# A Supervised Deep Learning Framework for Proactive Anomaly Detection in Cloud Workloads

Shaifu Gupta, Neha Muthiyan, Siddhant Kumar, Aditya Nigam and Dileep Aroor Dinesh
School of Computing and Electrical Engineering
Indian Institute of Technology Mandi, Kamand - 175005, H.P., India
Email: {shaifu_gupta, muthiyan_neha, siddhant_kumar}@students.iitmandi.ac.in, {aditya, addileep}@iitmandi.ac.in

***Abstract*—Cloud environment is highly prone to failures due to its distributed nature and inherent complexity. Proactive identification of failures aids the service providers to avert these failures by taking corrective actions before they actually happen. In this paper, we analyze the resource usage patterns to identify failures due to resource contention in cloud. The resource usage and performance metrics of the working system are analyzed at regular time instants to model the normal and anomalous working behaviors. A two stage framework has been implemented where a hybrid of long short term memory (LSTM) and bidirectional long short term memory (BLSTM) is used to predict the future resource usage and performance metric values in the first stage. In the second stage, the hybrid model is used to classify the expected state as either normal or abnormal. We evaluate the proposed anomaly detection model in a virtual environment set up using Docker containers. The experimental results show that the proposed algorithm outperforms state-of-the-art algorithms.**

## I. Introduction

Failures are inevitable in any working environment. Cloud environment is vulnerable to various anomalies like performance bottlenecks and hardware / software failures. Resource contention failures are one of the most crucial failures in a cloud environment. These failures can render some of the nodes futile or can result in delay to complete different jobs. For instance, CPU bottleneck can lead to slow running of applications, memory bottlenecks can cause crashing of individual services and running out of disk space can leave the entire node unusable [1]. The presence of failures results in violation of service level agreements. Therefore, the challenge for the service providers is to detect the expected failures/anomalies before they occur so that corrective actions can be taken to steer the system away from the impending anomalies.

Different methods have been explored for failure prediction in cloud environment by identifying anomalies in the resource usage and other performance metrics. Guan et al. [2] and Gu et al. [3] use Bayes classifiers for identification of anomalies. Dean et al. [4] use self organizing maps for unsupervised anomaly detection. Most of the existing models for anomaly identification in cloud workloads assume that the observations at different time instants are independent of each other. However, as the resource contention failures develop gradually over time, therefore it is important to analyze the failures/anomalies using a model that can exploit the dependencies between these observations. A work in [5] analyzes resource usage of different jobs using recurrent neural networks to determine the termination status of different jobs as success or failure.

The anomaly identification methods are broadly classified as reactive and proactive approaches. Reactive approaches identify failures at the moment of failure. The proactive methods for anomaly detection work towards identifying the anomalies by analyzing the future resource usage values and predicting anomalies before they actually happen. In this work we focus on proactive approach for anomaly detection. The works proposed in [3], [6], use Markov chain model for future resource usage prediction to achieve anomaly detection. It has been observed in [7] that cloud workloads are long range dependent in nature. Therefore, conventional time series resource prediction models are not suitable for resource prediction in cloud. In [8], univariate LSTM models are used for resource prediction in cloud workloads. In contrast to LSTM models, bidirectional LSTM (BLSTM) models process time series variations in both directions. In this work, we propose a proactive approach where a hybrid architecture of LSTM and BLSTM models is built for the prediction of future resource usage and anomaly detection in cloud workloads. To the best of our knowledge, the use of a hybrid architecture of forward and backward network LSTMs is not reported for future resource usage prediction in cloud workloads. In this work, we propose a prediction-classification framework for proactive anomaly detection in cloud workloads. Under this framework, proactive anomaly identification is performed in two stages. In the first stage, the hybrid architecture is used to generate the future resource usage and performance metrics patterns. In the second stage, the predicted resource usage patterns are analyzed for detecting anomalous behaviors. The proposed prediction-classification framework is evaluated in a virtual environment set up using Docker containers.

The contributions of this paper are twofold. First, we propose to build a hybrid LSTM and BLSTM models for future prediction of cloud resource metrics. Secondly, we propose to build and analyze the hybrid network for supervised classification of resource usage patterns as either normal/abnormal.

