An Empirical Investigation of Missing Data Handling in Cloud Node Failure Prediction

Minghua Ma
Yudong Liu
Microsoft Research
Beijing, China

Yuang Tong
Haozhe Li
Microsoft Research
Beijing, China

Pu Zhao
Yong Xu
Microsoft Research
Beijing, China

Hongyu Zhang
The University of Newcastle
Callaghan, NSW, Australia

Shilin He
Lu Wang
Microsoft Research
Beijing, China

Yingnong Dang
Microsoft Azure
Redmond, USA

Saravanakumar Rajmohan
Microsoft 365
Redmond, USA

Qingwei Lin
Microsoft Research
Beijing, China

ABSTRACT

Cloud computing systems have become increasingly popular in recent years. A typical cloud system utilizes millions of computing nodes as the basic infrastructure. Node failure has been identified as one of the most prevalent causes of cloud system downtime. To improve the reliability of cloud systems, many previous studies collected monitoring metrics from nodes and built models to predict node failures before the failures happen. However, based on our experience with large-scale real-world cloud systems in Microsoft, we find that the task of predicting node failure is severely hampered by missing data. There is a large amount of missing data, and the online latest data utilized for prediction is even worse. As a result, the real-time performance of the node prediction model is limited. In this paper, we first characterize the missing data problem for node failure prediction. Then, we evaluate several existing data interpolation approaches, and find that node dimension interpolation approaches outperform time dimension ones and deep learning based interpolation is the best for early prediction. Our findings can help academics and engineers address the missing data problem in cloud node failure prediction and other data-driven software engineering scenarios.

CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**; • **Software and its engineering** → **Maintaining software**.

KEYWORDS

Node failure prediction; Missing data; Cloud systems

*Qingwei Lin is the corresponding author of this work.

ESEC/FSE '22, November 14–18, 2022, Singapore, Singapore

ACM ISBN 978-1-4503-9413-0/22/11…$15.00
https://doi.org/10.1145/3540250.3558946

ACM Reference Format:
Minghua Ma, Yudong Liu, Yuang Tong, Haozhe Li, Pu Zhao, Yong Xu, Hongyu Zhang, Shilin He, Lu Wang, Yingnong Dang, Saravanakumar Rajmohan, Qingwei Lin. 2022. An Empirical Investigation of Missing Data Handling in Cloud Node Failure Prediction. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3540250.3558946

1 INTRODUCTION

With the rapid growth of cloud computing, a variety of workloads have been shifted into the cloud due to its scalability and reliability. To serve millions of users around the world, it is of great importance for cloud systems, such as Azure, AWS, and Google Cloud, to provide highly reliable services, since any interruption or outage will negatively impact user experience and result in considerable economic loss. For example, Facebook became globally unavailable for a period of six to seven hours on October 4, 2021 [37], and share price of the company dropped by nearly 5%.

To ensure the reliability of cloud systems, tremendous efforts have been devoted in the past decades, including failure prediction [17, 21], anomaly detection [18, 24, 25], and root cause diagnosis [11, 23]. Among these efforts, failure prediction is one of the most fundamental topics because it may foresee and avoid failure rather than react when it occurs, or after it has occurred. More specifically, cloud systems are typically built on top of millions of computing nodes, which provide physical hosts for virtual machines, storage, and network. Node failure, such as hard drive failure [21] and I/O block [45], is one of the top causes of service downtime [17]. Therefore, it is important to predict node failure in real time [41].

Previous studies build different machine learning models, especially temporal sequential models, such as LSTM [44], RNN [40], NTAM [21], and TCNN [34] to predict node failure in real-time. These models take the monitoring metrics, including node status and specifically designed events [17], as input. However, based on our experience working with the Microsoft 365 cloud systems, the node failure prediction model suffers from low accuracy. That is, the model is trained and tested on the offline monitoring metrics and cannot obtain satisfactory results. When the model is performed in an online environment, it performs even worse.

Because the failure prediction model is severely hampered by missing data in the monitoring metrics [5, 9, 11, 33]. The offline monitoring metrics suffer from missing data and the online missing data is more frequently caused by data delay, overload, *etc.* In the literature, a large number of studies on missing data handling have been proposed, which focus on interpolating the missing value [4, 35]. However, handling missing data for node failure prediction is still unexplored. As we will show in this paper (Section 3), missing data of monitoring metrics have some different characteristics than the generic missing data problem, although they have much in common. For example, missing data are treated equally in the general interpolation approaches, while monitoring metrics of nodes have different features, such as error counter, operation counter, EVENT, *etc.* Their data missing is caused by different reasons, which should be treated differently.

To better understand the missing data problem for node failure prediction in the cloud, we perform a large-scale empirical study of missing data on millions of nodes[1] in Microsoft, including widely-used Microsoft products, Microsoft 365. In this study, we collect over 7 GB of monitoring metrics and uncover a significant proportion of missing data for different causes. We also characterize missing data in three different dimensions, that is, how data is missing among nodes, within features, and over time. We find that the majority of node monitoring metrics have the same missing percentage, indicating similarity among nodes **Finding 1** . Besides, different features are caused by various reasons, *i.e.*, data delay, monitoring error, overload, or design. The EVENT feature must fill in empty with zero because it is missing by design and zero indicates that no event occurred at that moment **Finding 2** . Moreover, missing data dramatically increases in the latest time due to data delay. Monitoring metrics in the latest time contain additional information (*e.g.*, high I/O utilization) to help identify the failure nodes, but is likely missing, causing prediction performance in real time to degrade **Finding 3** .

Since missing data is a severe problem in real-world practice, we conduct experiments to assess the impact of missing data on node failure prediction for cloud systems. We find that the F1-score of the node failure prediction models declines by 57.33% when faced with 20% more missing data, indicating its negative impact and motivating us to explore solutions to handle it **Finding 4** . Therefore, we select widely-used interpolation approaches, *i.e.*, zero filling [27], forward filling [1], linear interpolation [30], average value interpolation [35], Gaussian distribution sampling interpolation [29], KNN based interpolation [7], and deep learning based interpolation [4], from previous studies. More specifically, we apply interpolation approaches to the time dimension and node dimension, which consider the similarity over time and among nodes, respectively. Interestingly, we find that different models prefer different interpolation approaches. The node dimension interpolation, on the other hand, performs substantially better than the time dimension interpolation by 3% on average **Finding 5** , which can be explained by our Finding 1 about similarity among nodes. These interpolation approaches are efficient for online data interpolation, except KNN based interpolation **Finding 6** . According to our Finding 3, we also evaluate the online data delay scenario to test the performance of early prediction based on various interpolation approaches. In this context, we find that deep learning based interpolation performs the best, suggesting its capabilities for early prediction **Finding 7** .

[1]Due to the policy of Microsoft, we cannot disclose the actual number in this paper.

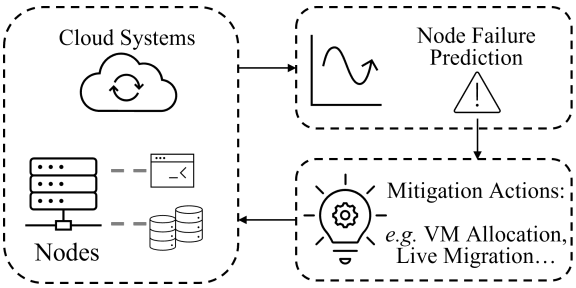


Figure 1: The overview of node failure prediction

To sum up, this work has the following contributions:

- To the best of our knowledge, we present the first comprehensive study on handling missing data for node failure prediction in real-world cloud systems.
- We point out that no existing interpolation approach is suitable for all prediction models, however, the node dimension interpolation approaches outperform the time dimension ones, and deep learning based interpolation is the best for early prediction.
- We discuss the implications and lessons learned that can help handle the missing data problem in cloud node failure prediction and other data-driven software engineering scenarios.

