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

Anonymous Author(s)

Abstract

With the development of cloud computing and big data, the reliability of data storage systems becomes increasingly important. Previous researchers have shown that machine learning algorithms based on SMART attributes are effective methods to predict hard drive failures. In this paper, we use SMART attributes to predict hard drive health degrees which are helpful for taking different fault tolerant actions in advance. Given the highly imbalanced SMART datasets, it is a nontrivial work to predict the health degree precisely. The proposed model would encounter overfitting and biased fitting problems if it is trained by the traditional methods. In order to resolve this problem, we propose two strategies to better utilize imbalanced data and improve performance. Firstly, we design a layerwise perturbation-based adversarial training method which can add perturbations to any layers of a neural network to improve the generalization of the network. Secondly, we extend the training method to the semi-supervised settings. Then, it is possible to utilize unlabeled data that have a potential of failure to further improve the performance of the model. Our extensive experiments on two real-world datasets demonstrate the superiority of the proposed schemes for both supervised and semi-supervised classification. The model trained by the proposed method can correctly predict the hard drive health degrees 5 and 15 days in advance.

1 Introduction

Nowadays, increasing numbers of industrial and academic institutes rely on data centers to store and process their data. The crash of data centers may incur tremendous loss or even catastrophic consequences. The reliability and the availability of data centers are of the utmost importance to data center administrators. However, the complex architecture and functionality of data centers lead to a serious problem of IT equipment failures, among which hard drives are the most frequently failing components [Sankar *et al.*, 2013; Botezatu *et al.*, 2016]. Hence, it is in high demand to take measures to handle the hard drive failure issue.

The self-monitoring, analysis and reporting technology (SMART) [Allen, 2004] has been implemented in almost all hard drives to monitor and analyze the internal attributes of hard drives. The previous researches demonstrate that the impending hard drives failure manifests itself through SMART statistics [Backblaze, 2016]. It is feasible to predict the impending failures by using SMART statistics. To improve failure prediction performance, many efforts have been made based on SMART attributes, including analyzing the failure behaviors of hard drives [Hughes *et al.*, 2002; Botezatu *et al.*, 2016], and designing machine learning algorithms for predicting hard drive failures [Murray *et al.*, 2005; Zhu *et al.*, 2013; Mahdisoltani *et al.*, 2017]. Most of these works focused on the proactive failure prediction, which forecasts hard drive failures in advance and gives a binary result identifying the hard drive as healthy or faulted. However, hard drives usually fail gradually rather than abruptly. To harness the potential of gradual change, it is necessary to develop a health prediction model which can predict the heath status of hard drives rather than merely providing a simple binary result.

Predicting the health status of hard drives is not a trivial task. Most hard drives undergo a deterioration process before they finally fail. The SMART statistics begin to accumulate deviation from the normal state days before the final failure. Hence, we need to extract the long-term temporal dependency in the SMART statistics to make an accurate prediction of the hard drive status. The most stubborn problem in the prediction of health status is that the data to be used in the model is highly imbalanced. Although hard drive failures occur frequently in data centers, the failure records are much fewer than the healthy records, merely accounting for less than 3% of the total records [Backblaze, 2016]. Due to the overfitting and biased fitting problems of most statistical and machine learning algorithms, the prediction model trained by the imbalanced SAMRT data is easy to be biased fitted to the healthy records and over fitted to the failure records, which results in a poor predictive performance. Some novel methods are desirable to tackle imbalance issue and establish a high quality prediction model by using the SMART statistics.

In order to extract the long-term temporal dependency embedded in the SMART statistics, we build a deep neural network based on the long short-term memory unit (LSTM) [Hochreiter and Schmidhuber, 1997] which specializes in processing sequential data [Graves *et al.*, 2013]. Nonetheless, the deep neural networks (DNNs) are notorious for the overfitting and biased fitting problems on the highly imbalanced datasets like the hard drive records used in this work. To solve this problem, we propose two strategies in this paper.

Firstly, we introduce the adversarial training strategy. The adversarial training [Szegedy *et al.*, 2013; Goodfellow *et al.*, 2015] is a strong regularization method that injects the perturbations affecting the neural network's inference in the most sensitive way during the training phase. The traditional adversarial training injects adversarial perturbations into the inputs to force the networks to learn a better distribution of the training data and avoid the overfitting problem [Miyato *et al.*, 2016; Kurakin *et al.*, 2017]. However, only injecting perturbations into the inputs may restrict the effectiveness of the adversarial training. In this paper, we enhance the adversarial training by enabling layerwise perturbation where the adversarial perturbation can be injected into any layer of the neural network rather than just the input layer.

