CLOUD FAULT DETECTION AND TOLERANCE USING INCEPTION BASED LSTM NETWORK

[1]**Neelutpol Gogoi**, Assam Down Town University
[2]**Dr.Manoj Kumar Sarma**, Assam Down Town University

Abstract:
The edge computing model The cloud has grown dramatically through using edge computing, which greatly aids IoT and mobile devices in completing complex tasks. However, fast progress leads to the disregard of security vulnerabilities in edge computing systems and associated enabled applications, that has been one of the most significant limits in smart cities. Loud infrastructures are often made up of heterogeneous servers that host several virtual machines with possibly varying specs and variable resource use. This causes several challenges with resource allocation, such as energy saving, fault tolerance, task balance, and so on. Fault tolerance within cloud computing is a major difficulty these days. The fundamental challenges with fault tolerance in cloud computing are detection as well as recovery. To address these issues, numerous fault tolerance strategies have been developed to decrease defects. As a result, the goal of this article is to develop and test an inception-based LSTM model for predicting defect and job termination status (i.e. failure or success). The proposed approach has an accuracy of 98% and a proportion of true positives of 99%. The false positive rate, on the other hand, is 12%, which is much greater than the results at the task level.

KEYWORDS: Fault tolerance, Security, cloud computing, Fault Tolerance, Failure Prediction
DOI Number: 10.48047/nq.2022.20.22.NQ10055 **NeuroQuantology2022;20(22): 730-740**

I. INTRODUCTION:

The Cloud Computing paradigm has arisen as a technology that provides its customers with a variety of computing services. Cloud computing has gained popularity in recent years owing to the benefits it offers, such as flexibility and availability in delivering computer resources at a cheaper cost [1]. With the ever-increasing demand of users for diverse resources, it is widely agreed that cloud computing is fast emerging as one of the new practical technologies. A cloud service provider must uphold service level agreements with its tenants. Because cloud infrastructure is made up of disparate components, it is certain to have flaws when they interact with one another. To overcome defects in a fast and seamless way, fault tolerance is a key duty. Because of the rise of cloud computing, delivering dependable service has become critical. However, the main disadvantage of the cloud computing paradigm is that if any of the hosts fails during data transfer, the whole process is disrupted. Transient faults may cause transient service outages and response timeouts. In cloud applications such as scientific research, financial, and safety sensitive applications, these sorts of failures might be disastrous. A fault tolerant system is necessary to mitigate the impact of such failures.

The remainder of the paper is structured as follows: The second section examines the related works of defect detection methods. Section 3 contains a full discussion of the suggested inception-based LSTM model. Section 4 discusses performance assessment and comparison of current and suggested methodologies. Finally, in section 5, the article is concluded based on the outcomes of the analysis.

II. LITERATURE SURVEY:

Hasan, M., et al. [2] presented the Flexible Fault Tolerance Framework to give user transparency in service category selection based on task completion deadline and slack time. Through flexible fault tolerance, E-FFTF effectively achieves task execution price savings. According to the experimental findings, E-FFTF is helpful to both cloud service providers and service customers. Rahman et al. [3] present an aggressive fault tolerant (AFT) approach for detecting and recovering from cloud-based failures. Using a smart decision agent, the aggressive fault detection and recovery module finds and recovers from problems. A smart decision agent makes decisions on various hardware, software, and communication flaws. To improve the cloud computing paradigm, Malik, M. et al. [4] presented a unique Hybrid Grey Wolf and Ant Lion Model (HGW-ALM) with lively standby replication (LSR). Furthermore, if any of the hosts has less capacity than its workload, the HGW-ALM model predicts that host and the LSR technique maintains the stated host. Furthermore, the tolerant mechanism rapidly processes the checkpoint strategy.

D. Saxena et al. [5] suggested an unique VM Significance Ranking and Resource Estimation based High Availability Management (SRE-HM) Model to improve service availability while minimising CDC costs. The model forecasts resource contention-based server failure and allocates required resources ahead of time to maintain the desired degree of service availability. For each VM, doing critical or non-critical activities, a significance ranking parameter is introduced and calculated, followed by the selection of an acceptable High Availability (HA) strategy based on its importance and user defined limitations. It provides CDC cost optimization by generating failure tolerance solutions for major VMs only, rather than all VMs.