The rest of this paper is organized as follows: we introduce the background of node failure prediction in Section 2. Then, we conduct an empirical study of missing data in cloud node failure prediction in Section 3. Next, the empirical evaluation of missing data interpolation is described in Section 4. We also discuss the insights and lessons learned in Section 5. Finally, the related work and conclusion are presented in Section 6 and Section 7, respectively.

2 BACKGROUND: NODE FAILURE PREDICTION FOR CLOUD SYSTEMS

Cloud systems. Cloud computing is a novel paradigm for delivering computing as a service through the Internet. Cloud systems, such as Microsoft Azure and Amazon Web Services, provide platforms for software companies and developers to deploy software applications and services. These cloud systems are built on top of numerous physical servers, or "nodes", which provide services through virtualization technique. For large-scale cloud systems, ensuring high service reliability is critical to provide good user experiences and prevent financial loss [6]. Although much effort has been made to ensure high service reliability, software or platform failures (such as software crashes, network outages, misconfigurations, memory leaks, hardware breakdowns, and so on) are still negligible and challenging to deal with. In this study, we investigate millions of nodes in Microsoft 365 Cloud Systems.

Table 1: Node monitoring metrics

Type	#	Description	Example(s)
Operation Timer	3	The timer for some common operations (operations for the hard driver disks such as magnetic head's seeking, spindle's spinning up, *etc.*)	*Seek_Time_Performance* *Spin_Up_Time* *Head_Flying_Hours*
Physical Characteristics	4	The measurement of physical characteristics, *e.g.*, temperature and humidity, *etc.*	*Airflow_temperature*
Operation Counter	11	Count of common operations, such as CPU utilization, load, and power consumption.	*Load_Cycle_Count* *Power-Off_Retract_Count*
Error Counter	12	Count of certain detectable errors. Usually these counters remains unchanged unless certain error occurs.	*High_Fly_Writes* *Multi_Zone_Error_Rates*
Logical Input & Output	12	Commonly used as an indicator of the Read or Write queue lengths of the node.	*LogicalDiskAvgRead_BucketIO* *LogicalDiskAvgWrite_BucketIO*
EVENT	12	Windows event are typically used for indicating exception handling for Windows servers.	*Event_id_7*

Node failure. Node failure has been identified as one of the most prevalent causes of system downtime [36], which can be caused by service overloading, overheating, or hardware issues. If a node fails, all virtual machines (VMs) on it will also fail, potentially affecting service availability. Although node failure is inevitable, it is critical to handle it efficiently, thus ensuring the reliability of cloud systems.

Node failure prediction. Figure 1 gives the overview of node failure prediction. Node failure prediction is important for improving service reliability [17], since it allows us to take mitigation measures like VM allocation or VM live migration [10] if a node is about to fail. To track the status of nodes, many monitoring metrics are collected in Microsoft 365, which are of time-series format. Previous research has discovered that comparable patterns, such as spike or surge, of the monitoring metrics exist before node failure [40, 44], making it possible to predict the node failure using the monitoring metrics. These patterns, on the other hand, are difficult to characterize just based on the human effort by setting some static thresholds of the monitoring metrics. To tackle this prediction problem, deep learning based approaches are adopted and well performed in the previous studies (see Section 4.2), because they have a high capacity to learn from the historical data distribution.

Challenges in real practice. To predict node failures in real-time, prediction models consume online monitoring metrics using a sliding window in Microsoft 365. Multiple node monitoring metrics are considered as features of the prediction models in a sliding window. These models have been fine-tuned based on historical data, yet they still fail to deliver satisfactory performance in the online situation. Based on our analysis, we find that the quality of monitoring metrics is not sufficient, that is, some values of the features are missing or invalid at a certain time. The high missing percentage, especially in the online setting due to data delay, results in the low performance of prediction models (see Section 3.1).

3 AN EMPIRICAL STUDY ON MISSING DATA IN CLOUD NODE FAILURE PREDICTION

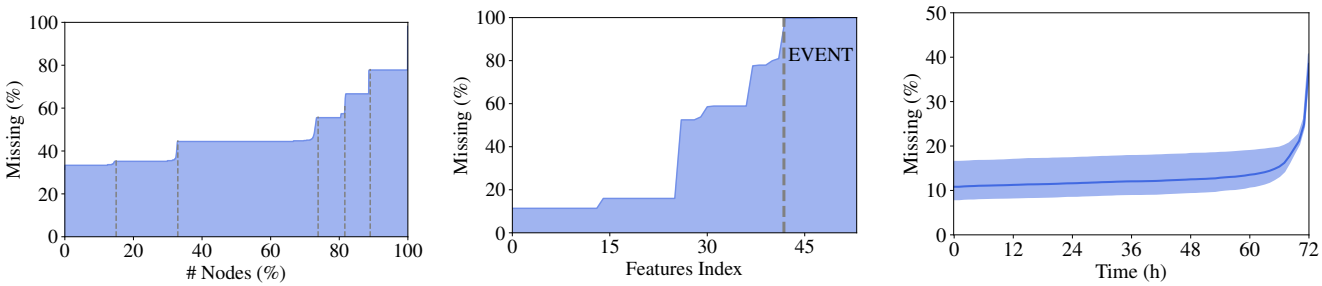
We undertake the first empirical study on a real-world cloud system to better understand missing data and its influence on node failure prediction. We conduct a study on millions of nodes as subjects in a large-scale cloud system, Microsoft 365. These subjects cover many well-known products, *e.g.*, Microsoft 365, which are used by millions of users worldwide. These subjects are from many application fields and developed by different product groups, demonstrating their diversity. The size of collected monitoring metrics is up to 7 GB. To our best knowledge, this is the most large-scale study that explores the data missing problem in the literature. According to the policy of Microsoft 365, We hide some details in this paper, such as the specific time period for node data collection and the specific number of collected node failures. Based on the maintenance records, these nodes are labeled as normal or failure by domain experts. In the study, we address the following research questions:

- **RQ1**: What are the characteristics of missing data in large-scale cloud systems?
- **RQ2**: What are the root causes of missing data?

Table 1 shows the monitoring metrics of a node. The monitoring metrics can be classified into six types, including operation timer, physical characteristics, operation counter, error counter, logical input/output, and EVENT data. The table contains a more extensive discussion of these types and examples. These monitoring metrics are collected in the same regular time interval, except for the EVENT data. The prediction model adopts a sliding window to process these metrics as a feature vector. If any value in the feature vector is empty, data is missing from the model's perspective. The *missing percentage* is a relevant statistics through normalization.

3.1 RQ1: Characterizing the Missing Data

We further investigate the missing data distribution among nodes, within features, and over time.



