Forecasting performance anomalies in the cloud using deep learning

Shijing Li, Tian Lan, Hanqing Hang
George Washington University, ASCENDING LLC
{shijing, tlan}@gwu.edu, ryo.hang@gmail.com

***Abstract*—With the rapid growth of cloud systems in big data age, cloud users are worried about the security of their data as there may exist potential risks of data corruption. Thus, to predict the abnormal status of cloud instances before real failures happen, we design a two-layer machine learning system to predict instance status anomalies in ahead. We use the monitoring metrics of cloud instances to indicate the status of them. The first layer is a LSTM(long short-term memory) network, which is used to predict the future metrics of the instances. The second layer is to compare the predicted future metrics with current ones and learn to judge whether anomalies will appear in the future. Our proposed system could adapt to a wide range of anomaly prediction problems, and the two-layer structure could extend alarming time and increase prediction accuracy. We use real-world monitoring metrics obtained from Amazon EC2 instances. For all experimental settings, the accuracy of our proposed system is varying from 96% to 99%, much better than other popular algorithms with accuracy from 83% to 92%.**

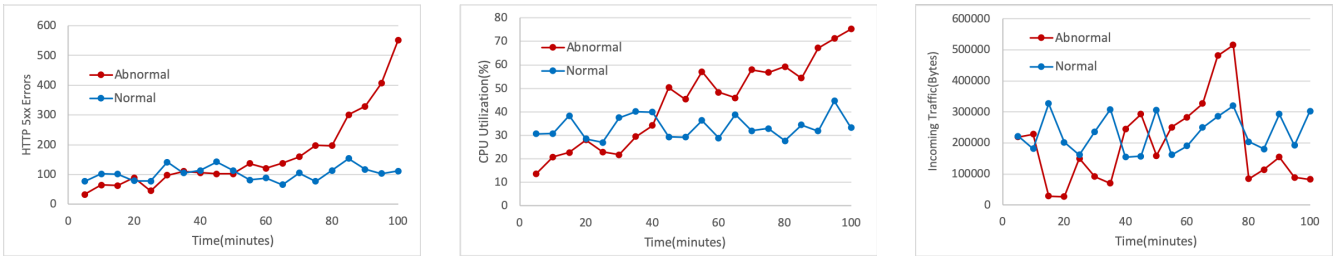
I. Introduction

In big data age, huge amount of data is being generated and transmitted to cloud servers from time to time, and cloud servers might communicate and cooperate with others to execute users' request, do distributed computing or data backup. This will lead to server failures or even crash[1], [2], [3]. Thus, users are worried about the security of their data as there may exist potential risks of data corruption, either accidently or maliciously[4]. For instance, according to the white paper from the Cloud Security Alliance (CSA), the Amazon Elastic Compute Cloud (EC2) crash caused permanent corruption of a range of data sets of various organizations[5]. Therefore, it is important to predict the abnormal status of cloud machines before real failures happen. However, most existed anomaly detection researches focused on network anomaly detection[16], [6], [17], or forecasting abnormal behaviors for edge devices[3], [8], [9], [10]. Only a few researches paid attention to forecast anomalies for cloud servers or data centers, and these works have their own weaknesses[11], [12].

In this paper, we aim to propose a novel two-layer machine learning framework for intelligent cloud performance prediction and anomaly detection for cloud servers like Amazon EC2 instances. We use Amazon CloudWatch service to collect monitoring metrics, like HTTP errors, network traffic, CPU utilization and etc., from Amazon EC2 instances, and use these metrics to predict abnormalities in the future. For each time point t, the values of metrics at time t form a metric vector. The first layer is a LSTM(Long short-term memory) network, which is used to predict the future metric vectors. The second layer is to compare the predicted future metrics vectors with current ones, and learn to judge whether abnormalities will appear. In normal cases, the values of metrics fluctuate in a relatively narrow range, but in abnormal situation, the values of metrics increase or decrease sharply. We compare the performance of our proposed system with other popular machine learning algorithms, including RF(random forest), ARIMA(Autoregressive Integrated Moving Average Model), and LR(Logistic Regression)[11], [12], [14], [15]. The accuracy of our system is varying from 96% to 99%, much better than others that are 83-92%. The novelty of our design includes that our system could adapt to a much wider range of prediction problems, and the two-layer structure could extend alarming time and increase prediction accuracy as well.

