Predicting Failures in Hard Drives with LSTM Networks

Fernando Dione S. Lima, Gabriel M. R. Amaral, Lucas G. M. Leite, Joao Paulo P. Gomes, Javam C. Machado
LSBD - Department of Science Computer
Federal University of Ceará, Fortaleza, Brazil
Email: {fernando.lima, gabriel.maia, lucas.goncalves, joao.pordeus, javam.machado}@lsbd.ufc.br

***Abstract*—Several research has been done to propose early failure detection techniques for hard disk drives in order to improve storage systems availability and avoid data loss. Failure prediction in such circumstances would allow for the reduction of downtime costs through anticipated disk replacements. Many of the techniques proposed so far mainly perform incipient failure detection thus not allowing for proper planning of such maintenance tasks. Others perform well only under a limited prediction horizon. In this work, we present a remaining useful life estimation approach for hard disk drives based on SMART parameters that is capable of predicting failures in both long and short term intervals by leveraging the capabilities of LSTM networks.**

I. Introduction

Hard disk drives are one of the main sources of failure in today's IT environments. Such failures cause services unavailability leading to a cost that raised 38% between 2010 and 2016 [1]. Also, data loss is another serious consequence of such failures. Despite the advent of solid-state drives (SSDs), the cost of their deployment still favors the use of hard-disk drives (HDDs) in cloud storage environments. The early detection of failures in HDDs would allow for the reduction of those costs and the prevention of data loss in these environments.

Today, HDDs manufacturers employ *Self-Monitoring, Analysis and Reporting Technology* (SMART) as a failure detection strategy. SMART is a system that monitors and collects data from the device (e.g. temperature, sector error count) and also that employs predictive models over this data to be able to detect anomalies that indicate a failing component. However, such models are proprietary and mainly employ threshold-based approaches leading to very low Failure Detection Rates (FDR), ranging from 3% to 10% [2].

Envisioning better failure prediction results, several techniques have been proposed based on SMART internally collected attributes, many of them limited to the detection of incipient failures [3], [4], [2], [5], [6], [7]. Others gradually quantify the state of the device with respect to failure, but achieve good accuracy only in the categorization of the last quarter of the device's life, therefore, having a short prediction horizon [8].

The present work aims to propose an approach for remaining useful life (RUL) prediction for HDDs in both long and short range intervals through their categorization in RUL intervals. To explore the long term temporal relations on SMART data, we propose to use the specific architectural variant of recurrent neural networks (RNNs) known as Long Short-Term Memory (LSTM) [9].

II. Related Work

Queiroz *et al.* [3] proposed a failure detection methodology based on the anomaly detection paradigm. In the first step, the Recursive Feature Elimination (RFE) algorithm was used to select the most relevant SMART features. With the selected features they created a baseline statistical model with SMART data of healthy HDDs. This model was fitted using a GMM. Also, they build baseline dissimilarity vectors from the healthy HDDs relative to the baseline model through the computation of log-likelihood. In the fault detection step, dissimilarity vectors are built for the test HDDs using the same procedure described for healthy HDDs. Using a sliding window approach, they estimate distributions for both dissimilarity vectors using KDE, within the window, and calculate these distributions divergences through Kullback-Leibler Divergence (KLD). If such divergence exceeds a predefined threshold, then a fault is detected. Their approach is able to achieve 92.21% FDR at 0% False Alarm Rate (FAR).

The problem of incipient failure detection was also investigated in Botezatu *et al.* [4]. As a first step, feature selection is employed to select relevant SMART attributes by detecting change points in the attributes time series. After feature selection, these time series are compacted through exponential smoothing for later construction of a model using the Regularized Greedy Forest classifier (RGF). This model achieves an accuracy of up to 98%, predicting failures with a Time In Advance (TIA) of 10-15 days. A transfer learning technique is also explored enabling the creation of classifiers for HDD models with few data, providing an enhancement of up to 50% in accuracy, if compared to a classifier built with only the data at hand.

