Layerwise Perturbation-Based Adversarial Training for Hard Drive Health Degree Prediction

Jianguo Zhang* Lifang He† Fan Duo‡ Ji Wang§ Christopher Tran* Philip S. Yu*¶

Abstract

With the development of cloud computing and big data, the reliability of data storage systems becomes increasingly important. Previous researchers have explored various statistical and machine learning methods based on SMART attributes. Most of them focused on predicting whether the hard drive needs to be replaced or not. However, hard drives in real-world data systems usually deteriorate gradually rather than suddenly. Predicting hard drives with different health degrees can be helpful to take different actions in advance . Hence, we assess the health degree of hard drives based on SMART attributes. Furthermore, data imbalance is also a challenging problem in this task, as there are usually less than 3% of failed hard drives in a data system. We propose two schemes to better handle imbalanced data and improve performance on minority classes. First, we design a layerwise perturbation-based adversarial training method which can add perturbations to any layers of a neural network (NN) to improve generalization. Second, we design a semi-supervised learning method which utilizes unlabeled data that have a potential of failure to further improve the performance of NNs. Our extensive experiments on two real-world datasets demonstrate the superiority of the proposed schemes for both supervised and semi-supervised classification, which can correctly predict the failure degrees of hard drives 5 and 15 days in advance.

1 Introduction

Nowadays, with huge progress in computing and rapid growth of data scale, high cost of data center downtime makes a critical challenge to the reliability of storage system. Storage system problems caused by hard drives failure frequently happen, although the probability of failure for each hard drive is pretty low. Hard drives failure may cause users to lose valuable information and even break down a whole data center. Large-scale data center exponentially enlarges this probability. To address this problem, Self-Monitoring, Analysis and Reporting Technology (SMART) [2] has been applied to hard drives, which monitors and analyzes internal attributes of each hard drive, aiming at timely alarms for failure. However, this technique cannot efficiently predict hard drive states, as it simply uses a threshold-based normalization strategy which has weak prediction power [17, 20]. To improve failure prediction accuracy, many efforts have been made based on SMART attributes, including analyzing the failure behaviors of hard drives [5, 9], and designing practical algorithms for assisting diagnosis of hard drive issues [17, 18, 26, 27, 31]. Specifically, most of previous studies focused on proactive failure prediction, which forecasts hard drive failure before hand and gives a binary result either the hard drive is healthy or it is close to the failure line. However, hard drives usually fail gradually rather than abruptly. To harness this potential of gradual change, it is necessary to develop a health degree model which can give the drive a health state assessment rather than a simple binary result.

On the other hand, data from storage systems are highly unbalanced, since we usually have less than 3% of failed hard drives in a data system while having large volume of healthy hard drives. This data imbalance will affect the model training and lead to biased fitting to healthy hard drives during the learning processes. To assess the health states of hard drives on such data, it is inevitable to consider the unbalanced relationship between failed hard drives and healthy hard drives. However, most of the statistical and machine learning algorithms optimize the overall classification accuracy, and hence sacrifice the prediction performance on the minority classes. It is desirable to study the problem of hard drive health degree prediction by taking into account the class imbalance of dataset.

In order to tackle above problems, we proposed two strategies. First, we introduce adversarial learning strategy. Adversarial learning [7, 24, 28] is a strong regularization method and adversarial perturbations provide the perturbations affecting the NN's predictions in the most sensitive way. An adversary applies adversarial perturbations on input data to force the NNs to learn a better distribution of data and away from overfitting. Recently it has been successfully used in image classification [11] and text classifi-

*Department of Computer Science, University of Illinois at Chicago. {jzhan51, ctran29, psyu}@uic.edu

†Department of Healthcare Policy and Research, Weill Cornell Medical School, Cornell University. lifanghescut@gmail.com

‡School of Electronic Engineering, Beijing University of Posts and Telecommunications. dolphinchbupt@gmail.com

§College of Information System and Management, National University of Defense Technology. wangji@nudt.edu.cn

¶Institute for Data Science, Tsinghua University.

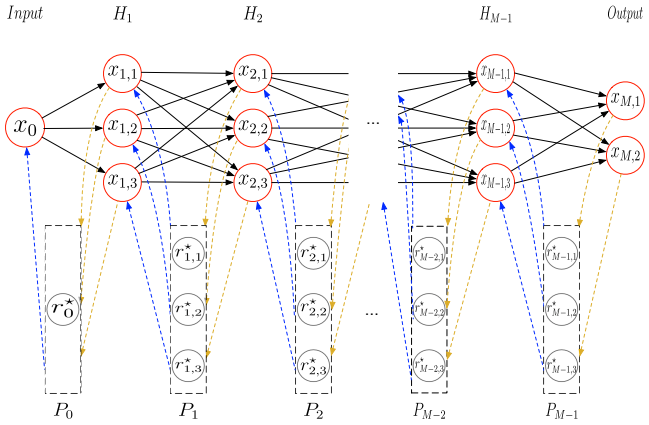

Figure 1: An overview of LPAT. First, it performs a feedforward process to compute the output of the neural network. Next, it performs back propagation to update parameters and store gradients to gradient accumulation layers. Then, it adds layerwise adversarial perturbations to each network layer and performs feedforward process to compute the output of the neural network. Finally, it performs back propogation to update parameters for all network layers. $\{H_i\}_{i=1}^{M-1}$ denotes hidden layers. $\{P_i\}_{i=0}^{M-1}$ are gradient accumulation layers. Yellow lines denote the process of P_i storing gradients back propagated to network layer i. Blue lines denote the process of adding layerwise adversarial perturbation $r_i^\star$ to X_i in network layer i.