A. Belgacem et al. [6] introduce a novel resource allocation model based on a multi-agent intelligent system and reinforcement learning approach (IMARM). It combines multi-agent features and the Q-learning technique to enhance cloud resource allocation performance. IMARM employs multi-agent system features to dynamically allocate and release resources, allowing it to adapt effectively to changing customer needs. Meanwhile, the reinforcement learning policy directs virtual machines to the optimal possible state based on the present state environment. A. Raj et al. [16] provide a framework for a real-time system event analysis user interface for log monitoring. Our user interface architecture provides a real-time, organised view of log events. After being processed and assessed, the logs are inspected on our UI interface. T. Asmawi et al. [7] present a thorough comparison and model assessment for prediction models of work and task failure. These models are constructed and trained using five typical machine learning algorithms and three deep learning algorithm versions. For training and testing the models, we utilise a benchmark dataset called Google Cloud Traces. F. Asadova et al. [8] created monitoring systems to watch the activity of cloud system components (e.g., nodes) as well as the served applications in the virtual environment. Most cloud environments now give graphics accelerators to their customers, which causes a variety of issues. However, the use of GPUs in deep

learning may aid in the identification of inappropriate behaviour. S. Bharany et al. [9] address the taxonomy of mistakes, faults, and failures and evaluate approaches for tolerating defects in cloud computing systems. Furthermore, the purpose of this study is to explore a number of essential research areas and sophisticated methods, such as artificial intelligence, deep learning, the Internet of Things, and machine learning, that might be used as an intelligent fault tolerance approach in the cloud environment.

A. Kumar et al. [10] suggested an intrusion detection strategy based on Fuzzy Min Max Neural Networks-Based Intrusion Detection System (FMMNN-IDS) that uses deep learning techniques. To find the min-max points, the fuzzy min-max learning algorithm is utilised, which is an expansion-contraction approach that can learn nonlinear class boundaries in a single pass through the data and allows for the incorporation of new and improved classes without retraining. Furthermore, the paper evaluated the model's performance in binary and multiclass classification, as well as how the number of neurons and learning rate impact the model's performance.

III. METHODOLOGY

In public cloud computing, there is a greater possibility that anything may go wrong as compared to private cloud computing. In the past, a number of vulnerabilities were uncovered in prominent public clouds, such as those operated by Amazon and Google. In 2013, Amazon's services were unavailable for around one hour, resulting in losses of approximately $5 million USD. In the next year, many of Google's services, including Calendar, Google Docs, and Gmail, failed to improve, which led to a significant drop in revenue totaling millions. As a direct result of this, in 2015, a number of services provided by Microsoft Azure, such as virtual servers, were unreachable in different regions throughout the globe. According to Infoworld (2016), the cost of rectifying an issue found in a cloud-based system may approach fifty thousand dollars. Keeping all of these statistics in mind, a framework or strategy is required for dependable and highly available cloud services. This framework or strategy must be able to prevent the occurrence of faults in the cloud environment and, in the event that a fault does occur, it must be remedied as quickly as possible. Failures may be separated into two categories: those that are architecture-based and those that are occurrence-based [11].

Table 1 Classification of failure and its cause [11]

Type of failure	Classification	Cause of failure
Service Failure	Architecture Based	Software Failure Scheduling Failure
Resource Failure		Hardware Failure
Correlated Failure	Occurrence Based	Based On Two Temporal Or Spatial Correlation Of Two Failure
Independent Failure		Denser System Packing Human Error Heat Issue

Our work may be connected with a service failure, the primary cause of which is driven by scheduling failure. This is because we are worried about the termination condition of the job and the task.

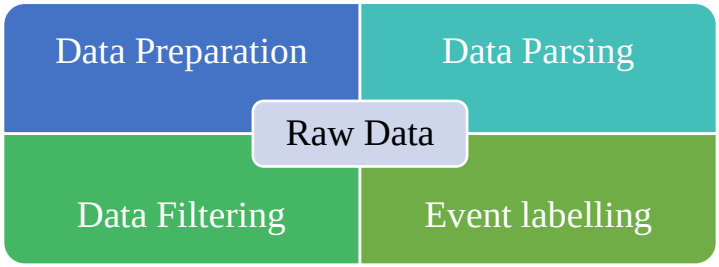

Figure 1: Architecture of Data Handling

Due to insufficient, fading error backflow, learning to retain knowledge over extended time intervals with recurrent backpropagation takes a long time. We briefly review Hochreiter's (1991) investigation of this problem before resolving it using long short-term memory, a novel, efficient, gradient-based method (LSTM).

