Optimizing Small-Sample Disk Fault Detection Based on LSTM-GAN Model

YUFEI WANG, XIAOSHE DONG, LONGXIANG WANG, WEIDUO CHEN, and
XINGJUN ZHANG, Xi'an Jiaotong University

In recent years, researches on disk fault detection based on SMART data combined with different machine learning algorithms have been proven to be effective. However, these methods require a large amount of data. In the early stages of the establishment of a data center or the deployment of new storage devices, the amount of reliability data for disks is relatively limited, and the amount of failed disk data is even less, resulting in the unsatisfactory detection performances of machine learning algorithms.

To solve the above problems, we propose a novel small sample disk fault detection (SSDFD)¹ optimizing method based on Generative Adversarial Networks (GANs). Combined with the characteristics of hard disk reliability data, the generator of the original GAN is improved based on Long Short-Term Memory (LSTM), making it suitable for the generation of failed disk data. To alleviate the problem of data imbalance and expand the failed disk dataset with reduced amounts of original data, the proposed model is trained through adversarial training, which focuses on the generation of failed disk data. Experimental results on real HDD datasets show that SSDFD can generate enough virtual failed disk data to enable the machine learning algorithm to detect disk faults with increased accuracy under the condition of a few original failed disk data. Furthermore, the model trained with 300 original failed disk data has a significant effect on improving the accuracy of HDD fault detection. The optimal amount of generated virtual data are, 20–30 times that of the original data.

CCS Concepts: • **Computer systems organization** → **Reliability; Dependable and fault-tolerant systems and networks**; • **Computing methodologies** → **Neural networks**; Ensemble methods; *Machine learning*;

Additional Key Words and Phrases: Fault detection, reliability, hard disk drives, deep learning, generative adversarial networks

ACM Reference format:

Yufei Wang, Xiaoshe Dong, Longxiang Wang, Weiduo Chen, and Xingjun Zhang. 2022. Optimizing Small-Sample Disk Fault Detection Based on LSTM-GAN Model. *ACM Trans. Arch. Code Optim.* 19, 1, Article 13 (January 2022), 24 pages.
<https://doi.org/10.1145/3500917>

¹<https://github.com/wyfidane/SSDFD>.

This research was funded by the National Key Research and Development Plan of China under grant No. 2016YFB1000303. Authors' address: Y. Wang, X. Dong, L. Wang (corresponding author), W. Chen, and X. Zhang, Xi'an Jiaotong University, Xian Ning west road No. 28, Xi'an City, Shaanxi 710049, China; emails: wyfidane@stu.xjtu.edu.cn, {xsdong, wlx419}@xjtu.edu.cn, victorno2@stu.xjtu.edu.cn, xjzhang@mail.xjtu.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

1544-3566/2022/01-ART13 \$15.00

<https://doi.org/10.1145/3500917>

1 INTRODUCTION

With the continuous development of storage technology, large-scale data centers usually deploy hybrid storage servers integrating a great diversity of **hard disk drives (HDDs)** in their underlying storage devices; examples include the data servers of Alibaba Pangu [9], Amazon [33], Google [14], Facebook [38], and Microsoft Azure [5]. In such data centers, it has been extremely challenging to ensure high availability and reliability for IT management, as various disk failures constantly occur in the field. Data centers usually adopt some data protection mechanisms, such as data copy or erasure codes [5, 24]. If the drives fail to recover the lost data despite the data protection capabilities, permanent data loss occurs, and the system cannot be used, which would be disastrous for the data centers. HDDs are fairly complex devices consisting of a wide variety of magnetic, mechanical, and electronic components, each of which can fail. As a result, HDDs have different levels and manifestations of failures for a variety of reasons, and this can be observed in many data centers [17, 32, 35, 36, 41]. Compared with traditional passive fault-tolerant techniques such as **Erasure Code (EC)** and **Redundant Arrays of Independent Disks (RAIDs)**, active fault detection techniques can guarantee the reliability and availability of large-scale storage systems in advance. Thus, the risk of data loss can be reduced by successfully identifying disk failures.

In order to monitor the health status of HDDs, manufacturers generally implement **self-monitoring, analysis and reporting technology (SMART)** [1] in the firmware of devices. The SMART attributes contain the disk state information and possible defects. Internally, disks use the so-called “threshold method” based on SMART values to evaluate the failures, which means the disks would raise alarms if the values of one or more of the SMART attributes cross the corresponding predefined threshold. However, this “threshold method” only achieved a 3%–10% **failure detection rate (FDR)** and a 0.1% **false alarm rate (FAR)** in practice [30]; in other words, this method is too conservative and misses opportunities to detect disk failures.

Several disk fault detection models based on SMART data with machine learning algorithms [40–42] have been proposed to improve the predictive performance. Unfortunately, these methods require a large amount of disk data to train the models. According to [35], in the early stages of the establishment of a data center or the deployment of new storage devices, a few failed disk data can be obtained. But in general, because the quality of disks in the same batch is relatively similar, the total amount of failed disk data is relatively small. Due to the small available sample size and the insufficient data, using a small amount of training data in traditional machine learning algorithms greatly increases the risks of overfitting or weak generalization, which weaken the performance of the model and seriously affect the reliability of the storage systems.

To obtain enough failed disk data to train the model, some data synthesis and extension methods can be considered. Under-sampling or over-sampling techniques such as **Synthetic Minority Oversampling Technique (SMOTE)** [7], Wilson’s **edited nearest neighbor (ENN)** [39], **Adaptive Synthetic sampling (ADASYN)** [21], and so on, can balance the dataset to some extent; however, there may be a large deviation between the simulated failed disk data and the real failed disk data by fitting the characteristic function of the sample with the traditional method, so these are not ideal data. Therefore, in the case of insufficient failed disk data, the construction of a disk fault detection model requires an effective method as the foundation for generating failed disk data.

To tackle the above-mentioned problem, this article proposes a novel **small-sample disk fault detection (SSDFD)** optimizing method, with synthetic data using **generative adversarial networks (GANs)** [15]. The proposed approach utilizes GAN to generate failed disk data conforming to the failed disk data distribution and expands the dataset with the generated data, then the classifiers are trained. Because the disk SMART attributes vary with the usage and are time-related;

therefore, we adopt the **Long Short-Term Memory (LSTM)**, which is good at learning the characteristics of time series data as the GAN generator to fit the distribution of SMART data, and use the multi-layer neural network as the discriminator to train the GAN-model to generate realistic failed disk data. With sufficient generated failed disk data samples, ML algorithms can detect disk faults more precisely than before with the small original failed disk data samples.

The rest of this article is organized as follows. Section 2 presents the related works. Section 3 presents the background knowledge. Section 4 describes the proposed method. The dataset description and the experimental results are listed in Sections 5 and 6, respectively. Finally, Section 7 summarizes the conclusion of this article.

2 RELATED WORKS

In recent years, well-known artificial intelligence technologies such as the CNN [25], LSTM [6], stacked automatic encoder, and deep belief network [15, 37] have been developed in a number of ways. In terms of data processing, adaptive feature learning and multilayer nonlinear mapping, machine learning and deep learning are superior to many traditional mathematical methods, and they solve many problems in scenarios such as image processing, label classification, and prediction. Machine learning and deep learning techniques are also widely used in disk fault diagnosis [22, 40, 41, 41].

