StreamDFP: A General Stream Mining Framework for Adaptive Disk Failure Prediction

Shujie Han, Patrick P. C. Lee, Zhirong Shen, Cheng He, Yi Liu, and Tao Huang

Abstract—We explore machine learning for accurately predicting imminent disk failures and hence providing proactive fault tolerance for modern large-scale storage systems. Current disk failure prediction approaches are mostly offline and assume that the disk logs required for training learning models are available a priori. However, disk logs are often continuously generated as an evolving data stream, in which the statistical patterns vary over time (also known as concept drift). Such a challenge motivates the need of online techniques that perform training and prediction on the incoming stream of disk logs in real time, while being adaptive to concept drift. We first measure and demonstrate the existence of concept drift on various disk models in production. Motivated by our study, we design StreamDFP, a general stream mining framework for disk failure prediction with concept-drift adaptation based on three key techniques, namely online labeling, concept-drift-aware training, and general prediction, with a primary objective of supporting various machine learning algorithms. We extend StreamDFP to support online transfer learning for minority disk models with concept-drift adaptation. Our evaluation shows that StreamDFP improves the prediction accuracy significantly compared to without concept-drift adaptation under various settings, and achieves reasonably high stream processing performance.

Index Terms—disk failure prediction, stream mining, concept drift, and online transfer learning

1 Introduction

Maintaining storage reliability is a critical yet challenging requirement for modern cloud-scale data centers, typically composed of thousands to millions of disks [26], [40]. In large-scale disk deployment, disk failures are prevalent [5]. Although traditional redundancy mechanisms, such as replication and RAID, are widely used for data protection, they are no longer sufficient for providing strong reliability guarantees in the face of prevalent failures [26].

To complement existing redundancy mechanisms, we explore the prediction of imminent disk failures based on *machine learning* as a proactive fault tolerance mechanism to pinpoint and replace soon-to-fail disks, before the actual disk failures happen. In particular, various machine learning algorithms (e.g., [5], [23]–[25], [27], [37], [40], [45]) are shown to achieve highly accurate prediction. Such algorithms capture disk logs with performance and reliability statistics (e.g., SMART (Self-Monitoring, Analysis and Reporting Technology)) as the training data from a set of disks with known labels (i.e., healthy or failed). They train a prediction model using the training data, and use the trained prediction model to predict if any unknown disk (i.e., no label) will remain healthy or become failed in near future. Evaluation on production workloads (e.g., SMART logs from Backblaze [1]) justifies the effectiveness of machine learning; for example, over 95% of failed disks can be predicted in advance with a very small false positive rate [5], [23], [27], [45].

Existing disk failure prediction schemes are mostly *offline*, meaning that all training data must be available in advance before training any prediction model. On the other hand, in practice, disk logs are continuously generated from disks over time. With the enormous scale of the disk population in production environments, it is infeasible to keep all past data for training, rendering offline approaches inadequate for long-term use. Recent work [37] explores *online* disk failure prediction based on the Online Random Forests (ORF) algorithm, by labeling the healthy and failed disk samples and updating the prediction model on the fly. We believe that online prediction is essential for large-scale disk deployment.

However, how to generalize online disk failure prediction for various machine learning algorithms remains unexplored and non-trivial. As different disk models are subject to reliability heterogeneity [22], it is impractical to identify the "best" learning algorithm that applies to all disk models. More importantly, the statistical patterns of disk logs vary over time (e.g., due to the aging of disks, or the additions/removals of disks in production). Such a phenomenon, known as *concept drift* [12], implies that we must carefully identify the proper window of samples for training: if we choose too few samples, we do not have sufficient samples to build an accurate prediction model; if we choose too many samples, the prediction model may be disturbed by the old samples that no longer correctly capture the failure characteristics of the current pool of disks due to concept drift.

This motivates us to regard disk failure prediction as a *stream mining problem*. By viewing disk logs as an evolving stream of time-series samples, we process the samples

- *S. Han is with Peking University, Beijing, China (shujiehan@pku.edu.cn) and the Chinese University of Hong Kong, Shatin, N.T., Hong Kong.*
- *P. Lee is with the Chinese University of Hong Kong, Shatin, N.T., Hong Kong (pclee@cse.cuhk.edu.hk).*
- *Z. Shen is with Xiamen University, Xiamen, China (shenzr@xmu.edu.cn).*
- *C. He, Y. Liu, and T. Huang are with Alibaba Group.*

- *An earlier conference version of this paper appeared in [16]. In this extended version, we integrate artificial neural network learning into* StreamDFP *and extend* StreamDFP *with online transfer learning for minority disk models. We also include additional HDD and SSD datasets from Backblaze and Alibaba in our evaluation.*
- *Corresponding author: Patrick P. C. Lee.*

through the following steps in real time: (i) train the prediction model incrementally over the stream of samples, (ii) detect concept drift and adapt the prediction model using a properly tuned number of samples, and (iii) predict the failure status of any unknown disk. Prior work has proposed algorithms on adapting machine learning to concept drift in stream mining (Section 2). An open question is to support and customize various machine learning algorithms with concept-drift adaptation for a diverse mix of disk models in disk failure prediction.

We propose StreamDFP, a general stream mining framework for disk failure prediction with concept-drift adaptation. StreamDFP is designed to support a variety of machine learning algorithms (rather than specific algorithms), based on three key techniques: (i) *online labeling*, which labels the samples for a disk on the fly; (ii) *concept-drift-aware training*, which incorporates concept-drift adaptation when training a prediction model; and (iii) *general prediction*, which supports both classification (i.e., whether a disk will fail in near future) and regression (i.e., how likely a disk will fail in near future). To summarize, this paper makes the following contributions:

- We motivate our work via an extensive measurement study on five SMART datasets on hard-disk drives (HDDs), four from the public Backblaze dataset [1] and one from Alibaba; the latter has a much larger disk population than the total of the four Backblaze datasets. We demonstrate not only the existence of concept drift in all five datasets, but also the variation of concept-drift existence across healthy/failed disks, disk models, and SMART attributes.
- We present the complete design of StreamDFP as a general stream mining framework that extracts features, labels samples, and trains a prediction model, all in real-time. We also incorporate online transfer learning [43] into StreamDFP for the prediction of minority disk models.
- We implement a complete prototype of StreamDFP based on Massive Online Analysis (MOA) [4] and integrate an artificial neural network algorithm, called Multilayer Perceptron (MLP) [19], into StreamDFP.
- We evaluate both prediction accuracy and stream processing performance of nine decision-tree-based algorithms and MLP on 15 datasets in total, including 12 datasets of HDDs from Backblaze and Alibaba as well as three datasets of solid-state drives (SSDs) from Alibaba. StreamDFP increases the precision, recall, and F1-score by 27.5-71.8%, 15.7-37.4%, and 26.8-53.2%, respectively, through concept-drift adaptation. It increases the prediction accuracy of online transfer learning for minority disk models through concept-drift adaptation (e.g., by up to 51.8% in F1-score). It supports fast stream processing: it performs training and prediction in 13.5 seconds on the daily SMART data of 37 K disks.

We open-source our StreamDFP prototype at **http://adslab.cse.cuhk.edu.hk/software/streamdfp**.

2 Background

2.1 Disk Failure Prediction

Our goal is to predict imminent disk failures over a collection of disks in production based on SMART statistics. SMART is a widely adopted disk monitoring tool for collecting performance and reliability statistics of a disk. Modern disks include SMART in their firmware. With SMART enabled, a disk periodically reports various numerical values (called *attributes*) on operational status and error information. Some SMART attributes provide useful indicators for soon-to-fail disks. For example, RAIDShield [26] suggests to proactively replace a disk in production whose reallocated sector count (i.e., the attribute SMART-5) exceeds 200. However, checking SMART attributes against thresholds for disk failure prediction is highly error-prone, as its accuracy heavily depends on how the thresholds are configured. In this work, we explore the use of machine learning in disk failure prediction.

Specifically, we formulate our disk failure prediction as a *stream mining problem*, by viewing the SMART attributes emitted by disks as a stream of *samples* over a time series. For each disk i, where i is a unique disk identifier, we denote the SMART attributes emitted by disk i at time t as a vector $\mathbf{x}_t^i$ (called the *input variable*), and denote the failure status of disk i at time t as a scalar variable y_t^i (called the *target variable*). We assume that the SMART attributes are generated at the granularity of *days*, so each time t refers to a particular day. We feed $\mathbf{x}_t^i$ into a *prediction model* (denoted by $\mathcal{M}$) to predict the future failure status of disk i (denoted by $\hat{y}_t^i$) (e.g., within the next 30 days). As the true output for y_t^i is known, we update $\mathcal{M}$ over time with $(\mathbf{x}_t^i, y_t^i)$ (called a *labeled sample*). We also refer to the samples that correspond to the failed disks and healthy disks as *positive samples* and *negative samples*, respectively.

In practice, we collect SMART attributes and predict failures over a collection of disks simultaneously at each time point. For brevity of discussion, we omit the superscript i and use $\mathbf{x}_t$ and y_t to refer to the SMART attributes and the failure status of the *whole* collection of disks, respectively.

We consider two types of prediction: (i) *classification*, in which we predict if disk i is either failed or healthy in the future, and y_t^i is equal to either 1 or 0, respectively; and (ii) *regression*, in which we predict the likelihood that disk i is failed, and set y_t^i as some continuous value between 0 and 1.

2.2 Concept Drift

Concept drift [12] describes the phenomenon that the relationship between the input variables and the target variable continuously changes over time. Mathematically, let t_0 and t_1 be two time points in a stream (assuming $t_0 < t_1$), and $p(\mathbf{x}_t, y_t)$ be the joint probability of $\mathbf{x}_t$ and y_t at time t. We say that concept drift occurs if $p(\mathbf{x}_{t_0}, y_{t_0}) \neq p(\mathbf{x}_{t_1}, y_{t_1})$; in this case, the prediction model $\mathcal{M}$ can no longer accurately map $\mathbf{x}_{t_1}$ to y_{t_1}.

In our problem, we focus on detecting the concept drift in $p(y_t|\mathbf{x}_t)$ (i.e., the posterior probability of the target variable y_t given the input variable $\mathbf{x}_t$), as it describes the change of our prediction results. Based on the Bayesian decision theory, we can express $p(y_t|\mathbf{x}_t) = \frac{p(y_t)p(\mathbf{x}_t|y_t)}{p(\mathbf{x}_t)}$, where $p(\mathbf{x}_t)$ is the marginal probability of $\mathbf{x}_t$, $p(y_t)$ is the prior probability of y_t, and $p(\mathbf{x}_t|y_t)$ is the conditional probability of $\mathbf{x}_t$ given y_t. Thus, the change of $p(y_t|\mathbf{x}_t)$ can be characterized as the changes in the components $p(y_t)$, $p(\mathbf{x}_t)$, and $p(\mathbf{x}_t|y_t)$. We measure the changes in such components (Section 3.2). Our goal is to adapt the prediction model $\mathcal{M}$ with $p(y_t|\mathbf{x}_t)$, whose change can affect the prediction accuracy.

