Estimating Request Workloads on Cloud Platforms Using a Second Order Optimized Deep Neural Network Model

Anjali Solanki[1], Prof. Pankaj Raghuwanshi[2]

Abstract— **Cloud workloads are inherently dynamic, influenced by unpredictable user behavior, seasonal traffic variations, and sudden spikes in demand caused by events such as sales campaigns or system updates. Traditional statistical prediction models often struggle to capture these rapidly changing patterns. Without accurate workload forecasting, cloud service providers risk under-provisioning—leading to service delays and downtime—or over-provisioning, which increases operational costs. This unpredictability makes intelligent forecasting a necessity. Accurate workload prediction enables cloud platforms to allocate resources dynamically and proactively. This improves server utilization, reduces idle time, and ensures that computational power is available when needed, without excessive redundancy. For enterprises, this means lower operational costs, while for service providers, it results in better infrastructure. However, due to the absence of seasonality in the data patterns coupled with the sporadic nature of cloud workload, accurate prediction is a challenge. This paper presents a deep learning based approach with data optimization for predicting cloud workloads. It has been shown that the proposed approach clearly outperforms existing approaches in terms of prediction accuracy.**

Keywords—Cloud Workload Estimation, Deep Neural Network (DNN), Steepest Descent Approach, Mean Absolute Percentage Error (MAPE).

I. INTRODUCTION

Cloud Computing has become one of the most sought after technologies which plays a pivotal role in several domains resorting to the high levels of data complexity, complex computation or applications needing hybrid platforms [1]. One of the most important aspects of cloud systems management is the fact that cloud servers sporadically face sudden surges in the number of requests often termed as cloud workload [2] This workload, if unforeseen can result in crash of the cloud server if alternate provisions are not made to handle the cloud workload [3]. This in term needs the estimate of cloud workloads in advance considering several governing factors. This is majorly critical especially for applications such as e-commerce and finance which may see sudden surges in requests [4]. Thus there is a clear necessity of cloud workload prediction using models which can estimate cloud workloads with high accuracy. Statistical techniques are not found to be as accurate as the contemporary artificial intelligence and machine learning based approaches [5]-[6]. In this paper, a back propagation based approach for estimating cloud workload is proposed using deep learning architecture.

II. DEEP LEARNING

Deep learning has evolved as one of the most effective machine learning techniques which has the capability to handle extremely large and complex datasets [7]. It is training neural networks which have multiple hidden layers as compared to the single hidden layer neural network architectures [8].
The architectural view of a deep neural network is shown in figure 1. In this case, the outputs of each individual hidden layer is fed as the input to the subsequent hidden layer [9]. The weight adaptation however can follow the training rule decided for the neural architecture [10]. There are various configurations of hidden layers which can be the feed forward, recurrent or back propagation etc [11].

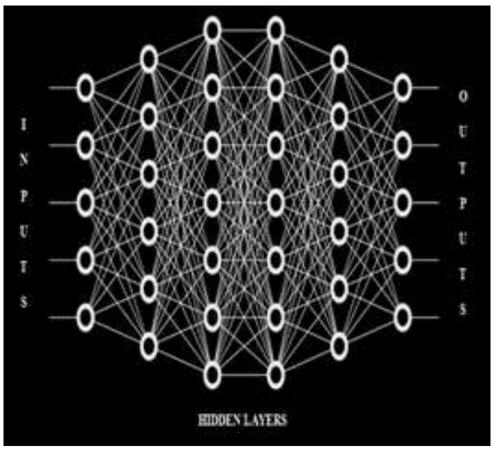


Fig.1 The Deep Neural Network Architecture

The figure above depicts the deep neural network architecture with multiple hidden layers [12]. The output of the neural network however follows the following ANN rule [13]:

$$Y = f(\sum_{i=1}^{n} X_i \,.\, W_i \;+\; \theta_i) \qquad \textbf{(1)}$$

Where,
X are the inputs
Y is the output
W are the weights
Ө is the bias.
f is the activation function.

Training of ANN is of major importance before it can be used to predict the outcome of the data inputs.

