mean and standard deviation of each feature, respectively [21].

The dataset was divided into training and testing subsets, with 80% used for model fitting and 20% reserved for performance evaluation. This ensured objective validation of the model's generalization capability.

To ensure a fair comparison between the proposed neural networks and the stochastic baseline estimators, all models were evaluated using the exact same simulation samples and under identical computational conditions. The stochastic MLE estimators and both neural architectures were exposed to the same number of repair rate realizations, identical Rayleigh parameter distributions, and consistent runtime constraints. This guarantees that any observed differences in performance arise solely from methodological effectiveness rather than dataset variation or computational advantage.

3. Results and Discussion

In this study, both FCNN and the LSTM NN demonstrated excellent performance in estimating the maximum and minimum repair rates, respectively. The performance of these NNs was evaluated across various configurations by modifying the optimizer, the number of neurons, and the learning rate. Four optimization algorithms were considered: Adam, SGD, RMSprop, and Adagrad. Experiments were conducted with network

architectures consisting of 64, 128, and 256 neurons. For the FCNN, the learning rates were set to 0.01, 0.001, and 0.0001, whereas for the LSTM network, they were adjusted to 0.025, 0.0025, and 0.00025. The initial training and validation were performed on a dataset containing 10,000 samples, followed by further evaluations on larger datasets with 100,000 and 1,000,000 samples to assess the generalization capability of the models. In the case of the FCNN, the Adam optimizer exhibited the best overall performance across all tested configurations. This was evident from its consistently lower MSE, MAE, and RMSE values, as well as its higher R-squared score compared to the other optimizers. Adam's ability to adapt learning rates dynamically for each parameter allowed it to converge more efficiently while avoiding the pitfalls of suboptimal local minima. In contrast, SGD and RMSprop struggled with convergence stability, and Adagrad, while performing reasonably well, exhibited slower learning behavior, particularly in later training stages. These findings indicate that Adam provides a well-balanced combination of stability and rapid convergence, making it the most suitable optimizer for training the FCNN model in this study. The results of this analysis are summarized in Table 2, which presents the performance metrics obtained for different optimizers.

Table 2. Performance comparison of optimizers (Adam, SGD, RMSprop, Adagrad) on the FCNN model ($N = 10$ k).

Test	Metric	ADAM	SGD	RMSprop	Adagrad
Test 1. (100 epochs)	MSE	0.0183	0.0040	0.0044	0.0010
	MAE	0.1073	0.0357	0.0482	0.0239
	RMSE	0.1354	0.0635	0.0660	0.0315
	R-squared	0.9850	0.9967	0.9964	0.9992
Test 2. (400 epochs)	MSE	0.0001	0.0001	0.0067	0.0002
	MAE	0.0058	0.0078	0.0544	0.0095
	RMSE	0.0087	0.0106	0.0818	0.0124
	R-squared	0.9999	0.9999	0.9945	0.9999
Test 3. (700 epochs)	MSE	0.0001	0.0001	0.0062	0.0001
	MAE	0.0043	0.0044	0.0585	0.0042
	RMSE	0.0057	0.0063	0.0786	0.0058
	R-squared	1.0000	1.0000	0.9949	1.0000

To further improve the performance of the FCNN, the impact of batch normalization and L2 regularization was investigated. The Adam optimizer was selected as the baseline, given its superior performance in the previous analysis. The analysis included four configurations: with batch normalization and L2 regularization, with batch normalization and without L2 regularization, without batch normalization and with L2 regularization, and without both batch normalization and L2 regularization. The results of this comparison are presented in Table 3.

Among these configurations, the best performance was observed in the model trained without batch normalization but with L2 regularization. This suggests that L2 regularization effectively prevents overfitting by penalizing large weights, leading to a more stable and generalizable model. On the other hand, batch normalization did not contribute significantly to performance improvement in this specific case, possibly due to the fully connected network structure and the nature of the dataset. These results indicate that, for the FCNN, L2 regularization should be prioritized over batch normalization when optimizing model performance. Consequently, best configuration for the FCNN model was found to be without batch normalization but with L2 regularization, achieving near perfect R-squared values and the lowest error metrics. The unexpectedly poor performance of batch normalization at 100 epochs may suggest that early-stage normalization disrupts

stable weight updates, while its lack of effectiveness at 400+ epochs indicates that L2 regularization alone is sufficient to prevent overfitting. These findings reinforce that batch normalization is not always beneficial in deep learning models and that its impact should be carefully evaluated depending on the architecture and training conditions.

Table 3. Comparison of FCNN configurations with and without batch normalization and L2 regularization (ADAM, $N = 10$ k).

