NTAM: Neighborhood-Temporal Attention Model for Disk Failure Prediction in Cloud Platforms

Chuan Luo*
Microsoft Research
Beijing, China

Pu Zhao*
Microsoft Research
Beijing, China

Bo Qiao
Microsoft Research
Beijing, China

Youjiang Wu
Microsoft Azure
Redmond, United States

Hongyu Zhang
The University of Newcastle
Newcastle, Australia

Wei Wu
Leibniz University Hannover
Hannover, Germany

Weihai Lu
Microsoft Research
Beijing, China

Yingnong Dang
Microsoft Azure
Redmond, United States

Saravanakumar Rajmohan
Microsoft 365
Redmond, United States

Qingwei Lin
Microsoft Research
Beijing, China

Dongmei Zhang
Microsoft Research
Beijing, China

ABSTRACT

With the rapid deployment of cloud platforms, high service reliability is of critical importance. An industrial cloud platform contains a huge number of disks, and disk failure is a common cause of service unreliability. In recent years, many machine learning based disk failure prediction approaches have been proposed, and they can predict disk failures based on disk status data before the failures actually happen. In this way, proactive actions can be taken in advance to improve service reliability. However, existing approaches treat each disk individually and do not explore the influence of the neighboring disks. In this paper, we propose Neighborhood-Temporal Attention Model (*NTAM*), a novel deep learning based approach to disk failure prediction. When predicting whether or not a disk will fail in near future, *NTAM* is a novel approach that not only utilizes a disk's own status data, but also considers its neighbors' status data. Moreover, *NTAM* includes a novel attention-based temporal component to capture the temporal nature of the disk status data. Besides, we propose a data enhancement method, called Temporal Progressive Sampling (*TPS*), to handle the extreme data imbalance issue. We evaluate *NTAM* on a public dataset as well as two industrial datasets collected from millions of disks in Microsoft Azure. Our experimental results show that *NTAM* significantly outperforms state-of-the-art competitors. Also, our empirical evaluations indicate the effectiveness of the neighborhood-ware component and the temporal component underlying *NTAM* as well as the effectiveness of *TPS*. More encouragingly, we have successfully applied *NTAM* and *TPS* to Microsoft cloud platforms (including Microsoft Azure and Microsoft 365) and obtained benefits in industrial practice.

*Chuan Luo and Pu Zhao contributed equally to this work and share the first authorship of this work. Qingwei Lin is the corresponding author (Email: qlin@microsoft.com).

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449867

CCS CONCEPTS

• **Hardware** → **Failure prediction**; • **Computer systems organization** → **Cloud computing**; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Disk Failure Prediction, Cloud Platforms, High Service Reliability, Neighborhood-Temporal Attention Model, Data Imbalance

ACM Reference Format:

Chuan Luo, Pu Zhao, Bo Qiao, Youjiang Wu, Hongyu Zhang, Wei Wu, Weihai Lu, Yingnong Dang, Saravanakumar Rajmohan, Qingwei Lin, and Dongmei Zhang. 2021. *NTAM*: Neighborhood-Temporal Attention Model for Disk Failure Prediction in Cloud Platforms. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3442381.3449867

1 INTRODUCTION

In recent years, plenty of software systems have been migrated to cloud platforms (such as Amazon Web Service, Microsoft Azure, and Google Could Platform) and deployed as online services [2, 9, 10, 23, 24, 40]. As cloud platforms are required to serve customer workloads on a 24/7 basis, high service reliability is extremely critical [5, 17]. A typical industrial cloud platform like Microsoft Azure uses a huge number of hard disk drives [44]. It has been found that disk failure is one of the most frequently failing component among IT equipment failures [3, 32] and has become one of the most important factors that contribute to the service downtime [28, 30, 37]. Service downtime can adversely affect customer experience and even cause huge financial loss [21, 26]. For example, it has been found that every minute of downtime costs about $9,000 [12].

In order to minimize the impact caused by disk failures, over the years many approaches [3, 11, 19, 35, 37, 42, 44, 46, 49, 50] have been proposed to predict disk failures before they actually happen.

These approaches predict disk failures mainly based on disks' internal status data, which is also called SMART (Self-Monitoring, Analysis, and Reporting Technology) data [1]. The SMART data records important safety indicators at the lifetime of a disk, and is hardware-level sensor data provided by firmware embedded in disk drives [37]. Furthermore, a recent study [44] also found that using system signals can empower the disk failure prediction task to achieve better performance. The problem of disk failure prediction is usually treated as a binary classification problem in machine learning: given data about a disk, predict whether this disk will fail or not in near future. If a disk is predicted to fail, proactive actions, such as replacement of the disk and live migration [27], can then be taken in time. The cloud platform can also transfer important data from failure-prone disks to the healthy ones in advance. In this way, the reliability of cloud platform could be improved.

In current practice, existing disk failure prediction approaches treat each disk individually and only consider each disk's own status data. However, it is well recognized that, in large-scale cloud platforms, a number of disks are installed in the same computing server. For each disk in a server, all the other disks in the server can be regarded as its neighbors. Since the computing environment is shared among neighboring disks, they would have similar failure patterns. Moreover, the neighboring disks would work together and interact with each other when completing a variety of computation and storage tasks on the server. As a result, the status data of those neighboring disks placed in the same computing server is strongly related, which can be utilized for improving the practical performance of disk failure prediction.

Furthermore, in large-scale cloud platforms, it is apparent that the number of healthy disks is much greater than that of the failed ones, which causes the extreme data imbalance problem for disk failure prediction. In order to mitigate this problem, existing approaches [3, 35] usually adopt under-sampling methods to select a subset of healthy disks rather than using the whole set of healthy disks in the training process. However, under-sampling methods could drop some useful information about the healthy disks, which would degrade the prediction performance in practice.

In this paper, we propose a novel deep learning based approach for disk failure prediction, dubbed Neighborhood-Temporal Attention Model (*NTAM*). Our *NTAM* approach includes two new components, *i.e.,* the neighborhood-aware component and the temporal component. Compared to existing approaches which only use a disk's own status data, the neighborhood-aware component underlying *NTAM* also takes neighborhood information into consideration, *i.e.,* encoding the status data of that disk's neighbors by a soft attention mechanism [43]. Furthermore, in contrast to existing approaches, *NTAM* can better capture the temporal information through an attention-based component (i.e., the temporal component).

In order to deal with the extreme data imbalance problem, we also propose a general and effective method named Temporal Progressive Sampling (*TPS*). Our *TPS* method can be treated as a data enhancement method, and is able to generate multiple failed samples for each failed disk. Compared to under-sampling methods utilized by existing disk failure prediction approaches, the advantage of *TPS* is that *TPS* not only retains all the characteristics of healthy disks, but also brings more failure patterns. In this way, *TPS* can help push forward the state of the art in disk failure prediction.

To evaluate the effectiveness of our proposed *NTAM* approach, we conduct extensive experiments to compare *NTAM* against 10 state-of-the-art disk failure prediction approaches on two industrial datasets; both industrial datasets include the status data of millions of disks and are collected from Microsoft Azure, which serves huge amount of customer workloads. The experimental results on both industrial datasets present that *NTAM* significantly outperforms all its competitors, which indicates that *NTAM* considerably advances the state of the art in disk failure prediction. Further experiments on a public dataset demonstrate the robustness of *NTAM*. More encouragingly, *NTAM* and *TPS* have been successfully applied to Microsoft cloud platforms (including Microsoft Azure and Microsoft 365), and improved the reliability of Microsoft cloud platforms.

The main contributions of this paper are as follows:

First, we propose a neighborhood-temporal attention model based approach dubbed *NTAM*, which is a novel approach for disk failure prediction. Through the neighborhood-aware component, our proposed *NTAM* approach utilizes not only the disk's own status data, but also considers the status data of its neighbors. Extensive experiments on industrial datasets show that *NTAM* considerably advances the state of the art in disk failure prediction.

Second, besides the neighborhood-aware component, *NTAM* also incorporates a novel temporal component to capture the temporal nature of the disk status data. Our extensive empirical evaluations present that the temporal component underlying *NTAM* performs much better than existing LSTM and temporal CNN based methods, which indicates the effectiveness of the temporal component underlying *NTAM*.

Finally, we propose a general and effective method called *TPS* to deal with the extreme data imbalance problem in disk failure prediction. Our experimental results clearly demonstrate that *TPS* is a general method for handling the extreme data imbalance problem and can consistently improve the practical performance of various disk failure prediction approaches.

