

Received June 30, 2021, accepted July 6, 2021, date of publication July 9, 2021, date of current version July 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3096039

Autonomic Workload Performance Modeling for Large-Scale Databases and Data Warehouses Through Deep Belief Network With Data Augmentation Using Conditional Generative Adversarial Networks

NUSRAT SHAHEEN¹, BASIT RAZA¹, AHMAD RAZA SHAHID¹,
AND AHMAD KAMRAN MALIK¹

Department of Computer Science, COMSATS University at Islamabad (CUI), Islamabad 45550, Pakistan

Corresponding author: Basit Raza (basit.raza@comsats.edu.pk)

ABSTRACT Databases and warehouses are experiencing workload of different types such as Decision Support System (DSS), Online Transaction Processing (OLTP) and Mixed workloads. Handling variety of workload in autonomic systems is a critical task. After self-configuring the workload, the next challenge is workload performance tuning that motivates towards self-predictive systems. Existing studies provide performance modeling solutions on small-scale data repositories of Database Management System (DBMS) and Data Warehouse (DWH) using either classical eager or lazy learning approaches. However, in real-world problems, we normally have to deal with large-scale data repositories. Therefore, there is a need to develop performance models that provide data augmentation to solve large-scale data repositories that are not publicly available. In this study, deep learning approaches have been investigated for performance tuning of large-scale data repositories. We propose a performance prediction model called Optimized GAN-based Deep Learning (OGDL) model. For data augmentation, Conditional Generative Adversarial Networks (CGAN) is applied. For autonomic perspective, we incorporated MAPE-K model to manage the workload autonomically. Different deep learning models are applied, and it was observed that Deep Belief Network (DBN) performed better as compared to other deep learning models such as Deep Neural Network (DNN). We performed a number of experiments and from results it is observed that deep learning models performed the best in comparison with classical machine learning and lazy learning and a 6 – 8% increase in accuracy is recorded in our experiments using DBN. The proposed OGDL model performed the best in workload performance predictions in an optimized way.

INDEX TERMS Autonomic computing, generative adversarial networks, stacked genetic algorithm, deep belief network, large-scale data repositories.

I. INTRODUCTION

Existing studies have provided solutions for recognizing and configuring workload autonomically in autonomic databases. Large-scale data repositories such as databases and data warehouses are experiencing large volume of data with complexity and heterogeneity. Workload management of these repositories are becoming difficult for the database and warehouse

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott¹.

administrator, which ultimately, leads to system performance degradation [1]. Performance modeling can play an important role in performance tuning. Existing studies have proposed performance prediction models and frameworks [2], [3]. As humans have limited capacity to manage parameter tuning for the large volume and complicated tasks in large-scale data repositories. Therefore, developing a workload performance prediction model for autonomic Database management systems (DBMSs) and Data warehouses (DWHs) is a challenging task that must deal with larger data sets. One of

the challenges is that small data sets are available and training on small data results in poor test accuracy of prediction model for large-scale data. However, training on large data sets entail higher complexity and consume more compute cycles.

Many researchers are involved in searching the solutions for building autonomic performance prediction frameworks, models, techniques, and approaches while addressing limited data availability and using various machine learning techniques. Research is being carried out not only to present the framework but also to define the steps prior to performance prediction framework which involves understanding the nature of data. To understand the nature of data, researchers presented various contribution with respect to workload characterization and dynamic behavior of data [4]–[6]. The study regarding the behavior and type of data is very important because it provides the basis for any data-oriented autonomic systems [7], [8].

Workload entering the system is varying in nature with different characteristics. So, workload handling systems without any workload monitoring mechanism cannot predict about the workload requirements and workload performance that can result in lack of efficient resource utilization and lack of efficient implementation of performance improvement techniques. Research regarding the workload monitoring and performance prediction supports the idea of designing an autonomous system which can perform all tasks related to workload management with least amount of human intervention. These tasks include workload monitoring, type identification, performance prediction and resource planning. A system which can act as autonomous system must comprise of the Autonomic Computing technology [9] which is incorporation of various self-management characteristics such as self-inspection, self-prediction, and self-optimization.

Many tools, frameworks and algorithms have been developed by different researchers for autonomous workload management [10], [11]. Many research studies presented workload performance prediction framework to help the data handling systems in performance improvement [2]. By using these performance prediction outputs, a system can manage to achieve better resource allocation, scheduling, self-adaptation, and optimization. These performance prediction studies are using various performance parameters, preferably those which have high impact on system performance, according to the nature of data, these systems are dealing with. The selection of suitable performance metrics is still a considerable research area.

Existing studies provide performance modeling frameworks using either lazy or eager learning. These frameworks were built to work on small-scale data repositories. Data volumes are exponentially increasing with variable data behaviors, so solutions provided by these performance prediction frameworks cannot provide required accuracy when incorporated with large data repositories. Lack of availability of testing and training data to model large-scale data repositories is also an issue. There is need to increase the volume of testing

and training data for developing workload management models and frameworks.

Data augmentation is frequently employed to resolve the issue. Generative Adversarial Network (GAN) is being increasingly used for this purpose by the deep learning community. In this study, we are using Conditional Generative Adversarial Networks (CGAN) for data augmentation. With the increase in volume of testing and training data machine learning models such as Lazy learners or eager learners faces the performance degradation.

Deep learning techniques have potential to provide better solutions for performance tuning frameworks with large amount of data. In this study, we present the Deep belief network (DBN) based performance prediction model. Results show improvement in accuracy as compared to already existing models while working with large-scale data repositories.

The objectives of this study are as follows. To develop an optimized model that can manage the workload of large-scale data repositories having limited amount of training data. To handle large volume of data using deep learning approaches. To develop a model that incorporate autonomic characteristics towards self-managing systems. The contributions of the study are as follows:

- We proposed an autonomic workload performance tuning model for large-scale data repositories named as Optimized GAN-based Deep Learning (OGDL) model. We have investigated the existing studies [2], [3] that has proven to be good for small-scale data repositories using lazy learning approaches such as CBR. For large-scale data repositories, CBR performance degrades in terms of case retrieval and searching from data repositories such as case-base. Our proposed OGDL model predicts the performance of OLTP, DSS and Mixed types of workloads. It deals with large-scale data repositories using deep learning such DBN model and overcome the problems of lazy learning-based performance tuning models.
- Since for large-scale data repositories we have to deal with huge data however, limited data is available publicly. Further, deep learning models requires more data for deeper learning. Therefore, we artificially generated data using CGAN model that has been used in many studies in different domains. Therefore, we developed a CGAN-based model for data augmentation to overcome limited data availability without compromising the data accuracy.
- In comparison with lazy learning, we have applied three different deep learning models that are Deep Neural Network (DNN), DBN and CGAN. For network parameter optimization Genetic Algorithm (GA) is used and for further improvement in optimization Stacked Genetic Algorithm (SGA) is also applied. Results show that our proposed OGDL model (GAN+DBN+SGA), based on deep learning is proved to be the best solution for large-scale data repositories of database and data warehouse and a 6 – 8% increase is observed as

compared to existing studies that are using lazy learning approach.

- We provided an Artificial Intelligence (AI) based system by enabling autonomic characteristics. To make our system autonomic, the proposed OGDG is based on basic architecture of autonomic computing consisting of Monitor, Analyze, Plan and Execute (MAPE-K) model and a feedback loop.

Organization of the study is as follows. Section II presents related work. Section III provides the proposed methodology for this study. Section IV provides proposed Optimized GAN-based deep learning (OGDL) model. Section V describes experiments performed for proposed study and discusses its results. Section VI concludes the study with its limitations and provides the future direction.

II. RELATED WORK

The basic focus now a days regarding management of the database workload is on achieving the autonomic solution. The aim of an autonomic solution is to build an intelligent system which can manage the workload without much human intervention. The intelligent systems with incorporation of Autonomic Computing (AC) technology should be capable of self-management. To show that how existing databases systems are autonomic, various studies are presented [12]–[14]. Few studies also investigate autonomic aspects in different DBMS like DB2 [15], and Oracle [16]. These studies highlighted different processes which are controlled and operated in an autonomic manner.

AC technologies are good at providing solutions for performance prediction and workload behavior monitoring. A CBR based approach is presented in [17] which predict the change in workload behavior. Various studies presented work on providing predictions related to workload performance such as query arrival times [17], [18]. However, more performance parameters can be used to reflect the workload behavior during execution for better workload management and resource utilization. In [20], authors presented the Fuzzy Inference System (FIS) based model to predict three performance parameters: buffer hit ratio, database size, and the number of users. Only one performance parameter is predicted, which is not able to completely model the performance of workload. The study [21] presented the modular approach for estimation of SQL query execution time. SQL query execution plan was predicted in [22], [23] and they proposed prediction of accurate estimation of disk visits and CPU time for large data size. For performance prediction of CPU, different studies proposed performance models such as query execution prediction model [24]. To monitor, control, analyze and predict the configuration of database configuration parameters, the framework namely MAG is presented in [25]. This proposed approach mainly focuses on the root causes of database performance problems. They are using ANN for prediction. A framework for performance and resource analysis is presented in [26] that predicted throughput, resource

consumption and bottleneck. An approach for self-tuning database system performance based on Fuzzy logic is presented in [27]. In this study, Fuzzy rules are designed by assuming the values of parameters. A plan-structured neural network is crafted for predicting the latency of query execution plans in relational DBMSs [28]. For a single query, it builds tree structured network to predict query latency. Tracking every query executed in the DBMS increases the computational cost of model construction. An AWPP framework is presented in [2] that predicts the performance of database workload using different performance parameters. It used lazy learning CBR approach for performance prediction. This study doesn't specify any mechanism to revise the missing solution. The system performance can decrease with the increase in the size of case-base. A cluster-based performance prediction framework for data warehouse is presented in [3]. This study presented the CBR-based approach and achieved the improvement in retrieval efficiency using clustering. However, solution finding in revised phase of CBR is not optimized. Our work is different from other existing frameworks because it uses deep learning approach, DBN, for database performance prediction. Our proposed model is based on large-scale data repositories, so we required large amount of training data. The studies that are dealing with large data repositories are facing the lack of sufficient amount of data available for model training. To solve this issue different studies are focusing on deep learning based data augmentation techniques like GAN [29].

The study [30] presents performance modeling of big data and to address the issue of non-availability of sufficient training data, they augment the data using GANs. To find the optimal solution in the search space they used Genetic Algorithm (GA). A GAN based framework to solve class imbalance problem in building the classifier for credit card fraud detection is presented in study [31]. For the credit card data fraudulent class has a smaller number of samples which creates biasness in data. So, to increase the data of fraudulent class they are using GAN. An algorithm is presented in study [32] for the prediction of students' performance under supportive learning via school and home tutoring. As academic data set is low in sample size, so to increase the data it uses a variant of GAN as data augmentation technique. This study claims that after implementation of proposed technique they get 8 – 29% increase in terms of different performance indicators. Deep learning-based techniques are more popular in image processing. A GAN based study is presented in [33] for the fusion of multi-resolution images.

After increasing the data set, machine learning approaches can meet with performance issues in handling such data. So deep learning solutions can be more suitable for large data sets. Various studies are using different deep learning solutions for the improvement of results in different fields. The studies [34], [35] presented the botnet detection systems using deep learning. Evaluation of effectiveness of shallow and deep networks is presented in [36]. Deep learning techniques are also helpful in attack detection in Networks [37].

A study for time series prediction is presented in [38]. This study presented DBN based methods which is using 3 stacks of RBM to overcome the limitations of MLP and ANNs. Another study [39] presented Optimally Configured and Improved Deep Belief Network (OCI-DBN) approach for heart disease prediction. After implementation and comparison with different feature selection techniques, this study selected Ruzzo-Tompa technique. This approach gets 94.6% accuracy in predicting heart diseases. A fault diagnosis model for gearbox fault diagnosis is proposed in study [40]. This model integrates grasshopper optimization algorithm (GOA) with DBN. GOA is for network parameter optimization for DBN. This study claims to achieve 99.5% diagnosis date. We are presenting OGD model which is based on DBN and for deep network parameter optimization we are using SGA. For data augmentation we are using CGAN.

