1 INTRODUCTION
NAND flash-based solid state drives (SSDs) represent an important storage tier in data centers holding most of today’s warm and hot data. Although SSDs are less cost-efficient than hard drives (HDDs), they provide several advantages over disks including orders of magnitude higher bandwidth (a million I/O operations per second), lower average read latency (less than 100μs), and lower energy consumption. As SSDs are built from semiconductors lacking mechanical components such as spinning disks, they are also more reliable and less prone to failures compared to HDDs. The number of SSDs shipped each year has increased steadily by 42.5% over the last decade [1], now exceeding exabytes of storage capacity every year. SSD manufacturers have employed three main techniques to increase the storage density over the past years including planar scaling, 3D integration, and multi-level cells. While beneficial for the storage density, these mechanisms have reduced the endurance, retention, and reliability of SSDs [52], [13], [76], requiring increasingly sophisticated encoding and fault tolerance mechanisms. Nevertheless, even with advanced fault tolerance techniques and low failure rates, large Hyperscale data centers utilizing 100,000’s of SSDs suffer from multiple device failures daily. Data center operators are interested in predicting SSD device failures for two main reasons. First, even with RAID [6] and replication [25] techniques in place, device failures induce transient recovery and repair overheads affecting the cost and tail latency of storage systems. Second, predicting near-term failure trends helps to inform the device acquisition process enabling to save costs and avoid capacity bottlenecks. As a result, it is important to predict both the short-term individual device failures as well as near-term failure trends.

Prior studies on predicting storage device failures [62], [53], [10], [75] focused primarily on traditional hard disks, however, due to the fundamentally different architecture of SSDs, prior techniques and findings are not readily applicable to SSDs. Research that has particularly focused on SSDs [8], [14], [19], [37] generally concentrated on understanding specific errors and issues within SSDs, limited to a controlled laboratory environment. Most studies that analyzed SSDs in the field focused on understanding correlations among specific workloads, their induced number of writes and bit errors, as well as their effect on the reliability of SSDs [21] [68], [23]. Alter [5] and Schroeder [69] analyzed authentic SSD logs collected in the Google cloud to leverage machine learning (ML) techniques for predicting the likelihood of SSD failures. While most related to our work, their proposed
models either fall short on determining failed drives, or produce a large number of false positives, thereby lowering the performance of the prediction models. In particular, these two prior works suffer from the following main challenges. First, as they utilize black-box ML techniques, they are unaware of the underlying failure reasons rendering it difficult to determine the failure types that these models can predict. Second, the models in prior work struggle with dynamic environments that suffer from previously unseen failures that have not been included in the training set. These two challenges are especially relevant for the SSD failure detection problem which suffers from a high class imbalance. In particular, the number of healthy drive observations is generally orders of magnitude larger than the number of failed drive observations, thus posing a problem for many ML models.

To address these challenges, we propose to utilize 1-class ML models that are trained only on the majority class. By ignoring the minority class for training, our 1-class models avoid overfitting to an incomplete set of failure types, thereby improving the overall prediction performance by up to 9.5% in terms of ROC AUC score. The benefit of our proposed technique becomes even more evident when we reduce the types of failures included in the training set of the baselines approaches, showing 13% to 33% improvements using our proposed 1-class approaches over prior work. Furthermore, we introduce a new learning technique for SSD failure detection, 1-class autoencoder, which enables interpretability of the trained models while providing high prediction accuracy. In particular, 1-class autoencoders provide insights into what features and their combinations are most relevant to flagging a particular type of device failure. This enables categorization of failed drives based on their failure type, thus informing about specific procedures (e.g., repair, swap, etc.) that need be applied to resolve the failure.

For analysis and evaluation of our proposed techniques, we leverage a cloud-scale dataset from Google that has already been used in prior work [5, 70]. This dataset contains 40 million observations from over 30,000 drives over a period of six years. For each observation, the dataset contains 21 different SSD telemetry parameters including SMART (Self-Monitoring, Analysis and Reporting Technology) parameters, the amount of read and written data, error codes, as well as the information about blocks that became non-operational over time. We determine the best performing ML approaches for predicting SSD failures and then explore optimization techniques, including feature selection and data normalization, to address the challenges of large feature spaces and highly imbalanced datasets. With these optimizations in place, our best approach outperforms all prior approaches by at least 9.5% ROC AUC score.

