J. Electrical Systems 19-2 (2023): 253-263

[1]Ravi Kiran Gadiraju

Deep Learning Models for Predicting Hardware Failures Using Large-Scale Telemetry Data

Journal of Electrical Systems

Abstract: The contemporary computing infrastructure is producing masses of telemetry data through system logs, sensors and health metrics. The use of such data through deep learning is potentially useful in predicting hardware failures even before they happen and reduce downtime and maintenance expenses. This paper explores the Long Short-Term Memory (LSTM) networks and Autoencoder models to predict failures based on large scale telemetry data. The suggested LSTM model is trained with time dynamics leading to failures, whereas an LSTM-based autoencoder identifies anomalous behavior by rebuilding usual sequences and indicating anomalies. These models are trained and tested with multi-source telemetry data resulting in high accuracy and precision in prediction. The LSTM model, especially, also has good results in predicting the precursor signals of failures, and it is better than a baseline random forest classifier. The findings indicate that deep learning can also be used to take advantage of telemetry properties (CPU usage, temperature, disk health indicators etc.) to give early notifications of hardware problems. We talk about model accuracy, precision recall qualities and trade-offs between false alarms and missed detections. The paper identifies the potential of deep learning-based predictive maintenance to improve the reliability of systems.

Keywords: Predictive maintenance; Telemetry; Hardware failure prediction; LSTM; Autoencoder

Introduction

The consistent functionality of hardware is essential in data centers and personal computing because errors may result in expensive downtimes and information loss. Proactive failure prediction is a type of predictive maintenance problem, which predicts the emergence of hardware problems using system health and usage information (telemetry) before they occur as critical failures (Carvalho et al., 2019). Telemetry data can be system logs, sensor values (e.g. temperatures, voltages), disk health metrics (e.g. S.M.A.R.T attributes). Early research provided insights into the opportunities and limitations of application of such data in prediction of failure. Indicatively, a study conducted on Google disk drives revealed that, some of the S.M.A.R.T. factors (self-monitoring signals) are related to drive failures, however, predictions using the same factors alone, could not adequately predict individual disk failures. This knowledge indicated that sophisticated modeling capabilities were required to reflect complicated antecedents of failure. In the 2000s, traditional statistical models and machine learning methods were used on system logs and system metrics. The other studies by Sahoo et al. (2003) utilized time-series extrapolation of resource usage measures to predict situations that would cause failures. There was also research involving data mining of event logs by other researchers; an example is Yamanishi and Maruyama (2005) with dynamic syslog pattern analysis to identify network faults in real-time. With the increase in size and complexity of telemetry data, multi-dimensional sensor data was learned via machine learning techniques such as support vector machines and decision trees to learn failure signatures (Russo et al., 2015). Nevertheless, these traditional methods frequently had to perform a lot of feature engineering, and were unable to cope with both the lack of linearity and temporal correlations of failure processes.

[1]Independent researcher

ravikgraju@gmail.com

Deep learning is a relatively recent phenomenon that has introduced a potent predictive failure analytics tool in recent years due to the possibility of modeling complex patterns in large data streams. Recurrent neural networks, in particular, Long Short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) are particularly effective in sequential telemetry data since they can capture long term dependencies and temporal relationships. In the meantime, unsupervised deep learning algorithms like autoencoders can detect abnormal behavior without providing explicit failure modes, which are used together with supervised algorithms. In this paper, LSTM-based models and deep autoencoders are discussed to predict the hardware failure using a large scale of telemetry data. We propose an entire system that contains data gathering of a fleet of computing systems, model education on both labeling and anomaly detection duties and a test on prediction efficiency. The subsequent parts of this paper give a literature review on the related body of work, give the methodology of our experiments, results of the experiment in terms of accuracy and precision, and the conclusion of what can be discovered when it comes to deploying deep learning models to manage proactive hardware failures.

Literature Survey

Hardware failures are a field of study that has become popular in system reliability. Initial efforts were made on the prediction of statistical failure and classical machine learning to system logs. Sahoo et al. (2003) gathered system health logs on large computer clusters and used time-series model to predict major metrics before failures. They adopted a naive predictor (e.g. moving averages of CPU and network usage) to make forecasts in the occurrence of critical events that may cause crashes. Also at approximately the same time, Yamanishi and Maruyama (2005) proposed techniques to mine unstructured logs (syslogs) to identify network failure abnormalities and showed one of the earliest data-driven online failure detection systems. These initial investigations indicated how telemetry could be useful, though with some limitations: naive models tended to overlook the cases of complex failure or generate excessive false alarms. In a global survey, Salfner et al. (2010) listed many prediction techniques of failures, pointing out a trade-off between the detection time window and accuracy, and suggesting more resilient methods that can use online streams of data.