III. PROPOSED METHODOLOGY

In this section, we describe the proposed methodology to conduct this study. We discussed the details of data set that is being used in our experiments, the selection of important workload performance related features through well-known features selection techniques, and evaluation metrics that are used for experiments. To overcome the limitations of available data set, data augmentation using CGANs is described.

A. DATA SET

In this study, we have used benchmark data sets that have been used by other researchers in their studies and is available at <http://tpc.org/>. We are dealing with two main types of data which are OLTP and DSS. For these two types of data, in many studies, same data sets were used such as TPC-E and TPC-H for OLTP and DSS respectively. We have used the standard benchmark workloads/queries available online [41] that includes standard 22 queries for TPC-E and TPC-H each. For the data set used in our experiments, we have also extended data by designing TPC-E and TPC-H like queries for database and data warehouse workloads. Regarding training and testing of machine learning based models, there is need of sufficient amount of data for good learning. Moreover, sufficient amount of data is also required to avoid the issues related to overfitting of machine learning models. Therefore, we have designed TPC-E and TPC-H like queries to extend our data set for better training of our models.

We executed all the queries and obtained status variable values that are 500 in total. A careful observation of the relationship between workload and performance parameters is done. If any performance parameter reflects considerable change during workload execution, then this parameter can be the candidate of workload performance metric vector. And all those parameters which are not affected or less affected by workload execution, are eliminated from the list of influenced parameters. We obtained 25 workload performance related parameters as shown in Table 1.

TABLE 1. Workload performance related candidate parameters.

| S# | Code | Feature Name |
|----|------|----------------------------------|
| 1 | F0 | BYTES_RECEIVED |
| 2 | F1 | BYTES_SENT |
| 3 | F2 | COM_SHOW_KEYS |
| 4 | F3 | CREATED_TMP_DISK_TABLES |
| 5 | F4 | CREATED_TMP_TABLES |
| 6 | F5 | HANDLER_COMMIT |
| 7 | F6 | HANDLER_READ_KEY |
| 8 | F7 | HANDLER_READ_RND |
| 9 | F8 | HANDLER_TMP_WRITE |
| 10 | F9 | INNODB_BUFFER_POOL_BYTES_DATA |
| 11 | F10 | INNODB_BUFFER_POOL_READ_REQUESTS |
| 12 | F11 | INNODB_BUFFER_POOL_READS |
| 13 | F12 | INNODB_DATA_READ |
| 14 | F13 | INNODB_DATA_READS |
| 15 | F14 | INNODB_DATA_WRITES |
| 16 | F15 | INNODB_DATA_WRITTEN |
| 17 | F16 | INNODB_PAGES_READ |
| 18 | F17 | SORT_ROWS |
| 19 | F18 | SELECT_FULL_JOIN |
| 20 | F19 | ROWS_READ |
| 21 | F20 | ROWS_SENT |
| 22 | F21 | ROWS_TMP_READ |
| 23 | F22 | INNODB_DBLWR_PAGES_WRITTEN |
| 24 | F23 | INNODB_DBLWR_WRITES |
| 25 | F24 | INNODB_LOG_WRITE_REQUESTS |

TABLE 2. Selected performance metric vector (PMV).

| S# | Code | Feature Name |
|----|------|----------------------------------|
| 1 | F2 | COM_SHOW_KEYS |
| 2 | F4 | CREATED_TMP_TABLES |
| 3 | F5 | HANDLER_COMMIT |
| 4 | F6 | HANDLER_READ_KEY |
| 5 | F7 | HANDLER_READ_RND |
| 6 | F10 | INNODB_BUFFER_POOL_READ_REQUESTS |
| 7 | F11 | INNODB_BUFFER_POOL_READS |
| 8 | F13 | INNODB_DATA_READS |
| 9 | F17 | SORT_ROWS |
| 10 | F18 | SELECT_FULL_JOIN |
| 11 | F21 | ROWS_TMP_READ |

B. FEATURES SELECTION

For features selection, out of 25 performance parameters, we applied features selection techniques to find the best contributing features. For performance features selection, we applied six feature selection methods. After applying different feature selection methods, we get different features selected by each method. Features selected by Random Forest method shows higher prediction accuracy. So, we selected all performance features which we get by applying Random Forest. These selected parameters are 11 and represented in Table 2. These performance parameters can be represented as Performance Metric Vector (PMV). For workload input feature extraction, we are using Workload Feature Vector (WFV). The WFV consists of 7 parameters which are presented in Table 3.

C. EVALUATION METRICS

The formulae for precision, f-measure, recall, and accuracy are shown in the following equations 1-4.

$$\text{Precision} = \frac{T_p}{T_p + F_p} \quad (1)$$

TABLE 3. Workload feature vector (WVF).

| S# | Feature Name |
|----|-----------------------------------|
| 1 | Number of sub-queries |
| 2 | Number of equality predicates |
| 3 | Number of selection predicates |
| 4 | Number of non-equality predicates |
| 5 | Number of joins |
| 6 | Number of aggregation columns |
| 7 | Number of sort columns |

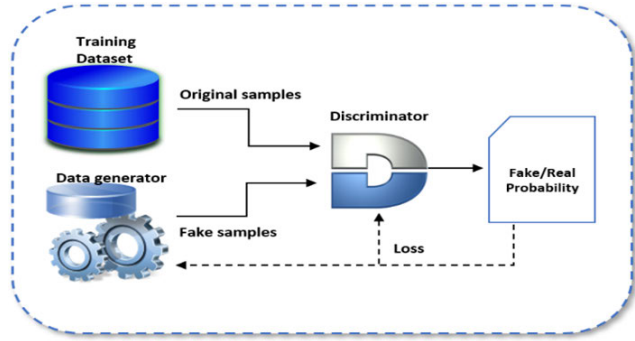


FIGURE 1. Data augmentation using CGAN.

$$\text{Recall} = \frac{T_P}{T_P + F_N} \quad (2)$$

$$F - \text{Measure} = \frac{2 * \text{precision} * \text{Recall}}{\text{precision} + \text{Recall}} \quad (3)$$

$$\text{Accuracy} = \frac{T_P + T_N}{T_P + T_N + F_N + T_N} \quad (4)$$

where T_P are the number of true positives, F_P are the number of false positives, F_N are the number of false negatives, and T_N are the number of true negatives.

D. CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS (CGAN) ARCHITECTURE

Due to availability of limited data and to provide a solution that performs reasonable well on large volumes of data, We proposed the performance model using the Conditional Generative Adversarial Networks (CGAN) [29], [30]. CGAN can work well with limited data without sacrificing accuracy. There are two main components of CGAN: namely, Generator denoted by Gen and the Discriminator denoted by Dis with class label condition on it. The working of CGAN is shown in Figure 1.

The aim of CGAN is to generate fake data which is as close to the real data as possible. Data generation is performed by the generator and the discriminator identifies if it is closer to real data or not.

E. TRAINING PROCESS USING CGAN

First, we prepared training data set. We trained Dis to increase the probability of s with real cases that maximize the $\text{Dis}(s)$, so that it can identify fake cases frequently that are generated by the Gen. The purpose of Gen is to deceive Dis by generating fake cases just like real cases. When Dis identifies a fake

Algorithm 1 Data Augmentation Using Conditional Generative Adversarial Networks

```

1: Input: Data set, Labels, Classes.
2: Output: New generated features of all classes.
3: Procedure = CGANAUGMENTATION(Data set, Labels, Classes)
4: Labels=OneHotEncodede(Labels)
5: Dataset=MinMaxNorm(Dataset)
6: NumFeatures=FindFeatures(Dataset)
7: NumSamples=FindSamples(Dataset)
8: for Each class in Classes do
9:   for Number of training iterations do
10:    Update the Discriminator using Dataset(Class) and Conditional Values Set C
11:    Generates samples set S using Vector Z Conditional Values Set C
12:    Update the Discriminator by ascending its gradient
13:    Update the Generator by descending its gradient
14:   end for
15: end for
16: for Each class in Classes do
17:   GeneratedFeatures = GenertaeFeatures (Generator, class, nsamples)
18: end for
19: return GeneratedFeatures
    
```

case, a loss is also generated. We also trained Gen to minimize loss $\log(1 - \text{Dis}(\text{Gen}(t)))$ synchronously. As the concept is taken from game theory where we reach at Nash equilibrium point the convergence point [42]. Therefore, training of Gen and Dis continues till convergence point. The gradient optimization of Gen is based on information provided from Dis. The process of training in CGAN is two-fold process so the gradient optimization of Gen performed better in comparison with neural network models. The learning based on gradient becomes stronger. The algorithm of data augmentation using CGAN is shown in Algorithm 1. As discussed above it takes data set, classes and labels as input and outputs the generated features. The labels are converted to one hot representation and data set is normalized using MinMax normalization. After pre-processing the CGAN is trained on the data and after training the features of each class are generated separately from generator. We have generated 400 features of each class.

F. DEEP BELIEF NETWORK (DBN)

DBN is a generative graphical model which is based on Restricted Boltzmann Machine (RBM) and consists of ANN. RBM is the unsupervised network and its structure is composed of hidden layers and visible layers. The connectivity between layers is strong but the connectivity between neurons on the same layer is weak, which means that there are connections between visible and hidden layers. RBM is the generative energy-based model with good connections between input and hidden layers but not between the nodes within

the layers. The standard RBM consists of binary-values in hidden and visible layers. Pre-training in DBN is done using greedy learning algorithms. In greedy learning algorithms, learning is done using layer-by-layer approach for generative weights. These weights determine about the dependencies of variables of one layer on the variables of another layer. As DBN is multi-layered structure having different layers of RBM machines. This divides the network in different layers and each layer is trained separately hence divides the training process. It is easier to train a shallow network as compared to deep complex network.

IV. PROPOSED OPTIMIZED GAN-BASED DEEP LEARNING (OGDL) MODEL

We proposed an optimized GAN-based deep learning model for workload performance tuning. The flowchart of methodology is shown in Figure 2 in three phases. In phase 1, the data set is pre-processed using the normalization technique. After that, we performed data augmentation using GAN to generate the number of instances which would be helpful in improvement of classifier performance. Next, we extracted the workload feature vector and selected important features using feature selection techniques. After selection of important features, the data is split into train, test, and validation set. In phase 2, we use SGA and GA for parameters selection based on natural selection process. After selection of best hyperparameters that would give better performance, different deep learning models such as deep belief network is trained on these parameters in phase 3. After training of the model, it is evaluated on the test data set. The process is executed and stopped on completion of all the folds. After completion of all training process, the performance evaluation is performed on the validation data.

