Memory Error Driven Server Failure Detection

Rafael A. Silva, Francisco Lucas F. Pereira, Victor A. E. Farias, Felipe T. Brito, Javam C. Machado

Laboratório de Sistemas e Bancos de Dados
Departamento de Computação – Universidade Federal do Ceará, Brazil
{rafael.albuquerque,lucas.falcao,victor.farias,felipe.timbo,javam.machado}@lsbd.ufc.br

Abstract. The correct functioning of Dynamic Random Access Memory (DRAM) is of fundamental relevance to the functioning of servers in data centers. Therefore, being able to detect server failure caused by memory errors is fundamental to the development of prediction methods that can be used to avoid server failure caused by memory errors. Thus, ensuring the continuous availability of the hosted services. In recent years, many authors proposed machine learning-based methods to predict server failure based on the occurrence of DRAM errors. However, from previous works, one can notice that this is a challenging task due to the lack of data and the irregularity in which memory errors occur. In this work, through feature engineering, we look forward to improving the classification accuracy of recurrent neural networks at dealing with irregularly sampled data in order to improve the accuracy in identifying servers that are nearing a failure state.

CCS Concepts: • **Computing methodologies** → **Machine learning algorithms**.

Keywords: Classification, LSTM, Failure Detection, Memory

1. INTRODUCTION

As information systems become central in fundamental human activities, such as entertainment, economy, and communication, the maintenance of these services becomes critical. One fundamental aspect related to the availability of these services is the reliability of the servers in which the systems are hosted. As noted in [Cheng et al. 2022], the proportion of server failures caused by memory error is substantial, reaching up to 30% of all hardware-related server failures. In this context, identifying prone to failure servers can improve the reliability of the services being provided by migrating them to a new server before a server failure occurs. However, as observed in [Yu et al. 2023], an inaccurate identification method resulting in too many false positives can lead to the support team being unable to handle them in a timely manner, while too many false negatives may lead to system downtime. Therefore, an accurate method for server failure detection is fundamental.

Many works in literature have employed machine learning techniques, such as Decision Trees [Cheng et al. 2022; Yu et al. 2021], Neural Networks [Cheng et al. 2022], and Long Short-Term Memory (LSTM) [Sun et al. 2019] to predict memory faults, and server failure caused by memory errors. As described in [Cheng et al. 2022], memory-related data are usually collected when a memory error occurs. Therefore, as memory errors are distributed over time, it would be necessary to analyze the time series of memory errors. Furthermore, memory errors are not uniformly distributed in time. This means that the occurrence of memory errors might be separated by a few seconds or a few days.

A natural approach to processing time series is to use the LSTM neural network architecture, as it is inherently designed to work with time series data. However, despite its suitability to work with time series, as in [Sun et al. 2019], it is not adequate to cope with irregularities in which data samples are collected. Although other works have been dedicated to using LSTM to process time series to

identify server failure caused by memory errors as in [Sun et al. 2019], the results are unsatisfactory.

In this work, we look toward improving the classification accuracy obtained when using LSTM to identify server failure through feature engineering to deal with irregularly sampled data. One of the main challenges that we faced in this work was to deal with the high irregularity in which memory errors occur, as for many servers the memory errors occur for a long time before server failure, while for others the errors occur only a few seconds before failure. In our approach, to deal with this irregularity, we created a new feature describing the time interval in which memory errors occur. With our approach, we obtained satisfactory accuracy in identifying servers that failed and servers that did not fail.

The remainder of this paper is divided as follows. Section 2 briefly reviews the related work. In Section 3 we describe the structure of the memory system and types of server failure caused by memory errors. Our approach to dealing with irregular sample data is presented in section 4. The experiments and the associated analysis are shown in Section 5. Finally, Section 6 concludes the paper and discusses the future directions.

2. RELATED WORK

Memory-related problems are one of the main causes of server failure in data centers. Therefore, an approach to predict server failure induced by memory errors is essential to allow for proactive hardware maintenance and migration of services to another server. Thus ensuring the continuous provisioning of services. Over the years, many works have been dedicated to proposing approaches to predict server failures, such as [Cheng et al. 2022; Meza et al. 2015; Bogatinovski et al. 2022; Yu et al. 2021; Du et al. 2020; Beigi et al. 2023; Yu et al. 2023; Sun et al. 2019].