The structure of the paper is organized as follows. In Section II, we review the models used for anomaly detection in cloud. In Section III, we present the dataset used in our studies. Section IV presents the architecture of the proposed models. In Section V, the results of the experiments performed for validating the proposed methods are presented. Conclusions

978-1-5386-4318-1/17/$31.00 ©2017 IEEE

and future work are presented in Section VI.

## II. Overview of existing anomaly identification models in cloud

Different methods have been explored for anomaly detection in cloud workloads. In this section, we review some of the prominent models for detection of performance anomalies in cloud workloads. The anomaly detection approaches are broadly divided into two categories, supervised anomaly detection methods and unsupervised anomaly detection methods. In supervised anomaly detection methods, labeled training data is used to accurately identify the normal and abnormal patterns in the resource usage. On the other hand in unsupervised approaches, no labeled data is used for training the models. These approaches, model only the normal behaviors in resource usage patterns and label a sample as normal or abnormal based on its deviation from the build model. These approaches can be further classified as reactive and proactive approaches. Reactive approaches identify failures at the moment of failure but proactive approaches work towards identifying the failures before they actually happen [4].

In [3], discrete Markov model is used with naive Bayesian classification to predict anomalies in performance. The feature (resource metric) $x$ is discretized into $m$ equal sized bins. A transition probability matrix, $\mathbf{P}$ of size $m \times m$ is learnt for the resource metric $x$. Let, $\mathbf{p}_t = [p_{t1}, p_{t2}, \ldots, p_{ti}, \ldots, p_{tm}]^\top$ be the state probability vector of $x$ at time $t$. Here, $p_{ti}$ is the probability of $x$ belonging to bin $i$. The $\mathbf{p}_t$ is obtained as,

$$\mathbf{p}_t = \mathbf{p}_{(t-1)}\mathbf{P} \tag{1}$$

where $\mathbf{p}_{(t-1)}$ represent the state probability vector of $x$ at time $t-1$. The predicted value is passed to the Bayesian model for classification. In a Markov model, the probability of a future value depends only on the current value and not on any past values. In order to accumulate effect of past values in the future predictions, a 2-dependent Markov model is used in [6] for future resource usage prediction. Here, the transitions depend on both the current value as well as the previous value. The naive Bayesian model used in [3] is extended to a tree augmented naive Bayes method for detection of anomalies. Both of these models use supervised learning for detecting of anomalies in cloud systems. Another work [2], utilizes an ensemble of Bayesian models for detection of failures. It learns normal behavior of the system and further detects anomalies based on its probability. An observation, $x$ is labeled as failure if $f(x) < th$ , where $f$ is the learned function and $th$ is an appropriate threshold. The system administrators verify the anomalies and the labeled data is then used in decision trees to detect the presence of anomalies. In [9], a mutual information and principle component analysis framework is used for selection of appropriate metrics for anomaly detection. They used decision tree for identification of anomalies. Guan et al. [10], proposed an approach to identify different failure behaviors in cloud systems by analyzing the correlation of failures with the principle components. They identified most relevant principle components and used Kalman filters for failure prediction. Most of the existing methods for resource usage prediction and anomaly detection assume that the observations separated by a long distance are independent of each other. But it has been observed in [7], that dependence does exist between distant observations. In [7], the presence of long range dependence is identified in cloud workloads. Long range dependence characterizes strong temporal correlations in past lags of the time series. Therefore, the conventional resource prediction models like Markov models are not suitable for future resource usage prediction. In [5], recurrent neural networks along with Hessian-free optimization are used for analyzing the resource usage of different jobs and predict their termination status as successful or failure. LSTM models [11] can model the long range dependencies in time series data. In [8], univariate LSTM models are used for resource prediction in cloud workloads. Forward forecasting models build using LSTM analyze only the progressive trends in the past history to predict future patterns. Integration of forward and backward prediction models [12], helps to learn both forward and backward dependencies in the time series.

In this work, we propose to build a hybrid of LSTM and BLSTM models for future resource usage prediction as well as supervised anomaly detection for predicting the expected anomalies. This proactive approach to detect anomalies is presented in Section IV. The effectiveness of the proposed framework is validated in a virtual environment set up using Docker containers [13]. The details of the virtual environment set up using Docker are presented in next section.

