Enhancing Data Center Reliability Through Deep Learning-Based Disk Failure Prediction

by

Yassaman Mardan

A thesis proposal submitted to the
Department of Computer Science
in conformity with the requirements for
the degree of Master of Science

Bishop's University
Canada
July 2025

Abstract

With the growing reliance on technology, data storage and failure prediction have become critical and challenging issues. Among various data storage methods, disks are the most commonly used components in storage systems, with the majority of information currently stored on them. As a result, disk failures can lead to irreparable damage. Although such failures are relatively rare, large-scale storage systems containing thousands of disks are still prone to severe failures, often resulting in permanent data loss. For this reason, maintaining the reliability of storage resources has always been a serious challenge. Various methods have been developed to detect and predict disk failures in data centers. However, identifying failures quickly and accurately remains a significant challenge. Due to the poor performance of old methods, researchers are motivated to use different techniques to detect failures earlier and more accurately to cover the weaknesses of the old methods. Deep learning is one of the most advanced techniques used for predicting disk failures. Thus, we train a Bidirectional Long Short-Term Memory (Bi-LSTM) deep neural network to diagnose and predict disk failures effectively. In deep learning-based failure detection methods, selecting effective features plays a crucial role in enhancing the model's accuracy and performance. However, using too many features can increase computational load, add unnecessary complexity, and reduce overall efficiency. To address this, we apply feature selection techniques as part of our methodology. Specifically, we use Pearson correlation and Chi-square tests to identify the most relevant features. The datasets used in this study are from Backblaze and Baidu. The results demonstrate that our model accurately detects failures with 98.36% accuracy and 97.8% precision, and successfully predicts failures up to 40 days in advance. Additionally, we develop a decision tree-based model to evaluate disk health status. This model predicts the remaining useful life of hard disks and classifies them into *Healthy*, *Warning*, or *Critical* states. By classifying the disk health features and then applying the trained Bi-LSTM, we achieve significant improvements, reaching an accuracy of 99.27% and a precision of 98.65%.

Acknowledgments

I would like to express my deepest gratitude to my professors, Prof. Madjid Allili and Prof. Mohammad Ayoub Alaoui Mhamdi, for their invaluable guidance, support, and encouragement throughout the course of this research. Their time, insightful feedback, and expertise have been instrumental in shaping the direction and quality of this thesis. I am truly grateful for their mentorship and for the opportunity to learn under their supervision.

List of Abbreviations

Abbreviation	Definition
HDDs	Hard Disk Drives
SMART	Self-Monitoring Analysis and Reporting Technology
RUL	Remaining Useful Life
RAID	Redundant Array of Independent Disks
Bi-LSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
HMM	Hidden Markov Model
BNN	Binary Neural Network
CDEF	Cloud Disk Error Forecasting
AUC	Area Under the Curve
CRF	Conditional Random Fields
LSTM	Long Short-Term Memory
MSE	Mean Squared Error
MAE	Mean Absolute Error
ReLU	Rectified Linear Unit
MCC	Matthews Correlation Coefficient
CART	Classification and Regression Tree
FPR	False Positive Rate
TPR	True Positive Rate
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives
SGD	Stochastic Gradient Descent

Contents

1 Introduction **1**
1.1 Problem Statement . . . 2

2 Literature Review **4**
2.1 Failure Detection Approach . . . 4
2.2 Disk Health Degree Prediction . . . 5
2.3 Remaining Useful Life Prediction . . . 7

3 Proposed Methodology **14**
3.1 Model Training . . . 26
3.2 Evaluation of Disk Health . . . 31

4 Experimental Results **37**
4.1 Evaluation Indicators . . . 37
4.2 Model Evaluation . . . 38
4.3 Results and Comparison . . . 44
4.3.1 Evaluating the Efficiency of the Trained Model . . . 44
4.3.2 Performance Evaluation for Disk Health Status . . . 49

5 Conclusion and Future Research Directions **54**

Bibliography **55**

List of Tables

2.1 The advantages and disadvantages of the conducted studies on disk failure prediction . . . 9
2.2 Comprehensive overview of disk failure prediction methods and their performance . . . 12

3.1 The number of data used in each disk model . . . 23
3.2 Example of decision rules learned by the decision tree classifier . . . 34

4.1 Hard drives SMART features in the BackBlaze dataset . . . 39
4.2 Selection of the top ten features with Chi-square test . . . 44
4.3 The set of features of Pearson correlation coefficient and Chi-square test . . . 44
4.4 The number of training and test data of failure prediction model . . . 44
4.5 Comparison of hyperparameters for the base and proposed models . . . 45
4.6 Evaluation results of the basic method with Adam's optimizer and 30 days before failure . . . 45
4.7 Evaluation results of the diagnosis model with Adam's optimizer and 40 days before failure . . . 45
4.8 Evaluation results of diagnosis model with SGD optimizer and 40 days before failure . . . 45
4.9 MAE value of the selected features . . . 49
4.10 Results evaluation of the other state-of-the-art methods and the proposed solution . . . 52

List of Figures

3.1 The architecture of an LSTM block 15
3.2 An example of Bi-LSTM network structure 16
3.3 Overview of the work flow for our approach 21

4.1 Correlation and Chi-square test of SMART features 42
4.2 Features correlation matrix 43
4.3 Loss function diagram with 5 time steps 46
4.4 Loss function diagram with 7 time steps 47
4.5 Loss function diagram with 10 time steps 47
4.6 Loss function diagram with 15 time steps 48
4.7 Decision tree for `SMART-5` trained to predict remaining days until failure 51

Chapter 1

Introduction

In today's digital world, the amount of data we generate is growing rapidly, making reliable storage systems more important than ever. Hard drives, which are the most common type of storage, are essential for keeping our data safe. However, when these drives fail, it can lead to serious problems, such as data loss or system outages [25]. This makes it crucial for the accurate prediction of when a disk might fail, allowing these issues to be prevented before they occur. [42].

Predicting disk failures is crucial for effective stock maintenance and budgeting. By anticipating potential failures, data centers can manage their inventory more efficiently, plan proactive replacements, and allocate resources strategically [23]. Early prediction helps organizations avoid emergency purchases, reduce unexpected downtime costs, and optimize the lifecycle of storage assets. Accurate failure prediction also plays a key role in maintaining the operational stability and cost-effectiveness of large-scale storage systems [5].

Hard disk drives (HDDs) have been the main storage solution for large data centers for many years. Like any other electronic device, HDDs do not last forever and eventually wear out. To keep track of their health, manufacturers and operators use a system called SMART (Self-Monitoring Analysis and Reporting Technology). This system records important data about the HDDs, such as how long they have been running, their temperature, error rates, and more [26]. Each manufacturer sets thresholds for these attributes, which are used to detect if a drive might fail. If a drive is working properly, its SMART attributes should stay within these thresholds.

Backblaze and Baidu are two major providers of publicly available disk failure datasets. Backblaze, a cloud storage company, offers detailed SMART data collected from thousands of hard drives in its data centers. Backblaze runs a data center with over 171,000 hard drives [3]. In 2020, they reported that about 0.93% of their drives failed—more than a thousand in one year [18]. This highlights the need for companies and manufacturers to know how much longer their hard drives will last, known as their remaining useful life (RUL). Accurately predicting the RUL can help minimize downtime, prevent disruptions, and protect valuable data. Similarly, Baidu, a leading Chinese technology company, has released large-scale disk failure data from its storage systems. These datasets are widely used in research to develop and evaluate models for predicting disk failures.

While SMART attributes are useful for spotting problems before a drive fails, predicting the RUL would be even more helpful, especially for data centers without RAID (redundant array of independent disks) technology. This data storage technology groups multiple physical disk drives into one or more logical units to enhance data redundancy, performance, or both. However, they are expensive and managing and configuring them is complex. Thus, accurately predicting the RUL would give data centers time to back up their data and plan for a replacement before the drive stops working. For companies like Backblaze, it would mean they could prepare to swap out failing drives without impacting their systems. Predicting RUL not only improves reliability but also ensures smoother operations in large-scale data centers.

1.1 Problem Statement

Hard disk drives are essential for data storage in large-scale systems, but their limited lifespan can lead to failures that cause data loss and downtime. Existing monitoring systems like SMART provide basic health tracking but often fail to predict failures early enough for preventive action. Current prediction methods struggle with adaptability, accuracy, and generalization across different drive models, leaving data centers unprepared to manage failures effectively. This research aims to address these challenges by developing a robust machine learning-based approach to accurately predict disk failure in a timely manner, ensuring better data safety and system reliability.

The main problem this research tackles is finding a better way to detect and predict disk failures, especially in large data centers. Despite advances in technology, traditional methods still have significant weaknesses. They can be complicated to use, may not work well with different types of data, and often struggle to keep up with changes in how disks fail over time [24, 25, 30].

We propose a method for disk failure detection and prediction using Bidirectional Long Short-Term Memory (Bi-LSTM) networks, a type of deep learning model well suited for time series data. To further improve model performance, key features from SMART attributes are selected using statistical methods like Pearson's correlation coefficient and the Chi-square test. These techniques ensure that only the most relevant features are used, reducing complexity and enhancing prediction accuracy.

This research has three main goals. First, we aim to create a strong model that can predict disk failures using advanced machine learning techniques. Second, we want to make sure that this model works well across different types of data and disk models, not just a single type. Third, we hope to improve how early we can predict disk failures, giving us more time to take action and avoid any negative consequences.

This thesis is organized as follows. Chapter 2 reviews the existing research and methods for the prediction of disk failures. Chapter 3 explains our proposed approach in detail, including how we prepare the data, train the model, and evaluate its performance. Chapter 4 presents the results of our experiments and compares our method with others. Finally, Chapter 5 concludes the research and suggests possible future directions for further improvement.

Chapter 2

Literature Review

This section begins with a general overview of disk failure detection and prediction. We then review the main categories of studies focused on evaluating disk health and predicting the remaining useful life of hard disks. Finally, we compare state-of-the-art methods from existing research in this field. A wide range of research has been conducted in the area of disk failure detection and prediction, focusing on various strategies for identifying potential failures and estimating the remaining useful life of hard drives. These studies explore different modeling techniques, data-driven approaches, and performance metrics to improve reliability and prevent data loss.

2.1 Failure Detection Approach

In failure detection approaches, the model typically evaluates the current state of a disk and classifies it as either healthy or unhealthy based on the observed features at a given point in time. This type of classification is limited to the present condition of the disk without considering its future behavior. In contrast, some extended methods go beyond detecting the current state and aim to estimate how much time remains before a potential failure occurs. This leads to the concept of remaining useful life (RUL) prediction, which is further discussed in the next section. Up to now, most of the conducted studies have focused solely on their individual research approach. However, Chang et al. [41] have presented a model that combines three approaches based on failure detection, health degree of disks, and remaining useful life prediction using a Recurrent Neural Network (RNN). In this discussion, we will first examine the failure detection-based approach, and then delve into the other two approaches in their respective sections. In their proposed model, Chang et al. [41] utilized a real-world dataset used in the paper by Zhou et al. [47] for model evaluation. They also employed three non-parametric methods, including the reversed rank test, rank-biserial correlation test, and Z-score [25], to select SMART features. Ultimately, 10 top features were selected. In the evaluation section of failure prediction, Chang and his colleagues compared their proposed method in this model with the following methods: the Hidden Markov Model (HMM), Binary Neural Network (BNN) as an artificial neural network, and CT, a classification tree-based method. In both the CT and BNN approaches, failure detection relied on a thresholding algorithm. Specifically, the model evaluated the most recent sequence of past samples prior to the current time point, where the size of the detection window determined how many previous observations were considered. If more than half of these samples were classified as failure instances, the disk was flagged as faulty.

In another study, Jiang and his colleagues proposed a new disk failure prediction model using Random Forests [40]. In their conducted research, they applied their proposed model to the Backblaze dataset. The model started with using the Wilcoxon Rank-Sum Test to differentiate between positive and negative features. Then recent samples for each disk were stored. Labels were assigned to each sample, positive for failure and negative for healthy status. Due to the imbalance between the number of healthy and faulty samples, they employed a method called Poisson-distributed stratified sampling [27]. During the evaluation, they compared the proposed model with Offline Random Forests, SVM, and decision trees. The results indicated that the Offline Random Forest model had superior predictive performance compared to SVM and decision trees. Furthermore, the Random Forest model exhibited stable failure detection rates ranging from 93% to 99%, outperforming all offline models.

As mentioned earlier, unlike offline disk failure prediction approaches, which have access to disk failure-related data before model construction, the offline models continually generate evolving data streams. These evolving models adapt to changing statistical patterns over time. To address these challenges, Han et al. [14] introduced an online Random Forest method named STREAMDFP. In this approach, a holistic stream extraction for disk failure prediction is utilized, incorporating concept drift adaptation.

2.2 Disk Health Degree Prediction

This approach evaluates disk condition by classifying it into health levels such as healthy, warning, or failed. Unlike methods that rely on a single snapshot, it analyzes sequences of disk states over time. By tracking how SMART attributes change across multiple time steps, the model can more effectively detect gradual degradation patterns that often lead to failure. This temporal perspective enables more accurate health assessments, enhances the reliability of early warnings, and reduces the risk of misclassification compared to approaches based solely on isolated data points [40].

Continuing from the previous study by Chang et al. [41], where they discussed the proposed recurrent neural network based fault detection model, in this section, they evaluated the performance of the proposed model in terms of disk health. The proposed model consists of three layers: input, output, and hidden layers. At each step, the hidden layer receives the current SMART attribute vector along with the previous hidden state, which represents the sequential health status of the disk up to that point. The result is then passed on to the next hidden layer. Furthermore, the results based on SMART attributes in hidden layers at different time instances are stored, and considering the trend of each attribute towards imminent failure, disk failure is predicted. Therefore, apart from the current input attributes, the prediction in this model depends on historical sequential information as well, and hidden layers act as internal memory. Hence, in this model, to evaluate disk health status, the remaining time is divided into different intervals, and the degree of disk health is defined based on the time before failure.

Generally, identifying faulty disks leads to improved system performance and enables services to be accessed through live migration of existing virtual machines and allocating new virtual machines to healthy disks. For example, in a study titled "Cloud Disk Error Forecasting (CDEF)", Xu et al. [42] proposed an approach in 2018 to predict disk errors with the goal of improving the availability of cloud service. Unlike many other approaches, their method combined traditional SMART attributes with system-level signals—aggregated indicators or diagnostic data collected from the entire system, rather than just individual hardware components. These signals often combine or reflect the overall operational status of the disk, capturing patterns of behavior that may indicate impending failure. The model is constructed using a cost-sensitive ranking model, which prioritizes the identification of faulty disks while minimizing the cost of misclassification. These costs are determined empirically and are calculated based on the conditions and criteria relevant to the system and services in question. Overall, the goal of determining these costs is to achieve a balanced trade-off between accurately detecting faulty disks and avoiding errors in identifying healthy disks.

Yang et al. [44] compared the performance of the CDEF approach with SVM and Random Forest approaches that solely utilize SMART attributes. The results showed that the proposed approach achieved a higher Area Under the Curve (AUC) for the True Positive Rate, with a value of around 0.93, which is greater than the other two methods. Furthermore, they also investigated the impact of feature selection on the proposed approach. They compared three feature selection methods: SMART-based features, SMART-based features combined with system-level signals, and their proposed method which combines both types of features. The research indicated that the combined feature selection method achieved better performance in terms of True Positive Rate (35.8%) compared to the traditional SMART-based method (27.6%) and the method using both SMART-based features and system-level signals (30.3%).