A. MATHEMATICAL FORMULATION OF PROPOSED OGDL MODEL

In the data preparation phase, we used CGAN for data augmentation for which target optimization function can be derived [30]–[33] as follows. CGAN is a type of GAN in which generator and discriminator are conditioned by adding extra information. In our case, we have added extra information on class labels for generating targeted data. Loss function for CGAN can be derived from binary cross-entropy loss function formula, which can be written as shown in equation 5.

$$\text{Loss}(\hat{r}, r) = [r \cdot \log \hat{r} + (1-r) \cdot \log (1 - \hat{r})] \quad (5)$$

where r = real data and \hat{r} = generated data

At discriminator during training label of data which is coming from real data distribution $P_{\text{rdata}}(s)$ is $r = 1$ and for generated data $\hat{r} = \text{Dis}(s, c)$, where s is data and c is the class label condition. Loss for real data at discriminator is shown in equation 6.

$$\text{Loss}(\text{Dis}(s), 1) = \log(\text{Dis}(s, c)) \quad (6)$$

For the generated data $P_{\text{gendata}}(s)$ coming from generator, data label is $r = 0$ and $\hat{r} = \text{Dis}(\text{Gen}(t, c))$ where t is random noise and c the class label condition. CGAN Loss for generated data is shown in equation 7.

$$\text{CGANLoss}(\text{Dis}(t), 0) = \log (1 - \text{Dis}(\text{Gen}(t, c))) \quad (7)$$

Total loss at discriminator is equal to both loses. Both equations (6) and (7) can be combined to calculate the total CGAN Loss of discriminator is shown in equation 8.

$$\text{CGANLOSS}_{\text{Dis}} = \log \text{Dis}(s, c) + \log (1 - \text{Dis}(\text{Gen}(t, c))) \quad (8)$$

The main objective of the discriminator is to correctly classify real and fake data. For this objective equation 8 should be maximized and for discriminator loss final function can be shown in equation 9.

$$\text{CGANLOSS}_{\text{Dis}} = \max [\log \text{Dis}(s, c) + \log (1 - \text{Dis}(\text{Gen}(t, c)))] \quad (9)$$

As the generator is working in competition with discriminator to generate the data near to real data. So, objective of generator to minimize the generator loss. So, at generators' end the equation 9 will be minimize as shown in equation 10.

$$\text{CGANLOSS}_{\text{Gen}} = \min [\log \text{Dis}(s, c) + \log (1 - \text{Dis}(\text{Gen}(t, c)))] \quad (10)$$

According to Goodfellow *et al.* [29] the generator (Gen) and discriminator (Dis) both players play a min-max game with value function $V(\text{Dis}, \text{Gen})$. So, we can combine equation 9 and 10 to get complete GAN function for discriminator and generator as shown in equation 11.

$$\text{CGANLOSS} = \text{Gen}^{\min} \text{Dis}^{\max} [\log \text{Dis}(s, c) + \log (1 - \text{Dis}(\text{Gen}(t, c)))] \quad (11)$$

The loss function from equation 11 is valid for single data point. The target optimization function for complete data set can be written as shown in equation 12.

$$\begin{aligned} \text{Gen}^{\min} \text{Dis}^{\max} V(\text{Dis}, \text{Gen}) \\ = \text{Gen}^{\min} \text{Dis}^{\max} (E^{s \sim P_{\text{rdata}}(s)} [\log \text{Dis}(s, c)] + E_{t \sim p_t(t)} \\ \times [\log(1 - \text{Dis}(\text{Gen}(t, c)))] \end{aligned} \quad (12)$$

In equation 12 $\text{Dis}(s, c)$ shows the probability of s that are derived from real cases of workload performance metric vector (PMV) instead of random generation from Generator Gen, t represents the random noise the random noise and $\text{Dis}(\text{Gen}(t, c))$ is the discriminator's output for generated sample. Where s represents training sample and c is class label condition.

Deep learning requires large amount of data to learn complex features mapping and understand it better. In our study, we have used conditional GAN (CGAN) where the model have to generate data from each label conditionally. The Gen learns to map latent space points to new features. In our study

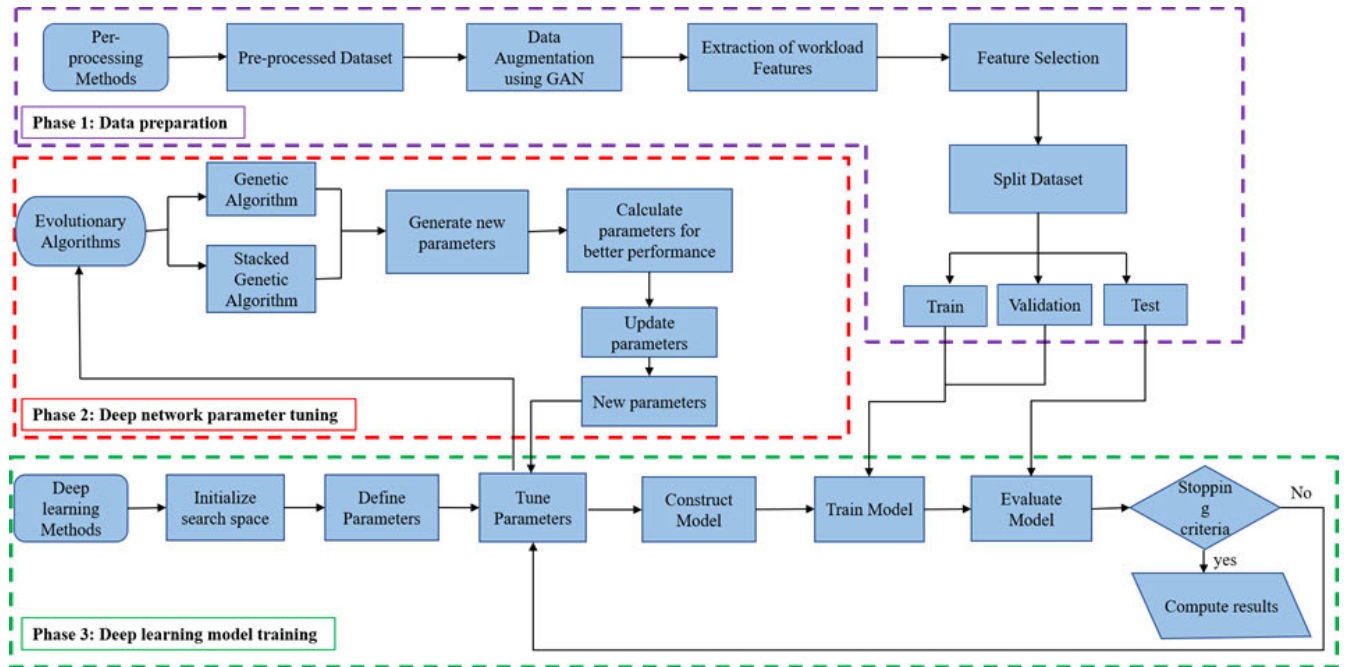


FIGURE 2. Flowchart of methodology of the proposed OGD model.

we have used 200 dimensional latent space vector of random numbers from a uniform distribution. Let the latent space be represented as t so, $t \sim U[0, 1]$. To make learning better, the data is normalized using MinMax normalization.

Our Dis architecture of CGAN model consists of input layers with features and labels as input. To encode the class labels, we have used the embedding layers, which transform the labels into dense vector representation. The embedding layer consists of two parameters, label index and the size of output. In our case, the size of output is the number of features. The Adam optimizer is used with $2e^{-4}$ learning rate and beta value of 0.5. A dropout between 0.4 and 0.7 is used between the layers to prevent overfitting and LeakyRelu with alpha value of 0.2. The idea behind Gen is to generate realistic data. The training process and feedback by Dis enables Gen to generate more realistic data. Similar to Dis, Gen also takes labels as input with noise generated from latent space t . Then the input layer is followed by embedding layer. The purpose of embedding layer is same as discussed above. Batch Normalization layers are used within different layers to make learning stabilize. CGAN is based on the minmax concept also means zero-sum game which means that if one wins other loses. In CGAN, we try to decrease the loss of Gen and increase the loss of Dis which means that the Dis classify generated data as real and Gen generates realistic data.

For network parameter optimization, we applied SGA with two stacks. Chromosome is the set of parameters which defines a proposed solution to the problem that we are interested to solve using GA. In this study the chromosome is the collection of network parameters, for those we need to find

optimized setting and can be shown in equation 13.

$$\text{ch}(F_i) = (f_1, f_2, f_3, \dots, f_n) \quad (13)$$

where $i = 1, 2, 3, \dots, m$

Here n is the number of network parameters and m is the population size. Population is randomly generated which assigns values to the performance parameters. Each individual of population will be considered as a candidate solution. Only those individuals will be considered as solution that qualify a certain criterion. So, every individual will undergo a testing process to find the best optimized solution. For mutation we setup 0.2 mutation rate. The fitness function is shown in equation 14.

$$\begin{aligned} \text{Fitness}(\text{fit}_{\text{fun}}) &= \text{network accuracy} \\ &\geq \text{Maximum}(\text{network accuracy}) \end{aligned} \quad (14)$$

For workload performance prediction, we are using DBN which consist of stacks of RBMs. Energy function for RBM is shown in equation 15.

$$E(v, h) = \sum_{i=1}^n a^i v^i - \sum_{j=1}^m b^j h^j - \sum_{i=1}^n \sum_{j=1}^m v^i w^{ij} h^j \quad (15)$$

In equation 15, the state vector for visible layer is denoted by $v = (v_1, v_2, v_3, \dots, v_n)$ and state vector for hidden layers is denoted by $h = (h_1, h_2, h_3, \dots, h_m)$. The bias weights (offsets) for visible layers are denoted by $a = (a_1, a_2, a_3, \dots, a_n)$ and bias weights for hidden layers are denoted by $b = (b_1, b_2, b_3, \dots, b_m)$. The w^{ij} represents the weights between visible layer i and hidden layer j from weight matrix. $\sum_{i=1}^n a^i v^i$ represents energy in the visible layer i ,

$\sum_{j=1}^m b^j h^j$ represents the energy in the hidden layer j , and $\sum_{i=1}^m v^i w^{ij} h^j$ represents the energy between the visible and hidden layers. We are using 200 hidden layers and initial weights are setup by unsupervised training. RBM defines a joint probability distribution function over the hidden layers and visible layers which is as shown in equation 16.

$$p(v, h/\theta) = \frac{\exp - E((v, h/\theta))}{l(\theta)} \quad (16)$$

where $\theta = (w^{ij}, a^i, b^j)$ and $l(\theta)$ is partition function.

B. AUTONOMIC PERSPECTIVE

We used MAPE-K model with feedback loop of autonomic architecture for incorporating autonomic perspective. Figure 3 presents the overall architecture of proposed feedback control loop.

C. WORKLOAD KNOWLEDGE-BASE

The knowledge base consists of data, which can be shared among the four phases of MAPE-K.

D. SENSORS AND EFFECTORS

The Sensors and Effectors are the essential supporting components of any autonomic system. Sensors collect the information which the system receives continuously. Sensors after collecting the information, provide it to different components of the system for the necessary actions. In our study, Sensors are Workload Feature Vector (WFV) and are receiving the incoming workload as the information. This information circulates through the system. After receiving the workload, Sensors forward this information to the monitor phase for initial monitoring of the workload. Effectors are the components which reflect the system's final state after completely processing the information. Effectors can also be referred as the set of operations that are employed by autonomic managers. In our study, Effectors are represented by Performance Metric Vector (PMV) which contain the information regarding the performance prediction of the incoming workload by the autonomic manager.

E. WORKLOAD MONITOR

The Monitor phase is the first phase in autonomic manager. The information captured by the Sensor enters into the monitor phase. This phase monitors the incoming information and starts with preprocessing of information. In our system, the monitor phase takes the workload and extracts the input feature vector which is sent to the next phase.

F. WORKLOAD ANALYZE

The analyze phase provides the ways to observe and analyze the information to determine if any change is required. Workload Analyzer analyses the workload by analyzing the workload performance parameters.

G. WORKLOAD PLAN

Plan phase works on to find the ways to respond to the change detected by the analyze phase. In our study, plan phase creates a procedure to handle and address the detected change. This phase implements the procedures which contains the optimization algorithm SGA and deep learning algorithm DBN for performance metric prediction.

H. WORKLOAD EXECUTE

The execute function finally executes to provide the ways to perform the necessary changes. This phase executes the plan generated by the plan phase. In this study, execute phase predicts the performance of the workload according to the shift in workload.