The authors in [Cheng et al. 2022] investigated the application of several traditional machine learning methods in predicting server failure caused by memory errors. Their intention was to predict server failures within 5 minutes before the actual failure in other to allow for service migration. Some of the machine learning methods were Random Forests (RF) [Breiman 2001], Support Vector Machine (SVM), and Multilayer Perceptron (MLP). However, as observed in [Sun et al. 2019], these methods are usually better suited to deal with tabular data. Therefore, the authors generated features such as the number of errors, mean time between errors, mean, median, and standard deviation of the number of memory errors.

In [Bogatinovski et al. 2022], instead of predicting server failures, the authors focused on predicting memory hardware failure. The objective in predicting memory failure is to allow the maintenance team to proactively replace memory modules that are prone to failure. Therefore, they created a system to predict hardware failure three hours before the actual failure. By analyzing the memory position in which the memory errors occur, the authors concluded that memory errors in nearby memory positions are a strong indicator of memory failure. The authors also proposed a new set of features that can be incrementally calculated from the time series. The used method to identify hardware failure in memory modules were XGBoost [Chen and Guestrin 2016] and Random Forest.

Another different approach was presented in [Du et al. 2020], in which the authors designed a predictor based on details related to memory errors to predict the occurrence of uncorrectable memory errors, which lead to server failure. The predictor counts the number of distinct memory addresses with observed errors within the last 24 hours. If the number of addresses exceeds a predefined threshold, the memory module is predicted to experience an uncorrectable error in the future.

Instead of trying to predict server failures before they happen, in [Sun et al. 2019], the authors use LSTM recurrent neural networks to identify healthy and failed servers by analyzing their entire time series of memory errors. As in many servers the memory errors occur for a long time before server failure, while for others the errors occur only a few seconds before failure, identifying a time interval

to predict server failure proved to be a difficult task to tackle. Thus, in this work, we will use LSTM to identify failed serves by analyzing their entire time series of memory errors.

3. BACKGROUND

In this section, we will describe the memory system of the servers from which the data we used in this work were collected. Additionally, we will describe the memory errors that can occur on a server and the types of server failures that are caused by memory errors.

3.1 Memory System

In this work, we are using publicly available data collected at Alibaba [Alibaba 2023]. The collected data contain details about memory errors from servers with two CPUs, and each CPU has two memory controllers. In addition, each memory controller is connected to three memory channels, and each memory channel is connected to up to two DIMMs (Dual In-Line Memory Module).

Each DIMM comprises multiple DRAM (Dynamic Random Access Memory) chips that are organized in ranks, and chips in the same rank can be simultaneously accessed in read/write operations. Furthermore, each DRAM chip is partitioned into banks, which are further partitioned into rows and columns. Each pair of rows and columns identifies a memory cell, which can store a single bit of data. The group of cells in a single row is usually called a page. In addition, a pair of row and column addresses identifies a unique 4-bit word in the DRAM device.

3.2 Memory Errors

Memory errors are events that result from reading a bit that is different from the bit that was originally written in a memory cell. Sometimes, memory errors are triggered by memory cells that are stuck-at bit due to hardware failure or flipped bits due to the interference of external factors. As observed in many works, [Ziegler and Lanford 1979; May and Woods 1979; Gong et al. 2017], multiple factors can trigger bits to flip in memory cells, such as electromagnetic interference, cosmic ray strikes, and faults in the DRAM chip internal structure.

To correct flipped bits, modern memory employs error correcting codes (ECC). To detect and correct memory errors using ECC, the memory controller periodically scans all DRAM cells in search of flipped bits. This process is called scrubbing [Awasthi et al. 2012]. Flipped bits can also be detected during read and write operations. Some examples of ECC are the single-error-correction-double-error-detection (SEC-DED) and the single device data correction (SDDC). The SEC-DED can be used to correct any single bit, while the SDDC can be used to correct up to four bits in a single word. However, if the number of erroneous bits exceeds the capability of the ECC, the error cannot be corrected. In this case, we say the error is an uncorrectable error (UE); otherwise, it is a correctable error (CE).