Traditional RNN, on the other hand, suffers from a major disadvantage when dealing with data having long-term dependencies. Error signals may exhibit exponential decay when they are backpropagated over time, resulting in long-term signals being basically lost as they are drowned by un-decayed short-term signals [A13]. To overcome this issue, we chose the LSTM network, which can simulate temporal interactions between hidden states while capturing long-term dependencies.

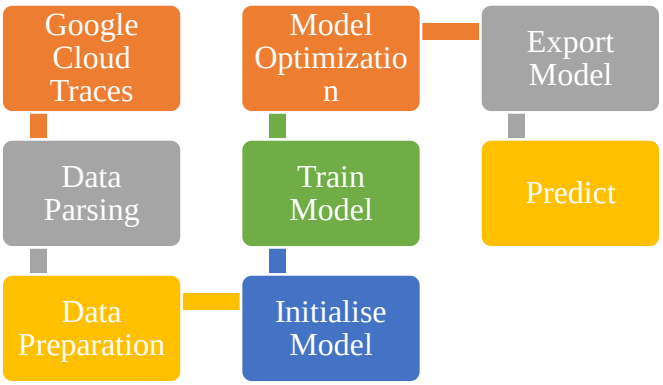

Fig. 2: Proposed Methodology

The weights change slowly during training, encoding general knowledge about the data. The LSTM model introduces an intermediate type of storage called memory cell. Unlike the standard RNN with a hidden layer, in LSTM, each ordinary node in the hidden layer is replaced by a memory cell. Each memory cell is a composite unit built from simpler nodes that are connected through some multiplicative nodes. A standard memory cell of a LSTM consists of the following elements.

In the Google cluster trace, original data tables of system and application metrics cover task resource usage measures and various attributes of the jobs, tasks, nodes and users in separate files.

Thus we get the job and task level structured data along with their respective termination statuses (i.e., failed or finished)

Let's call the first outcome from the deep learning module as $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_i, \cdots, \boldsymbol{y}_N]$using Nframes in total (whether test or training data), and the posterior probability of C classes at the frame level; that is to say, $\boldsymbol{Y} \in R^{C \times N}$.Similarly, this is what we get for subsystem two $[\boldsymbol{z}_1, \cdots, \boldsymbol{z}_i, \cdots, \boldsymbol{z}_N] \in R^{C \times N}$.(for the data corresponding to about 1.2GB, we get $\boldsymbol{C}$ equal to 40 and $\boldsymbol{N}$ equal to 12343). We are able to create the output of the simulation model at each frame $i = 1,2,3, \ldots, \boldsymbol{N}$to be achieved by the use of linear ensemble learning. Because of this, we are able to generate the outcome of the integrated system.

$$\boldsymbol{V}\boldsymbol{y}_i + \boldsymbol{W}\boldsymbol{z}_i \in R^C \quad (1)$$

a series of which, beginning with$i = 1,2,3, \ldots, \boldsymbol{N}$, is what is given into the system during testing to construct the fault event sequences. The two matrices, $\boldsymbol{V} \in R^{C \times C}$and $\boldsymbol{W} \in R^{C \times C}$, are the unrestricted parameters that should be taught during the training process, which are outlined in the following paragraphs.

Parameter estimation

To learn, we engage in activities that are overseen by an instructor.$\boldsymbol{V}$ and $\boldsymbol{W}$. The pre-labeled class targets at the segment level in the data sets provide the supervision failure scenario for this configuration:

$$\boldsymbol{T} = [\boldsymbol{t}_1, \cdots, \boldsymbol{t}_i, \cdots, \boldsymbol{t}_N] \in R^{C \times N} \quad (2)$$

The posterior probabilities $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_i, \cdots, \boldsymbol{y}_N]$ and $\boldsymbol{Z} = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_i, \cdots, \boldsymbol{z}_N]$, make up the input classification model. The total number of attributes that are included in the training set is denoted by the letter N.