First, we explain why our system could solve a wider range of predicting abnormalities problems than existed works. We choose different combinations of sensitive metrics from all of the metrics and set an abnormal threshold for each sensitive metric. If the value of any sensitive metric is beyond the threshold, the entire metric vector indicates an abnormal status. Existed works only focuses on solving one specific abnormal situation. In [11], authors use the utilization of CPU, memory and disk to predict instance overloading. In [12], authors only predict disk failures. In [16], authors use the amount of network traffic to predict web traffic abnormalities. However, in our proposed system, if we choose HTTP errors as sensitive metrics, we could predict instance failures; if we choose CPU utilization and network traffic as sensitive metrics, we could predict instance overloads; If we choose incoming and outgoing network traffics as sensitive metrics, we could predict abnormal web traffics.

Then, in Fig. 1, we show why our design could extend alarming time and increase prediction accuracy. Here, we show one normal example (blue line) and one abnormal example (red line) by setting the value of HTTP 5xx errors larger than 400 is abnormal, which implies instance failures. We could see the red line for CPU Utilization has an increasing trending which is the same as the trending of HTTP 5xx errors. The red line for incoming traffic increases at first and has a sharp decrease which is because the instances are not available and then they refuse incoming traffic. The trend change of CPU utilization and incoming traffic is even earlier than HTTP errors. That



(a) HTTP 5xx Errors (b) CPU Utilization (c) Incoming Traffic

Fig. 1. Normal/Abnormal Example

is why other metrics help to increase the abnormal prediction accuracy. However, existed researches only use the values of sensitive metrics to predict abnormalities without any help with other metrics. This leads to relatively short alarming time and smaller accuracy. In addition, in the normal case (blue line), the values of metrics fluctuate in a higher range compared to the abnormal case. High values will also mislead other existed algorithms.

In this paper, we make the following contributions:

(1) Our design could solve a wide range of prediction problems by selecting different combinations of sensitive metrics;

(2) The two-layer structure of our design could extend alarming time and increase prediction accuracy.

(3) Our design has high prediction accuracy in evaluations. The accuracy is up to 99%, much better than other popular algorithms with accuracy from 81% to 92%.

II. Related Work

In big data age, there are a variety of anomaly detection researches focusing on network intrusion[16], [6], [17], alarming abnormal energy consumption[10], [18], or forecasting abnormal behaviors for edge devices[3], [8], [9], [19]. Only a few researches paid attention to forecast anomalies for cloud servers or data centers, and these works have their own weaknesses[11], [12].

In [20], the authors only used neural network to predict the CPU workload in data centers to improve resource provision. In [21], PRACTISE, a neural network based framework was proposed to analyze workload traces from production data centers and focus on their VM usage patterns of CPU, memory, disk, and network bandwidth. However, both of these works did not mention anomaly detection concerns.

In [11], authors showed Random Forest method could predict the overload anomalies of servers, and proposed some scheduling strategies to prevent overload, where the overload of servers was defined as the utilization of CPU and memory over 80% and disk over 75%. In [12], authors also used Random Forest based algorithm to predicts disk failures. These works only focus on one specific anomaly detection problem. However, our proposed system could predict a variety of anomaly detection problems, including overload, server failures, abnormal network traffic and so on. In addition, we show our algorithm has much better anomaly forecasting accuracy than random forest method by avoiding misdeclaration for anomalies due to misleading normal patterns.