cation [14]. However, previous methods usually only add adversarial perturbations to inputs which may restrict model's performance. We extend adversarial training into hard drive healthy degree prediction task and design a layerwise adversarial training method. Second, to deal with data imbalance, a common approach is to import more minority categories. For failed hard drives, more sequences can be added by sampling more time intervals before a hard drive fails. However, it will bring another problem: how to label these sequences. Human labeling is a time and labor consuming task. For example, hard drives with failure sequences are difficult to classify. Hard drives with time intervals that are close to the final failure state are easy to label as a failure or "close to failure". However, hard drives with time intervals before the "close to failure" time interval may be difficult to classify, although the probability of failure may be high. In this case we may treat these instances as unlabeled data. Therefore, we apply a semi-supervised learning procedure to gain more useful information from unlabeled data.

To implement the above two strategies, we tackle the challenge by proposing a layerwise perturbation-based adversarial learning (LPAT) method. The proposed method utilizes the regularization ability of adversarial learning and magnifies this ability by adding perturbation layer to layer. In order to achieve time memory ability on long dependence sequences, an LSTM layer is also incorporated into our model, as final failure may depend on long range data. It is vital and appropriate to leverage time memory based model to evaluate the health status of hard drive and predict possible failure.

Our main contributions can be summarized as follows:

• We design a novel LPAT method to hard drive status prediction task. Instead of only adding perturbation to inputs of NNs, LPAT can flexibly add perturbation to specific layers or all layers to better address the data imbalanced problem. Besides, it can utilize unlabeled data to further improve the performance of NNs.

• To the best of our knowledge, we are the first to design a semi-supervised learning method with layerwise perturbation-based adversarial training for deep learning models, and use them to deal with hard drive status prediction task in both supervised and semi-supervised learning.

• Extensive experiments on two datasets demonstrate that LPAT can improve performance of hard drive health predictions on overall and minority classes for both supervised and semi-supervised learning. Specifically, it can predict hard drive health degrees 5 days or 15 days before failure using sequential SMART attributes, which is more useful in practice than only predicting healthy or failed.

2 Problem Definition

In this work, we aim to a model that utilizes the training dataset generated as described in the Data Preparation Section, and make drive replacement predictions for unseen data. We denote training dataset as $D = \{(x^{(1)}, y^{(1)}), ..., (x^{(N)}, y^{(N)})\}$, where $x^{(i)} \in X^{k\times n}$ is the i^{th} training sample consisting of continuous observation information from day t_i to day t_{i+k}. Each day has a feature vector of relevant n SMART attributes, and y denotes health degrees of a drive (here $y \in \{0, 1, 2\}$). Where '0' represents a "red alert" which means the remaining time for the hard drive is less than 5 days, '1' represents a drive that is "going to fail" in 15 days, and '2' is "healthy". Note that time intervals could be set differently [29]. Our goal is to learn a function $f : X \to \{0, 1, 2\}$ that minimizes the empirical loss $\hat{\mathcal{L}}(\Theta)$ based on training samples, where Θ denotes a vector of model parameter during training. Intuitively, our goal is to train a model that correctly predicts the health degree of a hard drive. $\hat{\mathcal{L}}(\Theta)$ can be formulated as:

$$\hat{\mathcal{L}}(\Theta) = -\frac{1}{N}\sum_{i=1}^{N} log\ p(y^{(i)}|x^{(i)}; \Theta) \tag{2.1}$$

3 Methodology

3.1 LSTM Based Basic Model Our basic model includes two dense (fully connected) layers, followed by a LSTM

layer and a dense layer. LSTM is the basic building block of the model. While several variations exist, we base our work on the following LSTM information:

$$\hat{i_t}, \hat{f}_t, \hat{o}_t, \hat{j}_t = W_{(\cdot)} x_t + U_{(\cdot)} h_{t-1} + b_{(\cdot)}, \tag{3.2}$$

$$c_t = i_t \odot j_t + f_t \odot c_{t-1}, \tag{3.3}$$

$$h_t = o_t \odot tanh(c_t), \tag{3.4}$$

where $\odot$ denotes element-wise multiplication and σ is a sigmoid function to make the gating values in [0,1]. $j_t = \tanh(\hat{j}_t)$ is a proposed update to the cell state. $(\cdot)_t = \sigma(\hat{\cdot})_t$ for types (i, f, o) are $input\ gate$, $forget\ gate$ and $output\ gate$. $x_t \in R^n$ is the input from the lower dense layer at time step t. With q LSTM units, the dimensionality of the weight matrices and bias vectors which need to be trained are $W_{(\cdot)} \in R^{q \times d}$, $U_{(\cdot)} \in R^{q \times d}$ and $b_{(\cdot)} \in R^q$ for all types (i, j, f, o). The *memory cells*, c_t, are designed to counteract the problem of vanishing/exploding gradient, thus enabling learning of dependencies over larger time than traditional recurrent networks. The $forget\ gate$ is for resetting the memory cells. The $input\ gate$ and $output\ gate$ are for controlling the input and output of the memory cells.

3.2 Adversarial Training and Virtual Adversarial Training Our work is closely related to adversarial training [7, 24] and virtual adversarial training [16]. We therefore formulate these two methods before introducing our method.