Our loss function is going to be constructed using the total square error as its foundation. It is possible for us to determine the training objective function of with the assistance of $\boldsymbol{L}_2$ regularisation.

$$E = \frac{1}{2}\sum_i \| \boldsymbol{V}\boldsymbol{y}_i + \boldsymbol{W}\boldsymbol{z}_i - \boldsymbol{t}_i \|^2 + \lambda_1 \| \boldsymbol{V} \|^2 + \lambda_2 \| \boldsymbol{W} \|^2, \quad (3)$$

where $\lambda_1$and$\lambda_2$are Lagrange multipliers that we use as two experimental hyper-parameters and tweak using training and validation data. Altering the parameters to improve (2)

$$\frac{\partial E}{\partial \boldsymbol{V}} = \boldsymbol{0} \text{ and } \frac{\partial E}{\partial \boldsymbol{W}} = \boldsymbol{0}, \quad (4)$$

what we get

$$\sum_i (\boldsymbol{V}\boldsymbol{y}_i + \boldsymbol{W}\boldsymbol{z}_i - \boldsymbol{t}_i)\boldsymbol{y}_i^T + \lambda_1 \boldsymbol{V} = \boldsymbol{0} \quad (\boldsymbol{5})$$

$$\sum_i (\boldsymbol{V}\boldsymbol{y}_i + \boldsymbol{W}\boldsymbol{z}_i - \boldsymbol{t}_i)\boldsymbol{z}_i^T + \lambda_2 \boldsymbol{W} = \boldsymbol{0} \quad (\boldsymbol{6})$$

This set of equations can be easily simplified to

$$\boldsymbol{V}(\boldsymbol{Y}\boldsymbol{Y}^T + \lambda_1 \boldsymbol{I}) + \boldsymbol{W}(\boldsymbol{Z}\boldsymbol{Y}^T) = \boldsymbol{T}\boldsymbol{Y}^T \quad (7)$$

$$\boldsymbol{V}(\boldsymbol{Y}\boldsymbol{Z}^T) + \boldsymbol{W}(\boldsymbol{Z}\boldsymbol{Z}^T + \lambda_2 \boldsymbol{I}) = \boldsymbol{T}\boldsymbol{Z}^T \quad (8)$$

In addition, an analysis provides a solution to the training issue:

$$[\boldsymbol{V}, \boldsymbol{W}] = [\boldsymbol{T}\boldsymbol{Y}^T, \boldsymbol{T}\boldsymbol{Z}^T] \begin{bmatrix} \boldsymbol{Y}\boldsymbol{Y}^T + \lambda_1 \boldsymbol{I} & \boldsymbol{Z}\boldsymbol{Y}^T \\ \boldsymbol{Y}\boldsymbol{Z}^T & \boldsymbol{Z}\boldsymbol{Z}^T + \lambda_2 \boldsymbol{I} \end{bmatrix}^{-1} \quad (9)$$

Adding three gates to the network's cell will help with the application of the idea of memory, so overcoming a drawback of the standard sequence network. If new information is presented to a cell at any point in time, the cell will generate, store, and eventually update a memory. The forget gate (f), the input I memory gate (c), and the output gate are the four gates that make up an LSTM.

It is possible to determine the new cell memory C_t, given the old memory $C_{t-1}$and the new cell memory C_t,.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (10)$$

The information to be deleted from working memory is decided upon by the forget gate.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (11)$$

The Memory Gate function is in charge of creating more potential memory.

$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (12)$$

The input gate controls how much data is transferred from the competing memory to the upgraded one.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (13)$$

The output gate controls how much data is read from the cell's memory.

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (14)$$

More precisely, the MSE is utilised in lieu of a loss function.

$$\mathcal{L}[\mathrm{Y};\Theta] = \|\mathrm{X} - \mathcal{F}[\mathrm{Y}]\|_2^2 \quad (15)$$

where.$\|.\|_2$ indicates the ℓ_2 norm, and Θ stands for the model $\mathcal{F}$ parameters that are acquired after training.

Where $\|.\|_2$ denotes the $\ell_2$norm andΘ denotes the model $\mathcal{F}$ information derived after training.

Algorithm for generating heart image using inception LSTM

Input: Google Cluster traces are collected for use in training.

$Train\ data\{\}, CNN\ Activation\ Function\{\}, Input\ Threshold, Th$

Output:

Getting Data from the Convolutional and Pooling Layer

Step 1: Configure the input block of $Train\ data\{\}$, selective $Activation\ Function\{\}$, $Epochs$,

Step 2 : $Feature\{\}\leftarrow Extract\ Features\ Using\ Conv\ \&\ Pooling\ Layer\ \{Train\ data\}$

Step 3 : $Feature\{\}\leftarrow Optimised\ Features\ Using\ Dense\ layer\ \{Train\ data\}$

Step 4 : $Training\ till\ Convergence\ reach\ \leftarrow LSTM\ layer\{\}$

Testing

Input: $Test\ data\{\}, Input\ Threshold, ThTrained\ Model$

Output: $Result\ Predicted\ Class$

Step 1: Read all t $Test\ data\{\}$, using below function for validating to training rules,

$$test\_Feature(data) = \sum_{m=1}^{n}(.\ Attribute\_Set[A[m] \ldots\ldots A[n] \leftarrow Test\_Data) \quad (16)$$

Step 2 : Choose Parameters from the Set of Extracted Attributes

$Extract\ Features\ Using\ Conv\ \&\ Pooling\ Layer$ and generate feature map using below function.