$\boldsymbol{X}$	Set of all data vectors
$\boldsymbol{x}_t$	Metric vector at time t
Γ	Set of abnormal threshold for sensitive metrics
γ_m	Abnormal threshold for the m^{th} sensitive metric
X_t	Metric vector at time t
$x_{i,t}$	The i^{th} metric at time t
q	Number of sensitive metrics
u	Alarming time
v	LSTM prediction window size

TABLE I
Table of Symbols

III. System Description

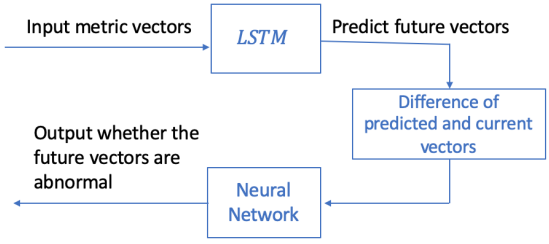


Fig. 2. Predict and Judge Ahead System

In this section, we describe the details of our two-layer machine learning architecture, shown in Fig. 2. The symbols we use are listed in Table I.

We first define our problem. Suppose we are given a metric vector set $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t, \ldots\}$, where each metric vector $\boldsymbol{x}_t = [x_{1,t}, x_{2,t}, \ldots, x_{n,t}]$ at any time t has n metrics(such as HTTP errors, network traffic, and etc.). Among the metrics, q metrics are selected as sensitive metrics. Here, $q \leq n$. Different combinations of the sensitive metrics can be used to solve different prediction problems. For example, if we select HTTP 4xx errors and HTTP 5xx errors as sensitive metrics, the system could be used to predict instance failures; if we select CPU utilization and network traffic as sensitive metrics, the system could be used to predict instance overloads. For each sensitive metric, an abnormal threshold is given, and the set of the abnormal thresholds for the q sensitive metrics is given

as $\Gamma = \gamma_1, \gamma_2, \ldots, \gamma_q$. If the value of any sensitive metric in a metric vector is larger than its abnormal threshold, the whole metric vector is seen as an abnormal vector. Alarming time u is given. For example, if u is set to 30 minutes, it means that the system will predict whether abnormalities will appear 30 minutes later. The system outputs the decision set $\boldsymbol{D} = \{d_1, d_2, \ldots, d_t, \ldots\}$, where d_t means the predicting abnormal judgement for time t. If $d_t = 1$, the system predict the future metric vector $\{x_{1,t+u}, x_{2,t+u}, \ldots, x_{n,t+u}\}$ will be abnormal, otherwise the future vector will be normal. The goal of the system is to predict the decision set $\boldsymbol{D}$.

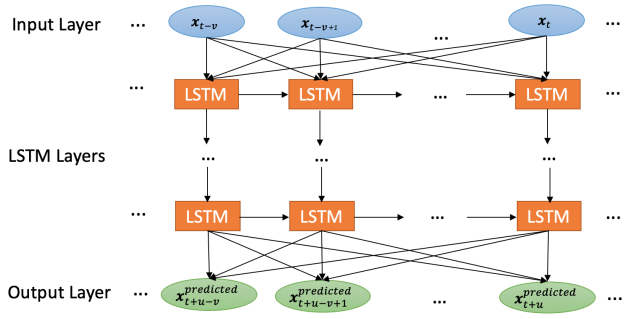


Fig. 3. Architecture of LSTM Module

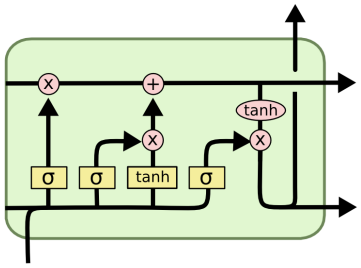


Fig. 4. Structure of LSTM Cell

Then we explain the details of the system, and the entire algorithm is described in Algorithm 1. Metric vectors $\boldsymbol{X}$ were first normalized, and then put into a LSTM (Long Short-Term Memory) module which is used to predict the future vectors. LSTM is an artificial recurrent neural network (RNN) architecture. Fig. 3 shows the architecture of this LSTM module. Fig. 4 shows the structure of orange LSTM cells in Fig. 3. The formulations of these LSTM cell can be shown as follows:

$$
\begin{aligned}
i_t &= \sigma(W^i \boldsymbol{x}_t + U^i h_{t-1}) \\
f_t &= \sigma(W^f \boldsymbol{x}_t + U^f h_{t-1}) \\
o_t &= \sigma(W^o \boldsymbol{x}_t + U^o h_{t-1} \\
\hat{c}_t &= tanh(W^c \boldsymbol{x}_t + U^c h_{t-1}) \\
c_t &= i^t \odot \hat{c}_t + f_t \odot \hat{c}_{t-1} \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

where W_i, W_f, W_o, U_i, U_o are weight matrics, $\boldsymbol{x}_t$ is the input vector for time t, h_t is the current exposed hidden state, c_t is the memory cell state, and $\odot$ is element-wise multiplication.