Category	Algorithm	Change detector
Classification tree	Hoeffding tree (HT) [9]	None
	Hoeffding adaptive tree (HAT) [3]	ADWIN [2]
Regression tree	Fast incremental model trees with drift detection (FIMT-DD) [21]	PH test [29]
Ensemble learning	Oza's bagging (Bag) [31]	None
	Oza's boosting (Boost) [31]	None
	Online random forests (RF) [14]	None
	Bagging with ADWIN (BA) [3]	ADWIN [2]
	Boosting-like online ensemble (BOLE) [8]	DDM [11]
	Adaptive random forests (ARF) [14]	ADWIN [2]
Neural networks	Multilayer perceptron (MLP) [19]	Backpropagation [17]

TABLE 1: Overview of incremental learning algorithms.

2.3 Change Detection

We perform change detection in stream mining to identify the existence of concept drift in $p(y_t|\mathbf{x}_t)$. Specifically, we define the absolute error, denoted by ϵ_t^i, at time t as $\epsilon_t^i = |\hat{y}_t^i - y_t^i|$, where $\hat{y}_t^i$ and y_t^i denote the predicted and true output for disk i at time t, respectively. We take a stream of ϵ_t^i's over a time window as input in change detection.

We can apply different change detectors. For example, ADaptive sliding WINdow (ADWIN) [2] keeps a variable-size sliding detection window of the most recent samples. It partitions the detection window into two sub-windows and monitors each of their average values. If the two sub-windows have significantly different average values[1], then it implies that a change happens, and ADWIN drops the older sub-window and replaces the detection window with the newer sub-window. Other change detectors include the Page-Hinckley (PH) test [32] and the Drift Detection Model (DDM) [11].

2.4 Incremental Learning Algorithms

To support adaptive disk failure prediction, we consider several state-of-the-art *incremental learning* algorithms that perform prediction on an input data stream and continuously update the prediction model using the labeled samples. Table 1 summarizes the incremental learning algorithms that we consider in the paper. Such algorithms are used to train a prediction model for classification or regression. We can classify them into two categories: *decision-tree-based algorithms* and *neural networks*. We can further classify the decision-tree-based algorithms into two sub-categories: *single decision trees* and *ensemble learning*. Instead of advocating a specific incremental learning algorithm for prediction, whose effectiveness highly varies across disk brands and models (Section 1), we focus on supporting general incremental learning algorithms for disk failure prediction.

Single decision trees. Several incremental learning algorithms maintain a single decision tree for prediction. The Hoeffding Tree (HT) [9] recursively updates the tree structure using a small subset of labeled samples and decides how many labeled samples are modeled by each tree node using the Hoeffding bound [18]. The Hoeffding Adaptive Tree (HAT) [3] builds on HT and associates ADWIN with each tree node. If ADWIN detects concept drift at a tree node, HAT creates an alternate tree rooted at the tree node and trains it separately. If the original tree has a larger error than the alternate tree, it will be replaced by the alternate tree. Both HT and HAT are designed for classification. On the other hand, FIMT-DD [21] is a regression tree that uses the PH test [29] as the change detector at each tree node. It has similar operations of creating and managing alternate trees like HT and HAT.

Ensemble learning. Single decision trees are limited in both diversity and lookahead ability for large amounts of data [33]. Ensemble learning is proposed to combine multiple decision trees as base learners in prediction. Classical (offline) ensemble learning methods include *bagging* [6], which draws random samples with replacements during training to improve the overall accuracy; *boosting* [10], which trains prediction models iteratively by increasing the weights for less accurate learners to improve the overall accuracy; and *random forests* [7], which train multiple base learners on re-sampled data (similar to bagging) and randomly select subsets of attributes for tree updates. To support online ensemble learning, we adopt Oza's online versions of bagging and boosting [31] and the online random forests in [14], such that they update the prediction models based on incoming labeled samples; however, these online methods do not address concept drift.

Bagging with ADWIN (BA) [3], Boosting-like online ensemble (BOLE) [8], and Adaptive Random Forests (ARF) [14] add concept-drift adaptation to the online versions of bagging, boosting, and random forests, respectively. Their idea is to associate a change detector with each decision tree in an ensemble of trees. If a tree has concept drift detected, it will be removed and substituted by a new tree root (e.g., in BA and BOLE) or a newly trained tree (e.g., in ARF).

Neural networks. We first consider *Multilayer Perceptron (MLP)* [19], a classical artificial neural network learning algorithm. Specifically, the MLP neural network comprises interconnected units called *neurons*. The neurons are connected by weights to form multiple learning layers, including an input layer, one or more hidden layers, and an output layer. The MLP neural network is often trained via stochastic gradient descent [34], and uses backpropagation [17] as concept-drift adaptation to propagate the errors between the predicted and true outputs backward to the neural network. Here, we consider MLP with backpropagation on stream mining, as opposed to offline neural network learning (e.g., [45]). We also consider *Recurrent Neural Network (RNN)* on stream mining in the digital supplementary file. However, we do not consider *Convolutional Neural Network (CNN)* on stream mining, as it needs to buffer much more time-series samples for training than the other incremental learning algorithms and hence incurs high memory overhead; we pose the design of CNN for stream mining as future work.

3 Concept-Drift Analysis

3.1 Datasets

We present a measurement study of the existence of concept drift on production datasets. Our concept-drift analysis

1. It is tested against the null hypothesis that the change between two average values is upper-bounded by a threshold (based on the Hoeffding bound [18]) with a significance level (set as 0.002 by default).

Dataset ID	Disk model	Capacity	Disk count	# failures	Period	Duration (months)
D1	Seagate ST3000DM001	3 TB	4,516	1,269	2014-01-31 to 2015-10-31	21
D2	Seagate ST4000DM000	4 TB	37,015	3,275	2013-05-10 to 2018-12-31	68
D3	Seagate ST12000NM0007	12 TB	35,462	740	2017-09-06 to 2019-06-30	22
D4	Hitachi HDS722020ALA330	2 TB	4,601	226	2013-04-10 to 2016-12-31	45
D5	Private disk model of Alibaba	6 TB	$\sim$250 K	$\sim$1,000	2019-01-01 to 2019-05-31	5

TABLE 2: Overview of datasets for concept-drift analysis.

ID	SMART attribute name	D1	D2	D3	D4	D5
S1	Read error rate	r|n	r|n	r|n	r|n	r|n
S2	Throughput performance	–	–	–	r|n	–
S3	Spin-up time	r|n	r|n	r|n	r|n	r|n
S4	Start/stop count	r|n	r|n	r|n	r|n	r|n
S5	Reallocated sector count	r|n	r|n	r|n	r|n	r|n
S7	Seek error rate	r|n	r|n	r|n	r|n	r|n
S8	Seek time performance	–	–	–	r|n	–
S9	Power-on hours	r|n	r|n	r|n	r|n	r|n
S10	Spin retry count	r|n	r|n	r|n	r|n	r|n
S12	Power cycle count	r|n	r|n	r|n	r|n	r|n
S183	SATA downshift error count	r|n	r|n	–	–	–
S184	End-to-end error	r|n	r|n	–	–	r|n
S187	Reported uncorrectable errors	r|n	r|n	r|n	–	r|n
S188	Command timeout	r|n	r|n	r|n	–	r|n
S189	High fly writes	r|n	r|n	–	–	r|n
S190	Airflow temperature	r|n	r|n	r|n	–	r|n
S191	G-sense error rate	r|n	r|n	–	–	r|n
S192	Power-off retract count	r|n	r|n	r|n	r|n	r|n
S193	Load cycle count	r|n	r|n	r|n	r|n	r|n
S194	Temperature celsius	r|n	r|n	r|n	r|n	r|n
S195	Hardware ECC recovered	–	–	r|n	–	r|n
S196	Reallocation event count	–	–	–	r|n	–
S197	Current pending sector	r|n	r|n	r|n	r|n	r|n
S198	Uncorrectable sector count	r|n	r|n	r|n	r|n	r|n
S199	UltraDMA CRC error count	r|n	r|n	r|n	r|n	r|n
S200	Write error rate	–	–	r|n	–	–
S240	Head flying hours	r|n	r|n	r|n	–	r|n
S241	Total LBAs written	r|n	r|n	r|n	–	r|n
S242	Total LBAs read	r|n	r|n	r|n	–	r|n

"r": Raw value; "n": Normalized value.
– The SMART attribute is not collected or the value is not provided.

TABLE 3: Overview of collected SMART attributes.

builds on five SMART datasets collected from two independent sources, as shown in Table 2. Our datasets are diverse, covering different disk models, manufacturers, and production environments. Thus, they allow us to validate the generality of our findings. We will consider additional SMART datasets[2] for more comprehensive evaluation of STREAMDFP (Section 7).

The first group of datasets is collected and made publicly available by Backblaze [1], which has released SMART datasets for various HDD models in its data centers since 2013. Here, we select the datasets namely D1, D2, D3, and D4, on four disk models that are among the largest disk populations with the highest disk failure rates. Note that the disk models are also selected and evaluated by prior studies [5], [27], [37].

The remaining dataset is a private SMART dataset collected at Alibaba. The dataset, namely D5, belongs to a specific HDD model with around 250 K disks, which are at least 6$\times$ as many as D2 and D3 and nearly 55$\times$ as many as D1 and D4. However, it only has around 1,000 failures (even fewer than those of D1 and D2), implying that the dataset is highly imbalanced as the failure rate is extremely low.

Table 3 provides an overview of the collected SMART attributes. The datasets span 29 SMART attributes in total. Each collected SMART attribute includes both the raw and normalized values.

3.2 Measurement of Concept Drift

We now study the existence of concept drift in each dataset. Recall from Section 2.2 that the change of $p(y_t|\mathbf{x}_t)$ can be characterized through the three components $p(y_t)$, $p(\mathbf{x}_t)$, and $p(\mathbf{x}_t|y_t)$. In the following, we measure the changes of each component to motivate the need of adapting to the change of $p(y_t|\mathbf{x}_t)$ in our disk failure prediction problem[3]. Here, we focus on binary classification (i.e., a disk is healthy or failed).

Measurement of $p(y_t)$. To understand the change of $p(y_t)$, we measure the percentage of failed disks over time (i.e., the percentage of $y_t^i = 1$ over the whole collection of disks) for each dataset. Given the long duration of each Backblaze dataset, we conduct the measurement on D1, D2, D3, and D4 on a monthly basis; on the other hand, we conduct the measurement on D5 on a daily basis.

Figure 1 shows the results. The percentage of failed disks highly oscillates over time. For example, the percentages of failed disks of D1 and D4 can reach as high as 9.7% and 9.1%, respectively; for D5, its daily percentage of failed disks ranges from 0 to 0.09%. One main reason for the highly varying failure rates is that new disks are kept being added, or old disks are kept being retired, over the entire measurement span, so the number of healthy disks varies significantly.