In another study, Santo et al. [9] introduced an approach for estimating remaining useful life based on LSTM on two datasets. The Baidu dataset, which reports data by the hour, and the Backblaze dataset, which reports daily from samples belonging to the Seagate STD4000000M disk. In this approach, initially in the preprocessing section, common features present in both datasets were selected as significant features. In total, twelve features were selected. Due to the imbalance in the number of healthy and unhealthy disks, they also generated some unhealthy disks using available methods. Based on the type of data, which is either hourly or daily, a regression tree with the raw feature of the Current Pending Sector Count was constructed and, based on the time-to-failure feature, the regression tree was partitioned. In the partitioning section, disk health levels were defined at various levels based on the time window used in the LSTM model, categorizing them into good, fair, warning, danger, and failure. Evaluation results show that the presented model on the Baidu samples with a 48-hour time window achieved 99.8% accuracy, and on the Backblaze dataset with a 14-day window, it achieved 98.45% accuracy. In this research, in addition to accuracy and precision metrics, failure detection rate and false alarm rate metrics were also calculated. For the Baidu dataset, they reached a failure detection rate of 98.2% and a false alarm rate of 0.2%, while for the Backblaze dataset, they reached a failure detection rate of 91.48% and a false alarm rate of 0.72%.

2.3 Remaining Useful Life Prediction

In the disk health-based approach, the system can raise alerts based on the current condition of the disk, but predictions are limited to recent data leading up to the present. In contrast, the remaining useful life-based approach allows for a broader perspective by analyzing a time window of past disk activity. This enables the model to predict potential failures several days in advance based on observed trends and changes in the data. This allows the technician to transfer the data to a healthy disk before disk failure, depending on the necessity of the data and the amount of time available before the failure occurs [20].

Continuing the review of various approaches, the remaining useful life-based method using Recurrent Neural Networks (RNNs) is explored in this section. In this approach, Chang et al. [41] attempted to compare their proposed model with two methods: Hidden Markov Model (HMM) and Conditional Random Fields (CRF). Evaluation results indicate that HMM and CRF are effective for short-term prediction tasks, while the capability and effectiveness of the RNN in predicting long-term dependent tasks were greater. Moreover, the average remaining useful time before failure in the RNN was at least 208.6 hours and at most 494.4 hours.

As most studies in the offline domain employed decision trees for their proposed models, in another study by Li et al. [19], a decision tree was used to investigate the remaining time before actual failure. In this model, statistical measures such as inverse cumulative distribution function and quantile function were utilized to differentiate between healthy and failed drivers, helping select suitable SMART features. For instance, 90% of healthy drivers lacked any predictive features, while only 60% of unsuccessful drives possessed some of these features. As a result, this approach introduces a clear distinction between healthy and failed drivers. Additionally, because there were fewer positive samples than negative ones, the decision tree model assigned greater weight to negative samples and used a fixed-size time window to help distinguish between them. For instance, if at least half of the samples within the window are positive, the disk is classified as faulty; otherwise, it is labeled as healthy. Li et al. [19] also used metrics like fault detection rate, false alarm rate, and the time period before failure for assessment purposes.

The evaluation results demonstrate that the proposed model exhibits better prediction accuracy and interpretability compared to other models. Moreover, the proposed model has predicted 93% of failures with a false alarm rate below 0.01% and a time period before failure. The reliability of the decision tree model is highlighted due to its utilization of multiple Hidden Markov Models, which is considered an advantage of the proposed approach. In general, the results of this study indicated that using a decision tree model can significantly enhance the reliability of storage systems.

In the research study of Austin et al. [8], the performance of LSTM and Bi-LSTM neural networks under the same conditions was compared, demonstrating that the Bi-LSTM model outperforms the LSTM model. Using the Backblaze dataset, data for the ST4000DM000 model was extracted for two scenarios: 60 and 120 days before drive failure. Decision tree and correlation coefficient matrix methods were employed to extract the most important features. The evaluation results showed that in the case of a 60-day window before failure, the Bi-LSTM model outperformed the LSTM model, achieving 96.4% accuracy for predicting failure up to 30 days in advance. However, in the second scenario, where the model was trained with data 60 days before failure and evaluated with data 120 days before failure, the Bi-LSTM model's accuracy dropped to 49.7%, indicating its inability to detect failures over a longer time span, which is one of the drawbacks of this study. Additionally, the study focused on a single hard drive model, lacking diversity in encountered data and not covering various types of hard drive models.

Following the examination of conducted research, in Table 2.1 and Table 2.2, the features of each studies from the perspective of advantages, disadvantages, the algorithms used, and evaluation criteria are compared. Items marked with ″ $-$ ″ in the tables indicate that the corresponding metric was not mentioned or calculated in the referenced study and is therefore not applicable.

Table 2.1: The advantages and disadvantages of the conducted studies on disk failure prediction

Year	Approach	Advantages	Disadvantages	Ref
2013	Fault detection	Using all SMART parameters with the help of mRMR* and FMMEA** methods. Requires less computational time compared to the SVM method.	Removing missing values	[38]
2016	Fault detection	Using the similarity vector to represent the disk's similarities over time	Suffering from model aging and incompatibility with dynamic patterns.	[28]
2017	Fault detection	Prediction of various disk models.	Lack of standardization of different SMART features and consequently not considering appropriate failure indicators.	[29]
2017	Fault detection	Failure prediction based on operational data	Reduction in model performance due to the use of the balancing method	[2]
2017	Failure detection and remaining useful life	Use of decision tree and GBRT for prediction and easier interpretability. Presenting the GBRT model to indicate the health status disks	Risk of overfitting due to deep trees and extensive feature engineering.	[19]
2019	Health status of the disk	Presenting a health ranking for disks and calculating the predicted failure duration based on the health status grade of the disk	Limitation of the dataset and lack of coverage for diverse data.	[47]
2019	Fault detection	Proper labeling of samples with latent faults such as hidden sector errors. Completing the missing data sets, Covering certain types of failures.	Imputing missing data cannot fully recover all samples, and the training phase uses only limited disk failures, excluding unpredictable cases.	[15]
2020	Fault detection	Reviewing algorithms in terms of efficiency, stability, and productivity	Not considering the sample selection method and its impact on efficiency.	[44]

Continued on next page

Year	Approach	Advantages	Disadvantages	Ref
2020	Health status of the disk	Presenting a new method for selecting majority disk models. Introducing a failure prediction model.	To create balance, it has removed the healthy samples. Thus, it does not have the capability to cover model diversity. Selecting 10% of target data in the test section.	[27]
2014	Fault detection	Presenting a fault detection approach using the Mahalanobis method.	Training data is selected based on experience, without a systematic or data-driven justification.	[36]
2016	Fault detection, remaining useful life, and disk health status	Considering the dependencies between different states of disks to measure various health statuses of disks.	The method uses a 7-day prediction window, which may not generalize well.	[40]
2018	Fault detection	Real-time prediction capability. Lesser memory requirements due to not storing data statically and preventing model aging	Only using two disk models due to limited data	[40]
2018	Health status of the disk	Creating a cost-sensitive ranking model to rank disk failures and minimize total cost. A new method for selecting stable features in system-level signals that change over time and environment.	Using limited data from one company, potentially not generalizable to other systems. Only using two metrics: true positive and false negative rate	[42]
2018	Predicting failure time	Using the BNFH*** method for predicting failure time. Comparison with RNN algorithms	Using a very limited dataset. Not reporting other evaluation metrics	[6]
2019	Remaining useful life	Considering the dynamic features of disk degradation	No comparison of the proposed approach with other approaches and with limited generated data	[37]

Continued on next page

Year	Approach	Advantages	Disadvantages	Ref
2020	Fault detection	Compatibility with the concept of drift. Predicting using classification (labeling as healthy or future failure) and regression (between the probability of future failure)	Using all available SMART attributes without filtering or ranking features. This may introduce noise and increase computational cost, as not all attributes are relevant or useful across all disk models.	[14]
2020	Predicting remaining useful life	Using the LSTM method. Shorter prediction time compared to SVM and random forest	Not updating the model	[17]
2021	Predicting remaining useful life	Using deep neural networks (Bi-LSTM) that have memory	Inability to adequately detect failures for the remaining time over long durations	[8]
2021	Predicting remaining useful life	Short training time with limited data	Misalignment with evaluation metrics of other studies	[7]
2022	Disk health status	Use of a deep neural network (LSTM) with memory. Modeling performed on two datasets, Backblaze and Baidu	Using only one disk model	[9]

*mRMR stands for Minimum Redundancy Maximum Relevance, a feature selection method that selects features with maximum relevance to the target and minimum redundancy among themselves.

**FMMEA stands for Functional Modeling-Based Failure Modes and Effects Analysis, which analyzes component-level failures and their impacts systematically.

***BNFH stand for Bayesian network based Method for Failure prediction in HDDs

Table 2.2: Comprehensive overview of disk failure prediction methods and their performance

Method	Algorithms	Dataset	PFT	Evaluated Criteria (%)			Ref.
				FDR	FAR	TPR	
CDEF	RG	Baidu	–	–	0.1	29.6, 41.6	[42]
CDEF	SVM	Baidu	–	–	0.1	18.8, 34.1	[42]
CDEF	RF	Baidu	–	–	0.1	7.2, 21.7	[42]
ORF	RF	SMART	–	93, 99	0.69	–	[40]
ORF	SVM	SMART	–	93, 99	0.69	–	[40]
ORF	DT	SMART	–	93, 99	0.69	–	[40]
–	–	Baidu	6 days	97.44	0.83	–	[37]
StreamDFP	–	SMART	9–13 days	–	–	–	[40]
Bi–LSTM	LSTM	BackBlaze	30 days	–	–	–	[8]
LSTM	–	BackBlaze	–	91.48, 0.72	–	–	[8]
Mahalanobis	SVM	SMART	24h	68	0	–	[36]
Mahalanobis	HMM	SMART	24h	68	0	–	[36]
LSTM	SVM, RF	BackBlaze	15h	–	1.3	–	[17]
Raoblack–wellized	SVM, RNN	Operational	149h	97.44, 0.83	–	–	[35]
Raoblack–wellized	MD	Operational	149h	97.44, 0.83	–	–	[35]

PFT*: Predicted Failure Time
-*: Not Applicable

Existing research in this area has explored various methods, such as using LSTM, Convolutional Neural Network (CNN), and decision trees, to predict disk failures [22]. However, these methods often have problems keeping up with the current state, require a lot of computing power, and may not work well with different types of data. This research aims to overcome these challenges by using a combination of careful feature selection and a Bi-LSTM network, providing a more flexible and accurate solution for predicting disk failures.

This research identifies specific challenges and limitations of existing methods in fault detection and disk health classification that the proposed approach aims to overcome. These include reliance on datasets that may not fully capture the spectrum of disk failure scenarios, such as those reporting only daily, which could miss finer temporal resolution insights. It also addresses the issue of generalizing across different datasets and disk models, which existing studies often struggle with, potentially reducing a model's effectiveness across various storage solutions. Furthermore, the thesis critiques the lack of transparency and replicability in the feature selection processes of current methods and their adaptability to different reporting intervals, aiming to improve prediction accuracy and timeliness through comprehensive dataset analysis, a rigorous feature selection process, model generalization, and adaptation to reporting frequencies.

Chapter 3

Proposed Methodology

In this section, we elaborate on our proposed approach to address some of the challenges discussed in the previous chapter. The goal of this research is to improve the process of fault detection and disk health classification, aiming to increase accuracy, reduce false positive rates in fault detection, and enhance the lead time for predicting faults before they occur. These improvements help data centers manage hard disks more efficiently and support more effective stock maintenance and budgeting.

To address this problem, we developed a Bi-LSTM model. To explain the Bi-LSTM approach, it is first necessary to describe the LSTM method. LSTM is a powerful type of recurrent neural network designed to overcome sudden or gradual issues that occur when learning long-term dependencies. The LSTM network is composed of units of sub-networks that are recurrently connected, forming what are known as memory cells. The idea behind the memory cell is to maintain its current state over time and regulate the flow of information through nonlinear units. Figure 3.1 illustrates the architecture of an LSTM cell [33].

While previous methods often rely on static models that struggle to adapt to changing disk health patterns, our approach improves flexibility by combining Pearson's correlation coefficient and the Chi-square test for feature selection. This strategy helps identify the most relevant SMART attributes, enhancing the model's performance. The developed Bi-LSTM network is designed to adapt to evolving data patterns and predict failures earlier, with the potential to detect issues up to 40 days in advance. Additionally, our method is built to generalize across different disk models, offering a broader applicability to diverse datasets. Through these efforts, we aim to build on previous research and contribute to ongoing advancements in disk failure prediction.

As shown in Figure 3.1, the input to each LSTM cell at each time step includes the current input x_t , the previous hidden state vector h_{t-1}, and the previous state vector c_{t-1}. Furthermore, each LSTM cell consists of three gates named the input gate, the output gate, and the forget gate [13, 43].

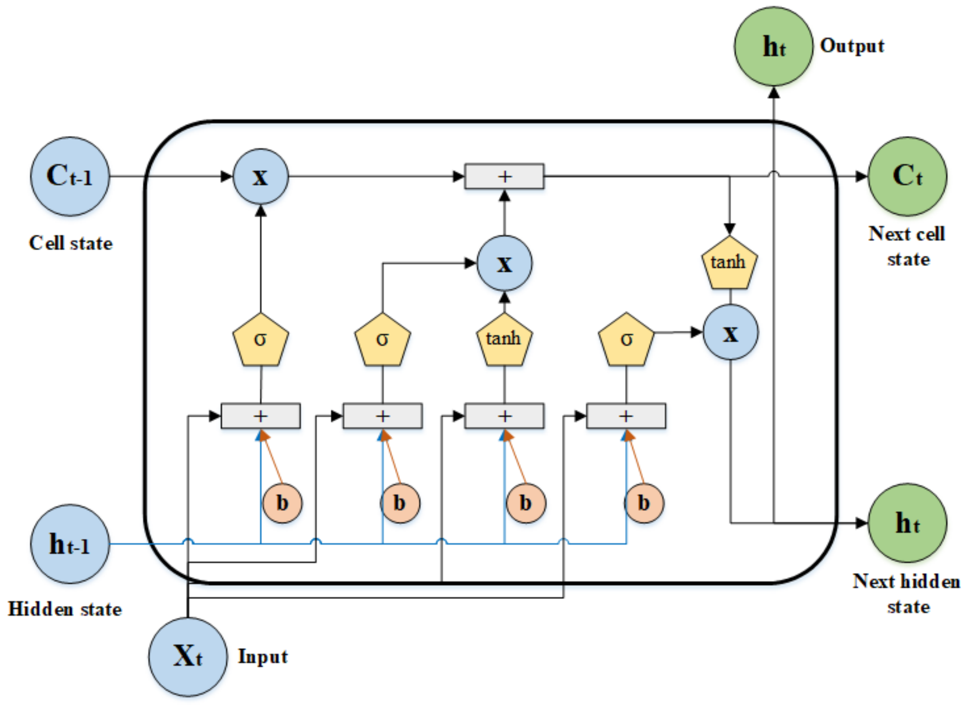

Figure 3.1: The architecture of an LSTM block

The LSTM (Long Short-Term Memory) cell is designed to process and store information over time in sequential data. The cell state (C_{t-1}) acts as long-term memory, carrying information from previous time steps, while the hidden state (h_{t-1}) stores short-term memory. Both states are passed along from one time step to the next. The input x_t is the data at the current time step, combined with the previous hidden state to influence updates.