Support Vector Machines (SVMs) can utilize kernel functions to effectively perform nonlinear classification and implicitly map their inputs to high-dimensional feature spaces [12]. Zhu et al. [44] studied the BP neural network and an improved SVM model to establish an HDD failure prediction model based on SMART data. Li et al. [28] proposed a new HDD failure prediction model based on the **Gradient Regression Tree (GBRT)**. The GBRT is a kind of gradient descent enhancement technology based on tree theories. It is an accurate and effective machine learning method that can be applied to regression and classification problems. In addition, the **Regularized Greedy Forests (RGF)** method is a powerful nonlinear classification approach derived from the GBRT that decouples structural search from optimization and uses the concept of structural sparsity to conduct greedy searches of forest nodes directly based on the forest structure. Mirela et al. [3] used this method to model disk faults, and the results were relatively good. In particular, LSTM removes or increases information to cell state through the “gate” structure, which can alleviate gradient explosion and gradient vanishing to a certain extent and effectively solve the problem of long-term backpropagation. LSTM also has been adopted to build HDD failure prediction models in recent works [10, 23, 29–31]. Hu et al. [23] replaced the input of LSTM network with the continuous operation record of HDDs considering the individual differences between each HDD sample. Therefore, the prediction model is able to learn the status information of the HDDs over a period of time and predict whether the HDDs will fail. Experimental results show that the proposed method can predict HDD faults with the precision of 86.31%.

However, machine-learning algorithms require a large amount of data, and their performances are not good enough when dealing with small sample problems [13]. Therefore, improving the performance of the machine-learning algorithms through data generation and other technologies should be considered. Data resampling is a common method to deal with imbalanced-learning problems, which mainly includes under-sampling, over-sampling, and hybrids methods. Under-sampling removes the majority class to balance the dataset such as ENN [39]. Over-sampling generates new samples for the minority class. SMOTE [7] made some improvements on the basis of the random over-sampling method. The basic idea is to calculate the k -nearest neighbors of each minority sample, and then randomly select some k -nearest neighbors from the original sample to synthesize new samples and add them to the dataset according to the formula. Compared with the random over-sampling method, SMOTE avoids the overfitting caused by simple duplicating

samples in the synthesis of new samples and plays a positive role in the generalization of the model, but it also has problems in the selection of nearby neighbors, and it is difficult to overcome the data distribution problem of imbalanced datasets, which tends to produce distribution marginalization. Han et al. optimized SMOTE and proposed BorderlineSMOTE [20]. BorderlineSMOTE takes into account the importance of the boundary and the samples nearby for classification, and only uses a few class samples on the boundary to synthesize new samples, thus improving the category distribution of samples. ADASYN [21] uses the distribution to automatically determine how many synthetic samples each minority class sample needs to produce, that is, to assign different weights to different minority samples, the more the surrounding majority class samples, the higher the weight assigned, but this method is susceptible to outliers. Although these methods can balance the dataset to some extent, the traditional methods are not accurate enough to fit the characteristic function of the samples.

Transfer learning is able to apply the model trained in the source domain to the target domain leveraging the similarity between data, tasks, or models. The target domain can still be modeled under the condition of insufficient labeled data. Han et al. [19] designed a deep transfer learning method on a small and unbalanced CT image dataset of the PNs and obtained higher efficiency than the traditional machine learning method for malignant classifications. Chen et al. [8] improved the recognition accuracy of electrocardiogram under small dataset by introducing transfer learning ideas into the training process of deep residual network. For the problem of HDD fault detection, applying transfer learning requires similar reliability characteristics between the old and new HDDs and plenty of data in the source HDDs domain. Normally, HDDs from different manufacturers have different electrical characteristics and different reliability attributes with different calculation rules. That is to say, if new HDDs are from different manufacturers than old HDDs, the reliability of the new HDDs is quite different from that of the old HDDs. In addition, the data shortage of new HDDs in the initial period of use also means the knowledge loss in the source domain, and this will both affect the effect of transfer learning. The method proposed in this article does not need a lot of existing knowledge in the source domain as the foundation. Our method expands the total amount of data by generating virtual data to solve the problem of low HDD fault detection accuracy caused by insufficient data.

As a very famous data generation technology in recent years, GANs have verified their data generation capability in many fields [18, 27, 34]. Greff et al. [16] proposed a cross-domain fault diagnosis method. When testing data under machine failure conditions that are not conducive to training, reliable cross-domain diagnostic results can be derived, and the proposed method can manually generate pseudo samples for domain adaptation. Guo et al. [18] proposed a data generation model called SDAE-GAN, and the experimental results showed that the developed SDAE-GAN method for planetary gearboxes has good anti-noise ability and achieves strong fault diagnosis performance in cases with small samples. A method incorporating GANs for statistical parametric speech synthesis was proposed in [34]. In [27], a generative adversarial network for generating realistic underwater images from in-air image and depth pairings was proposed. Using online learning technology, Xiao et al. [40] proposed an HDD failure prediction mechanism based on an online random forest. This model can discard outdated trees and introduce new trees according to the distribution characteristics of new data, making the model evolve with the arrival of new data and effectively improving the prediction performance.

Since the data are an important basis of ML algorithm performance, in view of the problems that lack disk reliability data in the early stages of the establishment of a data center or the deployment of new storage devices, we propose a novel failed disk data generation method, SSDFD, based on LSTM-DNN GAN. Considering that LSTM is good at dealing with long-term memory problems, the proposed GAN model with LSTM-based generator can better capture the time series characteristics

Fig. 1. Architecture of a GAN.

of HDD SMART data and generate more realistic virtual HDD SMART data compared with existing generation methods. And, the Gaussian noise is added after the virtual data are generated to make the generated virtual data closer to the HDD SMART data collected in the real environment. By synthesizing failed disk data, the performance of the ML algorithm is effectively optimized for disk fault detection.

3 PRELIMINARY KNOWLEDGE

3.1 Generative Adversarial Networks

Recently, generative models such as GANs [15] have attracted much interest from researchers and industrial practitioners. A GAN contains two components: A generator denoted as G and a discriminator denoted as D , as shown in Figure 1. G captures the potential distribution of real data samples and generates fake samples, and D determines whether the input is real data or generated data. GANs are equivalent to mini-max game problems. The optimization goal is to achieve Nash equilibrium, which can be described by the following formula:

$$\min_G \max_D V(D, G) = \min_G \max_D (E_{x \sim p_x(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log(1 - D(G(z)))]). \quad (1)$$

where x is the real data, its distribution is p_x , and z is a random vector. The generated data $G(z)$ can be obtained from G based on the input z . p_G is the distribution of $G(z)$. $D(x)$ is the probability that x comes from P_x instead of P_g . Ideally, if $x \sim P_x$, then $D(x) = 1$; if $x \sim P_g$, then $D(x) = 0$. D is trained to maximize the probability of assigning correct labels to real data x and generated data $G(z)$.

The model can be modified during training and can update itself with the parameters of another model. Given m real samples and m generated samples, the stochastic gradients of D and G are calculated by Equations (2) and (3), respectively:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(x^{(i)}) + \log(1 - D(G(Z^i)))], \quad (2)$$

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m [\log(1 - D(G(Z^i)))]. \quad (3)$$

By alternating iterations, the mini-max problem is likely to obtain the global optimal solution at $P_g = P_{data}$, the loss function converges, and the GAN reaches Nash equilibrium.

In practice, Equation (1) may not provide a sufficient gradient for G to learn. In the early stage of learning, the performance of G is poor such that the generated data are obviously different from the real data, and D can accurately distinguish the real data from the generated data. In this case, $\log(1 - D(G(z)))$ saturates. While training G , maximizing $\log D(G(z))$ is better than minimizing

Fig. 2. Architecture of an LSTM cell.

$\log(1 - D(G(z)))$. The objective function causes G and D to have the same fixed point and provides a strong gradient in the early learning stage. G implicitly defines a probability distribution P_g , which is the distribution of the sample $G(z)$ obtained when $z \sim P_z$. Hence, the estimators of P_{data} can converge to an improved state through adversarial training with enough space and training time. After completing the training of the GAN, G can effectively estimate the probability distribution of the original real data samples and generate fake samples in line with the real sample distribution to expand the overall sample size.