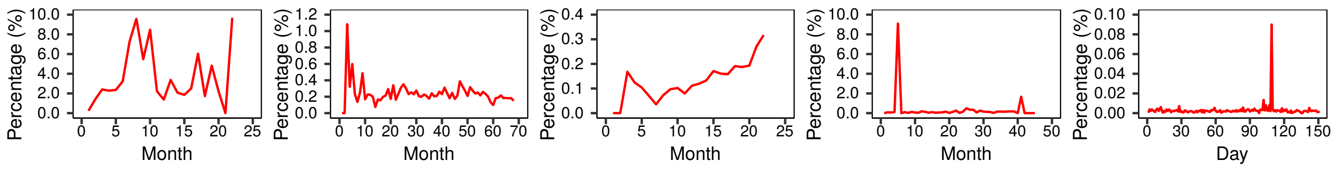
Measurement of $p(\mathbf{x}_t)$. We now measure the change of $p(\mathbf{x}_t)$. Here, we use the two-sample Kolmogorov-Smirnov (KS) test [28] to measure the change of each SMART attribute (based on its raw values) in $p(\mathbf{x}_t)$. Specifically, we compare the samples in two time periods, denoted by t_0 and t_1 under the null hypothesis that the samples of t_0 and t_1 are both drawn from the same probability distribution. We measure the p-value, such that a p-value that is smaller than a threshold (currently set as 5%) will reject the null hypothesis. Here, we measure how many SMART attributes have changed distributions (i.e., their null hypotheses are rejected).

We set the granularity of time periods for the Backblaze datasets as years, while that for the dataset D5 as months. Our datasets are imbalanced, with much fewer failed disks than healthy disks. Thus, we *downsample* the healthy disks to prevent them from dominating the overall distributions (we also apply downsampling in prediction; see Section 4.4). For failed disks, we take all their (positive) samples over a time period, while for healthy disks, we only take their (negative) samples at the last day of a time period.

Table 4 shows the number of SMART attributes with changed distributions in $p(\mathbf{x}_t)$ over the total number of SMART attributes being collected (we will discuss the changes of $p(\mathbf{x}_t|\text{failed})$ and $p(\mathbf{x}_t|\text{healthy})$ later). We see

2. Our findings of concept-drift analysis still hold on these datasets, but we omit the results from the paper in the interest of space.

3. Note that the changes of all three components do not necessarily imply the change of $p(y_t|\mathbf{x}_t)$. However, we claim that this is unlikely, and our evaluation in Section 7 shows that adapting to the change of $p(y_t|\mathbf{x}_t)$ is critical to improve the prediction accuracy.



(a) D1 (b) D2 (c) D3 (d) D4 (e) D5

Fig. 1: Percentage of failed disks of each dataset in each month (for D1, D2, D3, and D4) or each day (for D5) in the whole duration.

D1	Total	$p(\mathbf{x}_t)$	$p(\mathbf{x}_t\|\text{healthy})$	$p(\mathbf{x}_t\|\text{failed})$
2014 vs. 2015	24	15	10	8
D2	**Total**	$p(\mathbf{x}_t)$	$p(\mathbf{x}_t\|\text{healthy})$	$p(\mathbf{x}_t\|\text{failed})$
2013 vs. 2014	5	2	2	2
2014 vs. 2015	24	15	8	16
2015 vs. 2016	24	15	7	10
2016 vs. 2017	24	14	6	8
2017 vs. 2018	24	14	7	8
D3	**Total**	$p(\mathbf{x}_t)$	$p(\mathbf{x}_t\|\text{healthy})$	$p(\mathbf{x}_t\|\text{failed})$
2017 vs. 2018	22	13	13	8
2018 vs. 2019	22	14	13	11
D4	**Total**	$p(\mathbf{x}_t)$	$p(\mathbf{x}_t\|\text{healthy})$	$p(\mathbf{x}_t\|\text{failed})$
2013 vs. 2014	5	2	1	1
2014 vs. 2015	17	6	4	5
2015 vs. 2016	17	6	4	3
D5	**Total**	$p(\mathbf{x}_t)$	$p(\mathbf{x}_t\|\text{healthy})$	$p(\mathbf{x}_t\|\text{failed})$
Jan. vs. Feb.	24	13	13	1
Feb. vs. Mar.	24	13	13	4
Mar. vs. Apr.	24	13	11	15
Apr. vs. May	24	13	13	13

"Total": total number of collected SMART attributes; "$p(\mathbf{x}_t)$", "$p(\mathbf{x}_t|\text{healthy})$", "$p(\mathbf{x}_t|\text{failed})$": numbers of SMART attributes with changed distributions. Note that ST4 and HD2 only collect five SMART attributes (S1, S5, S9, S194, and S197) in 2013.

TABLE 4: Number of SMART attributes with changed distributions.

D1	S5	S10	S184	S187	S188	S197	S198
2014 vs. 2015				○ ‡	○ † ‡	‡	‡
D2	**S5**	**S10**	**S184**	**S187**	**S188**	**S197**	**S198**
2013 vs. 2014		–	–	–	–	‡	–
2014 vs. 2015	‡	‡	‡	‡	○ † ‡	‡	‡
2015 vs. 2016	‡			‡	○ †	‡	‡
2016 vs. 2017	‡			○ ‡		‡	‡
2017 vs. 2018	‡			‡	○ †	‡	‡
D3	**S5**	**S10**	**S184**	**S187**	**S188**	**S197**	**S198**
2017 vs. 2018	‡		–				
2018 vs. 2019	‡		–			‡	‡
D4	**S5**	**S10**	**S184**	**S187**	**S188**	**S197**	**S198**
2013 vs. 2014		–	–	–	–		–
2014 vs. 2015	‡		–	–	–	‡	
2015 vs. 2016			–	–	–		
D5	**S5**	**S10**	**S184**	**S187**	**S188**	**S197**	**S198**
Jan. vs. Feb.							
Feb. vs. Mar.							
Mar. vs. Apr.	‡			‡		‡	‡
Apr. vs. May	‡			‡		‡	‡

○ $p(\mathbf{x}_t)$ shows a changed distribution.
† $p(\mathbf{x}_t|\text{healthy})$ shows a changed distribution.
‡ $p(\mathbf{x}_t|\text{failed})$ shows a changed distribution.
– The SMART attribute is not collected.

TABLE 5: Changed distributions for critical SMART attributes.

that a significant fraction of SMART attributes has changed distributions. For example, D2 has more than half of the SMART attributes with changed distributions.

We further study the changes of several *critical* SMART attributes defined by Backblaze, which provide strong indicators for disk failures [1]. Table 5 shows the presence of changed distributions for each critical SMART attribute based on the KS test. We observe the change of $p(\mathbf{x}_t)$ for D1 and D2 in S187 and S188 for most time periods. However, D3, D4, and D5 do not show a change of $p(\mathbf{x}_t)$ in all critical SMART attributes, meaning that the change mainly appears in other non-critical SMART attributes.

Measurement of $p(\mathbf{x}_t|y_t)$. Finally, we study the change in $p(\mathbf{x}_t|y_t)$. We consider two conditional probability distributions, $p(\mathbf{x}_t|\text{healthy})$ and $p(\mathbf{x}_t|\text{failed})$, for healthy and failed disks, respectively. We revisit Tables 4 and 5 on the changed distributions across the SMART attributes.

From Table 4, a significant fraction of SMART attributes (e.g., at least one-fourth for D2) has changed distributions for healthy and failed disks. However, in D2 and D4, failed disks generally have more SMART attributes with changed distributions than healthy disks, but in D1, D3, and D5, it is opposite. Thus, the effects of changed distributions vary across disk models.

From Table 5, failed disks generally show changed distributions in some of the critical SMART attributes (and in all critical SMART attributes for D2 from 2014 to 2015). For example, Figure 2 shows the cumulative distributions of the S5 raw values for the failed disks in D2 from 2014 to 2016, and it shows clear shifts in the cumulative distributions.

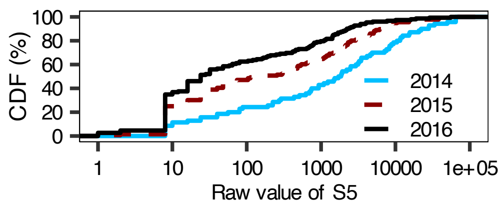


Fig. 2: Changed distributions of S5 for failed disks in D2.

One reason is that failed disks tend to show various failure symptoms on different critical SMART attributes (which measure the error counts), so the distributions of the critical SMART attributes also have high variations. However, the changed behaviors across disk models are highly varying for different critical SMART attributes.

Summary. Our measurement study shows two major observations. First, we observe the presence of changed distributions in $p(y_t)$, $p(\mathbf{x}_t)$, and $p(\mathbf{x}_t|y_t)$, indicating that the change of $p(y_t|\mathbf{x}_t)$ (i.e., concept drift) also likely exists. Second, the changed behaviors cannot be readily predicted, as they vary across healthy and failed disks, disk models, as well as SMART attributes. Thus, the mechanism for adapting to concept drift needs to be generally applicable for various changed behaviors.

4 Design

We present the design of StreamDFP, a general stream mining framework for disk failure prediction with concept-drift adaptation. Specifically, StreamDFP aims to address the following challenges:

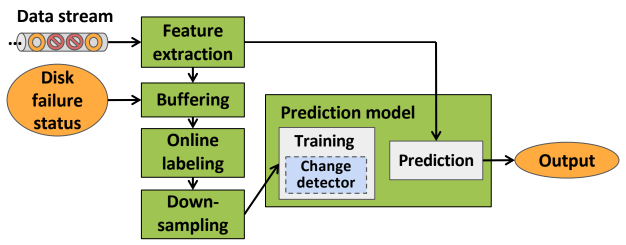


Fig. 3: Architecture of StreamDFP.

- **Online labeling.** Unlike offline learning, StreamDFP accesses a stream of samples from a collection of disks and labels the samples on the fly. It should accurately label the samples as either positive (for failed disks) or negative (for healthy disks) based on the current disk failure patterns.
- **Concept-drift-aware training.** StreamDFP builds on a number of incremental learning algorithms with concept-drift adaptation (Section 2.4). It should accurately detect and adapt to concept drift in training a prediction model specifically for disk failure prediction.
- **General prediction.** StreamDFP treats disk failure prediction as both classification and regression problems. For classification, StreamDFP directly answers if an unknown disk will remain healthy or will be failed in near future. For regression, StreamDFP should determine the likelihood that an unknown disk will fail.

4.1 Architectural Overview

Figure 3 shows the architecture of StreamDFP. StreamDFP takes a stream of samples on each day as input. It extracts SMART attributes as learning features (Section 4.2). It sets a sliding window to buffer recent samples and disk failure status, and labels the disks on the fly (Section 4.3). It down-samples the negative samples and feeds the labeled samples into the prediction model for training. With change detection enabled, StreamDFP detects concept drift explicitly during training and adapts the prediction model to concept drift. (Section 4.4). In the prediction phase, it uses the prediction model to output the prediction results (for both classification and regression) for an unknown disk (Section 4.5).

4.2 Feature Extraction

Given an input stream of samples, StreamDFP extracts the SMART attributes of each sample as the learning features for prediction model training. Here, we use *all* collected SMART attributes in Table 3 (including raw and normalized values) as learning features, instead of choosing a subset of SMART attributes based on historical disk logs like [37]. In practice, we may have no or only few historical disk logs for feature selection. Even though historical disk logs are available for us to identify the representative SMART attributes for failure characterization, the selected attributes may vary over time due to the changing distributions of the SMART attributes (Section 3.2). Thus, for simplicity, we take all collected SMART attributes as learning features.

4.3 Buffering and Online Labeling

StreamDFP needs to label the samples on the fly before feeding them into training. A straightforward approach is to label a disk sample as positive once the corresponding disk is diagnosed as failed, or as negative otherwise. However, a disk often does not fail immediately; instead, a soon-to-fail disk has actually shown failure symptoms (e.g., a sharp increase in the reallocated sector count [26]). Thus, StreamDFP also labels the samples of soon-to-fail disks as positive (in addition to the disk samples of actual failed disks), so as to better reflect the disk failure characteristics. A side benefit is that the proportion of positive samples also increases, which mitigates the well-known *data imbalance* issue in disk failure prediction [5], [27] as failed disks often account for a very small fraction over the entire disk population. A key challenge is how to label the samples of soon-to-fail disks on the fly.