The forget gate (σ) decides how much of the previous cell state should be retained or forgotten. It uses a sigmoid function to output a value between 0 and 1, where 0 means discard and 1 means retain. The forget gate's output multiplies with the previous cell state to "forget" unnecessary information. The input gate has two parts: a sigmoid function, which controls which values to update in the cell state, and a *tanh* function, which generates new candidate values to be added to the cell state. The updated cell state is a combination of the retained information and the new values.

The output gate decides what information should be passed as the hidden state for the next time step. It uses a sigmoid function to determine how much of the cell state contributes to the hidden state and applies *tanh* to the cell state to normalize it between -1 and 1. The final output, h_t, is the result of multiplying the output gate's value with the updated cell state. Throughout this process, bias terms (b)

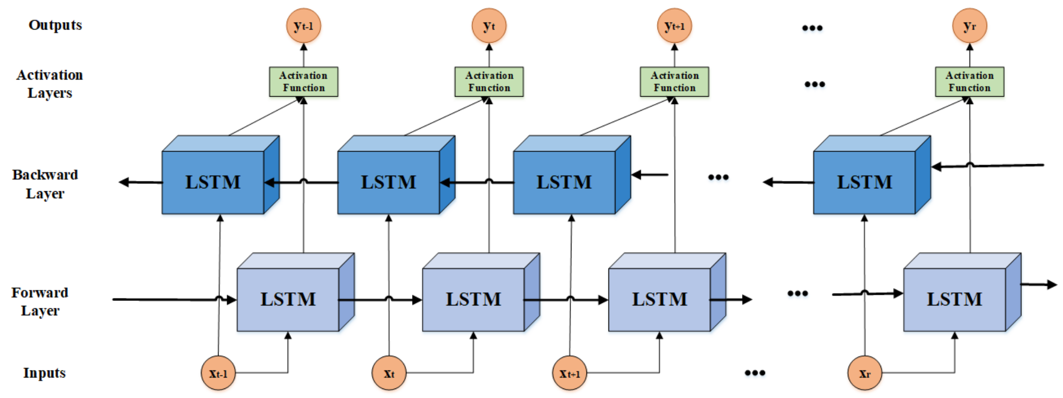

Figure 3.2: An example of Bi-LSTM network structure

are added to the gates, allowing for better learning. The LSTM structure, with its ability to selectively remember and forget information, makes it effective in tasks like predicting disk failure, where long-term dependencies are important.

Bidirectional Long-Short Term Memory (Bi-LSTM) [16], is a bidirectional recurrent deep neural network that consists of two separate recurrent networks in opposite directions, where each sequence of information is provided separately to each network. Bi-LSTM is typically used in situations where a sequential order of tasks is required. This type of network can be used in text classification models, speech recognition, and prediction [32, 46].

One of the differences that distinguishes Bi-LSTM from the regular LSTM is that in Bi-LSTM, information flows in two directions, forward or backward, whereas in LSTM, we can only have input flow in one direction, either backward or forward. In the forward layer of the Bi-LSTM, the sequence order is processed from the beginning to the end, in a way that each sequence receives and processes its next state. However, in the backward layer of the Bi-LSTM, the sequence order is from the end to the beginning, so that each sequence receives and processes its previous state. Ultimately, both layers are connected to an output layer. Figure 3.2 shows an example of the structure of the Bi-LSTM neural network.

The proposed Bi-LSTM tackles variations in disk failure patterns across different years and datasets by integrating advanced machine learning algorithms and thorough data analysis. By employing Bi-LSTM networks, the method effectively captures temporal dependencies, making it adaptable to evolving failure patterns. Rigorous data preprocessing, including feature normalization and selection, ensures that the model focuses on the most predictive indicators, enhancing its generalizability.

The flexible architecture of Bi-LSTM allows for easy adaptation and scaling, ensuring it can be applied to different data center environments. Additionally, continuous learning mechanisms enable the model to stay relevant and accurate over time, adapting to new failure patterns and technological changes. Overall, this approach provides a reliable, generalizable solution for predicting disk failures in diverse storage systems.

Deep neural networks have numerous hyperparameters, which have made machine learning systems very powerful. However, configuring these networks is challenging because there is no well-established theory on how to use these hyperparameters, and finding an appropriate model often requires experimenting with various configurations. In the following, we investigate the hyperparameters used in our proposed approach. The hyperparameter that we used are: Learning Rate, Batch Size, Number of Epochs, Dropout Rate, Early Stopping.

The learning rate controls how much the model adjusts its weights during training. A small learning rate helps prevent large, unstable updates, ensuring smooth convergence and reducing the risk of overfitting by preventing the model from becoming too sensitive to fluctuations or noise in the data. The batch size determines how many samples are processed before the weights are updated. A batch size that strikes a balance between efficiency and generalization allows the model to make more frequent updates, thus preventing it from memorizing specific data points and helping it generalize better. The model is trained for a set number of epochs, with early stopping to halt training when the validation loss stops improving, which prevents overfitting by ensuring that the model does not continue learning noise or irrelevant patterns after it has already learned the key features. A dropout rate is used, randomly disabling a percentage of neurons during training. This regularization technique forces the model to learn redundant representations and prevents it from relying too heavily on any single feature, reducing overfitting and helping the model generalize better to unseen data. Early stopping also helps prevent overfitting by monitoring the validation loss and halting training if the loss does not improve, ensuring that the model does not fit too closely to the training data. Finally, the LSTM and Bi-LSTM architecture processes data in both forward and backward directions. By capturing both short-term and long-term dependencies, the model avoids overfitting by learning complex patterns without becoming overly complex itself. Additionally, maintaining a balanced number of neurons helps prevent the model from becoming too large and prone to overfitting while still being capable of learning the necessary features. These hyperparameters collectively ensure that the model generalizes well, improving performance while avoiding overfitting.

One of the most important hyperparameters of Bi-LSTM is the dropout method [31]. Random dropout is a model regularization technique. During training, the neurons in a neural network can begin to memorize repeated or similar patterns from the training data, which can cause the model to overfit. To reduce this risk, the dropout technique is used—this involves randomly setting a portion of neuron weights to zero in each training round, helping the model to generalize better by preventing it from relying too heavily on specific features. This prevents neurons from focusing on a limited set of repetitive features and encourages them to learn from other features. Stochastic gradient descent is used to update neuron weights in the random dropout method, and each time, a subset of data is used to update the loss function. In the random dropout method, some of the neurons are randomly dropped from the model's learning process each time [21, 34].

The loss function measures the error made by the model during each training iteration. In other words, it reflects an inverse relationship with the model's accuracy: the better the prediction, the smaller the loss value, ideally approaching zero. Conversely, if the model's predictions are poor, the accuracy is low, and the loss value becomes larger. Loss functions are generally divided into two main categories: those used for classification tasks and those used for regression tasks. In regression, the model predicts continuous quantitative values, while in classification, it predicts labels or classes. Common examples of loss functions include mean squared error, mean absolute error, and cross-entropy. Cross-entropy is a commonly used loss function in classification tasks that measures how different the predicted probability distribution is from the true distribution. It evaluates the performance of a model by quantifying how close its predicted probabilities (ranging from 0 to 1) are to the actual labels.

Mean Squared Error (MSE) is one of the most commonly used loss functions, which calculates the average of the squared differences between the predicted values and the actual values [12]. Equation 3.1 illustrates how the Mean Squared Error is calculated where n represents the total number of predictions (or the batch size in each training iteration). The variable y_i is the actual RUL value for the i-th sample, while $\hat{y}_i$ is the predicted RUL value for the i-th sample. These values are used to calculate the Mean Squared Error (MSE) by measuring the squared difference between the predicted and actual RUL values for each sample.:

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n} \tag{3.1}$$

Another loss function is Mean Absolute Error (MAE), which, like MSE, calculates the difference between the predicted and actual model values but uses the absolute value of the difference. Then, it takes the average across the entire dataset [11]. Equation 3.2 shows how Mean Absolute Error is calculated:

$$MAE = \frac{\sum|y_i - \hat{y}_i|}{n} \tag{3.2}$$

Since MSE uses the squared errors, if there are outliers in the data and the error is larger than 1, MSE will significantly magnify the error. Therefore, it appears that outliers have a greater impact on MSE compared to MAE. Thus, using MAE can improve the model's performance.

Cross-entropy is a loss function commonly used in classification tasks to measure the dissimilarity between the predicted probability distribution and the true distribution. It quantifies the performance of a model whose output is a probability value between 0 and 1. Mathematically, for a binary classification problem, cross-entropy is defined as:

$$\text{Cross-Entropy Loss} = -\left[y\log(\hat{y}) + (1-y)\log(1-\hat{y})\right]$$

where y is the true label and $\hat{y}$ is the predicted probability.

Activation functions are essential components in artificial neural networks (ANNs), that determine the output of individual neurons and introduce non-linearity into the model. This non-linearity enables the network to learn complex patterns and relationships in the data beyond simple linear transformations. In the Bi-LSTM neural network, each neuron in each layer receives input values, multiplies them by corresponding neuron weights, adds biases, and passes the result through an activation function, which acts like a gate. This process repeats until reaching the final layer. Activation functions can take various forms, including the sigmoid function, hyperbolic tangent function, and rectified linear unit (ReLU) function.

Sigmoid Function is a mathematical function that takes any real number as input and produces an output between 0 and 1. A larger output means the output is closer to the input, and a smaller output means it is closer to zero [10, 45]. The sigmoid function compresses input values into a range between 0 and 1 and is used in binary classification tasks. It is defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

where x is the input to the neuron. This input x is typically computed as a weighted sum of input features plus a bias term:

$$x = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

In this equation, w_i represents the weight associated with input feature x_i, and b is the bias. The sigmoid function maps this input x to a probability-like output in the range $[0, 1]$, making it suitable for binary decision boundaries. Despite its advantages, the sigmoid function can lead to the vanishing gradient problem, particularly when input values are very large or very small, which can slow down model training.

Hyperbolic Tangent Function is similar to the sigmoid function but produces outputs in the range of -1 to 1. A larger input makes the output closer to 1, while a smaller input makes it closer to -1 [39]. The *tanh* function maps input values to the range $[-1, 1]$, offering zero-centered output:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Compared to the sigmoid, *tanh* often provides faster convergence in training but still suffers from vanishing gradients at extreme values.

Rectified linear unit (ReLU) function is an activation function that is highly popular in the field of deep learning. It is computationally efficient and allows networks to converge quickly due to its linear relationship. The ReLU function outputs the input directly if it is positive; otherwise, it outputs zero:

$$\text{ReLU}(x) = \max(0, x)$$

This function is widely used in deep neural networks due to its computational simplicity and effectiveness in mitigating the vanishing gradient problem. However, some neurons may "die" if they always receive negative inputs during training. It is less computationally intensive compared to sigmoid and hyperbolic tangent functions [1]. This function operates in such a way that it sets negative values to zero and keeps positive values (including zero) unchanged.

Among the existing studies, we considered the research conducted by Coursey et al. [8] for fault detection and lead time improvement and the study of Santo et al. [9] for disk health classification as fundamental papers. The reasons for selecting these two studies as foundational work include the similarity of the approach used in the foundational work with our proposed solution, the ability to use similar data for training and evaluating the proposed method, enabling better comparison, and achieving better results compared to other existing research.

In general, to develop a disk failure detection and prediction method, we need to extract and process a suitable dataset containing both healthy and unhealthy disks. This dataset should be sufficiently large to ensure higher accuracy and reliability in detection and prediction. we can generalize it for identifying various types of models and new unknown failures.

Figure 3.3 illustrates the overview of the work flow for our approach. Our proposed method consists of three phases. Feature extraction in the first phase involves selecting an effective set of features from healthy and unhealthy disks for training the fault detection model. Then, in the second phase, which is the model creation and evaluation phase, we aim to train the model for remaining useful life prediction up to disk failure. Using the final set of features, the processed dataset (obtained in phase one), and determining the model and optimization parameters, the model is trained, and the detection accuracy is evaluated using the time steps. If the model does not reach a sufficient accuracy level, it is retrained with new feature selections and adjustments to the time step,the detail explanation is in Section 3.1. In the next step, known as the health degree classification phase, we begin by reviewing the features selected in phase one and setting threshold levels them. The disks are then categorized into different health levels, and the trained model is evaluated more details will be discussed in Section 3.2.

For feature selection, we apply the Pearson correlation coefficient and the Chi-square test to identify the most predictive Self-Monitoring, Analysis, and Reporting Technology (SMART) attributes. Data preprocessing is handled through Z-score

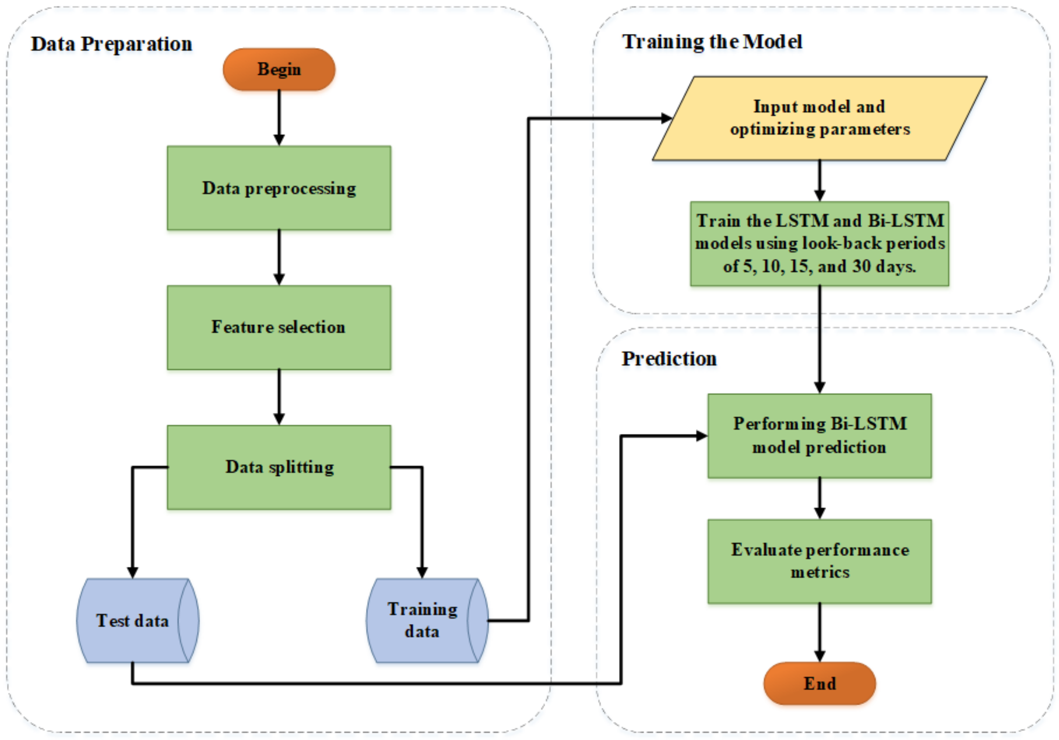
Data Preparation
Begin
Data preprocessing
Feature selection
Data splitting
Test data
Training data
Training the Model
Input model and optimizing parameters
Train the LSTM and Bi-LSTM models using look-back periods of 5, 10, 15, and 30 days.
Prediction
Performing Bi-LSTM model prediction
Evaluate performance metrics
End

Figure 3.3: Overview of the work flow for our approach

and Min-Max normalization to standardize feature values, along with mean imputation to deal with missing data, improving the quality of the input. The core of the model is a Bidirectional Long Short-Term Memory (Bi-LSTM) deep learning network, which captures temporal patterns in time-series data and adapts to changing disk failure behaviors.

To optimize performance, we carefully adjust the hyperparameters, such as the number of epochs, batch size, time step for prediction, learning rate, dropout probability, and the number of hidden layers. The effectiveness of the model is evaluated using a variety of metrics, including precision, recall, F-score and mean absolute error (MAE), providing a complete assessment of its predictive capabilities.

