Can We Trust Auto-Mitigation? Improving Cloud Failure Prediction with Uncertain Positive Learning

Haozhe Li[†], Minghua Ma[‡*], Yudong Liu[†], Pu Zhao[†], Shuo Li[§], Ze Li[‡], Murali Chintalapati[‡], Yingnong Dang[‡], Chetan Bansal[‡], Saravan Rajmohan[‡], Qingwei Lin[†], Dongmei Zhang[†]

[†]Microsoft, Beijing, China
[‡]Microsoft, Redmond, USA

***Abstract*—In the rapidly expanding domain of cloud computing, a variety of software services have been deployed in the cloud. To ensure the reliability of cloud services, prior studies focus on the prediction of failure instances, such as disks, nodes, switches, etc. The mitigation actions are initiated to resolve the underlying issue once the prediction output is positive. However, our real-world practice in Microsoft Azure revealed a decline in prediction accuracy, approximate 9%, subsequent to model retraining. The decrease is attributed to the mitigation actions, which can result in uncertain positive instances. Since these instances cannot be verified after mitigation, they may introduce additional noise into the model updating process. To the best of our knowledge, we are the first to identify this Uncertain Positive Learning (UPLearning) issue in the real-world cloud failure prediction scenario, and we design an Uncertain Positive Learning Risk Estimator (Uptake) approach to address this problem. By utilizing two real-world datasets for disk failure prediction and conducting node prediction experiments in Azure, which is a top-tier cloud provider serving millions of users, we demonstrate that our Uptake method can significantly enhance failure prediction accuracy by an average of 5%.**

***Index Terms*—Failure Prediction, Cloud Systems, PU Learning**

I. Introduction

Failure prediction and mitigation are vital across various domains, such as cloud computing [1]–[6], autonomous driving [7] and automatic control [8], where swift and precise responses are critical for maintaining reliability and efficiency [9]–[14]. By employing real-time data analysis combined with predictive models, these fields can proactively manage potential disruptions, minimizing risks before they escalate into significant issues and ensuring operational continuity and safety against unforeseen incidents [15]–[19].

In the context of cloud computing, similar predictive and mitigation strategies are employed to manage substantial workloads on platforms, such as Microsoft Azure, Amazon Web Services, and Google Cloud Platform. These providers continuously monitor and analyze thousands of metrics across their cloud infrastructures to deliver high-quality service for billions of users worldwide [20]–[24]. These metrics enable the detection and prevention of component failures within the cloud system, such as memory [25], disk [26], [27], nodes [28]–[30], and switches [31]. By employing machine learning and deep learning techniques on these metrics, it is possible to predict failures and proactively mitigate them. This not only boosts the availability and performance of cloud-based software systems but also reduces operational costs and risks.

At the forefront of cloud system reliability is the prediction of node failures, a critical step that enables us to anticipate problems before they fully manifest. Numerous previous studies aim to design better machine learning models to improve the performance of cloud failure prediction tasks. Commonly employed models are used, such as Recurrent Neural Network (RNN) [32], Long Short-Term Memory (LSTM) [33], Transformer [26], and Temporal Convolutional Network (TCNN) [34]. Prediction is the first step to manage the cloud failures. Once a prediction indicates a potential failure, swift mitigation actions are taken automatically to refresh and change the node's status. Failure auto-mitigation here means to attempt to make the failure alert disappear by taking automatic mitigation actions (*e.g.*, refresh the node) after the failure prediction is positive, without necessarily diagnosing and fixing the underlying bugs first. For example, if a computing node is predicted to fail, we might reboot it automatically [35]. This auto-mitigation alters the node's status, however, cannot provide insights into the root cause of the potential failure. Consequently, we cannot be certain whether the predicted failure will occur, which we call an *uncertain positive* instance.

The limitations of these auto-mitigation actions highlight the need for advanced prediction and mitigation techniques. Based on our experience with deploying prediction models for cloud failures in Microsoft Azure, we have observed significant challenges in updating these models during real-world online usage. Model updating is the process of retraining the machine learning model over time (weekly or monthly) to adapt to the changing cloud environment (new hardware and software) [36]. An empirical study conducted on both open datasets and Azure (see Section III), suggests that the prediction accuracy of updated models may decrease by about 9% over time because the uncertain positive instances bring noise to the model updating. This effect is compounded when using a continuously updated model that accumulates uncertain positive instances over time.

More specifically, Fig. 1 illustrates a toy example of the model updating scenario in the cloud failure prediction. The figure has two parts, *i.e.*, the *offline updating* and the *online updating*. The offline updating is an ideal scenario in which we can access the oracle of cloud instance status (shown as

* Corresponding author. Email: minghuama@microsoft.com
§ Work was performed when he was at Carnegie Mellon University

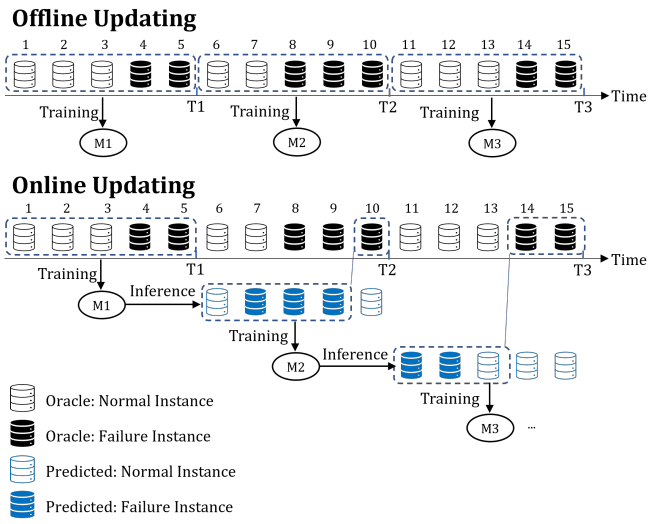

Fig. 1. Toy example of cloud failure prediction model with offline updating and online updating. The prediction model (M) is updated by training with data in the dashed box in each time stage (T). The number on the top of each sub-figure illustrates the instance ID.

the disk icons in black color). We may collect the monitoring metrics and status of the cloud instances in each period (T1, T2, T3, *etc.*). Then we train the model (M1, M2, M3, *etc.*) in each period without any concern. When it comes to the online scenario, however, we lack access to the oracle of instances status after T1, mirroring the conditions of the office scenario. In practice, we train a model in the T1 and make online inferences in the T2 stage, some instances (#7, #8, and #9) are predicted as failure and highlighted in blue. These instances are subsequently mitigated, and cannot access their actual status, which is noted as uncertain positive instances. Finally, together with the known failure instance (#10), we may retrain the model M2. Clearly, the #7 instance is a False Positive one that may introduce noise to model M2. Similarly, we may obtain a less accurate model (#11, #12 are uncertain positive instances, and #14, #15 are true positive instances) in the T3 stage since the model noise may be accumulated.

To the best of our knowledge, we are the first to identify the **U**ncertain **P**ositive **Learning** (UPLearning) challenge in the real-world scenario of cloud failure prediction. The most closely related research topic to UPLearning is the Positive Unlabeled Learning (PULearning) [37]–[45]. PULearning is a machine learning scenario for binary classification where the training set consists of a set of positively labeled instances and an additional unlabeled set that contains positive and negative instances in unknown proportions (so no training instances are explicitly labeled as negative). The risk estimator-based approach, which estimates the distribution of negative instances in the unlabeled set, is widely used to solve the PULearning problem [38], [39], [45]. However, our UPLearning involves uncertain positive instances, which are similar to the unlabeled set in the PULearning scenario, but also includes both positive and negative instances that system operators investigate after failures occur (see Section II). As a result, it is infeasible to directly apply the risk estimator approach used in PULearning.

In this paper, we propose an **U**ncertain **P**osi**t**ive Le**a**rning Ris**k** **E**stimator (UPTAKE), a novel approach for various cloud failure prediction models to achieve high prediction accuracy even with uncertain positive instances. UPTAKE regards uncertain positive instances as both positive and negative through a specially designed risk estimator during the model updating procedure. Besides, UPTAKE only improves the loss function which makes it easy to integrate with various machine learning models, such as RNN, LSTM, Transformer, and TCNN. To evaluate the effectiveness of our proposed UPTAKE approach, we conduct extensive experiments to compare UPTAKE against three model updating approaches on two public disk datasets, Alibaba and Backblaze, which both contain tens of thousands of disks over months. In addition, we apply UPTAKE to the scenario of node failure prediction in Azure, which is a top-tier cloud provider and serves millions of customers around the world. The experiment results on those three datasets demonstrate that UPTAKE outperforms any other model updating approaches over different prediction models (*i.e.*, RNN, LSTM, Transformer and TCNN). The F1-scores of UPTAKE are 45.26%, 70.16%, and 69.33% in scenarios of Alibaba, Backblaze and Azure, which are 4.85%, 4.17%, and 5.13% better than those achieved by the best baseline approaches, respectively.