Algorithm 1 Online Labeling

```
1:  procedure Main($t$, $\mathcal{W}$, $D_L$)
2:      for each failed disk $i$ on day $t$ do
3:          Find $y^i$ = time-series failure status of disk $i$ in $\mathcal{W}$
4:          if classification then
5:              Set $y^i_{t'} = 1$ for all $t' \in [t - D_L, t]$
6:          else if regression then
7:              for each day $t' = t - D_L$ to $t$ do
8:                  Set $y^i_{t'} = 1 - \frac{t-t'}{D_L+1}$
9:              end for
10:         end if
11:     end for
12: end procedure
```

StreamDFP buffers the recently received samples for online labeling. Specifically, it configures a sliding time window, denoted by $\mathcal{W}$, to buffer the samples of a sufficiently long recent period (30 days in our case), as well as the number of extra labeled days before the disk failure occurs, denoted by D_L. If a failed disk is found, StreamDFP labels the samples within D_L before the failure as positive (by default, all samples before failures are negative). Note that the number of samples within D_L may be less than the length of D_L when $\mathcal{W}$ is not full or limited samples are collected before a disk failure. By default, we set $D_L = 20$ days.

Algorithm details. Algorithm 1 shows the pseudo-code on online labeling (while buffering is done before the algorithm). It takes the inputs of the current day t, $\mathcal{W}$, and D_L.

We update the labels of the samples of all failed disks within D_L. We label the samples of soon-to-fail disks for classification and regression separately (Lines 2-11). For classification, we set the target variable $y^i_{t'} = 1$ for all $t' \in [t - D_L, t]$ (Lines 4-5). For regression, we update the labels with the failure probabilities. Specifically, for each day $t' \in [t - D_L, t]$, we define the failure probability $y^i_{t'}$ as the probability that disk i fails at any day from day $t - D_L$ up to day t'. Assuming that disk i fails at each day from day $t - D_L$ to day t with an equal probability $\frac{1}{D_L+1}$, the failure probability $y^i_{t'}$ is given by $y^i_{t'} = \frac{t'-(t-D_L)+1}{D_L+1} = 1 - \frac{t-t'}{D_L+1}$ (Line 8). Thus, $y^i_{t'}$ can be viewed as a linear function that increases with t', starting from $y^i_{t'} = \frac{1}{D_L+1}$ at $t' = t - D_L$ to $y^i_{t'} = 1$ at $t' = t$ (i.e., when the failure occurs) (Lines 6-10). Note that the failure probability defined here only approximates the likelihood that a soon-to-fail disk will actually fail over time, and we pose the analysis of the true failure probability of a disk as future work.

Algorithm 2 Downsampling and Training

```
1:  procedure MAIN($t$, $\mathcal{W}$, $\mathcal{M}$)
2:      Select all positive samples and the negative samples in
        the recent seven days from $\mathcal{W}$ into $\mathcal{W}'$
3:      for each $(\mathbf{x}_t^i, y_t^i) \in \mathcal{W}'$ do
4:          for each base learner $T \in \mathcal{M}$ do
5:              if $y_t^i$ == 0 then                ▷ negative samples
6:                  Set $k = Poisson(\lambda_n)$
7:              else                                ▷ positive samples
8:                  Set $k = Poisson(\lambda_p)$
9:              end if
10:             if $k > 0$ then
11:                 TRAIN($T$, $\mathbf{x}_t^i$, $y_t^i$, $k$)
12:                 Set $\hat{y}_t^i$ = PREDICT($\mathbf{x}_t^i$)
13:                 Update $T$ with DETECTCHANGE($T$, $\hat{y}_t^i$, $y_t^i$)
14:             end if
15:         end for
16:     end for
17: end procedure
```

4.4 Downsampling and Training

StreamDFP trains a prediction model based on the labeled samples. Before training, it is critical to mitigate the data imbalance issue [5], [27] for accurate prediction. In addition to labeling more samples as positive for soon-to-fail disks (Section 4.3), StreamDFP downsamples the negative samples to increase the proportion of positive samples. After downsampling, StreamDFP attaches a change detector to the prediction model to adapt to concept drift.

Algorithm details. Algorithm 2 shows the pseudo-code on downsampling and training. It takes the inputs of the current day t, $\mathcal{W}$, and $\mathcal{M}$.

StreamDFP downsamples the negative samples in a two-phase process. It first selects a subset of samples in $\mathcal{W}$, including all positive samples and the negative samples in the recent days (currently set as seven days to preserve sufficient negative samples), into $\mathcal{W}'$ (Line 2). It further downsamples the negative samples via *Poisson sampling*, which is commonly used in ensemble learning algorithms [14], [31] and online disk failure prediction [37]. We borrow the approach in [37] by customizing the hyper-parameters of the Poisson distribution, denoted by λ_n and λ_p, for the negative and positive samples, respectively. Specifically, we refer to a decision tree (for decision-tree-based learning) or a neuron (for neural network learning) as a *base learner*. For each base learner $T \in \mathcal{M}$, we generate a weight k from the respective Poisson distribution of either negative or positive samples, where k specifies the frequency that the sample is updated in the prediction model (Lines 5-9). Since a larger hyper-parameter implies a larger weight, we ensure that $\lambda_p > \lambda_n$ to ensure that the positive samples weigh more in the prediction model. More precisely, our evaluation varies λ_p and λ_n based on the given false positive rate for each incremental learning algorithm (Section 7). Note that we do not claim the novelty of this approach.

For each incremental learning algorithm, StreamDFP extends the corresponding prediction model $\mathcal{M}$ with a change detector (e.g., ADWIN [2], PH test [29], DDM [11], and backpropagation [17]) (Section 2.3). StreamDFP associates the change detector with each base learner $T \in \mathcal{M}$ (recall that $\mathcal{M}$ is composed of either decision trees or neurons (Section 2.4)). Specifically, if $k > 0$, StreamDFP first trains T with a labeled sample $(\mathbf{x}_t^i, y_t^i)$ and weight k (Line 11) and then performs the prediction to obtain $\hat{y}_t^i$ (Line 12). It then updates T with the detected changes between the predicted output $\hat{y}_t^i$ and the labeled target variable y_t^i (Line 13). Specifically, for decision-tree-based algorithms, StreamDFP compares $\hat{y}_t^i$ and y_t^i to detect if concept drift exists; if so, T is updated accordingly based on the incremental learning algorithm. For example, ARF [14] detects concept drift with ADWIN in each tree, which keeps the recent absolute errors between $\hat{y}_t^i$ and y_t^i in the detection window and monitors the change of average values of two sub-windows (Section 2.3). If ADWIN detects concept drift, ARF replaces T with a new decision tree trained by recent samples. For MLP, StreamDFP updates T with the propagated error between $\hat{y}_t^i$ and y_t^i backward from the output layer to the hidden layers.

Algorithm 3 StreamDFP

```
1:  Initialize $\mathcal{W}$ = empty sliding time window
2:  Initialize $\mathcal{M}$ = prediction model
3:  for each day $t$ do
4:      if $\mathcal{W}$ is full then
5:          Slide one day for $\mathcal{W}$
6:      end if
7:      Extract learning features to $\mathbf{x}_t$ from all disks
8:      Buffer ($\mathbf{x}_t$, $y_t$) into $\mathcal{W}$
9:      Call Algorithm 1 (online labeling) to label samples in $\mathcal{W}$
10:     Call Algorithm 2 (downsampling and training) to train
        $\mathcal{M}$ using $\mathcal{W}$
11:     if $\mathcal{W}$ is full then
12:         Set $\hat{y}_t = \mathcal{M}(\mathbf{x}_t)$
13:     end if
14: end for
```

4.5 Prediction

StreamDFP supports both classification and regression in prediction. For classification, it predicts whether a disk failure will occur, while for regression, it returns the failure probability of a disk within the near future. In particular, for regression, based on how we label a sample as positive (Section 4.3), the product $(1 - \hat{y}_t^i)(D_L + 1)$ can be viewed as the predicted residual lifetime of the disk.

Note that during training (Section 4.4), we call prediction once to detect concept drift (see Line 12 of Algorithm 2). However, we still need to call the prediction again after training, so that the prediction output is based on the updated prediction model due to concept drift.

4.6 Putting It All Together

Algorithm 3 shows the entire workflow of StreamDFP. We first initialize $\mathcal{W}$ and $\mathcal{M}$ (Lines 1-2). For each day t, if $\mathcal{W}$ is full, we slide $\mathcal{W}$ by one day (Lines 4-6). We then buffer the samples for online labeling as follows. We extract the learning features to $\mathbf{x}_t$ from all disks and buffer ($\mathbf{x}_t$, y_t) into $\mathcal{W}$ (Lines 7-8). We call Algorithm 1 to label samples in $\mathcal{W}$ and Algorithm 2 to train $\mathcal{M}$ using $\mathcal{W}$ (Lines 9-10). Finally, if $\mathcal{W}$ is full, it implies that we have buffered enough labeled samples for training and $\mathcal{M}$ is warmed up, and hence we use $\mathcal{M}$ with $\mathbf{x}_t$ to output the prediction results $\hat{y}_t$ (Lines 11-13).

5 Online Transfer Learning

In this section, we extend StreamDFP to support online transfer learning with concept-drift adaptation.

Background. In real-world disk deployment, some disk models, referred to as *minority disk models*, may have an insufficient number of samples for training the prediction model, which leads to poor prediction accuracy due to overfitting [36]. Thus, we apply *transfer learning* for disk failure prediction, whose idea is to leverage the prediction model trained on a disk model with a large number of disks (called the *source disk model*) to construct the prediction model for a minority disk model (called the *target disk model*). Specifically, we first train a prediction model (denoted by $\mathcal{M_S}$) on the samples from the source disk model. When the samples from the target disk model arrive, we start to train another prediction model (denoted by $\mathcal{M_T}$) and also update $\mathcal{M_S}$ with these samples. In the prediction phase, we combine the prediction results of both $\mathcal{M_S}$ and $\mathcal{M_T}$. To maintain the effectiveness of transfer learning, we assume that both source and target disk models are from the same manufacturer, so that both disk models are likely to share the similar SMART reporting approach.

Transfer learning has also been adopted by previous work [5], [42] for disk failure prediction. The main novelty here is to apply transfer learning into StreamDFP in the context of stream mining with concept-drift adaptation, meaning that we can only buffer a small number of samples for incremental learning as concept drift may exist in both source and target disk models.