The work presented by Chaves *et al.* [10] explores a Bayesian network for failure prediction of hard disk drives. To the best of our knowledge, this is the first work that addresses the problem of failure prediction. The proposed method starts by applying the same procedure used by [3] for feature selection. Later, they perform binning on the selected SMART data using the MDLP algorithm, except for Power on Hours (POH) attribute, which is discretized using equal width bins. After that, the Bayesian Network parameters are estimated, which take into account both POH, RUL and

SMART attributes as random variables. With this model they are able to achieve good mean and quadratic errors compared with the standard reliability based approach.

Recently, Chang Xu *et al.* [8] described a predictive model to label HDDs according with health degree levels. The health degrees used by the model splits the last month of the device into six classes according to the remaining life time, but argues that different settings can be employed. It makes use of a single layer RNN. After labeling the time series samples, a voting algorithm is employed in the outputs of the last N consecutive samples before a time point, resulting in the final estimate of the device's health level. This model achieves a FDR of 97.71% with a FAR of 0.06%.

Note that the works of Chaves *et al.* [10] and Chang Xu *et al.* [8] are the only that aim to predict the RUL of HDDs. By analyzing both works, we can state that Chang Xu *et al.* [8] showed the most promising results. A possible reason relies on the use of a RNN model, which is capable of directly handling the time dependency that is inherent to the problem. On the other hand, the method proposed by Chaves *et al.* [10] only considers a snapshot of the SMART attributes to make a RUL prediction. Although the method proposed by Chang Xu *et al.* [8] had a remarkable performance, it is well known that RNNs suffer from the problem of gradients vanishing/exploding. This problem is more evident for problems with long term dependencies. Considering the problem of HDDs failure prediction our main hypothesis is that RNN model may not achieve good results for long term predictions. For such cases, we propose the use a prediction model based on the LSTM network.

III. Theoretical Background

A. RNNs

Standard feed-forward neural networks assume the independence between samples within the training and test data sets. After the processing of each sample, the network's state (activation) is lost. Therefore, if such data has temporal relation, this approach can lead to unsatisfactory results. RNNs, on the other hand, are quite effective at capturing sequential dependencies since they have connections between units that work as internal states. Such states can be seen as memory cells, allowing the network to store and use past data.

In this context, the simple recurrent network (SRN) was proposed by Elman [11] and basically consists in concatenating hidden layer units such that each unit receives as input the hidden state vector of the previous unit as well as the current input from the sequence being processed. To learn the parameters of this network, i.e. weights and biases, a well known approach is to share them across those hidden layer concatenated units, that are produced by unfolding the recurrent unit into a desired number of steps, and apply back-propagation. This algorithm is known as back-propagation through time (BPTT) [12] and the errors are back-propagated both through the unfolded and the subsequent layers connections.

Given an input sequence $x_1, ..., x_T$ and the weight matrices for the recurrent, input and output connections, the following equations define the forward step of the Elman network for calculating the outputs. Here, the matrices W, U and V are used for denoting, respectively, the recurrent, input and output connections. Also, the network hidden state is represented by h and y stands for the network's output. Note that the subscript t indicates operations related to the index t within a sequence of size T.

$$\begin{aligned} h_t &= \phi_h(Wh_{t-1} + Ux_t + b_h) \\ y_t &= \phi_y(Vh_t + b_y) \end{aligned}$$

Usually, a sigmoid activation function is employed for ϕ_h and a softmax function for ϕ_y. The layer defined by those equations can further be stacked, resulting in a multi-layer recurrent neural network, capable of capturing more complex structures in sequences.

As described by Bengio et al. [13] these recurrent models suffer from the learning problems of exploding and vanishing gradients during it's training, the latter being the most common case, turning the learning of long-term dependencies into a difficult problem.

B. LSTMs

LSTM networks were first proposed by Hochreiter and Schmidhuber [9] to address the aforementioned issues related to gradient based learning methods when back-propagating over long (deep) sequences. It does so by enhancing previous recurrent networks to include both a memory cell and a gating mechanism, which allows for controlling what is kept (remembered) in memory and how the new input information contributes to what is already in this memory cell.

The method proposed in this work uses a version of the LSTM that is very similar to the one introduced by Gers et al. [14]. This model builds on a series of improvements that have been proposed to the standard version since its introduction. Such enhancements include the use of forget gates [15], that adds the capability of learning to remove (forget) information stored in the memory cell, and of peephole connections, which allows LSTMs to learn precisely spaced (timed) patterns (e.g. spikes) and counting of the internal states at the cost of additional weight matrices. It is important to notice that we do not make use of peephole connections.