III. ERROR FEEDBACK MECHNAISM

Back propagation is one of the most effective ways to implement the deep neural networks with the following conditions [14]:

1) Time series behavior of the data
2) Multi-variate data sets
3) Highly uncorrelated nature of input vectors

The essence of the back propagation based approach is the fact that the errors of each iteration is fed as the input to the next iteration. [15]. The error feedback mechanism generally is well suited to time series problems in which the dependent variable is primarily a function of time along with associated variables. Mathematically [16],

$$Y = f(t, V_1 \ldots . V_n) \qquad \textbf{(2)}$$

Here,
Y is the dependent variable
f stands for a function of
t is the time metric
V are the associated variables
n is the number of variables

The back propagation based approach can be illustrated graphically in figure 2 [17].

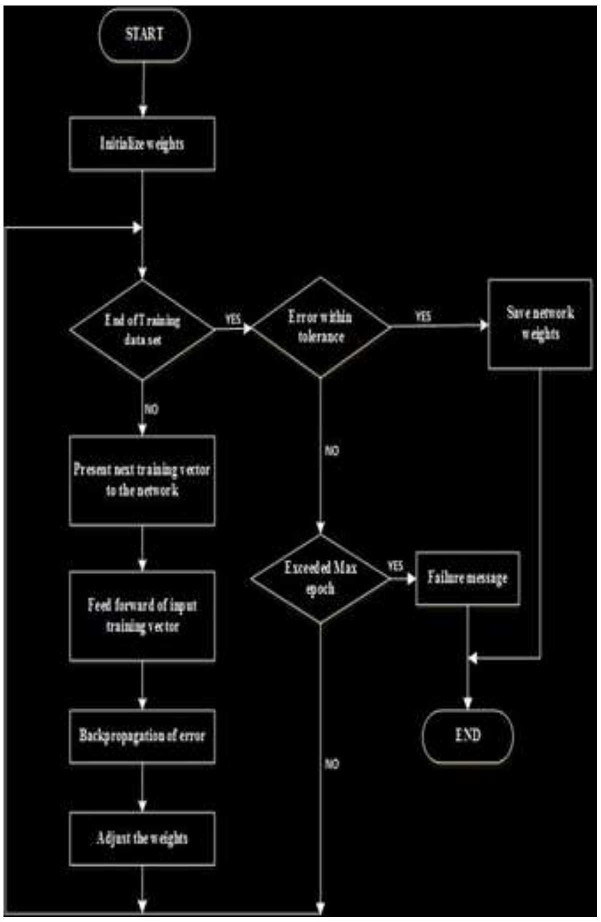


Fig.2 Error Feedback Mechanism

In case of back propagation, the weights of a subsequent iteration doesn't only depend on the conditions of that iteration but also on the weights and errors of the previous iteration mathematically given by [18]:

$$W_{k+1} = f(W_k, e_k, V) \qquad \textbf{(3)}$$

Here,
W_{k+1} are the weights of a subsequent iteration
W_k are the weights of the present iteration
e_k is the present iteration error
V is the set of associated variables
In general, back propagation is able to minimize errors faster than feed forward networks, however at the cost of computational complexity at times [19]. However, the trade off between the computational complexity and the performance can be clearly justified for large, complex and uncorrelated datasets for cloud data sets [20].

IV. METHODOLOGY

The gradient descent algorithms (GDAs) generally exhibit [21]:

1) Relatively lesser memory requirement
2) Relatively faster convergence rate

The essence of this approach is the updating of the gradient vector g, in such as way that it reduces the errors with respect to weights in the fastest manner [22]. Mathematically, let the gradient be represented by g and the descent search vector by p, then [23]:

$$p_0 = -g_0 \tag{4}$$

Where,
g_0 denotes the gradient given by $\frac{\partial e}{\partial w}$
The sub-script 0 represents the starting iteration
The negative sign indicates a reduction in the errors w.r.t. weights
The tradeoff between the speed and accuracy is clearly given by the following relations [24]:

$$W_{k+1} = W_k - \alpha g_x, \quad \alpha = \frac{1}{\mu} \tag{5}$$

Here,
w_{k+1} is the weight of the next iteration
w_k is the weight of the present iteration
g_x is the gradient vector
μ is the step size for weight adjustment in each iteration.