Integration of online transfer learning into StreamDFP. To predict disk failures for minority disk models under stream mining, we extend StreamDFP with *online transfer learning* [43], a general online transfer learning approach that supports various classification algorithms. In particular, we adopt the mistake-driven homogeneous online transfer learning approach, called HomOTL-II, in [43], and make two modifications in our integration. First, after we train $\mathcal{M_S}$ with the samples from the source disk model, instead of fixing $\mathcal{M_S}$ as in [43], we still update $\mathcal{M_S}$ with the samples from the target disk model. This allows $\mathcal{M_S}$ to adapt to the existence of concept drift. Second, when combining the prediction results of $\mathcal{M_S}$ and $\mathcal{M_T}$, instead of discounting the weights of both $\mathcal{M_S}$ and $\mathcal{M_T}$ for an incorrectly classified sample as in [43], we only check if $\mathcal{M_S}$ incorrectly classifies the samples from the target disk model, and if so, we discount the weight of $\mathcal{M_S}$ (denoted by θ) by a discounting weight parameter β (where $\beta \in (0, 1)$) (i.e., without discounting the weight of $\mathcal{M_T}$ even if $\mathcal{M_T}$ classifies the samples from the target disk model incorrectly). The reason is that we now treat $\mathcal{M_T}$ as the dominant prediction model. If we also discount the weight of $\mathcal{M_T}$, the combined prediction results of $\mathcal{M_S}$ and $\mathcal{M_T}$ may become highly ineffective with the insufficient positive samples due to the data imbalance issue (Section 4.3). On the other hand, we still discount the weight of $\mathcal{M_S}$ as it may incorrectly capture the failure patterns due to the differences of failure characteristics between source and target disk models.

Algorithm details. Algorithm 4 shows the pseudo-code on downsampling and training under modified online transfer learning, by extending the original downsampling and training procedures in Algorithm 2. Specifically, it takes the inputs of the current day t, the weight of $\mathcal{M_S}$ (θ), the sliding time window for the target disk model ($\mathcal{W_T}$), $\mathcal{M_S}$, and $\mathcal{M_T}$. It selects a subset of samples from the target disk model $\mathcal{W_T}$ into $\mathcal{W_T}'$ (Line 2), similar to Line 2 of Algorithm 2. For each sample of the target disk model in $\mathcal{W_T}'$, it trains each tree in both $\mathcal{M_S}$ (i.e., our first modification) and $\mathcal{M_T}$ with downsampling and concept-drift adaptation using a given incremental learning algorithm (Lines 3-5), similar to Lines 4-17 of Algorithm 2. It next checks whether $\mathcal{M_S}$ correctly predicts the sample (Line 6) and discounts θ by β if the prediction is incorrect (Lines 7-9) (i.e., our second modification). Finally, it returns θ (Line 11).

Algorithm 4 Downsampling and Training under Modified Online Transfer Learning

```
 1: procedure Main($t$, $\theta$, $\mathcal{W_T}$, $\mathcal{M_S}$, $\mathcal{M_T}$)
 2:     Select all positive and the negative samples of target disk
        model in the recent seven days from $\mathcal{W_T}$ into $\mathcal{W_T}'$
 3:     for each $(\mathbf{x}_t^i, y_t^i) \in \mathcal{W_T}'$ do
 4:         Train each decision tree $T \in \mathcal{M_S}$ with downsampling
            and concept-drift adaption          ▷ our 1st modification
 5:         Train each decision tree $T \in \mathcal{M_T}$ with downsampling
            and concept-drift adaption
 6:         Set $\hat{y}_t^i = \mathcal{M_S}(\mathbf{x}_t)$          ▷ our 2nd modification
 7:         if $\hat{y}_t^i \neq y_t^i$ then          ▷ incorrect classification
 8:             Set $\theta = \theta \times \beta$
 9:         end if
10:     end for
11:     return $\theta$
12: end procedure
```

Algorithm 5 StreamDFP with Online Transfer Learning

```
 1: Initialize $\mathcal{W_S}$ = empty sliding time window
 2: Initialize $\mathcal{W_T}$ = empty sliding time window
 3: Initialize $\mathcal{M_S}$ = prediction model of source disk model
 4: Initialize $\mathcal{M_T}$ = prediction model of target disk model
 5: Initialize $\theta = 1$
 6: Call Lines 3-14 of Algorithm 3 (StreamDFP) to train $\mathcal{M_S}$
    with the samples of source disk model in $\mathcal{W_S}$
 7: for each day $t$ do          ▷ samples of target disk model arrive
 8:     if $\mathcal{W_T}$ is full then
 9:         Slide one day for $\mathcal{W_T}$
10:     end if
11:     Extract learning features to $\mathbf{x}_t$ from all disks
12:     Buffer $(\mathbf{x}_t, y_t)$ into $\mathcal{W_T}$
13:     Call Algorithm 1 (online labeling) to label samples in
        $\mathcal{W_T}$
14:     Call Algorithm 4 (downsampling and training under
        modified online transfer learning) to train $\mathcal{M_S}$ and $\mathcal{M_T}$
        using $\mathcal{W_T}$ and update $\theta$
15:     if $\mathcal{W_T}$ is full then          ▷ our 2nd modification
16:         Set $\hat{y}_t = \frac{\theta}{1+\theta}\mathcal{M_S}(\mathbf{x}_t) + \frac{1}{1+\theta}\mathcal{M_T}(\mathbf{x}_t)$
17:     end if
18: end for
```

Algorithm 5 shows the entire workflow of StreamDFP with online transfer learning. We first initialize the empty sliding time windows (i.e., $\mathcal{W_S}$ for the source disk model and $\mathcal{W_T}$ for the target disk model), $\mathcal{M_S}$, $\mathcal{M_T}$, and θ (Lines 1-5). We next call Lines 3-14 of Algorithm 3 to train $\mathcal{M_S}$ with the source disk model in $\mathcal{W_S}$ (Line 6). For each day t when the samples of the target disk model arrive, if $\mathcal{W_T}$ is full, we slide $\mathcal{W_T}$ by one day (Lines 7-10). We extract learning features to

$\mathbf{x}_t$ from all disks and buffer $(\mathbf{x}_t, y_t)$ into $\mathcal{W}_\mathcal{T}$ (Lines 11-12). We then call Algorithm 1 to label samples in $\mathcal{W}_\mathcal{T}$ (Line 13). We call Algorithm 4 to train $\mathcal{M}_\mathcal{S}$ and $\mathcal{M}_\mathcal{T}$ using $\mathcal{W}_\mathcal{T}$ and update θ (Line 14). Finally, if $\mathcal{W}_\mathcal{T}$ is full, we compute the weighted average of the prediction results of $\mathcal{M}_\mathcal{S}$ and $\mathcal{M}_\mathcal{T}$ with $\mathbf{x}_t$ to output the prediction results $\hat{y}_t$ (Lines 15-17); note that if θ has been discounted due to the incorrect classification of $\mathcal{M}_\mathcal{S}$, then the effect of $\mathcal{M}_\mathcal{S}$ on the prediction results will decrease (i.e., our second modification).

6 Implementation Details

We implement a StreamDFP prototype in two parts. The first part is implemented in Python (with around 830 LoC), in which StreamDFP performs feature extraction, buffering, online labeling, the first phase of downsampling (i.e., selecting a subset of samples from $\mathcal{W}$), and finally writes the processed data into a local file system. The second part is written in Java (with around 1,530 LoC), in which StreamDFP reads the processed data from the local file system and feeds each sample into the second phase of downsampling (i.e., Poisson sampling) and training. We realize all decision-tree-based algorithms and their change detectors in Table 1 using Massive Online Analysis (MOA) [4]. We realize MLP [19] with backpropagation [17] via stochastic gradient descent [34]. We realize online transfer learning based on the mistake-driven homogeneous online transfer learning algorithm (HomOTL-II) [43]. The complete StreamDFP prototype forms a stream processing pipeline.

7 Evaluation

We present trace-driven evaluation results on the prediction accuracy and stream processing performance of StreamDFP on 15 datasets in total. We summarize our findings as follows.

- Enabling concept-drift adaptation in StreamDFP increases the classification accuracy for different incremental learning algorithms, although the highest accuracy among the algorithms varies across datasets. StreamDFP also makes earlier prediction of disk failures. (Exp#1)
- StreamDFP can accurately predict the likelihood of disk failures under regression. (Exp#2)
- Enabling concept-drift adaptation under online transfer learning in StreamDFP increases the classification accuracy for different ensemble learning algorithms. (Exp#3)
- Enabling online transfer learning in StreamDFP increases the classification accuracy of ensemble learning algorithms for various minority disk models. (Exp#4)
- StreamDFP takes within 13.5 seconds per day for processing 37 K disks in D2, making it viable for practical stream processing usage. (Exp#5)
- StreamDFP can effectively predict SSD failures with concept-drift adaptation. (Exp#6)

In the digital supplementary file, we report additional findings, including: (i) online labeling with extra labeled days improves the overall prediction accuracy in most cases, while the number of extra labeled days can be flexibly tuned via StreamDFP (Exp#S1); (ii) StreamDFP maintains the accuracy gains through concept-drift adaptation for various thresholds of false positive rates (Exp#S2); and (iii) StreamDFP supports RNN, which has similar prediction accuracy to MLP (Exp#S3).

7.1 Methodology

We use a total of 15 datasets to validate our evaluation. In addition to the five datasets shown in Table 2, we further select seven datasets in Backblaze and three public SSD datasets from Alibaba [39] (see Table 6). Specifically, the datasets D6, D7, and D8 cover three more recent disk models that are increasingly deployed from year 2019. Also, we use the disk models in D9, D10, D11, and D12 as the target disk models in online transfer learning, and they are also evaluated by prior studies [5], [42]. In particular, we use D2 as the source disk model for D9 and D10, and use D4 as the source disk model for D11 and D12, since D2 and D4 have large populations among the disk models within the same manufacturer. Also, the datasets D13, D14, and D15 cover the SMART datasets from three SSD models (from different manufacturers) with large numbers of failures over a two-year span.

We use all the available SMART attributes (e.g., see Table 3 for D1 to D5) as learning features for each dataset (Section 4.2). For online transfer learning, since D9 does not include the SMART attributes S191, S192, and S193, we remove these SMART attributes as learning features when training $\mathcal{M}_\mathcal{S}$ with D2 for D9. On the other hand, the other target disk models (D10 to D12) have the identical SMART attributes as the source disk models (i.e., D2 and D4).

We configure the durations for evaluation across datasets as follows. For the Backblaze datasets (i.e., D1 to D4 and D6 to D12) and the SSD datasets (i.e., D13 to D15), we find that the SMART attributes are not all available on the first day of the collection period, and the datasets have very different collection durations, ranging from 21 to 81 months. For consistent comparisons, we select the same 460 days of samples from each Backblaze and SSD dataset for evaluation. To ensure that all SMART attributes are available, we set the start date of each dataset as 2014-02-15 for D1, 2015-01-01 for D2, 2018-02-01 for D3, 2014-09-01 for D4, 2019-01-01 for D6 to D8, and 2018-01-03 for D13 to D15. For the Alibaba HDD dataset (i.e., D5), we select all 150 days of samples.

For the evaluation of online transfer learning, we set the start date of the target disk models as 2015-01-01 for D9, 2016-04-15 for D10, and 2015-12-15 for both D11 and D12 to ensure that sufficient samples are collected. Also, we need to set the start dates of the source disk models to be close to those of the target disk models due to concept drift (Section 3). Thus, we set the start date of D2 as 2014-07-01 for D9 and 2015-11-01 for D10, and the start date of D4 as 2014-09-01 for both D11 and D12.