3.2 Long Short-Term Memory (LSTM) Models

LSTM is an extension of the **Recurrent Neural Network (RNN)**. The LSTM network was first proposed by H. Schmidhuber [22] with a memory cell and further improved by Gers et al. [13] with an extra forget gate. It can effectively solve the vanishing gradient problem. The structure of the LSTM cell is shown in Figure 2.

To obtain long-term time dependence, the LSTM algorithm defines and maintains the unit state to regulate information flow; this is an important mechanism of LSTM [16]. The cell state C_{t-1} interacts with the intermediate output h_{t-1} and the subsequent input x_t to determine, which elements of the internal state vector should be updated, maintained, or removed based on the output of the previous time step and the input of the current time step. The formulas of the LSTM network are described as follows:

$$i_t = \sigma(x_t U^i + h_{t-1} W^i), \quad (4)$$

$$f_t = \sigma(x_t U^f + h_{t-1} W^f), \quad (5)$$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o), \quad (6)$$

$$\hat{C}_t = \tanh(x_t U^g + h_{t-1} W^g), \quad (7)$$

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t, \quad (8)$$

$$h_t = \tanh(C_t) * o_t. \quad (9)$$

where the operator $*$ represents element-wise multiplication and σ represents the sigmoid activation function; i , f , and o denote the input, forget, and output gates, respectively; W^i , W^f , W^o , and W^g represent the weight matrices that need to be learned during training; U^i , U^f , U^o , and U^g are coefficient matrices; $\hat{C}^t$ is a “candidate” hidden state that is calculated based on the current input and the previous hidden state; C_t is the internal memory of the unit; h_t represents the final output of the memory unit. Via the function of various gates, LSTM memory units can capture the

Fig. 3. Architecture of proposed approach.

complicated correlation features within time series in both the short and long terms, and this is a remarkable improvement over other RNNs.

4 PROPOSED METHOD

In a real environment, HDD faults tend to occur gradually over time. Therefore, the HDD reliability characteristics have a strong time-related correlation. Compared with traditional data generation methods, LSTM is better at capturing temporal characteristics of samples and extracting time-related features. Therefore, we adopt LSTM as the generator of the proposed GAN model to fit the probability distribution function of failed HDD samples, so that the GAN model can better capture the temporal characteristics of HDD-SMART data and generate more realistic virtual data. In addition, Gaussian noise is added after the virtual data are generated, so that the generated virtual data can be closer to HDD-SMART data collected from the real environment. Finally, the effectiveness of the proposed method is verified by experimental results.

4.1 SSDFD Models

The proposed method can generate synthesis data from random noise and add Gaussian noise to smooth the data according to the characteristics of the original real data. Its structure is shown in Figure 3. The main steps are as follows:

(1) The collected data contain all the characteristics of the real samples and are processed into a format suitable for training in the preprocessing stage. The noise is then fed into the generator model, and by learning the characteristics of these samples, the generator acquires primary generation capability. At this stage, the discriminator cannot be trained to acquire the primary discriminability.

(2) The generator obtains the ability of interference discrimination after training. To train the discriminator based on the generated data, its parameters must be fixed. At this stage, the generator and discriminator can capture the characteristics of the data because LSTM has a good ability to learn the characteristics of the time-series data, and the DNN can also process the connections between different dimensions of the data. The parameters of the LSTM-based generator and DNN-based discriminator are shared. The generator tries to make the generated data as close to

the real data as possible so that the discriminator cannot determine whether the data are fake or real. The discriminator recognizes whether the data are generated fake data, and it does this as much as possible. Iterative training is performed with the generator and discriminator by alternating learning mechanisms, and the parameters are updated to improve performance until Nash equilibrium is achieved.

(3) After Nash equilibrium is achieved, Gaussian noise is added to the data generated by the generator to smooth the data and enhance its representational ability. Then, the data samples are input into the fault classifier for fault detection, and the diagnosis results are output.

4.2 Generation Process

The structure of the generator is shown in Figure 4. The LSTM-based generator and the DNN-based discriminator constitute the GAN model. The effect of the generator is verified by the discriminator. The effect of the discriminator is verified by the real data $X_{real}x^m (m = 1 \dots M)$ and the generated fake data $X_{fake}x^k (k = 1 \dots K)$ together. The loss function iteratively trains the generator and discriminator using real labels and fake labels. Thus, the generator can generate fake samples that approximate the real samples after the iterations are completed. Then, the generated fake samples with Gaussian noise and the real samples are combined, and fault detection is carried out. The training process is as follows:

A random vector $z^k (1 \dots K)$ sampled from the random noise is input into the generator with the initial labels and are mapped to the hidden layer vector h_z^k . The corresponding labels y_{fake}^k are generated from the fake samples $x_{fake}^k (1 \dots K)$:

$$h_z^k = \tanh \left(f_t * C_{t-1} + i_t * \left(\tanh \left(x_t U^g + h_{t-1} W^g \right) \right) \right) * o_t, \quad (10)$$

$$x_{fake}^k = \sigma \left(W_z * h_z^k \right). \quad (11)$$

where U_g and W_g represent weight matrices and f_t , i_t , and o_t represent the forget, input, and output gates, respectively. $\tanh$ and σ are activation functions.

4.3 Training of the Discriminator

Fake samples $x_{fake}^k (k = 1 \dots K)$ are labeled 1, real samples $x_m (m = 1 \dots M)$ are labeled 0, and the corresponding output labels are d_{fake}^k and d_{real}^m . The training of the discriminator can be completed by minimizing the loss function of the proposed model. The **Binary Cross-Entropy (BCE)** loss is selected as the loss function because the problem is a binary classification problem. The calculation formulas are as follows:

$$L_{fake} = -W_{fake}^k \left[y_k \log d_{fake}^k + (1 - y_k) \log (1 - d_{fake}^k) \right], \quad (12)$$

$$L_{real} = -W_{real}^m \left[y_m \log d_{real}^m + (1 - y_m) \log (1 - d_{real}^m) \right], \quad (13)$$

$$L = \arg \min_{\theta} (L_{fake} + L_{real}). \quad (14)$$

where L_{fake} and L_{real} represent the BCE loss errors of fake and real labels, respectively, the loss function of the discriminator is expressed as L , and the parameter set is expressed as $\theta = \{\theta_1, \theta_2, \dots, \theta_{N+1}\}$.

Fig. 4. Structure of the generation method.

4.4 Training of the Generator

When the fake samples x_{fake}^k ($k = 1 \dots K$) are input into the discriminator, if the output label is 0, this means that the generated fake samples cannot pass the discriminator; if the output label is 1, then the discriminator can no longer distinguish whether the generated fake samples are real data or generated fake data. The training of the generator is completed by minimizing Equation (15):

$$L_g = -W_{fake}^k \left[y_k \log d_{fake}^k + (1 - y_k) \log (1 - d_{fake}^k) \right], \quad (15)$$

$$L_G = \arg \min_{\theta} (L_g). \quad (16)$$

The loss error of the label is represented by L_g , the loss function of the generator is L_G , and its parameter set is $\theta' = \{\theta_z, \theta'_z\}$. The output of the generator is similar to that of the discriminator, with an optimal value of 0.5.

4.5 Adversarial Training Mechanism

The purpose of the discriminator is to make the output vector d_m as close to 1 as possible when the real samples x_m ($m = 1 \dots M$) are input into the model and make the output vector d_{fake}^k as close to 0 as possible when the fake samples x_{fake}^k ($k = 1 \dots K$) are input to the model. If d_{fake}^k is very close to 1, then the samples generated by the generator can successfully “fool” the discriminator. The zero-sum game, in which the optimization goal is converted to a mini-max problem, occurs between the discriminator and the generator.