The indirect Hessian Matrix can be computed through the Jacobin as [25]:

$$\mathbf{H} = \mathbf{J_k}^{\mathbf{T}} \mathbf{J_k} \tag{6}$$

And

$$\mathbf{g} = \mathbf{J_k}^{\mathbf{T}} \mathbf{e} \tag{7}$$

Here,
H is the Hessian Matrix
J_k represents the Jacobian Matrix given by $\frac{\partial^2 e}{\partial {w}^2}$
J_k^T represents the transpose of the Jacobian Matrix.

The Quasi Newton or BFGS algorithm is thus computed as [26]:

$$w_{k+1} = w_k - \alpha[\frac{\partial^2 e}{\partial w^2}]^{-1} \frac{\partial H}{\partial w} \tag{8}$$

Here,
w_k & w_{k+1} denote the weights of the present and subsequent iterations.
α denotes the learning rate.
e denotes the error in the present iterations.

The speed of convergence enhance due to the indirect computation of the Hessian Matrix. The activation function used for the algorithm is the tan-sig function mathematically defined as [27]:

$$tansig(x) = \frac{2}{1+e^{-2x}} - 1 \tag{9}$$

The dimensional optimization is done through the Principal Component Analysis (PCA). The Principal Component Analysis is an optimization tool for the purpose of dimensional reduction of the data set. Consider a data set X having N samples. Out of the N sample, M samples may be highly correlated and hence may render low or little additional information to the training data [27].

$$M \underset{\in}{\rightarrow} N(\mathrm{X}) \tag{10}$$

Here,
M are the correlated samples
N are the total samples
X is the data set
If M samples are removed form the original data set, then there will be dimensional reduction in the data given by:

$$Y = X - M \tag{11}$$

Here,

Y is the dimensionally reduced data set for more effective training.

This input parameters used are
1) No. of servers
2) No. of users
3) Response time
4) Deviation delay value
5) Cloud Storage value
6) Mean Deviation value
7) Job Queueing value
8) Number of Operational Nodes
9) No. of Requests
The flowchart illustrates the summary of the system design.

The data is divided in the ration of 70:30 for training and testing data set bifurcation.
The final performance metrics computed for system evaluation are [28]-[30]:

1. **Iterations to Convergence.**
2. **Mean Absolute Percentage Error (MAPE)**

$$MAPE = \frac{100}{N} \sum_{i=1}^{N} \frac{p-a}{a} \tag{11}$$

3. Mean Squared Error:

$$MSE = \frac{\sum_{i=1}^{n} (p-a)_i^2}{n} \tag{12}$$

Here **p** and **a** stand for the predicted and actual values respectively.
The number of predicted samples is indicated by **n**.

4. Regression

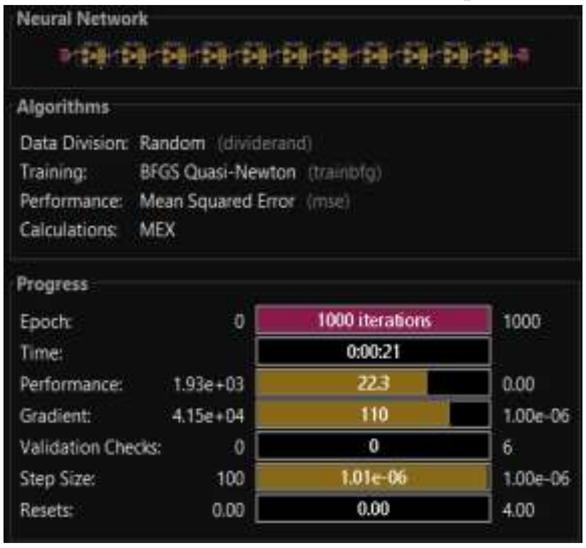
The essence of the proposed approach is to employ dimensional optimization using the PCA based approach and back propagation to develop and optimized training algorithm. The computation of the Hessian Matrix indirectly using the Jacobian reduced the computations complexity of the algorithm. The proposed algorithm can be presented as:

```
Algorithm:
Start
{
Step.1 Extract dataset and divide data into the ratio of 70:30 for training : testing.
Step.2 Apply dimensional optimization using PCA.
Step.3 Set initial learning rate $\mu = 0.1$ randomly, set maximum iterations as $Maxitr = 1000$, $e_{tolerance} = 10^{-6}$
Step.4: Feed forward training vector.
Step.5 Initialize weights randomly to initialize training.
Step.6: Updated weights as per the following training rule
        $w_{k+1} = w_k - \alpha[\frac{\partial^2 e}{\partial w^2}]^{-1} \frac{\partial H}{\partial w}$
Step.7 If (cost function stabilizes)
        Truncate training
        Else if (max. iterations are over)
        Truncate Training
        Else
        Feedback errors as inputs to subsequent iteration and iterate training rule as per step above.
Step. 8: On convergence, compute MSE, MAPE, $R^2$
}
Stop.
```