(a) Among nodes (b) Within features (c) Over time

Figure 2: Characterizing missing data

Among nodes. The missing percentage among nodes is illustrated in Figure 2(a), where the node index is ranked by its missing percentage. We can observe that a lot of nodes have the same missing percentage, for example, the majority of nodes have a missing percentage of less than 40%. This may be caused by the same root causes, such as overload or monitoring error (see Section 3.2). Furthermore, based on their missing percentage, these nodes may be grouped into six groups (separated by the dashed lines), indicating that the missing data problem can be treated uniformly according to the limit group of nodes.

> **Finding 1:** The majority of node monitoring metrics have the same missing percentage, indicating similarity among nodes.

Within features. Prediction models adopt 54 monitoring metrics as features, including time series features and 12 EVENT features, as shown in Table 1. We further investigate the missing percentage according to features. Figure 2(b) show the missing percentage of features. The missing percentage for time series features and EVENT features differs dramatically. For example, feature indexes from 1 to 13 have the missing percentages of less than 10%, while feature indexes 43 to 54, *i.e.*, EVENT features have the missing percentage of near 100%. Because EVENT features are only collected when certain events occur, not on a regular basis. It is not literally missing, but the value is missing from the perspective of the prediction model, which we called "missing by design" in Section 3.2.

> **Finding 2:** Because node monitoring metrics, *i.e.*, time-series and EVENT are different in missing percentage, separate missing data handling strategies are required.

Over time. Figure 2(c) shows the average and max-min bounds of the missing percentage in the 72-hour time window. The prediction model uses a 72-hour-long sliding window to predict the status of the node. Therefore, the hour 72 is the latest time. From this figure, we find that the missing percentage rises steadily before hour 60, with an average missing percentage of 10 to 13.3. After that, it increases dramatically, from 13.3 to 38.5. That is, the missing percentage significantly increases in the latest time. Engineers say that this is due to a time delay in data transmission from the node measurement to the centralized database. Monitoring metrics in the latest time contain additional information (*e.g.*, high I/O utilization) to help identify the failure nodes but is likely missing. Therefore, data missing is a severe problem for the prediction models in the online scenario.

> **Finding 3:** Missing percentage dramatically increases in the latest time due to data delay, making prediction models more difficult to predict.

3.2 RQ2: Root Causes of Missing Data

Based on manual recording and the domain knowledge of experienced engineers, missing data can be divided into four categories: *data delay*, *monitoring error*, *overload*, and *by design*. Note that measuring the percentages of these root causes is challenging, because it is difficult to track all these missing data situations.

Data Delay. The failure prediction model consumes real-time data in a centralized storage system to make the prediction. This data is distributedly collected on each machine, then transmit to the centralized storage system. The data may delay by minutes or even hours due to the network issues, synchronizing strategy, etc [20].

Overload. Some nodes suffer from high I/O stress, making them overload to collect monitoring metrics. The missing data caused by overload indicates these nodes are unhealthy.

Monitoring Error. Some missing data is caused by the problems of data collection or transmission in the monitoring system [33]. To better understand monitoring errors, we further classify them into the following four categories based on the error types: (i) *Network error*: the system encounters a failed network connection, thus the monitoring metrics are missing. (ii) *File system error*: the system encounters a failed file system operation. (iii) *Out of resource*: system resource, such as memory space, is used up. (iv) *Untimely interrupt*: the monitoring system is interrupted by some unintended events.

By Design. This category of missing data is produced intentionally. Some features are not collected in a periodical way, such as the EVENT data. These features are collected only when certain events occur, indicating that the system has reached a specified state. Because the prediction models need a fixed length of features

as input. These features are also concatenated with the periodical feature [22]. Therefore, we name that these data are missing in a by design way.

4 AN EMPIRICAL EVALUATION OF MISSING DATA INTERPOLATION

To solve the missing data problem for node failure prediction in cloud systems, we conducted the first empirical study to investigate *how existing data interpolation techniques perform in this context*. In this section, we first introduce some typical missing data interpolation techniques in Section 4.1, then illustrate our study design in Section 4.2. Finally, we answer the following three research questions in Section 4.3, Section 4.4 and Section 4.5, respectively.

- **RQ3**: Does missing data affect node failure prediction?
- **RQ4**: How well do data interpolation approaches perform?
- **RQ5**: What is the impact of online data delay?

4.1 Data Interpolation Approaches

In the literature, many data interpolation techniques have been discussed. We classified them into four categories according to the technical aspects they use, *i.e.*, Simple filling, numerical analysis, statistical approach, and machine learning based approach. We select one or two typical and widely used techniques from each category as the representatives to evaluate the effectiveness of missing data interpolation for node failure prediction.

- **Simple filling**
  Zero filling (Z) [27]: Zero filling replaces the missing values by zero. Because the normal value of monitoring metrics seldom equals zero, filling in the missing one with zero may make it different from the normal value and result in noise.
  Forward filling (F) [1]: Forward filling is another simple filling approach but considering the context information. It replaces the missing value with the nearest forward non-empty value. Note that the backward filling is not accepted in this scenario since we cannot guarantee that the new arriving value is not empty.
- **Numerical analysis**
  Linear interpolation (L) [30]: Linear interpolation is based on the numerical analysis technique, which interpolates the missing value by linear curve fitting.
- **Statistical approach**
  Average value interpolation (A) [35]: Average value interpolation is a statistical approach that calculates the average value of the remaining normal values and fill the missing positions by the average value. This type of interpolation considers the distribution of the original data source.
  Gaussian distribution sampling interpolation (G) [29]: Gaussian distribution sampling interpolation is based on the assumption that the data follows the Gaussian distribution. The parameters of the distribution are first estimated using the maximum likelihood estimation. Then it randomly chooses samples from this distribution to fill the missing values.
- **Machine learning based approach**
  KNN based interpolation (K) [7]: KNN based interpolation adopts the KNN algorithm to find the closest similarity value to the missing data and use it to fill in the missing value. More specifically, we first use the average interpolation to fill in the missing values and then use all features of the node to build a KD-tree to get the nearest neighbor node of the missing value. Finally, the missing value is updated by the average value of the nearest k neighbors.
  Deep learning based interpolation(D) [4]: Deep learning based interpolation approaches, combined with self attention mechanism [43], are powerful to impute time series data. We expect that by making the failure prediction model aware of the missing positions, the model will pay more attention to the actual values. Therefore, we calculate all the missing places of the input monitoring metrics before they are filled, and use the positions as an attention mask during training.

Specific design. According to *Finding 2*, we should treat the EVENT data according to its physical characteristic because it is missing by default. For the EVENT data, we use zero filling because zero indicates that no event occurred at that moment. For the other time-series features, We use the above seven approaches to interpolate the missing data.

Interpolation dimensions. These interpolation approaches may fill in the missing value based on the neighbor value given a dataset. To identify the neighbor value for node monitoring metrics, two dimensions, *i.e.*, time and node, should be considered. Taking forward filling for example, on one hand, we fill in the missing value by its earlier time given a certain feature of the node using time dimension interpolation by forward filling, denoted as F_t. That is, we use time dimension interpolation to account for temporal similarity. On the other hand, we fill in the missing value by its nearest node in the same cluster given a certain feature and timestamp in node dimension interpolation by forward filling, denoted as F_n. The node dimension interpolation makes use of the spatial similarity. For other approaches, such as linear interpolation, average value interpolation, and Gaussian distribution sampling interpolation. They can both fill the missing data in the time dimension and the node dimension.