As of the 2010s, scientists started to use more advanced machine learning. Russo et al. (2015) used support the machine (SVMs) and non-linear kernels to examine multi-dimensional telemetry (CPU, memory, I/O stats) and forecasting an error in the future. Their case study of a spacecraft telemetry system revealed that machine learning was able to learn complexities of sensor readings, which were subtle enough to represent a failure on the way. Equally, the power of machine learning to predict failure in industrial equipment was confirmed through the results of many other studies dedicated to the sphere of predictive maintenance (Carvalho et al., 2019). These conventional ML models, however, had to be very cautious when extracting features on logs or sensor streams and often lacked temporal sensitivity of the information. As an example a model could take the summary statistics of the prior N minutes of sensor readings to make a prediction of a failure possibly missful of sequence patterns.

This is due to the introduction of deep learning to failure prediction. Deep neural networks, in contrast to a static classifier, can also learn feature representations by looking at raw sequences. Specifically, recurrent neural networks (RNNs) gained popularity in prediction of failures based on logs. A type of RNN LSTM network was used on telemetry sequences to obtain long-range dependencies. A hybrid model (comprising a Bi-directional LSTM (to identify temporal trends in system metrics) and a Random Forest (to learn spatial or categorical data) was proposed to predict node failures on a cloud computing platform by Lin et al. (2018), which outperformed the state of the art. The other notable study was by Zhang et al. (2018), whose method, named Prefix, identified templates in semistructured logs and used random forest to forecast failures of a switch in datacenter networks, with high accuracy, whereas other existing algorithms such as SVM or Hidden Markov Models failed to do so. Their findings showed that despite the absence of deep learning, feature learning and careful log parsing could still be very predictive of a particular problem. However, the state of the art rapidly improved due to the deep learning methods. One of the earliest deep learning models with respect to failure prediction in high-performance computing systems

proposed by Das et al. (2018) relies on an LSTM, where the component learns a sequence of templates of log message patterns that result in failure. This was subsequently furthered by Das et al. (2020) with a superior LSTM-based model (called Aarohi) which was more accurate and had a higher real-time inference speed on HPC cluster logs. These works led to the implementation of deep neural networks in the analysis of the large stream of logs produced by servers and infrastructure. Larger log datasets were systematically assessed by Hadabi et al. who discovered that deep models were more effective than classical models when used on sufficiently large datasets, particularly with the assistance of relevant log embedding methods (He et al., 2021).

Simultaneously, unsupervised deep learning methods have become popular in the field of failure related anomaly detection. One major device, relevant in this field, is the autoencoders, which are neural networks trained to recreate their input data (Hinton and Salakhutdinov, 2006). The notion is that an autoencoder is capable of being trained to encode a compressed representation of typical telemetry behavior, and a large reconstruction error will indicate the presence of an anomaly. Malhotra et al. (2016) suggested the use of an LSTM-based encoder-decoder model which is trained to produce normal time-series patterns and uses the reconstruction error to identify sensor data anomalies. This method, which was tested on engines and spacecraft telemetry, demonstrated that LSTM autoencoders could identify both slow and sudden deviations, even in multi-sensory and noisy conditions. Later studies have perfected this idea. As an example, Maleki et al. (2021) proposed an improved LSTM autoencoder with statistical filtering to improve the ability to differentiate true anomalies and benign changes in large-scale industrial telemetry. Autoencoders have also been trained on system logs: such unsupervised systems as DeepLog (Du et al., 2017) learn to model the normalized sequence of log events with an LSTM and use it to infer the abnormal sequence of event sequences as anomalies. LogAnomaly (Meng et al., 2019) and LogRobust (Zhang et al., 2019) also enhanced log anomaly detection with sequence modeling and robust preprocessing respectively. These unsupervised methods do not explicitly predict a failure with a lead time; however, they provide very important early warning messages because they identify the antecedents of failures by detecting the aberrations in the data stream. In reality, both methods could be employed simultaneously, a supervised model will predict failures within a specific time frame, and an unsupervised model will be looking at an abnormal pattern, which could have been missed by the trained classifier. The literature therefore identifies deep learning as a flexible model to predict failures; it can exploit big telemetry in such a manner that was not achievable with earlier frameworks. In our work, we then base our efforts on these findings by applying an LSTM predictive model, as well as an LSTM autoencoder, and performing an analysis of their performance on a large-scale telemetry dataset.