Previous works on adversarial training have observed that training the model with adversarial examples acts as a regularization function and improves the performance and robustness of the base network on the test data. When applied to classification problems, adversarial training makes the DNN model robust to adversarial examples, so the classifier is robust and can learn a better distribution of training data. The process can be formed as a min-max problem. The adversary applies the worst perturbation to the data to maximize classification error, while the learning procedure tries to be robust to such perturbations through minimizing the misclassification error caused by the adversary. The min-max problem can be formulated as an additional cost function:

$$\min_{l} \max_{r, \|r\| \leq \epsilon} l(p(y^{(i)}|x^{(i)} + r, \Theta), q(y^{(i)})), \tag{3.5}$$

where ϵ is the magnitude of the perturbation, and r needs to be tuned. l is the non-negative cross entropy between two distributions. $q(y^{(i)})$ is a one-hot vector of corresponding labels $y^{(i)}$. This additional cost function can be simplified as:

$$-log\ p(y^{(i)}|x^{(i)} + r_{adv}; \Theta), \tag{3.6}$$

where

$$r_{adv}^{(i)} = arg \min_{r, \|r\| \leq \epsilon} log\ p(y^{(i)}|x^{(i)} + r; \hat{\Theta}), \tag{3.7}$$

and $\hat{\Theta}$ is a constant set to the current parameters of the classifier [14]. In general, the exact calculation for r is intractable for many deep learning networks. [7] proposed a linear approximation with L_2 norm constraints in equation (3.6) as:

$$r_{adv}^{(i)} \approx -\epsilon \frac{g}{\|g\|_2}, \tag{3.8}$$

where $g = \nabla_{x^{(i)}} log\ p(y^{(i)}|x^{(i)}; \hat{\Theta})$.

Rather than just use the one-hot vector $q(y^{(i)})$, [16] proposed the virtual adversarial training (VAT) which can use true conditional probability $q(y^{(i)}|x^{(i)})$ given the input $x^{(i)}$, and $q(y^{(i)}|x^{(i)})$ can be replaced by the current estimate $p(y^{(i)}|x^{(i)}; \hat{\Theta})$. The additional cost function of this method is the following:

$$KL\left[p(\cdot|x^{(i)}; \hat{\Theta}) \,||\, p(\cdot|x^{(i)} + r_{vat}; \Theta)\right], \tag{3.9}$$

$$r_{vat}^{(i)} = arg \max_{r, \|r\| \leq \epsilon} KL\left[p(\cdot|x^{(i)}; \hat{\Theta}) \,||\, p(\cdot|x^{(i)} + r; \hat{\Theta})\right], \tag{3.10}$$

where $KL\left[p \,||\, q\right]$ denotes the KL divergence between two distributions p and q. Through minimizing equation (3.9), the classifier attempts to be resistant to perturbations $r_{vat}^{(i)}$ which destroys the local smoothness of $p(x^{(i)}; \hat{\Theta})$ at $x^{(i)}$ in the most sensitive way. The definition of r_{vat} in equation (3.10) does not require correct label information, while the adversarial loss defined in equation (3.6) needs corresponding labels $y^{(i)}$. Thus VAT can be used for semi-supervised learning.

3.3 Layerwise Perturbation-based Adversarial Training In this work, we design a layerwise perturbation-based adversarial training method which applies adversarial perturbation to time series data and perturbing intermediate layer instead of only adding perturbation to inputs. Previous works either primarily applied them in image classification tasks or continuous word embedding with only adding perturbations to inputs in supervised learning ways.

Figure 1 illustrates a deep neural network equipped with the proposed method. For a model which consists of $M-1$ hidden layers, $m = 0$ is the input layer and $m = M-1$ is the final hidden layer. The gradient accumulation layer P_m serves two purposes. First, it temporarily stores the back propagation gradients on the output of the m^{th} layer for labeled and unlabeled data. Then it computes and stores the layerwise adversarial perturbations for the m^{th} layer. $\hat{x}_m$ is the output of the m^{th} layer. r_m represents the perturbation

added to $\hat{x}_m$. The layerwise adversarial perturbation for the m^{th} layer can be computed as:

$$r_m^\star = arg \max_{r_m, \|r_m\| \leq \epsilon} KL_m, \tag{3.11}$$

where

$$KL_m = KL\left[p(\cdot|\hat{x}_m; \hat{\Theta}_m) \,||\, p(\cdot|\hat{x}_m + r_m; \hat{\Theta}_m)\right]. \tag{3.12}$$

As $p(\cdot|\hat{x}_m; \hat{\Theta}_m)$ is differential with $\hat{\Theta}_m$ and $\hat{x}_m$, the layerwise adversarial perturbation can be computed using equation (3.11) and equation (3.12). However, the gradient of KL_m cannot be computed directly as the first derivative $\nabla_{r_m} KL_m|_{r_m=0}$ has the minimum value 0 when $r_m = 0$. Therefore, we randomly sampled a small random norm vector d, use 2nd-order Taylor approximation [16], and one-time power iteration method [6] on equation (3.11) to approximate the virtual adversarial perturbation of m^{th} layer as following:

$$r_m^\star \approx \epsilon_m \frac{g}{\|g\|_2} \tag{3.13}$$

where

$$g = \nabla_{\hat{x}_m + d} KL\left[p(\cdot|\hat{x}_m; \hat{\Theta}) \,||\, p(\cdot|\hat{x}_m + d; \hat{\Theta})\right] \tag{3.14}$$

Then for labeled and unlabeled dataset with size N', total loss are given by:

$$\mathcal{L}(\Theta) = \hat{\mathcal{L}}(\Theta) + \lambda \cdot \mathcal{L}_{lap} \tag{3.15}$$

where $\hat{\mathcal{L}}(\Theta)$ is the negative log-likelihood for the labeled dataset given by equation (2.1), and

$$L_{lap} = \frac{1}{N'} \sum_{i=1}^{N'} KL\left[p(\cdot|\hat{x}_m^{(i)}; \hat{\Theta}_m) \,||\, p(\cdot|\hat{x}_m^{(i)} + r_m^{\star\,(i)}; \hat{\Theta}_m)\right] \tag{3.16}$$

for $m = 0, 1, ..., M-1$.