Due to LSTM cells, for every time step, LSTM module includes an evaluation of the current value combined with the evolution of precedent information. Depending on the strength of the information each LSTM cell receives, it will decide to block it or pass it on. This backpropagation process improves the prediction accuracy especially when we input long term data.

We implement LSTM to predict the value of future metric vectors. The architecture of the LSTM module is shown in Fig. 3. We set v as LSTM prediction window size. For each time t, metric vectors $\{\boldsymbol{x}_{t-v}, \boldsymbol{x}_{t-v+1}, \ldots, \boldsymbol{x}_t\}$ are input to LSTM layers to predict future metric vector $\boldsymbol{x}_{t+u}^{predicted} = [x_{1,t+u}^{predicted}, x_{2,t+u}^{predicted}, \ldots, x_{n,t+u}^{predicted}]$, where u is the alarming time. After training, the prediction of future metric values is very precise and reaches accuracy above 99%.

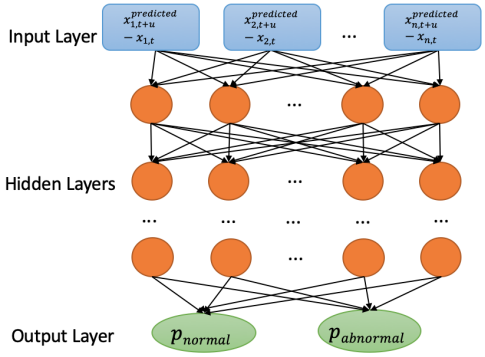


Fig. 5. Architecture of Neural Network Module

The predicted data vectors obtained from LSTM were then input to a feed-forward neural network, which is to judge whether there will be abnormalities in the future and worth warning users. The architecture of this Neural Network module is shown in Fig. 5. For each time t, the input of the neural network is $|\boldsymbol{x}_{t+u} - \boldsymbol{x}_t| = \{|x_{1,t+u} - x_{1,t}|, |x_{2,t+u} - x_{2,t}|, \ldots, |x_{n,t+u} - x_{n,t}|\}$. Here, each element of this input vector is the difference of current and predicted value of one specific sensitive metric. Each element acts as an input node of this neural network. To avoid negative inputs, we use the absolute values of them. In addition, since the metric vectors are all normalized at the beginning, the value of these input nodes are varying from 0 to 1. The output of the neural network is $y = \sigma[W(|\boldsymbol{x}_{t+u} - \boldsymbol{x}_t|) + B]$, where W and B are the weight and bias matrics of hidden layer nodes respectively. We use the commonly used activate function $\sigma = \frac{1}{e^{-z}}$, where $z = W(|\boldsymbol{x}_{t+u} - \boldsymbol{x}_t|) + B$. The neural network also decides a threshold α after enough training. If $y \geq \alpha$, the decision variable $d_t = 1$, which means the future vector $\boldsymbol{x}_{t+u}$ is judged to be abnormal; otherwise, $d_t = 0$, which means the future vector is predicted to be normal. The system will finally output a decision set $\boldsymbol{D}$ for all the time points. For normal cases, metric values always fluctuate in a

relatively narrow range in a period of time, so the difference between current and predicted vectors is relatively small. However, in abnormal cases, the difference will be much larger. After enough training, the feed-forward neural network module will learn normal and abnormal difference between the predicted and current values, and judge whether future metric vectors will be abnormal. Since the input of this neural network is based on the difference of current metric values and their future predicted values, the neural network will not be influenced by misleading normal cases where large metric values fluctuate in a narrow range on weekends. The predicted judgement accuracy of this neural network is also very precise, reaching above 96% in all experimental results described in Sec. IV.