The remainder of this paper is structured as follows. Section 2 summarizes the related work for disk failure prediction. Section 3 proposes *NTAM* for disk failure prediction and *TPS* for handling the extreme data imbalance problem. Section 4 reports and analyzes experimental results to demonstrate the effectiveness of *NTAM* and *TPS*. Section 5 introduces the application of *NTAM* and *TPS* in practice. We conclude this paper in Section 6.

2 RELATED WORK

Because of the importance of disk failures, many approaches have been proposed for disk failure prediction. Existing approaches mainly treat disk failure prediction as a binary classification problem in the area of machine learning. These approaches can be categorized into two classes: traditional machine learning based ones and deep learning based ones.

Traditional machine learning based approaches predict disk failures based on SMART data using support vector machine [46] and tree-based machine learning models [3, 11, 19, 35, 44]. In real-world applications, disks usually fail gradually rather than abruptly [46]. Nevertheless, it is difficult for traditional machine learning based approaches to process the temporal information effectively [37], so their performance is relatively moderate on real-world datasets.

In contrast to traditional machine learning based approaches, deep learning based approaches leverage deep neural networks, including recurrent neural network (RNN) [42], long short-term memory (LSTM) [46] and temporal convolutional neural network (TCNN) [37], and thus are able to make better use of the temporal information. Hence, deep learning based approaches can generally achieve performance improvement over traditional machine learning based ones for disk failure prediction. In industrial practice, a number of disks are located in the same computing server and interact with each other during daily usage. Hence, the status of neighboring disks is highly correlated. However, in previous works on disk failure prediction, existing deep learning based approaches do not consider any neighborhood information, therefore using only limited information, which is an issue in our perspective and would degrade the practical performance (as evidenced by our experimental results in Section 4).

Compared to existing approaches for disk failure prediction, our *NTAM* approach introduces a novel neighborhood-aware component to incorporate the neighborhood information. Moreover, *NTAM* integrates a new temporal component to capture the temporal information, to further improve the practical performance.

In practice, disk failure prediction approaches suffer from the extreme data imbalance problem [3, 11, 35, 37, 44, 46], since the number of failed disks is greatly smaller than that of healthy disks. To handle the extreme data imbalance problem in disk failure prediction, one common solution is to use under-sampling methods, such as clustering-based sampling method [3] and latest sampling method [35]. In particular, clustering-based sampling method [3] partitions healthy samples in the training set into a number of groups and uses the center sample of each group to represent the whole group, while latest sampling method [35] uses the latest samples in the training set of each healthy disk. However, such under-sampling methods would lose some characteristics of healthy disk samples. Another type of solution to handling the extreme imbalance data problem is the cost-sensitive method, including weight adjusting based one [19] and new loss function based one [37]. Particularly, weight adjusting based cost-sensitive method [19] changes the probability distributions of healthy and failed samples by adjusting their weights, while new loss function based cost-sensitive method [37] derives a new loss function by multiplying the basic binary cross-entropy with different coefficients to address the data imbalance problem.

Compared to existing methods for handling the imbalance data problem, our proposed *TPS* method is an effective data enhancement method, which can generate multiple failed samples for each failed disk. More particularly, *TPS* is able to deal with the extreme data imbalance and can improve the practical performance of various approaches for disk failure prediction.

3 OUR PROPOSED APPROACH

In this section, we present our proposed approach in detail. First, we introduce the problem definition of disk failure prediction. Then, we give the overview of our proposed approach dubbed Neighborhood-Temporal Attention Model (*NTAM*). After that, we describe the technical details of *NTAM*. Finally, we present an effective data enhancement method called Temporal Progressive Sampling (*TPS*), which is able to improve the practical performance of *NTAM*.

3.1 Problem Definition

This paper is devoted to proposing a deep learning based approach, which can predict whether a disk will fail or not, based on that disk's status data and neighborhood information. Before introducing the problem, we first give the definitions of a disk's status data and neighborhood information. In practice, a *feature vector* of n attributes of a disk's status is recorded at each timestamp (*e.g.,* hourly or daily). For a disk d_i, d_i's *status data* is a set consisting of d_i's h consecutive feature vectors recorded from timestamp t_i to timestamp t_i+h-1 (t_i is the beginning timestamp). As introduced before, in cloud platforms, a computing server usually contains more than one disk; hence, two different disks are neighbors when they are located in the same computing server. For a disk d_i, d_i's *neighborhood information* is a set of the status data of d_i's all neighbors.

The training set is a collection of N training samples, and is denoted as $D = \{(X_1, y_1), \dots, (X_N, y_N)\}$. For each training sample (X_i, y_i), X_i represents the corresponding disk d_i's status data and neighborhood information, *i.e.,* $X_i = (A_i, B_i)$, where $A_i \in \mathbb{R}^{h\times n}$ represents d_i's status data and B_i is denoted as d_i's neighborhood information (it is clear that B_i is a subset of unions of all A_i, *i.e.,* $B_i \subseteq \cup_{i=1}^{N} A_i$); $y_i \in \{0, 1\}$ is a label: $y_i = 1$ means that the corresponding disk will fail in near future, and $y_i = 0$ means 'healthy'. Hence, our objective is to minimize the *binary cross-entropy loss* [29] on the training set. The binary cross-entropy loss $\mathcal{L}$ is formally formulated as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot \log\left(\hat{y}_i\right) + (1-y_i)\cdot \log\left(1-\hat{y}_i\right)\right] \qquad (1)$$

where $\hat{y}_i$ is the predicted probability of the sample being positive (*i.e.,* predicted failure probability) for sample (X_i, y_i).

3.2 Overview of *NTAM*

As briefly introduced before, *NTAM* can predict disk failures by exploiting the neighborhood information and making the better use of the temporal information. The main technical challenges are as follows: 1) how to effectively incorporate neighborhood information and 2) how to better capture the temporal information.

The overview of our *NTAM* approach is illustrated in Figure 1. According to Figure 1, *NTAM* consists of three components, *i.e.,* the neighborhood-aware component, the temporal component and the decision component. We briefly overview each component as follows, and introduce the technical details of all components in the following subsections.

- **Neighborhood-aware component:** the input of this component consists of two parts, *i.e.,* the status data (A_i) of the corresponding disk d_i, and d_i's neighborhood information (B_i). This component utilizes a soft attention mechanism [43] to encode the neighborhood information B_i based on A_i, and then fuse the status data and the encoded neighborhood information together, resulting in a set of neighbor-encoded vectors.

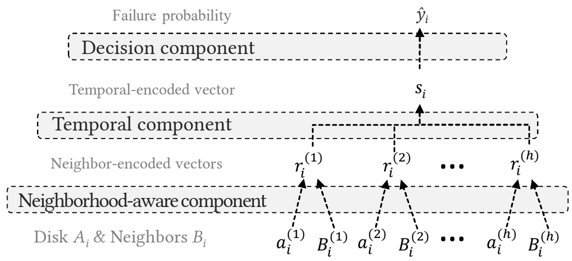

Figure 1: The overview of Neighborhood-aware Attention Model (*NTAM*).

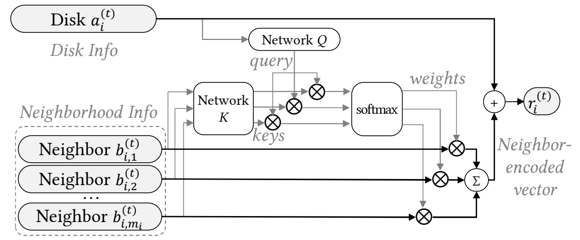

Figure 2: The architecture of the neighborhood-aware component underlying *NTAM*.

- **Temporal component:** this component takes the set of neighbor-encoded vectors generated by the neighborhood-aware component as its input. It introduces and utilizes a proposed, novel attention based neural network to encode and incorporate the temporal information, resulting in a temporal-encoded vector.
- **Decision component:** this component takes the temporal-encoded vector generated by the temporal component as its input, and calculates the failure probability of the corresponding disk. Then the decision component utilizes the failure probability to decide whether the corresponding disk will fail or not.

NTAM is an end-to-end deep learning based approach, and minimizes the binary cross-entropy loss (Equation 1) using the Adam optimizer. The model parameters of *NTAM* are updated through the back-propagation algorithm during the training process.