3.3 Server Failure

We say a server fails due to memory errors when the server is no longer able to support the memory operations needed for the functioning of the hosted services. For the dataset used in this work, three types of server failure were identified in [Cheng et al. 2022].

—UE-driven failure: Caused by the occurrence of UEs, in which the server crashes or cannot allow the hosted applications to access data in the memory.
—CE-driven failure: It results from the incapability of the memory controller to deal with a large number of correctable errors.
—Miscellaneous: It refers to a server failure caused by errors such as a disconnected memory module or when too many DRAM pages are inaccessible.

3.4 Long Short-Term Memory

Recurrent neural networks (RNNs) constitute a broad class of neural models designed to capture patterns in sequential data. Recurrent models surpass traditional ones when working with sequential information due to their capability to retain short-term dependencies between data. However, such models are not capable of coping with long-term dependencies [Bengio et al. 1992]. Such a fact motivated the design of the Long Short-Term Memory (LSTM) [Graves and Schmidhuber 2005]. LSTM employs control gates to manage the entrance and exit of information in the cell state. These gates are associated with internal memory to RNN cells, controlling the flow of information from the input and the previous states. Figure 1 (adapted from [Yan 2015]) represents an LSTM cell unit.

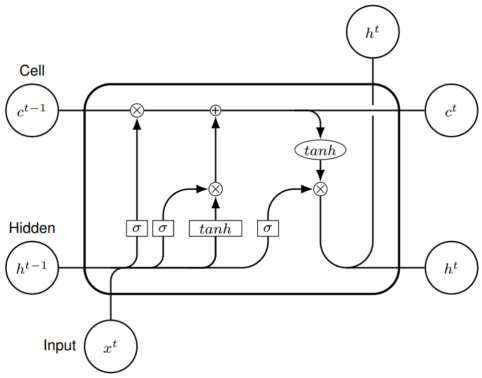


Fig. 1: LSTM cell

LSTM has the forget, input, and output gates to control the flow of information. The forget gate controls how much information the unit accepts from the input and the last state. On the other hand, the input state regulates how much information persists in the current state cell. Lastly, the output gate determines how much information the current cell state can output. The equations below describe how to calculate each of these gates, the cell states, and the hidden states for the forward pass of an LSTM layer. In these equations, t denotes the processed index within a sequence x. The symbol $\odot$ represents the Hadamard product. W and U are both the current and input matrices, respectively, with a subscript indicating the associated gate. The parameter b is the bias term, and the c and h are the cell and hidden state, respectively.

$$\tilde{c}_t = tanh(W_c h_{t-1} + U_c x_t + b_c) \quad \text{candidate state} \quad (1)$$

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i) \quad \text{input gate} \quad (2)$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f) \quad \text{forget gate} \quad (3)$$

$$c_t = i^t \odot \tilde{c}^t + f^t \odot c^{t-1} \quad \text{cell state} \quad (4)$$

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o) \quad \text{output gate} \quad (5)$$

$$h_t = o_t \odot tanh(c_t) \quad \text{output} \quad (6)$$

4. METHODOLOGY

The dataset we use in this work is highly unbalanced, as the number of healthy servers is much greater than the number of failed servers. In this section, we describe the process we used to deal with this unbalanced dataset. Furthermore, we also detail the approach to dealing with the irregularity in which data related to memory errors is collected.

4.1 Feature Engineering

As noted in [Siami-Namini et al. 2019], despite LSTM networks' satisfactory performance when using uniformly sampled sequential data, they tend to perform poorly when the data is not uniformly distributed in time. This characteristic is challenging when working with data related to memory errors because memory errors are irregularly distributed in time. To circumvent this limitation, we created a new feature describing the time interval in seconds between the occurrence of the memory errors. We adopt 0 as the time interval between the first memory error to its predecessor. Furthermore, we will not work with time series containing only one entry as they do not carry enough information to identify if a server is healthy or not. In this work, we will analyze the impact of using the time interval feature on classification accuracy.