Algorithm 1. ***PJA Algorithm***

Input: metric vector set $\boldsymbol{X}$*, alarm threshold* γ*, alarming time* u*, prediction window size* v*;*
Output: decision set $\boldsymbol{D}$*;*
Algorithm:
Normalize $\boldsymbol{X}$*;*
foreach *time* t
  foreach *time* $0 \leq t - v \leq w \leq t$
    Input $\boldsymbol{x}_w$ *to LSTM module;*
  Obtain future vector $\boldsymbol{x}_{t+u}^{predicted}$ *from LSTM module;*
  Input $|\boldsymbol{x}_{t+u} - \boldsymbol{x}_t|$ *to Neural Network module;*
  Neural Network outputs $y = \sigma[W(|\boldsymbol{x}_{t+u} - \boldsymbol{x}_t|) + B]$*;*
  if $y \geq \alpha$
    $d_t = 1$*;*
  otherwise
    $d_t = 0$.
end for
return $\boldsymbol{D}$.

These two subsystems need to cooperate to get accurate judgment for anomalies. For example, cloud instance monitoring metrics tend to have fluctuated large values on weekends, but they could not be anomalies. If we do not implement the first LSTM module, the feed-forward neural network will be easily misleaded by large fluctuated metrics on weekends and send false alarms every weekend. However, if we do not implement the second subsystem, we could only know the difference of predicted and current values of the vectors, but do not know how far of the difference will be abnormal.

IV. Evaluation

In this section, we present the results of extensive experiments that show the performance of our proposed algorithm over other popular approaches.

A. Data Collection

Our data contains Amazon CloudWatch monitoring metrics obtained from real Amazon EC2 instances that are in business use. The EC2 instances were set up in an autoscaling group containing tens of instances to cope with dynamic workloads. The Continuous Cost Optimization application running in these instances is used to help customers optimize the cost on AWS, by analyzing CloudWatch metrics, AWS billing reports, and making recommendations. Every 5 minutes, we use Amazon CloudWatch to collect 9 monitoring metrics from these instances, including CPU utilizations, network traffics, HTTP errors and etc.. The metrics are saved in gz files with timestamps. Among the 9 monitoring metrics, we select the total number of HTTP 4xx and 5xx errors appear in the 5 minutes to predict instance failures. This is because the number of HTTP 4xx and 5xx errors indicate the service of instances is unavailable or there is insufficient capacity in these instances to handle requests. Thus, every 5 minutes we could obtain one monitoring metric vector $\{x_1, x_2, \ldots, x_9\}$. We tag the whole metric vector as abnormal when the number of HTTP 5xx errors is higher than a threshold. For our experiments, we collected load balancer instance monitoring metrics in 4 months to train and test our system. In total, we collected 33738 metric vectors which include 393,642 monitoring metrics.

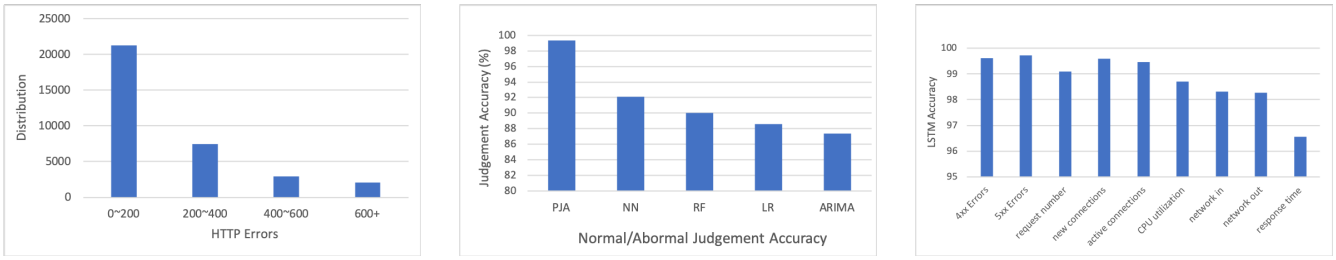
Besides HTTP 4xx and 5xx errors, other metrics have following meanings. Request number is the number of total requests processed in instances in 5 minutes. Response time is the average time elapsed, in seconds, after the request leaves an instance until a response from the target is received. We use the average of requests' response time in 5 minutes. The new connection number is the total number of new TCP connections established from clients to the instance and from the instance to targets. Active connection number is the total number of concurrent TCP connections active from clients to an instance and from the instance to targets. CPU utilization rate is the average percentage of allocated compute units that are currently in use on an instance in 5 minutes. Incoming traffic is the total number of bytes received on all network interfaces by an instance in 5 minutes. Outgoing traffic is the total number of bytes sent out on all network interfaces by an instance in 5 minutes. CPU utilization, incoming and outgoing traffics could also be chosen as sensitive metrics to predict instance overload or abnormal web intrusions.