I. FEEDBACK CONTROL LOOP

The implementation of Feedback control loop is to monitor the system continuously and to report the changes detected in the workload so that it can be responded accordingly. Autonomic control loop consists of two major components; a controller which implements all phases of autonomic system and a knowledge base as a managed resource. The controller captures the workload information continuously using Sensors. This information goes to the monitoring step where extraction of relevant features is carried out on the basis of query features. Similar workload patterns are considered stable, however, for different workload patterns we can observe the change which can be called as workload type change. The extracted feature vector then enters the analyze phase. In this phase workload change detection is computed. If significant change is detected then the control loop enters the plan phase. Here with the help of optimization search and deep belief network the performance prediction can be planned according to the workload type. Finally, the execute phase predicts the performance of workload. Starting from monitoring phase to the execute phase there is a continuous looping state which remains active and listens and responds to workload shift.

J. DEEP LEARNING USING DEEP BELIEF NETWORK

For making the prediction, the training model's goal is to generate the hypothesis, which means model learns the fitting function after analyzing the training data. In DBN, the workload performance prediction model is built in two stages. The first stage involves in the upward training and second stage is downward adjustments. RBM is based on unsupervised greedy layer-by-layer training. Training is carried out in first stage, which is greedy in nature, and initial weights are set by continuous unsupervised training. During training no backpropagation is performed to analyze errors. The backpropagation algorithm is set up after calculating the initial weights by training stage. After that backpropagation is performed for parameter tuning and performance optimization. The next task is to address the optimization problem for network by computing the number of layers, number of nodes and number of hyperparameters needed for best performance.

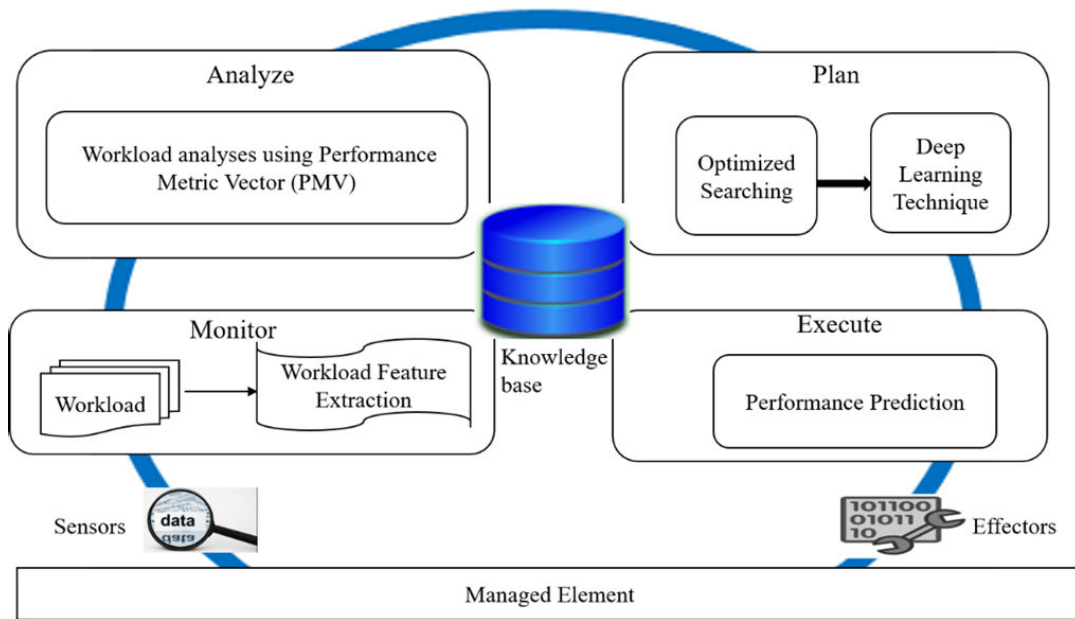


FIGURE 3. Proposed feedback loop for OGD.

Selection of number of layers and nodes affects the performance of a network. However, finding this optimal number is a major problem as it depends on the nature and characteristics of data set. To solve this problem, we used Stack Genetic Algorithm (SGA) which consists of two stacked GAs. Figure 4 shows the proposed approach for optimization of DBN using SGA. After optimization of network parameters, these parameters will be provided to the DBN model. For DBN configuration, a two-step method is followed: one is upward training and second is downward adjustments by utilizing the generative weights for fine tuning process. DBN is constructed by stacking of more than one RBMs. Number of stacks depend on the system requirement. A DBN can be trained in a greedy unsupervised way by training each RBM separately from it, in a bottom up manner. Hidden layers of previous RBM will be used as input layer for the next RBM. Initially pre-training is required for getting better optimization which starts by initializing weights of all layers. In the first step, pretraining initial weights are setup by performing unsupervised training using greedy learning approach. Initial weights determine the dependencies of variables from one layer to the other layer.

First RBM layer which is visible layer comes in state v_1 after receiving data for training. The initial weight which can be represented as w_1 is responsible for generating the hidden layers state h_1 . Initial weight w_1 is also responsible for reconstructing the initial v_1 state by hidden layer state. After completing the training of first RBM, RBM2 and RBM3 is also trained in same way. After initial weights are determined, backpropagation algorithm is used for fine tuning of parameters and performance optimization. This fine tuning helps us for the weight adjustments regularly with slight feature

modifications. DBN training is completed after two-step method.

K. GA AND STACK GA (SGA) FOR OPTIMIZATION

For better performance prediction, selection of optimized setting for DBN is very important. These settings include the number of network layers, the number of nodes in each layer, and the number of hyper parameters. Heuristic algorithms such as GA can give considerably good solutions for optimization problems. The GA works with randomly generated population. Each individual of population can be represented as a candidate solution of the optimization problem. This candidate solution is referred to as the chromosome, where the length of a chromosome is equal to the total number of parameters. For these parameters, all values have to find for perfect or suitable solution. Only that candidate solution will qualify as suggested solution which fulfills the criteria set by the objective function. After finding a solution, GA undergoes mutation and crossover steps for creating variations in the next generation. We use GA to find the optimized setting for our network. GA is good in providing best results with properly controlled setting but sometimes they are less accurate to fine-tune the parameters near local optimum points. So, we are using SGA for finding optimized setting for our network parameters.

Usually in GA, the optimal result is obtained after the mutation step. We implement local search operation after getting the result and for feature subset this result will be sent back to compute the fitness values again. For SGA when replacement is required, the feature subset has a chance to improve itself locally, the result is sent back for computation which help to gain its fitness value. This step is performed

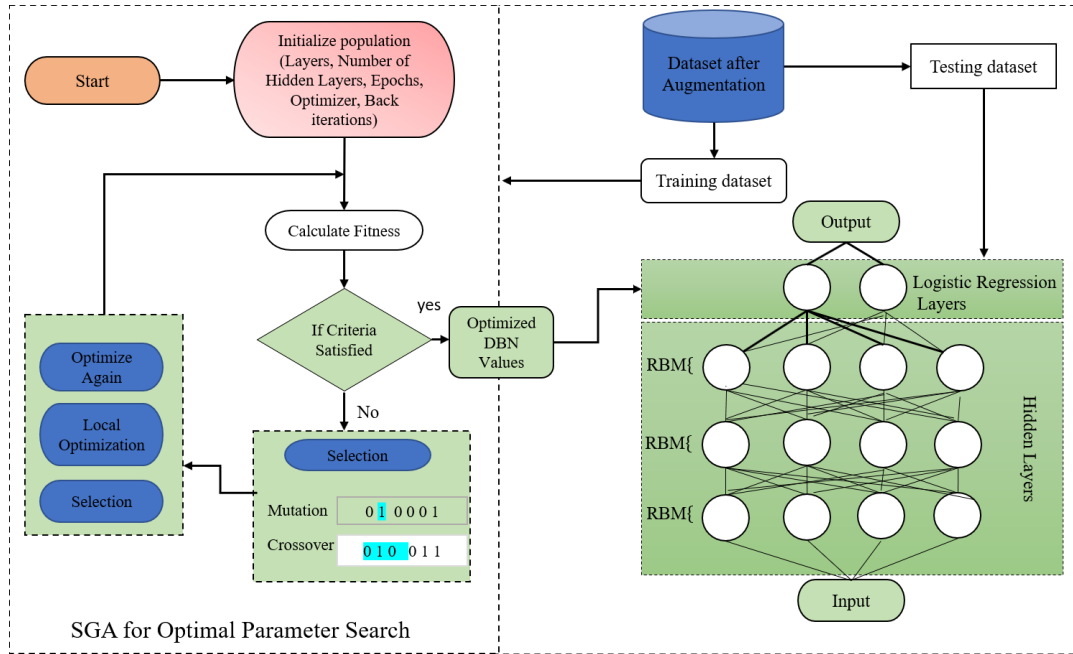


FIGURE 4. Visualization of proposed approach used for optimization of DBN using SGA.

repeatedly until SGA obtained desired optimal value. It provides the optimized values for DBN parameters. The optimal values solve the problem of optimization and increase the efficiency of the system. The algorithm for optimization of DBN using GA and SGA is shown as Algorithm 2. This procedure takes data sets, classes, labels, and the number of generations as input and outputs results on DBN trained on best parameters. At first, the pre-processing is done on data by normalizing the data and converting the labels into one-hot representation. After that, the number of samples and features from data set are retrieved. Then the genetic algorithm generates different numbers of networks based on the network parameters such as number of layers, epochs, iterations, and the data set. After initializing the population, the network generates different generation and based on the natural selection criteria (mutation and crossover) it selects the best individual (parameters) that gave good performance. Algorithm 2 shows Deep Belief Network using GA and SGA.

V. EXPERIMENTS AND RESULTS

We performed extensive experiments for the assessment of proposed Optimized GAN-based Deep Learning model (OGDL) using 10-Fold cross validation. We compared the proposed approach with previously proposed [2], [3] autonomic system and other deep learning based algorithms. We implemented all of the following approaches and compared their results with the proposed OGD (DBN+SGA) approach as shown in Table 7 and Table 8. Table 7 shows the accuracy using selected feature selection methods using different methods on Sensors. Table 8 shows the accuracy on

Algorithm 2 Deep Belief Networks Parameters Optimization Using Stacked Genetic Algorithm

```

1: Input: Data set, Labels, Classes, Generations.
2: Output: optimized parameters.
3: Procedure = OPTIMIZEDDBN(Data
  set, Labels, Classes, Generations)
4: Labels = OneHotEncode(Labels)
5: Dataset = MinMaxNorm(Dataset)
6: NumFeatures = FindFeatures(Dataset)
7: NumSamples = FindSamples(Dataset)
8: for Each Layer in Hidden Layers do
9:   for Each Epoch in Epoches do
10:    for Each Iteration in Iterations BackProps do
11:     network = CreateIndividual(Layers, Epoch, Iterations,
      Dataset, Labels, Classes)
12:     Networks.Add(network)
13:    end for
14:   end for
15: end for
16: for Each gen in Generations do
17:   GeneratedFeatures = GenertaeFeatures (Generator,
      class, nsamples)
18: end for
19: accuracy = TrainNetworks(Networks)
20: Networks = Crossover(Networks, accuracy)
21: Networks = Mutate(Networks, accuracy)
22: Networks = EvolveNetworks(Networks, accuracy)
23: bestnetworks = SortNetworks(Networks)

```

selected feature selection methods using different methods on Effectors.