For each dataset, we first warm-up the prediction model from scratch using the first 30 days of samples, same as the length of W (Section 4.3). We then predict disk failures in the next 30 days on a daily basis, and evaluate the accuracy for each day of prediction. Thus, the total durations of our evaluation last for 400 days for D1 to D4, D6 to D12, and D13 to D15, as well as 90 days for D5. For online transfer learning, we set the total durations for training $\mathcal{M}_\mathcal{S}$ for D2 as 100 days and for D4 as 400 days, since the disk count of D2 is nearly 8 $\times$ as many as that of D4 (Table 2).

Metrics. Our evaluation addresses both classification and regression. For classification, we use the following metrics:

Dataset ID	Disk model	Capacity	Disk count	# failures	Period	Duration (months)
D6	Seagate ST8000DM002	8 TB	10,187	383	2016-05-11 to 2020-06-30	50
D7	Seagate ST8000NM0055	8 TB	14,991	514	2016-12-06 to 2020-06-30	43
D8	HGST HMS5C4040BLE640	4 TB	16,346	253	2014-03-27 to 2020-06-30	81
D9	Seagate ST31500541AS	1.5 TB	2,188	384	2013-04-10 to 2018-03-29	60
D10	Seagate ST4000DX000	4 TB	222	79	2013-04-10 to 2017-12-14	57
D11	Hitachi HDS5C3030ALA630	3 TB	4,664	144	2013-04-10 to 2017-09-07	53
D12	Hitachi HDS723030ALA640	3 TB	1,048	72	2013-04-10 to 2017-06-13	51
D13	SSD MA1	480 GB	43,759	1,371	2018-01-01 to 2019-12-31	24
D14	SSD MB1	1920 GB	44,405	1,832	2018-01-01 to 2019-12-31	24
D15	SSD MC1	1920 GB	200,032	10,545	2018-01-01 to 2019-12-31	24

TABLE 6: Overview of ten additional datasets for evaluation (in addition to the five datasets shown in Table 2).

- *Precision:* The fraction of actual failed disks being predicted over all (correctly or falsely) predicted failed disks.
- *Recall:* The fraction of actual failed disks being predicted over all actual failed disks.
- *F1-score:* $\frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$.

For regression, we convert the disk failure likelihood reported by StreamDFP into the number of days of residual lifetime for more intuitive evaluation. We report the *average relative errors of the residual lifetime (ARE)*, defined as the average relative errors of the predicted residual lifetime with respect to the actual residual lifetime over all actual failed disks (we exclude the falsely predicted failed disks for this metric). If the ARE is positive, then it implies that the predicted likelihood of disk failures is larger than the actual likelihood and the residual lifetime is shorter than the actual one, and vice versa.

Default setup. We set the default extra labeled days D_L as 20 days, while we evaluate the impact of D_L in Exp#S1. For fair comparisons, when we evaluate the classification accuracy metrics, we fix the threshold of the average false positive rate (FPR) over the evaluation period for each algorithm; on each day, we compute the FPR as the fraction of falsely predicted failed disks over the total number of healthy disks in the next 30 days. For D1 to D4, D6 to D12, and D13 to D15, we set the default FPR threshold as 1% as in [37], while for D5, we set the default FPR threshold as 0.3%; we evaluate the impact on the accuracy for different FPR thresholds in Exp#S2. For ensemble learning (Section 2.4), we fix 30 decision trees, although including more decision trees (e.g., 100 trees) does not make significant improvements. For other parameters, we choose the default values as in MOA [4] (Section 6). For MLP, we fix one hidden layer with three neurons and set the learning rate as 0.5, although adding more hidden layers (e.g., five hidden layers) does not make significant improvements. In Exp#3 and Exp#4, we set the discounting weight parameter of online transfer learning $\beta = \frac{\sqrt{\#samples}}{\sqrt{\#samples}+\sqrt{ln2}}$ [43], where the number of samples is estimated by the disk count multiplied by the total duration (i.e., 400 days) for a target disk model.

In our experiments, we plot the averaged results over five runs, including the error bars with the minimum and maximum results across all five runs.

7.2 Results

Exp#1 (Effectiveness of concept-drift adaptation). We first consider the classification algorithms in Table 1 (except the regression algorithm FIMT-DD, which is evaluated in Exp#2). We divide the decision-tree-based algorithms into four pairs corresponding to before and after enabling concept-drift adaptation (i.e., HT vs. HAT, Bag vs. BA, Boost vs. BOLE, and RF vs. ARF). For D5, we focus on two pairs of ensemble learning algorithms, i.e., Bag vs. BA and RF vs. ARF, since they can parallelize the training of decision trees for fast execution. We also compare the effectiveness of decision-tree-based algorithms and neural network learning (i.e., MLP).

Figure 4 shows the results of the precision, recall, and F1-score for different datasets. Concept-drift adaptation improves the overall F1-score for each pair of decision-tree-based algorithms in all datasets, by up to 26.8%, 53.2%, 48.8%, 35.5%, 49.9%, 38.4%, 38.5%, and 40.3% for D1, D2, D3, D4, D5, D6, D7, and D8 respectively, while the improvements of precision and recall are up to 27.5-71.8% and 15.7-60.5% across the datasets, respectively. We see that concept-drift adaptation mainly improves the precision, while preserving the recall, in most cases; the only exceptions are the decreasing recall in BA (for D2 and D3) and ARF (for D2), as well as the decreasing precision in BOLE for D3, HAT for D6, and ARF for D8, mainly due to the trade-off between the precision and recall. Also, MLP with backpropagation can achieve the highest F1-scores on D4, D6, D7, and D8 (higher than those of decision-tree-based algorithms by 25.4%, 32.6%, 18.0%, and 31.0%, respectively). We observe that these four datasets have fewer disk failures than D1, D2, D3, and D5, indicating that MLP with backpropagation is effective for training with limited positive samples.

Overall, ARF achieves the highest F1-score for D1 and BA achieves the highest F1-score for D2, D3 and D5, while MLP with backpropagation achieves the highest F1-score for D4, D6, D7, and D8. Thus, it is difficult to identify the "best" algorithm for different datasets. This conforms to the main design goal of StreamDFP that it supports various incremental learning algorithms as a general framework rather than a specific algorithm.

Table 7 further evaluates the average number of days ahead of a disk failure when the prediction is made (i.e., the duration from the day when a disk is predicted as failed to the day when the disk failure occurs). Here, we only consider the failed disks that are correctly predicted. The decision-tree-based algorithms with concept-drift adaptation can predict disk failures earlier than those without concept-drift adaptation, by up to 2.9, 2.0, 1.8, 1.2, 3.9, 2.8, 2.5, and 3.8 days for D1, D2, D3, D4, D5, D6, D7, and D8 respectively. Also, MLP with backpropagation can predict disk failures early, which is overall close to the decision-tree-based algorithms with concept-drift adaptation. The reason is that concept-drift adaptation achieves better characterization of the failure patterns, thereby making earlier prediction.

Exp#2 (Accuracy of regression). We evaluate the accuracy of regression (in terms of ARE) with the regression tree

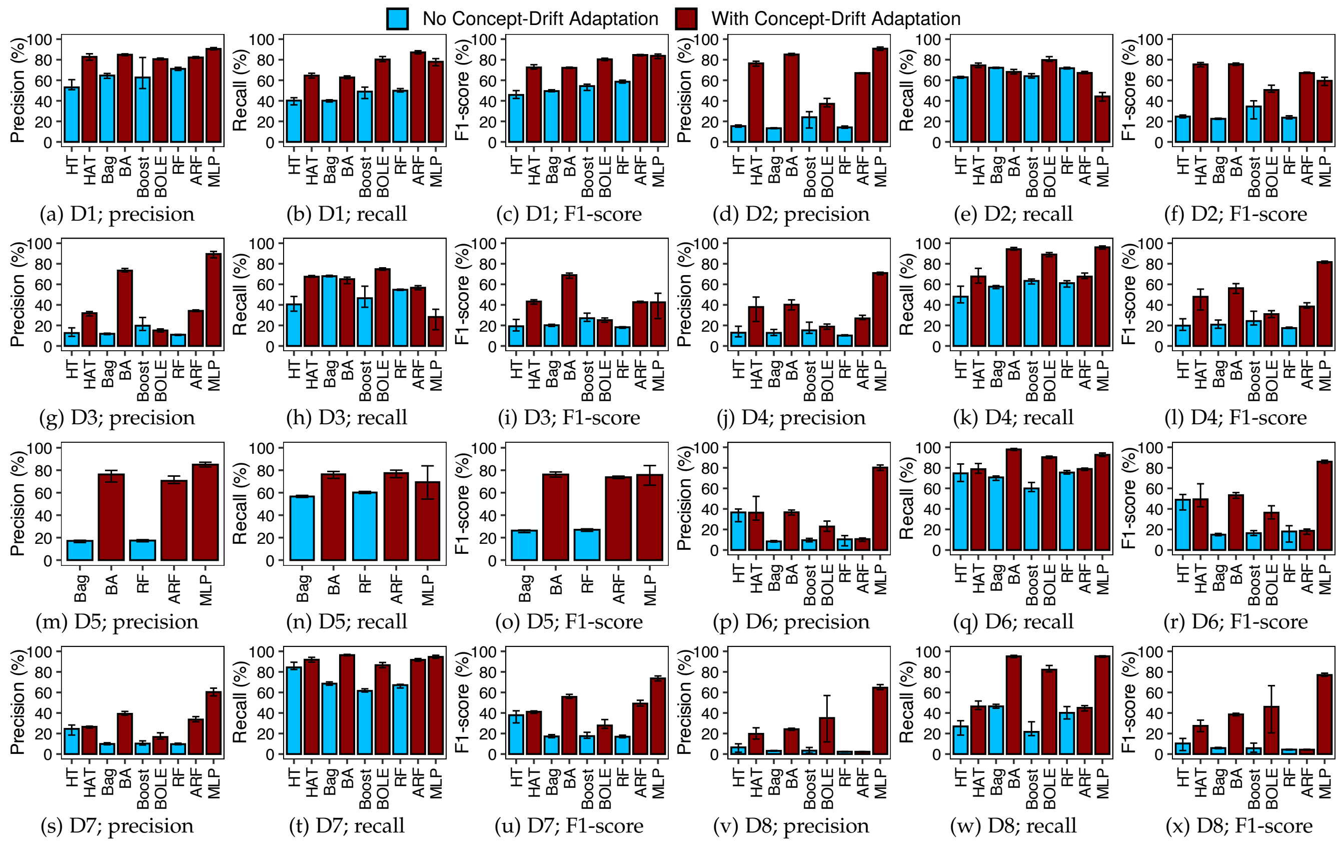


Fig. 4: Exp#1 (Effectiveness of concept-drift adaptation).

Algorithm	D1	D2	D3	D4	D5	D6	D7	D8
HT	11.1	9.1	9.1	11.6	–	12.8	13.6	9.1
HAT	12.3	10.7	10.8	12.0	–	13.2	14.1	12.9
Bag	9.6	9.4	10.9	13.2	7.7	11.5	11.8	11.1
BA	12.2	11.3	11.2	13.9	11.6	14.3	14.4	14.5
Boost	10.5	9.1	9.4	12.3	–	11.3	11.7	12.3
BOLE	13.3	10.6	11.2	13.5	–	13.1	13.3	13.4
RF	11.0	9.7	9.2	12.5	8.0	11.5	11.7	11.1
ARF	13.9	11.3	9.7	12.6	11.5	11.7	13.9	12.0
MLP	12.9	10.2	9.3	13.9	12.0	14.0	14.4	14.6

TABLE 7: Days in advance when a failed disk is predicted for different incremental learning algorithms in Exp#1.