To sum up, this work has the following contributions:

- We introduce the UPLearning problem in the context of cloud failure prediction and auto-mitigation, marking a pioneering effort to tackle this substantial challenge in the practical deployment of machine learning models in real-world cloud scenarios.
- To solve the UPLearning, we propose an uncertain positive learning risk estimator approach, dubbed UPTAKE, which is easy to integrate with various machine learning models.
- To illustrate the generality and effectiveness of UPTAKE, we conduct experiments on different cloud failure prediction scenarios, *i.e.*, two public disk failure datasets and the node failure from Azure. UPTAKE outperforms any other online model updating approach over four widely used failure prediction models. Moreover, UPTAKE has been successfully applied to Azure and it is proven to improve the reliability of cloud platforms.

II. BACKGROUND

Cloud failure prediction. Cloud failure refers to a state where cloud instances become unavailable due to hardware interruptions, code bugs, or high workload demands. Cloud failure can be categorized based on the affected instances, such as node failure [28], network issue [46], disk failure [26], service overloading [47], *etc.* In the following, we introduce two typical cloud failure prediction scenarios, *i.e.*, disk failure prediction and node failure prediction.

- Node failures may occur for a variety of reasons (*e.g.*, OS crashes, application bugs, misconfigurations, memory leaks,

TABLE I
EXAMPLES OF NODE & DISK MONITORING METRICS.

Type	Description	Examples (Feature belongs to Disk/Node)
Operation Timer	The implementation of a timer for various operations of hard disk drives, *e.g.*, magnetic head seeking and spindle activation, plays a crucial role in optimizing the performance of these devices.	*SpinUpTimeNorm* **(Disk)** *SpinRetryCountRaw* **(Disk)** *Spin_Retry_Count_VALUE* **(Disk)** *Spin_Up_Time_VALUE* **(Disk)**
Operation Counter	Count of common operations, *e.g.*, CPU utilization and power consumption, *etc.*	*PowerCycle* **(Node)** *PowerOnHours* **(Node)** *PowerConsumption* **(Node)**
Physical Characteristics	The measurement of physical characteristics, *e.g.*, temperature and humidity, *etc.*	*AirflowTempRaw* **(Node)** *TemperatureRaw* **(Node)** *Temperature* **(Node)**
Error Counter	A count of specific, identifiable errors is maintained. This count typically remains unchanged unless an error occurs, at which point the count is updated to reflect the occurrence of the error.	*Program_Fail_Cnt_Total_RAW_VALUE* **(Disk)** *Seek_Error_Rate_RAW_VALUE* **(Disk)** *TotalErrors* **(Node)** *IOEDCErrorCountRaw* **(Node)** *MediaErrors* **(Node)** *ErrorInfoLogEntryCount* **(Node)**
EVENT	Windows event are typically used for indicating exception handling for Windows servers.	*WindosStorportEvent_534* **(Node)** *WindosStorportEvent_554* **(Node)**

software incompatibility, overheating, and service exceptions). The node failure can be predicted by a wide range of monitoring metrics (signals indicating their temporal and spatial information). MING [28] predicts node failure by taking into account both spatial and temporal signals. Table I presents several monitoring metrics of a node, which are classified into six distinct categories: operation timer, physical characteristics, operation counter, error counter, logical input/output, and EVENT data. A detailed explanation of these categories, including examples, is illustrated in Table I. The collection of these metrics is conducted at regular intervals, except for EVENT data. To analyze these metrics, a sliding window technique is employed in the prediction model, transforming of the metrics into feature vectors.

- Disk failure in cloud systems is a type of hardware failure that can lead to service downtime. Disk failure can be predicted based on its status data, known as the SMART (Self-Monitoring, Analysis and Reporting Technology) [48], [49]. The monitoring metrics of a disk are SMART (Self-Monitoring, Analysis, and Reporting Technology) [48], [49]. These attributes, which include metrics such as temperature, spin-up time, and reallocated sectors (examples shown in Table I), are employed to predict potential failures. By providing early warning of such failures, SMART technology enables proactive maintenance and replacement to be performed prior to a failure occurring.

Based on insights gained from previous studies, we introduce and implement four widely used machine learning models for cloud failure prediction.

- **RNN** [32]: RNN is a widely-used deep learning model designed for sequential data. During prediction, RNN uses its recurrent unit to feature the difference between the normal state and failure state from the input sequential data.
- **LSTM** [33]: LSTM is an advanced version of recurrent neural network architecture that was designed to model chronological sequences and their long-range dependencies more precisely than conventional RNNs. With long-term features, LSTM can always perform better than RNN and can typically class difficult examples in RNN.
- **Transformer** [26]: The Transformer Model is a novel deep learning-based approach to failure prediction tasks with the attention mechanism, which can capture the temporal nature from instance status data. Transformer utilizes not only a single instance's status data but also considers its neighbors' status data to optimize its prediction performance.
- **TCNN** [34]: TCNN is a variation of Convolutional Neural Networks (CNN) for sequence modeling tasks, by combining aspects of RNN and CNN architectures.

Model updating. The distribution of online monitoring metrics changes with dynamic software and hardware updates in cloud systems, which causes the distribution learned by the previous model to deviate significantly from the online distribution [36]. As a result, the prediction performance of the model may degrade. To ensure prediction performance, machine learning models deployed online need to be updated over time (weekly or monthly). Model updating needs a considerable number of failure instances. Updating cloud failure prediction models using pre-training and fine-tuning [50] strategies is not feasible due to the imbalance between positive and negative instances, as well as the insufficient number of positive instances in real-world scenarios.

III. AN EMPIRICAL STUDY OF PREDICTION ACCURACY OVER TIME

In this study, we aim to uncover the problem based on our experience in deploying failure prediction models in Azure. We address the following research questions:

- RQ1: How does the accuracy of failure prediction change over time?
- RQ2: Why does the prediction accuracy decrease over time?

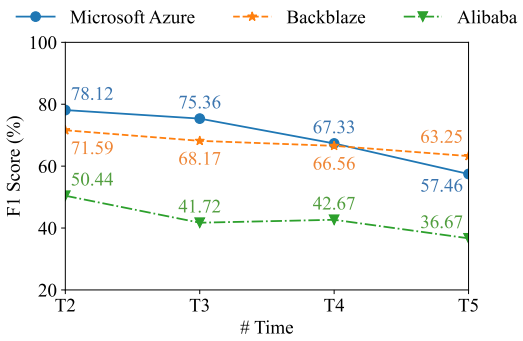

Fig. 2. Cloud failure prediction average F1-score over time.

A. Subjects

We adopt two public datasets [26], [27] for disk-level failure prediction, *i.e.*, Alibaba Cloud and Backblaze datasets. Besides, Azure provides large-scale datasets for node-level failure prediction.

Alibaba is collected from large-scale data centers and published by Alibaba Cloud for PAKDD 2020 Alibaba AIOps Competition [51], which contains millions of disks with a period of more than 16 months. In our experiment, we adopt the dataset within 10 months and split it into five continuous time phases of equal length. Each time phase contains more than 8,000 disk records with a period of two months and each record has 30 days long monitoring metrics. We use 17-dimension features for failure prediction.

Backblaze is a public dataset published by Backblaze, based on the hard drives in Backblaze data center [52]. Each disk of the dataset has a label indicating its status. In our experiment, we use five months from 2021Q4 to 2022Q1, and it contains over 90,000 disks in total. We split the dataset into five continuous time phases of equal length. Each time phase contains approximately 16,000 disk records and each record contains 38 dimension features with a period of 30 days.

Azure is a large-scale cloud system that includes node-level monitoring metrics. Over 500,000 node recordings overall from a period of 35 days are employed in our experiment. We divide all records into five continuous time phases of equal length. Each time phase comprises more than 100,000 node records. The feature input is the 23-dimension monitoring metrics of nodes in a 48-hour time window.

To demonstrate the generality and robustness of our investigation, we employed different periods for model updating: two months for Alibaba, one month for Backblaze, and one week for Azure. These periods were selected based on the number of positive instances available in each, as positive instances are significantly less frequent than negative ones in real-world datasets, as noted by Ntam [26]. We aim to ensure that each period contains a relatively sufficient number of positive instances, maintaining an imbalance rate (#Positive to #Negative ratio) of approximately 1%.