V. EXPERIMENTAL RESULTS

The experiment has been carried out on MATLAB 2025 version on a PC with 16GB of RAM, and i7 Intel Processor. The NASA cloud dataset has been used for the study. The parameters which have been used for the evaluation of the proposed work are:

1. MSE
2. MAPE
3. Regression
4. MSE w.r.t. the number of epochs



Neural Network

Algorithms

Data Division: Random (dividerand)
Training: BFGS Quasi-Newton (trainbfg)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	1000 iterations	1000
Time:		0:00:21	
Performance:	1.93e+03	22.3	0.00
Gradient:	4.15e+04	110	1.00e-06
Validation Checks:	0	0	6
Step Size:	100	1.01e-06	1.00e-06
Resets:	0.00	0.00	4.00

Fig.3 Designed Neural Network

The figure depicts the proposed model's training progress in terms of:
Iterations: 1000
Time elapsed: 21s
MSE: 22.3 at convergence.
Gradient: 110 at convergence
Learning Rate: $1.01 * 10^{-6}$ at convergence
Resets: 0

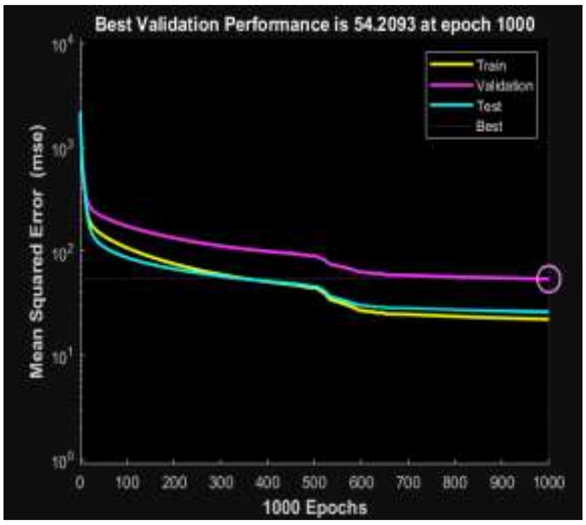


Fig.4 Training Convergence

Figure 4 depicts the training convergence in terms of the MSE of training, testing and validation. It can be observed that the model reaches convergence in 1000 iterations with final training MSE value at 22.3 and testing MSE of 54.2.