2 BACKGROUND

This section provides a brief introduction to flash device technology and its implication on the reliability of SSDs. We show that predicting device failures is a challenging, multi-dimensional data problem that requires development of sophisticated machine learning techniques.

Contemporary SSDs are semiconductor devices that persistently store the data in NAND flash arrays consisting of floating gate transistors [38]. The charge within the floating gate can only be altered by applying a high voltage (20V), forcing the electrons to tunnel [28] through a highly resistive isolation material, thereby slowing degrading the storage capability and causing transistor wear-out. Furthermore, modern NAND drives leverage different voltage levels for storing multiple bits of information in a single transistor. Sensing the correct cell value on a read becomes increasingly difficult after writing the cell frequently, gracefully degrading the ability to read out data successfully. Unfortunately, due to manufacturing and device variability, the number of writes required to trigger a device failure is difficult to predict. As manufacturers want to increase their yields, many sold SSDs already contain a number of non-functioning bad blocks. Hence, the number of healthy blocks even differs among new devices, thus affecting their overall lifetime. The garbage collection and wear-levelling procedures performed by SSDs, internally, further impede the ability of detecting device failures. These tasks introduce extra writes, transparent to the user, which are highly application-specific. In
particular, the write patterns (random vs. sequential) and read-write ratios affect the lifetime of a device considerably. In summary, the reasons leading to a particular device failure are highly application and device specific, rendering failure prediction by using a simple universal technique impossible. This is further motivated by Figure 1 showing a principle component analysis (PCA) for the 21 device-internal SSD features utilized in this study. As can be seen in this figure, several of the failed drives (outliers) cannot be easily separated from the the healthy drives (inliers), motivating the need for more sophisticated machine learning techniques.

3 1-CLASS FAILURE PREDICTION

Prior work on SSD failure prediction suffers from three shortcomings: (i) the limited overall accuracy of predicting failures, (ii) the inability of reliably predicting previously unseen failure types, and (iii) the lack of interpretability of predictions. To address these challenges, we provide the following contributions. First, we provide a comprehensive analysis of machine learning techniques to predict SSD failures with the highest recall and accuracy for both the majority and minority classes. We optimize our approaches by addressing the challenges of imbalanced data sets and feature explosion. Second, we show how 1-class predictive models can be used to predict previously unseen failures in a dynamic data center environment. Third, we propose 1-class autoencoder, an approach to interpret the predictions of our model, to enable understanding of the most important reasons for failures.

3.1 Accurate Prediction of SSD Failures

Our dataset contains 40 million observations from over 30,000 drives from Google data centers covering a time span of six years where each SSD observation contains the values of 21 distinct features. These features include SMART data, the amount of data read and written from the device, the emitted error codes, as well as the information about grown bad blocks which became non-operational over time due to wear-out. Predicting device failures from this data poses two challenging problems. First, due to the large feature space, machine learning models suffer from the curse of dimensionality, as the training and inference times of machine learning algorithms grow, often exponentially, with the number of features. Secondly, the data from which our models need to infer failures, suffers from a significant class imbalance problem, as there exists a significantly greater number of healthy drive observations than failed drive observations.

3.1.1 Feature Selection. To address the curse of dimensionality introduced by large number of feature, we developed a feature selection mechanism [22, 39, 40, 43] for improving SSD failure prediction. The goal is to select the smallest number of distinguishing features from this dataset to enable the highest accuracy and recall for detecting SSD failures. In contrast to prior work on finding anomalous behavior in cloud systems [79], we performed an extensive study of eight different filtering mechanisms to rank different features in the order of their importance for failure prediction. We observed that, except for the top 9 features selected, the order of the importance of features computed by the different feature selection algorithms varied substantially and, in fact, utilizing any one feature selection mechanism individually can lead to high variation in model performance. To address this challenge, we developed an approach to effectively combine the rankings of different feature selection algorithms, subsequently leading to the best model performance both in terms of training time and accuracy, as shown in Section 5.

3.1.2 Class Imbalance. A major challenge in anomaly detection is to deal with the inherent class imbalance problem. Among the over 30,000 drives that we examined, about 4,000 SSDs failed at some point in time, however, for the most of its lifetime, every SSD behaves like a healthy drive. This resulted in a training dataset containing over 40 million data points for healthy drives (majority class) while only 15,000 data points for failed drives (minority class). Distinguishing between healthy and failed drive observations is further aggravated by the fact that some of the drives were put back into service after repair and then failed again, requiring to be treated as separate failure observations.