The following equations detail how the hidden state and cell memory are calculated during the forward pass of the LSTM cell unit described in the previous paragraph. Here we assume that the LSTM block is given an input sequence $x_1, ..., x_T$, of size T, and x_t denotes the entry at index t in this sequence. Also, h_t and c_t represent the hidden state and cell memory, respectively.

$$\begin{aligned} \widetilde{c}_t &= tanh(W_c h_{t-1} + U_c x_t + b_c) && \text{candidate state} \\ i_t &= \sigma(W_i h_{t-1} + U_i x_t + b_i) && \text{input gate} \\ f_t &= \sigma(W_f h_{t-1} + U_f x_t + b_f) && \text{forget gate} \\ c_t &= i^t \odot \widetilde{c}^t + f^t \odot c^{t-1} && \text{cell state} \\ o_t &= \sigma(W_o h_{t-1} + U_o x_t + b_o) && \text{output gate} \\ h_t &= o_t \odot tanh(c_t) && \text{output} \end{aligned}$$

For a better visualization of how these different components relate and also a comparison between the LSTM cell and a SRN unit, see Figure 1.

C. Random Forest

Random Forest (RF) is an Ensemble Learning method proposed first in [17], consisting in the construction and execution of a myriad of Decision Trees, while the modeling and assembling of individual trees are performed in training time. In classification tasks, the mode of the classes computed by the trees is taken as output, while the mean is calculated for regression tasks.

A Decision Tree consists on a predictive model that relies in decision making. Each tree receives the whole training data as input and performs a series of comparisons of the data values among themselves or with constant values. The output of each comparison describes a 'branching' in the tree and leads to specific subsequent comparisons, ultimately leading to a specific leaf in the tree, which assigns the return value, be it the class id for classification or a numeric value for regression. The fitting of a decision tree is performed commonly by a greedy algorithm, which utilizes recursive partitioning, as described in [18].

Random Forests avoid Decision Tree's tendency to overfit [17] by creating an assembly of trees. Each one is fitted to varying versions of the data, where only a subset of the features is present. Most recent algorithms also utilize bootstrap aggregation [19].

IV. Proposed Method

The proposed method consists basically in three steps, namely, RUL Binning, Model Creation and Failure Prediction.

A. RUL Binning

Similar to the work of [10] we perform a discretization of the remaining useful life attribute of the HDDs, but with several particularities. Instead of evenly spaced intervals for discretization (e.g. trimesters, months, days), we apply a custom spaced binning of the RUL, allowing user specified configuration of the method hereby proposed. One can see that a benefit of this approach is to have a more fine-grained control of the prediction. For example, by defining a spacing between the discretization buckets that have smaller values as those buckets become closer to the end of the device's useful life, leads to a gradual increase in the information precision for such scenario.

After performing the RUL binning, we can tackle the problem as a multi-label classification task, instead of a regression problem.

B. Model Creation

As we are interested in the categorization of each SMART data sample from a given HDD time-series, it is straightforward to create a model using a LSTM many-to-many architecture. In this paradigm, the input of the network is a sequence and the output is also a sequence, in our case, both with the same length. In the proposed modeling, the input and output corresponds, respectively, to the SMART data time-series of a given HDD and their respective RUL bins.

Our model employs two stacked layers of LSTM networks, with standard feed-forward connections between the two layers, and recurrent ones within the same layer. Such multi-layer LSTMs are known to more naturally capture the structure of sequences and to achieve better performance on difficult temporal tasks [20].

Even though our experiments are performed on a dataset with a considerable amount of data, we apply Dropout [21] to the feed-forward connections of both LSTM layers, as a preventive measure.

A fully connected standard neural network, without activation function in its neurons (i.e. linear), is later applied to the output of the last LSTM layer at each timestep (SMART data sample). This effectively maps the hidden state vector into a vector whose dimensions match the number of intervals in which the RUL was discretized in the first step of our method.

Afterwards, the output (logits) of this layer are then given to a softmax activation function, which produces true discrete probability distributions of the RUL bins.