## III. Generation of dataset

In this study, we deploy a Docker [13] based virtual environment to continuously monitor the usage of different resources and performance metrics for the identification of upcoming resource contention failures. To model the real scenario, a mix of normal and failure workloads is generated. Five containers each having 2 CPUs and 1 GB RAM are used to set up the virtual environment. Stress tool [14] is used to generate different types of workloads. It forks worker processes to create a specific amount of CPU, memory, I/O, and disk burden. Different workload behaviors are generated randomly for different period of time namely, normal workload, bottlenecks in CPU, memory, I/O, disk and also combinations of different resource bottlenecks. As failures are rare in a working environment, therefore the failure load generation probability is set as $0.03$. Thus, nearly $3\%$ of the total data comprises failures. The metrics are collected from Docker statistics files, outside the containers because collecting the metrics from inside the stressed container results in the variation in the time at which the metrics are collected. A total of 17 metrics covering usage of different resources and different performance metrics are recorded. These are presented in Table I. The data is recorded by stressing the containers for 9 days and the metrics are collected every second. In the proposed work, we analyze these recorded metrics to predict future usage of resources and hence proactive identification of anomalies.

TABLE I
RESOURCE METRICS RECORDED FROM DOCKER CONTAINERS

| Metric | Description |
|---|---|
| *CPU* | CPU usage |
| *cache* | Cache usage |
| *RSS* | Memory of pages from shared libraries |
| *RSS_huge* | Memory of hugepages from shared libraries |
| *mapped_file* | Amount of memory mapped by processes |
| *dirty* | Number of dirty/modified pages |
| *writeback* | Amount of writeback |
| *pgpin* | Amount of page in |
| *pgpout* | Amount of page out |
| *pgfault* | Number of page faults |
| *pgmajfault* | Number of page major faults |
| *inactive_anon* | Inactive anonymous memory identified by kernel |
| *active_anon* | Active anonymous memory identified by kernel |
| *inactive_file* | Inactive anonymous cache memory identified by kernel |
| *active_file* | Active anonymous cache memory identified by kernel |
| *unevictable* | Amount of memory that cannot be reclaimed |
| *h_mem_lim* | Limits of memory being used by processes of any control group |

## IV. PROPOSED FRAMEWORK FOR PROACTIVE DETECTION OF ANOMALIES

An overview of the proposed framework for proactive detection of anomalies is presented in Figure 1. After collecting the resource metrics, the data is pre-processed to remove noise in the data. While pre-processing, missing values are filled by the mean of neighborhood values. Data averaging is used to reduce the effect of any sudden spikes in the recorded metrics. After cleaning, the data is passed to a feature selection component. Feature (resource metric) selection is performed for selecting the most relevant set of features and to reduce the training time of models. Correlation between different features is analyzed to filter the most significant features. Different correlation measures model distinct types of relationships from the data. Therefore, Pearson, Spearman and Kendall correlation coefficients [15] are computed and minimum redundancy maximum relevance rule [16] is used for selecting the relevant features. The correlation between features is analyzed and the features that are found insignificant from all the correlation measures are removed. In this way, the set of features are reduced to 9. The final set of resource metrics considered are: *CPU*, *cache*, *RSS*, *mapped_file*, *writeback*, *pgpin*, *pgpout*, *active_anon* and *active_file*.

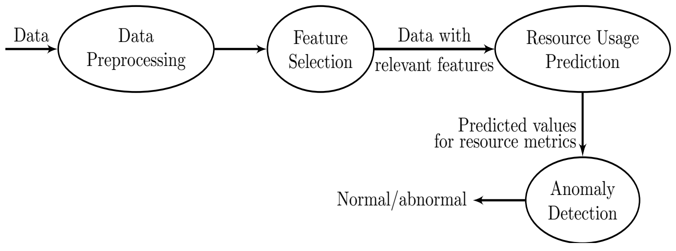


Fig. 1. Overview of proposed framework

It has been observed that long range dependence is present in cloud workloads [7]. The presence of long range dependence in the data under consideration is identified using Hurst exponent ($H$). The value of Hurst parameter is computed as $0.65$. The Hurst parameter $H > 0.5$ indicates that the data has long range dependence. LSTM models [11], have been proved efficient in dealing with data that have longterm dependencies.