Since the size of the majority class is three orders of magnitude larger than the size of the minority class, recognizing instances of the minority class during classification is challenging, since many of the ML algorithms are designed to be biased toward the majority class. The data points at which the minority instances are positioned among the majority instances in an imbalanced scheme contributes to the increase in misclassification rate, thus commonly referred to as data difficult factors [18]. These factors include, but are not limited to, small disjuncts, class overlap, borderline, noise, outliers, and rare instances [27].

Prior research [5], [79], [32], [54] used techniques including Random Forest [5, 46], Neural Networks [34], k-Nearest Neighbours (k-NN) [47], and 2-Class Support Vector Machines (SVM) [9] for predicting storage device failures. We observe that these approaches cannot cope well with the high class-imbalance and overfit to the failure types contained in the training dataset. In Section 5 we show that our proposed 1-class predictive models outperform prior works by up to 9.5% and reduce training times by up to 1.8×. We also evaluate our proposed feature selection techniques and perform a sensitivity study on how far ahead the models can predict the failures.
3.2 Predicting unseen failures

As outlined in Section 2, flash devices suffer from a variety of different failures induced by write amplification, grown bad blocks, controller errors, and backup battery issues. As some of the failures are workload-dependent and SSD technologies change, that is, the move from TLC to QLC cells, it is difficult to collect data about every failure type. Hence, it is unlikely that any training dataset would cover all types of device failures that may occur in the future.

We observed that previous approaches to detect SSD failures generally fail to predict unseen failure types that have not been experienced by the model during training. In this work, we propose to improve the adaptivity of the predictive models by training them only on the majority class instances. By utilizing only healthy drives as training data, the models can learn a strong representation of healthy drives, without overfitting to a limited set of known or previously seen failure types.

We introduce two mechanisms to enable this approach including 1-class isolation forest and 1-class autoencoders [73]. The generic isolation forest [50] is a popular algorithm for performing anomaly detection based on Random Forest. The algorithm leverages the fact that anomalous data points generally satisfy fewer conditions than normal data points. Hence, an anomaly score can be computed by counting whether the number of conditions required to separate a given data point is below a certain threshold. We also explored different contamination factors (the fraction of anomalous data points) to inform the model about this additional information. Utilizing these optimizations, we show in Section 5 that anomalous drives can be determined with a high recall of 0.99, even though the model had never seen a failed drive during training.

Furthermore, to the best of our knowledge, this is the first work to use 1-class autoencoders for predicting SSD failures. We designed an 1-class autoencoder based model that generates a compressed knowledge representation of the original input of healthy drive observations as well as a trained decoder which, in return, tries to generate healthy drive observations from the compressed representation. We remove all failed drive observations from the dataset for training the autoencoder model, in order to enable the model to learn a compressed representation of what a healthy SSD should look like. Reconstruction error [66] is used to interpret the decisions emitted by this model. Figure 2 shows the internal design of autoencoders. We first encode and then decode a particular sample of SSD using the autoencoder model. If the input and output are similar, the input likely corresponds to a healthy drive, whereas, if the input and output suffer from a large reconstruction error, then the sample is flagged as an anomaly (failed drive).

![Figure 2: Autoencoder Design](image)

As we show in Section 5, training on only the healthy drives provides the following benefits. First, the training is not limited by learning from a few samples in the minority classes. Second, the training examples from healthy drives are easier and cheaper to record, which improves the scalability of our approach. Third, ignoring the minority class during training improves the ability of the model to predict previously unseen failures.

3.3 Interpreting SSD Failures

Understanding the reasons for an SSD drive failure is of primary concern for manufacturers and data center operators to improve the reliability and to inform about the required maintenance and repair procedures. This enables them to choose appropriate drives for a particular workload, providing the best reliability as well as enabling fast re-servicing of drives. Providing an understanding of SSD failures also helps with increasing the transparency of our predictions and avoids running full diagnostic tests to determine the causes of a failure.

We leverage our neural network based 1-class autoencoder approach to enable this capability by creating a compressed lower-dimensional representation of healthy drive observations as explained in the previous section. We then use this representation to select anomalous observations that do not conform to the representation, thereby generating an output that differs significantly from representation of healthy observations. The observations that produce a reconstruction error greater than a chosen threshold are flagged as failures. We then categorize these generated outputs by separating them into buckets, each one representing the error while reconstructing the input for each feature. The features that produce a larger than average error for a particular drive are then marked as significant and reported. We show in Section 5 how interpreting this data provides insights into why the model predicted a particular device as a failed drive.