The gradients computed from mini-batch input samples aggregated through intermediate layers are used to calculate perturbations for the current mini-batch. Since equation (3.13) is a semi-supervised way which does not necessarily need label information, we do not need to shuffle mini-batch data to avoid overlap of class labels [19]. Note that the number of accumulation layers P is not necessarily equal to the number of model layers as we can add perturbations to any layers. When $P = 0$, we add layerwise adversarial perturbations only to the input. In this situation equation (3.11) becomes equation (3.10). $P = M - 1$ means to add layerwise adversarial perturbations to all layers of the model. The training procedure is summarized in algorithm 1. Note the parameter ϵ_i can be set to a fixed value for all layers or set differently for different layers. At training time, we add the gradient accumulation layer P_i after each neural network layer. P_i is initialized with zeros, it is only used to store backpropagated gradients and compute layerwise adversarial perturbations. At testing time, we remove all the gradient accumulation layers from the training model.

Algorithm 1 Layerwise Perturbation-Based Adversarial Training Algorithm

Input: Randomly initialized Network NN. B is the batch sampled at a training step s of size k, with labeled and unlabeled input training samples $(X, \cdot)$. $\{P_i\}_{i=0}^{P-1}$ are the gradient accumulation layers with stored layerwise adversarial perturbations $\{r_i^\star\}_{i=0}^{P-1}$ initialized with zero. All gradient accumulation layers inactive at the initial training step $s = 0$. ϵ_i denotes perturbation size.

1: **for** $s = 0, 1, ..., B-1$ **do**
2: Perform forward propagation to compute the output of network NN
3: Perform back propagation using loss function, each accumulation layer P_i temporarily stores the gradients backpropagated to the network layer i
4: Compute and store layerwise adversarial perturbation $r_i^\star$ for each layer i using equation (3.13)
5: Perform forward propagation with layerwise adversarial perturbation. Let X_i be the output to the network layer i, we have:

$$X_i = X_i + \epsilon_i \cdot r_i^\star$$

6: Perform back propagation to update parameters for all network layers
7: **end for**

4 Experiments

In order to empirically evaluate the effective of our proposed methods in addressing hard drive healthy degree prediction, we conduct experiments on two datasets and compare with several existing methods. In the following, we first introduce the datasets used in the experiemnts. Then we present the experimental results as well as analysis.

4.1 Data Preparation Our evaluation and analysis are based on the Backblaze dataset[1]. The hard drive statistics have been recorded and saved daily from the drives in the data centers since April 2013. At the end of 2016 it included 73,653 spinning hard drives. The dataset includes: (1) date, (2) serial number of the drive, (3) model number of the drive, (4) drive capacity, (5) SMART statictics, and (6) failure: 0 if the drive is alive, and 1 if this was the last day the hard drive was working before failing.

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

Table 1: Number of Healthy (H) and Failed (F) drives before and after data cleaning and aggregation.

	Original		Pre-Processing	
	# of H	# of F	# of H	# of F
ST-1	33800	938	30685	758
ST-2	8660	48	7932	47

In this paper, we build and evaluate our model based on SMART statistics of two different data sources in year 2016. For Seagate ST4000DM000 (ST-1), it has the largest population in the dataset of Backblaze. There are 33,800 healthy hard drives and 938 failed hard drives in the dataset. After data cleaning and aggregation [5], there are 30,685 healthy hard drives and 758 failed hard drives. To further evaluate our method, we collect another small dataset from Seagate ST8000DM002 (ST-2). There are 8660 healthy hard drives and 48 failed hard drives. After pre-processing there are 7932 healthy drives and 47 failed drives. Note ST-1 and ST-2 are different models although they are from the same hard drive manufacture [5]. Table 1 lists the details of the dataset. We evaluate our data based on these two datasets in our following experiments.

It is apparent that the dataset is extremely imbalanced. For example, there are only 2.47% of samples that are failed hard drives in ST-1. The prediction model will be biased to the healthy drives if the dataset is used directly. Hence, it will show poor performance to discriminate failed hard drives from healthy ones. To alleviate this problem, we try to select the representative subset of healthy hard drives to construct the dataset for training and testing our model, by which the amount of healthy hard drives can be reduced without a high loss of information about healthy hard drives. We first use the k-means clustering algorithm [13] to cluster the healthy hard drives into 10 clusters. Then, for each cluster, the top 30% samples closest to the centroid are selected as the representative subset of healthy hard drives. Finally, we generate a relatively less unbalanced dataset.

We follow previous works [5, 17, 29] which use z-score, rank-sum test and changepoint analysis to select features from 47 SMART attributes. As the values of different SMART attributes vary widely, we rescale the values of each selected SMART attributes by the following formula to avoid bias to SMART attributes with large values:

$$v' = \frac{v - v_{min}}{v_{max} - v_{min}} \tag{4.17}$$

where v is the original value of a SMART attribute, and v_{min} and v_{max} are the minimum value and the maximum value of a SMART attribute.