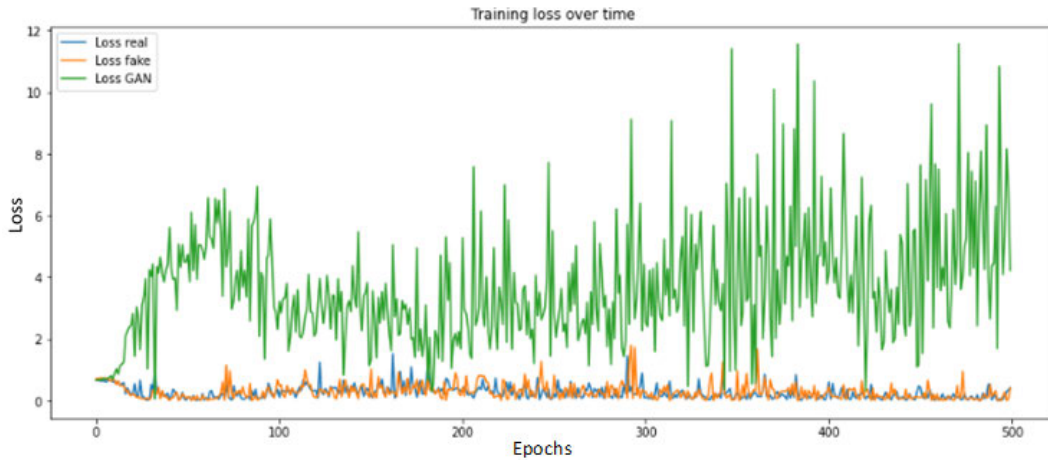


FIGURE 5. Training loss of CGAN.

TABLE 4. Details of parameters used for the training of CGAN for augmentation.

| Stage | Hyperparameter | Value |
|------------------|---|--------------------------------|
| Initialization | Weights | Xavier(Defaul) |
| | Discriminator (Sigmoid) Generator (tanh) | Default |
| Training | Epochs | 5000 |
| Model Parameters | Batch Size | 64 |
| | Loss Function | Binary Cross Entropy |
| | Optimizer | Adam (Alpha= 0.002 Beta = 0.5) |
| | Total Parameters | 1,260,241 |
| | Trainable Parameters | 724,611 |
| | Non-Trainable Parameters | 535,630 |

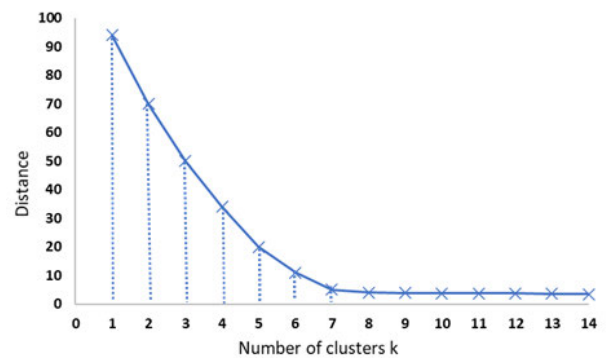


FIGURE 6. Elbow method for optimal k.

A. COMPARISON WITH STATE-OF-THE-ART MACHINE LEARNING AND DEEP LEARNING APPROACHES

1) GENERATIVE ADVERSARIAL NETWORKS (GANs)

We performed data augmentation for generating data. The details regarding parameters used to perform this study are shown in Table 4 that contains details about different stages of hyperparameter settings and their values along with selected model parameters. We initially used default settings in training of GAN for data augmentation. We also set default for the activation function for Discriminator and Generator. Total epochs executed were 5,000 in our experiments with batch size 64. For getting better results, Binary Cross Entropy is used as the GAN loss function and Adam optimizer was selected for optimization.

Figure 5 shows the training loss curves of CGAN used for generation of features. The green curve represents the loss of GAN which shows that as the number of epoch increases the loss of GAN increases, making it difficult for GAN to discriminate between real and generated features and as a result it treats them as same. On the other side, the loss of G represented as fake-features in orange curve decreases and the G generates features similar to the real ones.

Table 5 shows the results of Effectors before and after augmentation using GAN. The results clearly show the

improvement in performance after adding the augmentation data using GAN.

We also observed that before augmentation the performance of deep learning approaches (proposed DBN+SGA, DNN) is low as compared to lazy learning approaches, however, after augmentation they outperform the lazy-learning approaches which means the performance of deep learning algorithms is directly associated with size of data and features.

2) CASE-BASED REASONING (CBR)

We have performed experiments on traditional CBR [2] and Cluster-based CBR[3] for comparison of results. The Figure 6 shows the optimal values of clusters using k-means clustering for fetching cases from case-based repository. It shows the optimal value at the curve bend is 7. Figure 6 shows the Elbow method for optimal value of k.

We plotted the data point after generating the features through GAN. Figure 7 presents the clusters of features that show the generated features using GAN for seven classes that is class 0 to class 6. The data points of different classes are visualized in different colors. Fake generated

TABLE 5. Results of effectors on before and after augmentation using CGAN (*Before augmentation (B Aug) *After augmentation (A Aug)).

| Feature Selection/ Approaches | CBR [2] [3] | | DNN | | DBN | | DBN+GA | | Proposed OGDGDL (DBN+SGA) | |
|---------------------------------------|-------------|--------|--------|--------|--------|--------|--------|--------|---------------------------|--------|
| | *B Aug | *A Aug | *B Aug | *A Aug | *B Aug | *A Aug | *B Aug | *A Aug | *B Aug | *A Aug |
| Selected feature Selection algorithms | | | | | | | | | | |
| Random Forest Feature Selection | 65 | 91 | 46 | 80 | 59 | 94 | 60 | 94 | 62 | 97 |
| GA | 61 | 95 | 38 | 76 | 49 | 95 | 57 | 95 | 62 | 97 |
| Chi-Square | 70 | 96 | 51 | 73 | 57 | 92 | 64 | 96 | 71 | 97 |
| Recursive Features Elimination | 70 | 85 | 18 | 69 | 61 | 88 | 66 | 90 | 72 | 97 |
| Info Gain | 74 | 84 | 57 | 69 | 60 | 86 | 64 | 90 | 75 | 96 |
| Ruzzo Tumpa | 68 | 84 | 59 | 69 | 63 | 87 | 65 | 88 | 69 | 89 |

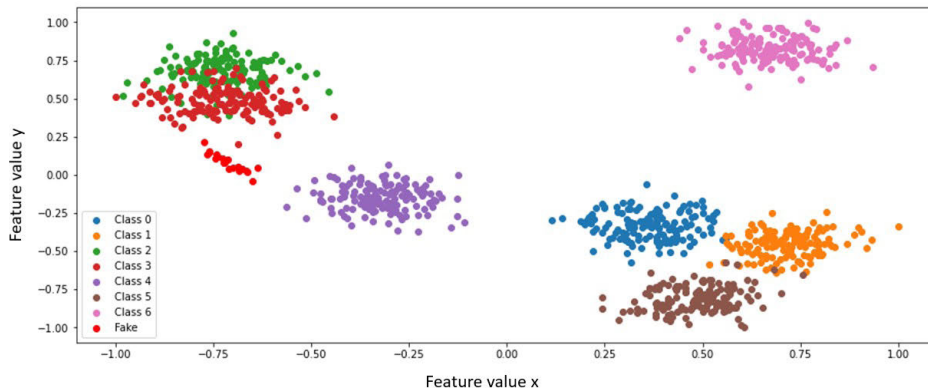


FIGURE 7. Optimal number of Data clusters.

features are shown in red color. The graph shows the generated features are close to the actual classes that means that the generated features are identical to the actual class. The results show that CBR achieves accuracy score of 91% on RF features selection, 95% on GA features selection, 96% on Chi-square, 85% on RFE, 84% on info-gain and 84% on Ruzzo-Tompa. Results shows that CBR performs best on features selected using Chi-Square features selection technique.

3) DEEP NEURAL NETWORK (DNN)

We also performed experiments using Multi-Layer Perceptron (MLP) with 5 hidden layers using log-like loss function with learning rate of 1e-5. Experimental results on different selected features achieved accuracy score of 80% using Random forest features selection, 76% using genetic algorithm, 73% using chi-square, and 69% using RFE, IG and RT.

4) DEEP BELIEF NETWORK (DBN)

DBN is a generative graphical model that is composed of multiple layers. They are composed of unsupervised networks such as Restricted Boltzmann Machines (RBM). We used hidden layer size of 200, learning rate of 0.1, number of epochs as 80 and batch size of 60. Accuracy scores using different features selected using feature selection techniques were evaluated. The evaluation results showed that 95% accuracy score is achieved using GA, 94% using RF, 92% using Chi-square, 82% using RFE, 87% using IG and 87% using RT. General parameter setting for DBN network is shown in Table 6.

TABLE 6. Parameter setting for DBN network.

| Parameter | Value |
|-------------------------|------------|
| hidden_layers_structure | [200, 200] |
| learning_rate_rbm | 0.01 |
| learning_rate | 0.1 |
| n_epochs_rbm | 80 |
| n_iter_backprop | 70 |
| batch_size | 60 |

5) DBN+GA

As in deep learning, hyperparameters plays a crucial role in model training and improvement in performance. For this purpose, we used GA for the hyperparameters optimization of DBN. Accuracy results showed that by optimizing DBN using GA the performance of DBN is increased. By using different features selection techniques, it achieves accuracy results of 94% using RF feature selection, 95% using GA, 96% using Chi-square, 90% using RFE, and 88% using RT.

6) PROPOSED OGDGDL (DBN+SGA) MODEL

We performed experiments using proposed OGDGDL model. For parameter optimization we used SGA. By using the local optimization of new individuals generated using natural selection, it adopted to have better individuals which leads to better classification results. The experimental results showed that it achieved 97% accuracy using features selection techniques GA, RF, Chi-Square, and RFE, 96% using IG and 89% using RT. The results show that our proposed OGDGDL model outperformed other techniques.

Figures 8 and 9 show the F1-Scores of techniques with different features selected on Sensors and Effectors respectively.

TABLE 7. Accuracy on selected feature selection methods using CBR, DNN and DBN on sensors.

| Selected feature selection algorithms | Features selected by each algorithm | CBR [2] [3] | DNN | DBN | DBN+GA | Proposed OGDL (DBN+SGA) |
|---------------------------------------|--|----------------|-------|-------|--------|-------------------------------|
| Random Forest Feature Selection | F10, F11, F13, F18, F5, F6, F7, F17, F2, F21, F4 | 85 | 88.75 | 87.88 | 91.63 | 92.71 |
| GA | F5, F6, F7, F19, F23 | 85 | 40.25 | 87.88 | 90.88 | 92.63 |
| Chi-Square | F15, F24, F16, F22, F5, F13, F10, F14, F7 | 84 | 74.13 | 90.25 | 91.25 | 92.63 |
| Recursive Features Elimination | F5, F6, F17, F10, F1, F12, F14, F7, F19, F8, F16 | 83 | 85.36 | 90.13 | 91.63 | 92.63 |
| Info Gain | F0, F1, F3, F4, F17, F15, F10, F11, F6, F7, F8 | 86 | 62.88 | 88.25 | 90 | 92 |
| Ruzzo Tumpa | F4, F7, F21, F23, F11, F13, F10 | 86 | 40.25 | 87.88 | 90.25 | 92 |

TABLE 8. Accuracy on selected feature selection methods using CBR, DNN and DBN with CGAN on effectors.

| Selected feature selection algorithms | Features selected by each algorithm | CBR [2] [3] | DDN | DBN | DBN+GA | Proposed OGDL (DBN+SGA) |
|---------------------------------------|--|----------------|-----|-----|--------|-------------------------------|
| Random Forest Feature Selection | F10, F11, F13, F18, F5, F6, F7, F17, F2, F21, F4 | 91 | 80 | 94 | 94 | 97 |
| GA | F5, F6, F7, F19, F23 | 95 | 76 | 95 | 95 | 97 |
| Chi-Square | F15, F24, F16, F22, F5, F13, F10, F14, F7 | 96 | 73 | 92 | 96 | 97 |
| Recursive Features Elimination | F5, F6, F17, F10, F1, F12, F14, F7, F19, F8, F16 | 85 | 69 | 88 | 90 | 97 |
| Info Gain | F0, F1, F3, F4, F17, F15, F10, F11, F6, F7, F8 | 84 | 69 | 86 | 90 | 96 |
| Ruzzo Tumpa | F4, F7, F21, F23, F11, F13, F10 | 84 | 69 | 87 | 88 | 89 |