As seen in Figure 3.3, the first phase consists of three steps: data collection, data processing, and feature extraction. The detailed pseudo code of the proposed methodology is presented in Algorithm 1.

In data collection step, we collect data from healthy and unhealthy disks. Currently, based on the previous work in the field of disk failure detection and prediction, we have used data from two sources: Backblaze and Baidu. Backblaze is a data storage provider for businesses and end-users worldwide, operating a data

Algorithm 1 Proposed Disk Failure Prediction Method

```
 1: Input: Raw SMART attributes from Backblaze and Baidu datasets
 2: Output: Predicted Remaining Useful Life (RUL) and Disk Health State

 3: procedure PHASE 1: FEATURE EXTRACTION
 4:     Collect disk data from 120 days before failure
 5:     Assign RUL values based on days before failure
 6:     Normalize data using Z-score and Min-Max methods
 7:     Handle missing values using mean imputation
 8:     Apply Pearson correlation to assess correlation with RUL
 9:     Apply Chi-square test for feature relevance
10:     Remove redundant features with high inter-feature correlation
11:     Select final feature set with high predictive power
12: end procedure

13: procedure PHASE 2: MODEL TRAINING AND EVALUATION
14:     Split data into training and testing sets
15:     Initialize Bi-LSTM model with tuned hyperparameters:
16:         Epochs, Batch Size, Learning Rate, Dropout, Time Step, Hidden Layers
17:     Train Bi-LSTM model on selected features
18:     Evaluate using Accuracy, Precision, Recall, F1-Score, MAE, and MCC
19:     if performance is unsatisfactory then
20:         Adjust time step or feature set and retrain
21:     end if
22: end procedure

23: procedure PHASE 3: HEALTH STATE CLASSIFICATION
24:     Reuse selected features from Phase 1
25:     Define thresholds for health categories
26:     Train decision tree classifier (Healthy, Warning, and Critical)
27:     Evaluate classification performance
28: end procedure
```

Table 3.1: The number of data used in each disk model

Disk Model	Number of data
Toshiba	1,172
Hitachi	8,859
Seagate	24,771
Western	289
Total	35,091

center that includes disks of different ages, capacities, manufacturers, and models. They have been collecting data from their data center disks regularly since 2013 [30]. Due to the generality of this dataset and its widespread use in various research, we utilized Backblaze dataset for our research in the years 2019 and 2020 [3].

In the proposed method, data from disks of four manufacturers, Toshiba, Hitachi, Seagate, and Western, has been used for 120 days before failure. Table 3.1 provides a list of disk models used along with the number of data samples utilized in the proposed method.

To create the RUL (Remaining Useful Life) column in the dataset, the process begins by identifying hard drives of each model for example model ST4000DM000 that have failed. Once a failed drive is found, the previous 60 to 120 days of SMART data for that same serial number are collected. This time-series data is stored in a long format, where each row represents a single day's measurements leading up to the failure.

Each row is then assigned a corresponding RUL value. The labeling starts from the day before the failure, which gets an RUL of 1, the day before that gets an RUL of 2, and so on—counting backward until the earliest day of available data. Instead of asking the model to simply classify whether a disk will fail on a specific day (yes or no), this approach trains it to predict a continuous value that reflects the number of days remaining until failure. As a result, the model provides more informative outputs—such as whether a drive is nearing failure or still has several weeks of safe operation left.

the next step is data preprocessing, in the dataset obtained from the previous step, the numerical ranges of disk features vary significantly due to differences in measurement scales. To enable more effective feature comparison, we apply two normalization methods: Z-score normalization and Min-Max normalization [44], which scales the values to the $[0, 1]$ range. In addition, disks may occasionally fail to report data due to sensor malfunctions or periods of inactivity, leading to missing values in the dataset. To address this issue, we use mean imputation, replacing missing values with the average value of each respective feature.

the third step is called feature extraction. The feature selection process in this thesis was designed to refine the dataset and identify the most relevant SMART attributes for accurate disk failure prediction. Given that not all SMART attributes contribute equally to predicting failures, redundant, irrelevant, or noisy features were eliminated to enhance model efficiency and accuracy.

Before applying feature selection techniques, the raw data were preprocessed. Z-score and Min-Max normalization methods were applied to standardize the numerical ranges of the features, ensuring consistency across attributes and making the dataset suitable for further analysis. Additionally, missing values were addressed using mean imputation, which replaced missing entries with the average value of the respective feature.

For the feature selection itself, two statistical methods were employed: Pearson's Correlation and Chi-squared test, which are discussed as follows.

Pearson's Correlation Coefficient method is used to assess the linear relationship between each SMART attribute and the remaining useful life (RUL) of the hard drives. Features with a high absolute correlation to RUL are prioritized, as they are deemed most predictive for failure. The Pearson correlation coefficient is calculated as:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{3.3}$$

where X represents a SMART attribute and X_i represents the value of variable X at the i-th observation in the dataset and symbol $\bar{X}$ denotes the mean (average) of all values of X in the dataset. Y represents the RUL and Y_i is the value of variable Y at the i-th observation and term $\bar{Y}$ stands for the mean of all values of Y. Also, r measures the degree of correlation between them. Attributes with a high absolute correlation (positive or negative) with RUL were considered strong candidates for inclusion in the prediction model, as they likely carry significant information about the degradation patterns leading to disk failure.

In addition to assessing correlation with RUL, we examine inter-feature correlations among SMART attributes to avoid redundancy and multicollinearity. Highly correlated features (e.g., $r > 0.85$) are carefully analyzed, and in cases where two attributes demonstrated strong interrelationship but similar predictive power, one of the features is removed to simplify the model and improve generalization.

To account for potential non-linear relationships, we apply the Chi-square test. The Chi-square test is one of the fundamental techniques in machine learning and deep learning for determining the relationship between multiple features. This test evaluates the independence of each feature with respect to the target variable (RUL), helping to identify statistically significant attributes for inclusion in the model. The Chi-square statistic is calculated as:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{3.4}$$

In the context of the Chi-square test applied to disk failure prediction, the terms O_i (observed frequency) and E_i (expected frequency) are used to assess whether a SMART feature is statistically associated with the remaining useful life (RUL) of a disk and k is simply the total number of groups or categories we are analyzing in the Chi-square test. The observed frequency O_i refers to the actual number of disk instances in the dataset that fall into specific combinations of feature values and RUL categories. For example, suppose we categorize the "reallocated sectors (RS)" feature into two groups: high (≥ 100) and low (< 100). If we find that 50 disks with high reallocated sector counts are actually close to failure (e.g., RUL $\leq$ 30 days), this count (50) is used as an observed value O_i. Other observed values are similarly collected for combinations like low RS with healthy disks (RUL $>$ 30 days), high RS with healthy disks, and so on.

On the other hand, expected frequencies E_i represent the counts we would expect to observe in each category if there were no relationship between the feature and the failure state—that is, if the distribution were completely random. These values are calculated based on proportions. Specifically, for each combination of feature level and target class (e.g., high RS and near failure), we multiply the total number of instances in the feature group by the total number in the RUL group, and then divide by the overall dataset size. For instance, if there are 70 disks with high RS and 60 disks in the near-failure class, and the total number of disks is 160, then the expected number of high-RS, near-failure disks is $(70 \times 60)/160 = 26.25$. Comparing the observed and expected frequencies across all such combinations allows the Chi-square statistic to quantify how strongly a feature is associated with disk failure. A larger Chi-square value indicates a greater deviation from the expected (random) distribution, suggesting that the feature may be a meaningful predictor of RUL.

In addition to analyzing individual attribute relationships with RUL, we examine the correlation between pairs of attributes. The values range from -1 to 1, where a positive value indicates a direct correlation, a negative value indicates an inverse correlation, and values closer to zero suggest little to no correlation. Attributes with an absolute correlation higher than a threshold T_C of 0.6 were grouped together, as this threshold generally marks the boundary between moderate and strong correlation. Based on statistical classification, correlations between 0.6 and 0.8 are considered strong, making a suitable boundary for grouping related attributes. If two attributes have an absolute correlation value greater than this threshold, they are placed in the same group, ensuring that all attributes within a group are closely related. If two highly correlated attributes provide similar information, only one is retained to reduce redundancy. This process helps improve the efficiency of the model while preserving its predictive power. The pseudo-code for the feature selection phase is presented in Algorithm 2.

The combined results of the feature selection methods are used to identify a final set of features with consistently strong predictive power. These selected

Algorithm 2 Phase 1: Feature Extraction

```
 1: procedure FeatureExtraction
 2:     Load disk dataset from Backblaze
 3:     Identify failed disks and extract data from 120 days before failure
 4:     for each disk record in the filtered data do
 5:         Assign Remaining Useful Life (RUL) = days before failure
 6:     end for
 7:     Impute missing values using mean imputation
 8:     Normalize features using Z-score and Min-Max normalization
 9:     for each feature in the dataset do
10:         Compute Pearson correlation with RUL
11:         Compute Chi-square statistic with discretized RUL categories
12:     end for
13:     Remove redundant features with high inter-feature correlation (e.g., correlation > 0.9)
14:     Select top predictive features based on Pearson and Chi-square scores
15:     return Final feature set and normalized dataset
16: end procedure
```

features are then used to train the Bidirectional Long Short-Term Memory (Bi-LSTM) model, ensuring that the model focused on the most relevant data while minimizing complexity. This feature selection process enhance both the efficiency and performance of the proposed failure prediction method.

3.1 Model Training

In the first phase, the dataset is preprocessed by removing ineffective features and identifying the most relevant ones. In the second phase, we develop a deep learning model and evaluate its performance in predicting the remaining useful life of disks, aiming for the highest possible accuracy up to the point of actual failure. Algorithm

As seen in Figure 3.3, the goal of this part is to build a model for detecting failures and predicting the remaining time until failure. Therefore, initially, we use the preprocessed dataset and the extracted features from the previous phase to construct the model in this phase. The model used in this research is of the type Bi-LSTM deep network. As mentioned before, the Bi-LSTM network consists of three layers: input, hidden, and output layers. The structure of this type of network is well-suited for handling sequential data due to its memory capabilities. After training the model, using the test dataset and an appropriate time step for predicting the remaining time until failure, we evaluate the results obtained from running the model. If acceptable accuracy is not achieved, we return to the model training section, adjusting hyperparameters of the network such as the number of layers, the number of neurons in each layer, the choice of input features, and the value of the dropout parameter. This process is repeated until we reach a trained model with suitable accuracy and a proper timeframe for failure prediction.

This network architecture consists of two parts: Bi-LSTM and LSTM. First, the model starts with an input layer, which takes in SMART data. Each data sample contains 15 time steps and 5 selected features. This data is fed into the network in a structured format with the shape (batch-size, time-steps, features), where the batch size is set to 32 to ensure efficient training.

The first LSTM layer has 128 neurons and is set to *return-sequences=True*, which means it passes the learned patterns to the next layers. This layer helps the model recognize short-term patterns in the data. Next, the Repeat Vector layer is used in the Bi-LSTM model to ensure that the output from the first LSTM layer, which captures information across multiple time steps, is appropriately processed by the subsequent layers. The Repeat Vector layer takes the output of the first LSTM layer and repeats it multiple times to match the number of time steps required by the next LSTM layer. This allows the model to maintain important learned information across time steps and ensures that the sequence is consistently processed throughout the entire network. By repeating the output, the model can better capture the relationships between past and present data, a crucial aspect of predicting disk failures based on time-series data from SMART attributes. This layer is used because it facilitates the flow of information between the layers and ensures that the model can learn complex patterns from both short-term and long-term dependencies, which improves the accuracy of predictions.

The Bi-LSTM layer, which is the core of this model, comes next. It consists of 128 neurons and processes the sequence in two directions—forward and backward. This allows the model to understand both past and future trends, leading to more accurate predictions. The ReLU activation function is applied here to help the model learn better without running into gradient issues. Another LSTM layer with 64 neurons follows the Bi-LSTM layer. This extra LSTM layer further enhances the model's ability to capture long-term dependencies in the dataset.

To prevent the model from overfitting to training data, dropout layers are included after the Bi-LSTM and LSTM layers. These dropout layers have a dropout rate of 0.35, meaning some neurons are randomly deactivated during training to help the model generalize better to new data. The dropout layers have an output shape of (0.0035), meaning they reduce the effect of certain neuron activations by scaling the outputs.

The output layer consists of a dense layer with a single neuron, which generates the final prediction for the remaining useful life of the hard drive. A linear activation function is used here, allowing the model to output continuous numerical values rather than categories. The linear activation function is the simplest type of activation function, defined by the formula $f(x) = x$ where x is the input to the neuron (i.e., the weighted sum of the inputs), and the output is directly proportional to this input, with no transformation applied. This contrasts with other types of activation functions (such as sigmoid or ReLU), which apply some form of non-linearity to the input. The output from this layer is then passed through a linear activation function, producing a continuous disk health score that quantitatively reflects the remaining useful life (RUL) of each disk. This score is typically ranges from 0 to 1, where 1 indicates a fully healthy disk and 0 suggests imminent failure. For example, a score of 0.85 might represent a disk in good condition, 0.45 could indicate a warning state, and a value below 0.2 would likely correspond to a critical or near-failure condition.

Although LSTM and Bi-LSTM algorithms are commonly used for classification tasks, they are also highly flexible and capable of handling regression problems. In our case, the ANN is designed to predict a continuous output rather than discrete labels because we aim to estimate the Remaining Useful Life (RUL) of a hard disk, which is inherently a regression problem. The goal is not just to classify a disk as 'healthy' or 'failed' but to provide a more informative, real-valued measure of how close the disk is to fail. By using a linear activation function in the output layer, the model produces continuous values that allow for more precise predictions, enabling early maintenance actions before a critical failure occurs.

To fine-tune the model, several hyperparameters were adjusted. The model was trained for 50 epochs, with an early stopping mechanism that monitored validation loss to stop training when no further improvement was observed. The Adam optimizer was used to adjust the model's weights efficiently, with an initial learning rate of 0.0001, which was adjusted dynamically during training. The loss function used was Mean Squared Error (MSE), which helps the model measure the difference between predicted and actual RUL values accurately.

The model was trained using two different optimizers: Stochastic Gradient Descent (SGD) and Adam optimizer. Both optimizers played an essential role in adjusting the model's parameters to minimize the loss function and improve predictive performance.

SGD is a widely used optimization method in deep learning. It updates the model's weights iteratively based on the gradient of the loss function. Unlike traditional gradient descent, which calculates the gradient over the entire dataset, SGD updates weights using a randomly selected small batch of data, making it more efficient for large datasets. For example, if the learning rate is set to 0.0001, each step updates the weights slightly in the direction that reduces error. However, SGD can be slow and may get stuck in local minima, so momentum was introduced to improve stability. The momentum term accumulates past gradients to provide a smoother and faster convergence. In this model, SGD with momentum (set to 0.9) is used, leading to more stable learning and reducing oscillations in weight updates.