3.3 Neighborhood-aware Component

In contrast to existing approaches which only utilize a disk's own status data, *NTAM* introduces a novel neighborhood-aware component that can encode and incorporate the neighborhood information. The architecture of the neighborhood component underlying *NTAM* is illustrated in Figure 2. The input of the neighborhood component includes a disk d_i's status data A_i and its neighborhood information B_i. Recall that A_i is a set of h feature vectors (h is the number of timestamps), *i.e.,* $A_i = \{a_i^{(1)}, \ldots, a_i^{(h)}\}$, where $a_i^{(t)}$ denotes d_i's feature vector at timestamp $t_i + t - 1$. Also, recall that B_i is a set of status data of d_i's all neighbors, *i.e.,* $B_i = \{B_{i,1}, \ldots, B_{i,m_i}\}$, where $m_i = |B_i|$ represents the number of d_i's neighbors, and $B_{i,j} \in \mathbb{R}^{h \times n}$ represents the status data of d_i's j-th neighbor; in fact, $B_{i,j}$ can be expressed as $B_{i,j} = \{b_{i,j}^{(1)}, \ldots, b_{i,j}^{(h)}\}$, where $b_{i,j}^{(t)}$ denotes the feature vector of d_i's j-th neighbor at timestamp $t_i + t - 1$. Also, $B_i^{(t)}$ is denoted as $B_i^{(t)} = \{b_{i,1}^{(t)}, \ldots, b_{i,m_i}^{(t)}\}$ which represents d_i's neighborhood information at timestamp $t_i + t - 1$.

It is arguable that, for a disk d_i, each neighbor of d_i has different impact on d_i's healthy status. As a result, it is advisable to design an effective mechanism to distinguish the influence of each neighbor. Hence, we leverage a soft attention mechanism [43] to capture the effect of each neighbor on the corresponding disk's healthy status, as shown in Figure 2.

Actually, our neighborhood-aware component will activate the soft attention mechanism for the input at each timestamp (*i.e.,* $a_i^{(t)}$ and $B_i^{(t)}$). An attention function is to obtain an output, on the basis of a query and a collection of key-value tuples (the query, keys, values and the output are all vectors) [39]. The output vector is a weighted accumulation of value vectors, where each value vector's weight is calculated based on the query vector and the corresponding key vector. In the neighboring-aware component underlying *NTAM*, the query vector is $q = Q(a_i^{(t)})$, the key vector for j-th neighbor is $k_j = K(b_{i,j}^{(t)})$, and the value vector for j-th neighbor is $b_{i,j}^{(t)}$, where both Q and K are fully connected networks. The weight w_j associated to each value vector is computed as follows, using the *softmax* function:

$$w_j = \frac{\exp(q \cdot k_j)}{\sum_{z=1}^{m_i} \exp(q \cdot k_z)} \tag{2}$$

Then the weighed accumulation of the neighborhood information at each timestamp can be represented as $c_i^{(t)} = \sum_{j=1}^{m_i} (w_j \cdot b_{i,j}^{(t)})$, where $c_i^{(t)} \in \mathbb{R}^n$.

Finally, for each timestamp $t_i + t - 1$, we can construct a neighbor-encoded vector $r_i^{(t)}$ for disk d_i based on d_i's own feature vector (*i.e.,* $a_i^{(t)}$) and the weighted accumulation of d_i's neighborhood information (*i.e.,* $c_i^{(t)}$): $r_i^{(t)} = a_i^{(t)} + c_i^{(t)}$, where $r_i^{(t)} \in \mathbb{R}^n$. In this way, since the neighbor-encoded vector $r_i^{(t)}$ incorporates the neighborhood information, $r_i^{(t)}$ is more informative than the original feature vector $a_i^{(t)}$; using the neighbor-encoded vector could achieve performance improvement.

In practice, the number of neighbors varies from one disk to another, and it is difficult for deep learning models to deal with the input samples with different sizes. We use a padding mechanism [39] to address this challenge: first, we use the notation M to denote the maximum number of disk neighbors, *i.e.,* $M = \max_{i=1}^{N}\{m_i\}$; then, for each disk d_i, the dimension of B_i is padded from $m_i + 1$ to M; the newly extended neighborhood information is filled with 0, and all weights of those newly padded neighborhood information are masked out (by setting to $-\infty$) in the input of the *softmax* function (in Figure 2) due to the useless padding information.

3.4 Temporal Component

Besides incorporating the neighborhood information, our *NTAM* approach also introduces a new temporal component to capture the temporal information, since utilizing the temporal information can improve the practical performance in disk failure prediction [37, 46].

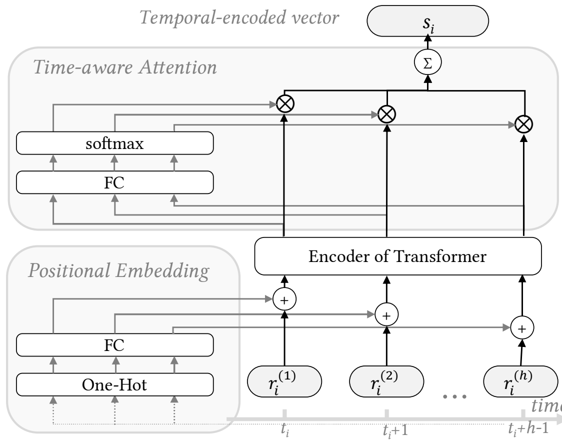

Figure 3: The architecture of the temporal component underlying *NTAM*. 'FC' refers to 'Fully Connected Network'.

In the context of disk failure prediction, different from the existing deep learning based approaches that utilize RNN [42], LSTM [46] and temporal CNN [37], the temporal component underlying *NTAM* is based on the attention mechanism.

The architecture of the temporal component underlying *NTAM* is demonstrated in Figure 3. The input of the temporal component is a sequence of neighbor-encoded vectors (ordered by their timestamps), which are the outputs of the neighborhood-aware component, *i.e.,* $R_i = \{r_i^{(1)}, \dots, r_i^{(h)}\}$. According to Figure 3, the temporal component consists of three key parts: positional embedding layer, encoder of Transformer, and time-aware attention layer. Each key part of the temporal component is described as follows.

Positional embedding layer: To leverage the order of the input sequence R_i, it is advisable to embed the position information of each element in R_i. We do so by applying the positional embedding network [7]. As indicated in Figure 3, for each element $r_i^{(t)}$ in the sequence R_i, the positional embedding network can generate a new vector (whose dimension is the same to $r_i^{(t)}$) based on $r_i^{(t)}$'s positional index and then add this new vector to $r_i^{(t)}$, resulting in ${r'}_i^{(t)}$. In this way, we can obtain $R'_i = \{{r'}_i^{(1)}, \dots, {r'}_i^{(h)}\}$.

Encoder of Transformer: In this stage, it is necessary to perform sequence modeling to encode the temporal information. It is well acknowledged that the Transformer-based models [39] achieve the state-of-the-art performance in many application scenarios of sequence modeling [31], such as natural language processing [47, 48] and speech recognition [20]. The standard Transformer is an *encoder-decoder* structure [39]. However, our task is to encode the temporal information (related to *encoder* of Transformer), but does not need to do generation (related to *decoder* of Transformer). Therefore, we adopt the *encoder* of Transformer to map R'_i to $R''_i = \{{r''}_i^{(1)}, \dots, {r''}_i^{(h)}\}$. In this way, each element in R''_i integrates information from other temporal feature vectors.

Time-aware attention layer: For a disk d_i, the impact of the feature vector on d_i's healthy status varies from one timestamp to the other. In order to better incorporate this information, we employ the position-based attention mechanism [6, 18, 41] to compute the

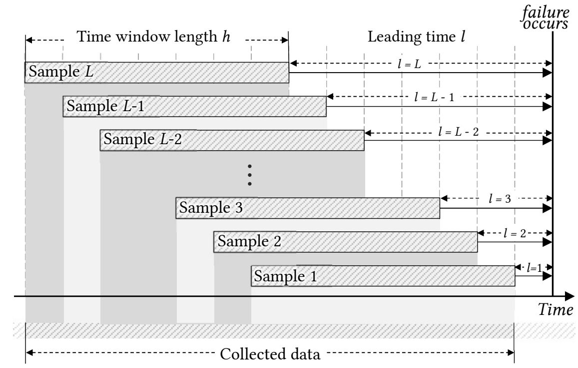

Figure 4: The design of the Temporal Progressive Sampling (*TPS*) method.

weight associated with each element ${r''}_i^{(t)}$ in R''_i. The computation of each weight uses the *softmax* function and is presented as follows:

$$v_t = \frac{\exp(FC({r''}_i^{(t)}))}{\sum_{z=1}^{h} \exp(FC({r''}_i^{(z)}))} \tag{3}$$

where *FC* is a fully connected network. Then we can construct a temporal-encoded vector $s_i = \sum_{t=1}^{h}(v_t \cdot {r''}_i^{(t)})$ for disk d_i, where $s_i \in \mathbb{R}^n$.