During the training of the model, the cross entropy objective function (i.e. negative log probability) is employed.

C. Failure Prediction

Finally, in order to perform the Failure Prediction step, the time-series of SMART data collected from a target HDD is given, in order, as input to the network built from the proposed architecture, described in the previous subsection. Basically, for each SMART data entry, the predicted RUL bin is the one whose probability is higher in the output of the softmax layer.

V. Experimental Results

The method proposed in this paper was implemented in Python, using Pandas 0.18.1 [22] and NumPy 1.12.0 libraries for data preprocessing. In addition, we used TensorFlow 1.0.1 [23] for implementing Neural Networks with equations running on GPU. In order to compare the different methods, we used metrics contained in scikit-learn 0.18.1 [24] library.

A. Dataset

For purposes of evaluating the method proposed in this article, we use the data provided by Backblaze Company. This dataset is comprised of the daily observation of 92,348 HDDs during the period from 04/10/2013 to 12/31/2016. These observations contain information regarding the serial number, model, capacity, fault, and 90 SMART attributes of each device. According to the Backblaze Company, a device is labeled as faulty if it stops working (does not turn on or does not receive commands), or if SMART self-test fails for attributes 5, 187, 188, 197, or 198. Most HDD models do not report all SMART attributes. In this case, the values not reported are left blank. In addition, different manufacturers and device models may report different attributes.

To avoid a potential overfitting, we chose to perform the tests with the Seagate ST4000DM000 model whose data are

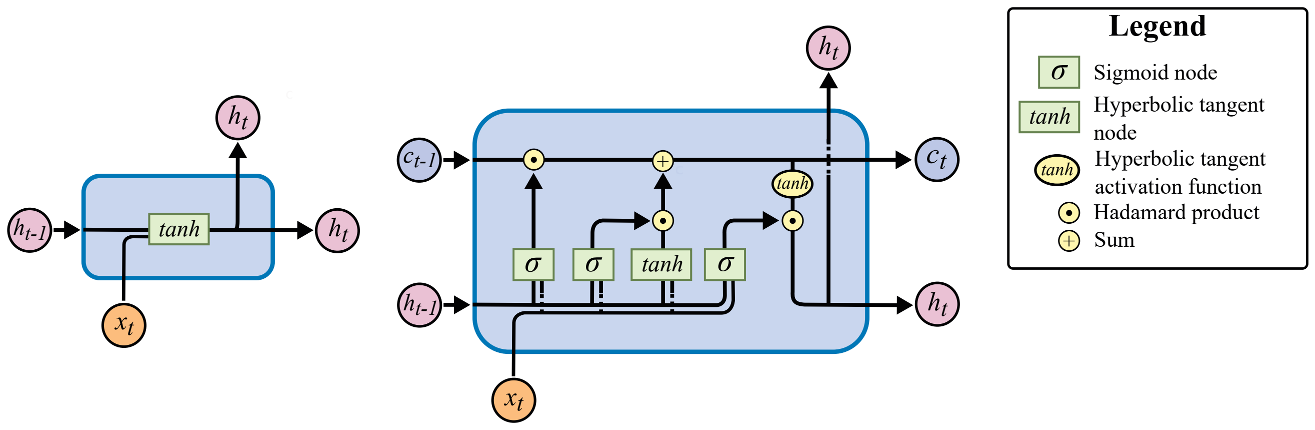

Fig. 1: Details of a SRN unit (left) and a LSTM memory cell (right). The nodes enclosed in a green box encapsulate the weight matrix multiplication plus a bias term given as input to the corresponding function in the box. Both Hadamard product and Sum operations are point-wise. (Adapted from [16].)