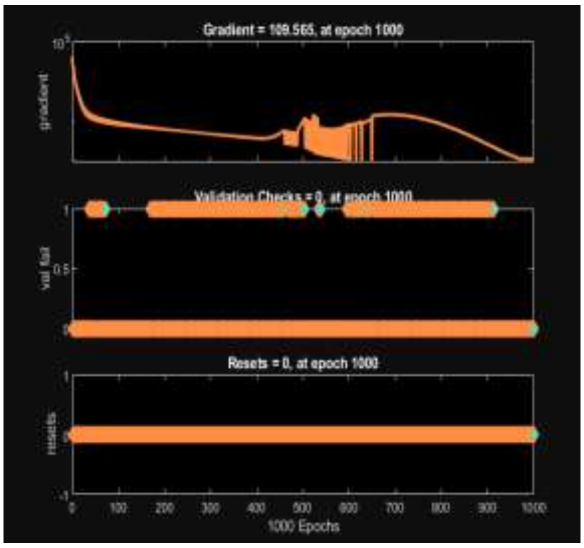


Fig.5 Training Parameters

The figure above depicts the different critical training parameters which happen be attain values of:
Gradient: 110 (approx.) at convergence.
Validation Fail: 0
Resets: 0
The regression values are presented next.

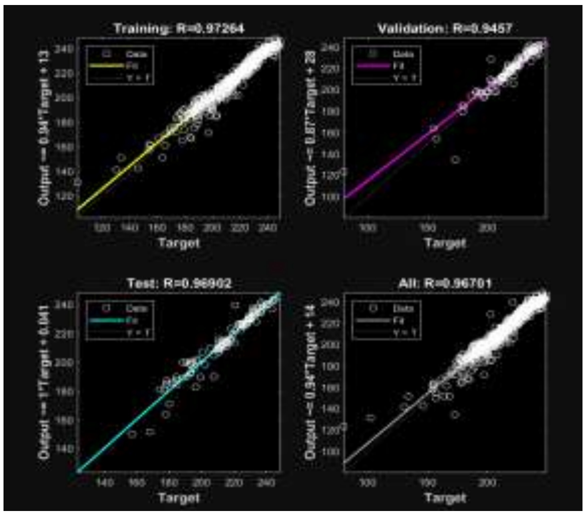


Fig.6 Regression Analysis

The figure above depicts the regression values for training, testing, validation and overall, with values:
Training: 0.972
Testing: 0.969
Validation: 0.945
Overall: 0.967

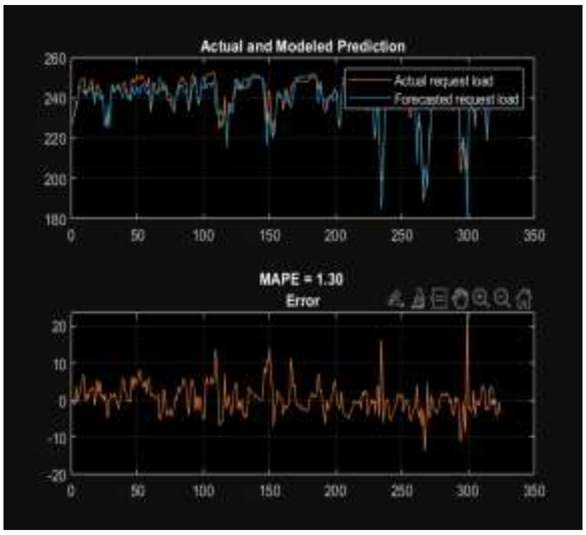


Fig.7 Prediction Error Analysis

The figure above depicts the %MAE for the proposed model which comes out to be 1.3%. The low value of the %MAE happens due to the dimensional optimization of data and optimized training approach. A summary of results and comparison with existing work in the domain is presented next:

Table 1. Summary of Results

S.No.	Parameter	Value
1	Data Split	70:30
2	ML Category	Deep Learning
3	Data Optimization	PCA
4	Iterations to Convergence	1000
5	MSE at Convergence	22.3
6	Gradient at Convergence	109.565
7	Validation Fails or Resets	0
8	R^2 (Training)	0.972
9	R^2 (Testing)	0.969
10	R^2 (Validation)	0.945
11	R^2 (Overall)	0.967
12	MAPE (Proposed Model)	1.3
13	Yuan et al. [28]	1.37
14	Yazdanian et al. [29]	5.9
15	Jeddi et al. [30]	6.4

A comparison with existing models in the domain of research has been presented next, in terms of %MAE, rendering higher prediction accuracy.
It can be observed that the proposed work clearly outperforms existing baseline approaches in terms of prediction accuracy with lower error rates.