Test	Metric	Configuration			
		Batch Norm, L2 Reg	**Batch Norm, No L2 Reg**	**No Batch Norm, L2 Reg**	**No Batch Norm, No L2 Reg**
Test 1. (100 epochs)	MSE	0.3110	0.1971	0.0068	0.0183
	MAE	0.3821	0.2966	0.0605	0.1073
	RMSE	0.5577	0.4439	0.0826	0.1354
	R-squared	0.7452	0.8385	0.9944	0.9850
Test 2. (400 epochs)	MSE	0.0030	0.0011	0.0000	0.0001
	MAE	0.0433	0.0236	0.0021	0.0058
	RMSE	0.0550	0.0332	0.0028	0.0087
	R-squared	0.9975	0.9991	1.0000	0.9999
Test 3. (700 epochs)	MSE	0.0010	0.0037	0.0001	0.0001
	MAE	0.0232	0.0333	0.0016	0.0043
	RMSE	0.0312	0.0608	0.0024	0.0057
	R-squared	0.9992	0.9970	1.0000	1.0000

After determining that Adam was the best-performing optimizer, further optimization was conducted by analyzing the impact of different learning rates (0.01, 0.001, and 0.0001) across networks with 64, 128, and 256 neurons. The results confirm that Adam consistently produced superior performance across all configurations. Specifically:

- Learning rates of 0.01 and 0.001 yielded excellent results, with MSE values close to zero and R-squared = 1.0000, indicating near perfect model accuracy.
- A learning rate of 0.0001 significantly degraded performance, leading to higher MSE, MAE, and RMSE values, along with reduced R-squared scores. This suggests that the learning rate was too small, preventing the model from converging efficiently.
- Networks with 128 and 256 neurons achieved the best results, particularly at lr = 0.01 and lr = 0.001, where the error metrics were minimal and R-squared remained at 1.0000.

While SGD and RMSprop also performed well, their results were slightly less consistent than Adam's. In contrast, Adagrad showed poor generalization, with negative R-squared values at lr = 0.0001 for 128 and 256 neurons, indicating a failure to capture meaningful relationships in the data, likely due to its aggressive learning rate decay, which prevents effective weight updates. These findings further confirm that Adam, with a learning rate of 0.01 or 0.001, is the most suitable choice for training the FCNN. The complete results of this analysis are summarized in Table 4. The results confirm that Adam is the most effective optimizer, achieving near-perfect performance across all configurations. While SGD improves with more training and larger networks, RMSprop shows inconsistent results, suggesting that it is highly sensitive to hyperparameters. These findings highlight the importance of careful optimizer selection and hyperparameter tuning to ensure model stability and convergence.

Table 4. Performance comparison of Adam, SGD, RMSprop, and Adagrad optimizers with different learning rates and network sizes for the FCNN model ($N = 10$ k).

Test	Epochs	Metric	Optimizer			
			ADAM	**SGD**	**RMSprop**	**Adagrad**
Test 31.	700, lr = 0.0100, 64 neurons	MSE	0.0001	0.0001	0.0320	0.0002
		MAE	0.0027	0.0042	0.1349	0.0107
		RMSE	0.0122	0.0042	0.1790	0.0151
		R-squared	0.9999	0.9999	0.9737	0.9998
	700, lr = 0.0010, 64 neurons	MSE	0.0001	0.0017	0.0012	1.5352
		MAE	0.0024	0.0228	0.0232	0.9576
		RMSE	0.0035	0.0407	0.0342	1.2390
		R-squared	1.0000	0.9986	0.9990	−0.2578
	700, lr = 0.0001, 64 neurons	MSE	0.8755	0.4151	0.9139	4.7482
		MAE	0.7373	0.5027	0.7250	1.7020
		RMSE	0.9357	0.6443	0.9560	2.1790
		R-squared	0.2827	0.6599	0.2512	−2.8903
Test 32.	700, lr = 0.0100, 128 neurons	MSE	0.0001	0.0001	0.0377	0.0001
		MAE	0.0011	0.0025	0.1601	0.0049
		RMSE	0.0017	0.0038	0.1943	0.0069
		R-squared	1.0000	1.0000	0.9691	1.0000
	700, lr = 0.0010, 128 neurons	MSE	0.0001	0.0006	0.0072	0.3010
		MAE	0.0018	0.0161	0.0599	0.3936
		RMSE	0.0026	0.0241	0.0849	0.5487
		R-squared	1.0000	0.9995	0.9942	0.7534
	700, lr = 0.0001, 128 neurons	MSE	0.0583	0.2636	0.0339	4.8856
		MAE	0.1398	0.4029	0.1109	1.8098
		RMSE	0.2414	0.5134	0.1840	2.2103
		R-squared	0.9522	0.7840	0.9723	−3.0028
Test 33.	700, lr = 0.0100, 256 neurons	MSE	0.0001	0.0001	0.0849	0.0000
		MAE	0.0011	0.0033	0.2240	0.0029
		RMSE	0.0015	0.0045	0.2914	0.0047
		R-squared	1.0000	1.0000	0.9304	1.0000
	700, lr = 0.0010, 256 neurons	MSE	0.0001	0.0012	0.0008	0.0404
		MAE	0.0016	0.0239	0.0254	0.1133
		RMSE	0.0023	0.0344	0.0288	0.2009
		R-squared	1.0000	0.9990	0.9993	0.9669
	700, lr = 0.0001, 256 neurons	MSE	0.0137	0.2597	0.0006	3.8895
		MAE	0.0629	0.4057	0.0190	1.5770
		RMSE	0.1171	0.5096	0.0246	1.9722
		R-squared	0.9888	0.7872	0.9995	−2.1867

To verify the generalization capability of the NN, the best-performing configuration—Adam optimizer with and without batch normalization and L2 regularization—was evaluated on larger datasets containing 100,000 and 1,000,000 samples. The objective was to determine whether the optimal training conditions observed on 10,000 samples would scale effectively to more extensive datasets. The results reaffirmed previous findings:

- Networks trained without batch normalization and with L2 regularization (weight decay) consistently achieved the best performance, maintaining low MSE, MAE, and RMSE values, while preserving an R-squared value close to 1.0000. Batch normalization did not improve accuracy on larger datasets and, in some cases, led to slightly worse generalization performance. This suggests that for this specific problem,

batch normalization does not provide a significant advantage and may even introduce unnecessary variance.