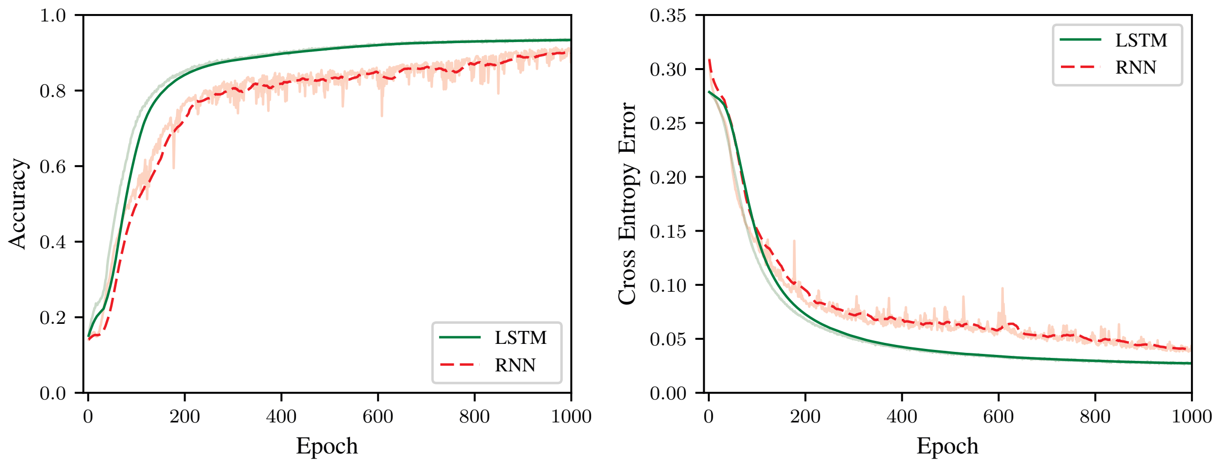

Fig. 2: Training accuracy (left) and loss (right) for 30 days prediction horizon.

most plentiful. This model has 36,555 observed disks, of which 1729 have failures. Of these, 32 were removed because their observation was interrupted without a label indicating a failure or by having submitted observations after being labeled as damaged. Thus, we used in fact for the test of the compared models a set of observations on 1,697 instances.

For this experiment, use all the SMART attributes collected by Backblaze for the chosen model. The selected attributes were: 1, 3, 4, 5, 7, 9, 10, 12, 183, 184, 187, 188, 189, 190, 191, 192, 193, 194, 197, 198, 199, 240, 241 and 242, each having its raw and normalized values included.

B. Performance Evaluation

To verify the performance of the LSTM model we performed short-term and long-term failure predictions. In both tests, the task is to classify each sample in one of the six RUL intervals. However, the RUL intervals are defined within one month before failure for the short-term prediction and one year for the long-term prediction, as can be seen on Figure 3. The proposed model was compared to an Elman RNN and a Random Forest classifier. Both the Elman and LSTM recurrent networks implemented for the experiments follow the multilayer architecture described in Section IV. To define the unrolling parameter of the BPTT applied during the training of the networks, a grid search was performed in the intervals $[2, 30]$ and $[2, 360]$ for short and long-term prediction scenarios respectively. The parameters found for the BPTT unrolling were 6, for the RNN, in both scenarios, and, for the LSTM, 30 days for short and 360 days for long-term. The internal memory was defined to have size 10 and 64 for short and long-term predictions, respectively, in both networks. For the experiments of Random Forest classifier, we used 200 estimators (trees) with no depth limit. In addition, we use bootstrap, which optimizes the construction of the forest, reducing the dependence between its trees. The other

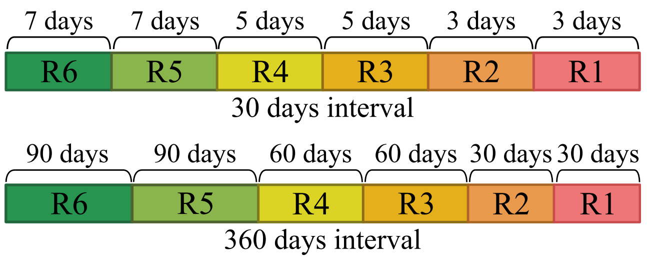

Fig. 3: RUL interval settings used in the experiments. The RUL decreases from left to right.

Classification performance

	30 Days		**360 Days**	
Model	Micro F1	Macro F1	Micro F1	Macro F1
LSTM	0.9840	0.9840	0.7169	0.6861
RNN	0.9819	0.9818	0.3044	0.2547
RF	0.2513	0.1660	0.2598	0.2389

TABLE I: Performance of the classifiers under different prediction horizon settings.

parameters were set as the standards of the sklearn library.