4 METHODOLOGY

Our dataset contains SSD Telemetry data from over 30,000 drives over a period of 6 years collected from Google datacenters. In total the dataset contains 40 million observations with 21 different telemetry parameters. Around 4,000 drives failed during this period leading to 15,000 observations classified as failed from the total of 40 million observations.
was still functional. We leverage this data in order to find out while the rest failed up to four times. For our work, we label each failure as a separate case. Drive replacement times, upon failure varied widely ranging from under a week (80% of the cases), to over three months (10% of the cases).

4.2 Feature Selection

One of our primary goals was to select the most distinguishing features that are highly correlated to the failures for training. We used three different feature selection methods, Filter [67], Embedded [45], and Wrapper [71] techniques, and implemented eight different algorithms including Pearson ranking [60], Spearman ranking [77], Chi square test [56], Analysis of Variance (ANOVA) [35], Recursive Feature Elimination [29], Extra Trees [51], Lasso Regularization [81], Elastic Net [82], and Ridge Regression [30], for selecting the most important features contributing to failures for our dataset.

4.2.1 Filter Methods. Filter methods for feature selection use statistical measures to provide scores for each feature. The features were then ranked by this score and only the top significantly correlated features were selected. Specifically, we used Pearson correlation, Spearman correlation, Elastic Net, and Kendall Tau ranking algorithms to rank the features.

4.2.2 Wrapper Methods. Wrapper methods select different combinations of features and then evaluate them to pick the most relevant features. A prediction model is typically used to evaluate the combinations and assign scores based on model accuracy. We used different search processes including Random Forest, Recursive Feature Elimination with Extra Trees classifiers and Logistic Regression to select the top features.

4.2.3 Embedded Methods. Embedded methods pick the most relevant features that contribute to the accuracy of the model during the creation and training of the model. LASSO (L1), Elastic Net, and Ridge Regression (L2) are the most commonly used regularization methods. These methods optimize the learning procedure by training models with lower complexity, where features with non-zero coefficients are selected for training the model, thus serving as methods for feature selection. The three methods above provide feature rankings which were then merged into a single list, giving equal importance to each method. As we show in Section 5, the elaborate feature selection process improves both the training time and the prediction accuracy significantly over the baseline that utilizes all 21 features. The resulting set of top features is shown in Table 2. We validated the feature selected with domain experts, who confirmed that there is a strong correlation between the features that were picked by the feature selection algorithms and actual parameters which indicate wear out and failures in SSDs.

4.1 Data Preprocessing

The dataset contained information on four different SSD models (MLC A, B, C and D) and contained no information on specific vendors. Our feature selection process (see Section 5.4.2) did not select the model as a significant feature and hence we excluded it during training process. Of the drives that failed, approximately 90% of the drives failed only once while the rest failed up to four times. For our work, we label each failure as a separate case. Drive replacement times, upon failure varied widely ranging from under a week (80% of the cases), to over three months (10% of the cases).

Around 30% of the drives that failed during the data collection process were replaced while the rest were removed, and hence no longer appeared in the dataset. As a result, we obtained approximately 300 observations for each healthy drive and 4 to 140 observations for each failed drive. The entire list of metrics of features present in the dataset is shown in Table 1.

Traditionally, the drive replacement policy at cloud service providers uses a rule-based approach [31]. Whenever certain parameters such as UECC error count, reserve block count, etc., reach a certain value, the drive is replaced. However, this approach suffers from two shortcomings. First, these rules do not comprehensively predict all the failures, and hence the drives fail unexpectedly in certain cases, resulting in data loss and application crashes. Second, these rule sets have also been shown to be overly conservative, leading to many cases where drives are replaced even though they were still operating normally. The aggregate number of drive failures per week is also beneficial for cloud providers as they can order replacements in advance. These issues motivated us to develop a more flexible and accurate approach based on machine learning techniques.