Another more effective approach to mitigating the overfitting and biased fitting is to increase the number of positive samples, i.e., the failure records. It is obvious that the anomaly of SMART statistics is evident when hard drives are close to failure. Therefore, we can label the records close to failure as the failure records directly. But for the records dozens of days ahead of the final failure, they may be less informative for the failure or even demonstrate the same features as the healthy records do. It is inappropriate to label these records as failure records arbitrarily. Labeling these records is a time consuming task and relies on the expert knowledge, which makes it unfeasible in reality. Hence, in traditional supervised training, these records are discarded in spite of the high probability that they show fault features. To fully harness these potential failure records, we extend the proposed adversarial training into the semi-supervised setting where the records far before the final failure are regarded as unlabeled data during the training phase.

Based on the above two strategies, we propose a **L**ayerwise **P**erturbation-based **A**dversarial **T**raining (LPAT) method to train our hard drive status prediction model. Note that although the LPAT is designed for predicting hard drive health status in this work, it can be used in other similar anomaly detection problems to mitigate the overfitting and biased fitting. The main contributions are summarized as follows:

- We design a novel LPAT method for hard drive status prediction tasks. Instead of only adding perturbation to inputs of neural networks, LPAT can flexibly add perturbation to a specific layer or all layers to better address the overfitting and biased fitting problem. Besides, it can utilize unlabeled data to further improve the performance of the prediction model.

- To the best of our knowledge, it is the first work toward designing a layerwise perturbation-based adversarial training with the semi-supervised setting for deep learning models. We use this model to predict hard drive status in both supervised and semi-supervised settings.

- Thorough experiments[1] on two datasets demonstrate that LPAT can improve the prediction performance in both supervised and semi-supervised settings. Specifically, the model can predict hard drive health status 5 days and 15 days before failure by using sequential SMART statistics, which is more useful in practice than only predicting healthy or failed.

2 Problem Definition

In this work, we aim to build a model trained by LPAT to predict the status of hard drives based on their SMART statistics. The training dataset is denoted as $D = \{(x^{(1)}, y^{(1)}), ..., (x^{(N)}, y^{(N)})\}$, where $x^{(i)} \in R^{k \times n}$ is the i^{th} training sample consisting of continuous SMART records from the day t_i to the day t_{i+k}. Each day has a feature vector of n SMART attributes, and y denotes health degrees of the hard drive. We predict hard drives by three different health degrees[2], i.e., $y \in \{0, 1, 2\}$. '0' represents a "red alert" which means the residual life of the hard drive is less than 5 days; '1' represents that the drive is "going to fail" in 15 days; '2' means "healthy". Our goal is to learn a function $f : X \rightarrow \{0, 1, 2\}$ that minimizes the empirical loss $\hat{\mathcal{L}}(f(X); Y)$ on the training dataset. Intuitively, we try to train a model that correctly predicts the health degree of a hard drive. $\hat{\mathcal{L}}(f(X); Y)$ can be formulated as:

$$\hat{\mathcal{L}}(f(X); Y) = -\frac{1}{N} \sum_{i=1}^{N} log\, p(y^{(i)} | x^{(i)}; \Theta), \quad (1)$$

where Θ is the model parameters learned during the training.

3 Methodology

3.1 LSTM Based Neural Network

The proposed prediction model includes two dense (fully connected) layers, followed by a LSTM layer and a dense layer. LSTM is the basic building block used to extract the long-term temporal dependency in sequential SMART statistics. Each component in the LSTM layer is formulated as:

$$\hat{\alpha}_t = W_\alpha x_t + U_\alpha h_{t-1} + b_\alpha, \quad (2)$$

$$c_t = i_t \odot j_t + f_t \odot c_{t-1}, \quad (3)$$

$$h_t = o_t \odot tanh(c_t), \quad (4)$$

where $\alpha \in \{i, f, o, j\}$. i_t, f_t, o_t are the *input gate*, *forget gate* and *output gate*, respectively. c_t represents the *memory cell* which is designed to counteract the problem of vanishing/exploding gradient, and thus enable extracting long-term temporal dependency. The *forget gate* is for resetting the memory cells. The *input gate* and the

[1]The code is open sourced at `https://github.com/Anymous-submission/LPAT`

[2]Note that we define the health degree intuitively in this paper, and other carefully designed definitions can be used with just a little modification.