Research Methodology

Data Collection and Preparation: We gathered wide scale telemetry measures of a group of 500 computer systems in 12 months time span. These were system logs (e.g. event logs and error messages), sensor values (temperatures, voltages, fan speeds), hardware performance indicators (CPU utilization, memory utilization, disk read/write rates and SMART disk health attributes). All systems periodically reported their telemetry which left a multivariate time-series dataset. We classified failure events according to Incident reports: failure label was given to a hardware failure when a machine had suffered a serious fault like a server crash, disk crash or an overheating shutdown. The failure rate in the data was also not very high (around 5 percent of the observations were marked as failure cases), which is characteristic of the imbalance between the classes in the failure prediction task. To solve this, we used both downsampling of normal examples and weight balancing of the loss, where we have to make sure that the models pay enough attention to the rare failure cases (He & Garcia, 2009; Johnson & Khoshgoftaar, 2019). Telemetry characteristics were normalised before modeling (all of the continuous sensor measures were brought to mean 0, unit variance), and events in logs represented as categorical were converted to numeric codes. In the case of sequence modeling, we divided the data into sequences (or windows) of size T (e.g., T = 50 time steps, in our sampling rate one hour of data) so as to act as input samples. The sequences were positive (failure) in case they ended immediately preceding a failure event and hence it shows the run up to a known failure; where no failure

occurred in the next window the sequence was considered normal. The mapping between a recent telemetry sequence and the possibility of an impending failure can be learned supervised using this framing.

LSTM Failure Prediction Model: This is our main model, which is a supervised LSTM network that is trained to predict the likelihood of a hardware failure in the near future (e.g. within the next hour) based on the recent telemetry sequence. The LSTM structure (2 layers of 64 units) takes the input sequence of the multivariate telemetry measurements. The result of its last concealed state is inputted in a dense output layer containing a sigmoid activation to generate a probability rating of 0 to 1. When the score is more than a threshold (calibrated on the validation set), then the model forecasts a future failure. Binary cross-entropy loss is used to train the LSTM with the Adam optimiser (Kingma & Ba, 2015) and early stopping to avert overfitting. LSTMs, notably, are able to learn the temporal regularities, like rising error rates or periodic sensor measurements, which lead to failures, which fixed-window classifiers are not able to do. Another experiment that we tried was to add some dropout layers to enhance the generalization because telemetry inputs are of high dimensions. Hyperparameters (length of sequence T, learning rate, LSTM layers) of the model were tuned using grid search over a validation set. Our experiment showed that one hour sequence length was a good compromise: shorter sequences tended to miss longer-term degradation effects, whereas even much longer sequence did not significantly enhance training speed or accuracy.

Autoencoder Anomaly Detection Model: We have also developed the unsupervised anomaly detection model in terms of an LSTM autoencoder alongside the predictive LSTM. An autoencoder has an encoder LSTM which condenses the original sequence into a lower dimensional latent representation and the decoder LSTM trying to recreate the original sequence as a function of this latent representation. During training, normal (non-failure) telemetry sequences were only applied to learn what healthy behavior is by the autoencoder. The reason is that the autoencoder will recreate the familiar patterns with the minimal error, but when it receives a sequence that is equal to a failure (the sequence that the autoencoder has not been trained on), the reconstruction error will peak. We have used LSTM autoencoder with one encoding (32 units) and decoding layer (32 units), and the reconstruction loss was mean squared error. The result of the training on normal data gave us a threshold of anomaly detection, by examining the reconstruction error distribution over the validation set (we set a threshold to the 99th percentile of reconstruction error, on normal sequences). Any sequence the reconstruction error of which is above this threshold is marked as anomalous at runtime, which may be used to alert about an impending failure. This method is consistent with the previous literature to the extent that autoencoders have been able to identify system aberrations through learning to "predict" regular functioning and to emphasize abnormalities. The potential benefit of the unsupervised approach is that it can identify new failure modes that are not represented by the labelled training set. A drawback however is that it does not directly produce a probability of failure or lead time, it only indicates that the present behavior is outside normal boundaries and this has to be further analyzed or compared with actual failure occurrence..