Test_Feature_Map [t……..n] = $\sum_{x=1}^{n}(t)\leftarrow test\_Feature$(x)

Test_FeatureMap [x] are the selected features in pooling layer.

Features are taken from the input by the convolutional layer, then passed on to the pooling layer, and finally saved in Test Feature Map.

Step 3: Now read entire taring dataset to build the hidden layer for classification of entire test data in sense layer,

$$train\_Feature(data) = \sum_{m=1}^{n} (.\ Attribute\_Set[A[m] \dots\dots A[n] \leftarrow Train\_Data) \quad (17)$$

Step 4 : This dataset will be used to generate a training map, which may then be used to generate a final map.

Train_FeatureMap [t……..n] = $\sum_{x=1}^{n}(t) \leftarrow train\_Feature$(x)

Train_FeatureMap[t] is a map of the hidden layer used to create a feature vector for use in the hidden layer's construction. That does an in-depth analysis of all test cases using the training data.

Step 5 : Once the feature map is created, the similarity weight between features in the dense layer and those in the pooling layer is determined for each and every occurrence.

$$Gen\_weight = CalcWeight\ (Test\_FeatureMap\ ||\ \sum_{i=1}^{n} Train\_FeatureMap[i]) \quad (18)$$

Step 6 : Evaluate the current weight with desired threshold

$$if(Gen\_weight >= qTh)$$

Step 7 : Out_List. add (trainF. class, weight)

Step 8 : Go to step 1 and continue when Test_Data == null

Step 9 : Return Out_List

IV. DATASET:

a. Google Cluster Traces 2011 Dataset[15]:

Google released their "Google Cluster Trace [14]" in May of 2011, which was a trace of a cluster consisting of 11,000 workstations. This trace includes information about the cell's activity during the last 29 days. The purpose of this research is to investigate the use of resources and the requirements of this trace in an effort to provide insight into production traces similar to those seen in cloud-based settings. The data for this version may be divided into two categories: machine details and details on both the work and the task. This is a snapshot of the workloads being processed by eight Google Borg compute clusters in the month of May 2019. The trace contains an explanation of each and every task submission, scheduling decision, and resource utilisation information pertaining to the tasks that were carried out in those clusters.

It is based on the trace of a single cluster that was taken in May of 2011, and this has made it possible for a wide range of research to be conducted on establishing the state-of-the-art for cluster scheduling algorithms, including cloud computing, as well as hundreds of analyses and studies.

Since 2011, machines and software have developed to a more mature state, workloads have changed, and the impact of fluctuation in workload has become more apparent. Researchers are now able to study these alterations as a result of the new trail. Data about alloc sets (shared resource reservations that are being used by tasks), job-parent information for master/worker connections like MapReduce processes, and CPU utilisation information histograms for each 5 minute period are some of the additional data that are included in the new dataset. A point sample is not the only type of data that is included in the new dataset.These new traces, much like the ones that came before them, are concerned with resource requests and utilisation, but they do not contain any information about end users, the data they store, or the access patterns they have to storage solutions and other services. The trace data has been made available through Google BigQuery, which makes it possible to do extensive analyses without the need of using any local resources. This webpage provides access directions, in addition to providing a comprehensive description of the traces.

V. EXPERIMENT & RESULTS:

We used the following criteria for recommended and well-known state-of-the-art procedures in order to assess the dependability of the system that was presented. Our goal was to determine whether or not the system could be relied upon. The well-known approach that represents the current state of the art is implemented and assessed making use of precisely the same training and testing process while operating on the same dataset.

$$Accuracy = \frac{True\ Positive\ images + True\ Nagative\ Images}{Positive\ images + Negative\ Images}$$

$$Sensitivity = \frac{True\ Positive\ images}{Positive\ images}$$

$$Specificity = \frac{True\ Nagative\ Images}{True\ Negative\ Images + False\ Positive\ images}$$

$$Precision = \frac{True\ Positive\ images}{True\ Positive\ images + False\ Positive\ images}$$

a. Evaluation Metrics

Before attempting to evaluate the efficacy of a prediction system, it is essential to define the metrics that will be used for the assessment. Our objective is to develop an effective predictor that can accurately forecast failure, account for a maximum number of failure occurrences, and provide an extremely low number of false positives. In our experiment, we evaluate performance using the following list of metrics: Accuracy, Precision, True Positive Rate (TPR) or Sensitivity or Recall, False Positive Rate (FPR), True Negative Rate (TNR) or Specificity, and F-measure or F1 Score. Accuracy is the degree to which a measurement is accurate or precise.