- Increasing the dataset size improved model stability, reducing variability in performance across different training runs. This confirms that the selected model configuration scales effectively and maintains robustness even with significantly larger datasets.

These findings indicate that L2 regularization is essential for preventing overfitting, especially as dataset size increases, while batch normalization does not contribute to performance improvements in this case. The detailed results of this evaluation are provided in Table 5.

Table 5. Evaluation of FCNN configuration on larger datasets (ADAM, 700 epochs).

Test	Dataset	Metric	Configuration			
			Batch, L2 Reg	**Batch, No L2 Reg**	**No Batch, L2 Reg**	**No Batch, No L2 Reg**
Test 13.	100k	MSE	0.0036	0.0001	0.0000	0.0001
		MAE	0.0465	0.0062	0.0013	0.0046
		RMSE	0.0597	0.0087	0.0020	0.0076
		R-squared	0.9971	0.9999	1.0000	1.0000
Test 14.	1M	MSE	0.0002	0.0001	0.0000	0.0000
		MAE	0.0109	0.0085	0.0015	0.0022
		RMSE	0.0153	0.0118	0.0021	0.0033
		R-squared	0.9998	0.9999	1.0000	1.0000

The results were excellent, and determining the best model was challenging due to the similar performance across configurations. The optimal model utilized the Adam optimizer, a learning rate of 0.01 (as seen in Test 31, Test 32, and Test 33, where the MSE is almost 0 and R-squared is nearly 1), with L2 regularization and no batch normalization, trained for 700 epochs on datasets of 10,000 samples. Additionally, the model with 128 neurons produced slightly better results compared to the 256-neuron configuration for all optimizers and learning rates tested. The loss throughout the epochs is shown in Figure 8, illustrating the convergence pattern of the training loss. The training loss decreases rapidly at the beginning, demonstrating the model's ability to efficiently learn from the training data. Interestingly, the combination of batch normalization with L2 regularization resulted in slightly higher MSE at 100,000 samples, suggesting that early-stage weight normalization may introduce additional variance. However, as the dataset size increases to 1M samples, all configurations achieve near-perfect R-squared values (1.0000), demonstrating that larger datasets enhance model stability and generalization. These findings reinforce that L2 regularization should be prioritized over batch normalization in FCNN for predictive maintenance applications. In Figure 9, a comparison between the true and predicted values is displayed, highlighting how closely the model's predictions align with the actual values after training. This comparison emphasizes the model's performance and its capacity to generalize effectively to unseen data. For the LSTM network, similar to the FCNN, various configurations were tested to assess the impact of optimizers, number of neurons, learning rates, and regularization techniques. The performance of the LSTM network was evaluated with the same four optimizers (Adam, SGD, RMSprop, and Adagrad) and with different network architectures, including 64, 128, and 256 neurons. The learning rates were set to 0.025, 0.0025, and 0.00025 to explore the effect of different step sizes during training.

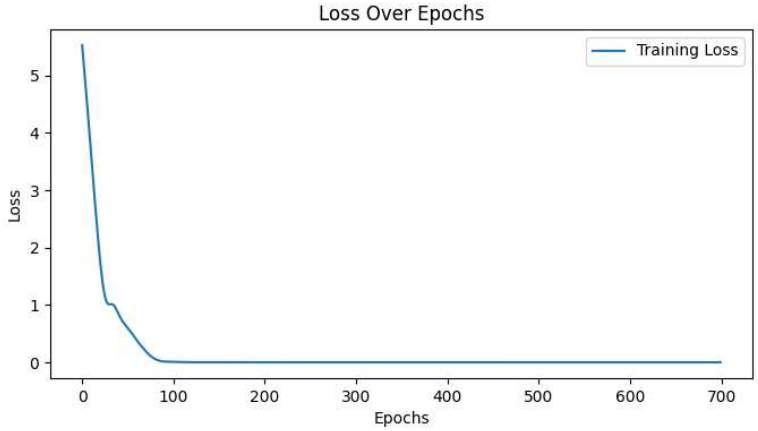

Figure 8. Loss curve during training (700 epochs) for the FCNN model with Adam optimizer and L2 regularization.