B. RQ1: Prediction Accuracy over Time

We undertake a detailed investigation on prediction performance trends across a continuous long range of time, which is divided into 5 equal-length time phases. At the start of each phase (except for the first *training* phase T1), model updating is performed using monitoring data collected from the previous phase, then deployed for online prediction. The average prediction performance of the model is calculated at each time phase.

The prediction F1-score (see Section V-A4) on 3 datasets across 4-time phases is illustrated in Fig. 2. The time in the figure starts from T2, which is because monitoring data in the T1 phase is used for the initial model training. Overall, there is a significant and consistent decrease in the accuracy of the failure prediction model over time for all datasets. Performance on Azure decreases from 78.22% on T2 to 57.46% on T5, a decrease of 20.66%. Backblaze and Alibaba also show decreases of 8.34% and 13.77%, respectively. The results on Alibaba do not show a decreasing trend in performance on T3 and T4 because the percentage and performance of failures are not completely consistent across time, but the performance of failure prediction on Alibaba also shows a significant decreasing trend of 13.77% when observed over the entire experimental time.

The decrease in prediction accuracy over time can be attributed to changes in the distribution of online data. However, it is important to note that this accuracy decrease is observed consistently across all three cloud systems and time phases. Furthermore, the retrained model is trained using the latest collected data, which has the least bias compared to the online data. We refer to this decrease in prediction accuracy after retraining as the UPLearning problem.

TABLE II
MITIGATION ACTIONS TO CLOUD FAILURES.

Action	Description
Live Migration	Move running VMs from to other failure-free nodes
VM Preserving Soft Reboot	Reboot the host OS kernel and preserve the VM states
Service Healing	Disconnect current VM and generate new assignment of the VM to healthy nodes
Mark Unallocatable	Block allocation of new VMs to a node
Avoid	Reduce the weight to allocate a new VM to a node

C. RQ2: Explanation of Accuracy Decrease

It is mitigation actions that contaminate the training data and lead to a decrease in prediction accuracy (*i.e.*, UPLearning) during model retraining. In the context of failure prediction in cloud systems, when a cloud component is anticipated to fail, mitigation actions [35] in Table II are promptly undertaken to either resolve the underlying failure or minimize the impact on services. The effectiveness of these mitigation actions, however, makes it impractical to verify the accuracy of the predictions. For instance, in the case of applying live migration to a predicted failing node, all run-time state is migrated to a new node, resulting in the release of workload on the original node [53]. This operation alters the original state, rendering it

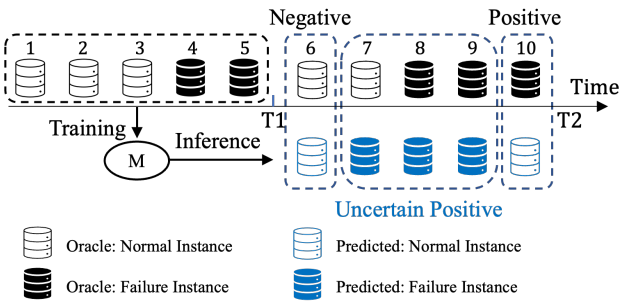

Fig. 3. Example of positive, negative, and uncertain positive in the scenario of online cloud failure prediction.

unverifiable whether the failure would have indeed occurred in the original state.

UPLearning problem. The UPLearning problem in cloud systems is mainly due to mitigation actions, which change the state of the instances and make it impossible to verify the accuracy of the predictions. In the context of cloud failure prediction, an online model (M) is initially trained using historical data with true labels (oracle labels) during period T1. This model predicts instances that are likely to fail, represented by blue-filled instances in period T2, applying mitigation actions to address the potential failure. However, it remains unknown whether these predicted instances will actually fail or not, as indicated by the black-bordered instances (#6-#10). *Uncertain positive* instances are those predicted as failures by the online model, such as instances #7, #8, and #9 in Fig. 3. However, the true labels (oracle labels) for these instances (#7: Negative, #8 and #9: Positive) cannot be verified after applying failure mitigation actions, because mitigation actions change the original state of the instances.

IV. Approach

In this section, we introduce a model updating approach to solve the UPLearning problem, named UPTAKE, which still consumes uncertain positive instances during model updating and significantly improves the online model updating performance for cloud failure prediction.

A. Problem Settings

Before diving into the approach details, we present a formal definition of the cloud failure prediction task. We define the cloud failure prediction task as a binary classification problem based on monitoring metrics. Essentially, we collect feature vectors with d dimensions, which contain different types of monitoring metrics from cloud infrastructures, at regular intervals (e.g. hourly or daily). These feature vectors are used to create a continuous time series of features, denoted as $X = [x_1, x_2, ..., x_l] \in \mathbb{R}^d$, where l is the number of timestamps in the feature X. For each feature X, we assign a label $Y \in 0, 1$ based on whether it represents a failure instance $(Y = 1)$ or a normal instance $(Y = 0)$ in the cloud system. The goal of cloud failure prediction is to train a classifier $f_X \rightarrow Y$ that can predict the class label Y when given a feature X with uncertain positive labels.

The training set is a collection of features X with their corresponding label Y, denoted as $\mathcal{N} = \{(X_1, Y_1), ..., (X_n, Y_n)\}$, where $n = |\mathcal{N}|$. Our target is to minimize prediction loss $\mathcal{L}$, (*e.g.*, the binary cross-entropy loss), on the training set.

In the scenario of cloud failure prediction, some labels $Y_i (i \in \{1, ..., n\})$ in the training set are uncertain, specifically, we have to use the training set which contains uncertain label $\hat{Y}$ to train and update the online model. The training loss to be minimized is represented as:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} l(f(X_i), \hat{Y}_i) \tag{1}$$

where l is the loss function, f is the binary classifier, and $(X_i, \hat{Y}_i), \quad i \in \{1, ...n\}$ is the sample from the training set. $\hat{Y}$ is the uncertain label (*i.e.*, the predicted label collected from the online model).

B. UPTAKE

Typically, the model's performance is assessed using the risk function [54]. A precise risk function enhances the performance of the failure prediction model. The UPTAKE approach ensures high prediction accuracy, especially when handling uncertain positive instances, and effectively addresses the UPLearning problem.

Risk estimator. In this section, we describe the binary classification representation of the risk function [54], [55]. Consider a feature set $X \in \mathbb{R}^d$ and its corresponding label $Y \in \{0, 1\}$. Let $p(x, y)$ be the underlying joint density of (X, Y), $p_{\mathrm{p}}(x) = p(x|Y = 1)$, and $p_{\mathrm{n}}(x) = p(x|Y = 0)$ represent the positive and negative marginals, respectively. Additionally, define $\pi_{\mathrm{p}} = p(Y = 1)$ as the class-prior probability and $\pi_{\mathrm{n}} = p(Y = 0) = 1 - \pi_{\mathrm{p}}$.

Let $f_X \rightarrow Y$ be the decision function, and $l : \mathbb{R} \times \{0, 1\} \rightarrow \mathbb{R}$ be the loss function, where $l(f(X), Y)$ measures the loss between the predicted output $f(X)$ and the actual label Y. Define $R(f)$ as the risk function for the given decision function $f_X \rightarrow Y$, $R^+_{\mathrm{p}}(f) = \mathbb{E}_{\mathrm{p}}[l(f(X), 1)]$, and $R^-_{\mathrm{n}}(f) = \mathbb{E}_{\mathrm{n}}[l(f(X), 0)]$, where $\mathbb{E}_{\mathrm{p}}[\cdot] = \mathbb{E}_{X\sim p_{\mathrm{p}}}[\cdot]$ and $\mathbb{E}_{\mathrm{n}}[\cdot] = \mathbb{E}_{X\sim p_{\mathrm{n}}}[\cdot]$, denoting the expectations of positive and negative variables. In standard binary classification, the risk of the decision function $f_X \rightarrow Y$ is denoted as:

$$R(f) = \mathbb{E}_{(X,Y)\sim p(x,y)}[l(f(X), Y)] = \pi_{\mathrm{p}} R^+_{\mathrm{p}}(f) + \pi_{\mathrm{n}} R^-_{\mathrm{n}}(f) \tag{2}$$

UPTAKE. In the context of cloud failure prediction, we encounter instances not only labeled as positive or negative but also instances with uncertain positive labels predicted by the online model. UPTAKE treats uncertain positive instances as having both positive and negative labels with different weights. The risk function for this scenario is represented as follows:

$$R(f) = \pi_{\mathrm{p}} R^+_{\mathrm{p}}(f) + \pi_{\mathrm{n}} R^-_{\mathrm{n}}(f) + (\pi_{\mathrm{p}} R^+_{\mathrm{u}}(f) + \pi_{\mathrm{n}} R^-_{\mathrm{u}}(f)) \tag{3}$$

where $R_{\mathrm{u}}^{+}(f) = \mathbb{E}_{\mathrm{u}}[l(f(X),1)]$ and $R_{\mathrm{u}}^{-}(f) = \mathbb{E}_{\mathrm{u}}[l(f(X),0)]$ represent the expectations of the decision function $f(X)$ when dealing with uncertain positive instances.