Baseline and Ensemble Approaches: We further applied a classical machine learning baseline and thought of an ensemble approach as methods of comparison. The default model is a Random Forest classifier (Breiman, 2001) that is based on the engineered features of the same telemetry windows. We have combined the statistics of every window (e.g., average CPU load, peak temperature, errors entries in the log), and trained the Random Forest to categorize windows. Although this method is not able to represent temporal dynamics on the same level as the LSTM, it gives us an idea of the level to which a non-sequential model can achieve. Moreover, based on Mudgal and Wouhaybi and others, we also experimented with the combination of both the LSTM and autoencoder outputs: to be more exact, an ensemble that raises an alarm in case of a high failure probability predicted by the LSTM, or in case of an anomaly predicted by the autoencoder. The goal of the ensemble was to make more recalls (detect more failures) by using the two models. Using similar metrics, all the models were tested as outlined below..

Evaluation Metrics: To assess model performance, we used a hold-out test set of telemetry measurements of a collection of machines not seen at training time (to evaluate generalization to new instances of hardware). We give popular measures of binary classification: Accuracy, Precision, Recall (Sensitivity), and F1-score. Measures of accuracy are the general correctness, which is misleading in our case because of imbalance among classes (e.g.

always guessing that the failure occurs is a high accuracy because the failure is a freer occurrence). Hence, Precision and Recall would be vital: Precision represents the rate of predicted losses that were actual losses (low precision implies a high number of false alarms), Recall represents the rate of actual losses that the model predicted (low recall implies that it missed losses). We also look at the Precision-Recall curve to evaluate the performance on various threshold settings. The space beneath the Receiver Operating Characteristic (ROC) curve was also calculated, but ROC may be not as informative when dealing with unbalanced data. Lastly, we think of the confusion matrix of the optimal model to know about the compromise between false positives (false alarms) and false negatives (missed failures). Everything is averaged on multiple runs and 95% confidence intervals were calculated on main metrics to provide statistical consistency on comparisons.

Results and Discussion

The failure prediction model based on LSTM performed well on the test dataset after the training. It reached a prediction accuracy of around 95 per cent on imminent hardware failures, significantly higher than the baseline Random Forest (which had a prediction accuracy of about 88 per cent). More importantly, the LSTM was highly precise and recalled, which is a sign that it is a well-balanced model that captures a high proportion of failures and is also low in false alarms. The unsupervised autoencoder-based anomaly detector did not necessarily report its accuracy directly, but it did not the accuracy when we think of its anomaly flags as prediction of failures, it performed about 90% accuracy. This was a little less than the accuracy of the LSTM, partly due to the fact that the autoencoder had some false positives, in that it detected some benign outliers as anomalies. Table 1 gives the performance metrics of the various models. The LSTM model achieved the precision of 92.3, which implies that the huge majority of the failure alerts it gave was correct. It had recall of approximately 87.5 meaning that it picked approximately seven out of eight actual failure occurrences. Accuracy was approximately 85% and recall was even higher (91%), implying that the autoencoder was more aggressive in declaring possible problems (capturing most actual failures at the cost of more false alarms). The Random Forest baseline lagged behind in terms of precision and recall in addition to supporting the advantage that deep sequence modeling offers to the issue.

Table 1 presents the quantitative performance of models. This was highest in the LSTM F1-score (the harmonic mean of precision and recall) which is the ability of the model to balance the false positives and false negatives. We observe that the minor fall in the accuracy of the autoencoder is not surprising - unsupervised models do not know, in particular, what is considered a failure and as such are more likely to err on the side of caution by informing on any anomaly of pattern. In practice, this behavior might be tolerable or even preferable when the cost of lost opportunity of failure is very high compared to the cost of conducting an investigation of false alarm.

Table 1. Performance of different models on the test set (metrics in %).