The data collected contained features in string, date time, and integer format. We ensured that all the data collected was transformed into numeric format so that it can be processed by the machine learning models. String values, such as Drive model name, were converted into categorical features, and date and time were converted into UNIX timestamps. We treated each data point as an independent observation and normalized all the non-categorical data values to be between 0 and 1. We created separate datasets, identified by the parameter $N$, by selecting daily observations before a predicted failure occurred. For instance, $N = 3$ contains all observations for each drive 3 days before the drive either failed or was still functional. We leverage this data in order to find out how far ahead our proposed models can predict the failures.
<table>
<thead>
<tr>
<th>Features</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>drive id</td>
<td>string</td>
<td>Unique ID assigned to each drive</td>
</tr>
<tr>
<td>model</td>
<td>string</td>
<td>Drive model type</td>
</tr>
<tr>
<td>timestamp</td>
<td>int</td>
<td>Time (in us) since the drive was first put in use</td>
</tr>
<tr>
<td>read count</td>
<td>int</td>
<td>Number of read operations in the drive’s lifetime</td>
</tr>
<tr>
<td>write count</td>
<td>int</td>
<td>Number of write operations in the drive’s lifetime</td>
</tr>
<tr>
<td>erase count</td>
<td>int</td>
<td>Number of erase operations in the drive’s lifetime</td>
</tr>
<tr>
<td>status read only</td>
<td>boolean</td>
<td>Status flag indicating if the drive is operating in read only mode</td>
</tr>
<tr>
<td>cumulative p/e cycle</td>
<td>int</td>
<td>Number of times a memory cell is erased and reprogrammed</td>
</tr>
<tr>
<td>factory bad block count</td>
<td>int</td>
<td>Number of non-operational data blocks upon drive purchase</td>
</tr>
<tr>
<td>cumulative bad block count</td>
<td>int</td>
<td>Number of blocks which became non-operational during the drive’s lifetime</td>
</tr>
<tr>
<td>status dead</td>
<td>boolean</td>
<td>Status flag indicating if the drive is currently failed</td>
</tr>
<tr>
<td>correctable error count</td>
<td>int</td>
<td>Number of uncorrectable ECC errors during read</td>
</tr>
<tr>
<td>erase error</td>
<td>int</td>
<td>Number of erase operations that resulted in an error</td>
</tr>
<tr>
<td>final read error</td>
<td>int</td>
<td>Number of read operations that resulted in an error, even upon retry</td>
</tr>
<tr>
<td>final write error</td>
<td>int</td>
<td>Number of write operations that resulted in an error, even upon retry</td>
</tr>
<tr>
<td>meta error</td>
<td>int</td>
<td>Number of errors while accessing the drive’s internal metadata</td>
</tr>
<tr>
<td>read error</td>
<td>int</td>
<td>Number of read operations that resulted in error, but succeeded upon retry</td>
</tr>
<tr>
<td>response error</td>
<td>int</td>
<td>Number of bad responses from the drive</td>
</tr>
<tr>
<td>timeout error</td>
<td>int</td>
<td>Number of operations that timed out without completion</td>
</tr>
<tr>
<td>uncorrectable error (UECC)</td>
<td>int</td>
<td>Number of uncorrectable ECC errors encountered during read operations</td>
</tr>
<tr>
<td>write error</td>
<td>int</td>
<td>Number of write operations that resulted in error, but succeeded upon retry</td>
</tr>
</tbody>
</table>

**Table 1: All 21 features collected**

<table>
<thead>
<tr>
<th>Final Selected Top Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>correctable error count</td>
</tr>
<tr>
<td>cumulative bad block count</td>
</tr>
<tr>
<td>cumulative p/e cycle</td>
</tr>
<tr>
<td>erase count</td>
</tr>
<tr>
<td>final read error</td>
</tr>
<tr>
<td>read count</td>
</tr>
<tr>
<td>factory bad block count</td>
</tr>
<tr>
<td>write count</td>
</tr>
<tr>
<td>status read only</td>
</tr>
</tbody>
</table>

**Table 2: Top features selected**

90% of the healthy drives but does not contain any samples of failed drives. For 1-class isolation forest, we use 250 trees, with a max_depth of 20 to get a good representation of a healthy drive from the input data. Increasing the tree size and max_depth beyond these values decreased precision of the model, indicating overfitting. We also experimentally explored the best value for the contamination factor hyperparameter. The initial hyperparameter values were based on domain knowledge and we performed extensive parameter sweeping and tuning (also for the baselines) to come up with the final hyperparameter values and models. While our training set has zero contamination (no failed drives), we need to inform the model about the contamination factor during inference so that the model can adjust the threshold to select between failed and healthy drives. The empirically determined contamination factor depends on the number of days the model needs to predict ahead and ranges between 0.016 and 0.002.

