International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-4, November 2019

DeepNeural Network with Particle Swarm Optimization algorithm based Cloud Resources Analysis and Prediction System

N. Subalakshmi, M. Jeyakarthic

Abstract— Deep Neural Network (DNN) classifier is a DL model for categorizing the exactness of systematic scaling orders in the groupings as an Administration (IaaS) layer of cloud computing. The hypothesis in the study is that calculation precision of scaling orders can be improved by demonstrating a reasonable time-arrangement expectation calculation dependent on the presentation plan after some time. In the examination, outstanding burden was considered as the exhibition metric, and DNN were utilized as time-arrangement expectation procedures. The aftereffects of the trial demonstrate that expectation exactness of DNN relies upon there mining task at hand plan of the framework under learning. Precisely, the outcomes demonstrate that DNN has better forecast exactness in the situations with occasional and expanding remaining task at hand plans, while DNN in predicting unexpected load design. In addition, particle swarm optimization (PSO) algorithm is applied for the optimal selection of hidden layer count to resolve the classical DNN model which has the issue of trapping into local minima and the need of manual selection of hidden layer nodes. Accurately, this study proposed a DNN-PSO design for a self-versatile expectation suite utilizing an autonomic framework technique. This suite can indicate the maximum appropriate forecast technique based on the performance design, which leads to more exact forecast outcomes..

Keywords: Cloud, DNN, Hidden Layer, PSO.

I. INTRODUCTION

DL is a developing feasible structure model and in the previous time its use has expanded a great deal of acceptance. The National Foundation of Standard and Innovation (NIST) expressed the imperative appearances n-request asset pooling, estimated administrations, fast flexibility, wide system access and self-administration. Versatility illustrative of distributed computing allows clients to obtain and discharge properties on interest, which diminishes their rate by making them pay for the assets they basically have utilized [2]. These administrations is available for open cloud, limited for personal usage, or presented on a crossover cloud [3].

Manuscript published on November 30, 2019.
* Correspondence Author
Mrs. N. Subalakshmi*, Computer Science and Engineering Wing, Annamalai University, Annamalai Nagar,Tamil Nadu, India. (Email: Subhaabi12@gmail.com)
Dr. M. Jeyakarthic Assistant Director (Academic), Tamil Virtual University, Chennai, Tamil Nadu, India. (Email: jeya_karthic@yahoo.com)

This study tended to the IaaS layer of open distributed computing situations. Over-provisioning and under-provisioning and are results of, correspondingly, immersion or misuse of assets, and are among the most significant difficulties cloud customers are tested with. One technique to overpowering these difficulties is to utilize an scaling framework. Scaling framework comprehends the all out presentation exchange off by over and again changing application assets dependent on its outstanding task at hand.

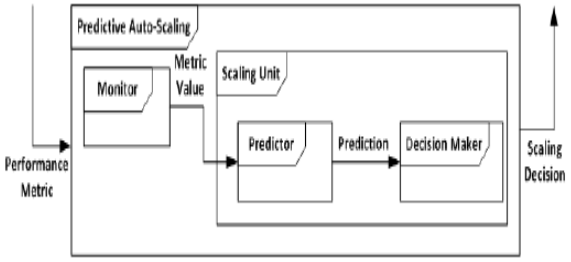

Fig. 1. Process involved in autoscaling system

As exposed in Fig. 1, Monitoring unit, Predicting unit, and Decision are the primary instruments of a prescient scaling framework. To catch late execution of distributed computing condition, scaling frameworks screen at least one execution metric(s). Here, we thought about outstanding task at hand as the presentation metric. As delineated in Fig. 1, Indicator utilizes execution metric's present an incentive from Monitor to estimate upcoming execution metric worth. Authors in [4] report that there are three trademark remaining burden designs in distributed computing situations, with each speaks to a commonplace application or situation. These examples are: unpredicted, occasional and developing. In the previously mentioned experimentation, sliding window procedure was utilized to prepare forecast calculations [5-8]. In this procedure, window size is one the most significant elements that significantly affects the forecast exactness. Along these lines, to expand classifier outcome, the impact of window size is expanded on expectation exactness in a portion of our test.