FIMT-DD. Figure 5 shows the ARE results for D1 to D8. Throughout the evaluation period, the means of ARE are -0.0014, -0.27, 0.13, 0.29, 0.58, -0.0055, 0.0048, and -0.52 (in days), while the standard deviations of ARE are 3.0, 2.4, 3.2, 5.6, 2.7, 4.6, 3.2, and 6.3 (in days), for D1, D2, D3, D4, D5, D6, D7, and D8 respectively. The results indicate that the predicted likelihood of disk failures is close to the actual likelihood. Also, the maximum absolute values of ARE for D1, D2, D3, D5, D6, D7, and D8 are smaller than 9, 6, 8, 6, 15, 7, and 15 days, respectively. On the other hand, the ARE for D4 is up to +20 days. The reason is that failure symptoms (e.g., sector errors) before failures for D4 last longer than those for other disk models due to the older average age [1], making the predicted residual lifetime for D4 much shorter than the actual one (i.e., the ARE is a large positive value).

Exp#3 (Effectiveness of concept-drift adaptation under online transfer learning). We consider three pairs of ensemble learning algorithms corresponding to before and after enabling concept-drift adaptation (i.e., Bag vs. BA, Boost vs. BOLE, and RF vs. ARF) under online transfer learning on the four pairs of source and target disk models (i.e., D2 and D9, D2 and D10, D4 and D11, as well as D4 and D12).

Figure 6 shows the accuracy results for the four pairs of disk models. Under online transfer learning, concept-drift adaptation improves the overall F1-score by up to 14.2%, 18.8%, 51.8%, and 30.0% for D9, D10, D11, and D12, respectively, while the improvements of precision and recall are up to 17.1-65.8% and 9.0-36.0% across the four datasets, respectively. We observe that concept-drift adaptation still mainly improves the precision, while preserving the recall, under online transfer learning in most cases. The only exceptions are the decreasing recall in ARF (for D9 and D10), BA (for D9), and BOLE (for D11), mainly due to the trade-off between the precision and recall.

Exp#4 (Effectiveness of online transfer learning). We compare the prediction accuracy with and without online transfer learning for the four target disk models (D9 to D12). With online transfer learning, the source disk models are still D2 for D9 and D10 as well as D4 for D11 and D12; without online transfer learning, we apply StreamDFP directly to the target disk models. We consider the three ensemble algorithms with concept-drift adaptation (i.e., BA, BOLE, and ARF).

Figure 7 shows the accuracy results. Online transfer learning improves the overall F1-score by up to 7.9%, 3.1%, 6.7%, and 5.3% for D9, D10, D11, and D12, respectively, while the improvements of precision and recall are up to 4.3-8.4% and 1.0-9.8% across the datasets, respectively.

Exp#5 (Execution performance of StreamDFP). We evaluate the stream processing performance of StreamDFP. We focus on D2, which has the largest disk population (with around 37 K disks) among all Backblaze datasets we consider. Note that for D5, since it is executed on a production server

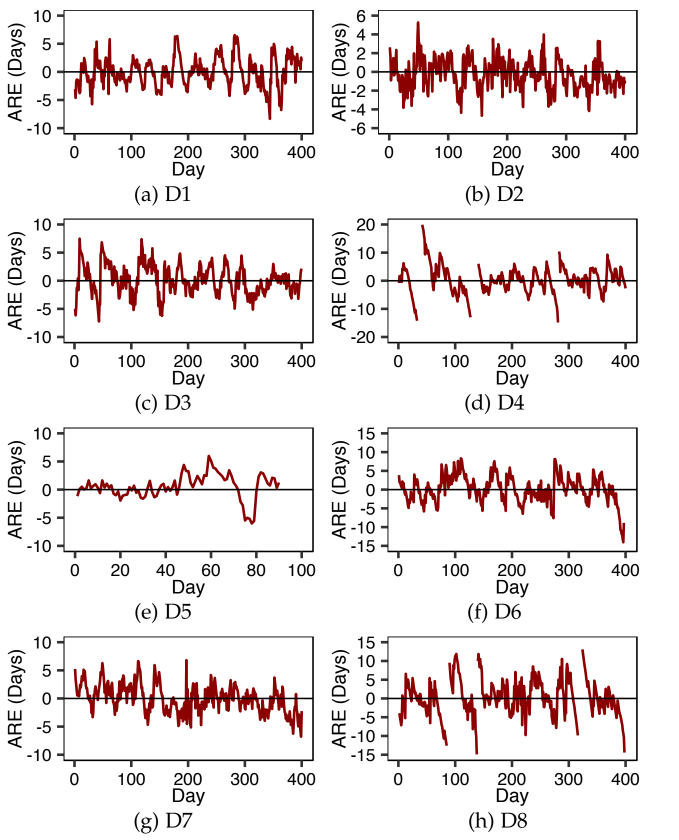


Fig. 5: Exp#2 (Accuracy of regression). Note that the gaps in D4 and D8 mean that no failed disks are observed in those periods.

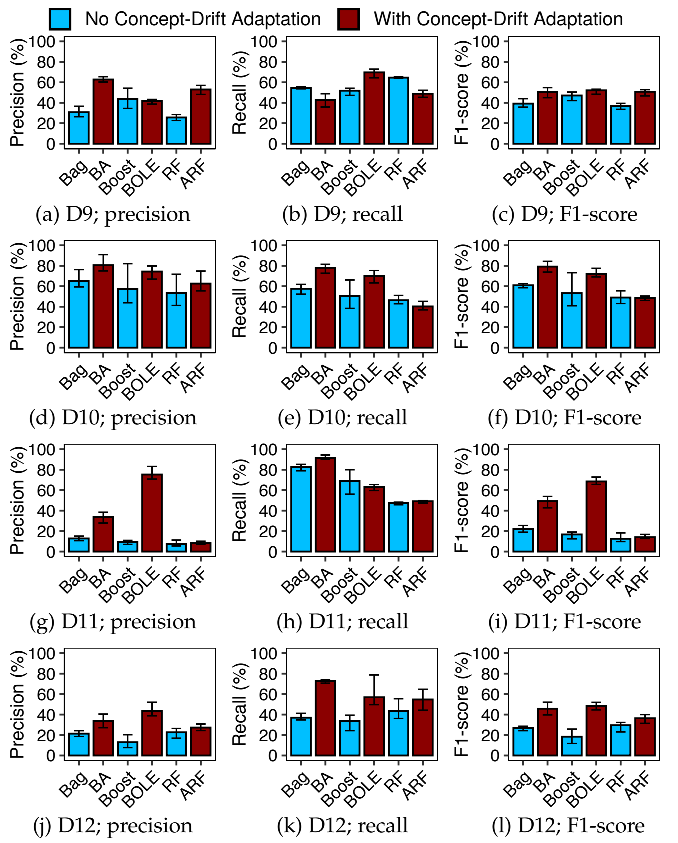


Fig. 6: Exp#3 (Effectiveness of concept-drift adaptation under online transfer learning).

at Alibaba, we cannot reveal the hardware setting of the server for our performance evaluation of StreamDFP.

We measure the execution time of StreamDFP on a local server equipped with a quad-core 3.4 GHz Intel Core i5-7500, 32 GiB RAM, and a Toshiba DT01ACA100 7200 RPM 1 TiB SATA hard disk. We run StreamDFP to predict disk failures in 400 days using the BA algorithm (which achieves the highest F1-score), and report the average execution time of one day. We use single-thread execution for feature extraction, buffering, online labeling, and prediction for each arriving sample; for training, we enable multi-threading (with 4 threads) for BA in MOA [4] to parallelize training across all tree learners.

Table 8 shows a breakdown of the per-day execution time of StreamDFP for D2, while the standard deviations of all five runs are in brackets. The most time-consuming step is training, while prediction is fast. This is expected, as training has complicated computation in growing multiple trees with the samples in recent days. Nevertheless, the execution time for training remains acceptable in practice. Overall, StreamDFP performs training and prediction within 13.5 seconds on the daily SMART data of 37 K disks. We believe that StreamDFP meets the performance need in large-scale disk deployment.

Exp#6 (Effectiveness of StreamDFP on SSD failure prediction). We apply StreamDFP to SSD failure prediction and evaluate its generality. We consider three pairs of ensemble learning algorithms corresponding to before and after concept-drift adaptation (i.e., Bag vs. BA, Boost vs. BOLE, RF vs. ARF) and MLP on D13, D14, and D15. Figure 8 shows that concept-drift adaptation of decision-tree-based algorithms improves the overall F1-score by up to 29.3%, 38.7%, and 51.1% for D13, D14, and D15, respectively, which conforms to the conclusions in Exp#1. Also, BA achieves the highest F1-score for D14 (65.7%) and D15 (68.3%), while MLP achieves the highest F1-score for D13 (71.9%). Thus, StreamDFP remains effective in SSD failure prediction.

8 Related Work

There have been extensive studies on disk failure prediction. Traditional prediction approaches are based on statistical techniques, such as Bayesian classifiers [15], hypothesis tests [20], support vector machines [30], Markov models [44], and rule-based methods [26]. Recent studies improve the prediction accuracy via machine learning algorithms, such as backpropagation neural networks [45], decision trees [23], gradient boost regression trees [24], random forests [27], and unsupervised learning (e.g., the nearest neighbor algorithm [13]). Also, some studies explore deep neural networks to predict disk failures, such as recurrent neural networks [38], layerwise perturbation-based adversarial training [41], and convolutional neural networks [25], [35]. SMARTer [25] leverages workloads and locations of disks in addition to SMART attributes to predict disk failures. Some studies show how disk failure prediction facilitates disk replacements [5] and scrubbing [27], as well as improves cloud service availability [40]. All the above studies are based on offline prediction and assume that all training data is available in advance.

The closest related work to ours is [37], which applies Online Random Forests (ORF) to disk failure classification

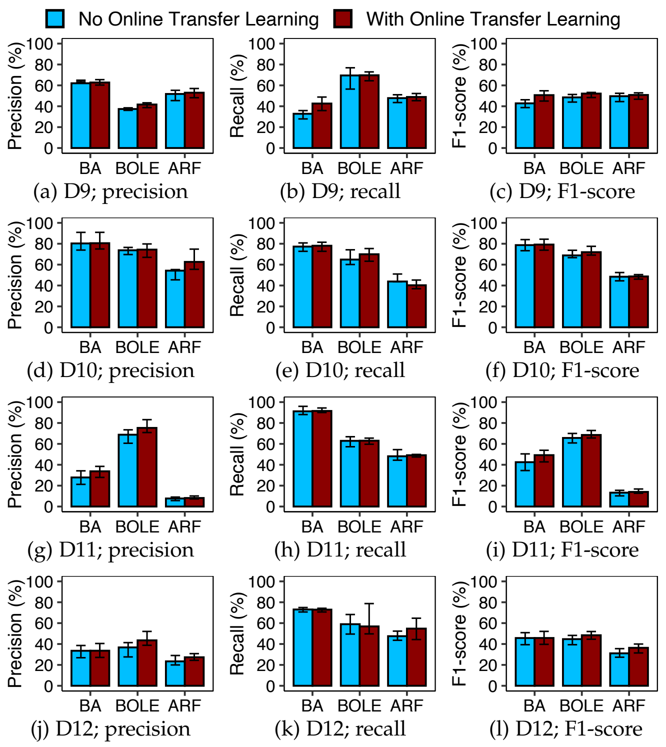


Fig. 7: Exp#4 (Effectiveness of online transfer learning).