The 1-class autoencoder model utilizes 4 hidden layers comprising of 50, 25, 25 and 50 neurons respectively. The neurons utilize a tanh activation function. We utilize the Adam optimizer [80] and train the model for 100 epochs. We use early stopping with a patience value of 5 ensuring that the training of the model stops when the loss does not decrease after 5 consecutive epochs. Increasing the number of hidden layers beyond 4 increases the training time significantly without providing performance benefits. We use 10-fold cross validation to evaluate all models.

### 4.4 Deployed System

The processed dataset containing only the top selected features is subsequently used for training the different ML models. In a datacenter we envision our SSD failure prediction technique to be implemented as shown in the block diagram in Figure 3. The telemetry traces are collected periodically from all SSDs in the datacenter and sent to the preprocessing pipeline transforming all input data into numeric values while filtering out incomplete and noisy values. Following data preprocessing, feature selection is performed to extract
the most important features from the data set. The preprocessed data is then either utilized for training or inference. For inference, device anomalies are reported and classified according to our 1-class autoencoder approach. SSDs can then be manually analyzed by a technician or replaced directly. As an alternative, a scrubber can be leveraged to validate the model predictions by performing a low level analysis of the SSD, finding grown bad sectors and other drive issues.

5 RESULTS

In this section, we compare the performance of our proposed 1-class isolation forest and 1-class AutoEncoder techniques to three baselines used in prior work. In particular, we compare against, Random Forest, 2-Class SVM, and Neural Networks (NN) as those have been used in prior work on SSD failure detection [5]. For the baselines, whenever available, we use the same model architecture and hyperparameters as proposed in prior work [5]. For the hyperparameters that we could not find in prior work, we performed a design space exploration and report the best numbers that we could find.

For SSD failure prediction, the primary goal is to predict all SSD failures since the cost of not capturing (mispredicting) a drive that is going to fail is higher than classifying a healthy drive as a failure which can be refuted by scrubbing [55]. Nevertheless, as performing scrubbing induces a performance overhead, achieving both high recall for failed devices and high accuracy for healthy devices is important as well. To satisfy these requirements, for all the experiments, we chose a high enough threshold to capture failures with a minimum recall value of 0.99 and then try to predict these failures with the fewest number of false positives.

For imbalanced datasets, traditional metrics (accuracy, precision, recall and f-score) alone can be deficient in measuring the performance of the classifier. Since the dataset is imbalanced, overfitting to the majority class (predicting all observations as the majority class) can skew performance and still reflect good overall precision, recall and f-score. The receiver operating characteristic curve, or ROC curve [11], is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold values. The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning [65]. The false-positive rate is also known as probability of false alarm [24] and can be calculated as (1 - specificity). The area under the curve (ROC AUC) [26] is calculated to give a single score for a classifier model across all threshold values. This is inline with prior work that utilizes the ROC AUC metric for evaluating anomaly detection models [5].

To evaluate the five ML techniques we first label all 40 million observations in the dataset to separate between healthy and failed drive observations. We then perform a 90% - 10% split of the dataset into training set and evaluation set. For training the 1-class models we remove all failed drive observations from the training set, however, the evaluation set is identical between our proposed 1-class techniques and the three baselines. We use 10-fold cross validation for evaluating all approaches.

5.1 Accurate Prediction of SSD Failures

Figure 4 illustrates the comparative performance of different ML techniques for predicting SSD failures one day ahead. Among the baselines, Random Forest performs best, providing a ROC AUC score of 0.85. Both our 1-class models outperform the best baseline. In particular, 1-class isolation forest achieves a ROC AUC score of 0.91, representing a 7% improvement over the best baseline while 1-class AutoEncoder, outperforms Random Forest by 9.5%.

ROC AUC determines the ability of a model to distinguish between classes (failed vs. healthy in our application). To achieve good performance, models need to achieve both high recall and precision for both failed and healthy classes. Figure 5 explores these metrics for the five approaches in more detail for N=1.
It shows that Random Forest performs equally well than our proposed 1-class Models in terms of Precision and Recall on the majority class of healthy (H) drives, however, performs considerably worse on predicting the minority class of failed (F) drives. For the minority class, 1-class AutoEncoders improve precision by 6% over Random Forest as well as by 68% and 72% over the Neural Network and SVM baselines respectively.