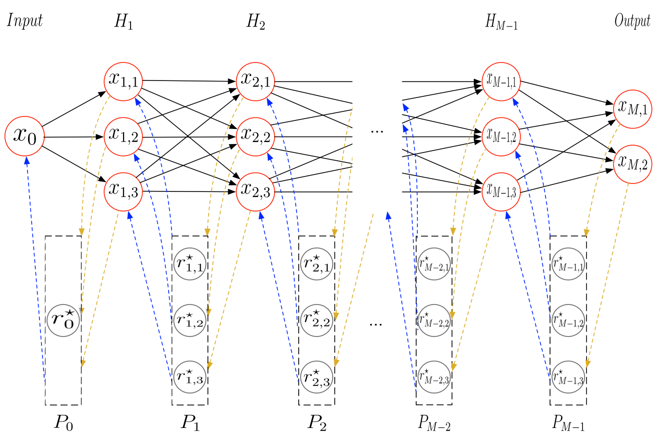


Figure 1: An overview of LPAT.

output gate are for controlling the input and the output of the memory cells. $\alpha_t = \sigma(\hat{\alpha}_t)$ when $\alpha \in \{i, f, o\}$, where σ is the sigmoid activation function. $j_t = \tanh(\hat{j}_t)$ is a proposed update to the cell state. $x_t \in R^n$ is the input from the lower dense layer at the time step t. h_t is the output of current block at the time step t. In the q LSTM units, the weight matrices $W_\alpha \in R^{q\times d}$, $U_\alpha \in R^{q\times d}$ and the bias vectors $b_\alpha \in R^q$ are the parameters to learn for $\alpha \in \{i, f, o, j\}$.

3.2 Layerwise Perturbation-Based Adversarial Training

Previous works on adversarial training concluded that training a DNN with adversarial examples acts as a regularizer and improves the robustness of the neural network on the test dataset. In order to mitigate the overfitting and biased fitting problem caused by the highly imbalanced data, we design LPAT to train the prediction model. Instead of only injecting perturbation into inputs, LPAT generates adversarial samples at the time series inputs and the intermediate layers. Fig. 1 illustrates how LPAT works when training the model.

In Fig. 1, for a model consisting of $M-1$ hidden layers, $m = 0$ is the input layer, and $\hat{x}_m$ is the output of the m^{th} layer. Each layer has a gradient accumulation layer P_m serving two functions: (1) it temporarily stores the back propagation gradients on the output of the m^{th} layer, which is denoted by the yellow line; and (2) it computes the adversarial perturbations $r^\star_i$ for the m^{th} layer and adds the perturbation to $\hat{x}_m$ as denoted by the blue line. Then, the NN performs feedforward process again to compute its new output. The training process can be formed as a min-max problem. The adversarial samples apply the worst perturbation to maximize the error of the model, while the model tries to be robust to such perturbations through minimizing the error caused by the adversary. The min-max problem for the m^{th} layer can be formulated as an additional cost function:

$$\min_{l} \max_{r_m, \|r_m\|\le\epsilon} l(p(y^{(i)}|\hat{x}^{(i)}_m + r_m, \Theta), q(y^{(i)})), \tag{5}$$

where ϵ is the magnitude of the perturbation, l is the non-negative cross entropy between two distributions representing the prediction error caused by the perturbation, and $q(y^{(i)})$ is a one-hot vector of the corresponding label $y^{(i)}$. The additional cost function can be simplified as:

$$-log\ p(y^{(i)}|\hat{x}^{(i)}_m + r_m; \Theta). \tag{6}$$

Then,

$$r^\star_m = arg \min_{r_m, \|r_m\|\le\epsilon} log\ p(y^{(i)}|\hat{x}^{(i)} + r_m; \hat{\Theta}), \tag{7}$$

where $\hat{\Theta}$ is a constant set to the current parameters of the model. In general, the exact calculation for r_m is intractable for most DNNs. We follow the method proposed in [Goodfellow *et al.*, 2015] to linearly approximate r_m with the L_2 norm constraints as:

$$r^\star_m \approx -\epsilon \frac{g}{\|g\|_2}, \tag{8}$$

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

3.3 Extension of Semi-Supervised Setting

We extend the proposed adversarial training into the semi-supervised setting to utilize the records relatively far before the final failure. Due to the difficulty of labeling these records, we regard them as unlabeled samples during the training. Then, the adversarial perturbation $r^\star$ is incalculable by using Eq. (8) without the label $y^{(i)}$. To depict the prediction error, we introduce the Kullback–Leibler (KL) divergence [Kullback and Leibler, 1951] which measures the divergence between the current output distribution $p(\cdot|x); \hat{\Theta})$ and the perturbed output distribution $p(\cdot|x + r; \Theta)$. The layerwise adversarial perturbation for the m^{th} layer can be calculated as:

$$r^\star_m = arg \max_{r_m, \|r_m\|\le\epsilon} KL_m, \tag{9}$$

where

$$KL_m = KL[p(\cdot|\hat{x}_m; \hat{\Theta}) \,||\, p(\cdot|\hat{x}_m + r_m; \hat{\Theta})]. \tag{10}$$