Note that zero filling and Deep learning based approach is independent of dimension. Besides, KNN is only suitable for interpolation on node dimension because it fills in the missing data with a similar node. For the other approaches, we conduct experiments on both time dimension and node dimension.

4.2 Study Design

Prediction Models. In this study, we present four widely-used and state-of-the-art techniques for node failure prediction.

- *RNN* [40]: Recurrent Neural Network is a widely-used deep learning model designed specifically for sequential data. In addition, it has been broadly employed for failure prediction.
- *LSTM* [44]: Long Short-Term Memory network is an extension of *RNN*, which can typically handle situations where *RNN* fails.
- *NTAM* [21]: Equipped with an attention mechanism, the transformer neural network achieved state-of-the-art performance in the node failure prediction task.
- *TCNN* [34]: Temporal Convolutional Neural Network is also adopted in node failure prediction to extract the complex dependencies of the monitoring metrics.

Datasets. We utilize the node monitoring metrics dataset mentioned in Section 3. In particular, we divide the dataset into training set, validation set, and test set by 6:1:1 according to the time order, which is also adopted in previous studies [17]. We use grid search to tune the parameters based on the validation set and report the results of the model on the test set. We do not use cross-validation to evaluate these prediction models, because using future data for training is impractical [21].

Implementations. We put these deep learning models into practice by following the instructions in the previous work. We use the implementations supplied by PyTorch 1.10.0 [28], a scalable deep learning framework. We use the identical parameter values as in the previous works for the compared approaches. We utilize grid search to determine the values of parameters based on the best F1-score in the validation set if it is not explicitly indicated in the previous work. We use the *torch.nn* module in PyTorch to build our deep learning models. Specifically, we set the parameters of the models as follows. If there's no specific description, the parameters are set as default.

- *RNN*: The RNN model uses a 2-layer bidirectional RNN module with *hidden_size=64*. We also apply dropout layers with a dropout rate equals to 0.25 to avoid over-fitting.
- *LSTM*: The LSTM model has almost the same structure as the RNN model. We use a 2-layer bidirectional LSTM model instead of the RNN module.
- *NTAM*: The NTAM model uses a one-layer Transformer encoder of three heads with hidden dimensions equal to 64. Three FC layers instead of one are used for prediction in order to avoid information missing.
- *TCNN*: The TCNN model uses a convolutional network as the backbone network structure. We use 2-dimensional convolutional layers with kernel size 3 and Maxpooling layers with kernel size 2. We also use *BatchNorm* layers and *ReLU* activation layers.

We train the model with a learning rate equals to 1e-4. The loss function is the cross Entropy loss [8], the optimizer is Adam with weight decay equals 5e-6. Each model is trained 100 epochs separately. Our study is conducted on Linux Ubuntu 20.04 LTS with 16 cores 32 threads AMD Epyc 7V12 3.9GHz and 256MB Cache, 220 GB memory, 64-bit operating system, and a single NVIDIA Tesla T4 GPU accelerator.

Metrics. Models determine whether the node will fail based on the features of node monitoring metrics in a time window. To evaluate the performance of models, we employ the widely used metrics Precision, Recall, F1-score, and Balanced Accuracy. Nodes are labeled as normal or failure based on maintenance records. True Positive (TP) is how many nodes predicted failure are correct, and True Negative (TN) is the number of nodes correctly predicted normal. False Positive (FP) refers to the number of nodes that predicted failure but turned out to be normal. False Negative (FN) is the number of nodes predicted normal but is actually failed.

- *Precision*: Precision measures how many actual failure nodes are predicted as failure nodes correctly by the models.

$$Precision = \frac{TP}{TP + FP}$$

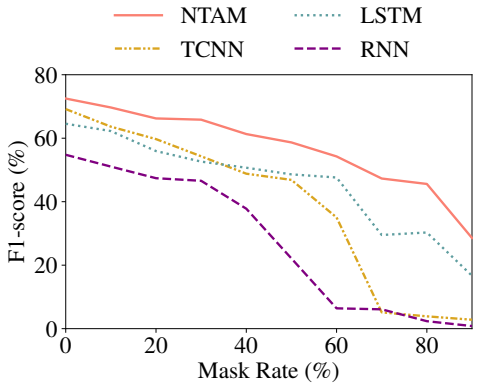


Figure 3: Impact of Data Missing

- *Recall*: Recall represents the ability to predict failure nodes, *i.e.*, how many failure nodes can be predicted by the models.

$$Recall = \frac{TP}{TP + TN}$$

- *F1-score*: F1-score is a metric that is the harmonic mean of precision and recall.

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

- *Balanced Accuracy*: Balanced accuracy is especially useful when the classes are imbalanced [16]. Balanced accuracy is calculated as:

$$BalancedAccuracy = (\frac{TP}{TP + FN} + \frac{TN}{TN + FP})/2$$

- *Efficiency*: We also investigate the efficiency of the studied data interpolation techniques, *i.e.*, the interpolation time of the training set and test set.

4.3 RQ3: Impact of Data Missing

To explore the performance of node failure prediction affected by data missing, we conduct experiments under different missing rates. We cannot obtain the oracle dataset without missing data, since the original data from the online pipeline is partially missing. As a result, for both the train and validation datasets, we mask the non-missing locations at various missing rates (the missing percentage of the original dataset is the basis) at random. To ensure the robustness, we compare the experiments under several approaches, *i.e.*, *LSTM*, *RNN*, *NTAM*, and *TCNN*, and utilize forward filling method to fill missing values.

Figure 3 shows the F1-score of multiple models with the data mask rate from 0 to 90%, where zero mask rate corresponds to the original data and 90% signifies that 90% of the original data is masked. We can use masking to simulate different missing rates and assess the model performance on the masked dataset. We can draw some conclusions based on Figure 3. First, the performance of the four techniques degrades substantially as the missing rate rises. The F1-score of node failure prediction declines by 57.33% at a missing rate of 20%, 49.66% at a missing rate of 40%, 35.83% at a missing rate of 60%, and 20.53% at a missing rate of 80%. The

Table 2: Performance comparison of various node failure prediction models with different interpolation approach, *i.e.*, *Z*ero, *F*orward, *A*verage, *L*inear, *G*aussian, *K*NN, and *D*eep learning. The subscript t (n) denotes interpolation in the time (node) dimension, and Avg_t (Avg_n) is the average value of the time (node) dimension's interpolation results.