As per [6], Profound Neural Systems (DNN), Neural Systems (NN) and Bolster Vector Machine (SVM) are the best expectation calculations to anticipate future framework attributes. Hence, in this work we utilized DNN, NN and SVM calculations as the Forecast segment. The principle commitments of this work:

• Contrasting DNN and NN and SVM expectation exactness with respect to the distinctive remaining task at hand examples

• Dissecting the effect of sliding window size on DNN with NN and SVM expectation exactness

• Recommending an abnormal state plan of a self-versatile forecast suite which picks the most reasonable expectation calculation dependent on the approaching remaining task at hand example.

The rest of the examination is arranged as seeks after: Area II talks about the foundation and related work. This is trailed by investigation of DNN in segment III. Area IV is devoted to the examinations and outcomes investigation. Area V end and potential headings for the future research are talked about.

II. RELATED WORKS

A. Load

Resource allocation for batch applications is frequently mentioned to as progress which includes satisfying a particular job performance limit [4]. Scheduling is widely researched in grid situations [4] and discovered in cloud environment. In the same way, the application with unchanging (or stationary) load design does not need a scaling model to allocate resources [9]. Hence, this study focuses on application which has underlying load pattern:

- Periodic load: Indicates the load with seasonal changes. It cover cyclic/bursting load of [4] and intermittent and overall lifetime loads.
- Unpredictable load: Indicates the altering loads. It covers unpredicted load.
- Rising load: Indicates loads with recent trends. It covers expanding and regularly altering load.

B. Amazon EC2 and TPC-W

So as to create previously mentioned remaining task at hand examples, one can utilize either genuine follow records or request standards. A Complete outline of suggestion records and application benchmark[4]. Essentially, benchmarks incorporate a web application with an outstanding burden generator which makes session-based solicitations to the applications underlying test. This study utilizes Java usage of TPC-W, because of its straightforwardness and broad online documents. Now also, the analysts are using TPC-W for leading auto-scaling tests.

C. Decision Making Technique

A set of five classifications: lining hypothesis, time-arrangement examination, support learning, control hypothesis, and edge-based strategies. Among these classifications, time-arrangement examination centers around the forecast side of the asset provision process and isn't a "basic leadership" strategy. Conversely, the edge-based strategy is an unadulterated basic leadership system while the remainder of the auto-scaling classifications (control hypothesis, lining hypothesis, and fortification adapting) by one way or another plays the Predictor and Decision Maker jobs simultaneously [10].

D. Forecast Techniques

The authors in [5] have checked NN and Straight Relapse calculations to anticipate the future estimation of CPU burden and they have presumed that NN outperforms Direct Relapse as far as exactness. Furthermore, they have demonstrated that precision of the two calculations relies upon the information window size. Then again, the creators in [5] have assessed distinctive AI expectation results. The creators have considered SVM, NN and Direct Relapse. They have checked the forecast aftereffects of these calculations utilizing three execution measurements: CPU usage, throughput, and reaction time. This paper utilized SVM and NN strategies on the expectation process. They are the two most precise AI calculations in the scaling domain [5] and has commonly utilized in other designing domains. In section 3 measured basics of the models and their particular arrangement in our trial is displayed.

III PSO WITH DNN MODEL

DL is a subfield of Machine Learning dependent on learning various levels of representing by making a order of structures where the advanced stages are characterized from the lower levels and a similar lower level highlights can support in characterizing numerous higher level features. DL structure expands the neural network(NN) by adding progressively shrouded layers to the system design between the information and yield layers to show increasingly unpredictable and nonlinear relations. This perception enhanced the investigator's consideration in the recent years for its good performance to become the best solution in many problems in medical image analysis applications such as classification, segmentation, registration and image denoising[7,10].DNN is another DL construction that is broadly utilized for order or relapse with achievement in numerous regions.