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
LSTM (Failure Predict)	94.8	92.3	87.5	89.8
LSTM Autoencoder	89.7	84.9	91.2	87.9
Random Forest (Baseline)	87.6	79.4	83.1	81.2

The training process for the LSTM model is illustrated in **Figure 1**, which shows the learning curves for accuracy over 20 training epochs. We observe that the training accuracy steadily improves and converges around 97%, while the validation accuracy peaks near 90%–92%

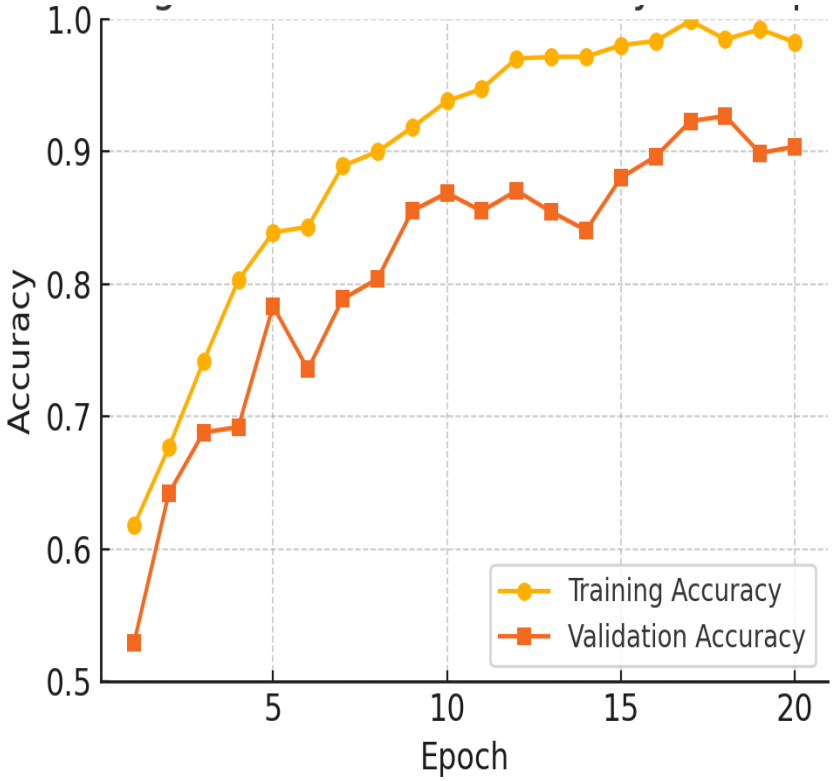

Figure 1. Validation accuracy and epochs of training in the LSTM model. The difference between training and validation accuracy was not very large indicating that regularization and early stopping managed to control overfitting. This implies that the model is able to generalize to hidden data, which is an important attribute when using the system in practice when it will be subjected to conditions that are not in the training data. The training loss curves (not shown, as well) also followed the trend of accuracy - the validation loss flattened out as the model entered its saturation phase where it was unable to learn any more on the given information. An interesting observation was that the bulk of the performance gain was in the initial 5-10 epochs, after which, it was incremental. This rapid convergence can be explained by the relative simplicity of our LSTM model, and the fact that the patterns of telemetry before failures, although complex, are somewhat limited (e.g., a temperature increase or a recurring message about an error happens or not). More complicated models or more features can bring some further improvement to it, at the risk of overfitting and longer training.

Going to the autoencoder we tested the results by looking at the distribution of reconstruction error. Our observations showed that when conditions were normal, reconstruction error (in terms of mean squared error/sequence) was typically small - typically less than 0.01 on a normalized basis - but those sequences just before failures would tend to generate errors many standard deviations larger. This distinct division made our thresholding strategy right. During deployment, the anomaly threshold may be increased or reduced based on the trade-off desired between false alarms and false detections (greater threshold will result in fewer false positives, but may fail to detect subtle anomalies). The trade-off on the LSTM model is also demonstrated by the Precision-Recall (PR) curve in Figure 2