ALGORITHM 1: Minibatch stochastic gradient descent training of the proposed SSDFD. The number of steps to apply to the discriminator, n is a hyper parameter.

```

1 Initialize sub-generators,  $G_{i:k}$ ; discriminator,  $D$ 
2 for number of training iterations do
3   for  $n$  steps do
4     Sample  $m$  samples  $\{x^1, \dots, x^m\}$  from  $p_{data}(x)$ 
5     for  $i = 1$  to  $k$  do
6       | Sample  $m_i$  noises  $\{z_i^1, \dots, z_i^{m_i}\}$  from  $p_g(z)$ 
7     end
8     Update  $D$  by ascending its stochastic gradient:
       $\nabla_{\theta_d} \frac{1}{m} \sum_{j=1}^{m_i} [\log D(x^{(i)}) + \sum_{i=1}^k \log(1 - D(G_i(z_i^j)))]$ 
9   end
10  Divide  $p_{data}(x)$  equally into  $k$  subsets  $p_{data}(x_i)$  based on their similar  $D(x)$ 
11  for  $i = 1$  to  $k$  do
12    |  $T_i = \min(D(x))$  ( $x \in p_{data}(x_i)$ )
13  end
14  Sample  $m$  noises  $\{z^1, \dots, z^m\}$  from  $p_g(z)$ 
15  Update  $G_i$  by descending its stochastic gradient:
     $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m [T_i \log(D(G_i(z_i^j))) + (1 - T_i) \log(1 - D(G_i(z_i^j)))]$ 
16 end
17 until the mini-max game reaches a Nash equilibrium
18 for  $n$  steps do
19   Sample  $m$  samples  $\{x^1, \dots, x^m\}$  from  $p_{data}(x)$ 
20   for  $i = 1$  to  $k$  do
21     | Sample  $m_i$  noises  $\{z_i^1, \dots, z_i^{m_i}\}$  from  $p_g(z)$ 
22   end
23   Update  $D$  by ascending its stochastic gradient:  $\nabla_{\theta_d} \frac{1}{m} \sum_{j=1}^{m_i} [\log D(x^{(i)}) + \sum_{i=1}^k \log(1 - D(G_i(z_i^j)))]$ 
24 end
25 until convergence
26 Any gradient-based learning rule can be used to update the gradient of the network. The Adam is
   adopted in the experiment.

```

The generator and the discriminator are optimized alternately during the training process. The parameters of the generator are first fixed and the output of the discriminator is maximized. Then, the parameters of the discriminator are fixed and the generator is optimized. The above steps are repeated until the generator and discriminator reach Nash equilibrium, and then the training is complete. Algorithm-1 summarizes the training process of SSDFD.

4.6 Adding Noise

Normally, noise is caused by many different sources of noise rather than a single source. Assuming that the real noise is treated as the summation of many random variables with different probability distributions, and each random variable is independent, then according to the central limit theorem, its normalized sum approaches a Gaussian distribution as the number of noise sources increases. The data are affected by many factors in the real environment. For the generated fake data, to effectively simulate the data in the real environment, Gaussian noise with an expectation of 0 and a standard deviation of 1 is added to the fake data in our experiments [43].

Table 1. Overview of the Dataset

	Backblaze HDD	Baidu HDD
Disk Model	ST12000-NM0007	ST31000-524NS
Capacity	4T	1T
Total Disks	37,004	23,395
Failed Disks	1156	433
Duration	12 months	20 days
Positive Items	12,721,076	3,857,616
Negative Items	1,156	20,480

5 DATASET DESCRIPTION AND PREPROCESSING

5.1 Datasets

To evaluate the performance and restrictions of the proposed method, we use: (1) the HDD dataset from Backblaze, which spans a period of 12 months ranging from January 2019 to December 2019 [2]; (2) the HDD dataset from Baidu [26]. The datasets contain SMART data about the disks and some basic information, such as timestamps and device serial numbers. It is important to note that the SMART attributes provided by different manufacturers may be different, and some attributes may have different meanings depending on the type of device. Therefore, we select one HDD model with complete data from the Backblaze dataset, one from the Baidu HDD dataset to perform the experiments. The basic information about the datasets is shown in Table 1. The Backblaze HDD dataset contains daily snapshots, and the sampling interval of the Baidu HDD dataset is 1 hour. At the initial stage of use, the variation trend of SMART attributes is relatively stable, and the failure rate is low; in the later part of the lifecycle, many SMART attributes show obvious changes, such as the Seek Error Rate, Uncorrectable Errors, and so on.

Total Disks contains both good disks and failed disks. Duration indicates how long the disk records last. Positive Items are the number of disk records for good disks. Negative Items are the number of disk records for failed disks.

5.2 Feature Selection

In our experiments, the Backblaze HDD and Baidu HDD are referred to as HDD A and HDD B, respectively. The dataset HDD A contains 42 disk models from different manufacturers. For each disk model in HDD A, the data include disk serial number, model, capacity, flag of failure, and so on, and 126 SMART attributes. However, some attributes are irrelevant to build the model, so before performing an in-depth analysis, we carry out “feature selection” to remove repetitive and irrelevant features, and this can also shorten the time required for model training and improve the performance of the model.

Feature selection removes redundant SMART attributes from the dataset, leaving a group of SMART attributes combination that describe the dataset best. The most important features are selected through **Recursive Feature Elimination (RFE)**, and SVM is selected as the estimator for RFE in our experiment. The RFE result is displayed in Figure 5. After RFE, 12 SMART attributes from HDD A are selected to build the corresponding model. Since HDD B only records 12 SMART attributes so we select all 12 attributes. The selected SMART attributes are listed in Table 2.

The SMART data in each dataset contain raw values denoted as Raw and normalized values denoted as Norm. The Norms in the dataset are calculated by Raw according to the manufacturer’s nonpublic custom formulas. As some Norms may result in a loss of data accuracy and the

Fig. 5. The RFE result of HDD A.

Table 2. Selected Features

Attribute Name	HDD A (BackBlaze)	HDD B (Baidu)
Raw Read Error Rate	Norm	Norm
Spin Up Time	Norm	Norm
Reallocated Sectors Count	Norm&Raw	Norm&Raw
Seek Error Rate	Norm	Norm
Power On Hours	Raw	Norm
Reported Uncorrectable Errors	Norm&Raw	Norm
High Fly Writes		Norm
Temperature		Norm
Hardware ECC Recovered		Norm
Current Pending Sector Count	Norm&Raw	Norm&Raw
Uncorrectable Sector Count	Norm&Raw	

corresponding Raw may be highly sensitive to changes in disk health, Raw and Norm are all used in the experiments, as shown in Table 2.

The range of values spanned by different features varies widely. To avoid bias towards features with large values, we apply feature scaling for data normalization according to the following formula:

$$x_n = \frac{x - x_{min}}{x_{max} - x_{min}}, \quad (17)$$

where x is the original value of a feature and x_{max} and x_{min} are the maximum value and the minimum value of this feature, respectively.

5.3 Visualization of Generated Samples

We selected some SMART attributes to visualize the generated samples and verify whether the proposed method can generate realistic enough failed disk data to detect hard disk failures. Detailed description of the datasets is given in Section 5.1. Due to space limitations, we selected three SMART attributes and plotted the general change trend of the real samples and generated samples. Because the good disk data in the dataset are sufficient and there is only a few failed disk data, so we focus on the failed disk data generation. As shown in Figure 6, we visualize three real failed disk data and the corresponding generated data. As we can see, the generated failed disk data display similar amplitudes and shape to the real data, with most peaks and troughs occurring at roughly

Fig. 6. The comparison of generated data and original data. (a) HDD A Raw read error rate. (b) HDD B Raw read error rate. (c) HDD B Seek error rate. The x-axis is time and the y-axis is the normalized value of the SMART attribute.

the same locations. We also observe fluctuations in the generated failed disk data. For example, in Figure 6(c), near time 50, the generated data show an earlier rise than the real data, followed by a larger drop. Although the real data look smoother, the proposed method can also capture the trend of changes in the SMART attributes of the failed disk data, indicating that the proposed method can effectively generate very realistic failed disk data.