It is crucial to note that π_{p} and π_{n} are hyper-parameters in this study. π_{p} estimates the proportion of real positive instances among uncertain positive instances, while $\pi_{\mathrm{n}} = 1 - \pi_{\mathrm{p}}$ estimates the percentage of negative instances. In this paper, π_{p} is set to the Precision value obtained during the T1 validation stage of hyper-parameter tuning. When π_{p} approximates the proportion of uncertain positive instances with true positive labels, UPTAKE demonstrates improved performance. In Section V-F, we discuss the parameter sensitivity of π_p.

Implementation in various models. UPTAKE is a generic approach that can be integrated with any machine learning model for cloud failure prediction.

Algorithm 1 illustrates the core implementation of UPTAKE. The input training data subscripted with p, n, and u are the instances labeled with positive, negative, and uncertain positive, respectively, and θ indicates the model parameter for decision function $f(X;\theta)$. To reduce the interference of uncertain positive instances to the direction of gradient descent, UPTAKE improves the loss function in the model updating procedure by adjusting the direction of the gradient.

Since UPTAKE only improves the original loss function without changing its property, it is suitable for a wide range of machine learning models across various applications.

Algorithm 1 Introducing UPTAKE in the gradient descent

Input: training data $(X_{\mathrm{p}}, X_{\mathrm{n}}, X_{\mathrm{u}})$;
hyperparameter $0 \leq \pi_{\mathrm{p}} \leq 1$, and $\pi_{\mathrm{n}} = 1 - \pi_{\mathrm{p}}$;
decision function $f(X;\theta)$; loss function $l(f(X), Y)$
Output: gradient ∇_θ to update θ for $f(X;\theta)$
1: *Improved loss*
$\mathcal{L} = \pi_{\mathrm{p}} l(f([X_{\mathrm{p}}, X_{\mathrm{u}}]), 1) + \pi_{\mathrm{n}} l(f([X_{\mathrm{n}}, X_{\mathrm{u}}]), 0)$
2: Set gradient $\nabla_\theta \mathcal{L}$ and update θ

V. EXPERIMENT

In this section, we conduct extensive experiments to demonstrate the effectiveness of our UPTAKE approach when integrated with existing failure prediction models. First, we present the experimental settings, including datasets, the failure prediction models, and three comparative approaches. Following the setup, we conduct experiments to verify the following research questions:

- **RQ3:** How does UPTAKE perform for UPLearning problem in cloud failure prediction?
- **RQ4:** How does UPTAKE perform in the scenario of online cloud systems?
- **RQ5:** Does UPTAKE exhibit consistent performance across different base models?
- **RQ6:** What is the impact of UPTAKE on predicting efficiency?
- **RQ7:** How do UPTAKE's parameters impact its prediction performance?

A. Settings

1) Datasets: To evaluate the reliability and applicability of UPTAKE, we employ two publicly accessible datasets: the Alibaba Cloud and Backblaze datasets [26], [27], for predicting disk failures. Furthermore, we integrate a vast industrial cloud system dataset to predict node failures. These datasets together form the foundation of our assessment and analysis. The specific configuration and details of these datasets align with what we outlined in Section III-A.

2) Implementations and Environments: Four deep learning models, *i.e.*, RNN, LSTM, Transformer, and TCNN, are implemented based on Python 3.8.13 and PyTorch 1.11.0 [56]. Identical parameter values are used as in the previous works for the compared approaches. The model was selected based on the epoch that achieved the highest F1-score on the validation set. All the experiments are conducted on Linux Dev Node with Ubuntu 20.04 LTS 64-bit operating system, 24 cores AMD Epyc 7V13, 220 GB memory, and NVIDIA Tesla A100 with 80 GB GPU memory.

3) Important Hyper-parameters: As explained in Section IV-B, the parameter π_{p} plays a crucial role in this study. Its value is derived from the Precision metric obtained during the T1 validation stage.

4) Evaluation metrics: The cloud failure prediction task is a binary classification task, aligning with existing research. The performance of UPTAKE is assessed using Precision, Recall, and F1-score, following standard conventions. Instances are labeled as either normal or failure based on their practical performance, with failure considered positive and normal considered negative. True Positive (TP) represents the number of correctly predicted failure instances, while True Negative (TN) represents the accurately predicted normal instances. Conversely, False Positive (FP) and False Negative (FN) indicate instances that were incorrectly predicted as normal and failure, respectively.

5) Compared Approaches: In the experiment section, we compare UPTAKE with two model updating approaches, which are offline updating (offline), and online updating with certain positive labels (certain). To the best of our knowledge, UP-Learning has not been proposed before. Therefore, there are no state-of-the-art solutions for model updating. We design the two model updating approaches as compared with UPTAKE.

- ***Offline* updating** employs ground truth labels for model training, which contain no uncertain labels. However, this approach is not feasible in practice because part of the ground truth labels are uncertain positive labels. It serves as an *upper bound* of the performance for UPTAKE and other subsequent approaches.
- **Online updating with *certain* positive labels** accumulates all records with certain labels for model updating. For instance, when updating the model for predicting phase T4, the *certain* approach collects all records with certain labels from phase T1 to phase T3. This includes all records in phase T1, as well as true negative (TN) and false negative (FN) records in phases T2 and T3.

TABLE III
PERFORMANCE COMPARISON OF APPROACHES ON ALIBABA DATASET. P, R, AND F1 DENOTE PRECISION, RECALL AND F1-SCORE, RESPECTIVELY.

Model	Approach	T2			T3			T4			T5			Avg.
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	F1
RNN	Offline	57.84	33.33	42.29	40.12	50.00	44.52	76.92	35.71	48.78	43.89	49.07	46.33	46.54
	Certain				35.95	42.31	38.87	79.59	23.21	35.94	49.12	17.39	25.69	33.50
	UPTAKE				40.30	41.54	**40.91**	47.65	42.26	**44.79**	35.96	39.75	**37.76**	**41.15**
LSTM	Offline	65.31	36.16	46.55	65.82	40.00	49.76	74.71	38.69	50.98	63.11	40.37	49.24	49.99
	Certain				54.22	34.62	42.25	80.43	22.02	34.58	59.09	24.22	34.36	37.06
	UPTAKE				56.18	38.46	45.66	64.76	40.48	**49.82**	58.72	39.75	**47.41**	**47.63**
Trans -former	Offline	62.41	46.89	53.55	61.80	42.31	50.23	68.47	45.24	54.48	61.42	48.45	54.17	52.96
	Certain				53.92	42.31	47.41	79.66	27.98	41.41	61.76	26.09	36.68	41.83
	UPTAKE				53.04	46.92	**49.80**	46.70	50.60	**48.57**	41.24	49.69	**45.07**	**47.81**
TCNN	Offline	57.72	40.11	47.33	43.59	52.31	47.55	60.38	38.10	46.72	58.88	39.13	47.01	47.09
	Certain				46.25	28.46	35.24	72.41	25.00	37.17	42.86	22.36	29.39	33.93
	UPTAKE				63.24	33.08	**43.43**	36.84	54.16	**43.85**	68.29	34.78	**46.09**	**44.46**
Avg.	Offline	60.82	39.12	47.43	52.83	46.16	48.02	70.12	39.44	50.24	56.83	44.26	49.19	49.15
	Certain				47.59	36.93	40.94	78.02	24.55	37.28	53.21	22.52	31.53	36.58
	UPTAKE				53.19	40.00	**44.95**	48.99	46.88	**46.76**	51.05	40.99	**44.08**	**45.26**

TABLE IV
PERFORMANCE COMPARISON OF APPROACHES ON BACKBLAZE DATASET. P, R, AND F1 DENOTE PRECISION, RECALL AND F1-SCORE, RESPECTIVELY.