### A. Resource prediction using hybrid model

We use variant of LSTM models in our prediction-classification framework for detection of anomalies in cloud workloads. A LSTM block consists of three gates. The input, output and forget gates regulate the flow of information in or out of the memory [11]. The output at any time $t$, $\mathbf{O}_t$ (predicted resource vector) is obtained as:

$$\mathbf{O}_t = \mathbf{W}^\top \mathbf{h}_t + \mathbf{b}_o \tag{2}$$

where $\mathbf{W}$ are the weights between the final hidden layer and the output. It is a $NC \times D$ matrix where $NC$ is the number of cells in the previous layer and $D$ is the feature dimension (number of resource metrics). We consider $D = 9$ as explained earlier. The $\mathbf{h}_t$ is the output of hidden layer and is of size $NC \times 1$. $\mathbf{b}_o$ is the bias of size $D \times 1$. Here, $\mathbf{O}_t = [O_{t1}, O_{t2}, \dots, O_{td}, \dots, O_{tD}]^\top$, where $O_{td}$ is the predicted value for $d^{th}$ resource metric at time $t$. The LSTM network predicts future by learning forward dependencies from $t = 1, 2, \dots, T$ in time series. On the other hand, forward and backward prediction networks integrate both forward and backward dependencies to learn the patterns. These models are also called bidirectional models. A bidirectional LSTM (BLSTM) network processes data in both directions with the backward layer iterating from $t = T, \dots, 1$ and forward layer iterating from $t = 1, \dots, T$. The output layer is updated by integrating the outputs of both forward and backward layers.

$$\mathbf{O}_t = \mathbf{W}_f^\top \mathbf{h}_{tf} + \mathbf{W}_b^\top \mathbf{h}_{tb} + \mathbf{b}_o \tag{3}$$

where $\mathbf{O}_t$ is the output at time $t$, $\mathbf{b}_o$ is the bias. $\mathbf{h}_{tb}$ and $\mathbf{h}_{tf}$ indicate the output of backward and forward hidden layers respectively and $\mathbf{W}_b$, $\mathbf{W}_f$ denote their respective weights. Intuitively, as a bidirectional LSTM model analysis both forward and backward dependencies in the resource usage patterns it is expected to be better than a unidirectional forward analysis LSTM model. However because of the addition of an extra hidden layer for processing of backward dependencies, a bidirectional model requires more computations than a unidirectional LSTM model. This makes using bidirectional LSTM at all the layers highly computationally intensive. Therefore, we propose to implement a hybrid model that uses a combination of different LSTM and BLSTM layers. Figure 2 presents the architecture of the hybrid model. This hybrid model is the hybrid of BLSTM, BLSTM, and LSTM models. It is a three-layered architecture, where a combination of both layers is explored for achieving better predictions within a reasonable computational overhead. The solid lines in the Figure represent the connections between different units of the model at a particular time step whereas the dotted lines indicate the connections between different units from one time step to other. In this architecture, we process the forward and backward dependencies in the input $\mathbf{I}_t$ at time instant $t$, using bidirectional processing layers of 128 cells in BLSTM and unidirectional processing layers of 256 cells in LSTM. As we increase the number of layers in the architecture, sigmoid activation tends to vanishing of gradients. However in rectified linear unit (ReLU), the gradient remains constant [17]. Thus, ReLU is used as the activation function. The final layer of the model is a dense layer of 9 cells where each cell corresponds to each feature dimension (number of resource metrics). In the next step, the output at time $t+1$ is generated using the input at time $t+1$, $\mathbf{I}_{t+1}$ and the previous output $\mathbf{O}_t$. This predicted resource metric vector $\mathbf{O}_t$ is now used as input to detect the working environment as either normal/abnormal.