The Stochastic Gradient Descent (SGD) algorithm follows these steps: first, it initializes the model weights randomly; second, it computes the gradient of the loss function with respect to each parameter; and third, it updates the weights using the following formula:

$$w_{t+1} = w_t - \eta \cdot \nabla L(w_t) \tag{3.5}$$

In this equation, w_t represents the current weights of the model and w_{t+1} is the updated weights after applying the gradient step. The term $\nabla L(w_t)$ is the gradient of the loss function $L(w)$, which indicates the direction and rate at which the loss increases most rapidly. To reduce the loss, we move in the opposite direction of this gradient. The parameter η, known as the learning rate, is a small positive constant that determines the step size taken in each iteration. In practical terms, this means that after each training iteration, the model evaluates how its current weights contribute to prediction errors and then updates those weights slightly in the direction that is expected to reduce future errors. A properly chosen learning rate ensures that the model converges efficiently toward an optimal solution. If the learning rate is too large, the model may overshoot the minimum and become unstable; if it is too small, convergence will be slow and may become stuck in a local minimum. For the SGD optimizer, the learning rate was set to $\eta = 0.0001$, and a momentum term $\gamma = 0.9$ was applied to smooth gradient updates and accelerate convergence. The model was trained for a maximum of 50 epochs with early stopping, and a batch size of 32 was used in both cases. These settings were chosen to balance training time and model performance while reducing the risk of overfitting.

The Adam optimizer is a more advanced optimization algorithm that combines the benefits of momentum-based SGD. It adapts the learning rate for each parameter dynamically, ensuring efficient convergence. Unlike SGD, which applies the same learning rate to all parameters, Adam adjusts it based on past gradients, making it particularly useful for deep networks like Bi-LSTM. It maintains two moving averages for each parameter: the first moment estimate (mean of past gradients) and the second moment estimate (uncentered variance). These help stabilize updates and prevent large fluctuations in learning. For example, if the learning rate starts at 0.0001, Adam automatically fine-tunes it to improve convergence speed and accuracy. In this model, Adam's parameters were set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$, allowing the optimizer to adapt well to different learning conditions and $\epsilon = 10^{-8}$ to prevent division by zero..

First moment estimate, m, is essentially the moving average of the gradients, similar to momentum in traditional gradient descent. Adam keeps track of this moving average to smooth out the gradient estimates and avoid abrupt changes in the direction of the gradient updates. The Adam algorithm works as follows: first, initialize first moment estimate m_0 and second moment estimate v_0 to zero then, compute the gradient g_t at time step t. the biased first moment estimate is given by:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.6}$$

In this equation, m_t is the first moment estimate at time step t, β_1 is the exponential decay rate for the first moment, and g_t is the gradient of the loss function with respect to the model parameters at time step t. The term m_{t-1} represents the first moment from the previous time step.

Then, update biased second moment estimate:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.7}$$

Here, v_t is the second moment estimate at time step t, and β_2 is the decay rate for the second moment. The squared gradient g_t^2 captures the magnitude of the current gradient's variance, while v_{t-1} is the second moment from the previous step.

Since both m_t and v_t are initially biased toward zero, Adam applies bias correction to account for their initialization:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.8}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.9}$$

These corrected estimates, $\hat{m}_t$ and $\hat{v}_t$, adjust the moments based on the decay rates raised to the power of the current time step t, ensuring stability and accuracy in early training stages.

Finally, the parameters are updated using the corrected first and second moment estimates. The update rule is as follows:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{3.10}$$

In this update rule, θ_t denotes the model parameters at time step t, and η is the learning rate. The denominator contains the square root of the second moment plus a small constant ϵ (typically 10^{-8}) to avoid division by zero. The result is an adaptive step size that adjusts for both the direction and magnitude of recent gradients, allowing Adam to efficiently converge even in noisy or sparse gradient environments.

For training, a sliding window approach was applied, where each input sample used data from the previous 15 time steps. This method allows the model to capture temporal dependencies by shifting the window forward step by step, creating multiple training samples from a single time-series dataset.

The sliding window works as follows: first, a window of 15 time steps is selected from the dataset, forming one input sample. Then, corresponding RUL value for the last time step in this window is used as the output label. In the third step, the window then shifts forward by one time step, creating a new sample with updated historical data and this process repeats until the entire dataset has been processed into overlapping input sequences. Using this approach, the model learns from multiple overlapping windows, ensuring it captures trends in the data without missing critical information. The overlapping sequences help the Bi-LSTM model generalize better, improving its ability to detect disk failure patterns in real-world applications. The model was trained using K-fold cross-validation, ensuring that it learned patterns effectively and avoided overfitting. The pseudo-code for Bi-LSTM model training and RUL prediction is presented in Algorithm 3.

3.2 Evaluation of Disk Health

In the second phase, the goal was to determine the remaining time until disk failure. However, if there are multiple types of failures with similar remaining times, it is not possible to determine the degree of disk failure based solely on the remaining time. Therefore, in the third phase, different levels of disk health status are defined, allowing us to specify the status of each disk based on its remaining time and the type of disk health status. This information can help technicians prioritize actions for different disks based on their health status.

According to the study [9, 41], disk health levels are divided into six categories: *Good*, *Very Fair*, *Fair*, *Soft Warning*, *Alert*, and *Red Alert*. In this phase, due to the use of data 60 days before failure and the limited time span, we only build our model based on three phases: healthy, Warning, and Critical. In this phase, we start by normalizing the dataset and using the features extracted in the previous phase.

First, we develop a decision tree-based method to evaluate the health condition of hard disk drives based on SMART attributes. The objective of this module is to classify HDDs into multiple discrete health levels—namely, Healthy, Warning, and Critical—in order to enable proactive maintenance strategies.

To achieve this, we implement a classification algorithm using a decision tree that learns optimal rules for distinguishing between different health categories. We train the decision tree using selected SMART attributes that demonstrate high correlation with the RUL, including selected SMART features. The decision tree starts with the root node, which represents the entire dataset of disk data. The first decision made is based on a feature, like Reallocated Sectors or Temperature. For example, the decision may be whether the Reallocated Sectors count exceeds a specific threshold. This initial split divides the data into two groups: disks with high reallocated sectors and disks with low reallocated sectors. The decision nodes are the internal nodes of the tree that represent decisions or tests made on the features. For instance, the next node may test "Sign up time" to further split the dataset into disks with fast sign up times and disks with slow sign up times. At each decision node, the data is divided based on the chosen feature's value, which helps create more homogeneous subsets of data.After several splits and decisions, the process reaches leaf nodes. Each leaf node represents a final classification or prediction. For example, a leaf node might predict that a disk with high Reallocated Sectors and slow Sign up Time is likely to fail within 30 days. Alternatively, it may predict that a disk with low Reallocated Sectors and fast Sign up Time is healthy and

Algorithm 3 Phase 2: Bi-LSTM Model Training and RUL Prediction

```
1:  procedure ModelTrainingAndEvaluation
2:      Split the preprocessed dataset into training and testing sets
3:      Initialize Bi-LSTM model with selected hyperparameters:
      epochs, batch size, learning rate, dropout rate
      time steps, number of hidden units
4:      Define model architecture:
      Input layer: shape [batch_size, time_steps, features]
      Bi-directional LSTM layer (e.g., 128 units)
      Additional LSTM layer (e.g., 64 units)
      Dense output layer with linear activation
5:      Compile the model:
      Loss Function: Mean Absolute Error (MAE)
      Optimizer options:
      ADAM — adaptive learning rate, $\beta_1 = 0.9$, $\beta_2 = 0.999$
      SGD — learning rate = 0.01, momentum = 0.9
6:      Apply $K$-fold cross-validation:
      Partition training data into $K$ folds
7:      for each fold $k$ do
8:          Train model on $K - 1$ folds, validate on fold $k$
9:          Record performance metrics (e.g., MAE, accuracy)
10:     end for
11:     Compute average metrics across all folds
12:     Finalize training on full training set (if performance is satisfactory)
13:     Evaluate on test set using:
      Accuracy, Precision, Recall, F1-score, MAE, MCC
14:     if model performance is unsatisfactory then
15:         Adjust time step, feature set, or optimizer configuration
16:         Retrain the model
17:     end if
18:     return Trained Bi-LSTM model
19: end procedure
```

will not fail soon. The decision tree follows the principles of the Classification and Regression Tree (CART) algorithm. At each node of the tree, the best feature and threshold are selected to split the dataset in a way that reduces class impurity. The quality of a split is evaluated using the *Gini impurity* criterion. The Gini impurity for a node containing a dataset D is defined as follows:

$$\text{Gini}(D) = 1 - \sum_{i=1}^{K} p_i^2 \tag{3.11}$$

Here, K denotes the number of classes, which in our case $K = 3$ corresponding to the classes Healthy, Warning, and Critical. The index i denotes the class label, and p_i represents the proportion of instances in D that belong to class i, computed as $p_i = n_i/n$, where n_i is the number of samples of class i and n is the total number of samples at that node.

For each feature, the algorithm considers all possible thresholds and evaluates the resulting splits. For a given feature A and a threshold t, the dataset is split into two subsets: D_{left}, containing all instances where $A \leq t$, and D_{right}, containing the remainder. The Gini impurity of the split is then calculated using Formula 3.11.

The feature and threshold pair that results in the lowest $textGini_{\text{split}}$ is selected to form the next node of the tree. This process is applied recursively until a stopping condition is met. The stopping criteria of our case is reaching a maximum tree depth or a minimum number of samples in a node. In our implementation, we set the maximum depth of the decision tree to four in order to maintain model interpretability and avoid overfitting. The tree was trained using the `DecisionTreeClassifier` class from the `scikit-learn` Python library. The input feature matrix X consisted of the three selected SMART attributes, while the target vector y contained the labeled health categories. The training process involved fitting the classifier using the following command:

```
clf = DecisionTreeClassifier(max_depth=4, random_state=42) clf.fit(X, y)
```

The developed decision tree produced a series of nested conditional rules that map SMART attribute thresholds to health labels. Table 3.2 represents an example of decision rules learned by the developed decision tree. For example, a portion of the learned rules is as follows:

```
if SMART_6 <= 0.35: if SMART_241 <= 0.15: → Predict: Healthy
else: if SMART_194 <= 0.75: → Predict: Warning else: → Predict: Healthy
else: if SMART_194 <= 0.35: → Predict: Critical else: → Predict: Critical
```

These rules form an interpretable model that can be applied in real-time systems to assess HDD health. A disk with a moderately high value of `SMART-6` and low temperature may be classified as *Critical*, while one with lower values of `SMART-6` and

Table 3.2: Example of decision rules learned by the decision tree classifier

Condition	**Prediction**
SMART_6 ≤ 0.35 & SMART_241 ≤ 0.15	Healthy
SMART_6 ≤ 0.35 & SMART_241 > 0.15 & SMART_194 ≤ 0.75	Warning
SMART_6 ≤ 0.35 & SMART_241 > 0.15 & SMART_194 > 0.75	Healthy
SMART_6 > 0.35 & SMART_194 ≤ 0.35	Critical
SMART_6 > 0.35 & SMART_194 > 0.35	Critical

`SMART-241` will likely be classified as *Healthy*. This level of explainability is critical for system administrators seeking to understand and act on model predictions.

The final model allows us to classify each disk instance into a health category with high accuracy. By integrating these classification results with the Bi-LSTM RUL predictions, we obtain a comprehensive model that combines temporal degradation modeling with rule-based status assessment, thereby improving both failure prediction and operational interpretability.

The process begins by selecting parameters for the decision tree regressor, including tree depth, number of divisions (splits), and minimum samples per leaf text (min_samples_leaf). For each SMART attribute, a DecisionTreeRegressor model is trained using that attribute as the sole predictor and the RUL as the target. The decision tree partitions the data into leaves based on the feature's value, learning thresholds that minimize prediction error. During this training, the Mean Absolute Error is calculated for each tree, which represents how closely the predicted RUL values align with the actual ones across the tree's leaves.

Once the labeling is complete, the categorized dataset is used to train the Bi-LSTM model that was constructed in the earlier phase. During training, the model processes sequences of SMART readings and learns to recognize temporal patterns that correspond to the different health classes. Finally, the trained Bi-LSTM model is evaluated using appropriate classification evaluation metrics, such as accuracy. These metrics provide insight into how well the model distinguishes between the health categories and how reliable its predictions are when applied to unseen test data. This evaluation step is crucial for validating the effectiveness of the proposed methodology in identifying disks at risk of failure and providing early warnings based on learned temporal patterns. The implementation of the disk health evaluation approach is presented in Algorithm 4.

Training on diverse datasets, like those from Backblaze, further broadens the model's applicability across different disk models and environments. The feature selection process, using Pearson correlation and Chi-squared tests, targets the most relevant features, minimizing reliance on dataset-specific attributes. Extensive evaluation and validation using metrics like accuracy and precision confirm the model's robustness across various settings. In the next section, the results of applying the proposed method will be presented and discussed.

Algorithm 4 Phase 3: Health State Classification Using Decision Tree

```
1: procedure HEALTHSTATECLASSIFICATION
2:     Reuse selected features from Phase 1
3:     Define disk health categories based on RUL thresholds:
       Healthy: $RUL > 60$ days
       Warning: $30 < RUL \leq 60$ days
       Critical: $RUL \leq 30$ days
4:     Encode health states numerically: 0 (Healthy), 1 (Warning), 2 (Critical)
5:     Split the data into training and test sets
6:     Initialize decision tree classifier with:
       Criterion: Gini impurity
       Max depth: 4–6 (tuned)
       Min samples per leaf: 10–20 (tuned)
7:     Train the classifier on the training data
8:     Predict health states on the test set
9:     Evaluate classification performance using:
       Accuracy, Precision, Recall, F1-Score
       Confusion Matrix to analyze class-wise performance
10:    if performance is unsatisfactory then
11:        Tune decision tree parameters (e.g., max depth, min samples)
12:        Retrain the classifier
13:    end if
14:    return Trained decision tree model
15: end procedure
```

The decision tree and the Bi-LSTM network serve distinct but complementary roles in the proposed methodology. The decision tree is not part of the final prediction model; rather, it is used as an evaluation tool to assess the predictive strength of individual SMART attributes. Each attribute is tested independently, and the Mean Absolute Error (MAE) is calculated to identify features that most effectively correlate with disk failure. These features—those with the lowest MAE—are then selected as inputs for the Bi-LSTM model. The Bi-LSTM processes these features as time-series sequences, learning temporal patterns to predict either the Remaining Useful Life (RUL) or the disk's health category. Thus, while the decision tree does not receive input from the ANN nor contribute to its output directly, it plays a critical role in refining the feature selection process to improve the ANN's learning performance.

The transition between Phase 2 and Phase 3 is not merely sequential but conceptually complementary. Phase 2 captures complex time-dependent degradation signals to produce fine-grained, real-valued RUL estimates. Phase 3 simplifies these insights by classifying the predictions into actionable categories, which are easier for engineers and automated systems to act upon. The reuse of selected features and the structure of thresholds derived from RUL predictions ensures a coherent, data-driven transition between the two phases. Together, these phases provide both precision (through regression) and interpretability (through classification), resulting in a robust and deployable disk failure prediction system.

Chapter 4

Experimental Results

4.1 Evaluation Indicators

As mentioned in the literature review section, most studies in disk failure prediction primarily considers Accuracy (as metric), False Positive Rate (FPR), True Positive Rate (TPR), and the time taken for prediction as the main evaluation metrics for evaluating the performance of the proposed methodology [9, 19, 29, 37, 44]. However, other evaluation metrics have also been reported in some papers but received less attention. In this section, we evaluate the performance of the proposed approach using several key metrics, including the Confusion Matrix, Accuracy, Precision, Recall, Matthews Correlation Coefficient (MCC), F-measure, and Mean Absolute Error (MAE). These metrics are used to validate the effectiveness of our methodology. In the following, we briefly describe each metric used to assess the model's performance.

A confusion matrix evaluates classification models by presenting four outcomes: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The True Positive Rate (TPR) (recall) measures the proportion of actual positives correctly identified and is calculated as:

$$TPR = \frac{TP}{TP + FN} \tag{4.1}$$

The False Positive Rate (FPR) quantifies the rate of misclassified negatives and is defined as:

$$FPR = \frac{FP}{FP + TN} \tag{4.2}$$

TPR emphasizes detection power, while FPR reflects specificity errors, both critical for evaluating model performance.