Metric	Model	Z	F_t	F_n	A_t	A_n	L_t	L_n	G_t	G_n	K_n	D	Avg_t	Avg_n	Avg
F1-score	*RNN*	55.94	57.83	53.33	50.62	59.74	52.50	58.44	57.33	**60.65**	55.70	52.56	54.57	**58.04**	55.88
	LSTM	64.67	64.56	69.57	62.89	66.67	64.56	66.67	66.67	69.14	67.88	**71.43**	64.67	**68.01**	66.79
	NTAM	65.50	71.95	72.84	71.43	**74.51**	70.11	69.46	71.43	72.84	73.86	66.67	71.23	**72.41**	70.96
	TCNN	65.79	71.26	**73.37**	65.82	70.66	69.62	72.05	68.57	72.83	50.63	70.73	68.82	**72.23**	68.30
	Average	62.98	66.40	67.28	62.69	67.90	64.20	66.66	66.00	**68.87**	62.02	65.35	64.82	**67.67**	65.48
Precision	*RNN*	**65.57**	57.14	58.82	51.25	63.89	53.85	62.50	63.24	64.38	57.89	55.41	56.37	**62.40**	59.45
	LSTM	63.53	67.11	67.09	62.50	**70.27**	67.11	65.12	**70.27**	70.00	67.47	69.77	66.75	**68.12**	67.29
	NTAM	62.92	71.95	73.75	76.39	**80.28**	66.30	68.24	76.39	73.75	69.15	70.27	72.76	**74.01**	71.76
	TCNN	71.43	67.39	71.26	68.42	69.41	72.37	**73.42**	64.52	69.23	52.63	70.73	68.18	**70.83**	68.26
	Average	65.86	65.90	67.73	64.64	70.96	64.91	67.32	68.61	**69.34**	61.79	66.55	66.02	**68.84**	66.69
Recall	*RNN*	48.78	**58.54**	48.78	50.00	56.10	51.22	54.88	52.44	57.32	53.66	50.00	53.05	**54.27**	52.88
	LSTM	65.85	62.20	64.63	60.98	63.41	62.20	68.29	63.41	68.29	68.29	**73.17**	62.20	**66.16**	65.52
	NTAM	68.29	71.95	71.95	67.07	69.51	74.39	70.73	70.73	71.95	**79.27**	63.41	**71.04**	**71.04**	70.84
	TCNN	60.98	75.61	75.61	63.41	71.95	67.07	67.07	73.17	**76.83**	48.78	70.73	69.82	**72.87**	68.29
	Average	60.98	67.08	65.24	60.37	65.24	63.72	65.24	64.94	**68.60**	62.50	64.33	64.03	**66.09**	64.38
Balanced Accuracy	*RNN*	74.34	**79.18**	74.32	74.91	77.99	75.53	77.38	76.16	78.60	76.75	74.92	76.45	**77.07**	76.37
	LSTM	82.85	81.04	82.26	80.42	81.66	81.04	84.08	81.66	84.09	84.08	**86.52**	81.04	**83.02**	82.70
	NTAM	84.07	85.92	85.93	83.50	84.72	87.12	85.30	83.50	85.93	**89.57**	81.66	85.01	**85.47**	85.20
	TCNN	80.44	87.73	87.75	81.65	85.91	83.49	85.32	86.51	**88.35**	74.31	85.31	84.85	**86.83**	84.25
	Average	80.43	83.47	82.57	80.12	82.57	81.80	83.02	81.96	**84.24**	81.18	82.10	81.84	**83.10**	82.13

results show that missing data reduces the accuracy of node failure prediction.

Second, The impact of missing data on different models varies. With an 80% missing rate, *TCNN* and *RNN* suffer the most, with their F1-scores dropping to less than 5%. *NTAM* is more consistent, with 40% F1-score at the 80% missing rate. *NTAM* also performs substantially better than *TCNN* at the basis. This implies that a superior approach, such as *NTAM*, is more data missing resistant.

> **Finding 4:** Data missing has a negative impact on the performance of the node failure prediction model, which motivates us to explore solutions to handle it.

4.4 RQ4: Investigation of Interpolation Approaches

4.4.1 Performance Comparison. To explore the performance of different interpolation approaches illustrated in Section 4.1, we conduct experiments on the subjects, and the results are shown in Table 2. In this table, the last three rows present the average performance metrics (*i.e.*, F1-score, Precision, Recall, and Balanced Accuracy) of time dimension approaches, node dimension approaches, and all approaches on the four models, respectively. Each column is an interpolation approach, and the last column shows the average performance of each model. The bold value for each metric in each row refers to the largest one among these interpolation approaches. According to Table 2, we have the following findings.

Model comparison. According to the last column of this table, *NTAM* performs best among four approaches, with the average F1-score of 70.96% for various interpolation approaches. *RNN* gets the worst performance, with the average F1-score of 55.88%. Because *NTAM* is based on the Transformer model, which is the state-of-the-art sequential model adopted in various applications, such as time series prediction, and natural language translation. It is more effective in terms of the network structure than the traditional *RNN* network. The *LSTM* model, on the other hand, has comparable performance and is widely used in the industry application since it is easy and reliable to apply [17].

No suitable interpolation approach for all models. According to the table, Gaussian distribution sampling interpolation from the node dimension, deep learning based interpolation, average value filling from the node dimension, and forward filling from the node dimension is the best interpolation algorithm for *RNN*, *LSTM*, *NTAM*, and *TCNN*, respectively. Their F1-scores range from 50.62% to 74.51%, demonstrating that different interpolation approaches have distinct effects on model performance.

Node dimension interpolation is superior to time dimension. The average F1-score for the node dimension is 67.67%, while that for the time dimension is 64.82%. On average, node dimension interpolation exceeds time dimension interpolation by about 3%. For example, based on *LSTM*, node dimension forward value filling, average value filling, linear interpolation approach, and Gaussian distribution sampling interpolation exceeds the corresponding time dimension based ones by 5.01%, 3.78%, 2.11%, and 2.47% in F1-score,

respectively. One probable explanation is that node dimension techniques take into account similarity among nodes (*e.g.*, forward filling by filling in the missing value using the nearest node), as shown in our Finding 1, and we will present case studies in Section 5. Similarly, previous studies [21] have used neighborhood information of hard drives to improve disk failure prediction performance.

Zero filling. Zero filling performs poorly when compared to other interpolation algorithms, despite its ease of implementation. This means that failing to use information from context timestamps or neighbor nodes could result in poor results.

Learning based interpolation. The performance of KNN interpolation and deep learning based interpolation approaches are not effective as expected. For example, the average F1-scores are 62.02%, 65.35%, below the average (65.79%). That further demonstrates the effectiveness of these well-studied interpolation techniques is discounted in our node failure context.

Different metrics may show different things. Although node dimension interpolation performs better than time dimension interpolation in all metrics, *i.e.*, F1-score, Precision, Recall, and Balanced Accuracy, we find that the best interpolation approaches for each performance metric in each row are not the same, because different metrics reflect different aspect. Distinct service teams have different metrics preferences. In a user interactive core service, for example, recall is more important than precision since operators do not want to overlook any potential node failure that could severely affect the user experience. As a result, depending on the actual needs of services, we can pick between metric and interpolation approaches.

> **Finding 5:** No existing interpolation approach is suitable for all prediction models, however, the node dimension interpolation approaches outperform time dimension ones by 3% on average.

4.4.2 Efficiency Comparison. To explore the efficiency of these interpolation approaches, we calculate the running time of these approaches, including the training phase and online test phase, and the results are shown in Table 3.

Table 3 shows the total interpolation time in the training set and the online interpolation time for each node, because the training set can be processed as a whole in the training phase, the online test needs to make predictions for each node. From this table, we have the following findings. First, for both the train set and the online test, the zero interpolation approach run the fastest, indicating that the strategy is simple and efficient. Second, the KNN interpolation strategy in train sets consumes significantly more time than other approaches. KNN is not suitable for imputation for missing data in real practice due to the trade-off between performance and efficiency. Third, the online test time of other approaches is less than 2 ms, making them appropriate for online use according to the domain experts from the production team.

> **Finding 6:** In the online prediction scenario, all interpolation approaches are efficient except KNN based interpolation.