Remarks: Through the neighborhood-aware component and the temporal component underlying *NTAM*, the output vector (*i.e.,* temporal-encoded vector s_i) is able to incorporate both the neighborhood information and the temporal information.

3.5 Decision Component

The decision component takes the temporal-encoded vector s_i as input, which is the output of the temporal component. Based on s_i, the decision component calculates the failure probability $\hat{y}_i$ of the corresponding disk d_i via a fully connected network, where hidden layers use *ReLU* [8] as their activation functions, the output layer uses *sigmoid* as its activation function, and between each two layers, a *dropout* mechanism [36] is added to improve the robustness [14]. In the final step, *NTAM* utilizes the predicted failure probability $\hat{y}_i$ to decide whether disk d_i will fail or not.

3.6 Temporal Progressive Sampling (*TPS*)

In the context of disk failure prediction for cloud platforms, since the number of healthy disks is much greater than that of the failed disks, both traditional machine learning based approaches and deep learning based approaches suffer from the extreme data imbalance problem [16, 33, 45]. In order to address this problem, a number of under-sampling methods [3, 35], which collect less healthy samples, have been proposed. Intuitively, since there are too many healthy disks, we can sample and use a subset of those healthy disks. The under-sampling process can group the healthy disk set into several clusters through a clustering algorithm, and then select a few samples from each cluster as representatives for the respective healthy disk cluster. However, this kind of under-sampling process would lose some useful information about healthy disks, which could result in a high false alarm rate.

Table 1: Introduction to all attributes of feature vector in *Dataset-1* and *Dataset-2*.

Name	Description
Timestamp	The timestamp t of the feature vector recorded.
Disk ID	The unique ID of disk d_i.
Node ID	The unique ID of each computing server (*i.e.,* node) which current disk is on. It can be used to find all neighbors of each disk.
SMART Attributes	The SMART attributes of disk d_i recorded at timestamp t, which contains useful information such as the *Current Pending Sector Count*, *Seek Error Rate*, *Soft Read Error Rate*, *etc.*
System-related Attributes	The system-related attributes [44] include *Windows Event 154*, *the error when Windows creating a paging file.*, *etc.*
Driver-related Attributes	The driver-related attributes are gathered from disk driver, and contain *Flush Count*, *IO Latency*, *Controller Reset*, *etc.*

In order to address the extreme data imbalance issue, we propose an effective method called Temporal Progressive Sampling (*TPS*) to generate more failed samples to complement the data distribution of failed disks. Hence, *TPS* can be regarded as a data enhancement method. Through generating more failed samples by *TPS*, the ratio between the number of healthy samples and that of failed samples would achieve a better balance.

Before describing *TPS*, we would like to first introduce *leading time*, which is an important concept in *TPS*. For a given failed disk, assuming that the disk failure occurs at timestamp t and the prediction action occurs at timestamp $t - l$, then the time period with length l between the occurrence of prediction action at $t - l$ and the occurrence of the disk failure at t is denoted as the leading time l.

We illustrate the design of *TPS* in Figure 4. As shown in Figure 4, during model training, for each failed disk, *TPS* collects more failure data samples within the leading time period progressively (*i.e.,* leading time l ranges from 1 to L, where L is a hyper-parameter for *TPS*, and its effect will be analyzed and discussed in Section 4.5). In this way, *TPS* not only generates more failed samples which can help mitigate the extreme data imbalance issue, but also captures more failure patterns, which records the gradually failing process and thus can enhance the learning process of our approach.

TPS is a general data enhancement method for dealing with the extreme data imbalance issue, and is able to improve the performance of various disk failure prediction approaches. Our experimental results (shown in Section 4.5) demonstrate that *TPS* can improve the performance of various disk failure prediction approaches.

4 EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the effectiveness of our *NTAM* approach by comparing it against 10 state-of-the-art competitors for disk failure prediction. First, we introduce the competitors, the experimental setup, and the industrial datasets adopted in our experiments. Then, we report and analyze the experimental results of *NTAM* and its 10 state-of-the-art competitors, in order to demonstrate the effectiveness of our proposed *NTAM* approach. After that, additional empirical evaluations are performed to demonstrate the effectiveness of the neighborhood-aware component and the temporal component underlying *NTAM*. Also, more experiments are performed to study the effectiveness of different combinations of disk attributes. Subsequently, we conduct extensive empirical evaluations to test the performance of various disk failure prediction approaches equipped with *TPS*, so as to analyze the effectiveness of *TPS*. Finally, to confirm the effectiveness of *NTAM* and *TPS*, we perform further empirical analysis to study the performance of *NTAM* and *TPS* on a public dataset.

Table 2: Summary of *Dataset-1* and *Dataset-2*.

Dataset	Training Set		Testing Set	
	#Positive	#Negative	#Positive	#Negative
Dataset-1	5,644	6,836,491	6,768	6,768,000
Dataset-2	5,352	5,451,237	5,196	5,196,000

4.1 Competitors

In our experiments, we compare *NTAM* against 10 recent and state-of-the-art competitors, which are listed as follows and indicated in **boldface**. **Support Vector Machine (*SVM*)** [50], **Decision Tree (*DT*)** [19], **Random Forest (*RF*)** [35] and **Gradient Boosting Decision Tree (*GBDT*)** [11] are influential machine learning algorithms, and have been successfully applied to disk failure prediction. **Regularized Greedy Forest (*RGF*)** [15] is a high-performance ensemble approach, and has achieved good performance in disk failure prediction [3]. **Cloud Disk Error Forecasting (*CDEF*)** [44] is a cost-sensitive ranking approach and have already shown its success in disk failure prediction. In order to show the effectiveness of the temporal component underlying *NTAM*, we also include deep learning based approaches, which can leverage temporal information, into our comparisons. In particular, **Recurrent Neural Network (*RNN*)** [42], **Long Short-Term Memory (*LSTM*)** [46] and **Temporal Convolution Neural Network (*TCNN*)** [37] are adopted, and actually all of them have already exhibited the state-of-the-art performance in disk failure prediction. In addition, we also compare our *NTAM* approach against a recently proposed approach for disk failure prediction, called **Convolutional Neural Network with Long Short-Term Memory (*CNN+LSTM*)** [22].

4.2 Experimental Setup

Following the existing work [3, 37, 46], for the industrial dataset, we evaluate *NTAM* and its competitors by calculating their precision, recall and F1-score. In Tables 3, 4, 5, 6, 8 and 9, all competing approaches' precision, recall and F1-score are reported in percentile. The results in **boldface** indicate the best performance for the corresponding dataset. All experiments are carried out on a workstation equipped with dual 16-core 2.30 GHz Intel Xeon E5-2673 CPUs, 425 GB RAM and 8 NVIDIA Tesla M40 GPUs (with 24 GB video memory of each GPU), running the operation system of Ubuntu Linux (16.04.5).

Table 3: Comparative results of *NTAM* and its 10 state-of-the-art competitors as well as *NTAM+TPS* and all competitors equipped with *TPS* on both *Dataset-1* and *Dataset-2*. P, R, and F1 are referring to precision, recall, and F1-score, respectively.

Approach	*Dataset-1*			*Dataset-2*			Approach	*Dataset-1*			*Dataset-2*		
	P	**R**	**F1**	**P**	**R**	**F1**		**P**	**R**	**F1**	**P**	**R**	**F1**
SVM [50]	67.10	36.86	47.59	65.89	36.24	46.76	*SVM+TPS*	71.45	43.19	53.84	70.21	44.18	54.24
DT [19]	65.20	42.51	51.47	66.35	43.14	52.28	*DT+TPS*	69.16	48.98	57.36	70.13	49.02	57.71
RF [35]	68.72	43.20	53.05	70.06	43.69	53.82	*RF+TPS*	71.21	49.52	58.42	72.80	49.34	58.82
GBDT [11]	69.43	44.78	54.44	71.92	45.31	55.59	*GBDT+TPS*	72.95	51.52	60.39	73.58	51.04	60.27
RGF [3]	65.64	46.55	53.92	71.76	46.61	56.51	*RGF+TPS*	70.04	52.41	59.95	73.81	51.73	60.82
CDEF [44]	63.09	50.18	55.90	70.73	48.62	57.63	*CDEF+TPS*	69.32	53.76	60.56	72.65	52.76	61.13
RNN [42]	68.72	48.25	56.69	69.91	50.15	58.40	*RNN+TPS*	71.55	54.78	62.05	72.08	53.91	61.68
LSTM [46]	72.28	47.25	57.14	73.42	49.68	59.26	*LSTM+TPS*	74.67	54.13	62.76	75.92	53.80	62.97
TCNN [37]	73.36	47.97	58.01	72.87	51.27	60.19	*TCNN+TPS*	77.10	53.76	63.35	77.36	55.03	64.31
CNN+LSTM [22]	72.12	48.55	58.03	74.01	50.06	59.72	*CNN+LSTM+TPS*	74.98	55.26	63.62	77.14	54.97	64.20
NTAM	**78.16**	**56.28**	**65.44**	**81.07**	**57.58**	**67.34**	*NTAM+TPS*	**82.97**	**63.54**	**71.97**	**84.22**	**64.41**	**72.99**