Step	Feature extraction	Buffer-ing	Online labeling	Down-sampling	Train-ing	Predic-tion
Time	0.0055 s (0.069 ms)	0.093 s (9.6 ms)	0.42 s (30.8 ms)	0.52 s (19.0 ms)	10.6 s (0.75 s)	1.9 s (0.14 s)

TABLE 8: Exp#5 (Execution performance of StreamDFP). We show the average per-day execution time of StreamDFP for D2; numbers in brackets are standard deviations of all five runs.

and automatically updates labels based on incoming SMART attributes. However, StreamDFP considers an inherently different perspective from ORF-based classification [37]: while ORF-based prediction focuses on online learning and tackles the aging issue in the prediction model, StreamDFP focuses on stream mining and adapts the prediction model to concept drift in data streams. StreamDFP addresses the following issues that are not considered in [37]: (i) providing a general framework that supports various stream mining algorithms (instead of ORF only) and customizes them with concept-drift adaptation; (ii) considering both classification and regression (instead of classification only); and (iii) validating its correctness in both HDD and SSD datasets at Alibaba.

9 Conclusion

We present StreamDFP, a general stream mining framework for disk failure prediction with concept-drift adaptation. StreamDFP is motivated by the existence of concept drift, backed by our measurement study on the SMART datasets from Backblaze and Alibaba. It also supports a variety of incremental learning algorithms. Our evaluation on decision-tree-based and neural network learning algorithms shows that concept-drift adaptation improves the prediction accuracy significantly under various settings and online transfer learning for minority disk models. StreamDFP also achieves high stream processing performance.

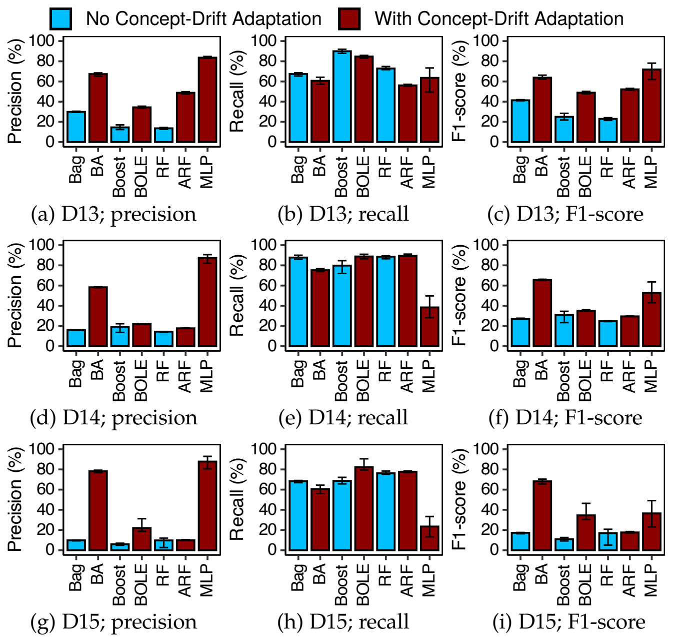


Fig. 8: Exp#6 (Effectiveness of StreamDFP on SSD failure prediction).

Acknowledgements

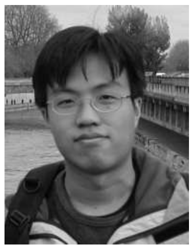
This work was supported in part by Alibaba Group via the Alibaba Innovation Research (AIR) program, National Key R&D Program of China (2021YFF0704001), and Natural Science Foundation of China (No. 62072381).

References

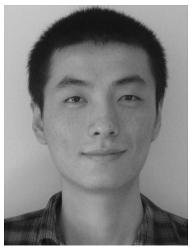
[1] Backblaze. https://www.backblaze.com/b2/hard-drive-test-data.html.
[2] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *Proc. of SIAM SDM*, 2007.
[3] A. Bifet and R. Gavaldà. Adaptive learning from evolving data streams. In *Proc. of Springer IDA*, 2009.
[4] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.
[5] M. M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann. Predicting disk replacement towards reliable data centers. In *Proc. of ACM SIGKDD*, 2016.
[6] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
[7] L. Breiman. Random Forests. *Machine learning*, 45(1):5–32, 2001.
[8] R. S. M. de Barros, S. G. T. de Carvalho Santos, and P. M. G. Júnior. A boosting-like online learning ensemble. In *Proc. of IEEE IJCNN*, 2016.
[9] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. of ACM SIGKDD*, 2000.
[10] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *Proc. of ACM ICML*, volume 96, pages 148–156, 1996.
[11] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, pages 286–295, 2004.
[12] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44, 2014.
[13] X. Gao, S. Zha, X. Li, B. Yan, X. Jing, J. Li, and J. Xu. Incremental prediction model of disk failures based on the density metric of edge samples. *IEEE Access*, 7:114285–114296, 2019.
[14] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495, 2017.
[15] G. Hamerly, C. Elkan, et al. Bayesian approaches to failure prediction for disk drives. In *Proc. of ACM ICML*, 2001.

[16] S. Han, P. P. C. Lee, Z. Shen, C. He, Y. Liu, and T. Huang. Toward adaptive disk failure prediction via stream mining. In *Proc. of IEEE ICDCS*, 2020.
[17] G. Hinton, D. Rumelhart, and R. Williams. Learning internal representations by error propagation. *Parallel distributed processing*, 1:318–362, 1986.
[18] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
[19] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2:359–366, 1989.
[20] G. F. Hughes, J. F. Murray, K. Kreutz-Delgado, and C. Elkan. Improved disk-drive failure warnings. *IEEE Trans. on Reliability*, 51(3):350–357, 2002.
[21] E. Ikonomovska, J. Gama, and S. Džeroski. Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, 23(1):128–168, 2011.
[22] S. Kadekodi, K. Rashmi, and G. R. Ganger. Cluster storage systems gotta have HeART: improving storage efficiency by exploiting disk-reliability heterogeneity. In *Proc. of USENIX FAST*, 2019.
[23] 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 *Proc. of IEEE/IFIP DSN*, 2014.
[24] J. Li, R. J. Stones, G. Wang, Z. Li, X. Liu, and K. Xiao. Being accurate is not enough: New metrics for disk failure prediction. In *Proc. of IEEE SRDS*, 2016.
[25] S. Lu, B. Luo, T. Patel, Y. Yao, D. Tiwari, and W. Shi. Making disk failure predictions SMARTer! In *Proc. of USENIX FAST*, 2020.
[26] A. Ma, R. Traylor, F. Douglis, M. Chamness, G. Lu, D. Sawyer, S. Chandra, and W. Hsu. RAIDShield: Characterizing, monitoring, and proactively protecting against disk failures. *ACM Trans. on Storage*, 11(4):17, 2015.
[27] F. Mahdisoltani, I. Stefanovici, and B. Schroeder. Proactive error prediction to improve storage system reliability. In *Proc. of USENIX ATC*, 2017.
[28] F. J. Massey. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.
[29] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi. Test of Page-Hinckley, an approach for fault detection in an agro-alimentary production system. In *IEEE Asian Control Conference*, 2004.
[30] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research*, 6(May):783–816, 2005.
[31] N. C. Oza. Online bagging and boosting. In *Proc. of IEEE SMC*, 2005.
[32] E. S. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.
[33] B. Pfahringer, G. Holmes, and R. Kirkby. New options for hoeffding trees. In *Proc. of Springer AI*, 2007.
[34] D. Saad. Online algorithms and stochastic approximations. *Online Learning*, 5:6–3, 1998.
[35] 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 *Proc. ACM/IEEE DAC*, 2019.
[36] I. V. Tetko, D. J. Livingstone, and A. I. Luik. Neural network studies. 1. comparison of overfitting and overtraining. *Journal of Chemical Information and Computer Sciences*, 35:826–833, 1995.
[37] J. Xiao, Z. Xiong, S. Wu, Y. Yi, H. Jin, and K. Hu. Disk failure prediction in data centers via online learning. In *Proc. of ACM ICPP*, 2018.
[38] C. Xu, G. Wang, X. Liu, D. Guo, and T.-Y. Liu. Health status assessment and failure prediction for hard drives with recurrent neural networks. *IEEE Trans. on Computers*, 65:3502–3508, 2016.
[39] F. Xu, S. Han, P. P. Lee, Y. Liu, C. He, and J. Liu. General feature selection for failure prediction in large-scale SSD deployment. In *Proc. of IEEE/IFIP DSN*, 2021.
[40] Y. Xu, K. Sui, R. Yao, H. Zhang, Q. Lin, Y. Dang, P. Li, K. Jiang, W. Zhang, J.-G. Lou, et al. Improving service availability of cloud systems by predicting disk error. In *Proc. of USENIX ATC*, 2018.
[41] J. Zhang, J. Wang, L. He, Z. Li, and S. Y. Philip. Layerwise perturbation-based adversarial training for hard drive health degree prediction. In *Proc. of IEEE ICDM*, 2018.
[42] J. Zhang, K. Zhou, P. Huang, X. He, Z. Xiao, B. Cheng, Y. Ji, and Y. Wang. Transfer learning based failure prediction for minority disks in large data centers of heterogeneous disk systems. In *Proc. of ACM ICPP*, 2019.
[43] P. Zhao, S. C. Hoi, J. Wang, and B. Li. Online transfer learning. *Artificial Intelligence*, 216:76–102, 2014.
[44] Y. Zhao, X. Liu, S. Gan, and W. Zheng. Predicting disk failures with HMM-and HSMM-based approaches. In *Proc. of IEEE ICDM*, 2010.
[45] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma. Proactive drive failure prediction for large scale storage systems. In *Proc. of IEEE MSST*, 2013.

Shujie Han received the B.Eng. degree from Northwestern Polytechnical University in 2017 and the Ph.D. degree in Computer Science and Engineering from the Chinese University of Hong Kong in 2021. She is currently a postdoc at Peking University. Her research interests include stream mining, storage reliability, and data deduplication.



Patrick P. C. Lee received the B.Eng. and the M.Phil. degrees from the Chinese University of Hong Kong in 2001 and 2003, respectively, and the Ph.D. degree in Computer Science from Columbia University in 2008. He is now a Professor of the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests are in storage systems, distributed systems, and networks.



Zhirong Shen received the B.S. degree from University of Electronic Science and Technology of China in 2010, and the Ph.D. degree in Computer Science from Tsinghua University in 2016. He is now an associate professor at Xiamen University. His current research interests include storage reliability and storage security. He is a member of the IEEE.

Cheng He received the BS and MS degrees in communication and information systems from University of Electronic Science and Technology, Chengdu, in 2003 and 2006, respectively. He is now a senior algorithm expert at Alibaba. His research interests include big data stream mining and online learning for intelligent management and control of data centers.

Yi Liu received the MS degree from Zhejiang University in 2016. He is now a senior algorithm engineer at Alibaba.

Tao Huang received the MS degree from Shanghai Jiao Tong University in 2018. He is now an algorithm engineer at Alibaba.