4.2 Class Imbalance

The dataset that we are working with initially contains $2,137$ time series from failed servers and $28,362$ time series from healthy servers. After removing the time series with only one entry, it remains $2,034$ time series from failed servers and $22,968$ time series from healthy servers. We divided the set of time series from failed servers in the proportion of 22% for testing, 25% for validation, and 53% for training. The number of time series from healthy servers for testing, validation, and training is such that these sets have the same number of time series from healthy and failed servers. The motivation behind creating balanced sets is to avoid overfitting to any of the classes when training the model and also to obtain better metrics in the experiments. As there are more healthy servers than failed servers, we created multiple training sets in which the time series from healthy servers are different in each training set. In total, we created 20 training sets. As we have multiple training sets, we train our LSTM model in multiple sections. In each section, the model is trained with a different training set. Also, each training section runs for at most $1,000$ epochs. We use the validation set to decide when a training section should stop. Thus avoiding overfitting the model to any of the training sets.

5. EXPERIMENTAL RESULTS

In this section, we describe the dataset being used in this work, how the experiments were performed, and the results obtained. The experiments were implemented in Python with the library Pytorch 2.0.0 [Imambi et al. 2021] for the creation and execution of the Machine Learning models, Pandas 1.5.3 [pandas development team 2020], and Numpy 1.23.5 [Harris et al. 2020] for data prepossessing.

5.1 Dataset

The dataset used contains DRAM errors from 250K servers at Alibaba. The collected data span 8 months. The dataset includes two data types that we use in this work: error logs and trouble ticket logs. The records in error logs describe the memory errors of all servers. The servers in the trouble ticket logs are those that failed.

In the error logs dataset, we have records of 75.1M of correctable errors from $30,496$ servers (including healthy and failed servers) and $87,186$ write errors from 351 servers. The logs contain records of details of any DRAM error event, including the server ID, DIMM ID, rank ID, bank ID, row ID, column ID, error time, and whether the error was detected during a scrubbing, read or write operation. As a result, for each server, we have a time series constituted by all of its memory errors. Thus, the dataset we are working with is a set of time series.

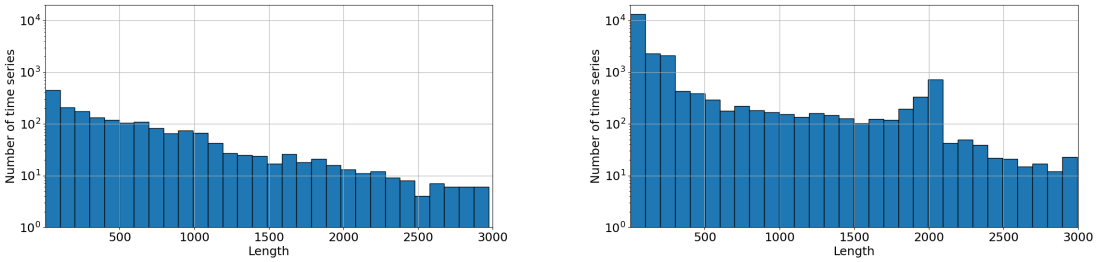
The trouble ticket logs portion of the dataset contains records of abnormal system-level events as system crashes. Once a failure is detected the maintenance team creates a trouble ticket containing the server ID, the time, and the failure type. In total, there are $3,017$ trouble tickets that can be used to identify the error logs of the failed servers. From all failed servers, for $2,137$ there is at least one

CE before failure. Therefore, a total of 880 servers failed without suffering any memory error. The distribution of the cause of the failed servers with at least one memory error is described in Table I.

Cause of failure	Number of failed servers
UE-drive	567
CE-driven	809
Miscellaneous	761

Table I: Cause of server failures in numbers.

In Figure 2, we present the frequency distribution of the size of the time series from failed and healthy servers after removing the time series of size one. The distribution comprehends time series of sizes up to 3000, representing over 90% of the time series for both failed and healthy servers.



(a) Time series size distribution for failed servers.

(b) Time series size distribution for healthy servers.

Fig. 2: Frequency distribution of time series size.

5.2 Classification Results

We performed two experiments; the first will be our baseline, and the second show the results obtained using our approach. In the first experiment, we use a dataset with the original set of features from the Alibaba dataset. In the second experiment, we are using the same dataset as the first experiment, but the only feature that we used is the time interval between the memory errors. For each experiment, we trained a model following the approach described in Subsection 4.2. For each time series in the test set, the classification indicating if a server is healthy or not is obtained at the end of the time series. The results of the experiments are shown in Table III.