B. Comparison Algorithms

To show the advantages of our proposed algorithm, we compare the performance of our system with 4 widely used algorithms.

RF: Random Forest(RF) is a classifier that uses multiple DTs to make a prediction, selecting the output category determined by a voting mechanism of all the individual decision trees' predictions[11][12].

ARIMA: Autoregressive Integrated Moving Average Model(ARIMA) is often used to analyze stationary stochastic processes and predict time series. The data sequence formed by the prediction object over time is considered as a random sequence that can be fitted by a class of constant coefficient difference equations. Once the mathematical model has been



(a) HTTP Errors Distribution (b) Comparing Judgement Accuracy (c) LSTM Prediction Accuracy

Fig. 6. Judgement Accuracy When Selecting HTTP Errors for Sensitive Metrics

established by the past time series, future values can be predicted from the fitted equations[11], [14], [21].

LR: Logistic Regression(LR) is a widely used classification machine learning algorithm. For problems where the samples' features can be continuous values and the range of the values is unbounded, logistic regression fits the data to the function with a binary dependent variable denoting our prediction[11][15].

NN: For comparison, we also use only neural network to forecast anomalies, getting rid of the first layer LSTM network in our design[20], [21].

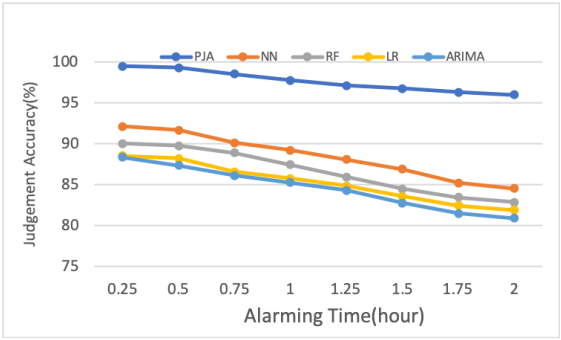
C. Training

To train our system, we divide the entire metric vectors data set into training set and testing set. Training set includes 60% of the data, and the testing set includes 40% of the data. First, we need to select sensitive metrics and set abnormal thresholds for them. By selecting different sensitive metrics, the system could solve different prediction problem. For example, if HTTP 4xx and 5xx errors are selected to be sensitive metrics, the system could predict instance failures. In our experimental evaluations, we set the thresholds for HTTP 4xx and 5xx errors as 400. The abnormal detection will become more sensitive when the threshold for HTTP errors is lower. Metric vectors with HTTP 4xx and 5xx errors larger than 400 will be tagged as abnormal.

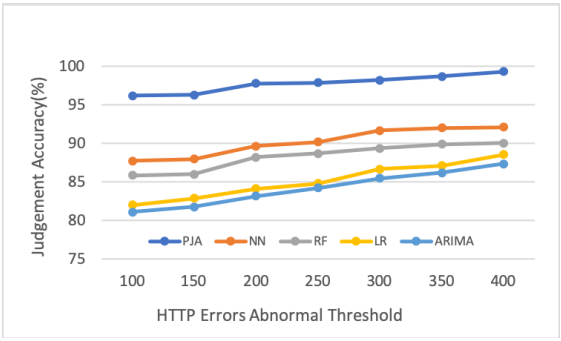
However, to train our system to predict future abnormalities, the metric vectors during alarming time before abnormal vectors also need to be tagged as abnormalities. In this way, after training, the system could predict future abnormalities that would happen, so the users would be alarmed abnormalities in ahead. We show how alarming time and abnormal threshold influences the judgement accuracy by real world experimental results in Section IV.