6 EXPERIMENTAL RESULTS

6.1 Evaluations of SSDFD

We use Precision and Recall as the metrics to measure the performances of different ML algorithms. Precision indicates the proportion of **true positives (TPs)** among all predicted failures. Recall represents the proportion of TPs within all actually failed disks. These metrics are defined as

$$Precision = \frac{TP}{TP + FP}, \quad (18)$$

$$Recall = \frac{TP}{TP + FN}, \quad (19)$$

where TP is “true positive”, FP is “false positive”, and FN is “false negative”.

The batch size in our experiment is set as 100. The generators for HDD A and HDD B have the same architectures, which are composed of four sublayers. The input layer has 12 units and represents the randomly sampled signals in shape (100, 12) of a normal distribution with a mean of 0 and a standard deviation of 1. The next layer is the first hidden layer and the size of the LSTM cell in the first hidden layer is 50. The third layer is also hidden which is same as the first one. The output layer is a linear connection with the shape (50, 12), tanh is used as the activation function. It is learned by Adam optimization algorithm with the learning rate of 0.0003.

Fig. 7. The costs of the generator and discriminator for (a) HDD A and (b) HDD B during training.

The discriminator architecture for HDD A and HDD B is as follows. The generated data in shape (100, 12) is delivered to the input layer. The next is a linear mapping, which transforms the data to shape (12, 50). The output layer is also a linear connection that transforms the data to shape (50, 1). Sigmoid is used as the activation function. The discriminator is learned by Adam optimization algorithm with the learning rate of 0.0003.

The generator and discriminator for HDD A and HDD B are trained separately. Figure 7 shows the costs of the generator and discriminator for the two models during the training process. It can be observed that after some training time, a balance is reached between the generator and the discriminator. The gradient is zeroed before each backpropagation, and this not only allows the model to avoid overfitting and modal collapse but also causes it to converge to the GAN model.

We implement the proposed method based on PyTorch 1.7.1 and scikit-learn 0.24.1. The experiment is done on the hardware environment of Intel Xeon E5-2620 v3 2.4 GHz CPU, 32 GB RAM, and Ubuntu 20.04 OS. The training is accelerated by two RTX 2080ti GPUs, and it is completed when the outputs of real samples and fake samples are both close to 0.5.

To demonstrate the effectiveness of SSDFD, five commonly used classification algorithms are compared: **multilayer perceptron (MLP)**, **random forest (RF)**, **logistic regression (LR)**, **decision tree (DT)**, and SVM. Each dataset is randomly divided into a training set and a test set with a ratio of 7:3. Then we expand the failed disk data in the training set, the generated failed disk data of different folds are added to the training set and the above classifiers are trained. Finally, the test is conducted by the test set to validate the effects of the model.

Before the large-scale experiments, we first conduct the faithfulness test of SSDFD. We randomly select 100, 300, 500, 700, and 900 failed disk data samples from HDD A and HDD B, respectively, to generate the virtual data based on each data volume, and then combine the original data with the generated virtual data to form the mixed dataset. The volume of the failed disk data in the mixed dataset is equal to the original dataset. For example, as the amount of failed disk data in HDD A is 1,156, we generate 1,056 pieces of virtual data based on 100 original failed disk data, then combine the 100 pieces of original data with 1,056 pieces of virtual data to form the mixed dataset, it is similar for other data volumes. As shown in Figure 8, for HDD A, due to the insufficient failed disk data in the original dataset, the mixed dataset based on 100 and 300 original data is far inferior to the original dataset in terms of both precision and recall. When the virtual data is generated based on the data volume of 500, 700, and 900, the precision and recall of the mixed dataset are significantly improved, which is still slightly lower than the original dataset, but close to it. The results of HDD B are shown in Figure 9. Similarly, the precision and recall of the mixed dataset based on 100 and 300 data volumes are much lower than that of the original dataset. However, as the data volume of failed disk data in the original dataset has increased a lot compared with HDD A, so the precision of the mixed dataset based on the data volume of 500, 700, and 900 increases faster than HDD

Fig. 8. The (a) precision and (b) recall of HDD A with N raw failure data plus generated data.

Fig. 9. The (a) precision and (b) recall of HDD B with N raw failure data plus generated data.

A, and almost reaches the same level as the original dataset. Although the recall is slightly lower than the original dataset, it is very close. It shows that SSDFD based on 500 original data is able to generate the virtual data with characteristics very close to real data under the condition of equal quantity comparison.

In order to verify the effect of adding different times of generated failed disk data on the accuracy of fault detection under the condition of different volumes of original failed disk data, we randomly select 50,000 pieces of good disk data and 900 pieces of failed disk data from HDD A and HDD B are selected so that the ratio of good disk data to failed disk data is set to 50:1 to simulate the small sample environment. For the real failed disk data volumes of 100, 300, 500, 700, and 900 in HDD A and HDD B, the generated failed disk data are added in amounts of 1–30 times that of each data volume ($100/300/500/700/900 \times 1\text{--}30$ fold) to train the classifiers, respectively. The effect of SSDFD is verified from multiple angles through the experimental settings of multiple generated failed disk data based on different volumes of original failed disk data.

Ten times experiments are proceeded and the mean values are taken as the experimental results. The precision and recall of the HDD A and HDD B are shown in Figures 10 and 11, respectively. The “Raw” in Figures 10 and 11 means the amounts of original failed disk data, “ x -fold” means the generated failed disk data is added by x times to the original data. For example, in Figure 10(a), “100” in caption means the amounts of original failed disk data is 100 and 1–30 fold ($100/500/1,000/1,500/2,000/2,500/3,000$) generated failed disk data are added respectively.

For HDD A, it can be seen from Figure 10(a)–(d) that when the original failed disk data volume is 100 or 300, the precision values of different methods are approximately 30%–40%, and the recall is approximately 40%. When 1–15-fold generated failed disk data are added, both precision and recall are slightly improved. When 20–30-fold generated failed disk data are added, the improvement is

Fig. 10. The precision and recall of HDD A with multiple generated samples.

Fig. 11. The precision and recall of HDD B with multiple generated samples.

particularly obvious. The increase rate for 100 original failed disk data is approximately 15% and that of 300 original failed disk data is even greater (approximately 25%). The enhancement is nearly saturated with 25-fold or 30-fold generated failed disk data. The precision and recall of 500 original failed disk data are shown in Figure 10(e) and (f). When adding 1-fold–15-fold data, the precision improvement yielded by different methods is significantly improved, with the average value being higher than 95% and the average improvement rate being approximately 15%. Since the training process is only based on failed disk data, the recall fluctuates slightly between 1-fold and 5-fold data, and this is similar to the recall fluctuations of 100 and 300 failed disk data. The precision and recall for 700 and 900 original failed disk data are shown in Figure 10(g)–(j). With 700 original failed disk data points, the average precision of the five methods is lower than 70%, and the average recall is just higher than 60%. The average precision for 900 original failed disk data using different methods is slightly higher than that obtained using 700 items, and the average recall is basically the same as that obtained using 700 items. When multiple failed disk data are added, there are obvious improvements in precision and recall, and the trends of other experimental groups are similar. The precision and recall improve obviously after adding 20-fold generated failed disk data, and the improvement is close to saturation. When continuously adding 25–30-fold generated failed disk data, the precision and the recall are relatively stable, and the averages are higher than 99% and 85%, respectively.