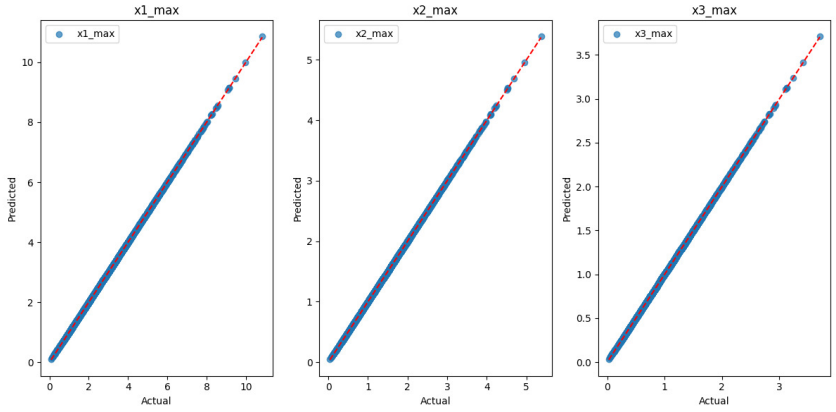

Figure 9. Comparison of true and predicted values for maximum repair rates.

The Adam optimizer again yielded the best performance across all tested configurations for the LSTM network. Its ability to adapt the learning rate dynamically for each parameter resulted in fast convergence, while also maintaining stability during the training process. Adam showed consistently lower MSE, MAE, and RMSE values, as well as higher R-squared scores, compared to other optimizers. This performance pattern was observed across various combinations of neurons and learning rates, reinforcing the findings from the FCNN analysis. In contrast, SGD and RMSprop showed less stability in terms of accuracy, while Adagrad, although performing reasonably well, faced challenges in some configurations. The results of this analysis are summarized in Table 6. SGD performs significantly worse at 100 epochs, with an MSE of 0.2485 and an R-squared value of only 0.0967, indicating that it requires longer training times to converge properly. Interestingly, Adagrad starts with higher error values at 100 epochs but improves significantly over longer training durations, suggesting that it benefits from extended training cycles but struggles in early learning stages. These findings highlight the importance of selecting optimizers based on training duration, with Adam providing the best balance of convergence speed and accuracy for this predictive maintenance application.

Next, the impact of regularization techniques was examined for the LSTM network. The results indicated that the best performance was achieved in the configuration without batch normalization and without L2 regularization, as shown in Table 7. This configuration demonstrated the lowest error metrics (MSE, MAE, and RMSE) and maintained high R-squared values, suggesting that for this specific task, L2 regularization and batch normalization did not contribute significantly to improving performance. While L2 regularization often helps prevent overfitting, its effect was not as pronounced in the LSTM model

as it was in the FCNN. Additionally, batch normalization did not provide a noticeable improvement in generalization for the LSTM network, supporting the conclusion that it is not always beneficial for every type of network architecture. The best configuration was found to be without batch normalization and without L2 regularization, achieving the lowest MSE and highest R-squared values. Surprisingly, batch normalization combined with L2 regularization resulted in the highest error values at 100 epochs, suggesting that batch normalization interferes with the sequential nature of LSTMs, disrupting information flow between time steps. Additionally, L2 regularization appears to be less effective for LSTMs than for FCNNs, likely because LSTMs already include internal mechanisms (gates) that regulate weight updates and prevent overfitting. These findings highlight that regularization techniques that work well for FCNNs may not necessarily benefit recurrent architectures like LSTMs.

Table 6. Performance comparison of optimizers (Adam, SGD, RMSprop, Adagrad) on the LSTM model ($N = 10$ k).

Test	Epochs	Metric	Optimizer			
			ADAM	**SGD**	**RMSprop**	**Adagrad**
Test 1.	100	MSE	0.0003	0.2485	0.0015	0.0026
		MAE	0.0123	0.3665	0.0272	0.0344
		RMSE	0.0184	0.4985	0.0386	0.0506
		R-squared	0.9988	0.0967	0.9946	0.9907
Test 2.	400	MSE	0.0003	0.0082	0.0015	0.0001
		MAE	0.0111	0.0491	0.0283	0.0067
		RMSE	0.0139	0.0905	0.0388	0.0109
		R-squared	0.9993	0.9703	0.9945	0.9996
Test 3.	700	MSE	0.0001	0.0040	0.0008	0.0002
		MAE	0.0049	0.0361	0.0196	0.0116
		RMSE	0.0073	0.0632	0.0279	0.0154
		R-squared	0.9998	0.9855	0.9972	0.9991

Table 7. Comparison of LSTM NN configurations with and without batch normalization and L2 regularization (ADAM, $N = 10$ k).

Test	Epochs	Metric	Configuration			
			Batch, L2 Reg	**Batch, No L2 Reg**	**No Batch, L2 Reg**	**No Batch, No L2 Reg**
Test 1.	100	MSE	0.0036	0.0007	0.0004	0.0003
		MAE	0.0387	0.0132	0.0134	0.0123
		RMSE	0.0603	0.0259	0.0200	0.0184
		R-squared	0.9868	0.9976	0.9985	0.9988
Test 2.	400	MSE	0.0022	0.0003	0.0027	0.0003
		MAE	0.0381	0.0120	0.0386	0.0111
		RMSE	0.0472	0.0159	0.0521	0.0139
		R-squared	0.9919	0.9991	0.9901	0.9993
Test 3.	700	MSE	0.0018	0.0001	0.0001	0.0001
		MAE	0.0296	0.0058	0.0074	0.0049
		RMSE	0.0422	0.0109	0.0111	0.0073
		R-squared	0.9935	0.9996	0.9996	0.9998

The optimal results for the LSTM model were achieved using the Adam optimizer with a learning rate of 0.025, no batch normalization, and no L2 regularization. Networks with 128 neurons produced the best results, yielding minimal errors and maintaining a

near-perfect R-squared score. These findings are summarized in Table 8. Adam is the best optimizer for LSTM networks, achieving the lowest MSE and highest R-squared values across different learning rates and network sizes. SGD performs poorly at low learning rates, indicating that it does not adapt well to the sequential nature of LSTMs without momentum tuning. Adagrad completely fails at lr = 0.00025, showing negative R-squared values due to excessive learning rate decay, which causes the model to stop learning. RMSprop exhibits inconsistent behavior, performing well in some cases but struggling in others, suggesting that it requires precise tuning for effective LSTM training. These findings highlight that Adam provides the best balance of stability and performance, while other optimizers require careful tuning to be effective.