D. Experimental Results

In Fig. 6(a), 6(b) and 6(c), we show the accuracy of our normal/abnormal judgement prediction when setting 30 minutes as alarming time and metric vectors with HTTP 4xx or 5xx errors ≥ 400 as abnormalities, compared to other 4 approaches-NN, RF, LR and ARIMA. In Fig. 6(a), we show the distribution of HTTP errors over all data. Here we can see, in 5 minutes, it is relatively abnormal when the number of HTTP errors is larger than 400. In Fig. 6(b), we see the judgement accuracy of our PJA(predict and judge in ahead) system is above 99%, much better than other algorithms' performance with accuracy from 87% to 92%. In Fig. 6(c), we show the prediction accuracy of LSTM before the predicted vectors are input to neural network. After training, LSTM achieves prediction accuracy above 98% for 8 metrics. For response time, LSTM has 96.56% prediction accuracy. This is because response time has much larger spanning compared to other metrics. Although the prediction accuracy of response time is 96.56%, the second layer neural network could learn through training to adapt this prediction error pattern and know these cases still belong to normal situation. Thus, the normal/abnormal judgement accuracy of neural network is all above 99%.



(a) Varying Alarm Time



(b) 4xx Errors Distribution

Fig. 7. Varing Metric Settings and Sensitive Choices

In Fig. 7(a) and 7(b), we show how the metric settings influence the predicted judgement accuracy. In Fig. 7(a), when alarm time is longer, the normal/abnormal judgement accuracy

is slightly decreasing from 99.48% to 96.01%. For other algorithms, the judgement accuracy decreases much larger and the accuracy of all of the other algorithms is below 90% for longer prediction than 0.75 hour. In Fig. 7(b), we fix the alarming time as 30 minutes and change the abnormal threshold for the number of HTTP errors. It can be seen that our system performs much better to adapt to more sensitive abnormal thresholds.

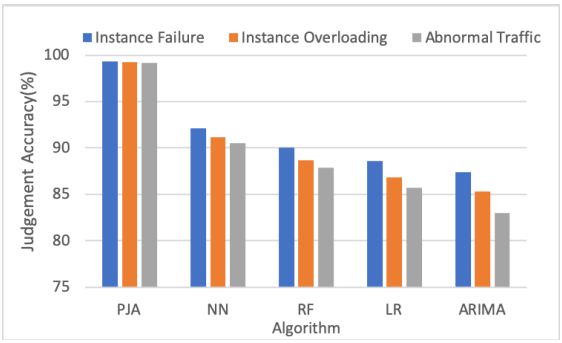


Fig. 8. Different Sensitive Metric Choices

In Fig.8, we show the performance comparison for algorithms solving different prediction problems. For predicting instance failures, we tag the metric vectors with HTTP 4xx or 5xx errors larger than 400 as abnormalities. For predicting instance overloading, we tag the metric vectors, containing incoming or outgoing traffic larger than the 80 percentile of their range, or CPU utilization is larger than 80%, as abnormal samples. For detecting abnormal web traffic, we tag the metric vectors with incoming or outgoing traffic larger than the 80 percentiles of their range as abnormalities. After training, our proposed algorithm PJA keeps judgement accuracy above 99%, beating others varing from 83% to 92%.

V. Conclusion

In this paper, we design a two-layer machine learning system to predict abnormal status of cloud servers, like Amazon EC2 instances, in ahead. We use the monitoring metrics of these instances to indicate the status of them. The first layer is a LSTM(long short-term memory) network, which is used to predict the future metrics of the instances. The second layer is to compare the predicted future metrics with current ones and learn to judge whether anomalies will appear in the future. The novelty of our design includes that our system could adapt to a much wider range of prediction problems, and the two-layer structure could extend alarming time and increase prediction accuracy as well. We use monitoring metrics obtained from real-world Amazon EC2 instances that are in business use. For all experimental settings, the predicted judgement accuracy is varying from 96% to 99%, while other algorithms only have accuracy varying from 83% to 92%.