4.3 Industrial Datasets

We collect two industrial datasets, *i.e., Dataset-1* and *Dataset-2*, from Microsoft Azure, which serves huge amount of customer workloads. Both of them are HDD (hard disk drive) datasets. Particularly, *Dataset-1* contains two-month data (July 2019 and August 2019), and similarly *Dataset-2* includes two-month data (February 2020 and March 2020). Both datasets include the status data of millions of disks over two months, and their sizes are several hundreds of times larger than the size of the public dataset in the context of disk failure prediction (the summary of the public dataset as well as the empirical analysis on the public dataset will be presented in Section 4.6). Also, in both industrial datasets, the status data of each disk d_i is recorded hourly, where each feature vector contains timestamp, disk ID, node ID, SMART attributes, system-related attributes and driver-related attributes, as illustrated in Table 1. In our work, for each disk d_i in each of our industrial datasets (*i.e., Dataset-1* and *Dataset-2*), the number of consecutive feature vectors in d_i's status data is set to $24 \times 7 = 168$ (*i.e.,* $h = 168$).

For both industrial datasets, the **failed** disks are labeled as Positive (**P**) samples, and the **healthy** disks are labeled as Negative (**N**) samples. After necessary pre-processing, we divide each industrial dataset into the training set and the testing set by time. For the *Dataset-1*, we treat the data of July 2019 as the training dataset and the data of August 2019 as the testing dataset. For the *Dataset-2*, we adopt the data of February 2020 as training dataset and the data of March 2020 as testing dataset. The number of positive samples (denoted as '#Positive') and the number of negative samples (denoted as '#Negative') for each of our industrial datasets (*i.e., Dataset-1* and *Dataset-2*) are demonstrated in Table 2. For each of our industrial datasets (*i.e., Dataset-1* and *Dataset-2*), the ratio between the number of positive samples and the number of negative samples is around 1:1,000.

4.4 Effectiveness of *NTAM*

We use both industrial datasets (*i.e., Dataset-1* and *Dataset-2* described in Section 4.3) to evaluate the practical performance of our proposed *NTAM* approach.

4.4.1 Comparisons against State-of-the-art Competitors. Table 3 presents the comparative results of *NTAM* and its 10 state-of-the-art competitors on both industrial datasets (*i.e., Dataset-1* and *Dataset-2*). As can be clearly seen in Table 3, on both *Dataset-1* and *Dataset-2*, deep learning based approaches (*i.e., RNN, LSTM, TCNN, CNN+LSTM* and *NTAM*) significantly outperform other approaches; this is not surprising, because deep learning based approaches have the ability to leverage the temporal information, which could make considerable improvements to the practical performance in disk failure prediction. When focusing on the comparisons among our *NTAM* approach and its competitors, it is clear that *NTAM* stands out as the best approach for solving both industrial datasets, and achieves much better performance than its all competitors in terms of all metrics (*i.e.,* precision, recall and F1-score).

In particular, in terms of the metric of precision on *Dataset-1* and *Dataset-2*, *NTAM* achieves the precision values of 78.16% and 81.07%, which are 4.80% and 7.06% greater than those achieved by the second best approach for each dataset *TCNN* and *CNN+LSTM*, respectively; in terms of the metric of recall on *Dataset-1* and *Dataset-2*, the recall values obtained by *NTAM* are 56.28% and 57.58%, which are 6.10% and 6.31% greater than those obtained by the second best approach for each dataset *CDEF* and *TCNN*, respectively; then we focus on the metric of F1-score on *Dataset-1* and *Dataset-2*, *NTAM* achieves the F1-score values of 65.44% and 67.34%, which are 7.41% and 7.15% greater than those achieved by the second best approach for each dataset *CNN+LSTM* and *TCNN*, respectively.

4.4.2 Effectiveness of Neighborhood-aware Component. In order to demonstrate the effectiveness of our proposed neighborhood-aware component underlying *NTAM*, we modify *NTAM* to disable the neighborhood-aware component, resulting in an alternative approach called *NTAM_alt1*. We conduct an experiment to directly compare *NTAM* against *NTAM_alt1* on both industrial datasets (*i.e., Dataset-1* and *Dataset-2*), and the related experimental results are presented in Table 4. According to Table 4, on both *Dataset-1* and *Dataset-2*, *NTAM* can achieve much better performance than *NTAM_alt1* in terms of all metrics of precision, recall and F1-score, which indicates the effectiveness of the neighborhood-aware component underlying *NTAM*.

Table 4: Comparative results of *NTAM* and its 3 alternative version (*i.e., NTAM_alt1, NTAM_alt2* and *NTAM_alt3*) on the industrial dataset. P, R, and F1 are referring to precision, recall, and F1-score, respectively.

Approach	*Dataset-1*			*Dataset-2*		
	P	R	F1	P	R	F1
NTAM_alt1	74.07	51.26	60.59	75.64	51.95	61.60
NTAM_alt2	74.84	53.90	62.67	76.98	54.28	63.67
NTAM_alt3	75.93	54.21	63.26	77.53	55.94	64.99
NTAM	**78.16**	**56.28**	**65.44**	**81.07**	**57.58**	**67.34**

In addition, we modify *NTAM* to replace the temporal component underlying *NTAM* with LSTM and TCNN, resulting in other two alternative approaches dubbed *NTAM_alt2* and *NTAM_alt3*, respectively. The experimental results of *NTAM_alt2* and *NTAM_alt3* on both industrial datasets (*i.e., Dataset-1* and *Dataset-2*) are also illustrated in Table 4. Experimental results show that, on both *Dataset-1* and *Dataset-2*, *NTAM_alt2* performs much better than *LSTM* (in Table 3), and *NTAM_alt3* obtains better performance than *TCNN* (in Table 3), confirming that our proposed neighborhood-aware component underlying *NTAM* can provide the significant performance improvement in disk failure prediction.

4.4.3 Effectiveness of Temporal Component. As discussed in Section 3, besides the neighborhood-aware component, the temporal component is also an important component underlying *NTAM*. Hence, we conduct experiments to confirm the effectiveness of the temporal component in practice. According to the results reported in Table 3, deep learning based approaches (including LSTM and TCNN), which can capture temporal information, exhibit good performance on the industrial dataset. Since *NTAM_alt1* can be treated as *NTAM* only using the temporal information, from Tables 3 and 4, *NTAM_alt1* performs better than *LSTM* and *TCNN* on both industrial datasets, indicating the effectiveness of the temporal component. Moreover, *NTAM_alt2* and *NTAM_alt3* are two alternative variants of *NTAM* by replacing the temporal component with *LSTM* and *TCNN*, respectively. The comparisons among *NTAM*, *NTAM_alt2* and *NTAM_alt3* demonstrate that *NTAM* can obtain higher precision, recall and F1-score than *NTAM_alt2* and *NTAM_alt3* on both industrial datasets, confirming the superiority of our proposed temporal component underlying *NTAM*.

4.4.4 Effectiveness of NTAM *with Different Combinations of Attributes.* As described in Section 4.3, both industrial datasets contain three main kinds of disk-related attributes: SMART attributes, system-related attributes and driver-related attributes. Actually, in the context of disk failure prediction, previous works [3, 11, 19, 35, 37, 42, 46, 49, 50] commonly use SMART attributes, and its effectiveness is well demonstrated in the related experimental results.

Being complementary to previous works which study the effectiveness of SMART attributes, this work conducts empirical evaluations to analyze the effectiveness of system-related attributes as well as driver-related attributes. In order to achieve this, we conduct more empirical evaluations of *NTAM* on the most recent industrial dataset (*i.e., Dataset-2*) using 4 combinations of attributes,

Table 5: Comparative results of *NTAM* on the most recent industrial dataset (*i.e., Dataset-2*) using different combinations of attributes.