Hyperparameter	Value
Epochs	1000
Optimizer	Rprop
LSTM Cell Hidden Size	16
LSTM layers	1
Dense Layer Sizes	[16, 32, 2]
Loss function	BCELoss

Table II: LSTM model hyperparameters.

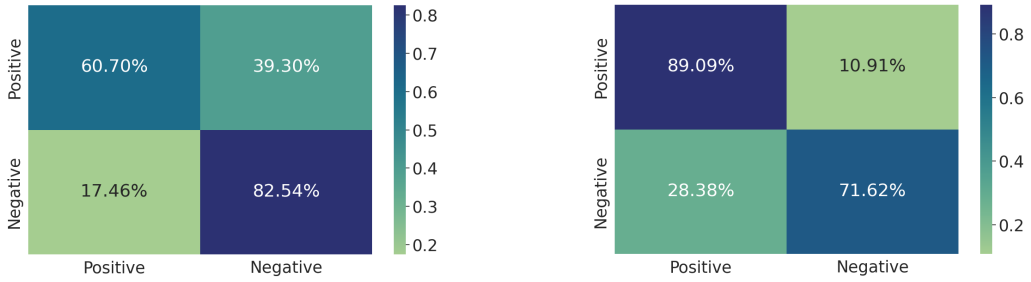
From Table III, we observe that when compared to the baseline, our approach provides a better result in correctly identifying failed servers. Furthermore, we note that the number of false negatives is much higher in the baseline. The downside of a system with many false negatives is that many servers would fail without any previous notification to the maintenance team.

In [Sun et al. 2019], the authors tackled the problem of identifying server failure caused by memory errors. They used LSTM to process the time series of memory errors containing features such as

Experiment	TP%	FP%	FN%	TN%	Precision	Recall	F1 Score	Accuracy
Baseline	60.69	17.46	39.30	82.53	0.77	0.60	0.68	0.70
Our approach	89.08	28.38	10.91	71.61	0.75	0.89	0.81	0.80

Table III: Test results.

memory speed, age, and details about corrected errors. In their approach, the authors obtained a precision of 0.50, recall of 0.55, and F1 scores of 0.53. From Table III, using the time interval between errors, with our methodology, we achieved better performance, with the precision of 0.75, 0.89 for recall, and 0.81 for F1 score.



(a) Confusion matrix for the baseline.

(b) Confusion matrix for our approach.

Fig. 3: Confusion matrix for the test reults in Table III.

6. CONCLUSION

Hardware failures are a major concern in data centers as they may result in service slowdown or unavailability, which becomes critical in an unprecedentedly connected society. To ensure fundamental services are continuously provided, over the years, much work has been done to predict hardware faults such as Hard Disk Drive (HDD) failure [Felix et al. 2023; Pereira et al. 2022], and memory failure [Bogatinovski et al. 2022]. In [Sun et al. 2019], the authors explored LSTM to identify server failure caused by memory errors; however, with limited success. In this work, we contribute by exploring new possibilities to improve the accuracy of LSTM to identify server failure driven by memory errors.

Despite being a natural choice for working with time series, the LSTM architecture presents some limitations when dealing with irregularly sampled data. In this work, we circumvent this limitation with some feature engineering. By creating a new feature to describe the time interval between memory errors, we obtained satisfactory classification accuracy in identifying healthy and failed servers.

Future developments encompass identifying new features that can be used to circumvent the time sample irregularity of the records describing memory errors. Furthermore, we intend to develop a prediction method to identify when a server is nearing a failure point and prevent service interruption by migrating it to other servers.

ACKNOWLEDGMENT

This research was partially funded by Lenovo, as part of its R&D investment under Brazilian Informatics Law, by CAPES grants 88887.609134/2021-00 and 88887.609129/2021, and CNPQ grants 307323/2022-6 and 316729/2021-3.

REFERENCES

Alibaba. Large-scale dataset for prediction of server failures due to dram errors, 2023. `https://tianchi.aliyun.com/dataset/132973` Accessed: (2024-07-15).