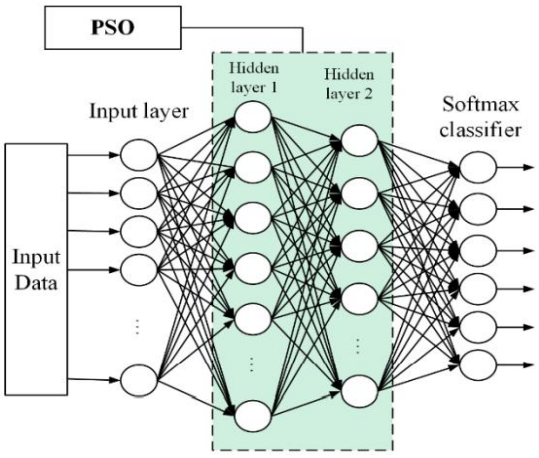

Fig. 2. PSO-DNN architecture

In addition, particle swarm optimization (PSO) algorithm is applied for the optimal selection of hidden layer count to resolve the classical DNN model which has the issue of trapping into local minima and the need of manual selection of hidden layer nodes as shown in Fig. 2.In our experiment we utilized profound neural system calculation characterized in the WEKA apparatus, known as MLP.

The parameters we utilized in our test are appeared in Table I. Parameter determination is characteristically originated on heuristics, as there is no technical equation or hypothesis that has been proposed to indicate the best parameters. We have chosen MLP parameters dependent on the best outcomes after a few preliminaries. PSO algorithm is based on the foraging nature of flocks. It produces the arbitrary solutions and determines the optimum ones with the optimal fitness value in an iterative way. This method is called as Back Propagation (BP) NN due to the fact that it possesses several advantages like easier to implement, better accuracy and faster convergence rate. In addition, it showed superior results while dealing with practical issues and been generally employed in the domain of DL. The classical kind of PSO comprises a collection of particles which have undergone communication over one another for reaching the optimal place in a repeated way. For resolving this issue, this algorithm will update diverse parameters like position, velocity and fitness value of every particle which is computed using scientific equation. The position of the particle indicates the candidate solutions to the issue required and is saved as individual best solution pibest. The alteration of position has more impact using the own optimal fitness value pfit that is a lowest value obtained in earlier rounds and move towards the global best position gbest indicating to the global fitness value gfit between every result obtained in the whole space. The PSO algorithm can be defined as follows.

$$V_i [k + 1] = wV_i [k] + c_1 rand_1 (p_{ibest} - P_i (k)) + c_2 rand_2 (g_{best} - P_i k)) \quad (1)$$

$$P_i (k + 1) = P_i (k) + V_i [k + 1] \quad (2)$$

where w is the inertia weight which assists the particles shift using the interior to an optimal position, ci indicates the constants and randi are the uniform random value, the position vector and velocity vector of the i-th particle are $P_i(k)$ and $V_i(k)$. Once the DNN is trained, the recognition takes place by the integration of PSO algorithm with DNN model. The process involved in PSO algorithm is shown in Fig. 3. The normalized feature vector is applied as utilized as the input of the network model. Next, the double hidden layer node count of DNN is fixed in an automatic way utilizing the global optimization ability of the PSO algorithm for obtaining optimum node count for improving the performance.

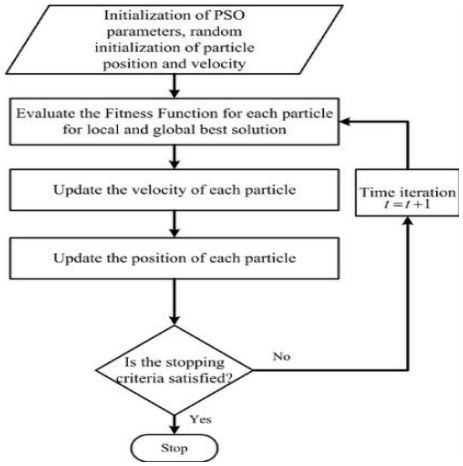

Fig. 3. Process involved in PSO algorithm

Table- I: Parameters settings

Parameter Name	Value
Hidden Layer Sizes	50
Epochs	10
Epsilon	1.0E-8
Rho	0.99

IV. PERFORMANCE EVALUATION & RESULTS

The authoritative objective of the test is enhancing forecast exactness of prescient autoscale framework. DNN are the most precise machine learning calculations [5] utilized for remaining task at load expectation. It is meant to investigate relationship among various load pattern and forecast accuracy of DNN.