The accuracy metric shows the percentage of correct identifications of faults. It represents the overall performance of a fault detection method and a well-trained model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.3}$$

Precision metric measures the percentage of correctly identified faulty disks. It is also known as True Positive Rate (TPR).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.4}$$

Recall is the percentage of true faulty disks correctly detected by the method. It highlights the model's performance in identifying actual failures.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.5}$$

Matthews Correlation Coefficient (MCC) is a composite metric that combines precision and recall. It is particularly useful when dealing with imbalanced two-class classification problems. MCC values range from -1 to +1, with +1 indicating perfect detection, 0 indicating random detection, and -1 indicating complete mismatch.

$$\text{MCC} = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP) \times (FN + TN) \times (FP + TN) \times (TP + FN)}} \tag{4.6}$$

The F-measure is a combination of precision and recall, calculated using the formula provided below.

$$F\text{-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.7}$$

Mean Absolute Error (MAE) was discussed in detail in Chapter 3.

These metrics collectively provide a comprehensive assessment of the proposed method's performance in disk failure prediction.

The coefficient of determination (R^2) is a metric used to measure how well the model's predictions match the actual target values. The equation 4.8 computes how well the model predictions $\hat{y}_i$ match the actual outcomes y_i, relative to a baseline model that always predicts the mean $\bar{y}$. The closer R^2 is to 1, the better the model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.8}$$

4.2 Model Evaluation

In this section, we explain the evaluation process and conduct experiments using the proposed method. To implement and evaluate the proposed method, Python 3.8 is used within the Jupyter environment. As mentioned in the proposed method section, in the first phase, preprocessing operations were performed on the dataset. The Backblaze dataset is built upon Self-Monitoring, Analysis, and Reporting Technology (SMART), which is a built-in feature in most modern hard drives. SMART actively monitors and records various indicators of drive health over time. These include key metrics such as read and write error rates, the count of reallocated sectors, and other performance-related parameters that help assess the condition and reliability of the disk.

The primary goal of SMART attributes is to alert users about potential disk failures at least 24 hours before they occur, assuming there is enough time to perform a full backup. While this was accurate for drives from ten to fifteen years ago, which had gigabyte-level capacities, it is not necessarily true for recent hard drives with significantly larger capacities, as they require more time to create a full backup.

Disk manufacturers have defined SMART values to monitor the reliability of their hard drives using various sensors and metrics. All SMART-enabled hard drives maintain a collection of attributes referred to as "raw" and "normalized" attributes. Normalized attribute values are vendor-specific and are derived based on criteria set by the manufacturers. Due to the diversity in standard criteria set by each manufacturer, we exclude normalized attributes from our dataset and only utilize the raw SMART attributes for further analysis.

As shown in Table 4.1, there are 64 raw SMART features, and considering all of these features is not feasible. As mentioned earlier, removing ineffective features from a dataset with a large number of features reduces complexity and training time for the model. Therefore, selecting a subset of features that have the most significant impact on fault detection and prediction is crucial for improving the performance of the fault detection and prediction method.

Table 4.1: Hard drives SMART features in the BackBlaze dataset

No.	SMART Parameter	Name
1	SMART 1 raw	Read Error Rate
2	SMART 2 raw	Throughput Performance
3	SMART 3 raw	Sign Up Time
4	SMART 4 raw	Start/Stop Count
5	SMART 5 raw	Reallocated Sectors
6	SMART 6 raw	Read Error Rate
7	SMART 7 raw	Seek Time Performance
8	SMART 8 raw	Power-On Hours
9	SMART 9 raw	Sign-up Retries
10	SMART 10 raw	Calibration Retries

Continued on next page

No.	SMART Parameter	Name
11	SMART 11 raw	Power Cycle Count
12	SMART 12 raw	Soft Read Error Rate
13	SMART 13 raw	Vendor-specific field
14	SMART 15 raw	Vendor-specific field
15	SMART 16 raw	Vendor-specific field
16	SMART 17 raw	Vendor-specific field
17	SMART 18 raw	Vendor-specific field
18	SMART 22 raw	Current Helium Level
19	SMART 23 raw	Vendor-specific field
20	SMART 24 raw	Vendor-specific field
21	SMART 168 raw	Vendor-specific field
22	SMART 170 raw	Reserved Block Count
23	SMART 173 raw	Wear Level Count
24	SMART 174 raw	Unexpected Power Loss
25	SMART 177 raw	Wear Level Count
26	SMART 179 raw	Used Block Count
27	SMART 181 raw	Used Block Count
28	SMART 182 raw	Erase Fail Count
29	SMART 183 raw	SATA Downshifts
30	SMART 184 raw	End-to-End error
31	SMART 187 raw	Uncorrectable Errors
32	SMART 188 raw	Command Timeout
33	SMART 189 raw	High Fly Writes
34	SMART 190 raw	Airflow Temperature
35	SMART 191 raw	G-Sense Errors
36	SMART 192 raw	Power-Off Retract Cycles
37	SMART 193 raw	Load/Unload Cycles
38	SMART 194 raw	Temperature Celsius
39	SMART 195 raw	Hardware ECC Recovered
40	SMART 196 raw	Reallocated Events
41	SMART 197 raw	Current Pending Sectors
42	SMART 198 raw	Offline Uncorrectable
43	SMART 199 raw	CRC Error Count
44	SMART 200 raw	Multi-Zone Error Rate
45	SMART 201 raw	Soft Read Errors
46	SMART 218 raw	Vendor-specific field
47	SMART 220 raw	Disk Shift
48	SMART 222 raw	Loaded Hours
49	SMART 223 raw	Load/Unload Retries
50	SMART 224 raw	Load Friction

Continued on next page

No.	SMART Parameter	Name
51	SMART 225 raw	Load/Unload Cycles
52	SMART 226 raw	Load-in Time
53	SMART 231 raw	Temperature
54	SMART 232 raw	Available Reserved Space
55	SMART 233 raw	Media Wearout Indicator
56	SMART 235 raw	Good Block Count
57	SMART 240 raw	Head Flying Hours
58	SMART 241 raw	Total LBAs Written
59	SMART 242 raw	Total LBAs Read
60	SMART 250 raw	Read Error Retry Rate
61	SMART 251 raw	Min Spares Remaining
62	SMART 252 raw	Bad Flash Block
63	SMART 254 raw	Free Fall Protection
64	SMART 255 raw	Load-in Time

We calculated the Pearson correlation coefficient between each SMART attribute and the RUL of the hard drives. This coefficient quantifies the strength and direction of the linear relationship between two variables, ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation). We computed the correlation for each disk. Then, took the absolute values, and averaged them to derive a global correlation score per feature. These results are visualized in 4.1, where the blue bars represent correlation scores and the red dots represent feature importance determined by the developed Chi-square test.

From this analysis, we observed that SMART attributes 6, 9, 241, 242 and 194 had the strongest correlations with RUL. While SMART 7 also showed a strong relationship, we prioritized SMART 194 due to its consistent use in related studies, allowing fair benchmarking. To supplement and validate the findings from Pearson correlation, we employ a Chi-square test based on the same feature set. The feature importance scores assigned by the Chi-square test are plotted as red dots in the figure.

The convergence of results from these two independent techniques adds confidence in the robustness of the selected features for predictive modeling. These attributes were ultimately chosen for training the Bi-LSTM model due to their strong predictive signal and interpretability.

Additionally, we evaluated multicollinearity between the selected features using a correlation matrix, as shown in Figure 4.2. We used the Seaborn library to visualize the feature-to-feature correlations, and only feature pairs with a correlation greater than 0.8 were retained for inspection. Highly correlated features were reviewed for redundancy, and in cases where one of two highly correlated features provided redundant information, it was removed to reduce overfitting and improve generalization.

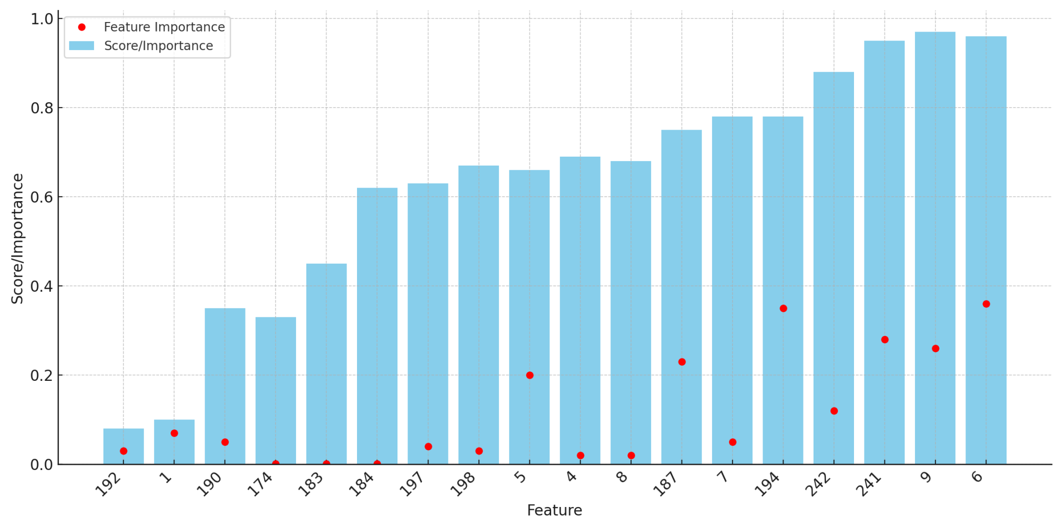

Figure 4.1: Correlation and Chi-square test of SMART features

As shown in Figure 4.2, SMART–7, SMART–6 and SMART–242, SMART-241, and SMART–9 exhibit the strongest correlations with each other. Since SMART 241 and 242 has less correlation together, we will remove SMART–9 from our selected features as well as SMART–7.

The second method for finding the most important SMART features is the Chi-square test. In this section, we apply the Chi-square test as a scoring function and use the *SelectKBest* class from the scikit-learn library to identify the top ten features of the dataset. The results are presented in Table 4.2.

Table 4.3 presents the combined set of features obtained from both the Pearson Correlation Coefficient and Chi-square test methods. The results show that SMART–194 and SMART–6 have overwhelmingly high Chi-square scores, indicating they are the most statistically significant features for distinguishing disk health states. The remaining features such as SMART–187, SMART–5, SMART–9 also show meaningful contributions but with significantly lower scores. This suggests that SMART–194 and SMART–6 and SMART–241 are likely to carry the strongest predictive power among the selected attributes.

As shown in Table 4.3, the common features are highlighted, and these are the selected features. Additionally, features SMART_7, SMART_242, SMART_1, SMART_5, and SMART_187, which are among the top features according to the Chi-square test method, have also been selected as final features, and they are also common with the features from previous studies [4, 8, 9].

In the second phase of the proposed method, to train the fault detection and prediction model, we utilize the Bi-LSTM, Repeat Vector layer, and the SGD optimizer.

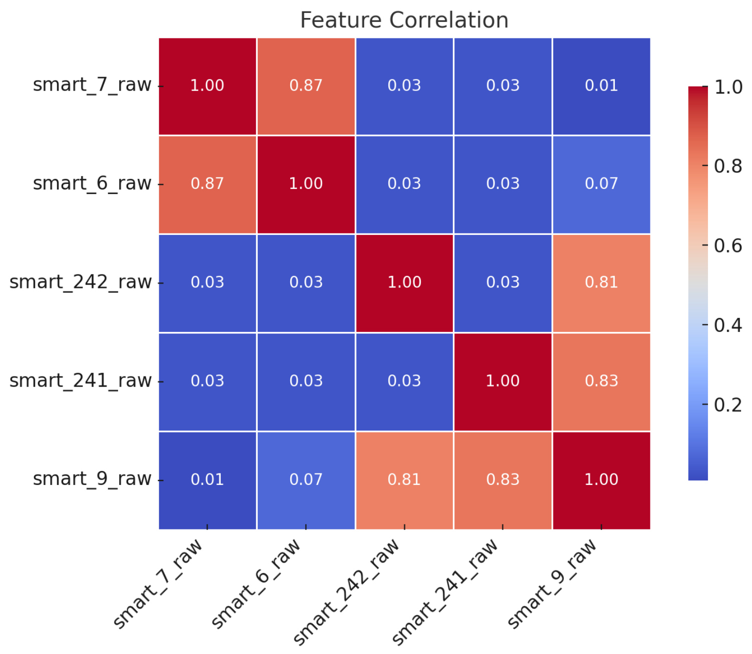

Figure 4.2: Features correlation matrix

This is in contrast to the baseline method, which used only a single Bi-LSTM layer and the Adam optimizer. To evaluate the proposed method and compare it with the baseline method, we train our model using both Adam and SGD optimizers with time steps of 5, 7, 10, and 15. To ensure fair comparisons, we use the same dataset and common hyperparameters with the baseline method.

In training the Bi-LSTM model, Adam initially provided faster convergence, helping the network adjust quickly to the data. However, after fine-tuning, SGD with momentum achieved the best performance when using a time step of 15, reaching an accuracy of 98.36%, with a recall rate of 99%, and a loss function value of 0.016. This shows that while Adam is efficient for early learning, SGD with momentum can provide better generalization and stability in long-term training.

The number of data points used for training and testing the model is provided in Table 4.4.

To synchronize the training of the model with appropriate performance, we manually review the hyperparameters and investigate them through trial and error. Table 4.5 lists the hyperparameters used in the base paper and the hyperparameters used for training the proposed model.

To evaluate the proposed model, we consider 80 percent of the data for training and used the hyperparameters mentioned in Table 4.5. The remaining 20 percent is used for model evaluation.

Table 4.2: Selection of the top ten features with Chi-square test

No.	Feature	Rank
1	SMART_194	3.73×10^{15}
2	SMART_6	3.80×10^{15}
3	SMART_8	0.04×10^{7}
4	SMART_5	2.07×10^{7}
5	SMART_241	2.93×10^{7}
6	SMART_9	2.71×10^{6}
7	SMART_7	1.03×10^{5}
8	SMART_187	2.28×10^{5}
9	SMART_1	0.83×10^{5}
10	SMART_242	1.48×10^{3}

Table 4.3: The set of features of Pearson correlation coefficient and Chi-square test

Method	Features
Pearson correlation coefficient	**SMART 194**, **SMART 6**, **SMART 241**, SMART 242
Chi-square test	**SMART 194**, **SMART 6**, SMART 8, SMART 5, SMART 242, SMART 9, **SMART 241**, SMART 187, SMART 1, SMART 7

4.3 Results and Comparison

4.3.1 Evaluating the Efficiency of the Trained Model

In this section, we present the fault detection model in each case, considering the optimizer used and various time intervals, and compare them with each other. In the training section, we use the aforementioned time intervals, and for the test section, we consider the remaining time to failure as 40 days. Table 4.6 shows the evaluation results of the base model using the Adam optimizer, and tables 4.7 and 4.8, respectively, show the evaluation results of the proposed model using the Adam and SGD optimizers in fault detection with 40 days prior to the actual failure.