CONCLUSION

Cloud computing has revolutionized the way organizations manage IT infrastructure by offering on-demand resources, scalability, and cost efficiency. However, cloud resources are finite and come with costs, making optimal allocation a crucial challenge. Workload prediction—the ability to anticipate future demand for computational, storage, and network resources—is essential for ensuring that applications meet performance requirements without over-provisioning. Deep learning, with its ability to model complex, non-linear patterns, offers a powerful approach to accurately forecast these workloads and improve cloud efficiency. As cloud computing continues to evolve with trends like edge computing, IoT integration, and AI-driven automation, the need for accurate workload forecasting will only grow. Deep learning's capacity for continual learning and adaptation positions it as a cornerstone technology for future cloud management systems. The proposed approach combines data optimization along with an optimized gradient descent approach predicting cloud workloads. It has been shown that the proposed work attains a %MAE of 1.3% with and overall R^2 value of 0.967, clearly outperforming existing work in the domain.

REFERENCES

[1] J. Gao, H. Wang and H. Shen, "Machine Learning Based Workload Prediction in Cloud Computing," 2020 29th International Conference on Computer Communications and Networks (ICCCN), 2020, pp. 1-9

[2] Z. Chen, J. Hu, G. Min, A. Y. Zomaya and T. El-Ghazawi, "Towards Accurate Prediction for High-Dimensional and Highly-Variable Cloud Workloads with Deep Learning," in IEEE Transactions on Parallel and Distributed Systems, 2020, vol. 31, no. 4, pp. 923-934.

[3] L. Wang and E. Gelenbe, "Adaptive Dispatching of Tasks in the Cloud," in IEEE Transactions on Cloud Computing, vol. 6, no. 1, pp. 33-45, 1 Jan.-March 2018

[4] S Afzal, G Kavitha, "Load balancing in cloud computing–A hierarchical taxonomical classification", Journal of Cloud Computing, Springer 2019, vol.8, no.22, pp.1-24.

[5] M. A. Shahid, N. Islam, M. M. Alam, M. M. Su'ud and S. Musa, "A Comprehensive Study of Load Balancing Approaches in the Cloud Computing Environment and a Novel Fault Tolerance Approach," in IEEE Access, 2020, vol. 8, pp. 130500-130526.

[6] M. Ala'anzy and M. Othman, "Load Balancing and Server Consolidation in Cloud Computing Environments: A Meta-Study," in IEEE Access, 2019, vol. 7, pp. 141868-141887.

[7] J. Bi, H. Yuan, M. Zhou and Q. Liu, "Time-Dependent Cloud Workload Forecasting via Multi-Task Learning," in IEEE Robotics and Automation Letters, 2019, vol. 4, no. 3, pp. 2401-2406,

[8] L Bao, J Yang, Z Zhang, W Liu, J Chen, "On accurate prediction of cloud workloads with adaptive pattern mining", Journal of Supercomputing, Springer 2023, vol.79, pp. 160–187.

[9] C Yang, Q Huang, Z Li, K Liu, F Hu, "Big Data and cloud computing: innovation opportunities and challenges", International Journal of Digital Earth, Taylor and Francis 2017, vol.10., no.1, pp.13-53.

[10] A. Rossi, A. Visentin, S. Prestwich and K. N. Brown, "Bayesian Uncertainty Modelling for Cloud Workload Prediction," 2022 IEEE 15th International Conference on Cloud Computing (CLOUD), Barcelona, Spain, 2022, pp. 19-29.

[11] Jitendra Kumar , Ashutosh Kumar Singh, "Workload prediction in cloud using artificial neural network and adaptive differential evolution", Future Generation Computer Systems, Elsevier 2018, vol.81, pp.41-52.

[12] D Saxena, AK Singh, "Auto-adaptive learning-based workload forecasting in dynamic cloud environment", International Journal of Computers and Applications, Taylor and Francis 2022, vol.44. no.6., pp.541-551.

[13] Z. Amekraz and M. Y. Hadi, "CANFIS: A Chaos Adaptive Neural Fuzzy Inference System for Workload Prediction in the Cloud," in IEEE Access, 2022, vol. 10, pp. 49808-49828.