We conduct healthy degree prediction in our experiments. Similar to the definitions used in [12, 29], we set the sample labels based on the residual life of hard drives and divide the remaining time into different ranges according to the time before failure. Label '0' is "red alert", which means the remaining time of the drive is less than 5 days, the hard drive should be replaced as soon as possible. '1' means "going to fail" in 15 days, the users should be prepare to replace the hard drive. '2' is "healthy". Although we label hard drives with larger than 15 days remaining as healthy, we take the failed hard drives samples as unlabeled data. This is because for failed hard drives with life time close to 15 days, it is hard to determine whether SMART attributes during this period belong to which health status, although there is high potential for a particular drive in this range to fail. For healthy drives we do not take any unlabeled data.

Table 2: Summary of datasets

	Train	Valid	Test	Unlabeled
ST-1	17137	4285	5356	8282
ST-2	755	189	236	-

We split each dataset into three sets of training set, validation set and test set. For supervised learning, we use 80% of all labeled data of each dataset as training set and the rest as testing set, and we we split 20% of each training set as validation set for tuning hyper-parameters. For semi-supervised learning, we add unlabeled data into training set and keep validation set and test set unchanged. We use ST-1 for both supervised learning and semi-supervised learning. We don't take unlabeled data for ST-2 and use it only in supervised learning. We summarize information about dataset samples in Table 2.

4.2 Baselines and Metrics To demonstrate the ability of our approach, we compare with the following methods:

• **SVM**: We use a Gaussian-RBF kernel based SVM, which is the most widely used method for classification.

• **DT**: It is a decision tree method, which creates a tree-like model to predict the value of a target variable by learning decision rules inferred from the data features.

• **RGF** [5]: It is a regularized greedy forests based method to automatically select SMART attributes and predict the impending replacements of hard drives. In experiment, we follow the setting of [5].

• **RNN** [29]: It is a recurrent neural network based model for predicting hard drive failure and giving health degrees, which treats the observed SMART attributes as time-sequence data. In experiment, we follow the setting of [29].

• **basic**: Our basic model which includes dense layers and LSTM layers. Specifically, in this paper, we use the following setup to train and evaluate our methods: Inputs-Dense-Dense-LSTM-Dense-Output.

• **basic+AT**: An adversarial training method proposed

in [7] which uses a fast gradient method to resist adversarial examples and improves robustness of neural networks by adding perturbations to inputs. Here we integrate this method into the above basic model as a compared method.

• **basic+VAT**: A virtual adversarial training method proposed in [15], which determines the adversarial direction and adds perturbations to inputs from the model distribution alone without necessarily using the label information. We use it the same as AT method.

For our layerwise perturbation-based adversarial training (LPAT) method, we compare three variations as follows:

• **LPAT-Bottom**: It adds adversarial perturbations to the bottom two layers: Inputs-P-Dense-P-Dense-LSTM-Dense-Outputs. Where P is the gradient accumulation layer.

• **LPAT-Top**: It adds adversarial perturbations to the top LSTM layer and dense layer: Inputs-Dense-Dense-LSTM-P-Dense-P-Outputs.

• **LPAT-All**: It adds adversarial perturbations to all layers: Inputs-P-Dense-P-Dense-P-LSTM-P-Dense-P-Outputs. For each layer, the epsilon which controls the strength of perturbation can be set differently.

For all compared methods, we measure the overall accuracy, and each class's precision, recall, F1 score and get the separate and Macro-averaged results. For the basic model, we set two bottom dense layers with 128 units and the activation set to None. The LSTM layer has 200 units. The final layer includes 3 units as we have 3 classes. The time sequence window for each sample is set to 20. We use RMSProp optimizer [4] with learning rate 0.001. The mini-batch size is 128. The epsilon ϵ controlling strength of adversarial perturbation are empirically set in $[0, 5]$. The hyperparameter λ set in $[0, 10]$. The training epochs are set to 255.

Our methods are implemented in Tensorflow [1] and keras. The NVIDIA CUDA Deep Learning Neural Networks accelerated the training process. All the experiments are trained and tested on four NVIDIA Tesla K80 GPUs. Codes are available on the Internet[2].

4.3 Supervised Learning We first conduct experiments using all labeled data. As can be seen from Table 3 and Table 4, the proposed LPAT method achieves the best results on both datasets. It improves 3% on accuracy and 4% on Macro-F1 score of ST-1 compared to the best baseline method. As LPAT can flexibly choose which layer and how many layers to add perturbations, it makes the learning more robust and have better performance than only adding perturbations to inputs. It also improves 4% on accuracy and 7% on Macro-F1 of ST-2 over other baseline methods. Besides, it is known that small dataset is easier to cause overfitting, our results on ST-2 show the regularization and generalization ability of layerwise adversarial learning in preventing overfitting. While classical machine learning methods are hard to perform well on this task. Among the baselines, RNN based on gradually changing sequential SMART attributes performs better than DT and SVM on both datasets. While our basic model equipped with a LSTM layer perform better than RNN based method. This is because deep neural network models can identity more complex data representations than traditional methods, which help improve performance. Another vital reason for better performance could be recurrent deep neural network methods have long time memory ability, which allows further improvement on time series task performance. On ST-1 dataset, RNN performs better than RGF and slightly worse than RGF on ST-2 dataset. As ST-1 and ST-2 datasets are collected by different models whose SMART attributes are also different. RGF which automatically select SMART attributes may perform better in feature obvious dataset [5]. Models with adversarial learning achieves better results than basic neural networks without adversarial training. VAT using true conditional probabilities of given inputs slightly improves over AT using one-hot vectors on both datasets.

We further test on different possible patterns and get results from different perturbation positions. For ST-1, in LPAT-Bottom, we add perturbations on the bottom two lay-

Table 3: Supervised learning on ST-1 data.