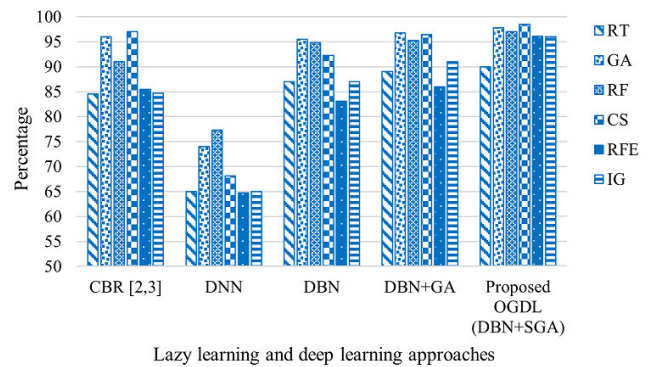
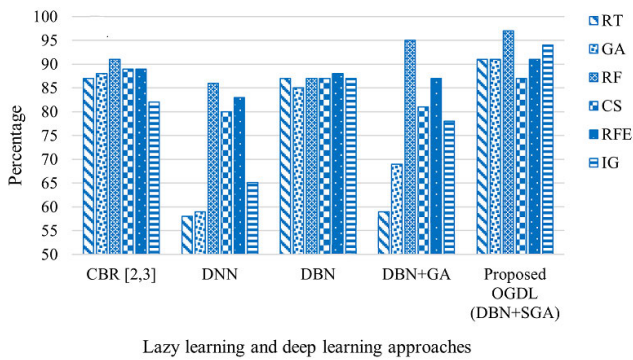


FIGURE 8. Comparison of F1-Scores with different features selection techniques on Sensors using different classification techniques.

FIGURE 9. Comparison of F1-Scores with different features selection techniques on Effectors using different classification techniques.

It can be seen that for Sensors it achieved best F1-Score using RF features selection. For Effectors it achieved good F1-Score using features selected by chi-square and RF. According to Table 7, prediction accuracy is high for parameters selected by RF on Sensors. According to Table 8, prediction accuracy is high for parameters selected using GA and RF for Effectors. So, for our proposed (DBN+SGA) model, we selected RF as feature selection technique. The results show that optimization of parameters is an important aspect in deep learning.

Figure 10 shows the loss curves for the Deep Belief Network. The loss till the 5th epoch shows no convergence towards minimum loss but after 25 epochs the loss reduces to 0.2 from 1.8 and then the convergence became slow. We performed experiments for observing actual vs predicted accuracy. Figure 11 and 12 shows the values of actual vs predict values for a few performance parameters. We have selected 11 performance parameters for workload performance prediction. We are showing two performance parameters here with their actual and predicted values for three types of workload. Figure 11 – 13 shows the predictions for OLTP, DSS and Mixed workloads on INNODB_PAGES_READ performance

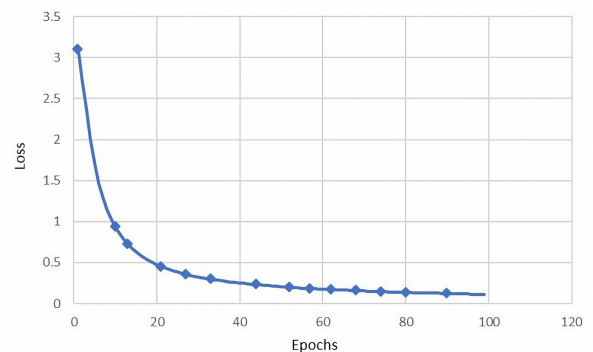


FIGURE 10. Deep belief network loss curves.

parameter. Figure 14 - 16 shows the predictions for OLTP, DSS and Mixed workloads on HANDLER_READ_KEY performance parameter.

Figure 17 shows the confusion matrix of the performance measure INNODB_BUFFER_POOL_READ_REQU ESTS with Chi-Square feature selection using DBN classifier. Figure 18 shows the confusion matrix of proposed OGDL (DBN+SGA).

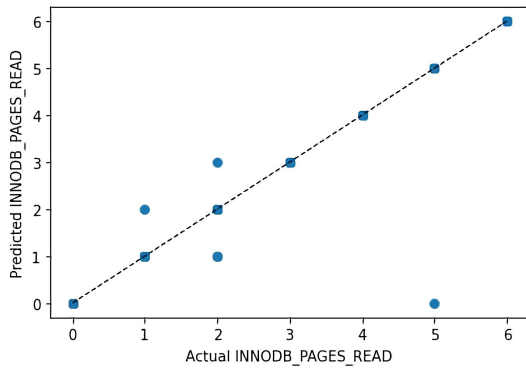


FIGURE 11. Queries testing results on proposed OGD model using INNODB_PAGES_READ performance parameter for OLTP Queries.

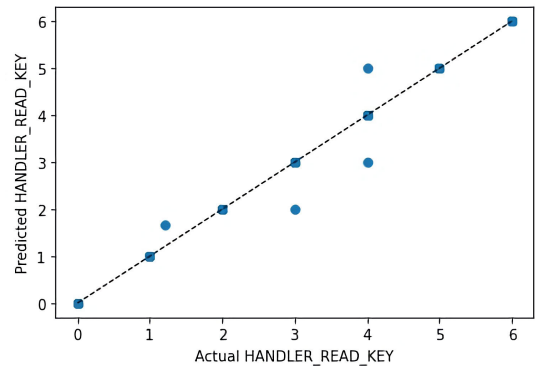


FIGURE 14. Queries testing results on proposed OGD model using HANDLER_READ_KEY performance parameter for OLTP queries.

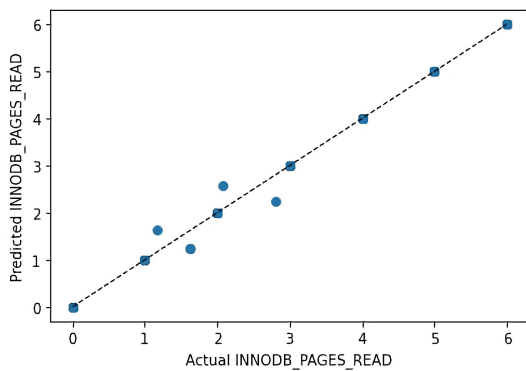


FIGURE 12. Queries testing results on proposed OGD model using INNODB_PAGES_READ performance parameter for DSS queries.

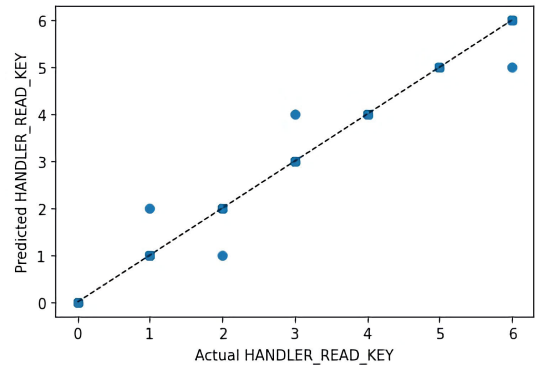


FIGURE 15. Queries testing results on proposed OGD model using HANDLER_READ_KEY performance parameter for DSS queries.

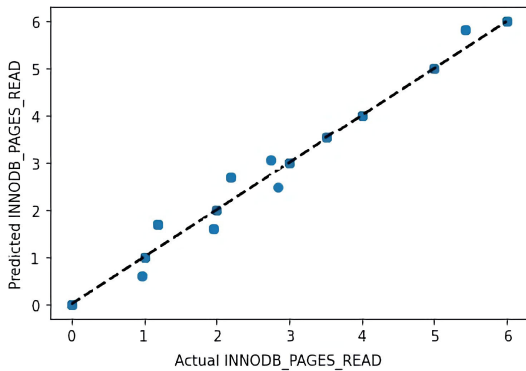


FIGURE 13. Queries testing results on proposed OGD model using INNODB_PAGES_READ performance parameter for Mixed queries.

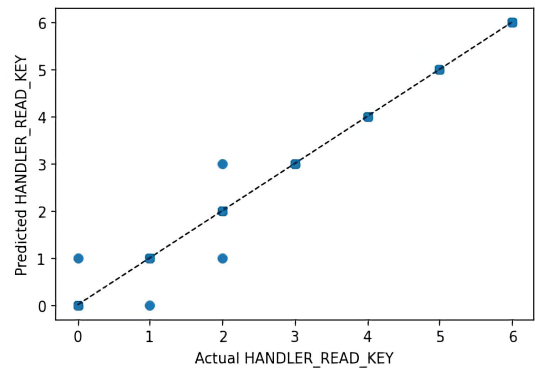


FIGURE 16. Queries testing results on proposed OGD model using HANDLER_READ_KEY performance parameter for Mixed queries.

B. VALIDATION THROUGH POST-HOC TESTS

We have performed post-hoc tests for validation of results. In these tests all pairwise comparisons are performed between different classifiers to identify the significance of pairwise difference for the proposed solution [43]. For hypothesis testing parametric and non-parametric tests can be performed. Non-parametric tests are also called distribution free because they do not assumed homogeneity of normal distribution. Hypothesis tested by the non-parametric test may be more appropriate for the research investigation. We are

more interested in significance test rather than info about population. So, we performed non-parametric test such as Friedman on PMV using KEEL software [44]. A set of pairwise comparison can be mapped to set of associated hypotheses. Any of post-hoc tests which can be used for non-parametric tests, works on this set of hypotheses. Test statistics for comparing the *i*th and *j*th classifiers is shown in equation 17 [45].

$$z = \frac{(\text{Rank}_i - \text{Rank}_j)}{\sqrt{\frac{n(n+1)}{6M}}} \tag{17}$$

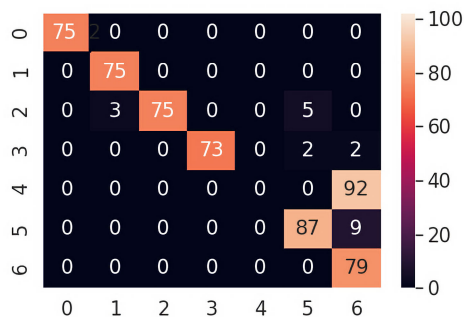


FIGURE 17. Confusion matrix using DBN.

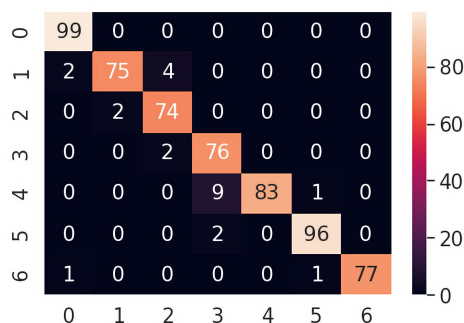


FIGURE 18. Confusion matrix results of proposed OGDL model.

where z is used corresponding probability from the value of normal distribution, that is compared with value of α . $Rank_i$ is the average rank computed through the Friedman test for the i th classifier, n is the number of classifiers to be compared and M is the number of data sets used in the comparison. Difference between the tests is only the way they adjust the value of α to satisfy for multiple comparisons. We also consider post-hoc tests like Shaffer, Holm and Nemenyi. We have conducted Friedman test $n * n$ which tests all hypotheses in $n * n$ comparison having logical relation among them. Table 9 shows the ranking obtained for algorithms using Friedman procedure. Test ranks the proposed OGDL model as best classifier. For results the corresponding probability (p -value) with confidence α level is computed, and p -value is returned for $\alpha = 0.05$ and $\alpha = 0.01$. For PMV, the Table 10 shows the p -value for Holm and Shaffer procedures with $\alpha = 0.05$ which shows the p -value and adjustment of through these procedures. From the normal distribution table, the z value is used to determine the p -value that is compared with the level of significance and adjusted for various comparisons. We have tested and formulated the null hypothesis. Assumption for null hypothesis is that all the algorithms are equal and there is no significant difference between algorithms. In this work, we have considered PMV and five approaches DDN, CBR, DBN, DBN+GA, and proposed OGDL that we compared and evaluated for hypothesis testing using p -values and adjusted p -values (APVs) [45]–[47]. Significance of a hypothesis test can be found using p -value. The p -value will serve as the probability of observing the two samples on the basis of basic hypothesis (null hypothesis).