Model	Approach	T2			T3			T4			T5			Avg.
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	F1
RNN	Offline	50.88	84.67	63.56	62.75	73.56	67.73	74.29	73.24	73.76	76.33	62.32	68.62	70.04
	Certain				57.89	50.57	53.99	80.91	41.78	55.11	83.67	39.61	53.77	54.29
	UPTAKE				59.31	69.54	**64.02**	62.21	87.32	**72.66**	57.73	81.16	**67.47**	**68.05**
LSTM	Offline	63.59	85.40	72.90	70.79	82.18	76.06	76.71	78.87	77.78	70.71	81.64	75.78	76.54
	Certain				71.81	61.49	66.25	86.79	43.19	57.68	82.86	42.03	55.77	59.90
	UPTAKE				63.32	83.33	**71.96**	61.72	74.18	**67.38**	54.79	80.19	**65.10**	**68.15**
Trans -former	Offline	71.60	84.67	77.59	74.87	85.63	79.89	76.23	79.81	77.98	72.44	78.74	75.46	77.78
	Certain				69.54	69.54	69.54	83.48	45.07	58.54	82.41	43.00	56.51	61.53
	UPTAKE				71.36	87.36	**78.55**	71.31	81.69	**76.15**	64.77	82.61	**72.61**	**75.77**
TCNN	Offline	63.54	83.94	72.33	67.35	75.86	71.35	74.24	79.81	76.92	79.63	62.32	69.92	72.73
	Certain				77.45	45.40	57.25	79.07	47.89	59.65	80.92	51.21	62.72	59.87
	UPTAKE				59.26	73.56	**65.64**	62.45	81.22	**70.61**	63.49	77.29	**69.72**	**68.66**
Avg.	Offline	62.40	84.67	71.60	68.94	79.31	73.76	75.37	77.93	76.61	74.78	71.26	72.45	74.27
	Certain				69.17	56.75	61.76	82.56	44.48	57.75	82.47	43.96	57.19	58.90
	UPTAKE				63.31	78.45	**70.04**	64.42	81.10	**71.70**	60.20	80.31	**68.73**	**70.16**

B. RQ3: Performance of UPTAKE

We assess our proposed UPTAKE, using two public datasets referred to as Alibaba and Backblaze. The results are presented in Table III and Table IV respectively. In these tables, the last column displays the average F1-score for each row. The final three rows depict the average results of all four models across different time phases and using different approaches.

In the rest of the table, every set of three columns showcases the performance metrics (Precision, Recall, and F1-score) for different time phases. Notably, the time phases in Table III and Table IV commence from T2, as the data in T1 is utilized for the initial model training. It is observed that the three approaches yield identical results at T2 since it marks the first stage to obtain model inference results without encountering any UPLearning problem (as illustrated in Fig. 1).

The performance of these models remains competitive with previous work [27], despite the inherent challenges of cloud failure prediction tasks in real-world scenarios. Notably, online disk failure prediction often faces limitations due to low F1-scores. This challenge largely stems from the prevalence of missing data and significant imbalances in the data distribution, posing significant obstacles to improvement.

Upon analyzing Table III and Table IV, we observe that UPTAKE performs admirably on these two public datasets. The average F1-scores are 45.26% for Alibaba and 70.16% for Backblaze. According to the results shared above, we have the following findings. The results of various methods applied to Alibaba's data demonstrate low performance due to the complexity of the dataset. The highest F1-score achieved on

TABLE V
PERFORMANCE COMPARISON OF APPROACHES ON AZURE. P, R, AND F1 DENOTE PRECISION, RECALL AND F1-SCORE, RESPECTIVELY.

Model	Approach	T2			T3			T4			T5			Avg.
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	F1
RNN	Offline	88.31	86.96	75.16	88.59	67.11	76.36	71.43	59.14	64.71	82.24	68.75	74.89	71.99
	Certain				93.94	46.62	62.31	83.16	43.82	57.39	93.50	48.70	64.04	61.25
	UPTAKE				94.62	49.62	**65.10**	84.65	51.88	**64.33**	93.28	57.81	**71.38**	**66.94**
LSTM	Offline	80.60	79.67	80.13	66.19	78.76	71.93	77.44	68.28	72.57	80.79	63.54	71.14	71.88
	Certain				47.45	61.28	53.49	78.95	56.45	65.83	68.65	93.69	68.65	62.66
	UPTAKE				59.58	79.51	**68.12**	65.05	75.54	**69.90**	84.00	60.16	**70.11**	**69.38**
Trans -former	Offline	65.60	96.55	78.12	75.27	78.95	77.06	76.45	67.20	71.53	71.03	73.44	72.22	73.60
	Certain				80.54	56.02	66.08	82.63	52.42	64.14	91.28	51.82	66.11	65.44
	UPTAKE				72.38	77.82	**75.54**	65.37	72.04	**68.54**	71.74	68.75	**70.21**	**71.43**
TCNN	Offline	80.87	75.70	78.20	67.89	79.89	73.40	84.69	66.94	74.77	64.25	76.30	69.76	72.64
	Certain				45.24	57.14	50.50	86.51	50.00	63.37	91.08	50.52	64.99	59.62
	UPTAKE				63.72	77.26	**69.84**	71.28	73.39	**72.32**	57.77	78.39	**66.52**	**69.56**
Avg.	Offline	78.85	84.72	77.90	74.49	76.18	74.69	77.50	65.39	70.90	74.58	70.51	72.00	72.53
	Certain				66.79	55.27	58.10	82.81	50.67	62.68	86.13	61.18	65.95	62.24
	UPTAKE				72.58	71.05	**69.65**	71.59	68.21	**68.77**	76.70	66.28	**69.56**	**69.33**

the leader-board is only 49.07%. [1]

Comparison with different updating approaches. When examining the last column of Table III and Table IV, it becomes evident that *offline* updating consistently outperforms other approaches. It achieves an average F1-score of 49.15% in Alibaba and 74.27% in Backblaze. However, it is important to note that offline updating, while showing the best performance, is not practically applicable as it requires the access to all certain labels, which is often unavailable in real-world scenarios. Therefore, we present this approach as the "upper bound" of the prediction model, representing the best possible performance using all certain labels. Among the practical and feasible approaches, UPTAKE demonstrates the best performance. It achieves an average F1-score of 45.26% in Alibaba and 70.16% in Backblaze. In contrast, *Certain* performs the worst, with an average F1-score of 36.58% in Alibaba and 58.90% in Backblaze. We attribute the lower accuracy of *Certain* to the accumulation of phases, which causes the proportion of positive and negative instances to deviate from the actual distribution. Specifically, the *Certain* method disregards all uncertain positive samples, even though only an average of 60.84% in Alibaba and 19.27% in Backblaze of positive labels are certain positive. This discrepancy in handling uncertain positive samples results in a distribution drift, leading to less accurate performance.

Model comparison. Analyzing the UPTAKE results for each model displayed in Table III and Table IV, we observe that UPTAKE consistently demonstrates different performance improvements compared to the *Certain* approach across various models. In general, each model in our experiment experiences enhancements when using UPTAKE. Specifically, RNN exhibits the most substantial improvement, achieving an average F1-score increase of 7.02% across all datasets, while LSTM shows the smallest improvement, with an average F1-score increase of 1.64%. However, when compared with offline updating, we do not observe a consistent pattern in model performance across all datasets. This discrepancy can be attributed to biases introduced in the data distribution among different stages, which affect the performance of various models in distinct ways. In summary, UPTAKE consistently outperforms *Certain* model updating approaches and approaches the performance of the *Offline* (*i.e.*, the upper bound) across different prediction models. This consistency underscores its generality and robustness in enhancing model performance.

C. RQ4: Online Performance

We have deployed UPTAKE to one of the top-tier cloud systems in the world, Azure, which suffer from the UPLearning problem before using our approach. We have conducted an online experiment (in the A/B testing environment to obtain ground truth) for a period of time over five weeks from July 2022 to August 2022.

Table V presents a comparison of model performance for the model updating approaches mentioned before, following the same format as the tables in Section V-B. UPTAKE outperforms *Certain*, achieving an average F1-score of 69.33%. This performance surpasses Certain by an average of 7.09% and falls slightly below the *Offline* performance by 3.2% in terms of F1-score.

The accumulation of uncertain positive labels impacts the performance of node failure prediction. Unlike failure prediction on public disk datasets, the node failure prediction model is more concerned with the quality of labels rather than the proportion of positive samples to negative samples. In this context, the accuracy and reliability of labels are important in determining the effectiveness of the prediction model.