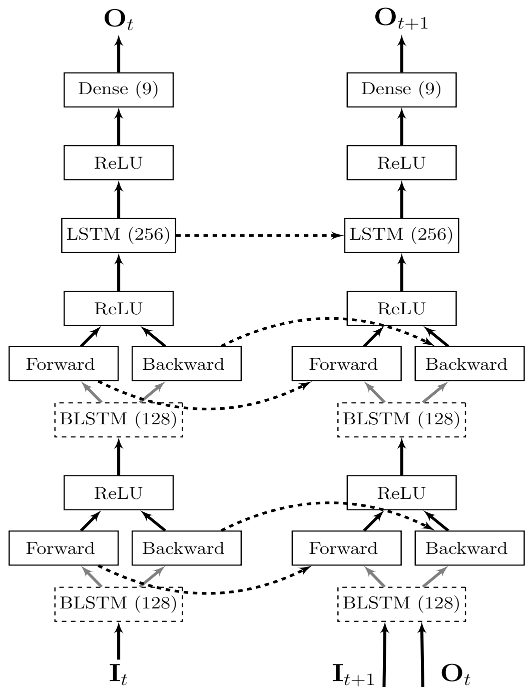


Fig. 2. Architecture of Hybrid model (BLSTM, BLSTM, LSTM)

### B. Anomaly detection using hybrid model

The resource usage patterns predicted using the hybrid model are passed to the anomaly detection module for identification of upcoming resource bottleneck anomalies. The failures are gradual in nature, the boundary between normal and abnormal behaviors is not precise. Therefore, detecting anomalies by learning a model that encompasses only normal behaviors is very difficult. Hence, we use window based supervised learning models for identification of failures. We represent each training example as a window $\mathbf{I}$, where $\mathbf{I} = (\mathbf{O}_{(t-c)}, \ldots \mathbf{O}_{(t)}, \ldots \mathbf{O}_{(t+c)})$. Here, $\mathbf{O}_t$ indicates the predicted resource vector at time $t$. Here, $t$ is the time of event (normal/failure) and $c$ denotes the number of observations considered before and after the event at time $t$. We also use the proposed hybrid architecture in Figure 2 for the detection of upcoming anomalies. A supervised two class hybrid model is trained for classifying the predicted resource usage workload as either normal or abnormal. In the hybrid model used for future resource usage prediction, linear activation is used at the Dense layer whereas for anomaly detection, the softmax activation is used at the Dense layer. The future resource predictions are continuously analyzed for the detection of upcoming resource bottleneck anomalies. Therefore, the prediction-classification architecture is designed with the motivation of serving as a proactive anomaly identification model for better resource provisioning in cloud.

## V. Results and discussions

In this section, we present the results of the studies performed for proactive anomaly detection in cloud workloads. The experiments are performed in a virtual environment set up using Docker containers. In the present study, we use 7 days data for training and validating the proposed models. Keras [18] with Tensorflow at its backend is used for the implementation of hybrid model of LSTM and BLSTM. The training is performed on NVIDIA Tesla K80 GPU. The number of layers in the models are varied from 1 to 5 and the cells in each layer are varied from 8 to 256. The look-ahead window of the resource prediction model is kept as 4 minutes and the predictions generated in these 4 minutes are passed for anomaly detection. Adam optimizer is used as an optimizer for training the hybrid model. To counter the effect of error propagation in the future resource predictions, the model is fed with the newly collected data every 4 minutes and the process is iteratively repeated for online anomaly detection.

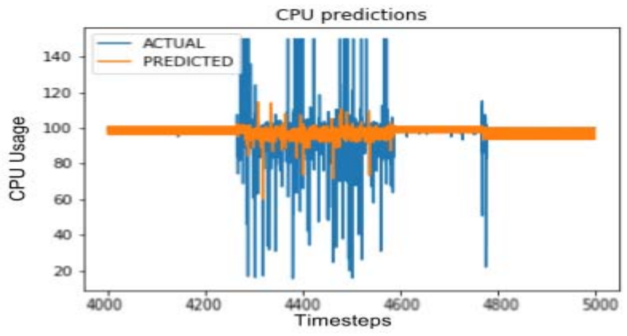
### A. Resource prediction

Table II compares the performance of different variations of hybrid model for predicting all resource metrics. We observe that the hybrid of BLSTM, BLSTM and LSTM performs the best. In this model, the data is passed directly to the bidirectional layer in contrast to a forward prediction layer in the first hybrid model. It is better to capture both forward and backward dependencies directly from the time series. The hybrid of BLSTM, LSTM and BLSTM uses the first layer as the bidirectional layer for capturing the dependencies in the data. But afterwards, it passes the learnt features to unidirectional layer before passing it to a bidirectional layer. The central LSTM layer breaks the “communication” between the two BLSTM layers and thus affecting the predictions.