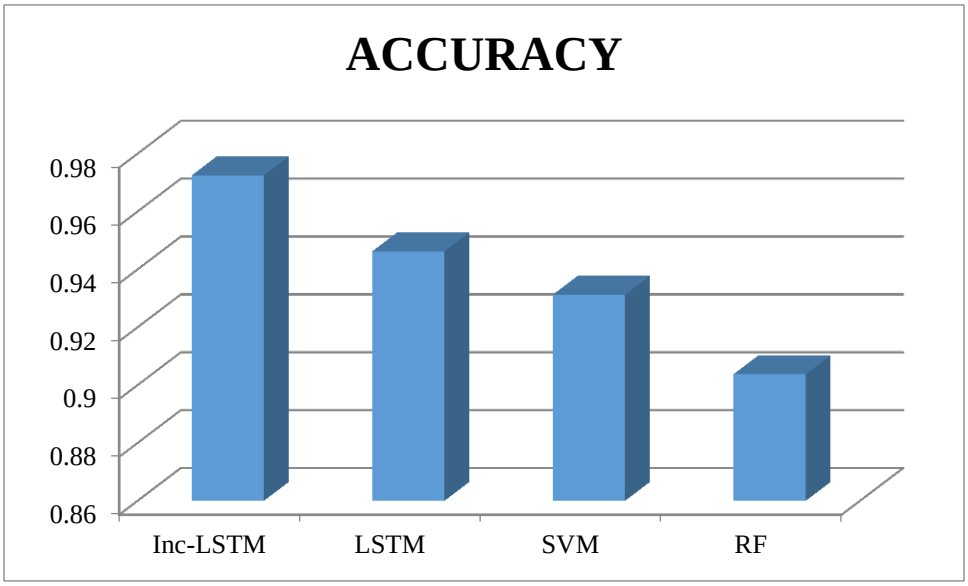

Figure 3: Comparison of accuracy of Proposed method with Existing

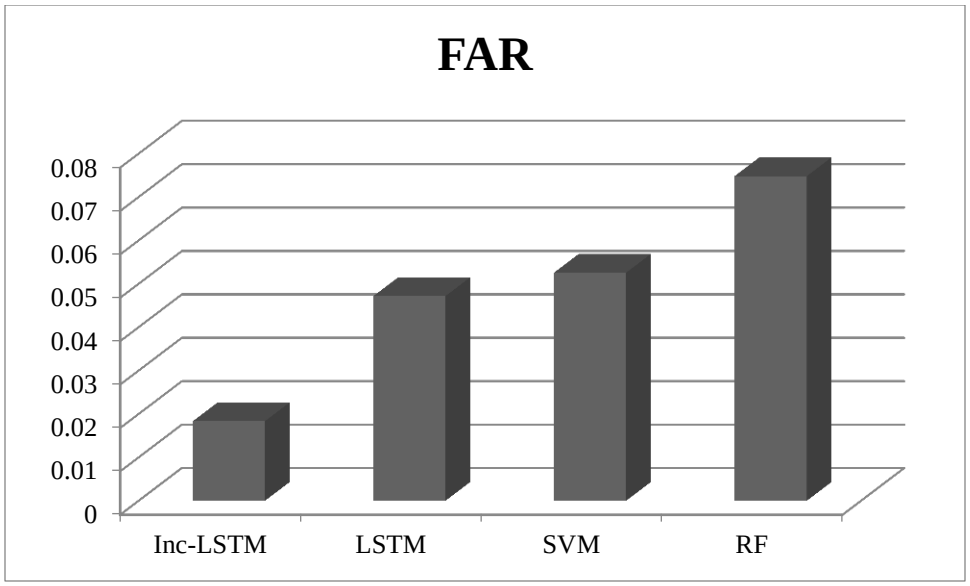

Figure 4: Comparison of FAR of Proposed method with Existing

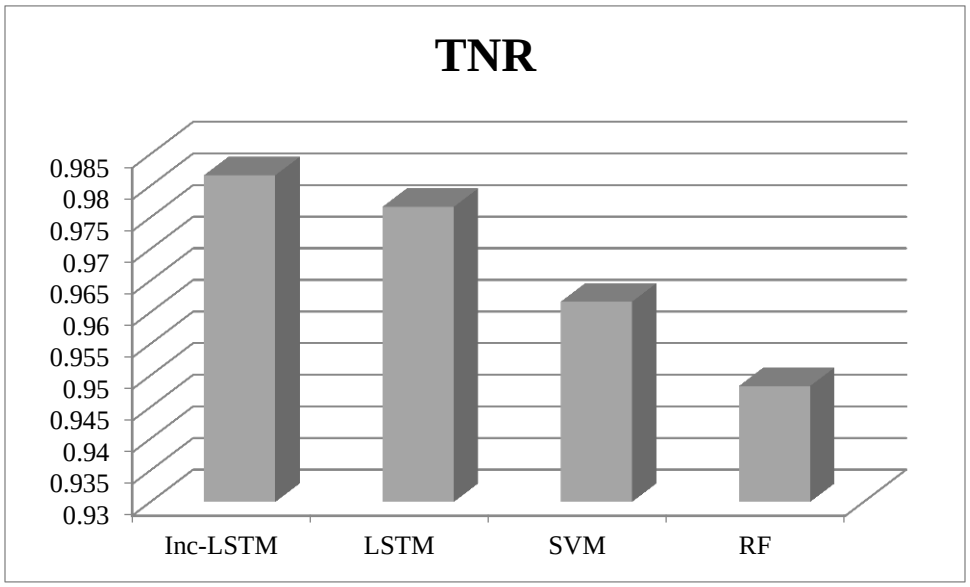

Figure 5: Comparison of TNR of Proposed method with Existing

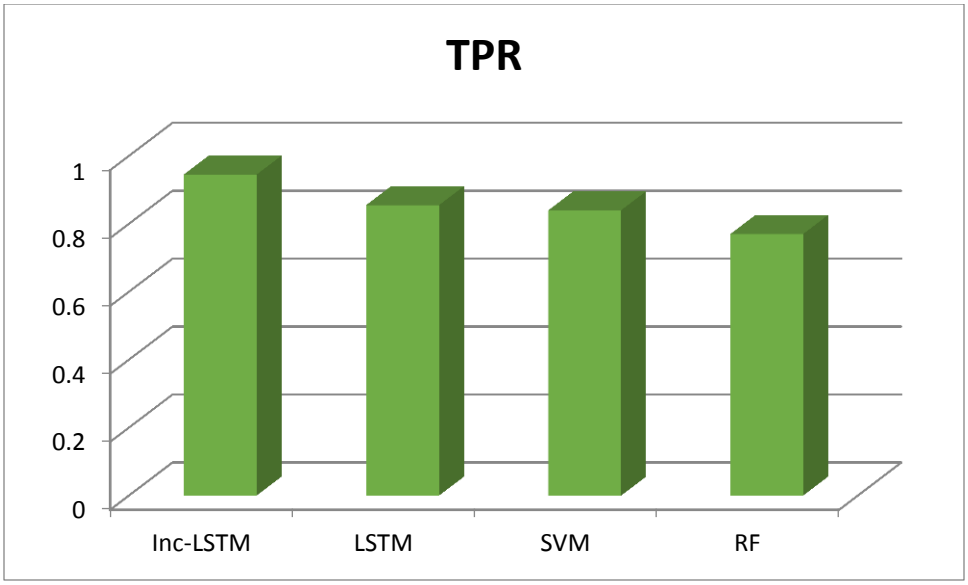

Figure 6: Comparison of TPR of Proposed method with Existing

These are the results of the prediction with 10-fold cross validation shown in Fig. We find that the accuracy is 81%, and the percentage of true positives is 83%. On the other hand, the false positive rate is 20%, which is rather higher when compared to the findings at the task level.

VI. CONCLUSION

Failure in the cloud is one of the most important problems since it may cost cloud service providers millions of dollars in addition to the loss of productivity that industrial users experience when it happens. Failure prediction is one of the approaches that may be used to avoid the occurrence of a failure, and fault tolerance management is the primary way that should be taken to handle this problem.We find that the accuracy is 81%, and the percentage of true positives is 83%. On the other hand, the false positive rate is 20%, which is rather higher when compared to the findings at the task level.