Figure 6 shows the ROC AUC score for the three baselines and our proposed 1-class techniques with a variable percentage of failed drives included in the training set. Note that our 1-class techniques do not include any failed drives in the training set and hence we plot their performance as a straight line. The baselines’ performance, however, depends significantly on the number of minority samples in the training set. For instance, if only 50% of the failed drive observations are included in the training set, our proposed 1-class Autoencoder technique outperforms Random Forest by 13% and NN and SVM by 33%. This shows that particularly in dynamic environments, our 1-class techniques are a better choice than the techniques utilized in prior works.

5.3 Interpreting SSD Failures

This work proposes 1-class Autoencoders for interpreting SSD failures. In particular, our technique exposes the reasons determined by our model to flag a particular device failure. This is achieved by utilizing the reconstruction error generated by the model while reproducing the output using the trained representation of a healthy drive. The failed drives do not conform to the representation, thereby, generating...
an output which differs significantly from the actual observations producing a large reconstruction error. We study the reconstruction error per feature to generate the failure reasons. The features which contribute more than the average error per feature to the reconstruction error, is defined as a significant reason.

Figure 7 shows how often a feature was flagged as a significant failure reason by the autoencoder model, aggregated for all observations from failed drives. The y-axis displays all features utilized by the model, representing a potential failure reason while the x-axis shows the failed drive number. For each drive, we report the failure reason by means of a scatterplot. From Figure 7, we can see that many failed drives show a higher than normal number of correctable errors counting the number of failed reads that could be corrected leveraging error correcting codes (ECC). This indicates that a high number of uncorrectable errors frequently leads to failures, however, it is also only a significant feature in approximately 35% of the drives.

Cumulative bad block represents another important reason determined by the model indicating SSD failures as it shows frequent anomalies, however, again only in less than 30% of the cases. In summary, this analysis shows that there exist particularly relevant features that indicate device failures in many cases, however, only the combination of several features enables accurate failure prediction. We also note that, cumulative UEC (Uncorrectable error count) which has been researched extensively [15, 36, 64] for SSD failure correlation contributed to less than 1% of the failures according to our Autoencoder based model.

5.4 Sensitivity Studies

In the following we provide two additional sensitivity studies. In the first, we evaluate the ability of our models to predict failures multiple days in advance. Predicting further ahead is beneficial for logistical reasons and acquisition purposes. In the second study, we evaluate the effect of feature selection on the five approaches.

5.4.1 Predicting ahead in Time. To optimize drive maintenance and the acquisition of new spare drives, it is preferable to predict drive failures further ahead in time. While the previous sections have focused on predicting one day ahead, Figure 8 evaluated ROC AUC performance on predicting multiple days (N) ahead. As expected, for all five models, prediction performance degrades when predicting further ahead. So while for 1-class Autoencoders performance degrades considerably, 1-class isolation forest can maintain the performance better. In particular, for \( N = 4 \), the 1-class isolation forest model becomes the best performing technique outperforming the Random Forest baseline by 6% and SVM and NN by 11% and 13% correspondingly in terms of RUC-AUC score.

5.4.2 Feature Selection. As mentioned above we used feature selection algorithms for selecting the most important features contributing to failures in our dataset. Tables 1 and 2 list the features before and after feature selection. Figure 9 demonstrates the potential benefit of using feature selection by comparing the model performance (in terms of ROC AUC score improvement) with the original 21 features against the performance of the model trained with only 9 features. As can be seen, all techniques benefit from feature selection, for instance, Random Forest’s absolute ROC AUC score improves by 3.7% when utilizing feature selection, while 1-class
Autoencoder’s ROC AUC performance increases by 3.3%. Feature selection also reduces the number of features used for training the models resulting in up to 41% (for autoencoder models) reduction in training times as can be seen from Table 3.

6 DISCUSSION
We discuss our proposed 1-class models below.

6.1 1-Class Isolation Forest
Anomaly detection approaches leveraging isolation forests are generally trained on both the majority and minority class. Perhaps surprisingly, we found that isolation forests trained only on the minority class performed exceptionally well, particularly for detecting unseen failures outperforming the performance of baseline approaches (Random Forest, 2 class SVM and Neural Network based models). 2-class models use data from both classes to learn a representation for each class. Since the number of samples for failed drives in our case is significantly lower than good working SSDs, the model has less samples to learn from and hence is more likely to misclassify previously unseen failed SSDs. 1-class models, in contrast, learn a representation of a good working SSD and are more likely to classify previously unseen anomalies correctly (1-class models do not suffer from overfitting to the limit training set of failed SSDs).