[14] A. K. Singh, D. Saxena, J. Kumar and V. Gupta, "A Quantum Approach Towards the Adaptive Prediction of Cloud Workloads," in IEEE Transactions on Parallel and Distributed Systems, 2022, vol. 32, no. 12, pp. 2893-2905.

[15] S Sharifian, M Barati, "An ensemble multiscale wavelet-GARCH hybrid SVR algorithm for mobile cloud computing workload prediction", International Journal of Machine Learning and Cybernetics, Springer 2019, vol.10, pp. 3285–3300.

[16] D Alberg, M Last," Short-term load forecasting in smart meters with sliding window-based ARIMA algorithms", Vietnam Journal of Computer Science, Springer 2018, vol.5, pp. 241–249.

[17] B. Feng, Z. Ding and C. Jiang, "FAST: A Forecasting Model With Adaptive Sliding Window and Time Locality Integration for Dynamic Cloud Workloads," in IEEE Transactions on Services Computing, 2023, vol. 16, no. 2, pp. 1184-1197.

[18] T. Zhang and B. Yang, "Big Data Dimension Reduction Using PCA," 2016 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 2016, pp. 152-157.

[19] M. I. Abdulhussain and J. Q. Gan, "An experimental investigation on PCA based on cosine similarity and correlation for text feature dimensionality reduction," 2015 7th Computer Science and Electronic Engineering Conference (CEEC), Colchester, UK, 2015, pp. 1-4.

[20] D Li, X Wang, J Huang, "Diagonal BFGS updates and applications to the limited memory BFGS method", Computational Optimization and Applications, Springer 2022, vol.81, PP. pages829–856.

[21] IE Livieris, "An advanced active set L-BFGS algorithm for training weight-constrained neural networks", Neural Computing and Applications, Springer 2020, vol.32, pp. 6669–6684.

[22] M Huisman, A Plaat, JN van Rijn, "Stateless neural meta-learning using second-order gradients", Machine Learning, Springer 2022, vol.111, pp.3227–3244.

[23] R. Xin, S. Pu, A. Nedić and U. A. Khan, "A General Framework for Decentralized Optimization With First-Order Methods," in Proceedings of the IEEE, 2020, vol. 108, no. 11, pp. 1869-1889.

[24] M. Liu, L. Chen, X. Du, L. Jin and M. Shang, "Activated Gradients for Deep Neural Networks," in IEEE Transactions on Neural Networks and Learning Systems, 2023, vol. 34, no. 4, pp. 2156-2168.

[25] L. Li and J. Hu, "Fast-Converging and Low-Complexity Linear Massive MIMO Detection With L-BFGS Method," in IEEE Transactions on Vehicular Technology, 2022, vol. 71, no. 10, pp. 10656-10665.

[26] Y Wang, Z Wang, H Huang, "Stochastic adaptive CL-BFGS algorithms for fully complex-valued dendritic neuron model", Knowledge-Based Systems, Elsevier, 2023, vol.277, art.no. 110788.

[27] A Mokhtari, M Eisen, A Ribeiro, "IQN: An incremental quasi-Newton method with local superlinear convergence rate", SIAM Journal on Optimization, SIAM 2018, vol.28, no.2, pp.1-25.

[28] H. Yuan, J. Bi, S. Li, J. Zhang and M. Zhou, "An Improved LSTM-Based Prediction Approach for Resources and Workload in Large-Scale Data Centers," in IEEE Internet of Things Journal, vol. 11, no. 12, pp. 22816-22829, 15 June15, 2024

[29] P Yazdanian, S Sharifian, E2LG: a multiscale ensemble of LSTM/GAN deep learning architecture for multistep-ahead cloud workload prediction", Journal of Supercomputing, Springer 2022, vol. 77, pp.11052–11082.

[30] S Jeddi, S Sharifian, "A hybrid wavelet decomposer and GMDH-ELM ensemble model for Network function virtualization workload forecasting in cloud computing", Applied Soft Computing, Elsevier 2021, vol.88., Art.No. 105940.