VII. REFERENCES:

[1] T. Islam, K. Lim and D. Manivannan, \Blending Convergent Encryptionand Access Control Scheme for Achieving A Secure and Storage E_cient Cloud," in Proceedings of the IEEE Consumer Communications & Networking Conference (CCNC-2020), January 2020.

[2] Hasan, M., Goraya, M.S., Garg, T. (2022). E-FFTF: An Extended Framework for Flexible Fault Tolerance in Cloud. In: Nayak, P., Pal, S., Peng, SL. (eds) IoT and Analytics for Sensor Networks. Lecture Notes in Networks and Systems, vol 244. Springer, Singapore. https://doi.org/10.1007/978-981-16-2919-8_4

[3] Rahman, M.M., Rouf, M.A. (2022). Aggressive Fault Tolerance in Cloud Computing Using Smart Decision Agent. In: Arefin, M.S., Kaiser, M.S., Bandyopadhyay, A., Ahad, M.A.R., Ray, K. (eds) Proceedings of the International Conference on Big Data, IoT, and Machine Learning. Lecture Notes on Data Engineering and Communications Technologies, vol 95. Springer, Singapore. https://doi.org/10.1007/978-981-16-6636-0_26

[4] Malik, M.K., Singh, A. & Swaroop, A. A planned scheduling process of cloud computing by an effective job allocation and fault-tolerant mechanism. J Ambient Intell Human Comput 13, 1153–1171 (2022). https://doi.org/10.1007/s12652-021-03537-7
[5] D. Saxena and A. K. Singh, "A High Availability Management Model Based on VM Significance Ranking and Resource Estimation for Cloud Applications," in IEEE Transactions on Services Computing, 2022, doi: 10.1109/TSC.2022.3206417.
[6] Ali Belgacem, Saïd Mahmoudi, Maria Kihl, Intelligent multi-agent reinforcement learning model for resources allocation in cloud computing, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 6, Part A, 2022, Pages 2391-2404, ISSN 1319-1578,https://doi.org/10.1016/j.jksuci.2022.03.016.
[7] Tengku Asmawi, T.N., Ismail, A. & Shen, J. Cloud failure prediction based on traditional machine learning and deep learning. J Cloud Comp 11, 47 (2022). https://doi.org/10.1186/s13677-022-00327-0
[8] F. Asadova, G. Kertész, R. Lovas and S. Szénási, "Fault detection in GPU-enabled Cloud Systems – An Overview," 2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI), 2022, pp. 000317-000322, doi: 10.1109/SAMI54271.2022.9780804.
[9] Salil Bharany, Sumit Badotra, Sandeep Sharma, Shalli Rani, Mamoun Alazab, Rutvij H. Jhaveri, Thippa Reddy Gadekallu, Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy, Sustainable Energy Technologies and Assessments,
Volume 53, Part B, 2022, 102613, ISSN 2213-1388, https://doi.org/10.1016/j.seta.2022.102613.
[10] Anil Kumar, Rajabov Sherzod Umurzoqovich, Nguyen Duc Duong, Pratik Kanani, Arulmani Kuppusamy, M. Praneesh, Minh Ngyen Hieu, An intrusion identification and prevention for cloud computing: From the perspective of deep learning, Optik, Volume 270, 2022, 170044, ISSN 0030-4026, https://doi.org/10.1016/j.ijleo.2022.170044.
[11] 3. Gill SS, Buyya R (2018) Failure management for reliable cloud computing: a taxonomy, model, and future directions. Comput Sci Eng 22(3):52–63
[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Comput. 9, 8 (November 15, 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735]
[13] F.A. Gers, J. Schmidhuber, and F.A. Cummins. Learning to Forget: Continual
Prediction with LSTM. Neural Computation, 12(10):2451{2471, Jan. 2000.
[14] M. Alam, K. A. Shakil and S. Sethi, "Analysis and Clustering of Workload in Google Cluster Trace Based on Resource Usage," 2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES), 2016, pp. 740-747, doi: 10.1109/CSE-EUC-DCABES.2016.271.
[15] https://drive.google.com/file/d/10r6cnJ5cJ89fPWCgj7j4LtLBqYN9RiI9/view
[16] A. Raj, S. Jadon, H. Kulshrestha, V. Rai, M. Arvindhan and A. Sinha, "Cloud Infrastructure Fault Monitoring and Prediction System using LSTM based predictive maintenance," 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2022, pp. 1-6, doi: 10.1109/ICRITO56286.2022.9964554.