The experimental results of HDD B are shown in Figure 11, in which the precision and recall for 100 and 300 original failed disk data points are displayed in Figure 11(a)–(d). Due to the small amount of data, the precision and recall of different methods are relatively low, and the performance of the GAN model is not good enough. When the amount of generated failed disk data added is less than a 10-fold increase, the precision and recall of different methods are slightly improved, and the precision and recall of 1-fold data are even lower than those of the original failed disk data. After adding 15-fold generated failed disk data, the precision rates of different methods are increased by approximately 5%–10%, and the average recall is increased by less than 5%. A significant improvement can be observed when 20–30-fold generated failed disk data are added, and the average precision rates for 100 and 300 raw original failed disk data rises to nearly 60% and 80%, respectively. In terms of recall, the average recall for 300 original failed disk data points exceeds 60%, which is approximately 10% higher than that of the average of 100 groups. The precision rates corresponding to the original failed disk data with amounts of 500, 700, and 900 points are shown in Figure 11(e), (g), and (i), and the recall values are shown in Figure 11(f), (h), and (j). When the volume of original failed disk data increases, the average precision of the different methods increases from 60%–75%, and the average recall also increases by approximately 10%. After the addition of different multiples of generated failed disk data, the overall trend of HDD B is similar to that of HDD A, and the improvement is significant after 15-fold generated failed disk data are added. When 20–30-fold generated failed disk data are added, the increase is almost unable to be observed, and the average precision and the recall exceed 99% and 85%, respectively.

When the generated failed disk data added is 20–30 times that volume of the original data, the improvements in precision and recall achieved by the different ML algorithms are close to saturation.

Moreover, to better observe the usage conditions of the proposed method, Figures 12 and 13 depict the precision and recall of training the model on the basis of 100, 300, 500, 700, and 900 original failed disk data with 30-fold generated failed disk data added. When 100 original failed disk data points are used to train the GAN model, for HDD A and HDD B, the precision is approximately 60%, and the recall is only 40%–50%. When the data volume reaches 300, the average precision and recall of HDD A are raised to 80% and 70%, respectively. The average precision and recall of HDD B are raised to 75% and 60%, respectively. Compared with the increases seen for 100 original failed

Fig. 12. The (a) precision and (b) recall of HDD A based on the GAN with N samples.

Fig. 13. The (a) precision and (b) recall of HDD B based on the GAN with N samples.

disk data, the increase, in this case, is more obvious. With 500 original failed disk data, the precision rates of different algorithms on HDD A and HDD B are significantly improved, almost exceeding 95%, and the average recall is also over 70%. As the volume of original failed disk data continues to increase, the precision rates for the two HDD datasets basically increase to the limit, and it is difficult to increase further. The average recall of the two HDD datasets also reaches 80%.

Looking at the precision and recall of HDDs in general, the trained model has a remarkable effect on the accuracy of HDD fault detection when it is trained with 300 original failed disk data.

The authors in [40] used the FDR and FAR to evaluate the effect of their method, where the meaning of FDR is the same as that of “recall” used in this article. The content is shown in Figures 10–13 in this article exhibits a phenomenon similar to that of [40], that is, in the initial usage stage of the disks, the detection effects of the ML algorithms are not ideal because of the lack of relevant reliability data. Furthermore, our method trains the model with less failed disk data than the amount used in [40] and achieves the same experimental results. Also, literature [23] adopted precision and recall as evaluation metrics to verify the prediction effect of the proposed LSTM model compared with DT, RF, and SVM based on the dataset from Backblaze. For different algorithms, our experimental results outperform [23] in both precision and recall. This is because we improve the structure of the proposed model according to the characteristics of disk reliability data by considering its temporal features, and this enables us to capture the characteristics of failed disk data more reasonably than [40] and [23], also the datasets are expanded with less raw data. For the other algorithms used in the comparative experiments in [40] and [23], our method can also achieve equivalent experimental results or outperform their results based on less failed disk data, thereby fully illustrating the effectiveness of the method proposed in this article.

6.2 Evaluations Compared to Other Generating Methods

In addition to the experimental results of SSDFD, we also conduct experiments based on other generating methods for comparison. SMOTE [7] utilizes KNN to interpolate data to generate new

Fig. 14. The comparison of (a) precision and (b) recall for HDD A based on SSDFD and SMOTE.

Fig. 15. The comparison of (a) precision and (b) recall for HDD B based on SSDFD and SMOTE.

data, the generated data does not exist in the original dataset so that new information can be added. SMOTE can reduce the risk of overfitting compared to basic over-sampling. This technique is a commonly used method to deal with imbalanced data, and has been unanimously recognized by the academia and industry, so we add the comparative experiments between SMOTE and SSDFD. Due to space limitations, Figures 14 and 15 only show the comparison of precision and recall for HDD A and HDD B with 30-fold generated data based on SSDFD and SMOTE, respectively. As shown in Figure 14, the precision and recall performance of ML algorithms based on SSDFD are all better than SMOTE under the condition of 100-900 original failed disk data of HDD A. The average precision of SSDFD is about 20% higher than SMOTE, and the average recall is about 15% higher. A similar phenomenon can be observed in Figures 15(a) and 14(a). For the precision of HDD B, under the condition of 100-900 original failed disk data, the performance of ML algorithms based on SSDFD is better than that based on SMOTE, and the average precision of SSDFD is about 13% higher than SMOTE. However, Figure 15(b) shows different trends from that of Figure 14(b). The SSDFD outperforms SMOTE when the original failed disk data volume is in the range 100-500, and SMOTE outperforms SSDFD when the original failed disk data volume is 700-900, but the average recall of SMOTE is only 3% higher than that of SSDFD. In general, SSDFD outperforms SMOTE in most test cases for different datasets, which fully demonstrates the effectiveness of SSDFD.

6.3 Evaluations Compared to Ensemble Learning Algorithms

Ensemble learning is a common solution to solve the imbalanced learning problem, it can also improve the accuracy of classification. In view of this, we also compared the effects of SSDFD

Fig. 16. The (a) precision and (b) recall of HDD A compared with Bagging and AdaBoost.

Fig. 17. The (a) precision and (b) recall of HDD B compared with Bagging and AdaBoost.

with Bagging [4] and AdaBoost [11] in the experiment. Bagging uses DT as the base estimator, the number of base estimators is 10, the resample strategy is only resampling the minority class. AdaBoost uses DT as the base estimator, the number of base estimators is 50, the boosting algorithm is “SAMME.R”. Due to space limitations, we only select MLP which performs the worst in fault detection and the RF, which performs the best to compare with Bagging and AdaBoost under the condition of different amounts of original failed disk data with 30-fold generated failed disk data added. As shown in Figure 16, for the precision of HDD A, MLP with the lowest average accuracy surpasses Bagging and AdaBoost in different original data volumes, while RF shows the better performance. For the recall of HDD A, when the original data volume is 100, the performance of Bagging and AdaBoost outperforms MLP and RF based on SSDFD. However, with the increase of the amount of original data, the performance of MLP and RF begin to rise. MLP almost outperforms Bagging and AdaBoost in the case of 300–900 data volume, and RF also shows better effect. In Figure 17, the precision of Bagging and AdaBoost is close to MLP and RF in the case of 700 and 900 original data volumes, while both precision and recall are lower than MLP and RF in the case of other original data volumes, and the overall trend is similar to that of HDD A. Overall, Bagging and AdaBoost are better than ML algorithms based on SSDFD only under a few conditions, but the MLP which performs the worst in fault detection among the five ML algorithms even has better precision and recall than Bagging and AdaBoost in most conditions, which fully demonstrates the superiority of SSDFD.