	Accuracy	Precision	Recall	Macro-F1
DT	0.8235	0.7362	0.7470	0.7409
SVM	0.6419	0.3981	0.4638	0.4282
RGF	0.7435	0.6838	0.5484	0.5613
RNN	0.8432	0.8012	0.7982	0.7998
basic	0.8212	0.8034	0.7993	0.8015
basic+AT	0.8682	0.8175	0.8134	0.8151
basic+VAT	0.8701	0.8256	0.8209	0.8232
LPAT+Bottom	0.8885	0.8398	0.8502	0.8447
LPAT+Top	0.9003	0.8552	0.8606	0.8576
LPAT+All	**0.9016**	**0.8576**	**0.8671**	**0.8622**

Table 4: Supervised learning on ST-2 dataset.

	Accuracy	Precision	Recall	Macro-F1
DT	0.8220	0.7395	0.7258	0.7313
SVM	0.3432	0.3766	0.3724	0.3154
RGF	0.9195	0.8697	0.8334	0.8492
RNN	0.8771	0.8392	0.7860	0.8088
basic	0.9153	0.8595	0.8450	0.8450
basic+AT	0.9195	0.8620	0.8546	0.8578
basic+VAT	0.9237	0.8787	0.8660	0.8660
LPAT+Bottom	0.9407	0.9047	0.9124	0.9074
LPAT+Top	**0.9661**	**0.9441**	**0.9308**	**0.9392**
LPAT+All	0.9619	0.9415	0.9215	0.9311

[2]https://github.com/JianguoZhang1994/Laywise-Perturbation-Based-Adversarial-Learning

Table 5: Supervised learning results on ST-1 and ST-2, remaining time no more than 5 days ("red alert") belongs to '0' class, "going to fail" in 15 days belongs to '1' class.

	ST-1						ST-2					
	Precision		Recall		Macro-F1		Precision		Recall		Macro-F1	
	≤ 5	≤ 15	≤ 5	≤ 15	≤ 5	≤ 15	≤ 5	≤ 15	≤ 5	≤ 15	≤ 5	≤ 15
basic+AT	0.729	0.769	0.735	0.832	0.722	0.819	0.731	0.933	0.607	0.777	0.667	0.848
basic+VAT	0.735	0.806	0.709	0.859	0.722	0.831	0.727	0.952	0.857	0.740	0.787	0.833
LPAT+Bottom	0.722	0.816	0.743	0.862	0.732	0.834	0.806	0.940	0.892	0.870	0.847	0.904
LPAT+Top	**0.745**	0.831	0.721	0.853	0.737	0.842	**0.893**	0.943	**0.893**	**0.926**	**0.893**	**0.935**
LPAT+All	0.725	**0.837**	**0.801**	**0.870**	**0.761**	**0.853**	0.889	**0.961**	0.857	0.907	0.873	0.933

ers. This method works better than adding perturbation on input layer only, but is outperformed by LPAT-Top, which adds perturbations on the top layers like LSTM layer and dense layer. Among them LPAT-ALL achieves the best results. One reasonable explanation could be the effect of the LSTM layer. Adding perturbations on the bottom layers before the LSTM layer will make perturbations wrapped in the LSTM layer and show less effect to the loss function. This undermines the power of the adversary thus not helping much on regularization compared to adding perturbations after the LSTM layer. For ST-2, LPAT-Top achieves the best results on accuracy and Macro-F1 score, and its performance has little difference with LPAT-ALL. As we know, smaller datasets are easier to overfit. We speculate mainly addding perturbations to top or all layers can have better regularization ability on small dataset.

Compared with Table 3 and Table 4 which show overall performance, Table 5 shows the results on minority classes. Note the training samples size of '1' class is two times larger than that of '0' class on both datasets. Recall represents the failure detection rate, i.e. the fraction of failed drives that are correctly classified as failed. Precision represents the ability for a classifier to capture all hard drives fails in 5 and 15 days. The proposed methods have better performance of both precision and recall than other methods on both datasets. In addition, LPAT-ALL improves 2% of F1 score than LPAT-Bottom on ST-1, LPAT-All improves 3% of F1 score than LPAT-Bottom on ST-2. This demonstrates adding perturbations to different layers further improves the performance of classifiers. Based on the observation, our methods can give technicians and users more time to take different actions 5 to 15 days in advance.

4.4 Semi-supervised Learning In this part, we focus on the ST-1 dataset which includes unlabeled data. Figure 2 shows the validation loss using all unlabeled data and 100% labeled data for AT, VAT, and LPAT+All. We can see that VAT and LPAT+All produce lower validation loss, since they can utilize unlabeled data, while VT can only use labeled data. AT becomes unstable as the number of epochs increases, although it can conduct the task of resisting adversarial perturbations. LPAT+All keeps the loss lower for the longest time compared to the other methods, since it can resist adversarial perturbations in every layer, and thus it has better performance than VAT which can only adds perturbations to inputs.

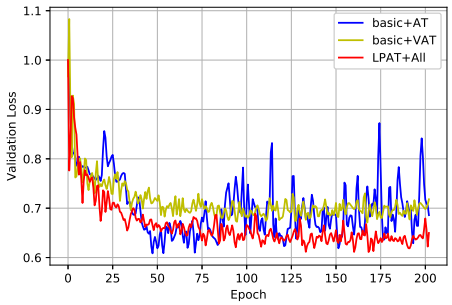

Figure 2: Validation loss of adversarial training (AT), virtual adversarial training (VAT) and layer wise adversarial training in all layers (LPAT+All). $\lambda = 1$ for all methods. $\epsilon = 0.5$ for VT and VAT. For LPAT+All, $\epsilon = 1$ for all layers except the last layer set to $\epsilon = 0.1$. The optimal value differs between different methods, but the value in $[0, 2]$ usually performs well for above methods and provides a fair comparison.