A. Experimental Environment

To organize the examination condition, we passed on Java execution of Amazon EC2 benchmark on TPC-W structure. Fig. 4represents to compositional layout of our preliminary course of action. As showed up in Fig. 4, the test course of action includes 3 virtual machines. Table II presents detail of these virtual machines. It is noted lessen test unconventionality we simply observed execution of the server level and acknowledged the record isn't an issue. Therefore, an about astounding virtual machine is supposed to be committed to record level.

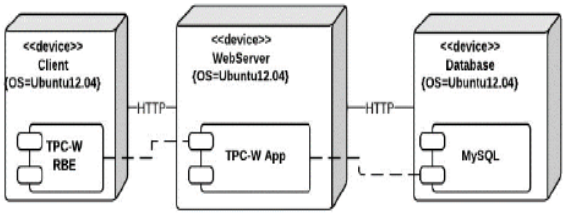

Fig. 4.Experimental setup

Table- II: Hardware specification of servers for experiment.

	Memory / Storage	Processing core
Client	1/4GB	4
Webserver	1/4GB	4 core
Storage	2/20GB	8 core

B. Proposed Methods of DNN

After producing definite assignments, the precision of DNN is anticipated in "periodic" remaining task at hand examples DNN are test of directed learning class of DL procedures. At that point, the made model is assessed on applied database. Another significant factor is information highlights. In DL an element is an individual quantifiable property of a marvel being watched. Subsequently, in this analysis – notwithstanding our principle objective (for example impact of various outstanding task at hand examples on SVM and NN forecast precision) – we examined impact of window size on the expectation exactness of DNN, SVM and NN, as well.

C Evaluation Metrics

Exactness of the outcomes can be assessed dependent on various measurements, for example, Mean Absolute

Percentage Error (MAPE). Also, R2 Forecast Accuracy is a proportion of integrity of-fit, which it's worth falls inside the range [0, 1] and is normally connected to straight relapse models. Because of the impediments of PRED (25) and R2 Forecast Accuracy, we utilized MAPE measurements. Proper meanings of these measurements are:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \frac{|YP_i - Y_i|}{Y_i}$$

Where YPi is the anticipated yield and Yi is the real yield for I-th perception, and n represents quantity of perceptions through the forecast is made. MAPE typically communicates exactness as a rate and is a famous measurement in insights, particularly in pattern approximation. Smaller MAPE qualities show a progressively arrangement.

D. Result analysis

Tables 3 demonstrate the results attained by distinct models in the forecast process of load patterns. Fig.4 compare MAPE values for proposed DNN and Existing NN and SVM in the load patterns. we have only presented MAPE values of DNN,NN and SVM in the load patterns. In addition, based on Fig. 5, for small window sizes NN holds somewhat improved forecast accurateness associated to SVM and proposed DNN (for window size = 2, MAPE NN value is 4.91, which is slightly less than MAPE SVM value that is 2.76 and MAPE proposed DNN value that is 2.19). The simulation outcome displays that DNN is better than SVM and NN.