7 CONCLUSIONS

To solve the problem of low fault detection accuracy due to the lack of failed disk data, a small sample disk fault detection optimizing method SSDFD based on LSTM-GAN is proposed in this article. By learning the characteristics of the original failed disk data, the proposed method generates

virtual failed disk data and adds noise to simulate the impact of the real environment on the data. The datasets used in this article are two HDD datasets published by the industry. The experimental results show that the method has good universality. Under different operating conditions, the total amount of data can be increased by generating virtual failed disk data for different types of HDDs. To improve the detection accuracy of different ML algorithms, the problem of small sample disk fault detection is effectively solved.

Moreover, the experimental results show that for HDDs, training the model with 300 original failed disk data has a significant effect on the accuracy of fault detection. Additionally, when the amount of generated failed disk data is 20–30 times that of the original data, the improvements in the precision and recall rates achieved by the proposed method for the fault detection process using various ML algorithms are close to saturation, and the effect of continuously generating additional virtual failed disk data has a limited effect on disk fault detection. Therefore, the optimal amount of data to generate is 20–30 times that of the original failed disk data. In future work, we intend to use multi-nodes for distributed training and accelerate the training process of the proposed model leveraging data parallelism or model parallelism.

ACKNOWLEDGMENTS

Yufei Wang would like to thank Xiaoshe Dong, Xingjun Zhang, Longxiang Wang and Weiguo Wu, for their valuable advice and reviews of this paper.

REFERENCES

- [1] Bruce Allen. 2004. Monitoring hard disks with smart. *Linux Journal* 117 (2004), 74–77.
- [2] Backblaze. 2020. The Backblaze Hard Drive Data and Stats. Retrieved October 20, 2020 from <https://www.backblaze.com/b2/hard-drive-test-data.html>.
- [3] Mirela Madalina Botezatu, Ioana Giurgiu, Jasmina Bogojeska, and Dorothea Wiesmann. 2016. Predicting disk replacement towards reliable data centers. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.), ACM, 39–48. DOI : <https://doi.org/10.1145/2939672.2939699>
- [4] Leo Breiman. 1996. Bagging predictors. *Machine Learning* 24, 2 (1996), 123–140.
- [5] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, Jaidev Haridas, Chakravarthy Uddaraju, Hemal Khatri, Andrew Edwards, Vaman Bedekar, Shane Mainali, Rafay Abbasi, Arpit Agarwal, Mian Fahim ul Haq, Muhammad Ikram ul Haq, Deepali Bhardwaj, Sowmya Dayanand, Anitha Adusumilli, Marvin McNett, Sriram Sankaran, Kavitha Manivannan, and Leonidas Rigas. 2011. Windows azure storage: A highly available cloud storage service with strong consistency. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles 2011*. Ted Wobber and Peter Druschel (Eds.), ACM, 143–157. DOI : <https://doi.org/10.1145/2043556.2043571>
- [6] Lixiao Cao, Jingyi Zhang, Jing-Yue Wang, and Zheng Qian. 2019. Intelligent fault diagnosis of wind turbine gearbox based on Long short-term memory networks. *2019 IEEE 28th International Symposium on Industrial Electronics*, IEEE, 890–895. DOI : <https://doi.org/10.1109/ISIE.2019.8781108>
- [7] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16 (2002), 321–357.
- [8] Longting Chen, Guanghua Xu, Sicong Zhang, Jiachen Kuang, and Long Hao. 2019. Transfer learning for electrocardiogram classification under small dataset. In *Proceedings of the Machine Learning and Medical Engineering for Cardiovascular Health and Intravascular Imaging and Computer Assisted Stenting - First International Workshop, MLMECH 2019, and 8th Joint International Workshop, CVII-STENT 2019, Held in Conjunction with MICCAI 2019, Proceedings Lecture Notes in Computer Science*, Vol. 11794. Springer, 45–54. DOI : https://doi.org/10.1007/978-3-030-33327-0_6
- [9] Alibaba Cloud. 2018. Pangu—the High Performance Distributed File System by Alibaba Cloud. Retrieved October 10, 2018 from https://www.alibabacloud.com/blog/pangu_the_high_performance_distributed_file_system_by_alibaba_cloud_594059.
- [10] Fernando Dione dos Santos Lima, Gabriel Maia Rocha Amaral, Lucas Goncalves de Moura Leite, João Paulo Pordeus Gomes, and Javam de Castro Machado. 2017. Predicting failures in hard drives with LSTM networks. In *Proceedings of the 2017 Brazilian Conference on Intelligent Systems, BRACIS 2017*. IEEE Computer Society, 222–227. DOI : <https://doi.org/10.1109/BRACIS.2017.72>

- [11] Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55, 1 (1997), 119–139. DOI : <https://doi.org/10.1006/jcss.1997.1504>
- [12] Sandipan Ganguly, Ashish Consul, Ali Khan, Brian Bussone, Jacqueline Richards, and Alejandro Miguel. 2016. A practical approach to hard disk failure prediction in cloud platforms: Big data model for failure management in datacenters. In *Proceedings of the 2nd IEEE International Conference on Big Data Computing Service and Applications, BigDataService 2016*. IEEE Computer Society, 105–116. DOI : <https://doi.org/10.1109/BigDataService.2016.10>
- [13] Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural Computation* 12, 10 (2000), 2451–2471. DOI : <https://doi.org/10.1162/089976600300015015>
- [14] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles 2003.*, Michael L. Scott and Larry L. Peterson (Eds.), ACM, 29–43. DOI : <https://doi.org/10.1145/945445.945450>
- [15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (Eds.), 2672–2680. <http://papers.nips.cc/paper/5423-generative-adversarial-nets>.
- [16] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2017. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems* 28, 10 (2017), 2222–2232. DOI : <https://doi.org/10.1109/TNNLS.2016.2582924>
- [17] Haryadi S. Gunawi, Riza O. Suminto, Russell Sears, Casey Golliher, Swaminathan Sundararaman, Xing Lin, Tim Emami, Weiguang Sheng, Nematollah Bidokhti, Caitie McCaffrey, Deepthi Srinivasan, Biswaranjan Panda, Andrew Baptist, Gary Grider, Parks M. Fields, Kevin Harms, Robert B. Ross, Andree Jacobson, Robert Ricci, Kirk Webb, Peter Alvaro, H. Birali Runesha, Mingzhe Hao, and Huaicheng Li. 2018. Fail-slow at scale: Evidence of hardware performance faults in large production systems. *ACM Transactions on Storage* 14, 3 (2018), 23:1–23:26. DOI : <https://doi.org/10.1145/3242086>
- [18] Jiayi Guo, Bin Lei, Chibiao Ding, and Yueting Zhang. 2017. Synthetic aperture radar image synthesis by using generative adversarial nets. *IEEE Geoscience and Remote Sensing Letters* 14, 7 (2017), 1111–1115. DOI : <https://doi.org/10.1109/LGRS.2017.2699196>
- [19] Fangfang Han, Linkai Yan, Junxin Chen, Yueyang Teng, Shuo Chen, Shouliang Qi, Wei Qian, Jie Yang, William Moore, Shu Zhang, and Zhengrong Liang. 2020. Predicting unnecessary nodule biopsies from a small, unbalanced, and pathologically proven dataset by transfer learning. *Journal of Digital Imaging* 33, 3 (2020), 685–696. DOI : <https://doi.org/10.1007/s10278-019-00306-z>
- [20] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *Proceedings of the International Conference on Intelligent Computing*. Springer, 878–887.
- [21] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the International Joint Conference on Neural Networks, part of the IEEE World Congress on Computational Intelligence*. IEEE, 1322–1328. DOI : <https://doi.org/10.1109/IJCNN.2008.4633969>
- [22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780. DOI : <https://doi.org/10.1162/neco.1997.9.8.1735>
- [23] Lihan Hu, Lixin Han, Zhenyuan Xu, Tianming Jiang, and Huijun Qi. 2020. A disk failure prediction method based on LSTM network due to its individual specificity. In *Proceedings of the Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES-2020*, Procedia Computer Science, Vol. 176, 791–799. DOI : <https://doi.org/10.1016/j.procs.2020.09.074>
- [24] Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, and Sergey Yekhanin. 2012. Erasure coding in windows azure storage. In *Proceedings of the 2012 USENIX Annual Technical Conference*. Gernot Heiser and Wilson C. Hsieh (Eds.), USENIX Association, 15–26. Retrieved from <https://www.usenix.org/conference/atc12/technical-sessions/presentation/huang>.
- [25] Luyang Jing, Ming Zhao, Pin Li, and Xiaoqiang Xu. 2017. A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox. *Measurement* 111 (2017), 1–10.
- [26] Jing Li, Xinpui Ji, Yuhua Jia, Bingpeng Zhu, Gang Wang, Zhongwei Li, and Xiaoguang Liu. 2014. Hard drive failure prediction using classification and regression trees. In *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, IEEE Computer Society, 383–394. DOI : <https://doi.org/10.1109/DSN.2014.44>
- [27] Jie Li, Katherine A. Skinner, Ryan M. Eustice, and Matthew Johnson-Roberson. 2018. WaterGAN: Unsupervised generative network to enable real-time color correction of monocular underwater images. *IEEE Robotics and Automation Letters* 3, 1 (2018), 387–394. DOI : <https://doi.org/10.1109/LRA.2017.2730363>
- [28] Jing Li, Rebecca J. Stones, Gang Wang, Xiaoguang Liu, Zhongwei Li, and Ming Xu. 2017. Hard drive failure prediction using decision trees. *Reliability Engineering & System Safety* 164 (2017), 55–65. DOI : <https://doi.org/10.1016/j.res.2017.03.004>