$p(\cdot|\hat{x}_m; \hat{\Theta})$ is differential with $\hat{\Theta}$ and $\hat{x}_m$. 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 $r_m = 0$. Hence, we approximate it with the second-order Taylor series:

$$KL_m \approx \frac{1}{2} {r_m}^T H(\hat{x}_m, \hat{\Theta}) r_m, \tag{11}$$

where $H(\hat{x}_m, \hat{\Theta})$ is the Hessian matrix. Then, $r^\star_m$ is the first dominant eigenvector $u(\hat{x}_m, \hat{\Theta})$ of $H(\hat{x}_m, \hat{\Theta})$ with magnitude ϵ, $r^\star_m = \epsilon \overline{u(\hat{x}_m, \hat{\Theta})}$, where $\overline{(\cdot)}$ represents the unit vector in the direction of $(\cdot)$. Given a randomly sampled unit vector e, according to the power iteration method [Golub *et al.*, 2000], $u(\hat{x}_m, \hat{\Theta})$ is the convergence of the iteration $e \leftarrow \overline{He}$, and He can be approximated as follow:

$$He \approx \frac{\nabla_r KL_m|_{r=\xi e} - \nabla_r KL_m|_{r=0}}{\xi}, \tag{12}$$

where $\xi \neq 0$ is an arbitrary small value. Here, we approximate the convergence by the one-time power iteration [Golub *et al.*, 2000]. Then, the approximation of $r^\star_m$ is:

$$r^\star_m \approx \epsilon \frac{g}{\|g\|_2}, \tag{13}$$

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 ad-
    versarial perturbations $\{r_i^\star\}_{i=0}^{P-1}$ initialized with zero. All
    gradient accumulation layers inactive at the initial train-
    ing step $s = 0$. $\epsilon_i$ denotes perturbation size.
1: for $s = 0, 1, ..., B - 1$ do
2:     Perform forward pass to compute the output of $NN$
3:     Perform back pass using loss function, each accumu-
       lation layer $P_i$ temporarily stores the gradients back-
       propagated to the network layer $i$
4:     Compute and store layerwise adversarial perturbation
       $r_i^\star$ for each layer $i$ using Eq. $(13)$
5:     Perform forward pass. Let $X_i$ be the output to the net-
       work layer $i$, we have: $X_i = X_i + \epsilon_i \cdot r_i^\star$
6:     Perform back pass to update parameters
7: end for
```

where

$$g = \nabla_r KL[p(\cdot|\hat{x}_m; \hat{\Theta}) \,||\, p(\cdot|\hat{x}_m + r; \hat{\Theta})]|_{r=\xi e}. \tag{14}$$

Thus, for all the training dataset including the labeled and the unlabeled data with size N', the full loss is given by:

$$\mathcal{L}(\Theta) = \hat{\mathcal{L}}(\Theta) + \lambda \cdot \mathcal{L}_{lap}, \tag{15}$$

where $\hat{\mathcal{L}}(\Theta)$ is the negative log-likelihood for the labeled data defined by Eq. (1), and

$$L_{lap} = \frac{1}{N'} \sum_{i=1}^{N'} KL[p(\cdot|\hat{x}_m^{(i)}; \hat{\Theta}_m) \,||\, p(\cdot|\hat{x}_m^{(i)} + r_m^{\star\,(i)}; \hat{\Theta}_m)], \tag{16}$$

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

The gradients computed from mini-batch inputs aggregated through intermediate layers are used to calculate perturbations for the current mini-batch. 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 only inject perturbations into the input. $P = M - 1$ indicates that all layers of the model are perturbed. The training procedure is summarized in Algorithm 1. Parameter ϵ_i can be set to a fixed value for all layers or set differently for different layers. During training, we add the gradient accumulation layer P_i after each neural network layer. During testing, all the gradient accumulation layers are removed from the model.

4 Experiments

In this section, we first introduce the datasets. Then, we present the results along with the analysis.

4.1 Data Preparation

Our evaluation and analysis are based on the Backblaze dataset[3]. At the end of 2016, it recorded 73,653 spinning hard drives. Each entity in the dataset includes the serial number of the hard drive, the SMART statistics, the status, i.e., failure or alive, and other necessary information. In this paper, we use two models of hard drives. The first one is Seagate ST4000DM000 (ST-1). There are 33,800 healthy hard drives and 938 failed hard drives of ST-1 in the dataset. After data cleaning and aggregation [Botezatu *et al.*, 2016], there are 30,685 healthy hard drives and 758 failed hard drives. To further verify our method, we collect another small dataset from Seagate ST8000DM002 (ST-2). Table 1 lists the details.

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

Table 1: Number of Healthy (H) and Failed (F) hard drives.

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

Table 2: Summary of datasets

	Train	Valid	Test	Unlabeled
ST-1	17137	4285	5356	10326
ST-2	755	189	236	-

Apparently, the dataset is extremely imbalanced. The prediction model encounters serious overfitting and biased fitting problems if the dataset is used directly, even with the help of LPAT. Hence, we select the representative subset of healthy hard drives to construct the dataset for training and testing our model. The K-means clustering algorithm [Kanungo *et al.*, 2002] is used to cluster the healthy hard drives into 10 clusters based on every day's records of SMART attributes. Then, for each cluster, the top 30% samples closest to the centroid are selected as the representative subset of healthy hard drives. Finally, we follow the previous works [Murray *et al.*, 2005; Botezatu *et al.*, 2016; Xu *et al.*, 2016] to select distinctive features and rescale the values of SMART statistics. Similar to the definitions used in [Li *et al.*, 2016; Xu *et al.*, 2016], we predict the hard drives based on their residual life. Label '0' is "red alert", which means the residual life is less than 5 days. Label '1' means "going to fail" in 15 days. Label '2' represents "healthy". For the failed hard drives with more than 15-day residual life, we regard them as unlabeled data.

We split each dataset into three subsets, i.e., the training set, the validation set, and testing set. For the supervised setting, we use 80% of all labeled data as the training set and the rest as the testing set. We use 20% of the training set as validation set for tuning hyper-parameters. For the semi-supervised setting, we add unlabeled data into the training set and keep the validation set and the testing set unchanged. The information about the datasets is summarized in Table 2.

4.2 Baselines and Metrics