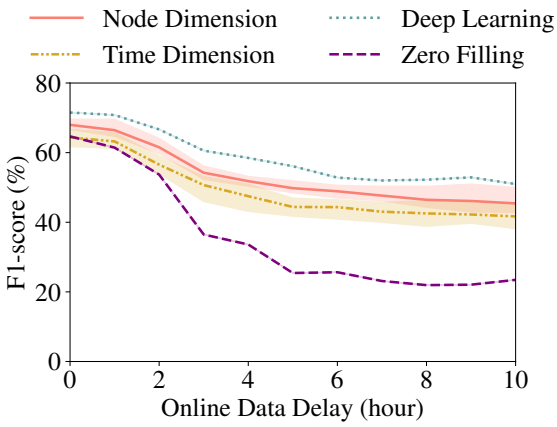


Figure 4: The F1-score of varies interpolation approaches with online data delay. The average with min/max boundary of F1-score in different approaches (Forward, Average, Linear, Gaussian, and KNN) is shown by the node and time dimensions.

4.5 RQ5: Impact of Online Data Delay

As we described in Section 3.2, data delay is one of the major reasons for missing data in the node failure prediction scenario. Because the monitoring metrics collected from each node are written to the centralized database in an asynchronous way, it may take several hours in practice. The delayed data, obviously, cannot be used by real-time models, resulting in missing data. However, there is delayed data when we collect data from a database for offline training. As a result, the true rate of missing online data may be higher than what we observe from the data.

To simulate the higher missing rate in an online context than we observed from the historical dataset, we mask data from the last several hours (from 0 to 10) in the test set without affecting the missing rate of the training dataset. This experiment can also evaluate how quickly a model can predict node failure in advance without obtaining the future data. Using different interpolation approaches, Figure 4 shows the F1-score of the *LSTM* model when faced with different online data delays. Other models reach the same conclusion, however, due to space constraints, we exclude their results. The horizontal axis denotes the number of data delay hours, and the vertical axis denotes the F1-score value. We have the following observations according to this figure.

First, in comparison to no delay and one-hour delay, the F1-score of all approaches is practically steady. After that, regardless of whether interpolation approaches are used, the performance of node failure prediction reduces significantly when more online data is delayed, demonstrating once again that the data missing issue hurts the performance of node failure prediction.

Second, node dimension interpolation approaches outperform time dimension interpolation approaches, as we see in Finding 5. The deep learning based interpolation technique, on the other hand, outperforms all other data interpolation approaches. One possible explanation is that the deep learning based interpolation technique uses self-attention to examine the node and temporal information

Table 3: Efficiency comparison of different interpolation approaches

Method	Zero	F_t	F_n	A_t	A_n	L_t	L_n	G_t	G_n	K_n	D
Total Training Set (s)	3.70	4.34	8.24	30.94	19.09	73.10	51.13	100.40	31.93	62930.00	1648.48
Online Test / Node (ms)	0.03	0.02	0.05	0.21	0.06	0.7	0.09	1.28	0.2	79.37	0.78

of missing data, which is appropriate in a data delay scenario and valuable for early prediction.

> **Finding 7:** Deep learning based interpolation approach performs the best in the online data delay scenario, highlighting its capabilities for early prediction.

5 DISCUSSION

In this section, we present case studies of the missing data. Next, we discuss insights for handling missing data in node failure prediction. Then, we summarize lessons learned from the real-world deployment of the node failure prediction solution in Microsoft 365 Cloud Systems. Finally, we discuss threats to validity.

5.1 Case studies

To illustrate the effectiveness of node dimensions interpolation approaches, we present several cases in our datasets. Figure 5 shows one feature of seven sampled nodes across the timeline. In this figure, the blue bar is the position of the missing value. The ground truth of node status, *i.e.*, failure or normal is presented on the right.

From this figure, the missing data of Node 1 and Node 2 are complementary to one other over time, and they both belong to the failure nodes. Nodes 3 and 4, as well as Nodes 5, 6, and 7, are normal nodes and exhibit the same behavior. As a result, using the neighborhood node value to interpolate the missing value makes sense. Furthermore, having a stronger interpolation technique will help with failure prediction.

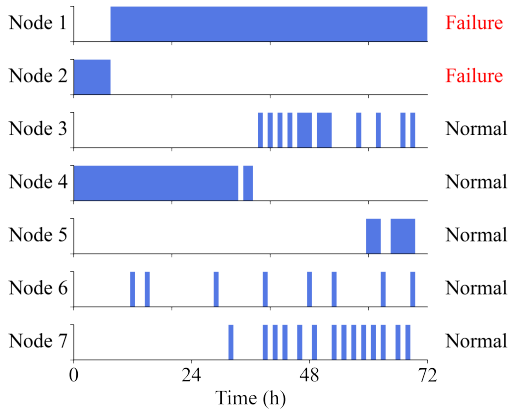


Figure 5: Data missing case studies. The Blue bar indicates that the value is missing at that time.

5.2 Insights in Handling Missing Data

Temporal and spatial characteristic of data in Cloud. Cloud systems, with computing resources distributed all over the world, generate spatial-temporal data by nature. Different from existing data missing handling research, our empirical study shows the effect of data sufficiency in both temporal and spatial (node) dimensions, and the latter may be more important to node failure prediction. This finding also conforms to the study in disk failure prediction [21]. More specifically, *NTAM* makes use of node neighborhood knowledge by attention mechanism in the transformer model. Therefore, we need to pay attention to the characteristic of data in the spatial dimension when developing data-driven solutions for cloud systems.

Generality of the study. Missing data is a severe problem in the real world, not only in the scenario of node failure prediction. For example, we analyze the BackBlaze dataset[2] and the Alibaba Cloud dataset[3]. The S.M.A.R.T.[4] feature value for each timestamp is provided in these datasets, and missing data is detected by the empty value of the timestamp. BackBlaze is a published hard driver dataset from a real-world cloud storage provider, which has about 1.01% failure rate of hard drivers according to their 2021 report[5]. Analysis of about 500 nodes over one month suggests that the missing rate (NaN / total value) is 74.4%. Alibaba Cloud dataset is served as PAKDD2020 Alibaba AIOps Competition, which provides S.M.A.R.T. data from 200 thousand hard disk drives over one year in Alibaba Cloud's data centers. The total missing rate is about 91%. These high data missing rates indicate that missing data is a general challenge for node failure prediction in the real world. We believe our findings may also guide these scenarios in handling missing data because they have the same temporal and spatial characteristics. In future work, we will address the missing data problem for the general failure prediction task for software systems.

5.3 Lessons Learned

Solving data missing from source. Section 4.4 shows the feasibility of implementing interpolation methods to handle data missing. However, these interpolation methods still can not fully solve the missing problem, since true data missed is hard to restore completely correctly. The most straightforward solution is to solve data missing from the source. In other words, ensure the data collection process. As illustrated in Section 3.2, reasons leading to data missing in node failure prediction can be mainly divided into four categories: *data delay*, *monitoring error*, *overload*, and *by design*. The

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

[3]https://tianchi.aliyun.com/dataset/dataDetail?dataId=70251

[4]https://en.wikipedia.org/wiki/S.M.A.R.T.

[5]https://www.backblaze.com/blog/backblaze-drive-stats-for-2021/

root cause for missing data in other scenarios is also similar to one in node failure prediction. Although it is hard to completely solve the four issues to obtain data without missing, it is supposed to optimize the storage system, keep more stable networking, ensure more reasonable source allocation, and design a more effective data format to alleviate data missing as much as possible. We believe it is the best solution to handle data missing in real practice.