Table- III: MAPE Values for Existing and Proposed Methods

Window Size	MAPE Proposed DNN	MAPE NN	MAPE SVM
2	2.1902	4.9108	2.7604
3	2.1890	5.1113	2.7295
4	2.1876	4.8585	2.7356
5	2.1865	4.8553	2.7391
6	2.1843	4.5919	2.7319
7	2.1790	4.6787	2.7389
8	2.1787	4.3998	2.7339
9	2.1760	4.4980	2.6765
10	2.1589	3.3097	2.6742

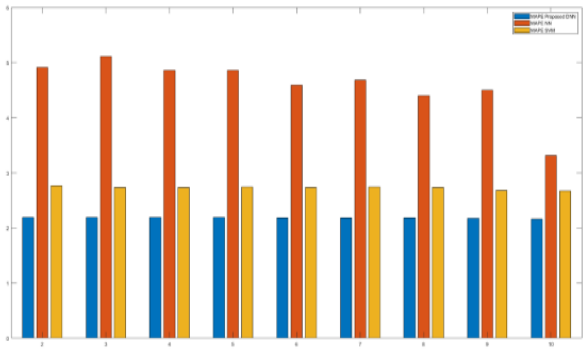

Fig. 5. MAPE Values for Existing and Proposed Methods

IV CONCLUSION

A PSO-DNN model is presented to group the prescientcaling frameworks for the IaaS layer of distributed computing. As per the hypothesis, expectation precision of prescient scaling frameworks can be improved by showing a suitable time-arrangement forecast calculation dependent on approaching incoming load pattern. In the investigation, the impact of remaining burden designs on forecast precision of DNN was contemplated by utilizing MAPE factors as exactness appraisal criteria. Our discoveries demonstrated that in the situations with "developing" or "occasional" remaining task at hand examples DNN has better expectation exactness contrasted with SVM and NN, while in the conditions with "unpredicted" outstanding task at hand examples DNN beats NN and SVM. Our outcomes likewise demonstrated that expanding the window size just has sway on the situations with "unpredicted" remaining task at hand example with increment in forecast precision of DNN. Be that as it may, in different situations (i.e., developing or intermittent remaining burden designs), expanding the window size does not improve the DNN exactness. The future scope would be contemplate the effect of the database layer and dormancy on the forecast and basic leadership exactness of the various calculations dependent on outstanding burden designs just as the distinctive window size

REFERENCES

1. P. Mell, T. Grance, "The NIST definition of cloud computing," NIST special publication, 2011, pp 800-145.
2. A. Y. Nikravesh, S. A. Ajila, C.H. Lung, "Cloud resource autoscaling system based on Hidden Markov Model (HMM)", Proc. of the 8th IEEE International Conference on Semantic Computing, June 2014.
3. A. A. Bankole, "Cloud client forecast models for cloud resource provisioning in a multitier web application environment", Master of Applied Science Thesis, Electrical and Computer Engineering Department, Carleton University, 2013.
4. T. Lorido-Botran, J. Miguel-Alonso, J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," Journal of Grid Computing, vol. 12, no. 4, December 2014.
5. A. Y. Nikravesh, S. A. Ajila, C. H. Lung, "Measuring forecast sensitivity of a cloud auto-scaling system", Proc. of the 7th IEEE International Workshop on Service Science and Systems, in collaboration with the 38th International Computers, Software & Applications Conference, July 2014.
6. S. A. Ajila, A. A. Bankole, "Cloud client forecast models using machine learning techniques," Proc. of the IEEE 37th Computer Software and Application Conference, 2013.
7. Load Patterns for Cloud Computing, 2010, [Online], Available: http://watdenkt.veenhof.nu/2010/07/13/loadpatterns-for-cloudcomputing/.
8. J. R. Williams, F. R. Burton, R. F. Paige, F. C. Polak, "Sensitivity analysis in model-driven engineering," Proc. Of the 15th International Conference on Model Driven Engineering Languages and Systems, 2012. [9] H. W.
9. Cain, R. Rajwar, M. Marden, M. H. Lipasti, "An architectural evaluation of Java TPC-W," Proc. Of the 7th International Symposium on High Performance Computer Architecture, 2001.
10. C. Fehling, F. Leymann, R. Retter, W. Schupeck, P. Arbitter, Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications, Springer, 2014.
11. Mohsen, H., El-Dahshan, E. S. A., El-Horbaty, E. S. M., & Salem, A. B. M. (2018). Classification using deep learning neural networks for brain tumors. Future Computing and Informatics Journal, 3(1), 68-71.
12. Amazon Elastic Compute Cloud (Amazon EC2), 2013. [Online], Available: http://aws.amazon.com/ec2.