Table I shows the performance of each method for the long and short term prediction tasks. Since the datasets are unbalanced, we used Micro and Macro F-measures as performance indicators. The Micro-averaged F-score basically consists of calculating the F-score taking the sum of the true positives, false positives and false negatives for all classes. The Macro-averaged version simply performs an average over the individual F-scores calculated for each class. Therefore, the Micro F1 tends to bias the metric toward the most populated classes, whereas the Macro treats all classes equally [25].

As can be noticed, for the short-term prediction task, the recurrent methods achieved similar results. However, for the long term predictions, LSTM achieved the best results, followed by the RNN. The best performance of the recurrent methods are expected since both use historical information to perform future predictions. The significant performance gap between LSTM and RNN in long term predictions, enforces our initial hypothesis that gradient vanishing/exploding problems may be observed in Elman networks.

A more detailed analysis regarding the performances of RNN and LSTM can be seen in the confusion matrices in Tables II, III, IV and V. It is interesting to perceive that for the long term prediction (360 days), in addition to achieving the best Micro and Macro F1 scores, the LSTM model tends to produce errors that are concentrated in classes near the correct one. Since this is not observed for the RNN we can state that the LSTM classification errors can be seen as less severe than RNN errors.

Another illustration of the difference between LSTM and RNN can be seen in Figures 2 and 4. These figures show the training accuracy and loss function evaluations for short and long term predictions respectively. For the short term

		Predicted					
		R_1	**R_2**	**R_3**	**R_4**	**R_5**	**R_6**
Actual	**R_1**	**1985**	10	14	14	7	12
	R_2	14	**1953**	4	11	9	12
	R_3	7	0	**1392**	3	3	7
	R_4	2	0	0	**1392**	3	1
	R_5	0	0	0	0	**834**	3
	R_6	0	0	0	0	0	**834**

TABLE II: Confusion Matrix LSTM 30 Days.

		Predicted					
		R_1	**R_2**	**R_3**	**R_4**	**R_5**	**R_6**
Actual	**R_1**	**1972**	17	18	14	7	14
	R_2	12	**1953**	5	11	9	13
	R_3	3	2	**1392**	3	3	9
	R_4	0	0	2	**1391**	4	1
	R_5	0	3	0	0	**831**	3
	R_6	0	1	0	0	0	**833**

TABLE III: Confusion Matrix RNN 30 Days.

prediction we can see that the methods exhibit similar curves indicating that both models learn along the iterations. However, it is noticeable that the RNN can not decrease the training error or improve the performance on the test set for the long-term prediction task.

VI. Conclusion

In this paper we propose the use of a LSTM neural network for failure prediction in Hard Disk Drives. We evaluated the proposed model in long and short term prediction tasks and compared it to an Elman neural network and a Random Forest classifier.

Our experiments showed that the LSTM model had the best performance on the long term prediction task. On the short term prediction, all methods had similar results. Future works

		Predicted					
		R_1	**R_2**	**R_3**	**R_4**	**R_5**	**R_6**
Actual	**R_1**	**5397**	1800	875	891	954	729
	R_2	691	**5248**	1552	906	1103	643
	R_3	752	612	**11415**	2732	1964	1543
	R_4	403	279	560	**11897**	2652	1789
	R_5	260	235	144	552	**18914**	2919
	R_6	129	17	159	149	275	**18736**

TABLE IV: Confusion Matrix LSTM 360 Days.

		Predicted					
		R_1	**R_2**	**R_3**	**R_4**	**R_5**	**R_6**
Actual	**R_1**	**2935**	1468	1206	983	2503	1551
	R_2	2047	**1330**	1201	1122	2656	1787
	R_3	2932	2101	**2106**	2155	5647	4077
	R_4	1762	1329	1495	**1911**	5833	5250
	R_5	1416	1034	1180	1803	**7225**	10366
	R_6	579	348	329	410	2896	**14903**

TABLE V: Confusion Matrix RNN 360 Days.