References

[1] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[2] Y.-X. Yang, H. Li, W.-K. Zheng, Y. Bai, Z.-M. Liu, and J.-J. Zhang, "Experimental study on calcining process of secondary coated ceramsite solidified chromium contaminated soil," *Science of Advanced Materials*, vol. 11, no. 2, pp. 208–214, 2019.

[3] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2019, pp. 0305–0310.

[4] A. D. Mishra and Y. B. Singh, "Big data analytics for security and privacy challenges," in *2016 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2016, pp. 50–53.

[5] H. Yu and Z. Yang, "Decentralized and smart public auditing for cloud storage," in *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2018, pp. 491–494.

[6] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset," *IEEE Access*, vol. 9, pp. 22 351–22 370, 2021.

[7] D. K. Bhattacharyya and J. K. Kalita, *Network anomaly detection: A machine learning perspective*. Crc Press, 2013.

[8] C. V. Murudkar and R. D. Gitlin, "Qoe-driven anomaly detection in self-organizing mobile networks using machine learning," in *2019 Wireless Telecommunications Symposium (WTS)*. IEEE, 2019, pp. 1–5.

[9] K. Sultan, H. Ali, and Z. Zhang, "Call detail records driven anomaly detection and traffic prediction in mobile cellular networks," *IEEE Access*, vol. 6, pp. 41 728–41 737, 2018.

[10] A. Capozzoli, M. S. Piscitelli, S. Brandi, D. Grassi, and G. Chicco, "Automated load pattern learning and anomaly detection for enhancing energy management in smart buildings," *Energy*, vol. 157, pp. 336–352, 2018.

[11] R. Cao, Z. Yu, T. Marbach, J. Li, G. Wang, and X. Liu, "Load prediction for data centers based on database service," in *2018 IEEE 42nd annual computer software and applications conference (COMPSAC)*, vol. 1. IEEE, 2018, pp. 728–737.

[12] J. Xiao, Z. Xiong, S. Wu, Y. Yi, H. Jin, and K. Hu, "Disk failure prediction in data centers via online learning," in *Proceedings of the 47th International Conference on Parallel Processing*, 2018, pp. 1–10.

[13] E. Amazon, "Amazon ec2," *línea]. Available: https://aws. amazon. com/es/ec2*, 2018.

[14] M. Fernandes, A. Canito, J. M. Corchado, and G. Marreiros, "Fault detection mechanism of a predictive maintenance system based on autoregressive integrated moving average models," in *International Symposium on Distributed Computing and Artificial Intelligence*. Springer, 2019, pp. 171–180.

[15] W. J. Meurer and J. Tolles, "Logistic regression diagnostics: understanding how well a model predicts outcomes," *Jama*, vol. 317, no. 10, pp. 1068–1069, 2017.

[16] T.-Y. Kim and S.-B. Cho, "Web traffic anomaly detection using c-lstm neural networks," *Expert Systems with Applications*, vol. 106, pp. 66–76, 2018.

[17] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE access*, vol. 6, pp. 48 231–48 246, 2018.

[18] J. Pereira and M. Silveira, "Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 1275–1282.

[19] D. Damopoulos, S. A. Menesidou, G. Kambourakis, M. Papadaki, N. Clarke, and S. Gritzalis, "Evaluation of anomaly-based ids for mobile devices using machine learning classifiers," *Security and Communication Networks*, vol. 5, no. 1, pp. 3–14, 2012.

[20] G. M. Wamba, Y. Li, A.-C. Orgerie, N. Beldiceanu, and J.-M. Menaud, "Cloud workload prediction and generation models," in *2017 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 2017, pp. 89–96.

[21] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni, "Practise: Robust prediction of data center time series," in *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 126–134.