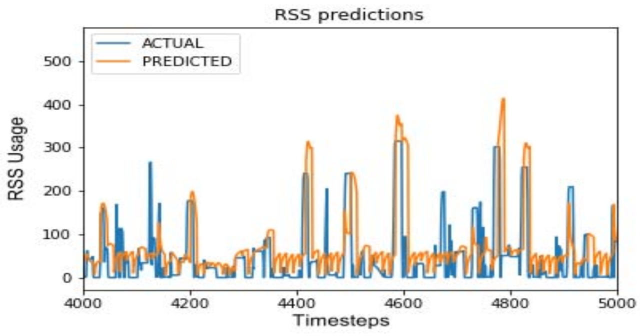
In order to compare the predictions of the proposed hybrid model, we also implement the state-of-the-art models on the data described in Section III. We compare the performance of the hybrid model with Markov model, recurrent neural network (RNN), gated recurrent unit (GRU), LSTM and BLSTM methods. Table III presents the mean square error (MSE) in the future predictions of different resource metrics. It is observed that the LSTM models perform better than RNN and Markov model. This is because unlike RNNs, LSTM models can handle long range dependencies well and do not have vanishing gradient problem. BLSTM and hybrid model perform better than other existing models. This is because these models integrate both forward and backward dependencies to learn the

TABLE II
MEAN SQUARE ERROR (MSE) OF PREDICTION OF DIFFERENT HYBRID MODELS FOR PREDICTING RESOURCE METRICS

| **Hybrid Model** | ***CPU*** | ***RSS*** | ***cache*** | ***mapped_file*** | ***writeback*** | ***pgpin*** | ***pgpout*** | ***active_anon*** | ***active_file*** |
|---|---|---|---|---|---|---|---|---|---|
| LSTM, BLSTM, BLSTM | 0.00238 | 0.02135 | 0.00079 | 0.00974 | 0.00026 | 0.00234 | 0.00473 | 0.00752 | 0.00222 |
| BLSTM, LSTM, BLSTM | 0.00573 | 0.34150 | 0.00243 | 0.00750 | 0.00091 | 0.00213 | 0.00779 | 0.02302 | 0.00197 |
| **BLSTM, BLSTM, LSTM** | **0.00250** | **0.01346** | **0.00076** | **0.00790** | **0.00100** | **0.00230** | **0.00406** | **0.00654** | **0.00185** |



(a) *CPU*



(b) *RSS*

Fig. 3. Out-of-sample predictions of different resource metrics by hybrid model

TABLE III
MSE OF RESOURCE METRIC PREDICTIONS BY DIFFERENT MODELS

| **Model** | ***CPU*** | ***RSS*** | ***cache*** | ***mapped_file*** | ***writeback*** | ***pgpin*** | ***pgpout*** | ***active_anon*** | ***active_file*** | **Training time (per epoch) in seconds** |
|---|---|---|---|---|---|---|---|---|---|---|
| Markov Model | 0.03600 | 0.04900 | 0.01900 | 0.05430 | 0.00956 | 0.0467 | 0.34090 | 0.04781 | 0.01452 | 5 |
| RNN | 0.01962 | 0.03458 | 0.00403 | 0.02082 | 0.00264 | 0.01006 | 0.11800 | 0.01183 | 0.00918 | 15 |
| GRU | 0.02413 | 0.03139 | 0.00588 | 0.01122 | 0.00607 | 0.00115 | 0.01092 | 0.00744 | 0.00185 | 50 |
| LSTM | 0.00268 | 0.02367 | 0.00086 | 0.00807 | 0.00120 | 0.00206 | 0.00507 | 0.01395 | 0.00089 | 63 |
| BLSTM | 0.00228 | 0.01572 | 0.00081 | 0.00799 | 0.00111 | 0.00189 | 0.00478 | 0.01239 | 0.0098 | 119 |
| **Hybrid model (BLSTM, BLSTM, LSTM)** | **0.00250** | **0.01346** | **0.00076** | **0.00790** | **0.00100** | **0.00230** | **0.00406** | **0.00654** | **0.00185** | **91** |