In summary, UPTAKE excels in handling the UPLearning problem compared to previous online updating approaches. Additionally, it proves effective in mitigating performance

[1]https://tianchi.aliyun.com/competition/entrance/231775/rankingList

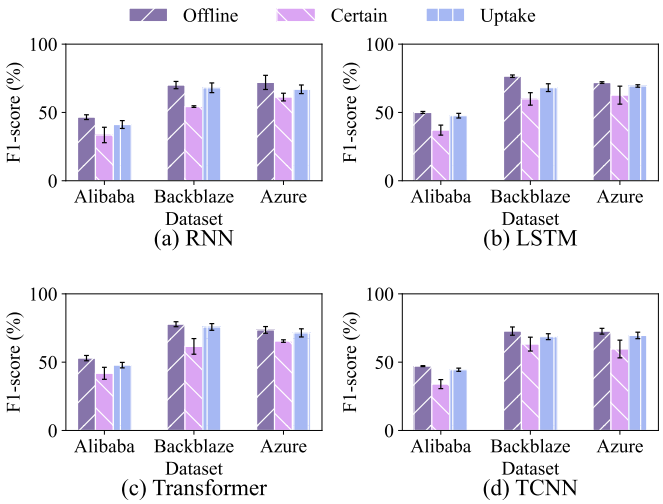

Fig. 4. The average F1-score comparison of UPTAKE and its two compared approaches on three datasets: Alibaba, Backblaze, and Azure. Each sub-figure corresponds to a specific type of base model. The error bar in the figure represents the upper and lower bounds of the results, indicating the robustness of UPTAKE across different base models.

TABLE VI
EFFICIENCY COMPARISON OF UPTAKE AND ITS TWO COMPARED APPROACHES ON ALIBABA, BACKBLAZE AND AZURE DATASET. EACH VALUE INDICATES THE AVERAGE TIME (S) TAKEN OF THE TRAINING EPOCH IN THE MODEL UPDATING PROCESS.

	Offline	Certain	UPTAKE
Alibaba	5.38	12.58	**4.31**
Backblaze	4.38	12.75	**4.12**
Azure	11.54	27.90	**10.95**

degradation over time, showcasing its robustness and reliability in real-world applications.

D. RQ5: Robustness

To further empirically evaluate the performance of UPTAKE, we counted the results of UPTAKE for different base models (*i.e.*, the model in Section V-A2) on different cloud systems (*i.e.*, Alibaba, Backblaze and Azure).

The provided Fig. 4 displays the average F1-score across all time phases for different model update strategies on diverse base models and datasets. It is essential to clarify that "*Offline*" denotes the theoretical optimal performance of the current system at a given time, assuming ideal conditions. However, achieving this theoretical best performance in practical applications is hindered by the UPLearning problem.

Fig. 4 clearly demonstrates UPTAKE's superior performance over other model update strategies across all scenarios and models. Notably, it closely approaches the theoretical best performance (*Offline*). UPTAKE exhibits greater stability over time, as indicated by the smaller fluctuations in performance metrics, highlighting its robustness and reliability compared to other model updating strategies.

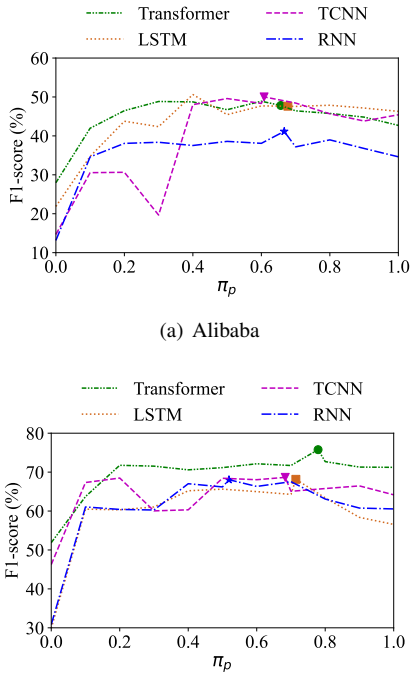

Fig. 5. F1-score of UPTAKE under different π_{p} on two public datasets. The dots are the parameters chosen by our solution.

E. RQ6: Efficiency

Besides prediction performance, efficiency is also a critical metric for online model updating. Consequently, we compare the running time of UPTAKE with other approaches proposed in Section V-A5. Since the difference between different approaches only exists in the training step, we compared the average time cost of an epoch with the same hyper-parameters on each time phase.

From the results in Table VI, these approaches perform almost similarly in efficiency except *Certain*, and UPTAKE take the least time to train during an epoch with 4.31s in Alibaba, 4.12s in Backblaze, and 10.95s in Azure. The *Certain* approach is slowest because *Certain* accumulates instances with certain labels in all former phases, which leads to a larger training set. The larger the dataset it uses, the longer time it takes to train.

F. RQ7: Parameter Sensitivity

We investigate the impact of the only parameter π_{p} used in UPTAKE. It reflects the ratio of positive instances in all uncertain positive instances. Our algorithm suggests using the Precision value on the T1 stage to set π_{p}, which is an estimation using historical information.

In this experiment, we grid search the value of π_{p} from 0 to 1 on a 0.1 base step. Fig. 5 shows the effectiveness of different π_{p} in terms of F1-score. In this figure, each line is the average F1-score of the prediction model (RNN, LSTM, Transformer, and TCNN) on the Alibaba and Backblaze datasets, respectively. The dots represent the π_{p} chosen by our solution.

Despite minor perturbations under different π_{p}, the overall performance is stable, and our selected π_{p} consistently yield near-optimal performance, which indicates UPTAKE is robust in practice without the need to tune parameters extensively.

VI. Discussions

A. Threats to Validity

Internal threats. Our implementation choices are a potential internal threat. To address this, we utilize established implementations for our deep learning models, detailed in Section V-A2. Moreover, our implementation is openly accessible, ensuring transparency and enabling future research replication.

External threats. External threats stem from model and data selection, as well as comparisons with other methods. We mitigate selection bias by incorporating diverse datasets from three distinct cloud systems, covering both node and disk-level data. These datasets align with industry standards, enhancing the representativeness of our results. Additionally, this study pioneers investigations into the UPLearning problem, selecting standard model updating strategies and a theoretical optimal strategy, as explained in Section V-A5. We remain open to exploring new cloud component data and model updating techniques in future research to enhance our study's depth.

Construct threats. Construct threats concern our chosen metrics and parameters. We utilize widely-accepted metrics such as precision, recall, and F1-score for effectiveness evaluation, and training time for efficiency measurement. Future evaluations will incorporate additional metrics for a more comprehensive assessment. Parameters in UPTAKE are defined based on established rules (Section V-A3), with detailed discussions about the primary parameter, π_{p}, in Section V-F, ensuring a well-informed evaluation framework.

B. Deployment

Our UPTAKE framework is deployed on Azure, a platform with millions of nodes serving a wide customer base. The process consists of three core phases: data preparation, model retraining, and model deployment.

Data preparation. During this phase, collected data is cleaned and engineered to ensure completeness and quality. These steps are essential for enhancing the reliability and effectiveness of subsequent model retraining.

Model retraining. In this stage, the failure prediction model undergoes retraining. To address challenges related to the UPLearning problem, UPTAKE is integrated into the retraining process, effectively mitigating associated issues.

Model deployment. The retrained model is deployed on *AzureML*, a platform designed for seamless management and deployment of online models. Once deployed, the model actively performs online cloud failure prediction, providing insights into potential failure events within the cloud system.

To gauge UPTAKE's effectiveness in terms of business impact, we conduct A/B testing, measuring reduced mitigation action times and improved service availability. Compared to the online retraining strategy discussed in Section III, UPTAKE significantly reduced required mitigation actions and enhanced service availability. These results demonstrate UPTAKE's substantial benefits for online failure prediction models, highlighting its effectiveness and positive business impact.

C. Future Directions for Model Evaluation

Evaluation over long running periods. The current scope of our research primarily focuses on demonstrating the efficacy of our predictive model through short-term, controlled experiments designed to mirror real-world scenarios. This approach was chosen to provide a clear, immediate comparison between traditional models and our proposed method. We acknowledge the importance of evaluation on long time periods and consider it an essential direction for future research, intending to extend our analysis to cover more periods and assess the sustainability and adaptability of our model over time.

Changing proportions of failures over time. Our research acknowledges that failure rates and their patterns are not static, but vary across different time windows, reflecting the dynamic nature of cloud environments. The initial focus on specific real-world ternaries was to establish a baseline for immediate impact and measurement. Moving forward, we plan to systematically investigate the model's response to varying failure proportions over extended periods, aiming to develop more adaptive predictive strategies that can accommodate the evolving conditions of cloud infrastructures.