TABLE 9. Algorithms and their ranking by using Friedman procedure.

| Algorithms | Ranking | Best classifier |
|-------------------------|---------|--------------------|
| Proposed OGDL (DBN+SGA) | 1 | Average classifier |
| DBN+GA | 2.3333 | |
| DBN | 3.0833 | |
| CBR | 3.5833 | Worst classifier |
| DNN | 5 | |

The p -value can be interpreted on the basis of pre-selected significant level called Alpha (α). For α a common value is 5% or 0.05. If p -value lies below the selected significant level, then test rejects the null hypothesis.

$$p - \text{value} \leq \alpha \quad \text{rejects the null hypothesis}$$

$$p - \text{value} > \alpha \quad \text{fails to rejects the null hypothesis}$$

Nemenyi’s procedure rejects those hypotheses that have an unadjusted p -value ≤ 0.005 . Holm’s procedure rejects those hypotheses that have an unadjusted p -value ≤ 0.007143 . Shaffer’s procedure rejects those hypotheses that have an unadjusted p -value ≤ 0.005 .

Bergmann’s procedure rejects the following hypotheses:

- CBR vs. Proposed OGDL (DBN+SGA)
- DNN vs. DBN+GA
- DNN vs. Proposed OGDL (DBN+SGA)

These results show that performance of all approaches is similar but with slight difference. P -value within multiple comparisons reflects the probability error for some comparisons and does not considers the remaining comparisons from same family. To solve this issue, APVs can be used because they consider all multiple tests that are conducted. Table 11 shows APV values obtained through Shaffer, Holman, Bergman and Nemenyi for PMV. Different post-hoc procedures used in analysis compute APVs in different ways [46] which are shown in the equations 18– 21:

$$\text{Holm APV}_i : \min \{v; 1\}, \text{ where } v = \max \{ (m-j + 1) p_j : 1 \leq j \leq i \} \quad (18)$$

$$\min \{v; 1\}, \text{ where } v = m \cdot p_i.$$

$$\begin{aligned} \text{Nemenyi APV}_i \\ = \min \{v; 1\}, \text{ where } v = m \cdot p_i \end{aligned} \quad (19)$$

$$\begin{aligned} \text{Shaffer static APV}_i \\ = \min \{v; 1\}, \text{ where } v = \max \{t_j p_j : 1 \leq j \leq i\} \end{aligned} \quad (20)$$

$$\begin{aligned} \text{Bergmann–Hommel APV}_i : \min \{v; 1\}, \text{ where} \\ v = \max \{ |I| \cdot \min \{p_j, j \in I : I \text{ exhaustive}, i \in I\} \} \end{aligned} \quad (21)$$

Our proposed OGDL is suitable for workload management in large-scale data repositories. Already existing methods are working with small data sets and with that data sets their performance is good. However, they degrades the performance on large volume data. The proposed OGDL model helps to predict the performance of workload better than the existing models because existing models are based on machine learning or lazy learning. However OGDL model is based on deep learning and produce better results. Our proposed model

TABLE 10. The P-values for $\alpha = 0.05$ for Holm and Shaffer.

| I | Algorithms | $z=(RO-Ri)/SE$ | P - value | Holm | Shaffer |
|----|------------------------------------|----------------|-----------|----------|----------|
| 1 | CBR vs. DBN | 0.547723 | 0.583882 | 0.05 | 0.05 |
| 2 | DBN vs. DBN+GA | 0.821584 | 0.411314 | 0.025 | 0.025 |
| 3 | CBR vs. DBN+GA | 1.369306 | 0.170904 | 0.016667 | 0.016667 |
| 4 | DBN+GA vs. proposed OGDL (DBN+SGA) | 1.460593 | 0.144127 | 0.0125 | 0.0125 |
| 5 | CBR vs. DNN | 1.551881 | 0.120691 | 0.01 | 0.01 |
| 6 | DNN vs. DBN | 2.099603 | 0.035764 | 0.008333 | 0.008333 |
| 7 | DBN vs. proposed OGDL (DBN+SGA) | 2.282177 | 0.022479 | 0.007143 | 0.008333 |
| 8 | CBR vs. proposed OGDL (DBN+SGA) | 2.8299 | 0.004656 | 0.00625 | 0.008333 |
| 9 | DNN vs. DBN+GA | 2.921187 | 0.003487 | 0.005556 | 0.008333 |
| 10 | DNN vs. proposed OGDL (DBN+SGA) | 4.38178 | 0.000012 | 0.005 | 0.005 |

TABLE 11. The APV-values obtained for Holm and Shaffer.

| I | Algorithms | P-value | APV-Nemenyi | APV-Holm | APV-Shaffer |
|----|------------------------------------|----------|-------------|----------|-------------|
| 1 | CBR vs. DBN | 0.583882 | 5.838824 | 0.822628 | 0.822628 |
| 2 | DBN vs. DBN+GA | 0.411314 | 4.113138 | 0.822628 | 0.822628 |
| 3 | CBR vs. DBN+GA | 0.170904 | 1.709035 | 0.603454 | 0.576508 |
| 4 | DBN+GA vs. proposed OGDL (DBN+SGA) | 0.144127 | 1.44127 | 0.603454 | 0.576508 |
| 5 | CBR vs. DNN | 0.120691 | 1.206908 | 0.603454 | 0.482763 |
| 6 | DNN vs. DBN | 0.035764 | 0.357638 | 0.214583 | 0.214583 |
| 7 | DBN vs. proposed OGDL (DBN+SGA) | 0.022479 | 0.224789 | 0.157352 | 0.134873 |
| 8 | CBR vs. proposed OGDL (DBN+SGA) | 0.004656 | 0.046563 | 0.03725 | 0.027938 |
| 9 | DNN vs. DBN+GA | 0.003487 | 0.03487 | 0.031383 | 0.020922 |
| 10 | DNN vs. proposed OGDL (DBN+SGA) | 0.000012 | 0.000118 | 0.000118 | 0.000118 |

overcome the non-availability of large-scale data publicly available by artificially generating data through CGAN. That is required for modeling of large-scale data repositories for better training and testing of data. With the increase in volume of data, existing methods show less accuracy in comparison with their accuracy with limited data. So, for large-scale data, our proposed solution is efficient. Results show that our proposed model's accuracy is higher as compared to already existing methods.

VI. CONCLUSION

This study proposed a performance prediction model OGDL for large-scale data repositories using deep learning approach. To overcome the issue of non-availability of large data sets, CGAN is proposed for data augmentation that generated better augmentation results. Among different features selection techniques, Random Forest feature selection remained the best in our experiments. Different deep learning approaches are examined through experiments and it is observed that the deep learning approach DBN with SGA optimization performed the best. We compared our proposed OGDL model results with state-of-the-art eager learning and lazy learning approaches. The comparison showed that deep learning approaches are better for solving the workload performance prediction in large-scale data repositories. Results shows that proposed OGDL model outperformed the existing approaches. On average, 30% increase in data size is observed after augmentation. An increase of 6-8% in accuracy is observed after augmentation in comparison with existing studies as shown in Table 5. With respect to autonomic perspective of our model, our results on Sensors and Effectors are also promising. The proposed model is a step forward towards performance modeling in autonomic systems for large-scale data repositories.

Limitation of the proposed model is that it is experimented and tested using MySQL database. Other well-known commercial databases such as ORACLE by Oracle Corporation and DB2 by IBM can be investigated for workload management and perform tuning. Data preparation is done for workload features and performance features extraction by executing various workloads. However, for making it more autonomic it could be done detecting the database and data warehouse workload/ queries using Natural Language Processing (NLP) techniques.

Future work includes proposing a framework that can perform workload characterization and its performance prediction autonomically. Handling adaptiveness of changing behaviour of workload can be of further interest. Machine learning, lazy learning and deep learning mechanism could be further investigated to understand how learning can be improved by performing calibration of the learnt model. Further, to improve performance and speed, the concepts of parallelism could be introduced. Moreover, other autonomic characteristics could be incorporated moving towards fully autonomic systems.

REFERENCES

- [1] B. Raza, A. Sher, S. Afzal, A. K. Malik, A. Anjum, Y. J. Kumar, and M. Faheem, "Autonomic workload performance tuning in large-scale data repositories," *Knowl. Inf. Syst.*, vol. 61, no. 1, pp. 27–63, Oct. 2019, doi: [10.1007/s10115-018-1272-0](https://doi.org/10.1007/s10115-018-1272-0).
- [2] B. Raza, Y. J. Kumar, A. K. Malik, A. Anjum, and M. Faheem, "Performance prediction and adaptation for database management system workload using case-based reasoning approach," *Inf. Syst.*, vol. 76, pp. 46–58, Jul. 2018, doi: [10.1016/j.is.2018.04.005](https://doi.org/10.1016/j.is.2018.04.005).
- [3] B. Raza, A. Aslam, A. Sher, A. K. Malik, and M. Faheem, "Autonomic performance prediction framework for data warehouse queries using lazy learning approach," *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106216, doi: [10.1016/j.asoc.2020.106216](https://doi.org/10.1016/j.asoc.2020.106216).
- [4] Z. Zewdu, M. K. Denko, and M. Libsie, "Workload characterization of autonomic DBMSs using statistical and data mining techniques," in *Proc. Int. Conf. Adv. Inf. Netw. Appl. Workshops (AINA)*, May 2009, pp. 244–249, doi: [10.1109/WAINA.2009.159](https://doi.org/10.1109/WAINA.2009.159).