Missing maybe also an information. Instead of alleviating the data missing issue, another thought is utilizing data missing for node failure prediction. According to our investigation, a high missing rate may be a strong signal to indicate nodes' failure. More specifically, the average of the top 0.1% missing rate from normal nodes and failed nodes are 77.82% and 87.03%, respectively. Because node failure might result in a large amount of missing data. For example, a node's I/O may be overloaded, preventing it from collecting monitoring metrics and just reporting EVENT data. We believe that the missing distribution of each node can be treated as additional information for failure prediction. In addition, suddenly continuous data missing may also indicate a failure. In other scenarios involved with data missing, we can also correlate data property with the missing issue, and explore the chance to guide corresponding downstream tasks via missing data itself.

5.4 Threats to Validity

Internal Threats. The implementations of our method and the compared approaches pose an internal threat to validity. To reduce this threat, we employ established tool implementations for the different deep learning models used in our work, which are described in Section 4. Two authors also double-checked the code.

External Threats. The subjects are the main external threats to validity. We collect millions of nodes' monitoring metrics from the Microsoft 365 for this research. These nodes come in a variety of models and from different manufacturers, making them diverse. Nonetheless, the subjects utilized may not be representative of node failures in other companies. We will look into additional node failures from other companies in the future.

Construct Threats. Construct threats are mainly parameters and labeled data. For the parameters in the compared approaches, we set them using the values given in the previous work or grid search. We will further investigate the impact of parameters in the future. The failure nodes were labeled by failure records. Despite the fact that these data may contain noise, we feel Microsoft 365's failure management is mature and attentive. Thus, this threat is negligible.

6 RELATED WORK

Failure prediction. Many approaches for predicting node failure, similar to hard drive disk failure prediction, have been proposed in recent years. Machine learning or deep learning based techniques are commonly used because node failure prediction may be considered as a binary classification problem [19]. Machine learning based approaches for node failure prediction, such as support vector machines [44] and tree based models [3, 15], use monitoring metrics collected in a time window to predict whether the node will fail in the coming time window. These machine learning approaches, on the other hand, poorly handle the temporal information of nodes [34]. Deep learning approaches such as the recurrent neural network (RNN) [40], long short-term memory (LSTM) [17], and temporal convolutional neural network (TCNN) [34] can capture the temporal information and outperform classical machine learning based approaches. Our work is orthogonal to previous failure prediction approaches, since we aim to understand the missing data issue of node failure prediction.

Missing data interpolation. Besides the seven studied approaches in our work, there are many missing data interpolation methods [32]. For example, 3-D autoregressive models and Markov random fields are used to interpolate missing data in image sequences [14]. A Singular Value Decomposition (SVD) based method, weighted K-nearest neighbors, and row average are evaluated for DNA microarrays missing interpolation [35]. Low-rank recovery and semi-supervised regression are adopted for software effort estimation [12], since the effort data missing occurs in real-world data collection. Different from the above-mentioned related work, our work conducts an empirical study to evaluate the effectiveness of typical data interpolation approaches in cloud node failure prediction.

Missing data in the software systems. Over the years, many missing data handling approaches have been proposed. For example, query-oriented data cleaning with Oracles [2] can add a missing answer. Y! [39] diagnoses missing events in distributed systems with negative provenance. Rex [26] and BDA [31] detect missing files in the configuration using correlated change analysis. GRAPE [42] proposes a general framework for feature imputation and label prediction in the presence of missing data. Missing data is also a significant issue in general software analytics, since data is the foundation of the data-driven approach. Previous software analytics studies [13] deal with missing data in software defect prediction. An automatic missing data recovery algorithm called ReLink [38] can help construct quality defect datasets for follow-up research such as software defect prediction and software maintenance. Unlike the above work, we use industry data to conduct an empirical study of missing data in the failure prediction scenario and investigate existing interpolation methods for dealing with missing data.

7 CONCLUSION

Node failure prediction is an important task in cloud systems for ensuring system reliability. When applying node failure prediction in practice, we notice a problem with the missing data of node monitoring metrics. We conduct a large-scale real-world empirical study on millions of nodes in Microsoft to better understand the missing data. From this study, we find that missing data is a severe problem, especially in the online setting. It may result in poor node failure prediction performance. We have seven interesting findings, such as node dimension interpolation approaches outperform time dimension ones, and deep learning based interpolation is the best for early prediction. Our findings could help tackle the missing data problem in cloud node failure prediction and other data-driven software engineering scenarios.

ACKNOWLEDGMENTS

We would like to thank the strong support from Dongmei Zhang. Hongyu Zhang's work was partially supported by Australian Research Council Discovery Projects (DP200102940, DP220103044).

REFERENCES

[1] K. Mohaideen Abdul Kadhar and G. Anand. 2021. *Preparing the Data.* Apress, Berkeley, CA, 99–119.

[2] Moria Bergman, Tova Milo, Slava Novgorodov, and Wang-Chiew Tan. 2015. Query-oriented data cleaning with oracles. In *Proceedings of the Conference on Management of Data (SIGMOD).* ACM, 1199–1214.

[3] Mirela Madalina Botezatu, Ioana Giurgiu, Jasmina Bogojeska, and Dorothea Wiesmann. 2016. Predicting Disk Replacement towards Reliable Data Centers. In *Proceedings of the Knowledge Discovery and Data Mining (SIGKDD).* ACM, 39–48.

[4] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. 2018. Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems* 31 (2018).

[5] Chengliang Chai, Lei Cao, Guoliang Li, Jian Li, Yuyu Luo, and Samuel Madden. 2020. Human-in-the-loop outlier detection. In *Proceedings of the Conference on Management of Data (SIGMOD).* ACM, 19–33.

[6] 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 the International World Wide Web Conferences (WWW).* ACM, 2659–2665.

[7] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory (TIT)* 13, 1 (1967), 21–27.

[8] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of operations research* 134, 1 (2005), 19–67.

[9] Supratim Deb, Zihui Ge, Sastry Isukapalli, Sarat Puthenpura, Shobha Venkataraman, He Yan, and Jennifer Yates. 2017. Aesop: Automatic policy learning for predicting and mitigating network service impairments. In *Proceedings of the Knowledge Discovery and Data Mining (SIGKDD).* ACM, 1783–1792.

[10] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, et al. 2020. Protean: VM allocation service at scale. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI).* USENIX, 845–861.

[11] Shilin He, Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, Michael R Lyu, and Dongmei Zhang. 2018. Identifying impactful service system problems via log analysis. In *Proceedings of the Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE).* ACM, 60–70.

[12] Xiao-Yuan Jing, Fumin Qi, Fei Wu, and Baowen Xu. 2016. Missing data imputation based on low-rank recovery and semi-supervised regression for software effort estimation. In *Proceedings of the International Conference on Software Engineering (ICSE).* ACM/IEEE, 607–618.

[13] Sunghun Kim, Hongyu Zhang, Rongxin Wu, and Liang Gong. 2011. Dealing with noise in defect prediction. In *Proceedings of the International Conference on Software Engineering (ICSE).* IEEE, 481–490.