Our approach does not require training on all different failure types to detect failures and hence both generalizes and scales well when provided with new healthy observations. The approach outperforms autoencoders when predicting more than two days ahead and is faster to train requiring fewer training samples. Nevertheless, it was not able to outperform autoencoders (for N=1 and N=2) due to a higher false positive rate (anomalies as reported by the model which are not actual failures). We plan to use second level supervised binary classification in the future to teach the model about the known failures to eliminate more false positives during evaluation.

6.2 1-Class Autoencoder
To our knowledge, this is the first application of a deep learning based 1-class autoencoder for predicting SSD failures. We used the data from healthy drives to create an encoded representation of a healthy drive. Upon providing the test data
points to the encoded representation, we recorded the difference between the observed and generated output. Since the anomalous data points do not fit the encoding well, they tend to have higher error values. As in 1-class isolation forests, the autoencoder does not need to train on the minority dataset. Autoencoders performed best while predicting failures up to 2 days ahead, achieving highest accuracy, precision and ROC AUC score with a recall of 0.99. It performed worse than 1-class isolation forests when predicting ahead 3 or more days achieving lower precision, however, autoencoders enable interpretation of the model predictions. In particular, we can learn why the model flagged an observation as a failure to inform the repair and maintenance procedure.

<table>
<thead>
<tr>
<th>ML Technique</th>
<th>Features</th>
<th>Training Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>9</td>
<td>695.4</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>1095.6</td>
</tr>
<tr>
<td>Neural Network</td>
<td>9</td>
<td>1496.87</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>2550.87</td>
</tr>
<tr>
<td>SVM</td>
<td>9</td>
<td>1156.6</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>1885.6</td>
</tr>
<tr>
<td>Isolation forest</td>
<td>9</td>
<td>499.57</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>686.54</td>
</tr>
<tr>
<td>Autoencoder</td>
<td>9</td>
<td>1750.57</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>2781.89</td>
</tr>
</tbody>
</table>

Table 3: Model training time (N = 1)

Anomaly detection techniques using both traditional machine learning [61] and deep learning techniques [63] have been successfully applied in various fields of research. Adewumi [2] provide a detailed review of deep learning-based techniques for fraud detection. A broad survey of deep anomaly detection (DAD) methods for cyber-intrusion detection is presented by Kwon [44]. An overview of DAD techniques for the Internet of Things (IoT) and big-data anomaly detection is introduced by Mohammadi and Mehdi [58]. Sensor networks anomaly detection has been reviewed by Ball [7]. The state-of-the-art deep learning based techniques for video anomaly detection along with various categories have been presented in Kiran [41]. Other applications of anomaly detection include predicting failures in cloud systems [72], the medical domain [48], and self-driving vehicles [4] [3]. Zhang [79] introduced ATAD, a method of detecting anomalies in cloud systems, by training the model on one dataset and using transfer learning to use the model for another dataset. Our work contrasts in using a combination of several feature selection techniques to select the most relevant features for training the model for generating failure reasons. In addition to anomaly detection applications, machine learning has been applied to improve the performance and efficiency of storage systems and SSDs [12, 16, 33, 42, 49].

8 CONCLUSION

This paper provides a comprehensive analysis of machine learning techniques to predict SSD failures in the cloud. Therefore, we collect SSD telemetry information from over 30,000 drives over a period of six years from Google’s datacenters. We observe that prior works on SSD failure prediction suffers from the inability to predict previously unseen failure types motivating us to explore 1-class machine learning models such as 1-class isolation forest and 1-class autoencoder. We show that our approaches outperform prior work by 9.5% ROC-AUC score by significantly improving on the prediction accuracy for failed drives. For dynamic environments, where only a subset of the different drive failure types are part of the training set, our 1-class techniques improve over the baselines by 13%. Finally, we show that 1-class autoencoders enable interpretability of model predictions by exposing the reasons determined by the model for predicting a failure.

9 ACKNOWLEDGEMENTS

We thank Arif Merchant from Google for providing access to the Google traces and helpful feedback on the manuscript. We also thank the anonymous reviewers for their comments. This work was generously supported by Samsung Semiconductor Inc. and NSF grant #1942754.
REFERENCES


Improving the Accuracy, Adaptability, and Interpretability of SSD Failure Prediction Models

SoCC '20, October 19–21, 2020, Virtual Event, USA