13. RackSpace, The Open Cloud Company, 2012. [Online], Available: http://rackspace.com.
14. RightScale Cloud management, 2012. [Online], Available: http://rightscale.com.
15. M. Z. Hasan, E. Magana, A. Clemm, L. Tucker, S. L. D. Gudreddi, "Integrated and autonomic cloud resource scaling," Proc. Of the Network Operations and Management Symposium, 2012
16. J. Kupferman, J. Silverman, P. Jara, J. Browne, "Scaling into the cloud," Technical Report, Computer Science Department, University of California, Santa Barbara, 2009.
17. N. Roy, A. Dubey, A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," Proc. Of the 4th IEEE International Conference on Cloud Computing, 2011.
18. S. Islam, J. Keung, K. Lee, A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," Future Generation Computer Systems, vol. 28, no. 1, pp 155 – 165, 2012.
19. Nicholas I. Sapankevych, R. Sankar, "Time Series Prediction Using Support Vector Machines: A Survey", IEEE Computational Intelligence Magazine, vol. 4, no. 2, May 2009.
20. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, H. Witten, "The WEKA data mining software: an update," SIGKDD Explorations, vol. 11, no. 1, 2009.
21. Trevor, H., Tibshirani, R., and Friedman, J., The Elements of Statistical Learning: Data Mining, Inference, and Prediction, New York: Springer, February, 2009.
22. S. Arlot, A. Celisse, "A survey of cross-validation procedures for model selection", Journal of Statistics Surveys, vol. 4, pp 40–79, 2010
23. I. Witten, E. Frank, "Data mining practical machine learning tools and techniques with Java implementations," Academic Press, San Diego, 2000.
24. D. Garlan, B. Schmerl, "Model-based adaptation for self-healing systems", Proc. of the 1st Workshop on Self-Healing Systems, 2002.
25. R. Sterritt, B. Smyth, M. Bradley, "PACT: personal autonomic computing tools", Proc. of the 12th IEEE International Conference and Workshops on the Engineering of ComputerBased Systems, pp. 519–527, 2005
26. J.P. Bigus, D.A. Schlosnagle, J.R. Pilgrim, W.N. Mills III, Y. Diao, "ABLE: a toolkit for building multiagent autonomic systems". IBM Syst. Journal, vol. 41, no. 3, pp 350–371, 2002
27. M.L. Littman, N. Ravi, E. Fenson, R. Howard, "Reinforcement learning for autonomic network repair", Proc. of the 1st International Conference on Autonomic Computing, pp. 284– 285, 2004.
28. J. Dowling, E. Curran, R. Cunningham, V. Cahill, "Building autonomic systems using collaborative reinforcement learning", Knowledge Eng. Rev. no. 21, pp 231–238, 2006.
29. K. Hwang, X. Bai, Y. Shi, M. Li, W. Chen, Y. Wu, "Cloud Performance Modeling and Benchmark Evaluation of Elastic Scaling Strategies," IEEE Transactions on Parallel and Distributed Systems, vol. PP, no. 99, 2015

AUTHORS PROFILE

M. Jeyakarthic is Assistant Director of Tamil Virtual Academy, Chennai. He received M.C.A. and M.Phil from Madurai Kamaraj University. Ph.D. and M.B.A.(E-Business) from Annamalai University. During the years 2003 -2019, he worked as Assistant Professor in Dept. Of Computer and info. Science, Annamalai University. His current research involves in the fields of Business Analytics, Cloud computing, Wireless application protocols, and Tamil computing. Currently he is responsible for designing Tamil computing course activities in TVA.

N. Subalakshmi, is Assistance Professor of Annamalia University, Chidambaram. She received M. C. A from bharadhidasan University, Trichy. M. Phil from Annamalai university