- [29] Xiaojian Li, Liyu Zhu, Cuiping Zhang, Haopeng Yang, Hailan Wang, and Jiajia Zhang. 2021. Failure prediction for temporal dependency of hard drives. In *Proceedings of the 11th International Workshop on Computer Science and Engineering*. 1–10. DOI : <https://doi.org/10.18178/wcse.2021.06.054>
- [30] Joseph F. Murray, Gordon F. Hughes, and Kenneth Kreutz-Delgado. 2005. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research* 6 (2005), 783–816. Retrieved from <http://jmlr.org/papers/v6/murray05a.html>.
- [31] Rahul Nandgave and Dr. A. R. Buchade. 2021. Predictive maintenance of storage systems using LSTM networks. *International Journal of Current Engineering and Technology* Special Issue-8 (2021), 988–991.
- [32] Iyswarya Narayanan, Di Wang, Myeongjae Jeon, Bikash Sharma, Laura Caulfield, Anand Sivasubramaniam, Ben Cutler, Jie Liu, Badriddine M. Khessib, and Kushagra Vaid. 2016. SSD failures in datacenters: What? When? and Why?. In *Proceedings of the 9th ACM International on Systems and Storage Conference, SYSTOR 2016*, ACM, 7:1–7:11. DOI : <https://doi.org/10.1145/2928275.2928278>
- [33] Mayur R. Palankar, Adriana Iamnitchi, Matei Ripeanu, and Simson Garfinkel. 2008. Amazon S3 for science grids: A viable solution?. In *Proceedings of the 2008 International Workshop on Data-Aware Distributed Computing*. Association for Computing Machinery, 55–64. DOI : <https://doi.org/10.1145/1383519.1383526>
- [34] Yuki Saito, Shinnosuke Takamichi, and Hiroshi Saruwatari. 2018. Statistical parametric speech synthesis incorporating generative adversarial networks. *IEEE/ACM Transactions on Audio, Speech and Language Processing* 26, 1 (2018), 84–96. DOI : <https://doi.org/10.1109/TASLP.2017.2761547>
- [35] Bianca Schroeder and Garth A. Gibson. 2007. Disk failures in the real world: What does an MTTF of 1, 000, 000 hours mean to you?. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies*. Andrea C. Arpaci-Dusseau and Remzi H. Arpaci-Dusseau (Eds.), USENIX, 1–16. Retrieved from <http://www.usenix.org/events/fast07/tech/schroeder.html>.
- [36] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. 2016. Flash reliability in production: The expected and the unexpected. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies*. Angela Demke Brown and Florentina I. Popovici (Eds.), USENIX Association, 67–80. Retrieved from <https://www.usenix.org/conference/fast16/technical-sessions/presentation/schroeder>.
- [37] Haidong Shao, Hongkai Jiang, Xun Zhang, and Maogui Niu. 2015. Rolling bearing fault diagnosis using an optimization deep belief network. *Measurement Science and Technology* 26, 11 (2015), 115002.
- [38] Muralidhar Subramanian, Wyatt Lloyd, Sabyasachi Roy, Cory Hill, Ernest Lin, Weiwen Liu, Satadru Pan, Shiva Shankar, Sivakumar Viswanathan, Linpeng Tang, and Sanjeev Kumar. 2014. f4: Facebook’s warm BLOB storage system. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation*. Jason Flinn and Hank Levy (Eds.), USENIX Association, 383–398. Retrieved from <https://www.usenix.org/conference/osdi14/technical-sessions/presentation/muralidhar>.
- [39] Dennis L. Wilson. 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics* 3 (1972), 408–421.
- [40] Jiang Xiao, Zhuang Xiong, Song Wu, Yusheng Yi, Hai Jin, and Kan Hu. 2018. Disk failure prediction in data centers via online learning. In *Proceedings of the 47th International Conference on Parallel Processing*. ACM, 35:1–35:10. DOI : <https://doi.org/10.1145/3225058.3225106>
- [41] Yong Xu, Kaixin Sui, Randolph Yao, Hongyu Zhang, Qingwei Lin, Yingnong Dang, Peng Li, Keceng Jiang, Wenchi Zhang, Jianguang Lou, Murali Chintalapati, and Dongmei Zhang. 2018. Improving service availability of cloud systems by predicting disk error. In *2018 USENIX Annual Technical Conference*. Haryadi S. Gunawi and Benjamin Reed (Eds.), USENIX Association, 481–494. Retrieved from <https://www.usenix.org/conference/atc18/presentation/xu-yong>.
- [42] Ji Zhang, Yuanzhang Wang, Yangtao Wang, Ke Zhou, Sebastian Schelter, Ping Huang, Bin Cheng, and Yongguang Ji. 2020. Tier-scrubbing: An adaptive and tiered disk scrubbing scheme with improved MTTD and reduced cost. In *Proceedings of the 57th ACM/IEEE Design Automation Conference*. IEEE, 1–6. DOI : <https://doi.org/10.1109/DAC18072.2020.9218551>
- [43] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing* 26, 7 (2017), 3142–3155. DOI : <https://doi.org/10.1109/TIP.2017.2662206>
- [44] Bingpeng Zhu, Gang Wang, Xiaoguang Liu, Dianming Hu, Sheng Lin, and Jingwei Ma. 2013. Proactive drive failure prediction for large scale storage systems. In *IEEE 29th Symposium on Mass Storage Systems and Technologies*. IEEE Computer Society, 1–5. DOI : <https://doi.org/10.1109/MSST.2013.6558427>

Received April 2021; revised September 2021; accepted November 2021