time series patterns and hence generate better predictions than other unidirectional models. Table III also shows the observed per epoch training times of different models. The hybrid model is able to capture similar performance as a 3-layered BLSTM model. However, the time taken to train the hybrid model is significantly smaller than the BLSTM model. In case of the hybrid model, the training time is reduced by 24% of the training time of BLSTM model. This results in huge savings which is really significant for a cloud environment where a large number of users dynamically enter and leave environment to access the services hosted by cloud. In such a scenario, the service providers want models to be trained accurately and fast so that they can take decisions/actions within time. In a BLSTM model, the output is obtained after the interactions between both forward and backward layers leading to more complexity and overhead in time. However, it is observed that replacing some of the BLSTM layers in the model with the LSTM layers reduces the interactions and complexity of the model and hence the training time is also reduced. Figure 3 presents the future predictions of key resource metrics such as CPU and RSS. From Figure 3, it is observed that the predictions generated by the model are very close to the actual resource consumption values. We have not presented the other predictions due to the limitation of pages. However, it is observed that the prediction generated by the hybrid model is close to actual resource consumption values for other resource metrics as well.

## *B. Anomaly detection*

The predictions generated by the hybrid model are continuously analyzed by the anomaly detection module for the detection of any anomalies. As the failures are highly uncommon in a real scenario, the instances of failure/anomaly are rare as compared to the data modeling normal behaviors. In such a scenario, the classification algorithms do not work well because they aim to minimize the overall error rate, rather than paying special attention to the minority class. Hence in order to accurately capture both normal and abnormal behaviors, class balancing is performed. Therefore, cost sensitive learning approach is used for training the model, whereby we compute the class weights and a high penalized cost is assigned for misclassification of the minority class. Table IV gives the comparisons of different anomaly detection

TABLE IV
COMPARISON OF PERFORMANCE OF DIFFERENT ANOMALY DETECTION MODELS

| Model | TPR (Recall) | TNR | FPR | FNR | Precision | F1-Score |
|---|---|---|---|---|---|---|
| SOM | 0.31 | 0.63 | 0.36 | 0.69 | 0.01 | 0.01 |
| Bayes Classifier | 0.29 | 0.93 | 0.06 | 0.70 | 0.04 | 0.07 |
| RNN | 0.96 | 0.86 | 0.13 | 0.02 | 0.10 | 0.19 |
| LSTM | 0.90 | 0.92 | 0.07 | 0.09 | 0.10 | 0.19 |
| BLSTM | 0.88 | 0.93 | 0.06 | 0.10 | 0.09 | 0.17 |
| **Hybrid model (BLSTM, BLSTM, LSTM)** | **0.94** | **0.90** | **0.09** | **0.05** | **0.12** | **0.22** |

models (i) Self organizing map (SOM) (ii) Bayes Classifier (iii) RNN (iv) LSTM (v) BLSTM and (vi) proposed hybrid (BLSTM, BLSTM, LSTM). The comparison is given in terms of different metrics, rate of correctly identified anomalies is given by true positive rate (TPR or recall), true negative rate (TNR) indicates the correctly classified normal behaviors, false positive rate (FPR) is the incorrectly captured anomalies and false negative rate (FNR) indicates the failures that are misclassified as normal. The precision and F1-Score are also reported. From the Table, it is observed that RNN, LSTM, BLSTM and hybrid model perform better than self organizing map (SOM) and Bayes method. SOM is an unsupervised anomaly identification method and does not use any labels for identification of anomalies, it performed the worst. The sequential models exploit the time dependencies between observations and hence can capture the gradually evolving anomalies. The hybrid model has a high true positive rate (TPR) and high F1-Score. As compared to the other models, it could detect the presence of most of the anomalies. Although the TPR of RNN is higher than hybrid model, but it is observed that TNR in RNN is less than the hybrid model. The precision and F1-Score of the hybrid model are also better than the RNN model. In the classification case also, the hybrid model reported lesser training time than the BLSTM model by 27%.