To demonstrate the performance improvement of the proposed approach, we compare it with decision tree (DT), regularized greedy forest (RGF) [Botezatu *et al.*, 2016], and recurrent neural network (RNN) [Xu *et al.*, 2016]. Besides, the proposed LSTM based neural network trained without adversarial training methods (Basic), the prediction model trained by the adversarial training method proposed in [Goodfellow *et al.*, 2015] (Basic+AT), and the prediction model trained by the virtual adversarial training method proposed in [Miyato *et al.*, 2017] (Basic+VAT) are also used as baselines. For these baselines, we tune parameters on the validation set and report their results on the testing set.

For the proposed LPAT, we design three variations: (1)

Table 3: Overall results in supervised setting

	ST-1				ST-2			
	Accuracy	Precision	Recall	Macro-F1	Accuracy	Precision	Recall	Macro-F1
DT	82.4	73.6	74.7	74.1	82.2	74.0	72.6	73.1
RGF	74.4	68.4	54.8	56.1	92.0	87.0	83.3	84.9
RNN	84.3	80.1	79.8	79.9	87.7	83.9	78.6	80.9
basic	82.1	80.3	79.9	80.2	91.5	86.0	84.5	84.5
basic+AT	86.8	81.8	81.3	81.5	92.0	86.2	85.5	85.8
basic+VAT	87.0	82.6	82.1	82.3	92.4	87.9	86.6	86.6
LPAT+Bottom	88.9	84.0	85.0	84.5	94.1	90.5	91.2	90.7
LPAT+Top	90.0	85.5	86.1	85.8	**96.6**	**94.4**	**93.1**	**93.9**
LPAT+All	**90.2**	**85.8**	**86.7**	**86.2**	96.2	94.2	92.2	93.1

Table 4: Detailed results in supervised setting

	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	72.9	76.9	73.5	83.2	72.2	81.9	73.1	93.3	60.7	77.7	66.7	84.8
basic+VAT	73.5	80.6	70.9	85.9	72.2	83.1	72.7	95.2	85.7	74.0	78.7	83.3
LPAT+Bottom	72.2	81.6	74.3	86.2	73.2	83.4	80.6	94.0	89.2	87.0	84.7	90.4
LPAT+Top	**74.5**	83.1	72.1	85.3	73.7	84.2	**89.3**	94.3	**89.3**	**92.6**	**89.3**	**93.5**
LPAT+All	72.5	**83.7**	**80.1**	**87.0**	**76.1**	**85.3**	88.9	**96.1**	85.7	90.7	87.3	93.3

LPAT-Bottom where the adversarial perturbations are added to the bottom two layers, (2) LPAT-Top where the adversarial perturbations are added to the LSTM layer and the top dense layer, and (3) LPAT-All where the adversarial perturbations are added to the all layers.

We measure the overall accuracy, and each class's precision, recall, and Macro-F1 score. For the basic prediction model, we set the two bottom dense layers with 128 units and the activation as None. The LSTM layer has 200 units. The final dense layer includes 3 units as we have 3 classes. The time sequence window for each sample is set to 20. We use RMSProp optimizer [Dauphin *et al.*, 2015] with learning rate 0.001. The mini-batch size is 128. The epsilon ϵ is empirically set in $[0, 50]$. The hyperparameter λ is set between $[0, 5]$. The training epochs are set to 210.

4.3 Supervised Setting

We first conduct experiments in the supervised setting where all the training data are labeled. As can be seen from Table 3, the proposed LPAT method achieves the best results on both datasets. It improves accuracy to 3.2% and Macro-F1 to 3.9% on 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 training more robust and brings better performance on the testing set. It also improves accuracy to 4.2% and Macro-F1 to 7.3% on ST-2 over the other baseline methods. As it is known a small dataset like ST-2 is much easier to cause the overfitting problem. Our results on ST-2 show the regularization and generalization ability of layer-wise adversarial learning in preventing overfitting.

RNN performs better than RGF on ST-1 while slightly worse than RGF on ST-2. ST-1 and ST-2 datasets are collected on different hard drive models whose SMART attributes are also different. RGF which automatically selects SMART attributes may perform better in feature obvious dataset [Botezatu *et al.*, 2016]. Models with the adversarial learning achieve better results than the basic one without the adversarial training. VAT slightly improves the performance than AT on the both datasets.

We further investigate the impact of different perturbation patterns. For ST-1, LPAT-Bottom works better than only adding perturbation to the input layer while worse than

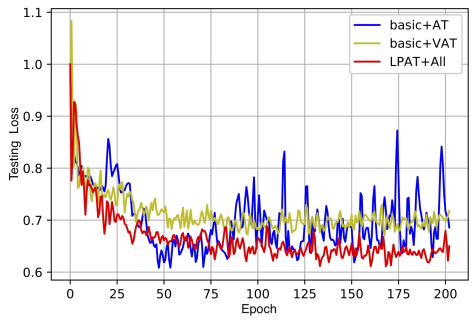


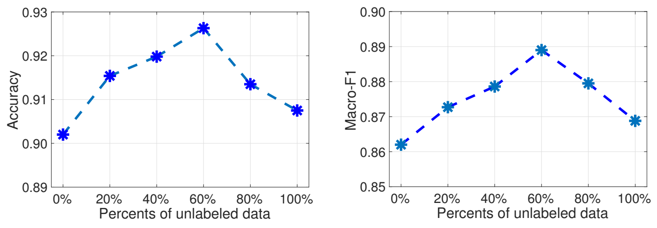
Figure 2: Testing loss of AT, VAT and LPAT+All. $\lambda = 1$ for all methods. $\epsilon = 30$ for AT and VAT. $\epsilon = 20$ on all layers for LPAT+All. The optimal value of ϵ differs between different methods. But the value of ϵ in $[0, 50]$ usually performs well for above methods and provides a fair comparison.

LPAT-Top. Among the three perturbation patterns, LPAT-All achieves the best results, which could be partly attributed to the LSTM layer. Adding perturbations to the bottom layers before the LSTM layer will make perturbations wrapped in the LSTM layer and show less effect on the loss function. It undermines the effectiveness of the adversary perturbations. Thus, LPAT-Bottom is less effective than adding perturbations after the LSTM layer. For ST-2, LPAT-Top achieves the best performance on accuracy and Macro-F1 score; its performance is only marginally better than LPAT-All. Based on these results, we speculate that mainly adding perturbations to top or all layers can have better regularization ability.