[14] Anil C Kokaram, Robin D Morris, William J Fitzgerald, and Peter JW Rayner. 1995. Interpolation of missing data in image sequences. *Transactions on Image Processing (TIP)* 4, 11 (1995), 1509–1519.

[15] 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 the International Conference on Dependable Systems and Networks (DSN).* IEEE/IFIP, 383–394.

[16] Zhenhao Li, Tse-Hsun Chen, and Weiyi Shang. 2020. Where shall we log? studying and suggesting logging locations in code blocks. In *Proceedings of the International Conference on Automated Software Engineering (ASE).* IEEE/ACM, 361–372.

[17] Qingwei Lin, Ken Hsieh, Yingnong Dang, Hongyu Zhang, Kaixin Sui, Yong Xu, Jian-Guang Lou, Chenggang Li, Youjiang Wu, Randolph Yao, et al. 2018. Predicting node failure in cloud service systems. In *Proceedings of the Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE).* ACM, 480–490.

[18] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. 2016. Log clustering based problem identification for online service systems. In *Proceedings of the International Conference on Software Engineering Companion (ICSE-Companion).* ACM/IEEE, 102–111.

[19] Yudong Liu, Hailan Yang, Pu Zhao, Minghua Ma, Chengwu Wen, Hongyu Zhang, Chuan Luo, Qingwei Lin, Chang Yi, Jiaojian Wang, et al. 2022. Multi-task Hierarchical Classification for Disk Failure Prediction in Online Service Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 3438–3446.

[20] Sidi Lu, Bing Luo, Tirthak Patel, Yongtao Yao, Devesh Tiwari, and Weisong Shi. 2020. Making Disk Failure Predictions SMARTer!. In *Proceedings of the Conference on File and Storage Technologies (FAST).* USENIX, 151–167.

[21] 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 International World Wide Web Conferences (WWW).* ACM, 1181–1191.

[22] Ao Ma, Rachel Traylor, Fred Douglis, Mark Chamness, Guanlin Lu, Darren Sawyer, Surendar Chandra, and Windsor Hsu. 2015. RAIDShield: characterizing, monitoring, and proactively protecting against disk failures. *Transactions on Storage (TOS)* 11, 4 (2015), 1–28.

[23] Minghua Ma, Zheng Yin, Shenglin Zhang, Sheng Wang, Christopher Zheng, Xinhao Jiang, Hanwen Hu, Cheng Luo, Yilin Li, Nengjun Qiu, et al. 2020. Diagnosing root causes of intermittent slow queries in cloud databases. *Proceedings of the VLDB Endowment* 13, 8 (2020), 1176–1189.

[24] Minghua Ma, Shenglin Zhang, Junjie Chen, Jim Xu, Haozhe Li, Yongliang Lin, Xiaohui Nie, Bo Zhou, Yong Wang, and Dan Pei. 2021. Jump-Starting Multivariate Time Series Anomaly Detection for Online Service Systems. In *Proceedings of the Annul Technical Conference (ATC).* USENIX, 413–426.

[25] Minghua Ma, Shenglin Zhang, Dan Pei, Xin Huang, and Hongwei Dai. 2018. Robust and rapid adaption for concept drift in software system anomaly detection. In *Proceedings of the 29th International Symposium on Software Reliability Engineering (ISSRE).* IEEE, 13–24.

[26] Sonu Mehta, Ranjita Bhagwan, Rahul Kumar, Chetan Bansal, Chandra Maddila, B Ashok, Sumit Asthana, Christian Bird, and Aditya Kumar. 2020. Rex: Preventing bugs and misconfiguration in large services using correlated change analysis. In *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI).* USENIX, 435–448.

[27] Bertrand Muquet, Zhengdao Wang, Georgios B Giannakis, Marc De Courville, and Pierre Duhamel. 2002. Cyclic prefixing or zero padding for wireless multicarrier transmissions? *Transactions on communications* 50, 12 (2002), 2136–2148.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS).* Vol. 32. MIT Press, 8024–8035.

[29] Daniel Peña and George C Tiao. 1991. A note on likelihood estimation of missing values in time series. *The American Statistician* 45, 3 (1991), 212–213.

[30] ALEXANDER L Read. 1999. Linear interpolation of histograms. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 425, 1-2 (1999), 357–360.

[31] Weiyi Shang, Zhen Ming Jiang, Hadi Hemmati, Brain Adams, Ahmed E Hassan, and Patrick Martin. 2013. Assisting developers of big data analytics applications when deploying on hadoop clouds. In *Proceedings of the International Conference on Software Engineering (ICSE).* IEEE, 402–411.

[32] Qinbao Song and Martin Shepperd. 2007. Missing data imputation techniques. *International journal of business intelligence and data mining* 2, 3 (2007), 261–291.

[33] Ming Sun, Ya Su, Shenglin Zhang, Yuanpu Cao, Yuqing Liu, Dan Pei, Wenfei Wu, Yongsu Zhang, Xiaozhou Liu, and Junliang Tang. 2021. CTF: Anomaly Detection in High-Dimensional Time Series with Coarse-to-Fine Model Transfer. In *Proceedings of the International Conference on Computer Communications (INFOCOM).* IEEE, 1–10.

[34] 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 the Design Automation Conference (DAC).* ACM, 20.

[35] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. 2001. Missing value estimation methods for DNA microarrays. *Bioinformatics* 17, 6 (2001), 520–525.

[36] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. 2010. Characterizing cloud computing hardware reliability. In *Proceedings of the Symposium on Cloud Computing (SoCC).* ACM, 193–204.

[37] Wikipedia. 2021. Facebook Outage. (2021). https://en.wikipedia.org/wiki/2021_Facebook_outage.

[38] Rongxin Wu, Hongyu Zhang, Sunghun Kim, and Shing-Chi Cheung. 2011. Relink: recovering links between bugs and changes. In *Proceedings of the Symposium on the Foundation of Software Engineering European Software Engineering Conference (ESEC/FSE).* ACM, 15–25.

[39] Yang Wu, Mingchen Zhao, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. 2014. Diagnosing missing events in distributed systems with negative provenance. *SIGCOMM Computer Communication Review* 44, 4 (2014), 383–394.

[40] 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. *Transactions on Computers* 65, 11 (2016), 3502–3508.

[41] 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 the USENIX Annul Technical Conference (ATC).* USENIX, 481–494.

[42] Jiaxuan You, Xiaobai Ma, Daisy Yi Ding, Mykel J. Kochenderfer, and Jure Leskovec. 2020. Handling Missing Data with Graph Representation Learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, Hugo

Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.), Vol. 33. MIT Press, 19075–19087.

[43] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2019. Self-attention generative adversarial networks. In *Proceedings of the International Conference on Machine Learning (ICML)*. ACM, 7354–7363.

[44] Jianguo Zhang, Ji Wang, Lifang He, Zhao Li, and Philip S. Yu. 2018. Layerwise Perturbation-Based Adversarial Training for Hard Drive Health Degree Prediction. In *Proceedings of the International Conference on Data Mining (ICDM)*. IEEE, 1428–1433.

[45] Qiao Zhang, Guo Yu, Chuanxiong Guo, Yingnong Dang, Nick Swanson, Xinsheng Yang, Randolph Yao, Murali Chintalapati, Arvind Krishnamurthy, and Thomas Anderson. 2018. Deepview: Virtual Disk Failure Diagnosis and Pattern Detection for Azure. In *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, Renton, WA, 519–532.