Furthermore, we analyze the influence of different percents of unlabeled data. We experiment with different sizes of unlabeled training samples $N_{ul} = \{0\%, 20\%, 40\%, 60\%, 80\%, 100\%\}$ and observe the effect of N_{ul} on the overall accuracy. Note the total labeled training samples are 17137 and original unlabeled training samples are 8282. We randomly sample 10000 samples from labeled training data and treat the remaining labeled data as unlabeled to increase size of unlabeled data, and now the unlabeled training data becomes 15419. We use the validation set of fixed size 4285 and test set of fixed size 5356. We use all samples excluding validation set and test set to train the neural network. For example, when $N_{ul} = 20\%$, the training samples are $10000 + 15419 \times 0.2 = 13084$. The architecture

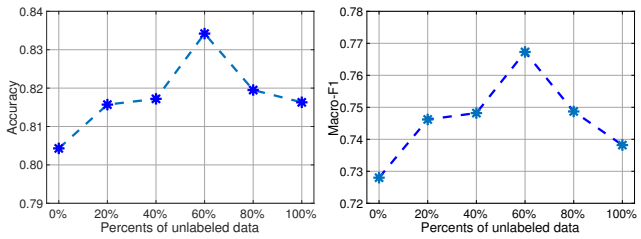

Figure 3: Accuracy and Macro-F1 on ST-1 with different percents of unlabeled training data. 0% means only using labeled training data, 100% means using all unlabeled training data.

of neural network are same with the basic model described in supervised learning part.

Figure 3 shows the performance with different percents of unlabeled data. We can see that when we use unlabeled data, both the accuracy and Macro-F1 score improve compared with purely supervised learning where unlabeled data is 0%. When unlabeled data size increases, the accuracy and Macro-F1 also increase. They achieve the highest scores when using 60% unlabeled data. After that the accuracy and Macro-F1 score slightly drop down. We speculate that this is because when there are too many unlabeled data, the layerwise perturbation-based adversarial training method will focus more on resisting perturbations and adversarial loss on unlabeled examples will overwhelmed supervised loss, and hence the model prioritizes being robust to perturbations rather than correctly predicting hard drives healthy degrees.

5 Related Work

A series of models have been proposed to predict the impending failures of hard drives. A nonparametric model was proposed by [10] to detect the anomalous SMART values of hard drives. [8] designed a Bayesian classifier to predict the hard drive failures. In order to capture the temporal features embedded in the SMART statistics, some researchers resorted to the time series analysis methods. [30] employed the hidden Markov model to predict the imminent failures. [5] used the Bayes-based model to detect the changepoint of SMART statistics, and then compacted the time sequence representation by using an exponential smoothing technique. Besides, the down sampling technology was implemented in [5] to solve the unbalanced dataset problem. All the above works merely attempted to predict the failure, and cannot provide further information about the status of hard drives. [29] tried to assess the health status of hard drives based on their residual lives. They designed a simple recurrent neural network to classify hard drives as different groups. According to the health status, different protective measures can be taken to improve the storage system reliability. However, the accuracy of their prediction is relatively poor due to the simpleness of their model.

Adversarial training is a strong regularization method which was originally introduced in [7, 24]. Their works show that several deep learning neural networks are vulnerable to a very small perturbation in the direction that the model's assignment of labels to an unseen class in the most adversarial(sensitive) way. This small perturbation is called adversarial perturbation, which has been shown has a better performance [16, 25] than dropout [22] and models trained with random perturbations [7, 16] like adding Gaussian noise [3]. They also find that training the models to be robust to against adversarial perturbations is effective to improve performance on testing dataset. [16] proposed the virtual adversarial method which expands adversarial training method into semi-supervised learning areas, which improve robustness of the models by utilizing the model's posterior distribution against local perturbations around each input data point. They also demonstrate the effects in image classification tasks [11] and text domain [14]. [16] used virtual adversarial training method to text domain by applying perturbations to the word embedding. [19] designed methods to improve robustness and performance of deep neural networks such as VGGnet [21], InceptionV3 [23] by perturbing intermediate layers. However, above methods either can only be used in supervised learning domains or can only add adversarial perturbations to inputs in semi-supervised learning way.

6 Conclusion and Future Work

In this paper, we propose a layerwise perturbation-based adversarial training method with an application on hard drive health degree predictions. In contrast to traditional methods which usually predict hard drive health in a binary fashion, the proposed method focuses on different health degrees. Additionally, it can not only add adversarial perturbations to the input, but also to the intermediate layers or all layers, which improves the generalization and performance of NNs. Our method can also utilize unlabeled data to further improve performance on predictions. Extensive experiments on supervised and semi-supervised tasks demonstrate the superiority of our proposed methods.

In the future, we will explore improving performance on more healthy degree levels which can give better instructions for technicians and users to take different actions. Also, our method calculates adversarial perturbations each time based on current mini-batch, which is time consuming. We will further optimize the method to reduce computation time and extend it into other domains like text classification or self-driving cars, which includes large-scale unlabeled or hard to labeled data.

References

[1] M. ABADI, A. AGARWAL, P. BARHAM, E. BREVDO, Z. CHEN, C. CITRO, G. S. CORRADO, A. DAVIS, J. DEAN, M. DEVIN, ET AL., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, arXiv preprint arXiv:1603.04467, (2016).

[2] B. ALLEN, *Monitoring hard disks with smart*, Linux Journal, (2004), pp. 74–77.