Table 4 details the results on each class. Note that the size of the training samples of class '1' is two times larger than that of class '0' on both datasets. Precision represents the ability for a classifier to capture hard drives fails in 5 and 15 days precisely. Recall represents the failure detection rate, i.e. the fraction of failed drives that are correctly classified as failed. The proposed methods outperform basic+AT and basic+VAT in terms of precision and recall on the both datasets. In addition, LPAT-All improves Macro-F1 to 3.9% on class '0' and 2.2% on class '1' than basic+VAT for ST-1, LPAT-Top improves Macro-F1 to 10.6% on class '0' and 10.2% on class '1' for ST-2. These results demonstrate that adding perturbations to different layers improves the performance of the classifier. Based on the observation, our methods can give operators and users more flexibility to take different actions 5 and 15 days before the final failure.

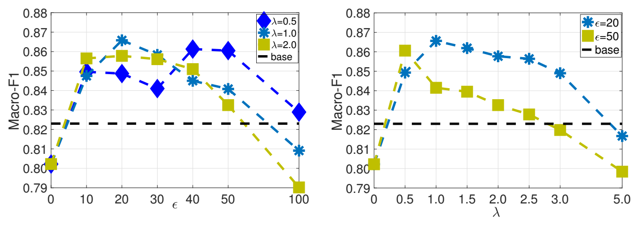
4.4 Semi-Supervised Setting

In this part, we focus on the ST-1 dataset which includes unlabeled data. Fig. 2 shows the loss on testing set when trained using all unlabeled and labeled data for AT, VAT, and LPAT+All. It shows that VAT and LPAT+All which utilize the unlabeled data produce lower testing losses. The testing loss of AT becomes unstable, a sign of overfitting, with the increase of the epochs. LPAT+All generates the lowest testing loss and keeps the tendency over the training. Thanks to the resistance to the adversarial perturbations at every layer, LPAT+All performs better than VAT which merely resists to



(a) Accuracy (b) Macro-F1

Figure 3: Accuracy and Macro-F1 on ST-1 with different percents of unlabeled training data.



(a) Effect of ϵ (b) Effect of λ

Figure 4: Effect of ϵ and λ for supervised learning on ST-1 dataset. 'base' represents the best result (basic+VAT) of baselines.

the perturbations at the input.

Furthermore, we analyze the influence of different percents of unlabeled data. We conduct the experiments with different percents of unlabeled training samples $N_{ul} = \{0\%, 20\%, 40\%, 60\%, 80\%, 100\%\}$ and observe the effect of N_{ul} on the overall accuracy.

From Fig. 3, we can see that when using unlabeled data, both the accuracy and the Macro-F1 score are improved compared with the purely supervised setting When the percent of unlabeled data increases, the accuracy and the Macro-F1 firstly increase. They achieve the highest scores around 60% unlabeled data. After that, the accuracy and the Macro-F1 score slightly decrease. We speculate that this is because when there are too many unlabeled data, the layer-wise perturbation-based adversarial training method will focus more on resisting the perturbations and the adversarial loss on unlabeled examples rather than the supervised loss, and hence the model prioritizes being robust to perturbations rather than correctly predicting hard drives health degrees.

4.5 Parameter Analysis

LPAT mainly involves two hyperparameters: ϵ and λ, where ϵ controls the intensity of perturbation added to different layers and λ controls the trade-off between the supervised loss and the adversarial loss. Fig. 4(a) shows the effects of ϵ with different values of $\lambda \in \{0.5, 1.0, 2.0\}$. LPAT achieves the best result when $\epsilon = 20$ and $\lambda = 1$. We can find that the performance (i.e., Macro-F1) is not very sensitive to the parameter ϵ. When $\epsilon \in [10, 50]$, it typically can improve the neural networks performance. But when the value of ϵ is extraordinarily large, the performance would drop seriously. Fig. 4(b) shows the effects of λ with different values of $\epsilon \in \{20, 50\}$. When $\lambda \in [0.5, 2.5]$, the proposed method generates better results than 'base'.

5 Related Work

A series of models have been proposed to predict the impending failures of hard drives. [Hamerly *et al.*, 2001] 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. [Zhao *et al.*, 2010] employed the hidden Markov model to predict the imminent failures. [Botezatu *et al.*, 2016] 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 [Botezatu *et al.*, 2016] 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. [Xu *et al.*, 2016] tried to predict 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. However, the accuracy of their prediction is relatively poor due to the simpleness of their model.