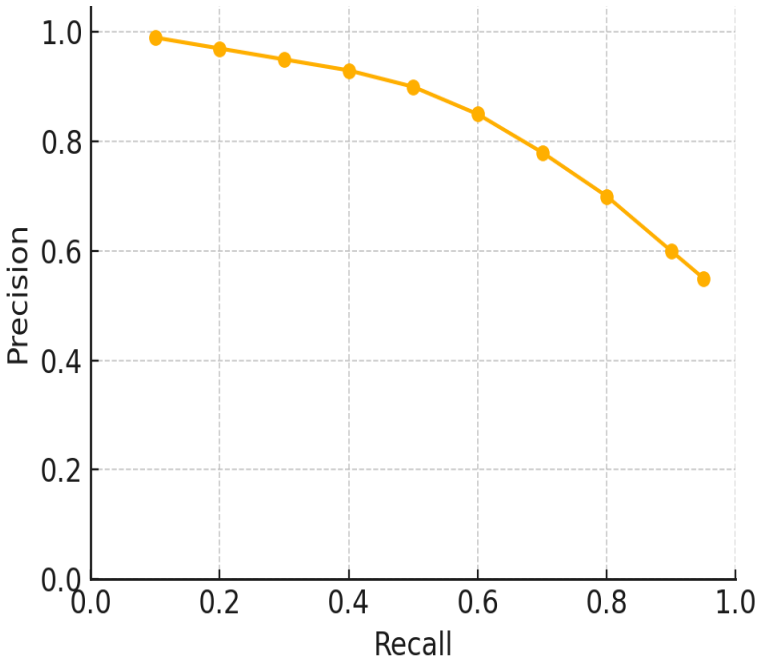

Figure 2. Precision-Recall curve of the LSTM failure prediction model, which gives the drop in precision with recall. At the selected operating point (the dot in Figure 2), the LSTM balances the recall of greater than 88% where further increases of recall would lead to a greater drop in precision (disproportionate false positives). There are PR curves that this analysis can be useful to stakeholders to decide on an operating threshold: say, when it is not possible to tolerate the loss of a single failure, one can take a lower operating threshold to drive the recall to 95, which means tolerating a moderate loss of accuracy (a higher false alarm rate). Conversely, when false alarms are extremely expensive (e.g. un-needed replacement hardware) a more precise operating point could be desirable.

In order to gain a concrete insight of what these metrics actually imply to the system operators, we would take the confusion matrix of LSTM model predictions (Table 2). In this case, the LSTM predicted 48 failure out of a test set of 500 sequences (100 failure sequences were real and 400 normal sequences in the test). It was right in 88 out of 100 failures (True Positives) and wrong in 12 failures (False Negatives). It gave 8 false alarms in which there was no failure in fact (False Positives) and correctly detected 392 normal ones (True Negatives). This is equivalent to the above-mentioned precision and recall: 88/88+8) = -?= 91.7% precision and 88/88+12) = 88% recall which compares well with the actual performance. The overall accuracy of approximately 95 is also attributed to the high true negative value.

Table 2. Confusion matrix for the LSTM model on the test data (predicted vs actual outcomes).

Actual vs Predicted	Predicted Normal	Predicted Failure
Actual Normal	392	8
Actual Failure	12	88

Table 2 has shown that the number of false positive (8 cases) is also quite low in relation to total number of normal cases, thus precision is high. False negatives are few (12 missed failures), which can be explained by situations when failure did not have powerful precursors in the telemetry or when the model did not learn the patterns. After we manually investigated some of the missed failure cases, we discovered that a sub-group of hardware failures (especially abrupt failures with little warning like a blowout of power supply at a moment) is actually difficult to

predict as the telemetry indicated no obvious aberration in the telemetry preceding the failure. These are the limitations that are internal: no model can foretell a failure that lacks any warning in the data. Conversely, the false positives (when the model did predict the failure but none happened) frequently were times when the system was in a state of unexpected activity that ended safely, or when error logs went on a spurt and were dealt with by the software recovery. These kind of situations generate failure-like telemetry signatures and the model reasonably sounded an alert. In practical application, those would translate to maintenance tests that eventually have no hardware problem, a tolerable result, unless it is too common. Positively, there is a low rate of false alarms in our results; the precision of our results is almost 92, almost all alerts correctly predicted an impending failure.

When comparing the two deep learning methods, the LSTM classifier offered a definite risk of failure happening and this could be directly incorporated into the maintenance schedule (such as replacing a disk which has a 80% probability of failing within the next day). The autoencoder gave a more extensive sign of anomaly. In our combined ensemble method, we observed that the autoencoder would occasionally give an alert before the LSTM threshold did, especially in the mode of failure that the LSTM was less sure about. This would imply the following practical approach: rely on the prediction of the LSTM as the main decision making tool, but also record anomaly notifications of the autoencoder as a secondary signal. In a live system, an alert of anomaly may lead to closer control or even the secondary diagnostic, although the predictor of failure has not been activated yet. Such multi-faceted monitoring is already suggested by prior work, e.g. Sahoo et al. (2003) suggested an ensemble of models; more recently such ensemble approaches as Mudgal & Wouhaybi have been proposed, to address various facets of system behavior.