VII. Related work

Failure prediction. In recent years, various approaches for predicting cloud failure have appeared, including hard drive disk failure and node failure prediction [25], [29], [30], [57]. As a binary classification problem, machine learning and deep learning mechanisms are widely used for failure prediction. Machine learning approaches for cloud failure prediction, such as support vector machine [33] and tree models [58]–[61], use several monitoring metrics collected in a time window to predict whether the cloud component will fail soon. However, these machine learning approaches struggle to handle the complex temporal information of cloud systems [34]. Deep learning approaches such as RNN [32], LSTM [28] and TCNN [34] can better capture the temporal correlation of the complex monitoring metrics than classical machine learning approaches. In recent years, Transformers have outperformed conventional deep-learning approaches. The state-of-the-art performance is achieved by NTAM [26], which incorporates both temporal and spatial information into failure prediction. Our research is orthogonal to previous failure prediction approaches since we aim to solve the model updating issue and boost the overall performance of failure prediction.

Uncertain Labels. UPLearning solves uncertain/unlabeled problems for classification tasks [37]–[43]. It defines positive samples and unlabeled ones for the *training phase*. For example, an unbiased risk estimator [44] is proposed to solve this problem. Different from PULearning, the UPLearning is

identified in the *model updating phase*, where the samples for retraining have three classes: positive, negative and uncertain positive samples. This approach allows the model to refine its predictions by incorporating the uncertainty of samples, thereby enhancing its overall robustness and accuracy.

VIII. Conclusion

Cloud failure prediction and auto-mitigation are crucial to maintain the reliability of cloud systems. Through our practical experience from deploying cloud failure prediction models on Azure, we identify a significant issue known as UPLearning, which arises during the model updating process. This issue critically impairs the prediction performance in real-world scenarios. To address this, we introduce a novel model updating approach, referred to as UPTAKE, which enhances the performance of various predictive models including RNN, LSTM, Transformer, and TCNN. Our experiments on both public and private cloud datasets have shown that UPTAKE substantially outperforms baseline methods, achieving significantly improved performance. Furthermore, by integrating UPTAKE into our cloud platforms, we have proven tangible benefits in practical applications and demonstrated the robustness of UPTAKE in enhancing the failure prediction and auto-mitigation in the real-world cloud environments.

Acknowledgment

We thank Microsoft engineering teams engaged with us, contributing their expertise and efforts to this research. We also thank the reviewers, whose insightful feedback improve the quality of this work.

References

[1] C. Colman-Meixner, C. Develder, M. Tornatore, and B. Mukherjee, "A survey on resiliency techniques in cloud computing infrastructures and applications," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2244–2281, 2016.

[2] B. Alouffi, M. Hasnain, A. Alharbi *et al.*, "A systematic literature review on cloud computing security: threats and mitigation strategies," *IEEE Access*, vol. 9, pp. 57 792–57 807, 2021.

[3] Z. Yu, M. Ma, C. Zhang, S. Qin, Y. Kang, C. Bansal, S. Rajmohan, Y. Dang, C. Pei, D. Pei *et al.*, "Monitorassistant: Simplifying cloud service monitoring via large language models," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, 2024, pp. 38–49.

[4] V. Ganatra, A. Parayil, S. Ghosh, Y. Kang, M. Ma, C. Bansal, S. Nath, and J. Mace, "Detection is better than cure: A cloud incidents perspective," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 1891–1902.

[5] P. Jin, S. Zhang, M. Ma, H. Li, Y. Kang, L. Li, Y. Liu, B. Qiao, C. Zhang, P. Zhao *et al.*, "Assess and summarize: Improve outage understanding with large language models," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 1657–1668.

[6] Q. Lin, T. Li, P. Zhao, Y. Liu, M. Ma, L. Zheng, M. Chintalapati, B. Liu, P. Wang, H. Zhang *et al.*, "Edits: An easy-to-difficult training strategy for cloud failure prediction," in *Companion Proceedings of the ACM Web Conference 2023*, 2023, pp. 371–375.

[7] A. Stocco, P. J. Nunes, M. d'Amorim, and P. Tonella, "Thirdeye: Attention maps for safe autonomous driving systems," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–12.

[8] H. Lu, H. Wang, Q. Wu, H. Luo, Q. Zhao, B. Liu, Q. Si, S. Zheng, W. Guo, and N. Ren, "Automatic control and optimal operation for greenhouse gas mitigation in sustainable wastewater treatment plants: A review," *Science of the Total Environment*, vol. 855, p. 158849, 2023.

[9] Z. Xie, S. Zhang, Y. Geng, Y. Zhang, M. Ma, X. Nie, Z. Yao, L. Xu, Y. Sun, W. Li *et al.*, "Microservice root cause analysis with limited observability through intervention recognition in the latent space," in *Proceedings of the 30th ACM Conference on Knowledge Discovery and Data Mining*, 2024.

[10] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen *et al.*, "Automatic root cause analysis via large language models for cloud incidents," in *Proceedings of the Nineteenth European Conference on Computer Systems*, 2024, pp. 674–688.

[11] Z. Zeng, Y. Zhang, Y. Xu, M. Ma, B. Qiao, W. Zou, Q. Chen, M. Zhang, X. Zhang *et al.*, "Traceark: Towards actionable performance anomaly alerting for online service systems," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2023, pp. 258–269.

[12] H. Li, M. Ma, Y. Liu, S. Qin, B. Qiao, R. Yao, H. Chaturvedi, T. Tran, M. Chintalapati, S. Rajmohan *et al.*, "Codec: Cost-effective duration prediction system for deadline scheduling in the cloud," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 298–308.

[13] R. Ding, C. Zhang, L. Wang, Y. Xu, M. Ma, X. Wu, M. Zhang, Q. Chen, X. Gao, X. Gao *et al.*, "Tracediag: Adaptive, interpretable, and efficient root cause analysis on large-scale microservice systems," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 1762–1773.

[14] M. Ma, Z. Yin, S. Zhang, S. Wang, C. Zheng, X. Jiang, H. Hu, C. Luo, Y. Li, N. Qiu *et al.*, "Diagnosing root causes of intermittent slow queries in cloud databases," *Proceedings of the VLDB Endowment*, vol. 13, no. 8, pp. 1176–1189, 2020.

[15] Z. Yu, C. Pei, X. Wang, M. Ma, C. Bansal, S. Rajmohan, Q. Lin, D. Zhang, X. Wen, J. Li *et al.*, "Pre-trained kpi anomaly detection model through disentangled transformer," in *Proceedings of the 30th ACM Conference on Knowledge Discovery and Data Mining*, 2024.

[16] N. Zhao, J. Zhu, Y. Wang, M. Ma, W. Zhang, D. Liu, M. Zhang, and D. Pei, "Automatic and generic periodicity adaptation for kpi anomaly detection," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1170–1183, 2019.

[17] Z. Wang, C. Pei, M. Ma, X. Wang, Z. Li, D. Pei, S. Rajmohan, D. Zhang, Q. Lin, H. Zhang *et al.*, "Revisiting vae for unsupervised time series anomaly detection: A frequency perspective," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 3096–3105.

[18] Y. Chen, C. Zhang, M. Ma, Y. Liu, R. Ding, B. Li, S. He *et al.*, "Imdiffusion: Imputed diffusion models for multivariate time series anomaly detection," *VLDB Endowment*, vol. 17, no. 3, p. 359–372, 2023.

[19] M. Ma, S. Zhang, J. Chen, J. Xu, H. Li, Y. Lin, X. Nie, B. Zhou, Y. Wang, and D. Pei, "Jump-starting multivariate time series anomaly detection for online service systems," in *Proceedings of the Annul Technical Conference (ATC)*. USENIX, 2021, pp. 413–426.

[20] M. Babaioff, R. Lempel, B. Lucier, I. Menache, A. Slivkins, and S. C.-w. Wong, "Truthful online scheduling of cloud workloads under uncertainty," in *Proceedings of the international conference on World Wide Web*, 2022, pp. 151–161.

[21] L. Zeng, P. Huang, K. Luo, X. Zhang *et al.*, "Fograph: Enabling real-time deep graph inference with fog computing," in *Proceedings of the international conference on World Wide Web*, 2022, pp. 1774–1784.

[22] A. Alhilal, T. Braud, B. Han, and P. Hui, "Nebula: Reliable low-latency video transmission for mobile cloud gaming," in *Proceedings of the international conference on World Wide Web*, 2022, pp. 3407–3417.

[23] Y. Jiang, C. Zhang, S. He, Z. Yang, M. Ma, S. Qin, Y. Kang, Y. Dang, S. Rajmohan, Q. Lin *et al.*, "Xpert: Empowering incident management with query recommendations via large language models," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–13.