Table 8. Performance comparison of Adam, SGD, RMSprop, and Adagrad optimizers with different learning rates and network sizes for the LSTM NN model ($N = 10$ k).

Test	Epochs	Learning Rate	Neurons	Metric	Optimizer			
					ADAM	**SGD**	**RMSprop**	**Adagrad**
Test 31.	700	0.025	64	MSE	0.0011	0.0161	0.0105	0.0012
				MAE	0.0217	0.0776	0.0860	0.0284
				RMSE	0.0325	0.1270	0.1026	0.0342
				R-squared	0.9962	0.9414	0.9617	0.9957
		0.0025	64	MSE	0.0012	0.2448	0.0026	0.0073
				MAE	0.0246	0.3621	0.0385	0.0591
				RMSE	0.0350	0.4947	0.0509	0.0852
				R-squared	0.9955	0.1102	0.9906	0.9736
		0.00025	64	MSE	0.0092	0.3443	0.0014	0.8367
				MAE	0.0695	0.4085	0.0295	0.6814
				RMSE	0.0961	0.5868	0.0377	0.9147
				R-squared	0.9665	−0.2518	0.9948	−2.0417
Test 32.	700	0.025	128	MSE	0.0010	0.0028	0.0105	0.0001
				MAE	0.0233	0.0281	0.0794	0.0076
				RMSE	0.0319	0.0529	0.1023	0.0111
				R-squared	0.9963	0.9898	0.9620	0.9996
		0.0025	128	MSE	0.0001	0.1620	0.0027	0.0007
				MAE	0.0080	0.2906	0.0437	0.0164
				RMSE	0.0107	0.4024	0.0520	0.0261
				R-squared	0.9996	0.4112	0.9902	0.9975
		0.00025	128	MSE	0.0003	0.2988	0.0001	0.7653
				MAE	0.0092	0.3854	0.0050	0.7051
				RMSE	0.0160	0.5466	0.0074	0.8748
				R-squared	0.9991	−0.0863	0.9998	−1.7822
Test 33.	700	0.025	256	MSE	0.0002	0.0021	0.0162	0.0004
				MAE	0.0116	0.0223	0.1009	0.0138
				RMSE	0.0157	0.0457	0.1271	0.0190
				R-squared	0.9991	0.9924	0.9412	0.9987
		0.0025	256	MSE	0.0001	0.1896	0.0022	0.0002
				MAE	0.0065	0.3197	0.0331	0.0094
				RMSE	0.0084	0.4354	0.0470	0.0123
				R-squared	0.9997	0.3108	0.9920	0.9995
		0.00025	256	MSE	0.0003	0.3202	0.0001	0.4697
				MAE	0.0106	0.3941	0.0047	0.5983
				RMSE	0.0159	0.5658	0.0073	0.6853
				R-squared	0.9991	−0.1639	0.9998	−0.7075

Overall, these results confirm the superiority of the Adam optimizer for training both FCNN and LSTM models, highlighting its efficiency in handling various architectures and learning rates. Additionally, for the LSTM network, the absence of batch normalization and L2 regularization proved to be the most effective configuration, aligning with the findings from the FCNN model. The results for the larger datasets of 100,000 and 1,000,000 samples are summarized in Table 9, where the metric values show consistency with those achieved using the 10,000 sample dataset, further validating the model's robust generalization capabilities. LSTM models perform best without batch normalization and L2 regularization, with near-zero error metrics across configurations. The slightly higher MSE for batch normalization + L2 regularization at 100,000 samples suggests that batch normalization disrupts temporal learning in LSTMs, while L2 regularization adds little benefit due to the model's built-in gating mechanisms. However, as the dataset size increases to 1M samples, all configurations achieve near-perfect R-squared values (1.0000), demonstrating that large datasets enhance model stability and reduce the need for additional regularization techniques.

Table 9. Evaluation of LSTM configuration on larger datasets (ADAM, 700 epochs).

Test	Dataset	Metric	Configuration			
			Batch, L2 Reg	**Batch, No L2 Reg**	**No Batch, L2 Reg**	**No Batch, No L2 Reg**
Test 13.	100k	MSE	0.0026	0.0001	0.0007	0.0002
		MAE	0.0398	0.0080	0.0189	0.0095
		RMSE	0.0513	0.0108	0.0264	0.0138
		R-squared	0.9905	0.9996	0.9975	0.9993
Test 14.	1M	MSE	0.0269	0.0268	0.0001	0.0001
		MAE	0.1021	0.1020	0.0068	0.0068
		RMSE	0.1639	0.1636	0.0101	0.0100
		R-squared	0.9028	0.9026	0.9996	0.9996

Figure 10 depicts the loss across epochs, highlighting the convergence trend of the training loss. In Figure 11, a comparison of the true and predicted values for the minimum repair rates is presented, showcasing how accurately the LSTM model predicts the actual values after training.