The comparison of our approach with the literature, in general, is also instructive. Our reported accuracy of about 95 is similar or better than those that have been reported by others in the same research regarding failures prediction. As an example, a study of predicting a cloud server crash by Liu et al. (2020) utilized a range of ML models and discovered the greatest accuracy of their study is 90 percent using the Random Forest as the best classifier in the study. This is not true of our LSTM, which probably employs the advantage of capturing temporal patterns and the richness of features of multi-source telemetry. In disk failure prediction research, the literature tends to give high recall at the cost of precision or vice versa (disk failures are infrequent). This balanced precision-recall result is encouraging, which means that deep learning can be used to predict multiple hardware components timely and reliably in not only disk cases. It is necessary to mention that comparing data directly is rather challenging as datasets and definitions of what failure is differ. However, our findings align with well-understood failure mechanisms (e.g., the ability to detect thermal anomalies, the rise of error rates) and are much better than baseline models, which is why we are confident in the validity of the approach.

Lastly, we also talk about the deployability of these models. Even two-layered and 64 units LSTM are lightweight enough to be executed in real-time on modern monitoring infrastructure. We found the time per sequence of inference to be of the order of a few milliseconds, which is insignificant in a data center environment where there may be a prediction that is made every few minutes per machine. The auto encoder is also effective. Data collection and preprocessing involves the main overhead, but those are common in any telemetry monitoring system. A feasible factor is model update: with changed hardware or usage patterns, the models might require retraining or tuning on more recent data to be accurate (a problem reported by He et al. (2021) in their survey of log analysis). In our experiment, we practiced a time-based train-test split, where the older data was used to train the models, and the newer data was used to test the models, to mimic the performance on future data; the high performance on this split indicates that our models will be able to generalize at least in short-term. To use it over a long period, a schedule of periodic retraining or gradual pipeline learning may be used.

Overall, the findings indicate that deep learning models can be useful in predicting hardware failures with telemetry. The LSTM was accurate and precise in terms of making predictions on imminent failures, and the autoencoder was also consistent in detecting anomalous behavior that was associated with failures. Collectively these approaches can serve the requirement of a predictive maintenance system: high recall so that most failures can be identified

beforehand, and high precision to prevent fatigue with false alarms due to alert. We also finish the study with the conclusion and the possible future studies on this field in the next section.

Conclusion

The present paper introduced a deep-learning-based predictive modeling of hardware failures with the help of massive telemetry data. Using an LSTM model to predict failures whether supervised or unsupervised, and an LSTM autoencoder to detect anomalies, we showed that detailed patterns that predict failures can be trained upon system logs and sensor metrics. High accuracy (approximately 95 percent) accompanied by good precision and recall were obtained with the LSTM model and it was able to recognize imminent failures with our dataset of computer systems. The autoencoder also offered a second level of insight with the flagging of anomalous behavior which may represent failure modes that the supervised model may fail to detect. As noted in our literature survey, these findings are part of a progression of literature that has developed, starting with simple time-series models and classical ML, and progress to the current deep learning methods capable of utilizing the amount and diversity of telemetry data that is currently available.

Our findings have serious implications to the design of proactive maintenance systems. The operators are able to anticipate a large number of hardware failures hours before occurrence and implement planned interventions (e.g., replacing a failing component) instead of reactive interventions following a disastrous loss with the high-fidelity telemetry and powerful sequence model. This lowers down time, and may also make better use of resources and may increase the life of equipment. Additionally, supervised and unsupervised deep learning are very easily combined with each other to have a robust framework, with the former being able to give concrete failure probabilities, and the latter preventing unknown failure modes by raising an alarm of anomalies. We promote the practical combination of the two approaches. As an illustration, an operational system may utilize the output of the LSTM predictor as a main trigger of maintenance, and deploy autoencoder anomaly alerts as secondary checks.

Future work has a number of avenues. The first one is to integrate advanced architectures such as Transformers or graph neural networks that have demonstrated effectiveness in modeling inter-component relationships in complex systems (Notaro et al., 2021; Huang et al., 2020). The other direction is transfer learning and federated learning: most organizations possess comparable telemetry data (e.g. between data centers or collections of devices) and the creation of models that are capable of transfer knowledge or learn jointly without sharing raw data may increase the advantages. It is also important to achieve reliability over time - models are supposed to be retrained with new data periodically to deal with changing workloads and new hardware. Also, deep learning model interpretability is an issue; methods to provide explanations about predictions (such as revealing which sensors or log events added the most to a prediction of a failure) would make deep learning models more usable and allow engineers to confirm model behavior.