Adversarial training is a strong regularization method introduced in [Goodfellow *et al.*, 2015; Szegedy *et al.*, 2013]. Their works show that several 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 sensitive way. This small perturbation is called adversarial perturbation, which has demonstrated a better performance [Miyato *et al.*, 2015; Wang *et al.*, 2017] than dropout [Srivastava *et al.*, 2014] and random perturbations [Goodfellow *et al.*, 2015; Miyato *et al.*, 2015] like Gaussian noise. [Miyato *et al.*, 2015] proposed the virtual adversarial method which expands adversarial training method into semi-supervised settings, which improves robustness of 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 [Kurakin *et al.*, 2017] and text domain [Miyato *et al.*, 2016]. [Sankaranarayanan *et al.*, 2017] designed methods to improve robustness and performance of DNNs such as VGGnet [Simonyan and Zisserman, 2015] by perturbing intermediate layers. However, above methods either are used in supervised settings or add adversarial perturbations only to inputs in semi-supervised settings.

6 Conclusion

We propose a layerwise perturbation-based adversarial training method with an application on hard drive health degree prediction. Differing from traditional methods which usually predict hard drive health in a binary fashion, the proposed method focuses on different health degrees. Additionally, the proposed LPAT 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 the model. Besides, it utilize unlabeled data to further improve the performance. Extensive experiments demonstrate the superiority of our proposed methods.

References

[Allen, 2004] Bruce Allen. Monitoring hard disks with smart. *Linux Journal*, (117):74–77, 2004.

[Backblaze, 2016] Backblaze. Hard drive reliability review for 2015. *https://www.backblaze.com/blog/hard-drive-reliability-q4-2015/*, 2016.

[Botezatu *et al.*, 2016] Mirela Madalina Botezatu, Ioana Giurgiu, Jasmina Bogojeska, and Dorothea Wiesmann. Predicting disk replacement towards reliable data centers. In *KDD*, pages 39–48. ACM, 2016.

[Dauphin *et al.*, 2015] Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *NIPS*, pages 1504–1512, 2015.

[Golub *et al.*, 2000] Golub, Gene H, and Henk A Van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1):35–65, 2000.

[Goodfellow *et al.*, 2015] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *In ICLR*, 2015.

[Graves *et al.*, 2013] A. Graves, A. r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE ICASSP*, pages 6645–6649, 2013.

[Hamerly *et al.*, 2001] Greg Hamerly, Charles Elkan, et al. Bayesian approaches to failure prediction for disk drives. In *ICML*, pages 202–209, 2001.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Hughes *et al.*, 2002] Gordon F Hughes, Joseph F Murray, Kenneth Kreutz-Delgado, and Charles Elkan. Improved disk-drive failure warnings. *IEEE Transactions on Reliability*, 51(3):350–357, 2002.

[Kanungo *et al.*, 2002] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE TPAMI*, 24(7):881–892, 2002.

[Kullback and Leibler, 1951] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[Kurakin *et al.*, 2017] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *In ICLR*, 2017.

[Li *et al.*, 2016] Jing Li, Rebecca J Stones, Gang Wang, Zhongwei Li, Xiaoguang Liu, and Kang Xiao. Being accurate is not enough: New metrics for disk failure prediction. In *SRDS*, pages 71–80. IEEE, 2016.

[Mahdisoltani *et al.*, 2017] Farzaneh Mahdisoltani, Ioan Stefanovici, and Bianca Schroeder. Proactive error prediction to improve storage system reliability. In *USENIX ATC*, pages 391–402, 2017.

[Miyato *et al.*, 2015] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.

[Miyato *et al.*, 2016] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *In ICLR*, 2016.

[Miyato *et al.*, 2017] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *In ICLR*, 2017.

[Murray *et al.*, 2005] Joseph F Murray, Gordon F Hughes, and Kenneth Kreutz-Delgado. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *J. Mach. Learn. Res.*, 6(May):783–816, 2005.

[Sankar *et al.*, 2013] Sriram Sankar, Mark Shaw, Kushagra Vaid, and Sudhanva Gurumurthi. Datacenter scale evaluation of the impact of temperature on hard disk drive failures. *ACM Transactions on Storage*, 9(2):1 – 24, 2013.

[Sankaranarayanan *et al.*, 2017] Swami Sankaranarayanan, Arpit Jain, Rama Chellappa, and Ser Nam Lim. Regularizing deep networks using efficient layerwise adversarial training. *arXiv preprint arXiv:1705.07819*, 2017.

[Simonyan and Zisserman, 2015] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *In ICLR*, 2015.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(Jan):1929–1958, 2014.

[Szegedy *et al.*, 2013] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[Wang *et al.*, 2017] Qinglong Wang, Wenbo Guo, Kaixuan Zhang, Alexander G Ororbia II, Xinyu Xing, Xue Liu, and C Lee Giles. Adversary resistant deep neural networks with an application to malware detection. In *KDD*, pages 1145–1153. ACM, 2017.

[Xu *et al.*, 2016] 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(11):3502–3508, 2016.

[Zhao *et al.*, 2010] Ying Zhao, Xiang Liu, Siqing Gan, and Weimin Zheng. Predicting disk failures with hmm- and hsmm-based approaches. In *ICDM*, pages 390–404. Springer, 2010.

[Zhu *et al.*, 2013] Bingpeng Zhu, Gang Wang, Xiaoguang Liu, Dianming Hu, Sheng Lin, and Jingwei Ma. Proactive drive failure prediction for large scale storage systems. In *MSST*, pages 1–5. IEEE, 2013.