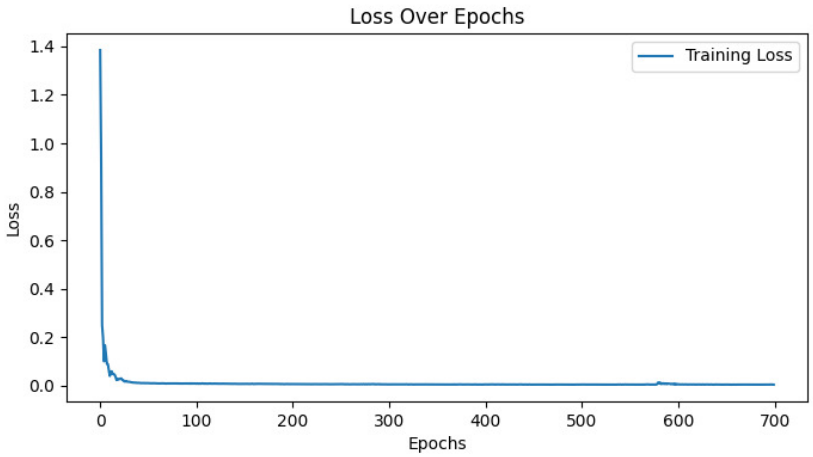

Figure 10. Loss curve during training (700 epochs) for the LSTM model with Adam optimizer.

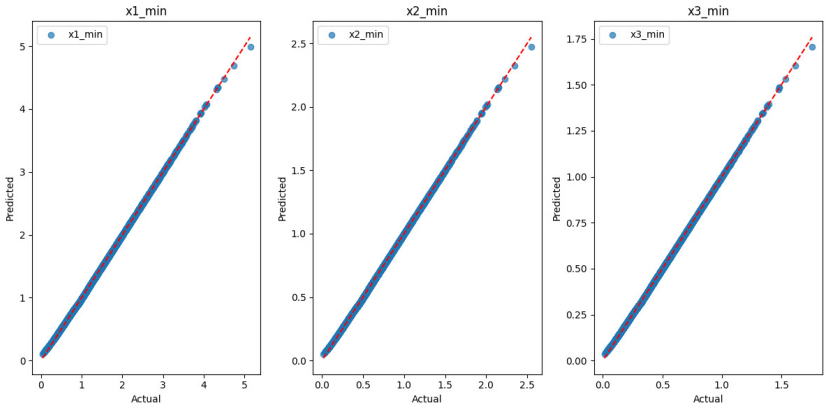

Figure 11. Comparison of true and predicted values for minimum repair rates.

The sensitivity analysis presented in Table 10 illustrates the behavior of both the FCNN and LSTM models under different values of the availability parameter A. By varying A values across three representative scenarios, we observe that the prediction accuracy of both models remains stable for observed availability levels, with performance degradation becoming noticeable only for extreme values. These findings confirm that the models are robust to variations in system availability and therefore well suited for practical implementation in environments where operational conditions may fluctuate. Furthermore, the comparative behavior across architectures provides empirical justification for the modeling choices: while the FCNN performs reliably in non-sequential settings, the LSTM yields superior accuracy in regimes where temporal dependencies within repair-rate sequences play a significant role.

Table 10. Test performance of the FCNN and LSTM models across three A-parameter scenarios ($N = 10$ k).

Model	A-Set	MSE	MAE	RMSE	Train Time (s)
FCNN	0.70, 0.80, 0.90	0.0305	0.1338	0.1746	10.33
FCNN	0.75, 0.85, 0.95	0.1117	0.2526	0.3343	10.33
FCNN	0.60, 0.79, 0.99	0.6335	0.6039	0.7959	10.33
LSTM	0.70, 0.80, 0.90	0.0007	0.0160	0.0256	162.81
LSTM	0.75, 0.85, 0.95	0.1164	0.2791	0.3411	162.81
LSTM	0.60, 0.79, 0.99	0.6036	0.6292	0.7769	162.81

In addition to accuracy metrics, Table 10 reports the corresponding training times and convergence epochs, offering a clear view of the computational feasibility of the proposed approach. The FCNN achieves faster convergence due to its feed-forward structure, whereas the LSTM requires longer training times as a consequence of its recurrent gating operations. Despite this difference, both models maintain execution times compatible with real-time predictive maintenance scenarios, confirming their suitability for deployment in practical environments. This analysis demonstrates that the proposed NNs generalize consistently across varying availability conditions without signs of overfitting, and it provides further justification for selecting FCNN and LSTM architectures as efficient and scalable alternatives to classical stochastic estimators.

Compared with classical Monte Carlo estimation, whose convergence requires a large number of samples, and Bayesian methods that rely on explicit likelihood updates, the proposed neural architectures deliver instant predictions and bypass the need for iterative probabilistic inference. This makes them highly suitable as fast surrogate estimators for real-time repair rate prediction in maintenance planning systems.