Table 4.4: The number of training and test data of failure prediction model

Data type	Number of training data	Number of test data	Total
Healthy Data	27840	6961	34801
Unhealthy Data	232	58	290

Table 4.5: Comparison of hyperparameters for the base and proposed models

Evaluation method	Epochs	Batches	Time step	Learning rate	Dropout prob.	Hidden layers
Base model	50	64	5, 10, 15, 30	–	–	1
Proposed model	300	128	5, 7, 10, 15	0.0001	0.0035	5

Table 4.6: Evaluation results of the basic method with Adam's optimizer and 30 days before failure

Time step	Accuracy	R^2	MAE value
5	93.90%	0.999	0.131
10	93.40%	0.997	0.190
15	96.40%	0.998	0.120
30	96.00%	0.987	0.483

Table 4.7: Evaluation results of the diagnosis model with Adam's optimizer and 40 days before failure

Time step	Accuracy (%)	Precision (%)	Recall (%)	F-score	False neg. rate	Loss func.	MAE value	MCC value
5	89.25	80.00	85.16	63.32	0.560	0.410	0.720	0.58
7	94.70	91.52	96.40	93.50	0.300	0.230	0.076	0.72
10	93.20	91.00	94.80	92.70	0.019	0.170	0.050	0.86
15	95.70	96.80	97.00	96.10	0.014	0.021	0.038	0.94

Table 4.8: Evaluation results of diagnosis model with SGD optimizer and 40 days before failure

Time step	Accuracy (%)	Precision (%)	Recall (%)	F-score	False neg. rate	Loss func.	MAE value	MCC value
5	91.00	86.00	89.49	88.35	0.2800	0.320	1.3700	0.45
7	93.70	91.52	95.50	92.96	0.0200	0.210	0.1200	0.64
10	96.00	95.30	97.00	96.40	0.0180	0.130	0.0300	0.92
15	98.36	97.80	99.00	98.83	0.0014	0.016	0.0012	0.95

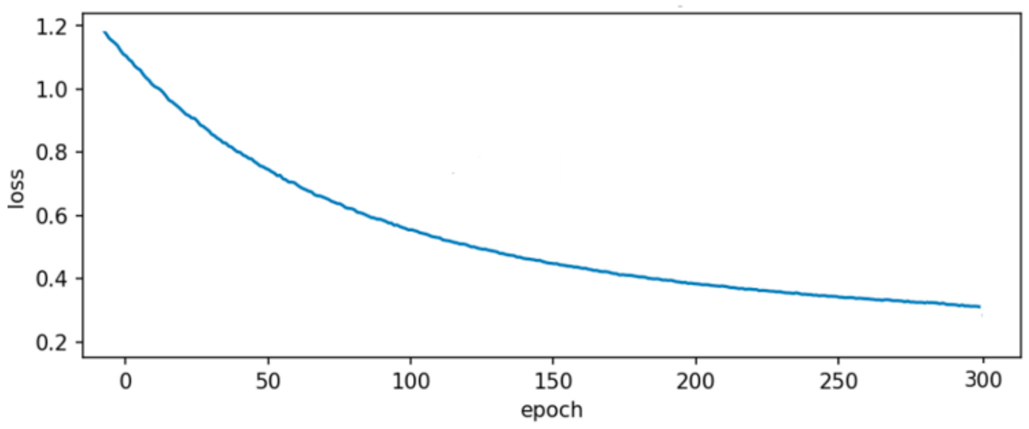

Figure 4.3: Loss function diagram with 5 time steps

As shown in the above tables, the performance of the proposed model using the SGD optimizer and with a time step of 15 has been better compared to the base method with various time steps. In the base model, the best performance at a time step of 15 and only 30 days before the failure was with an accuracy of 96.4%. In contrast, the proposed model with the SGD optimizer, with a similar time step of 15 and 40 days before the failure, has the best performance with an accuracy value of 98.36%, the lowest loss function with a value of 0.016, and a recall rate of 99%. These results indicate that the model, with the least error and with a 99% recall, correctly identifies the fault detection when the actual fault occurs. Therefore, the efficiency of the proposed model in correctly distinguishing between healthy and unhealthy disk samples has improved compared to the base method, and the time before failure in the proposed model has also performed better than the base model.

In Figures 4.3 to 4.6, the minimization of the loss function in each epoch in the proposed model are shown.

As shown in Figures 4.3 to 4.6; the value of the loss function in 4-time steps of 5, 7, 10, and 15 has gradually decreased with an equal number of epochs and reached a value less than 0.01. The effectiveness of the Bi-LSTM model was evaluated by analyzing the loss function trends across different time-step settings. These trends show how well the model minimizes prediction errors and adjusts to different time ranges.

For the 5-time-step setting, the loss quickly decreased from about 0.4 to 0.05 within the first 10 epochs and then stabilized. This sharp decline indicates that the model quickly learns short-term patterns. However, the early stabilization suggests that extending training beyond this point does not bring significant improvements, likely due to the limited time window.

Expanding the time step to 7 increased the initial loss to about 0.5, which then

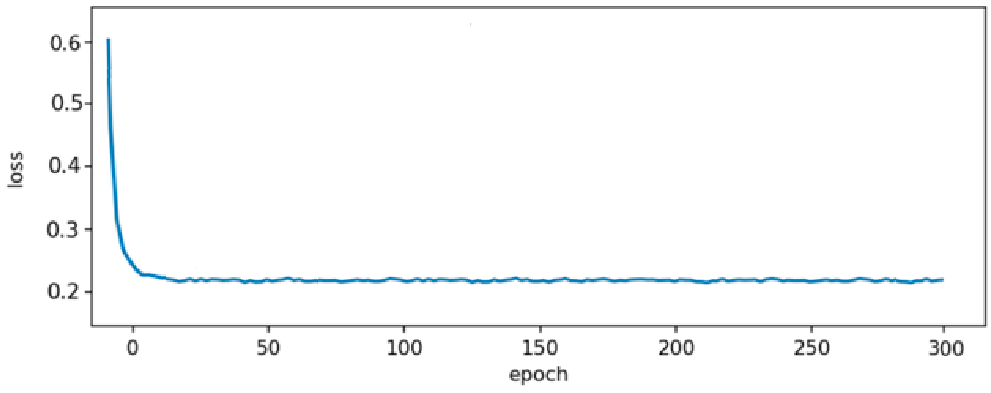

Figure 4.4: Loss function diagram with 7 time steps

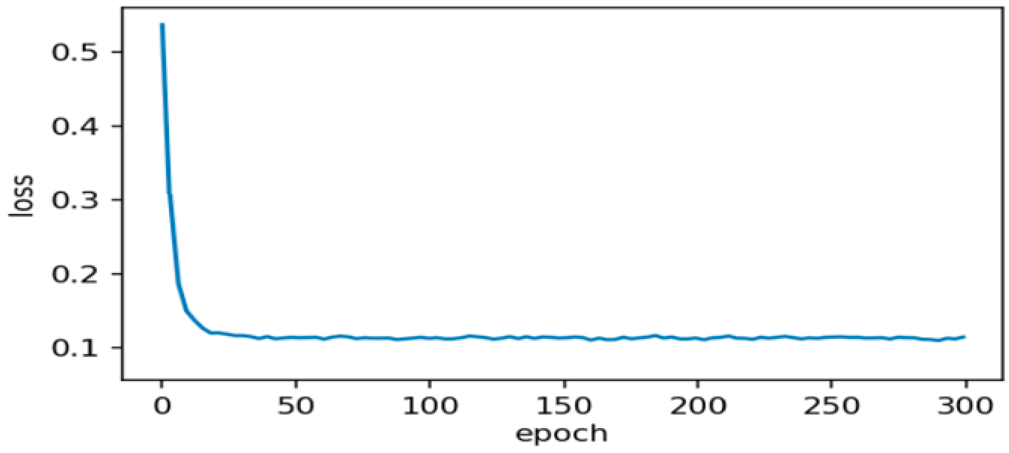

Figure 4.5: Loss function diagram with 10 time steps

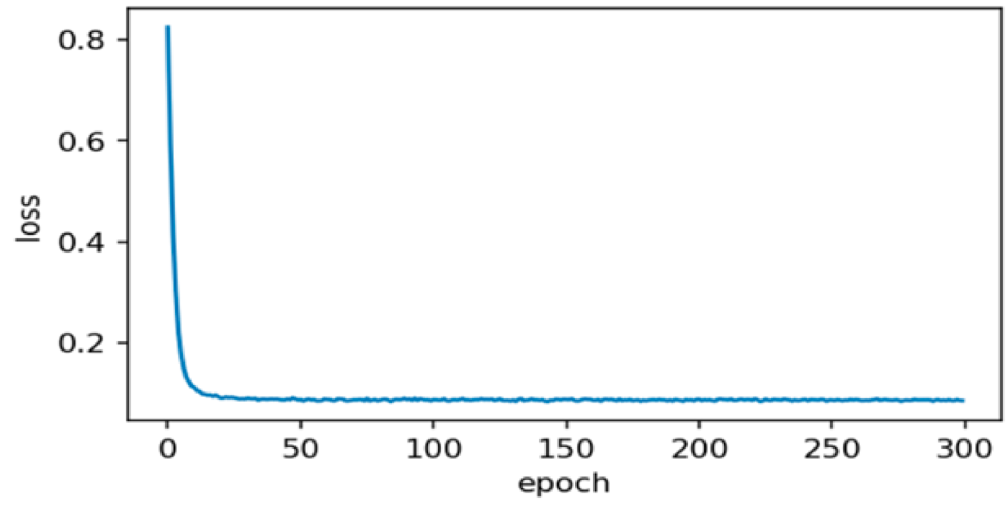

Figure 4.6: Loss function diagram with 15 time steps

gradually dropped to 0.076 by epoch 10 and stabilized around 0.02–0.03 after epoch 15. This shows that a wider time range helps the model learn better, but it also takes more epochs to fine-tune predictions and reduce residual errors.

For the 10-time-step configuration, the initial loss was lower, around 0.2, and it decreased steadily to 0.05 by epoch 15 before settling at about 0.03–0.04 after epoch 20. This longer convergence time suggests that handling medium-term dependencies requires more computational resources while maintaining efficiency.

The 15-time-step setup produced the best performance. The loss dropped from about 0.4 to 0.016 within 10 epochs and stabilized near 0.001–0.002. This quick convergence to almost zero error indicates the model's strong ability to recognize long-term patterns, aligning with its high accuracy of 98.36% for 40-day failure predictions. Additionally, stabilizing within 10 epochs makes it suitable for real-time use in dynamic storage environments.

All configurations showed stable loss within 10 to 20 epochs, demonstrating the Bi-LSTM model's ability to balance fast learning with reliability. Longer time steps resulted in lower final loss values, proving that the model effectively utilizes historical data to improve predictions. The observed trade-off between time-step length and convergence speed suggests that while longer time windows enhance accuracy, they do not necessarily compromise efficiency. The strong performance of the 15-time-step model confirms that extended temporal information can be effectively used without causing excessive computational overhead. Therefore, the results indicate that the trained neural network, with the increase of time steps and over time, has been properly trained and achieved better accuracy, and it has an

Table 4.9: MAE value of the selected features

No.	SMART number	MAE
1	Smart_6_raw	0.1489700
2	Smart_9_raw	0.1846040
3	Smart_8_raw	0.0615700
4	Smart_194_raw	0.0041856
5	Smart_242_raw	0.1090580
6	Smart_5_raw	0.0007600
7	Smart_241_raw	0.1167400

acceptable accuracy and loss function value. On the other hand, because quick training is important for practical use, the proposed model is designed to train efficiently, with an average training time of just 11 minutes.

4.3.2 Performance Evaluation for Disk Health Status

In this section, we evaluate the model's performance concerning the classification of the disk's condition. As mentioned in the third phase of the proposed workflow, we initially built a decision tree for each feature based on the MAE criterion and the number of samples. We then identified the feature with the lowest MAE value and the best decision tree. the objective was to determine which individual SMART attributes could most accurately estimate the number of days remaining until failure, referred to in the dataset as *day-num*.

To define *day-num*, the dataset was restructured such that each hard disk record was annotated with the number of days remaining before its observed failure. For failing hard drives, the final record prior to failure was labeled as *day-num* = 0, the record preceding it as *day-num* = 1, and so forth. This backward counting approach transformed each disk's timeline into a failure countdown, enabling the model to treat time-to-failure as a supervised regression problem. Non-failing drives were excluded from this labeling to focus the model's learning process on true degradation trajectories.

Each SMART feature was then individually evaluated using a univariate regression tree model. For this purpose, the $DecisionTreeRegressor$ from the scikit-learn library was configured with the $AbsoluteEerror$ criterion, which directly minimizes MAE during tree construction. The maximum depth of the tree was set to 4 to prevent overfitting, and the minimum number of samples per leaf was fixed at 5 to ensure sufficient representation in each split. The tree was trained on each SMART feature separately, with *day-num* serving as the target variable, and the resulting MAE was recorded as the feature's predictive performance score. Table 4.9 displays the obtained features along with the MAE value for each.

The results of this analysis are presented in Table 4.9, which lists the evaluated

SMART features along with their corresponding MAE values. The lowest MAE was observed for SMART 5 raw, with an error of 0.0007600, indicating it was the most informative individual feature for estimating the time to failure. SMART 194 raw and SMART 8 raw also performed well, yielding MAE values of 0.0041856 and 0.0615700, respectively. Conversely, SMART 9 raw and SMART 6 raw exhibited comparatively higher MAE scores, suggesting weaker relevance for this specific prediction task. In the next step, we first classify the `'num_day'` feature, which counts the number of days until the failure in the dataset, according to the four defined disk states. Then, based on the two obtained features, we feed the model trained in the second phase of the proposed method.

In order to further analyze the predictive power of individual SMART attributes, a decision tree was constructed using the `SMART 5 raw` feature, which had previously demonstrated the lowest MAE value (0.00076) during the feature evaluation phase. The purpose of this decision tree was to visualize the relationship between `SMART 5 raw` values and the estimated number of days remaining until failure (`day_num`). The decision tree was trained using the `DecisionTreeRegressor` model from the `scikit-learn` library, configured with the `absolute_error` criterion to minimize the mean absolute error directly. A maximum tree depth of four and a minimum of five samples per leaf were applied to maintain both interpretability and generalizability.

The resulting decision tree is presented in Figure 4.7. It reveals a set of threshold-based decision rules that map `SMART 5 raw` values to different predictions for the remaining days to failure. Each internal node in the tree represents a binary decision based on the value of `SMART 5`. For instance, the root node may split the dataset based on whether `SMART 5` is less than or equal to a particular threshold. Based on this decision, samples are passed either to the left or the right child node, where further conditions are recursively evaluated. This process continues until a terminal (leaf) node is reached.

Each leaf node provides a summary of the decision outcome for the samples that satisfy the conditions along the path leading to that node. Specifically, the node displays the number of samples it contains, the predicted average value of `day_num` for those samples, and the associated mean absolute error. The tree structure indicates how specific ranges of `SMART 5 raw` correspond to estimated remaining lifespans, revealing that even this single SMART attribute can yield informative and interpretable estimates of time to failure.

This decision tree thus confirms the critical role of `SMART 5 raw` as a highly informative predictor. Its strong performance supports its selection as one of the primary inputs to the Bi-LSTM model. Moreover, the tree visualization provides practical insight into how changes in SMART feature values translate into failure risk levels.