Attribute Combination	Precision	Recall	F1-Score
SMART	71.29	52.26	60.31
SMART+System	77.53	55.68	64.81
SMART+Driver	75.68	55.10	63.78
SMART+System+Driver	**81.07**	**57.58**	**67.34**

i.e., SMART (only using SMART attributes), SMART+System (using SMART and system-related attributes), SMART+Driver (using SMART and driver-related attributes) and SMART+System+Driver (using all SMART, system-related, and driver-related attributes). The related experimental results are presented in Table 5.

According to the experimental results in Table 5, both *NTAM* using SMART+System and *NTAM* using SMART+Driver achieve much better performance than *NTAM* using only SMART attributes in terms of all metrics, which indicates that both system-related and driver-related attributes can effectively contribute to the performance improvement in disk failure prediction. Also, *NTAM* using SMART+System+Driver in turn performs much better than *NTAM* using SMART+System as well as *NTAM* using SMART+Driver in terms of all metrics, indicating that leveraging more useful attributes could lead to better practical performance in disk failure prediction.

4.5 Analysis of *TPS*

Since our *TPS* method is proposed to address the extreme data imbalance, in this section, we conduct empirical evaluations to analyze *TPS*. In particular, we first study the effect of hyper-parameter L for *TPS*, and then evaluate the effectiveness of our *TPS* method.

4.5.1 Effect of Hyper-parameter L in TPS. We equip *NTAM* with *TPS*, resulting in an enhanced disk failure prediction approach dubbed *NTAM+TPS*. As described in Section 3.6, *TPS* introduces a hyper-parameter L, which determines the number of failed samples generated for each failed disk. In order to study the effect of hyper-parameter L for *TPS*, we conduct the experiments of *NTAM+TPS* on the most recent industrial dataset (*i.e., Dataset-2*) with L ranging from 2 to 32 with the increment of 2, and the related experimental results are illustrated in Figure 5.

From Figure 5, we observe that, when the hyper-parameter L increases from 2 to 16, the F1-score value achieved by *NTAM+TPS* is consistently increased. Also, *NTAM+TPS* achieves the maximum F1-score value of 72.99% with L=16, which is much better than the F1-score value achieved by *NTAM* (as presented in Table 3). Similarly, *NTAM+TPS* with L=16 achieves better precision and recall than *NTAM*. The main reason why *TPS* works well is that *TPS* introduces more variations of failure patterns, which makes *NTAM* learn with more useful information.

When L is increased from 16 to 32, the F1-score achieved by *NTAM+TPS* degrades. Actually, this is not surprising – the larger leading time means the longer time to fail, so the characteristics of the failure are less obvious, which makes the given disk with such large leading time be less divergent with the healthy disks.

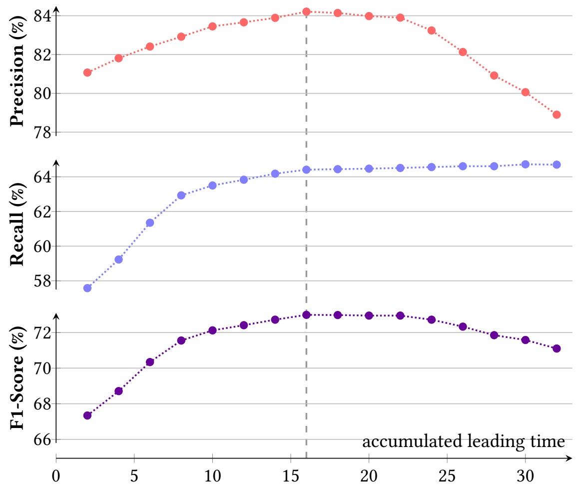

Figure 5: Experimental results of *NTAM+TPS* with different hyper-parameter settings of L on the most recent industrial dataset (*i.e., Dataset-2*).

4.5.2 *Effectiveness of* TPS. To evaluate the effectiveness of *TPS* comprehensively, besides *NTAM+TPS*, we equip *NTAM*'s all 10 competitors with *TPS* using the same hyper-parameter setting (i.e., L=16), resulting in 10 enhanced versions of *NTAM*'s competitors. The experimental results of *NTAM*'s competitors equipped with *TPS* are reported in Table 3. From Table 3, *TPS* is able to improve all 10 *NTAM*'s competitors in terms of precision, recall and F1-score. The results demonstrated in Table 3 indicate that our proposed *TPS* method is generally applicable to various disk failure prediction approaches and is able to improve their practical performance.

4.5.3 *Comparisons among* TPS *and State-of-the-art Methods for Handling Data Imbalance Problem.* To further evaluate the effectiveness of *TPS*, we conduct an experiment to compare *TPS* against existing state-of-the-art methods for handling data imbalance problem in disk failure prediction. In this comparative experiment, we adopt 4 state-of-the-art methods that are widely used to deal with the data imbalance problem in the context of disk failure prediction: 1) clustering-based method [3]; 2) latest sampling method [35]; 3) weight adjusting based cost-sensitive method [19]; 4) new loss function based cost-sensitive method [37]. For the sake of simplicity, in this work those 4 state-of-the-art methods for handling the data imbalance problem in disk failure prediction are named $M1$, $M2$, $M3$ and $M4$, respectively. We note that $M1$ and $M2$ are under-sampling methods, while $M3$ and $M4$ are cost-sensitive methods.

The comparative results of *NTAM* equipped with *TPS* and those 4 state-of-the-art methods for handling the data imbalance problem on *Dataset-2* are summarized in Table 6. From Table 6, it is apparent that, in terms of the metrics of precision, recall and F1-score, the performance *NTAM+TPS* is better than that of *NTAM* equipped with all 4 competing methods for handling the data imbalance problem. For instance, the precision, recall and F1-score achieved by *NTAM+TPS* are 84.22%, 64.41% and 72.99%, respectively, which are 1.39%, 4.67% and 3.58% greater than the precision, recall and F1-score achieved by the second best approach *NTAM+M4*, respectively.

Table 6: Comparative results of *NTAM* equipped with *TPS* and other methods for handling data imbalance problem on the most recent industrial dataset (*i.e., Dataset-2*).

Approach	Precision	Recall	F1-Score
NTAM+M1	82.41	59.65	69.20
NTAM+M2	82.12	58.93	68.62
NTAM+M3	81.04	58.52	67.96
NTAM+M4	82.83	59.74	69.41
NTAM+TPS	**84.22**	**64.41**	**72.99**

The comparative results in Table 6 clearly indicate that *TPS* achieves the state-of-the-art performance for handling the data imbalance problem in the context of disk failure prediction.

4.6 Empirical Evaluation on Public Dataset

To further empirically evaluate the robustness and the effectiveness, we additionally adopt a public dataset called *Backblaze*[1], which has been widely used in existing works [3, 37, 46] and is a standard, well-adopted benchmark for evaluating the performance of approaches for disk failure prediction.

The public *Backblaze* dataset contains the timestamp, disk model, serial number, SMART attributes and label of each disk. We use its data from January 2017 to December 2018 as the training set, and adopt its data from January 2019 to June 2019 as the testing set. The training and testing sets contain the data of 58,586 disks over 30 months. Also, the number of positive samples (denoted as '#Positive') and the number of negative samples (denoted as '#Negative') for the public dataset are listed in Table 7. Since this dataset is collected on a daily basis, we use consecutive 15 days SMART data to represent each disk's status following the standard practice [46]. Therefore, for the number of consecutive feature vectors in a disk's status data, we set $h = 15$ in the public dataset.

Table 8 summarizes the comparative results of *NTAM* and all its state-of-the-art competitors on the public dataset. Also, Table 9 reports the comparative results of *NTAM* and all its state-of-the-art competitors with *TPS* on the public dataset. The experimental results in Tables 8 and 9 confirm that deep learning based approaches exhibit better performance on disk failure prediction. Table 8 presents that *NTAM* can achieve better performance in terms of all metrics (*i.e.,* precision, recall and F1-score) than the second best approach (*i.e., CNN+LSTM* for precision and F1-score, and *TCNN* for recall). Also, Table 9 demonstrates that *NTAM+TPS* consistently achieves better performance in terms of all metrics (*i.e.,* precision, recall and F1-score) than the second best approach (*i.e., CNN+LSTM+TPS* for precision, and *TCNN+TPS* for recall and F1-score). It is not surprising that our proposed approaches (*i.e., NTAM* and *NTAM+TPS*) achieve better precision, recall and F1-score results on the public dataset than those numbers on two industrial datasets, because the ratio between positive samples and negative samples is more balanced in the public dataset.

Since the public dataset does not record the disk neighborhood information, we cannot directly assess the effectiveness of the neighborhood-aware component on this dataset. Nevertheless, the

[1] https://www.backblaze.com/b2/hard-drive-test-data.html

Table 7: Summary of the public dataset.