Figures 12 and 13 illustrate the comparison between the actual and predicted PDFs for maximum and minimum repair rates, respectively. These graphs demonstrate the effectiveness of the proposed NN-based optimization approach in estimating repair rate distributions within a performance-based logistics system. In Figure 12, the predicted PDF closely follows the actual PDF for maximum repair rates, indicating a high level of accuracy in capturing the underlying distribution of the data.

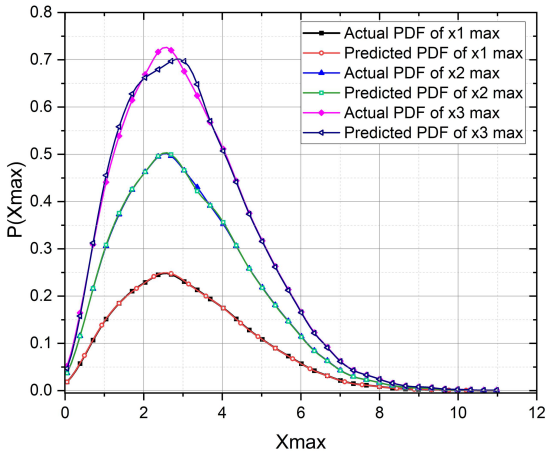

Figure 12. Comparison of actual and predicted probability density functions (PDFs) for maximum repair rates.

Similarly, Figure 13 presents the comparison for minimum repair rates, showing a strong alignment between predicted and actual values. The minor deviations observed in both graphs suggest that the model effectively generalizes across different repair rate scenarios while maintaining a low error margin. These results confirm that the NN-based approach is capable of accurately estimating repair rate distributions, making it a reliable tool for predictive maintenance and logistics optimization.

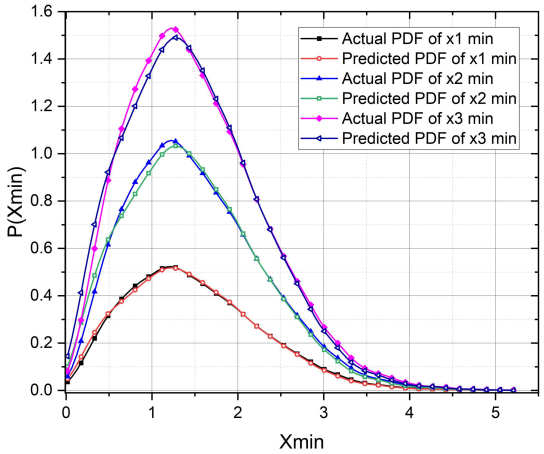

Figure 13. Comparison of actual and predicted probability density functions (PDFs) for minimum repair rates.

4. Conclusions and Future Work

This paper presents a novel NN-based framework for repair rate estimation in PBL systems, bridging the gap between traditional stochastic modeling and modern data-driven predictive maintenance. By training FCNN and LSTM networks on repair rate samples generated from a stochastic model, the framework offers a computationally efficient and flexible alternative to traditional analytical formulations. Unlike conventional analytical ap-

proaches, the proposed method eliminates the need for complex mathematical derivations and provides a scalable, real-time solution that can adapt to diverse operational conditions.

The results suggest that both NN architectures achieve high predictive accuracy, with low error metrics and strong generalization across different dataset sizes and system configurations. The FCNN is effective in modeling maximum repair rates, while the LSTM captures sequential patterns associated with minimum repair rates. Together, these architectures demonstrate that deep learning can replicate and surpass the performance of analytical estimators while drastically reducing computation time, which is critical for real-time repair planning and predictive maintenance applications. These findings indicate that deep learning provides a promising complement to conventional stochastic approaches by reducing computational complexity and enabling near-real-time estimation capabilities.

The primary strengths of this study lie in its integration of reliability engineering principles with advanced AI architectures and its demonstration of strong scalability across multiple availability scenarios. The predicted repair rates offer direct utility in scheduling maintenance interventions, planning spare parts inventory, and managing system availability. By leveraging the predictive precision and fast inference of FCNN and LSTM models, maintenance planners can improve decision-making, minimize downtime, and optimize resource allocation, leading to measurable cost and time efficiencies. In addition to performance improvements, this research demonstrates the potential of integrating machine learning with reliability engineering to support predictive maintenance decision-making within PBL frameworks. The models' adaptability and scalability make them suitable candidates for deployment in maintenance planning and logistics optimization. Across all tested availability scenarios, the FCNN and LSTM architectures consistently achieved higher predictive accuracy and superior numerical stability relative to the stochastic baseline, confirming the practical advantages of data-driven estimation methods.

The present study intentionally employs a minimal feature set composed solely of repair rate samples in order to isolate and evaluate the predictive capabilities of the proposed neural architectures without confounding effects from auxiliary variables. This controlled environment enabled a transparent and rigorous comparison between classical stochastic estimators and the proposed neural models. While this provides a clean and controlled benchmark, future work could benefit from expanding the input space to include operational, environmental, and condition monitoring variables, enabling richer feature representations and improving applicability in real maintenance environments. Further the proposed models could be tested on actual maintenance records through pilot studies within PBL maintenance systems to evaluate real-world applicability and accuracy. Once trained, both architectures are capable of real-time inference, enabling integration into predictive maintenance decision support tools for dynamic repair planning. Future research should therefore incorporate empirical maintenance records to validate the models under realistic conditions and to identify domain-specific patterns not observable in simulations. Such extensions would also facilitate the development of hybrid neural–stochastic frameworks that combine the interpretability of analytical models with the predictive power demonstrated by deep learning in this study.