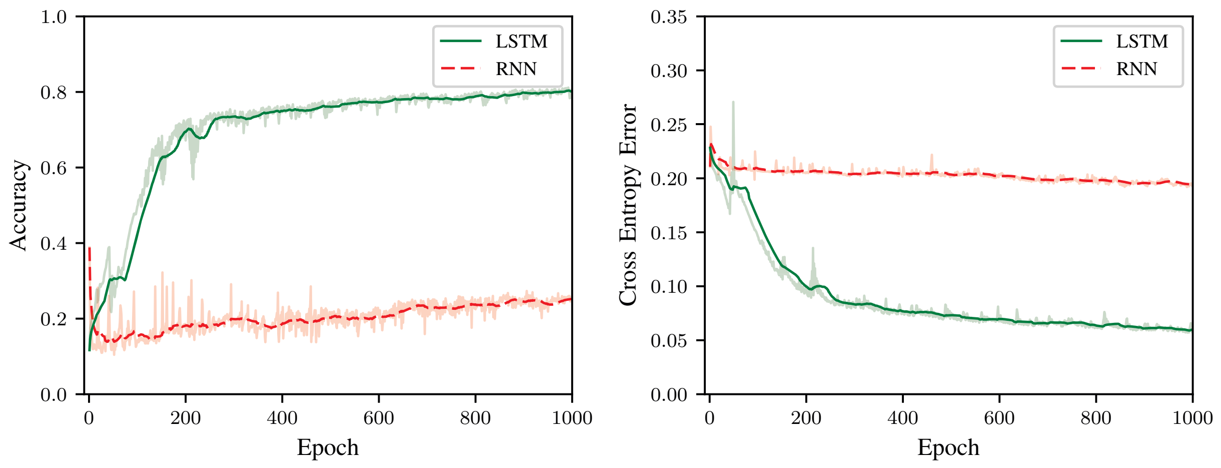

Fig. 4: Training accuracy (left) and loss (right) for 360 days prediction horizon.

include the study of LSTM regression for failure prediction in HDDs.

Acknowledgment

This work was partially funded by Lenovo, as part of its R&D investment under Brazil's Informatics Law, and LSBD/UFC.

References

[1] "Data center downtime costs," Tech. Rep. [Online]. Available: https://www.vertivco.com/en-us/insights/articles/pr-campaigns-reports/benchmark-series/

[2] 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*, vol. 6, no. May, pp. 783–816, 2005.

[3] L. P. Queiroz, F. C. M. Rodrigues, J. P. P. Gomes, F. T. Brito, I. C. Chaves, M. R. P. Paula, M. R. Salvador, and J. C. Machado, "A fault detection method for hard disk drives based on mixture of gaussians and non-parametric statistics," *IEEE Transactions on Industrial Informatics*, 2016.

[4] M. M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann, "Predicting disk replacement towards reliable data centers," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 39–48, 2016.

[5] G. F. Hughes, J. F. Murray, K. Kreutz-Delgado, and C. Elkan, "Improved disk-drive failure warnings," *IEEE Transactions on Reliability*, vol. 51, no. 3, pp. 350–357, 2002.

[6] 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*, vol. 62, no. 1, pp. 136–145, 2013.

[7] Y. Wang, E. W. Ma, T. W. Chow, and K.-L. Tsui, "A two-step parametric method for failure prediction in hard disk drives," *IEEE Transactions on industrial informatics*, vol. 10, no. 1, pp. 419–430, 2014.

[8] 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*, vol. 65, no. 11, pp. 3502–3508, 2016.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] I. C. Chaves, M. R. P. de Paula, L. G. Leite, L. P. Queiroz, J. P. P. Gomes, and J. C. Machado, "Banhfap: A bayesian network based failure prediction approach for hard disk drives," in *Intelligent Systems (BRACIS), 2016 5th Brazilian Conference on*. IEEE, 2016, pp. 427–432.

[11] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[12] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[13] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[14] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[15] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[16] C. Olah, "Understanding lstm networks," 2015, [Online; accessed 2017-04-26]. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[17] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1. IEEE, 1995, pp. 278–282.

[18] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[19] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," in *Advances in neural information processing systems*, 2013, pp. 190–198.

[21] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] W. McKinney, "Pandas: a python data analysis library," 2008–, [Online; accessed 2017-04-26]. [Online]. Available: http://pandas.sourceforge.net

[23] 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.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.