Dataset	#Positive	#Negative
Training set	1,642	47,708
Testing set	210	46,217

Table 8: Comparative results of *NTAM* and its 10 state-of-the-art competitors on the public dataset.

Approach	Precision	Recall	F1-Score
SVM	63.12	59.78	61.41
DT	67.98	65.04	66.48
RF	71.72	68.16	69.89
GBDT	77.91	70.34	73.93
RGF	74.85	69.67	72.17
CDEF	78.01	72.83	75.33
RNN	78.76	73.15	75.85
LSTM	79.04	74.53	76.72
TCNN	79.58	74.92	77.18
CNN+LSTM	81.03	74.25	77.49
NTAM	**84.01**	**76.43**	**80.04**

results show that *NTAM+TPS* outperforms other approaches (especially those deep learning based ones) when the dataset does not contain the neighborhood information, which confirms the effectiveness of *TPS* and the temporal component underlying *NTAM*.

5 APPLICATION IN PRACTICE

We have successfully applied our proposed *NTAM+TPS* approach to Microsoft cloud platforms (including Microsoft Azure and Microsoft 365), in order to help improve the service reliability. Microsoft cloud platforms have achieved global scales on worldwide networks of data centers across many regions and serve huge amount of workloads. It is critically important for cloud platforms to ensure high service reliability [13].

In our industrial practice, the prediction task runs as an hourly task, which collects the most recent signals from each computing node. After collecting all necessary signals, feature engineering is performed, including extracting useful attributes, concatenating different type of attributes (*i.e.,* SMART ones, system-related ones and driver-related ones [34, 38]) by *Disk ID*, arranging the feature snapshot to time series, appending neighbor information for each disk, *etc*. Subsequently, the prepared feature vectors are fed to *NTAM+TPS* and the failure probabilities are output. Finally, the disks with high failure probabilities will be selected for follow-up proactive mitigation actions, such as blocking new allocation on risky servers, and live migrating existing virtual machines off the risky servers.

We analyze the virtual machine interruption caused by disk failure. Based on the data collected from Microsoft cloud platforms (including Microsoft Azure and Microsoft 365) before and after the deployment of *NTAM+TPS*, our proposed approach significantly reduced the number of virtual machine interruptions for those cloud platforms. Hence, our proposed *NTAM+TPS* approach has considerably improved the service reliability of Microsoft cloud platforms, and obtained benefits in industrial practice.

Table 9: Comparative results of *NTAM+TPS* and its 10 state-of-the-art competitors equipped with *TPS* on the public dataset.

Approach	Precision	Recall	F1-Score
SVM+TPS	67.55	65.81	66.67
DT+TPS	71.06	70.68	70.87
RF+TPS	75.33	73.36	74.34
GBDT+TPS	81.97	76.89	79.35
RGF+TPS	78.11	74.60	76.31
CDEF+TPS	81.72	78.13	79.88
RNN+TPS	81.04	79.08	80.05
LSTM+TPS	82.85	80.07	81.43
TCNN+TPS	83.41	80.24	81.79
CNN+LSTM+TPS	84.03	79.56	81.74
NTAM+TPS	**87.37**	**82.86**	**85.05**

6 CONCLUSION

Disk failure is one of the major reasons that cause cloud platforms unreliable. Predicting disk failures plays a crucial role in industrial practice. In this paper, we propose a novel neighborhood-temporal attention based approach dubbed *NTAM* for disk failure prediction. Compared to existing approaches which only focus on that disk's own status data, *NTAM* is a novel approach which also considers a disk's neighbors' status data. Moreover, *NTAM* introduces a novel attention based temporal component to capture the temporal nature of the disk status data. Our experiments on industrial and public datasets demonstrate that *NTAM* achieves much better performance than its 10 state-of-the-art competitors, indicating that *NTAM* considerably advances the state of the art. Further evaluations also confirm the effectiveness of the neighborhood-aware component and the temporal component underlying *NTAM*. Furthermore, we propose an effective method *TPS* to deal with the extreme data imbalance problem in disk failure prediction, and the related experimental results demonstrate that *TPS* can improve the performance of various disk failure prediction approaches. More encouragingly, *NTAM* and *TPS* have been successfully applied to Microsoft cloud platforms (including Microsoft Azure and Microsoft 365) and obtained benefits in industrial practice.

For future work, we plan to incorporate our approach with the techniques of automated feature engineering [4] and positive-unlabeled learning [25] to address the challenges of feature engineering and label noise in disk failure prediction, respectively.

ACKNOWLEDGMENTS

We would like to thank Murali Chintalapati and Jim Kleewein for their great support and sponsorship. We would also like to thank Sebastien Levy and Randolph Yao for the collaboration on the deployment of disk failure prediction in Microsoft Azure, and to thank Susy Yi, Mona Su, Paul Wang, Andrew Zhou, Robert Gu, Victor Rühle and Yogesh Bansal for the collaboration on the deployment of disk failure prediction in Microsoft 365.

REFERENCES

[1] Bruce Allen. 2004. Monitoring Hard Disks with SMART. *Linux Journal* (2004).

[2] Danilo Ardagna, Barbara Panicucci, and Mauro Passacantando. 2011. A Game Theoretic Formulation of the Service Provisioning Problem in Cloud Systems. In *Proceedings of WWW 2011*. 177–186.

[3] Mirela Madalina Botezatu, Ioana Giurgiu, Jasmina Bogojeska, and Dorothea Wiesmann. 2016. Predicting Disk Replacement towards Reliable Data Centers. In *Proceedings of KDD 2016*. 39–48.

[4] Xiangning Chen, Qingwei Lin, Chuan Luo, Xudong Li, Hongyu Zhang, Yong Xu, Yingnong Dang, Kaixin Sui, Xu Zhang, Bo Qiao, Weiyi Zhang, Wei Wu, Murali Chintalapati, and Dongmei Zhang. 2019. Neural Feature Search: A Neural Architecture for Automated Feature Engineering. In *Proceedings ICDM 2019*. 71–80.

[5] Yujun Chen, Xian Yang, Qingwei Lin, Hongyu Zhang, Feng Gao, Zhangwei Xu, Yingnong Dang, Dongmei Zhang, Hang Dong, Yong Xu, Hao Li, and Yu Kang. 2019. Outage Prediction and Diagnosis for Cloud Service Systems. In *Proceedings of WWW 2019*. 2659–2665.

[6] Chenyou Fan, Yuze Zhang, Yi Pan, Xiaoyue Li, Chi Zhang, Rong Yuan, Di Wu, Wensheng Wang, Jian Pei, and Heng Huang. 2019. Multi-Horizon Time Series Forecasting with Temporal Attention Learning. In *Proceedings of KDD 2019*. 2527–2535.

[7] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proceedings of ICML 2017*. 1243–1252.

[8] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of AISTATS 2011*. 315–323.

[9] Jiazhen Gu, Chuan Luo, Si Qin, Bo Qiao, Qingwei Lin, Hongyu Zhang, Ze Li, Yingnong Dang, Shaowei Cai, Wei Wu, Yangfan Zhou, Murali Chintalapati, and Dongmei Zhang. 2020. Efficient incident identification from multi-dimensional issue reports via meta-heuristic search. In *Proceedings of ESEC/FSE 2020*. 292–303.

[10] Jiazhen Gu, Jiaqi Wen, Zijian Wang, Pu Zhao, Chuan Luo, Yu Kang, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, Bo Qiao, Liqun Li, Qingwei Lin, and Dongmei Zhang. 2020. Efficient customer incident triage via linking with system incidents. In *Proceedings of ESEC/FSE 2020*. 1296–1307.

[11] Xiaohong Huang. 2017. *Hard Drive Failure Prediction for Large Scale Storage System*. Ph.D. Dissertation. UCLA.

[12] Ponemon Institute. 2016. Cost of Data Center Outages. *Data Center Performance Benchmark Series* (2016).

[13] Hiranya Jayathilaka, Chandra Krintz, and Rich Wolski. 2017. Performance Monitoring and Root Cause Analysis for Cloud-hosted Web Applications. In *Proceedings of WWW 2017*. 469–478.

[14] Ishan Jindal, Matthew S. Nokleby, and Xuewen Chen. 2016. Learning Deep Networks from Noisy Labels with Dropout Regularization. In *Proceedings of ICDM 2016*. 967–972.

[15] Rie Johnson and Tong Zhang. 2014. Learning Nonlinear Functions Using Regularized Greedy Forest. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 5 (2014), 942–954.