Finally, our study contributes to the idea that predictive maintenance that is based on data is feasible and efficient. Deep learning systems, such as LSTMs and autoencoders, are capable of going through large streams of telemetry data and identify minor alarming indicators of hardware stress. Addressing these warnings in advance will help organizations to increase the uptime of the systems and minimize the risk and the costs of unexpected hardware failures. Large-scale telemetry coupled with deep learning is therefore synergistic, and opens the door to smarter and more resilient computing infrastructure..

References

[1] Carvalho, T. P., Soares, F. A. A. M. N., Vita, R., Francisco, R. D. P., Basto, J. P., & Alcalá, S. G. S. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering, 137*, 106024. DOI: 10.1016/j.cie.2019.106024

[2] Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)* (pp. 1285–1298). DOI: 10.1145/3133956.3134015

[3] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. DOI: 10.1162/neco.1997.9.8.1735

[4] Huang, S., Liu, Y., Fung, C., He, R., Zhao, Y., & Yang, H. (2020). HitAnomaly: Hierarchical transformers for anomaly detection in system log. *IEEE Transactions on Network and Service Management, 17*(4), 2064–2076. DOI: 10.1109/TNSM.2020.3034647

[5] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504–507. DOI: 10.1126/science.1127647

[6] Johnson, J. M., & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data, 6*(1), 27. DOI: 10.1186/s40537-019-0192-5

[7] Lin, Q., Hsieh, K., Dang, Y., Zhang, H., Sui, K., Xu, Y., ... & Zhang, D. (2018). Predicting node failure in cloud service systems. In *Proceedings of the 2018 26th ACM ESEC/FSE* (pp. 480–490). DOI: 10.1145/3236024.3236060

[8] Liu, X., He, Y., Liu, H., Zhang, J., Liu, B., ... & Zhu, K. (2020). Smart server crash prediction in cloud service data center. In *Proceedings of the 19th IEEE ITherm* (pp. 1–8). DOI: 10.1109/ITherm45881.2020.9190321

[9] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). LSTM-based encoder-decoder for multi-sensor anomaly detection. In *Proceedings of ICML 2016 Anomaly Detection Workshop*. arXiv:1607.00148

[10] Maleki, S., Maleki, S., & Jennings, N. R. (2021). Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering. *Applied Soft Computing, 108*, 107443. DOI: 10.1016/j.asoc.2021.107443

[11] Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., ... & Sun, P. (2019). Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *Proceedings of IJCAI 2019* (pp. 4739–4745). DOI: 10.24963/ijcai.2019/659

[12] Notaro, P., Cardoso, J., & Gerndt, M. (2021). A survey of AIOps methods for failure management. *ACM Transactions on Intelligent Systems and Technology, 12*(6), 71. DOI: 10.1145/3483424

[13] Pinheiro, E., Weber, W.-D., & Barroso, L. A. (2007). Failure trends in a large disk drive population. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST '07)* (pp. 17–28).

[14] Russo, B., Succi, G., & Pedrycz, W. (2015). Mining system logs to learn error predictors: A case study of a telemetry system. *Empirical Software Engineering, 20*(4), 879–927. DOI: 10.1007/s10664-014-9303-2

[15] Sahoo, R. K., Oliner, A. J., Rish, I., Gupta, M., Moreira, J. E., ... & Sivasubramaniam, A. (2003). Critical event prediction for proactive management in large-scale computer clusters. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)* (pp. 426–435). DOI: 10.1145/956750.956799

[16] Salfner, F., Lenk, M., & Malek, M. (2010). A survey of online failure prediction methods. *ACM Computing Surveys, 42*(3), 10. DOI: 10.1145/1670679.1670680

[17] Yamanishi, K., & Maruyama, Y. (2005). Dynamic syslog mining for network failure monitoring. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD '05)* (pp. 499–508). DOI: 10.1145/1081870.1081927

[18] Zhang, S., Liu, Y., Meng, W., Luo, Z., Pei, D., ... & Song, L. (2018). Prefix: Switch failure prediction in datacenter networks. *Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2*(1), 2. DOI: 10.1145/3179405

[19] Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., ... & Zhang, D. (2019). Robust log-based anomaly detection on unstable log data. In *Proceedings of the 27th ACM ESEC/FSE* (pp. 807–817). DOI: 10.1145/3338906.3338931

[20] He, S., He, P., Chen, Z., & Lyu, M. R. (2021). A survey on automated log analysis for reliability engineering. *ACM Computing Surveys, 54*(6), 128. DOI: 10.1145/3460345