[3] P. BACHMAN, O. ALSHARIF, AND D. PRECUP, *Learning with pseudo-ensembles*, in Advances in Neural Information Processing Systems, 2014, pp. 3365–3373.

[4] Y. BENGIO, *Rmsprop and equilibrated adaptive learning rates for non-convex optimization*.

[5] M. M. BOTEZATU, I. GIURGIU, J. BOGOJESKA, AND D. WIESMANN, *Predicting disk replacement towards reliable data centers*, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 39–48.

[6] G. H. GOLUB AND H. A. VAN DER VORST, *Eigenvalue computation in the 20th century*, Journal of Computational and Applied Mathematics, 123 (2000), pp. 35–65.

[7] I. J. GOODFELLOW, J. SHLENS, AND C. SZEGEDY, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572, (2014).

[8] G. HAMERLY, C. ELKAN, ET AL., *Bayesian approaches to failure prediction for disk drives*, in Proceedings of the Eighteenth International Conference on Machine Learning, 2001, pp. 202–209.

[9] G. F. HUGHES, J. F. MURRAY, K. KREUTZ-DELGADO, AND C. ELKAN, *Improved disk-drive failure warnings*, IEEE Transactions on Reliability, 51 (2002), pp. 350–357.

[10] G. F. HUGHES, J. F. MURRAY, K. KREUTZ-DELGADO, AND C. ELKAN, *Improved disk-drive failure warnings*, IEEE Transactions on Reliability, 51 (2002), pp. 350–357.

[11] A. KURAKIN, I. GOODFELLOW, AND S. BENGIO, *Adversarial machine learning at scale*, arXiv preprint arXiv:1611.01236, (2016).

[12] 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 Reliable Distributed Systems (SRDS), 2016 IEEE 35th Symposium on, IEEE, 2016, pp. 71–80.

[13] J. MACQUEEN ET AL., *Some methods for classification and analysis of multivariate observations*, in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, Oakland, CA, USA., 1967, pp. 281–297.

[14] T. MIYATO, A. M. DAI, AND I. GOODFELLOW, *Adversarial training methods for semi-supervised text classification*, arXiv preprint arXiv:1605.07725, (2016).

[15] T. MIYATO, S.-I. MAEDA, M. KOYAMA, AND S. ISHII, *Virtual adversarial training: a regularization method for supervised and semi-supervised learning*, arXiv preprint arXiv:1704.03976, (2017).

[16] T. MIYATO, S.-I. MAEDA, M. KOYAMA, K. NAKAE, AND S. ISHII, *Distributional smoothing with virtual adversarial training*, arXiv preprint arXiv:1507.00677, (2015).

[17] 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 (2005), pp. 783–816.

[18] M. S. ROTHBERG, *Disk drive for receiving setup data in a self monitoring analysis and reporting technology (smart) command*, May 17 2005. US Patent 6,895,500.

[19] S. SANKARANARAYANAN, A. JAIN, R. CHELLAPPA, AND S. N. LIM, *Regularizing deep networks using efficient layerwise adversarial training*, arXiv preprint arXiv:1705.07819, (2017).

[20] B. SCHROEDER AND G. A. GIBSON, *Disk failures in the real world: What does an mttf of 1, 000, 000 hours mean to you?*, in FAST, vol. 7, 2007, pp. 1–16.

[21] K. SIMONYAN AND A. ZISSERMAN, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556, (2014).

[22] N. SRIVASTAVA, G. HINTON, A. KRIZHEVSKY, I. SUTSKEVER, AND R. SALAKHUTDINOV, *Dropout: A simple way to prevent neural networks from overfitting*, J. Mach. Learn. Res., 15 (2014), pp. 1929–1958, http://dl.acm.org/citation.cfm?id=2627435.2670313.

[23] C. SZEGEDY, V. VANHOUCKE, S. IOFFE, J. SHLENS, AND Z. WOJNA, *Rethinking the inception architecture for computer vision*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[24] C. SZEGEDY, W. ZAREMBA, I. SUTSKEVER, J. BRUNA, D. ERHAN, I. GOODFELLOW, AND R. FERGUS, *Intriguing properties of neural networks*, arXiv preprint arXiv:1312.6199, (2013).

[25] Q. WANG, W. GUO, K. ZHANG, A. G. ORORBIA II, X. XING, X. LIU, AND C. L. GILES, *Adversary resistant deep neural networks with an application to malware detection*, in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 1145–1153.

[26] Y. WANG, Q. MIAO, E. W. MA, K.-L. TSUI, AND M. G. PECHT, *Online anomaly detection for hard disk drives based on mahalanobis distance*, IEEE Transactions on Reliability, 62 (2013), pp. 136–145.

[27] Y. WANG, Q. MIAO, AND M. PECHT, *Health monitoring of hard disk drive based on mahalanobis distance*, in Prognostics and System Health Management Conference (PHM-Shenzhen), 2011, IEEE, 2011, pp. 1–8.

[28] D. WARDE-FARLEY AND I. GOODFELLOW, *11 adversarial perturbations of deep neural networks*, Perturbations, Optimization, and Statistics, (2016), p. 311.

[29] 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 Transactions on Computers, 65 (2016), pp. 3502–3508.

[30] Y. ZHAO, X. LIU, S. GAN, AND W. ZHENG, *Predicting disk failures with hmm-and hsmm-based approaches.*, in Proceedings of the 10th Industrial Conference on Advances in Data Mining, Springer, 2010, pp. 390–404.

[31] B. ZHU, G. WANG, X. LIU, D. HU, S. LIN, AND J. MA, *Proactive drive failure prediction for large scale storage systems*, in Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on, IEEE, 2013, pp. 1–5.