## VI. CONCLUSIONS AND FUTURE WORK

Proactive resource usage anomaly detection helps the service providers to provide reliable service to its users. In this paper, we proposed a supervised prediction-classification scheme where in the first step the future resource predictions are generated and in the second step predictions are analyzed for the presence of any anomalies. We evaluated the proposed framework in a virtual environment set up using Docker containers. A hybrid of LSTM and BLSTM models is proposed to predict future resource usage patterns and proactive anomaly detection in cloud workloads. The predictions of different resource prediction models viz. RNN, LSTM, bidirectional LSTM and a hybrid of LSTM and BLSTM models are compared and it is observed that the hybrid model performs best. The hybrid model generates better predictions as well as requires lesser time for training the model. We also compared the predictions of the hybrid model for supervised classification of normal/abnormal time sequences. We observed that it has a high true positive rate and can identify the presence of most of the anomalies as compared to other state-of-the art methods. In future, we plan to extend this work for identifying the root cause of the detected anomalies. Root cause analysis will helps us to find bottleneck resources that can help to avert the failures before the failure actually happens.

## REFERENCES

[1] O. Ibidunmoye, F. Hernández-Rodriguez, and E. Elmroth, “Performance anomaly detection and bottleneck identification,” *ACM Computing Surveys*, vol. 48, no. 1, pp. 4:1–4:35, Jul. 2015.

[2] Q. Guan, Z. Zhang, and S. Fu, “Ensemble of bayesian predictors and decision trees for proactive failure management in cloud computing systems,” *Journal of Communications*, vol. 7, no. 1, pp. 52–61, 2012.

[3] X. Gu and H. Wang, “Online anomaly prediction for robust cluster systems,” in *25th International Conference on Data Engineering (ICDE)*. Shanghai, China: IEEE, 2009, pp. 1000–1011.

[4] D. J. Dean, H. Nguyen, and X. Gu, “UBL: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems,” in *9th international conference on Autonomic computing (ICAC)*. New York, USA: ACM, 2012, pp. 191–200.

[5] X. Chen, C. D. Lu, and K. Pattabiraman, “Failure prediction of jobs in compute clouds: A Google cluster case study,” in *International Symposium on Software Reliability Engineering Workshops*. IEEE, 2014, pp. 341–346.

[6] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan, “PREPARE: Predictive performance anomaly prevention for virtualized cloud systems,” in *32nd International Conference on Distributed Computing Systems (ICDCS)*. Macau, China: IEEE, 2012, pp. 285–294.

[7] S. Gupta, A. D. Dileep, and T. A. Gonsalves, “Fractional difference based hybrid model for resource prediction in cloud network,” in *5th International Conference on Network, Communication and Computing (ICNCC)*. Kyoto, Japan: ACM, 2016, pp. 93–97.

[8] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, “Host load prediction with long short-term memory in cloud computing,” *The Journal of Supercomputing*, pp. 1–15, 2017.

[9] S. Fu, “Performance metric selection for autonomic anomaly detection on cloud computing systems,” in *Global Telecommunications Conference (GLOBECOM)*. Kathmandu, Nepal: IEEE, 2011, pp. 1–5.

[10] Q. Guan and S. Fu, “Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures,” in *32nd International Symposium on Reliable Distributed Systems (SRDS)*. Braga, Portugal: IEEE, 2013, pp. 205–214.

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

[12] H. Wakuya and J. M. Zurada, “Bi-directional computing architecture for time series prediction,” *Neural Networks*, vol. 14, no. 9, pp. 1307–1321, 2001.

[13] D. Merkel, “Docker: Lightweight linux containers for consistent development and deployment,” *Linux Journal*, vol. 2014, no. 239, 2014.

[14] stress(1) - linux man page. https://linux.die.net/man/1/stress.

[15] M. G. Kendall, “Rank correlation methods.” 1948.

[16] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[17] X. Pan and V. Srikumar, “Expressiveness of rectifier networks,” in *33rd International Conference on International Conference on Machine Learning (ICML)*. New York, NY, USA: JMLR, 2016, pp. 2427–2435.

[18] F. Chollet, “Keras,” https://github.com/fchollet/keras, 2015.