Since in the predictive phase, the model has forecasted a remaining useful life of 40 days until the actual failure, if there are 50 to 60 days left before the failure, it

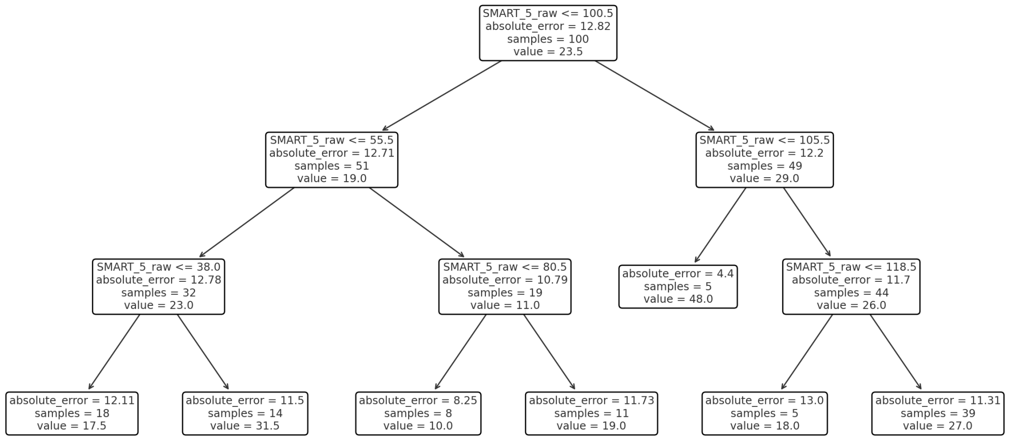

Figure 4.7: Decision tree for `SMART-5` trained to predict remaining days until failure

is considered a good condition. From 40 to 50 days is considered a warning status because based on the remaining useful life prediction until the failure obtained in the second phase of the proposed method, a warning should be given to the technician at most 40 days before the actual failure. If there are fewer than 20 days left until the failure, a red warning is issued to the technician.

In the study by Santo et al. [9], time intervals of 5, 7, 10, and 14 days are considered. Since we consider a 15-day interval in our proposed method, for comparing our results with the base paper, we only review the results of the 14- day interval from the base paper. Table 4.10 shows the evaluation results of the other state-of-the-art methods compared to the proposed method.

As shown in Table 4.10, in comparing our proposed method with existing approaches, we observe that the LSTM model of Santo et al. [9] achieved an accuracy of 98.45% with a precision and recall of over 98%, evaluated 45 days before failure. The Random Forest model by Aussel et al. [2] reported slightly lower accuracy at 96.4%, with a 60-day prediction window but did not specify precision and recall. The CNN-LSTM approach by Lu et al. [22] focused on precision and recall, achieving values of 93% and 94%, respectively, but did not report accuracy or specific time steps. Our proposed method outperforms these models, achieving a 99.27% accuracy, 98.65% precision, and 99.04% recall with a 60-day prediction window, demonstrating superior predictive capability.

The proposed method effectively addresses the dynamic nature of disk health and failure prediction by continuously updating its predictions with real-time data,

Table 4.10: Results evaluation of the other state-of-the-art methods and the proposed solution

Evaluation method	**Time before failure (days)**	**Time step (days)**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
LSTM [9]	45	14	98.45	98.33	98.34
Random Forest [2]	20	-	-	95.00	67.00
CNN-LSTM [22]	10	-	-	93.00	94.00
decision tree [19]	14	-	95.50	93.00	93.00
GBRTs [19]	14	-	87.20	90.00	87.00
Proposed approach	60	15	99.27	98.65	99.04

leveraging historical disk performance data to identify trends, and dynamically selecting relevant features. This approach ensures the model remains effective despite evolving workloads, hardware changes, and usage patterns. To mitigate challenges, the method involves continuous training to keep the model up-to-date, balancing early failure detection with minimizing false alarms, and employing techniques to handle data drift, ensuring sustained performance even as data characteristics change over time.

This study, like many others in the field, utilized the Backblaze dataset due to its accessibility, specifically focusing on data from 2019 and 2020. While this dataset provided a robust foundation, the feature set selected may be optimized for the specific disk models used in this study, limiting its generalizability to other models or datasets, such as those from Alibaba or Baidu.

Overall, our proposed method tackles variations in disk failure patterns across different years and datasets by integrating advanced machine learning algorithms and thorough data analysis. By employing Bi-LSTM networks, the method effectively captures temporal dependencies, making it adaptable to evolving failure patterns. Rigorous data preprocessing, including feature normalization and selection, ensures that the model focuses on the most predictive indicators, enhancing its generalizability.

Training on diverse datasets, like those from Backblaze and Baidu, further broadens the model's applicability across different disk models and environments. The feature selection process, using Pearson correlation and Chi-squared tests, targets the most relevant features, minimizing reliance on dataset-specific attributes. Extensive evaluation and validation using metrics like accuracy and precision confirm the model's robustness across various settings.

The method's flexible architecture allows for easy adaptation and scaling, ensuring it can be applied to different data center environments. Additionally, continuous learning mechanisms enable the model to stay relevant and accurate over time, adapting to new failure patterns and technological changes. All in all, this approach provides a reliable, generalizable solution for predicting disk failures in diverse storage systems.

Chapter 5

Conclusion and Future Research Directions

In this research, we addressed the critical challenge of predicting disk failures in large-scale data storage systems by leveraging deep learning and feature selection techniques. By applying Pearson correlation and Chi-square tests, we identified the most relevant features from the Backblaze and Baidu datasets, effectively reducing model complexity while enhancing performance. We developed a Bi-LSTM model that showed excellent predictive performance, achieving 98.36% accuracy and 97.8% precision, and was able to predict failures as early as 40 days in advance. Furthermore, we developed a decision tree-based health evaluation model to estimate the remaining useful life of hard disks and categorize them into Healthy, Warning, and Critical states. This model further improved prediction accuracy to 99.27% and precision to 98.65%. Overall, our approach provides a reliable and efficient solution for early disk failure detection and health assessment, contributing to improved data center reliability and reduced risk of data loss.

In the future, the proposed method can be further developed to better handle the evolving nature of data storage systems, including the growing use of edge computing. Additionally, the architecture can be improved to support continuous learning by regularly updating the model with new data. This would allow it to adapt to recent patterns while gradually forgetting outdated or less relevant information, helping maintain accuracy and performance over time. Such adaptability would make the system more scalable and practical for real-world applications where disk usage and behavior change frequently. Also, future work could implement a hard drive RUL prediction model into a real system, testing the performance of the model on real-time hard drive data to prevent them from failure. Additionally, future research could expand the dataset by including more disk models and data sources, and improve prediction granularity by adjusting the model to predict failures by the hour rather than the day. These enhancements will ensure the model remains robust, adaptable, and effective in various storage environments.

Bibliography

[1] A. F. Agarap, *Deep learning using rectified linear units (relu)*, arXiv preprint arXiv:1803.08375, (2018), pp. 1–7.

[2] N. Aussel, S. Jaulin, G. Gandon, Y. Petetin, E. Fazli, and S. Chabridon, *Predictive models of hard drive failures based on operational data*, in 2017 16th ieee international conference on machine learning and applications (icmla), IEEE, (2017), pp. 619–625.

[3] Backblaze.com, *Hard Drive Test Data*. Retrieved July 2025.

[4] S. Basak, S. Sengupta, S.-J. Wen, and A. Dubey, *Spatio-temporal ai inference engine for estimating hard disk reliability*, Pervasive and Mobile Computing, 70 (2021), p. 101283.

[5] M. M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann, *Predicting disk replacement towards reliable data centers*, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (2016), pp. 39–48.

[6] I. C. Chaves, M. R. P. De Paula, L. G. Leite, J. P. P. Gomes, and J. C. Machado, *Hard disk drive failure prediction method based on a bayesian network*, in 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, (2018), pp. 1–7.

[7] Z. Chen, M. Wu, R. Zhao, F. Guretno, R. Yan, and X. Li, *Machine remaining useful life prediction via an attention-based deep learning approach*, IEEE Transactions on Industrial Electronics, 68 (2020), pp. 2521–2531.

[8] A. Coursey, G. Nath, S. Prabhu, and S. Sengupta, *Remaining useful life estimation of hard disk drives using bidirectional lstm networks*, in 2021 IEEE International Conference on Big Data (Big Data), IEEE, 2021, pp. 4832–4841.

[9] A. De Santo, A. Galli, M. Gravina, V. Moscato, and G. Sperlì, *Deep learning for hdd health assessment: An application based on lstm*, IEEE Transactions on Computers, 71 (2020), pp. 69–80.

[10] J. Dombi and T. Jónás, *Generalizing the sigmoid function using continuous-valued logic*, Fuzzy Sets and Systems, 449 (2022), pp. 79–99.

[11] B. S. Everitt and A. Skrondal, *The Cambridge dictionary of statistics*, Cambridge University Press, 2010. p. 22.

[12] V. Ganesh and M. Kamarasan, *Parameter tuned bi-directional long short term memory based emotion with intensity sentiment classification model using twitter data*, in 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), IEEE, (2020), pp. 1–6.

[13] F. A. Gers, J. Schmidhuber, and F. Cummins, *Learning to forget: Continual prediction with lstm*, Neural computation, 12 (2000), pp. 2451–2471.

[14] S. Han, P. P. Lee, Z. Shen, C. He, Y. Liu, and T. Huang, *Toward adaptive disk failure prediction via stream mining*, in 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), IEEE, (2020), pp. 628–638.

[15] S. Han, J. Wu, E. Xu, C. He, P. P. Lee, Y. Qiang, Q. Zheng, T. Huang, Z. Huang, and R. Li, *Robust data preprocessing for machine-learning-based disk failure prediction in cloud production environments*, arXiv preprint arXiv:1912.09722, (2019), pp. 1–12.

[16] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation, 9 (1997), pp. 1735–1780.

[17] L. Hu, L. Han, Z. Xu, T. Jiang, and H. Qi, *A disk failure prediction method based on lstm network due to its individual specificity*, Procedia Computer Science, 176 (2020), pp. 791–799.

[18] A. Klein, *2020 Hard Drive Reliability Report by Make and Model*. Retrieved January 2025.

[19] J. Li, R. J. Stones, G. Wang, X. Liu, Z. Li, and M. Xu, *Hard drive failure prediction using decision trees*, Reliability Engineering & System Safety, 164 (2017), pp. 55–65.

[20] F. D. S. Lima, F. L. F. Pereira, L. G. Leite, J. P. P. Gomes, and J. C. Machado, *Remaining useful life estimation of hard disk drives based on deep neural networks*, in 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, (2018), pp. 1–7.

[21] Y. Liu, L. Wang, T. Shi, and J. Li, *Detection of spam reviews through a hierarchical attention architecture with n-gram cnn and bi-lstm*, Information Systems, 103 (2022), p. 101865.

[22] S. Lu, B. Luo, T. Patel, Y. Yao, D. Tiwari, and W. Shi, *Making disk failure predictions {SMARTer}*, in 18th USENIX Conference on File and Storage Technologies (FAST 20), (2020), pp. 151–167.

[23] A. Ma, R. Traylor, F. Douglis, M. Chamness, G. Lu, D. Sawyer, S. Chandra, and W. Hsu, *Raidshield: characterizing, monitoring, and proactively protecting against disk failures*, ACM Transactions on Storage (TOS), 11 (2015), pp. 1–28.

[24] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, *A large-scale study of flash memory failures in the field*, ACM SIGMETRICS Performance Evaluation Review, 43 (2015), pp. 177–190.

[25] J. F. Murray, G. F. Hughes, K. Kreutz-Delgado, and D. Schuurmans, *Machine learning methods for predicting failures in hard drives: A multiple-instance application.*, Journal of Machine Learning Research, 6 (2005), pp. 783–816.

[26] NTFS.com, *S.M.A.R.T. Attributes*. Retrieved January 2025.

[27] N. C. Oza and S. J. Russell, *Online bagging and boosting*, in International Workshop on Artificial Intelligence and Statistics, PMLR, (2001), pp. 229–236.

[28] L. P. Queiroz, F. C. M. Rodrigues, J. P. P. Gomes, F. T. Brito, I. C. Chaves, M. R. P. Paula, M. R. Salvador, and J. C. Machado, *A fault detection method for hard disk drives based on mixture of gaussians and nonparametric statistics*, IEEE Transactions on Industrial Informatics, 13 (2016), pp. 542–550.

[29] C. A. Rincón, J.-F. Pâris, R. Vilalta, A. M. Cheng, and D. D. Long, *Disk failure prediction in heterogeneous environments*, in 2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), IEEE, (2017), pp. 1–7.

[30] B. Schroeder, R. Lagisetty, and A. Merchant, *Flash reliability in the field: The expected and the unexpected*, in Usenix FAST, (2016), pp. 1–15.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, The journal of machine learning research, 15 (2014), pp. 1929–1958.

[32] Y. Verma, *Complete guide to bidirectional lstm (with python codes)*, Analytics India Magazine, (2021).

[33] G. Wang, Y. Wang, and X. Sun, *Multi-instance deep learning based on attention mechanism for failure prediction of unlabeled hard disk drives*, IEEE Transactions on Instrumentation and Measurement, 70 (2021), pp. 1–9.

[34] L. Wang, X. Xu, Q. Su, Y. Song, H. Wang, and M. Xie, *Automatic gear shift strategy for manual transmission of mine truck based on bi-lstm network*, Expert Systems with Applications, 209 (2022), p. 118197.

[35] Y. Wang, L. He, S. Jiang, and T. W. Chow, *Failure prediction of hard disk drives based on adaptive rao–blackwellized particle filter error tracking method*, IEEE Transactions on Industrial Informatics, 17 (2020), pp. 913–921.

[36] Y. Wang, S. Jiang, L. He, Y. Peng, and T. W. Chow, *Hard disk drives failure detection using a dynamic tracking method*, in 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), vol. 1, IEEE, 2019, pp. 1473–1477.

[37] Y. Wang, E. W. Ma, T. W. Chow, and K.-L. Tsui, *A two-step parametric method for failure prediction in hard disk drives*, IEEE Transactions on industrial informatics, 10 (2013), pp. 419–430.

[38] Y. Wang, Q. Miao, E. W. Ma, K.-L. Tsui, and M. G. Pecht, *Online anomaly detection for hard disk drives based on mahalanobis distance*, IEEE Transactions on Reliability, 62 (2013), pp. 136–145.

[39] E. W. Weisstein, *Hyperbolic functions*, 2003. Wolfram Research, Inc.

[40] J. Xiao, Z. Xiong, S. Wu, Y. Yi, H. Jin, and K. Hu, *Disk failure prediction in data centers via online learning*, in Proceedings of the 47th International Conference on Parallel Processing, (2018), pp. 1–10.

[41] C. Xu, G. Wang, X. Liu, D. Guo, and T.-Y. Liu, *Health status assessment and failure prediction for hard drives with recurrent neural networks*, IEEE Transactions on Computers, 65 (2016), pp. 3502–3508.

[42] Y. Xu, K. Sui, R. Yao, H. Zhang, Q. Lin, Y. Dang, P. Li, K. Jiang, W. Zhang, J.-G. Lou, et al., *Improving service availability of cloud systems by predicting disk error*, in 2018 USENIX Annual Technical Conference (USENIX ATC 18), (2018), pp. 481–494.

[43] S. Yan, *Understanding lstm and its diagrams*, MLReview. com, (2016).

[44] Q. Yang, X. Jia, X. Li, J. Feng, W. Li, and J. Lee, *Evaluating feature selection and anomaly detection methods of hard drive failure prediction*, IEEE Transactions on Reliability, 70 (2020), pp. 749–760.

[45] W. Yu, Q. Cheng, and S. Guan, *Improved variable step-size least mean square algorithm based on sigmoid function*, Procedia Computer Science, 199 (2022), pp. 1466–1473.

[46] I. N. Yulita, M. I. Fanany, and A. M. Arymuthy, *Bi-directional long short-term memory using quantized data of deep belief networks for sleep stage classification*, Procedia computer science, 116 (2017), pp. 530–538.

[47] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma, *Proactive drive failure prediction for large scale storage systems*, in 2013 IEEE 29th symposium on mass storage systems and technologies (MSST), IEEE, (2013), pp. 1–5.