[24] X. Qin, M. Ma, Y. Zhao, J. Zhang, C. Du, Y. Liu, A. Parayil, C. Bansal, S. Rajmohan, Í. Goiri *et al.*, "How different are the cloud workloads? characterizing large-scale private and public cloud workloads," in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2023, pp. 522–530.

[25] S. Zhang, P. Jin, Z. Lin, Y. Sun, B. Zhang, S. Xia, Z. Li, Z. Zhong, M. Ma, W. Jin *et al.*, "Robust failure diagnosis of microservice system through multimodal data," *IEEE Transactions on Services Computing*, vol. 16, no. 6, pp. 3851–3864, 2023.

[26] C. Luo, P. Zhao, B. Qiao, Y. Wu, H. Zhang, W. Wu, W. Lu, Y. Dang, S. Rajmohan, Q. Lin, and D. Zhang, "NTAM: neighborhood-temporal attention model for disk failure prediction in cloud platforms," in *Proceedings of the International World Wide Web Conferences*. ACM, 2021, pp. 1181–1191.

[27] Y. Liu, H. Yang, P. Zhao, M. Ma, C. Wen, H. Zhang, C. Luo, Q. Lin, C. Yi *et al.*, "Multi-task hierarchical classification for disk failure prediction in online service systems," in *Proceedings of the Conference on Knowledge Discovery and Data Mining*, 2022, pp. 3438–3446.

[28] Q. Lin, K. Hsieh, Y. Dang, H. Zhang, K. Sui, Y. Xu, J.-G. Lou, C. Li, Y. Wu, R. Yao *et al.*, "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, 2018, pp. 480–490.

[29] M. Ma, Y. Liu, Y. Tong, H. Li, P. Zhao, Y. Xu, H. Zhang, S. He, L. Wang, Y. Dang *et al.*, "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*, 2022, pp. 1453–1464.

[30] L. Li, X. Zhang, S. He, Y. Kang, H. Zhang, M. Ma, Y. Dang, Z. Xu, S. Rajmohan, Q. Lin *et al.*, "Conan: Diagnosing batch failures for cloud systems," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2023, pp. 138–149.

[31] S. Zhang, Y. Liu, W. Meng, Z. Luo, J. Bu, S. Yang, P. Liang, D. Pei, J. Xu, Y. Zhang *et al.*, "Prefix: Switch failure prediction in datacenter networks," *Proceedings of the ACM on Measurement and Analysis of Computing Systems (SIGMETRICS)*, vol. 2, no. 1, pp. 1–29, 2018.

[32] C. Xu, G. Wang, X. Liu, D. Guo, and T.-Y. Liu, "Health status assessment and failure prediction for hard drives with recurrent neural networks," *Transactions on Computers*, vol. 65, no. 11, pp. 3502–3508, 2016.

[33] J. Zhang, J. Wang, L. He, Z. Li, and P. S. Yu, "Layerwise perturbation-based adversarial training for hard drive health degree prediction," in *Proceedings of the International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 1428–1433.

[34] X. Sun, K. Chakrabarty, R. Huang, Y. Chen, B. Zhao, H. Cao, Y. Han, X. Liang, and L. Jiang, "System-level hardware failure prediction using deep learning," in *Proceedings of the Design Automation Conference (DAC)*. ACM, 2019, p. 20.

[35] S. Levy, R. Yao, Y. Wu, Y. Dang, P. Huang, Z. Mu, P. Zhao, T. Ramani, N. Govindaraju, X. Li *et al.*, "Predictive and adaptive failure mitigation to avert production cloud {VM} interruptions," in *Proceedings of the Operating Systems Design and Implementation (OSDI)*. USENIX, 2020, pp. 1155–1170.

[36] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2018, pp. 13–24.

[37] C. Elkan and K. Noto, "Learning classifiers from only positive and unlabeled data," in *Proceedings of the international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 213–220.

[38] M. N. Nguyen, X.-L. Li, and S.-K. Ng, "Positive unlabeled learning for time series classification," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, ser. IJCAI'11. AAAI Press, 2011, p. 1421–1426.

[39] X.-L. Li, P. S. Yu, B. Liu, and S.-K. Ng, "Positive unlabeled learning for data stream classification," in *Proceedings of the SIAM international conference on data mining (SDM)*. SIAM, 2009, pp. 259–270.

[40] B. Liu, W. S. Lee, P. S. Yu, and X. Li, "Partially supervised classification of text documents," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 2, no. 485, 2002, pp. 387–394.

[41] X. Li and B. Liu, "Learning to classify texts using positive and unlabeled data," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, vol. 3, 2003, pp. 587–592.

[42] W. S. Lee and B. Liu, "Learning with positive and unlabeled examples using weighted logistic regression," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 3, 2003, pp. 448–455.

[43] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," in *Proceedings of the International Conference on Data Mining (ICDM)*. IEEE, 2003, pp. 179–186.

[44] M. C. Du Plessis, G. Niu, and M. Sugiyama, "Analysis of learning from positive and unlabeled data," *Proceedings of the Advances in Neural Information Processing Systems*, vol. 27, 2014.

[45] R. Kiryo, G. Niu, M. C. Du Plessis, and M. Sugiyama, "Positive-unlabeled learning with non-negative risk estimator," *Proceedings of the Advances in Neural Information Processing Systems*, vol. 30, 2017.

[46] Y. Chen, X. Yang, Q. Lin, H. Zhang, F. Gao, Z. Xu, Y. Dang, D. Zhang, H. Dong, Y. Xu, H. Li, and Y. Kang, "Outage prediction and diagnosis for cloud service systems," in *Proceedings of the International World Wide Web Conferences*. ACM, 2019, pp. 2659–2665.

[47] H. Zhou, M. Chen, Q. Lin, Y. Wang, X. She, S. Liu, R. Gu, B. C. Ooi, and J. Yang, "Overload control for scaling wechat microservices," in *Proceedings of the Symposium on Cloud Computing (SoCC)*, ser. SoCC '18. New York, NY, USA: ACM, 2018, p. 149–161.

[48] Wikipedia contributors, "S.m.a.r.t. — Wikipedia, the free encyclopedia," 2022, [Online; accessed 28-September-2022]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=S.M.A.R.T.&oldid=1109378579

[49] S. Lu, B. Luo, T. Patel, Y. Yao, D. Tiwari, and W. Shi, "Making disk failure predictions smarter!" in *Proceedings of the Conference on File and Storage Technologies (FAST)*. USENIX, 2020, pp. 151–167.

[50] H. Xu, N. Kang, G. Zhang, C. Xie, X. Liang, and Z. Li, "Nasoa: Towards faster task-oriented online fine-tuning with a zoo of models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 5097–5106.

[51] T. D. Sets, "Large-scale disk failure prediciton dataset," https://tianchi.aliyun.com/dataset/dataDetail?dataId=70251, 2022.

[52] Backblaze, "The backblzae hard drive data and stats," https://www.backblaze.com/b2/hard-drive-test-data.html, 2022.

[53] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, 2005, pp. 273–286.

[54] M. C. du Plessis, G. Niu, and M. Sugiyama, "Analysis of learning from positive and unlabeled data," in *Proceedings of the Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. MIT Press, 2014.

[55] M. D. Plessis, G. Niu, and M. Sugiyama, "Convex formulation for learning from positive and unlabeled data," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1386–1394.

[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proceedings of the Advances in Neural Information Processing Systems*. MIT Press, 2019, vol. 32, pp. 8024–8035.

[57] C. Zhao, M. Ma, Z. Zhong, S. Zhang, Z. Tan, X. Xiong, L. Yu, J. Feng, Y. Sun *et al.*, "Robust multimodal failure detection for microservice systems," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 5639–5649.

[58] J. Li, X. Ji, Y. Jia, B. Zhu, G. Wang, Z. Li, and X. Liu, "Hard drive failure prediction using classification and regression trees," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*. IEEE/IFIP, 2014, pp. 383–394.

[59] M. M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann, "Predicting disk replacement towards reliable data centers," in *Proceedings of the Knowledge Discovery and Data Mining*. ACM, 2016, pp. 39–48.

[60] X. Huang, "Hard drive failure prediction for large scale storage system," Ph.D. dissertation, UCLA, 2017.

[61] Y. Xu, K. Sui, R. Yao, H. Zhang, Q. Lin, Y. Dang, P. Li, K. Jiang, W. Zhang, J. Lou, M. Chintalapati, and D. Zhang, "Improving service availability of cloud systems by predicting disk error," in *Proceedings of the USENIX Annul Technical Conference (ATC)*. USENIX, 2018, pp. 481–494.