[16] Bartosz Krawczyk. 2016. Learning from Imbalanced Data: Open Challenges and Future Directions. *Progress in Artificial Intelligence* 5, 4 (2016), 221–232.

[17] Sebastien Levy, Randolph Yao, Youjiang Wu, Yingnong Dang, Peng Huang, Zheng Mu, Pu Zhao, Tarun Ramani, Naga Govindaraju, Xukun Li, Qingwei Lin, Gil Lapid Shafriri, and Murali Chintalapati. 2020. Predictive and Adaptive Failure Mitigation to Avert Production Cloud VM Interruptions. In *Proceedings of OSDI 2020*. 1155–1170.

[18] Huayu Li, Martin Renqiang Min, Yong Ge, and Asim Kadav. 2017. A Context-aware Attention Network for Interactive Question Answering. In *Proceedings of KDD 2017*. 927–935.

[19] Jing Li, Xinpu Ji, Yuhan Jia, Bingpeng Zhu, Gang Wang, Zhongwei Li, and Xiaoguang Liu. 2014. Hard Drive Failure Prediction Using Classification and Regression Trees. In *Proceedings of DSN 2014*. 383–394.

[20] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. 2019. Neural Speech Synthesis with Transformer Network. In *Proceedings of AAAI 2019*. 6706–6713.

[21] Ze Li, Qian Cheng, Ken Hsieh, Yingnong Dang, Peng Huang, Pankaj Singh, Xinsheng Yang, Qingwei Lin, Youjiang Wu, Sebastien Levy, and Murali Chintalapati. 2020. Gandalf: An Intelligent, End-To-End Analytics Service for Safe Deployment in Large-Scale Cloud Infrastructure. In *Proceedings of NSDI 2020*. 389–402.

[22] Sidi Lu, Bing Luo, Tirthak Patel, Yongtao Yao, Devesh Tiwari, and Weisong Shi. 2020. Making Disk Failure Predictions SMARTer!. In *Proceedings of FAST 2020*. 151–167.

[23] Chuan Luo, Bo Qiao, Xin Chen, Pu Zhao, Randolph Yao, Hongyu Zhang, Wei Wu, Andrew Zhou, and Qingwei Lin. 2020. Intelligent Virtual Machine Provisioning in Cloud Computing. In *Proceedings of IJCAI 2020*. 1495–1502.

[24] Chuan Luo, Bo Qiao, Wenqian Xing, Xin Chen, Pu Zhao, Chao Du, Randolph Yao, Hongyu Zhang, Wei Wu, Shaowei Cai, Bing He, Saravanakumar Rajmohan, and Qingwei Lin. 2021. Correlation-Aware Heuristic Search for Intelligent Virtual Machine Provisioning in Cloud Systems. In *Proceedings of AAAI 2021*.

[25] Chuan Luo, Pu Zhao, Chen Chen, Bo Qiao, Chao Du, Hongyu Zhang, Wei Wu, Shaowei Cai, Bing He, Saravanakumar Rajmohan, and Qingwei Lin. 2021. PULNS: Positive-Unlabeled Learning with Effective Negative Sample Selector. In *Proceedings of AAAI 2021*.

[26] Meng Ma, Jingmin Xu, Yuan Wang, Pengfei Chen, Zonghua Zhang, and Ping Wang. 2020. AutoMAP: Diagnose Your Microservice-based Web Applications Automatically. In *Proceedings of WWW 2020*. 246–258.

[27] Michael Menzel and Rajiv Ranjan. 2012. CloudGenius: Decision Support for Web Server Cloud Migration. In *Proceedings of WWW 2012*. 979–988.

[28] Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu. 2015. A Large-Scale Study of Flash Memory Failures in the Field. In *Proceedings of SIGMETRICS 2015*. 177–190.

[29] Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. MIT Press.

[30] Molly S. Quinn, Katherine Campbell, and Mark T. Keane. 2019. The Expected Unexpected & Unexpected Unexpected. In *Proceedings of CogSci 2019*. 2627–2633.

[31] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2020. Compressive Transformers for Long-Range Sequence Modelling. In *Proceedings of ICLR 2020*.

[32] Sriram Sankar, Mark Shaw, Kushagra Vaid, and Sudhanva Gurumurthi. 2013. Datacenter Scale Evaluation of the Impact of Temperature on Hard Disk Drive Failures. *ACM Transactions on Storage* 9, 2 (2013), 1–24.

[33] Shaukat Ali Shahee and Usha Ananthakumar. 2018. An Adaptive Oversampling Technique for Imbalanced Datasets. In *Proceedings of Industrial Conference on Data Mining 2018*. 1–16.

[34] Huasong Shan, Yuan Chen, Haifeng Liu, Yunpeng Zhang, Xiao Xiao, Xiaofeng He, Min Li, and Wei Ding. 2019. ϵ-Diagnosis: Unsupervised and Real-time Diagnosis of Small-window Long-tail Latency in Large-scale Microservice Platforms. In *Proceedings of WWW 2019*. 3215–3222.

[35] Jing Shen, Jian Wan, Se-Jung Lim, and Lifeng Yu. 2018. Random-Forest-Based Failure Prediction for Hard Disk Drives. *International Journal of Distributed Sensor Networks* 14, 11 (2018).

[36] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[37] Xiaoyi Sun, Krishnendu Chakrabarty, Ruirui Huang, Yiquan Chen, Bing Zhao, Hai Cao, Yinhe Han, Xiaoyao Liang, and Li Jiang. 2019. System-Level Hardware Failure Prediction using Deep Learning. In *Proceedings of DAC 2019*. 20.

[38] Amoghavarsha Suresh and Anshul Gandhi. 2019. Using Variability as a Guiding Principle to Reduce Latency in Web Applications via OS Profiling. In *Proceedings of WWW 2019*. 1759–1770.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proceedings of NIPS 2017*. 5998–6008.

[40] Changjun Wang, Weidong Ma, Tao Qin, Xujin Chen, Xiaodong Hu, and Tie-Yan Liu. 2015. Selling Reserved Instances in Cloud Computing. In *Proceedings of IJCAI 2015*. 224–231.

[41] Xuejian Wang, Lantao Yu, Kan Ren, Guanyu Tao, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Dynamic Attention Deep Model for Article Recommendation by Learning Human Editors' Demonstration. In *Proceedings of KDD 2017*. 2051–2059.

[42] Chang Xu, Gang Wang, Xiaoguang Liu, Dongdong Guo, and Tie-Yan Liu. 2016. Health Status Assessment and Failure Prediction for Hard Drives with Recurrent Neural Networks. *IEEE Transactions on Computers* 65, 11 (2016), 3502–3508.

[43] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of ICML 2015*. 2048–2057.

[44] Yong Xu, Kaixin Sui, Randolph Yao, Hongyu Zhang, Qingwei Lin, Yingnong Dang, Peng Li, Keceng Jiang, Wenchi Zhang, Jian-Guang Lou, Murali Chintalapati, and Dongmei Zhang. 2018. Improving Service Availability of Cloud Systems by Predicting Disk Error. In *Proceedings of USENIX ATC 2018*. 481–494.

[45] Qiang Yang and Xindong Wu. 2006. 10 Challenging Problems in Data Mining Research. *International Journal of Information Technology & Decision Making* 5, 04 (2006), 597–604.

[46] Jianguo Zhang, Ji Wang, Lifang He, Zhao Li, and Philip S. Yu. 2018. Layer-wise Perturbation-Based Adversarial Training for Hard Drive Health Degree Prediction. In *Proceedings of ICDM 2018*. 1428–1433.

[47] Xiangyu Zhao, Longbiao Wang, Ruifang He, Ting Yang, Jinxin Chang, and Ruifang Wang. 2020. Multiple Knowledge Syncretic Transformer for Natural Dialogue Generation. In *Proceedings of WWW 2020*. 752–762.

[48] Xinyan Zhao, Feng Xiao, Haoming Zhong, Jun Yao, and Huanhuan Chen. 2020. Condition Aware and Revise Transformer for Question Answering. In *Proceedings of WWW 2020*. 2377–2387.

[49] Ying Zhao, Xiang Liu, Siqing Gan, and Weimin Zheng. 2010. Predicting Disk Failures with HMM- and HSMM-Based Approaches. In *Proceedings of Industrial Conference on Data Mining 2010*. 390–404.

[50] Bingpeng Zhu, Gang Wang, Xiaoguang Liu, Dianming Hu, Sheng Lin, and Jingwei Ma. 2013. Proactive Drive Failure Prediction for Large Scale Storage Systems. In *Proceedings of MSST 2013*. 1–5.