Author Contributions: M.D., S.P., N.K., D.Đ. and S.M. contributed equally to the conceptualization, methodology, investigation, and writing of this manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: The authors gratefully acknowledge the support from the Serbian Ministry of Science, Technological Development and Innovation (Contract No. 451-03-65/2024-03/200123).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kontrec, N.; Panić, S.; Petrović, M.; Milošević, H. A Stochastic Model for Estimation of Repair Rate for System Operating under Performance-Based Logistics. *Ekspl. Niezawodn.—Maint. Reliab.* **2018**, *20*, 68. [CrossRef]
2. Milković, V.; Osman, K.; Jankovich, D. Reliability-Based Model for Optimizing Resources in the Railway Vehicles Maintenance. *Mech. Eng. J.* **2024**, *11*, 24-00070. [CrossRef]
3. Numsong, A.; Posom, J.; Chuan-Udom, S. Artificial Neural Network-Based Repair and Maintenance Cost Estimation Model for Rice Combine Harvesters. *Int. J. Agric. Eng. Technol.* **2023**, *16*, 38–47. [CrossRef]
4. Ouadah, A.; Zemmouchi-Ghomari, L.; Salhi, N. Selecting an Appropriate Supervised Machine Learning Algorithm for Predictive Maintenance. *Int. J. Adv. Manuf. Technol.* **2022**, *119*, 4277. [CrossRef]
5. Dhada, M.; Parlikad, A.K.; Steinert, O.; Lindgren, T. Weibull Recurrent Neural Networks for Failure Prognosis Using Histogram Data. *Neural Comput. Appl.* **2022**, *35*, 3011. [CrossRef]
6. Kontrec, N.; Panić, S.; Panić, B.; Marković, A.; Stošović, D. Mathematical Approach for System Repair Rate Analysis Used in Maintenance Decision Making. *Axioms* **2021**, *10*, 96. [CrossRef]
7. Sharma, J.; Mittal, M.L.; Soni, G. Condition-Based Maintenance Using Machine Learning and the Role of Interpretability: A Review. *Int. J. Syst. Assur. Eng. Manag.* **2024**, *15*, 1345. [CrossRef]
8. Zhai, S.; Kandemir, M.G.; Reinhart, G. Predictive Maintenance Integrated Production Scheduling by Applying Deep Generative Prognostics Models: Approach, Formulation and Solution. *Prod. Eng.* **2022**, *16*, 65. [CrossRef]
9. Jahangard, M.; Xie, Y.; Feng, Y. Leveraging Machine Learning and Optimization Models for Enhanced Seaport Efficiency. *Marit. Econ. Logist.* **2025** , 1–42. [CrossRef]
10. Gu, X.; Han, J.; Shen, Q.; Angelov, P.P. Autonomous Learning for Fuzzy Systems: A Review. *Artif. Intell. Rev.* **2023**, *56*, 7549–7595. [CrossRef]
11. Pagano, D. A Predictive Maintenance Model Using Long Short-Term Memory Neural Networks and Bayesian Inference. *Decis. Anal. J.* **2023**, *6*, 100174. [CrossRef]
12. Li, Z.; He, Q.; Li, J. A Survey of Deep Learning-Driven Architecture for Predictive Maintenance. *Eng. Appl. Artif. Intell.* **2024**, *133*, 108285. [CrossRef]
13. Benhanifia, A.; Ben Cheikh, Z.; Oliveira, P.M.; Valente, A.; Lima, J. Systematic Review of Predictive Maintenance Practices in the Manufacturing Sector. *Intell. Syst. Appl.* **2025**, *26*, 200501. [CrossRef]
14. Kim, J.-M.; Yum, S.-G.; Das Adhikari, M.; Bae, J. A LSTM Algorithm-Driven Deep Learning Approach to Estimating Repair and Maintenance Costs of Apartment Buildings. *Eng. Constr. Archit. Manag.* **2024**, *31*, 369–389. [CrossRef]
15. Aminzadeh, A.; Sattarpanah Karganroudi, S.; Majidi, S.; Dabompre, C.; Azaiez, K.; Mitride, C.; Sénéchal, E. A machine learning implementation to predictive maintenance and monitoring of industrial compressors . *Sensors* **2025**, *25*, 1006. [CrossRef] [PubMed]
16. Li, W.; Li, T. Comparison of Deep Learning Models for Predictive Maintenance in Industrial Manufacturing Systems Using Sensor Data. *Sci. Rep.* **2025**, *15*, 23545. [CrossRef] [PubMed]
17. Kontrec, N.; Panić, S.; Panić, B. Availability-Based Maintenance Analysis for Systems with Repair Time Threshold. *Yugosl. J. Oper. Res.* **2025**, *35*, 585–593. [CrossRef]
18. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
19. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML), Lille, France, 7–9 July 2015; Volume 37, pp. 448–456.
20. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
21. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825.