Awasthi, M., Shevgoor, M., Sudan, K., Rajendran, B., Balasubramonian, R., and Srinivasan, V. Efficient scrub mechanisms for error-prone emerging memories. In *IEEE International Symposium on High-Performance Comp Architecture*. IEEE, pp. 1–12, 2012.

Beigi, M. V., Cao, Y., Gurumurthi, S., Recchia, C., Walton, A., and Sridharan, V. A systematic study of ddr4 dram faults in the field. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, pp. 991–1002, 2023.

Bengio, Y., De Mori, R., Flammia, G., and Kompe, R. Global optimization of a neural network-hidden markov model hybrid. *IEEE transactions on Neural Networks* 3 (2): 252–259, 1992.

Bogatinovski, J., Kao, O., Yu, Q., and Cardoso, J. First ce matters: On the importance of long term properties on memory failure prediction. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, pp. 4733–4736, 2022.

Breiman, L. Random forests. *Machine learning* vol. 45, pp. 5–32, 2001.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. pp. 785–794, 2016.

Cheng, Z., Han, S., Lee, P. P., Li, X., Liu, J., and Li, Z. An in-depth correlative study between dram errors and server failures in production data centers. In *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, pp. 262–272, 2022.

Du, X., Li, C., Zhou, S., Ye, M., and Li, J. Predicting uncorrectable memory errors for proactive replacement: An empirical study on large-scale field data. In *2020 16th European Dependable Computing Conference (EDCC)*. IEEE, pp. 41–46, 2020.

Felix, G. S., Pereira, F. F., Praciano, F. D., Gomes, J. P., and Machado, J. C. Dynamic sample weighting to predict the remaining useful life of hard disk drives. In *Anais do XI Symposium on Knowledge Discovery, Mining and Learning*. SBC, pp. 89–96, 2023.

Gong, S.-L., Kim, J., and Erez, M. Dram scaling error evaluation model using various retention time. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, pp. 177–183, 2017.

Graves, A. and Schmidhuber, J. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 4. IEEE, pp. 2047–2052, 2005.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature* 585 (7825): 357–362, Sept., 2020.

Imambi, S., Prakash, K. B., and Kanagachidambaresan, G. Pytorch. *Programming with TensorFlow: solution for edge computing applications*, 2021.

May, T. C. and Woods, M. H. Alpha-particle-induced soft errors in dynamic memories. *IEEE transactions on Electron devices* 26 (1): 2–9, 1979.

Meza, J., Wu, Q., Kumar, S., and Mutlu, O. Revisiting memory errors in large-scale production data centers: Analysis and modeling of new trends from the field. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, pp. 415–426, 2015.

pandas development team, T. pandas-dev/pandas: Pandas, 2020.

Pereira, F. L. F., Bucar, R. C., Brito, F. T., Gomes, J. P. P., and Machado, J. C. Predicting failures in hdds with deep nn and irregularly-sampled data. In *Brazilian Conference on Intelligent Systems*. Springer, pp. 196–209, 2022.

Siami-Namini, S., Tavakoli, N., and Namin, A. S. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International conference on big data (Big Data)*. IEEE, pp. 3285–3292, 2019.

Sun, X., Chakrabarty, K., Huang, R., Chen, Y., Zhao, B., Cao, H., Han, Y., Liang, X., and Jiang, L. System-level hardware failure prediction using deep learning. In *Proceedings of the 56th Annual Design Automation Conference 2019*. pp. 1–6, 2019.

Yan, S. Understanding lstm networks. *Online). Accessed on August* vol. 11, 2015.

Yu, F., Xu, H., Jian, S., Huang, C., Wang, Y., and Wu, Z. Dram failure prediction in large-scale data centers. In *2021 IEEE International Conference on Joint Cloud Computing (JCC)*. IEEE, pp. 1–8, 2021.

Yu, Q., Zhang, W., Notaro, P., Haeri, S., Cardoso, J., and Kao, O. Himfp: Hierarchical intelligent memory failure prediction for cloud service reliability. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, pp. 216–228, 2023.

Ziegler, J. F. and Lanford, W. A. Effect of cosmic rays on computer memories. *Science* 206 (4420): 776–788, 1979.