- [5] M. Awad and D. A. Menascé, "Automatic workload characterization using system log analysis," in *Proc. 41st Int. IT Capacity Perform. Conf.*, 2015, pp. 1–11.
- [6] N. Shaheen, B. Raza, and A. K. Malik, "A CBR model for workload characterization in autonomic database management system," in *Proc. 14th Int. Conf. Emerg. Technol. (ICET)*, Nov. 2018, pp. 1–6, doi: [10.1109/ICET.2018.8603615](https://doi.org/10.1109/ICET.2018.8603615).
- [7] M. C. Calzarossa, L. Massari, and D. Tessera, "Workload characterization: A survey revisited," *ACM Comput. Surv.*, vol. 48, no. 3, pp. 1–43, Feb. 2016, doi: [10.1145/2856127](https://doi.org/10.1145/2856127).
- [8] S. R. Shishira, A. Kandasamy, and K. Chandrasekaran, "Workload characterization: Survey of current approaches and research challenges," in *Proc. 7th Int. Conf. Comput. Commun. Technol. (ICCCCT)*, 2017, pp. 151–156, doi: [10.1145/3154979.3155003](https://doi.org/10.1145/3154979.3155003).
- [9] M. Salehie and L. Tahvildari, "Autonomic computing," *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–7, Jul. 2005, doi: [10.1145/1082983.1083082](https://doi.org/10.1145/1082983.1083082).
- [10] A. Mateen, B. Raza, M. Sher, M. M. Awais, and N. Mustapha, "Workload management: A technology perspective with respect to self-characteristics," *Artif. Intell. Rev.*, vol. 41, no. 4, pp. 463–489, Apr. 2014, doi: [10.1007/s10462-012-9320-8](https://doi.org/10.1007/s10462-012-9320-8).
- [11] M. Zhang, P. Martin, W. Powley, and J. Chen, "Workload management in database management systems: A taxonomy," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1386–1402, Jul. 2018, doi: [10.1109/TKDE.2017.2767044](https://doi.org/10.1109/TKDE.2017.2767044).
- [12] B. Raza, A. Mateen, T. Hussain, and M. M. Awais, "Autonomic success in database management systems," in *Proc. 8th IEEE/ACIS Int. Conf. Comput. Inf. Sci. (ICIS)*, Jun. 2009, pp. 439–444, doi: [10.1109/ICIS.2009.202](https://doi.org/10.1109/ICIS.2009.202).
- [13] A. Mateen, B. Raza, M. Sher, M. M. Awais, and T. Hussain, "Evolution of autonomic database management systems," in *Proc. 2nd Int. Conf. Comput. Automat. Eng. (ICCAE)*, vol. 1, Feb. 2010, pp. 33–37, doi: [10.1109/ICCAE.2010.5452007](https://doi.org/10.1109/ICCAE.2010.5452007).
- [14] B. Raza, A. Mateen, M. Sher, M. M. Awais, and T. Hussain, "Autonomic view of query optimizers in database management systems," in *Proc. 8th ACIS Int. Conf. Softw. Eng. Res., Manage. Appl. (SERA)*, 2010, pp. 3–8, doi: [10.1109/SERA.2010.11](https://doi.org/10.1109/SERA.2010.11).
- [15] A. Mateen, B. Raza, T. Hussain, and M. M. Awais, "Autonomicity in universal database DB2," in *Proc. 8th IEEE/ACIS Int. Conf. Comput. Inf. Sci. (ICIS)*, Jun. 2009, pp. 445–450, doi: [10.1109/ICIS.2009.203](https://doi.org/10.1109/ICIS.2009.203).
- [16] B. Raza, A. Mateen, M. Sher, M. M. Awais, and T. Hussain, "Autonomicity in oracle database management system," in *Proc. Int. Conf. Data Storage Data Eng. (DSDE)*, Feb. 2010, pp. 296–300, doi: [10.1109/DSDE.2010.72](https://doi.org/10.1109/DSDE.2010.72).
- [17] N. Shaheen, B. Raza, A. R. Shahid, and H. Alquhayz, "A novel optimized case-based reasoning approach with K-means clustering and genetic algorithm for predicting multi-class workload characterization in autonomic database and data warehouse system," *IEEE Access*, vol. 8, pp. 105713–105727, 2020, doi: [10.1109/access.2020.3000139](https://doi.org/10.1109/access.2020.3000139).
- [18] C. Gupta, A. Mehta, and U. Dayal, "PQR: Predicting query execution times for autonomous workload management," in *Proc. Int. Conf. Auton. Comput.*, Jun. 2008, pp. 13–22, doi: [10.1109/ICAC.2008.12](https://doi.org/10.1109/ICAC.2008.12).
- [19] L. Ma, D. Van Aken, A. Hefny, G. Mezerhane, A. Pavlo, and G. J. Gordon, "Query-based workload forecasting for self-driving database management systems," in *Proc. Int. Conf. Manage. Data*, May 2018, p. 15, doi: [10.1145/3183713.3196908](https://doi.org/10.1145/3183713.3196908).
- [20] S. F. Rodd, Â. Umakant, P. Kulkarni, and U. P. Kulkarni, "Adaptive self-tuning techniques for performance tuning of database systems: A fuzzy-based approach with tuning moderation," *Soft Comput.*, vol. 19, no. 7, pp. 2039–2045, Jul. 2015, doi: [10.1007/s00500-014-1389-3](https://doi.org/10.1007/s00500-014-1389-3).
- [21] R. Singhal and M. Nambiar, "Predicting SQL query execution time for large data volume," in *Proc. 20th Int. Database Eng. Appl. Symp.*, 2016, pp. 378–385, doi: [10.1145/2938503.2938552](https://doi.org/10.1145/2938503.2938552).
- [22] W. Wu, Y. Chi, H. Hacigümüş, and J. F. Naughton, "Towards predicting query execution time for concurrent and dynamic database workloads," *Proc. VLDB Endowment*, vol. 6, no. 10, pp. 925–936, Aug. 2013, doi: [10.14778/2536206.2536219](https://doi.org/10.14778/2536206.2536219).
- [23] W. Wu, Y. Chi, S. Zhu, J. Tatemura, H. Hacigümüş, and J. F. Naughton, "Predicting query execution time: Are optimizer cost models really unusable?" in *Proc. IEEE 29th Int. Conf. Data Eng. (ICDE)*, Apr. 2013, pp. 1081–1092, doi: [10.1109/ICDE.2013.6544899](https://doi.org/10.1109/ICDE.2013.6544899).
- [24] C. Chi, Y. Zhou, and X. Ye, "Performance prediction for performance-sensitive queries based on algorithmic complexity," *Tsinghua Sci. Technol.*, vol. 18, no. 6, pp. 618–628, Dec. 2013, doi: [10.1109/tst.2013.6678907](https://doi.org/10.1109/tst.2013.6678907).
- [25] A. Khattab, A. Algergawy, and A. Sarhan, "MAG: A performance evaluation framework for database systems," *Knowl.-Based Syst.*, vol. 85, pp. 245–255, Sep. 2015, doi: [10.1016/j.knosys.2015.05.010](https://doi.org/10.1016/j.knosys.2015.05.010).
- [26] B. Mozafari, C. Curino, A. Jindal, and S. Madden, "Performance and resource modeling in highly-concurrent OLTP workloads," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, 2013, p. 301, doi: [10.1145/2463676.2467800](https://doi.org/10.1145/2463676.2467800).
- [27] Z. Ding, Z. Wei, and H. Chen, "A software cybernetics approach to self-tuning performance of on-line transaction processing systems," *J. Syst. Softw.*, vol. 124, pp. 247–259, Feb. 2017.
- [28] R. Marcus and O. Papaemmanouil, "Plan-structured deep neural network models for query performance prediction," *Proc. VLDB Endowment*, vol. 12, no. 11, pp. 1733–1746, Jul. 2019, doi: [10.14778/3342263.3342646](https://doi.org/10.14778/3342263.3342646).
- [29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020, doi: [10.1145/3422622](https://doi.org/10.1145/3422622).
- [30] M. Li, Z. Liu, X. Shi, and H. Jin, "ATCS: Auto-tuning configurations of big data frameworks based on generative adversarial nets," *IEEE Access*, vol. 8, pp. 50485–50496, 2020, doi: [10.1109/ACCESS.2020.2979812](https://doi.org/10.1109/ACCESS.2020.2979812).
- [31] F. Ugo, D. S. Alfredo, P. Francesca, Z. Paolo, and P. Francesco, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," *Inf. Sci.*, vol. 479, pp. 448–455, Apr. 2019, doi: [10.1016/j.ins.2017.12.030](https://doi.org/10.1016/j.ins.2017.12.030).
- [32] K. T. Chui, R. W. Liu, M. Zhao, and P. O. D. Pablos, "Predicting students' performance with school and family tutoring using generative adversarial network-based deep support vector machine," *IEEE Access*, vol. 8, pp. 86745–86752, 2020, doi: [10.1109/ACCESS.2020.2992869](https://doi.org/10.1109/ACCESS.2020.2992869).
- [33] J. Ma, H. Xu, J. Jiang, X. Mei, and X.-P. Zhang, "DDcGAN: A dual-discriminator conditional generative adversarial network for multi-resolution image fusion," *IEEE Trans. Image Process.*, vol. 29, pp. 4980–4995, 2020, doi: [10.1109/TIP.2020.2977573](https://doi.org/10.1109/TIP.2020.2977573).
- [34] R. Vinayakumar, M. Alazab, S. Srinivasan, Q.-V. Pham, S. K. Padannayil, and K. Simran, "A visualized botnet detection system based deep learning for the Internet of Things networks of smart cities," *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 4436–4456, Jul. 2020, doi: [10.1109/TIA.2020.2971952](https://doi.org/10.1109/TIA.2020.2971952).
- [35] V. Ravi, M. Alazab, S. Srinivasan, A. Arunachalam, and K. P. Soman, "Adversarial defense: DGA-based botnets and DNS homographs detection through integrated deep learning," *IEEE Trans. Eng. Manag.*, early access, Mar. 12, 2021, doi: [10.1109/TEM.2021.3059664](https://doi.org/10.1109/TEM.2021.3059664).
- [36] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating effectiveness of shallow and deep networks to intrusion detection system," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1282–1289, doi: [10.1109/ICACCI.2017.8126018](https://doi.org/10.1109/ICACCI.2017.8126018).
- [37] S. Sriram, R. Vinayakumar, M. Alazab, and S. Kp, "Network flow based IoT botnet attack detection using deep learning," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 189–194, doi: [10.1109/INFOCOMWKSHPS50562.2020.9162668](https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9162668).
- [38] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi, "Time series forecasting using a deep belief network with restricted Boltzmann machines," *Neurocomputing*, vol. 137, pp. 47–56, Aug. 2014, doi: [10.1016/j.neucom.2013.03.047](https://doi.org/10.1016/j.neucom.2013.03.047).
- [39] S. A. Ali, B. Raza, A. K. Malik, A. R. Shahid, M. Faheem, H. Alquhayz, and Y. J. Kumar, "An optimally configured and improved deep belief network (OCI-DBN) approach for heart disease prediction based on Ruzzo-Tompa and stacked genetic algorithm," *IEEE Access*, vol. 8, pp. 65947–65958, 2020, doi: [10.1109/ACCESS.2020.2985646](https://doi.org/10.1109/ACCESS.2020.2985646).
- [40] J. Gai, J. Shen, H. Wang, and Y. Hu, "A parameter-optimized DBN using GOA and its application in fault diagnosis of gearbox," *Shock Vib.*, vol. 2020, pp. 1–11, Mar. 2020, doi: [10.1155/2020/4294095](https://doi.org/10.1155/2020/4294095).
- [41] *TPC-Homepage V5*. Accessed: May 18, 2020. [Online]. Available: <http://www.tpc.org/>
- [42] Q. Fan, K. Zeitouni, N. Xiong, Q. Wu, S. Camtepe, and Y.-C. Tian, "Nash equilibrium-based semantic cache in mobile sensor grid database systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 9, pp. 2550–2561, Sep. 2017, doi: [10.1109/TSMC.2016.2523949](https://doi.org/10.1109/TSMC.2016.2523949).
- [43] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Stat.*, vol. 11, no. 1, pp. 86–92, Mar. 1940, doi: [10.1214/aoms/1177731944](https://doi.org/10.1214/aoms/1177731944).
- [44] *KEEL: A Software Tool to Assess Evolutionary Algorithms for Data Mining Problems (Regression, Classification, Clustering, Pattern Mining and So On)*. Accessed: Oct. 13, 2020. [Online]. Available: <http://www.keel.es/>

- [45] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.
- [46] S. García and F. Herrera, "An extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons," *J. Mach. Learn. Res.*, vol. 9, no. 12, pp. 2677–2694, 2008.
- [47] B. Trawiński, M. Smętek, Z. Telec, and T. Lasota, "Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms," *Int. J. Appl. Math. Comput. Sci.*, vol. 22, no. 4, pp. 867–881, Dec. 2012, doi: [10.2478/v10006-012-0064-z](https://doi.org/10.2478/v10006-012-0064-z).



NUSRAT SHAHEEN is currently pursuing the Ph.D. degree with the Department of Computer Science, COMSATS University at Islamabad (CUI), Islamabad, Pakistan. She is also working as a Lecturer with the Department of Computer Science, CUI. Her research interests include databases, data mining, machine learning, and autonomic workload management.



BASIT RAZA received the master's degree in computer science from the University of Central Punjab, Lahore, Pakistan, and the Ph.D. degree in computer science from International Islamic University Islamabad and the University of Technology Malaysia, in 2014. He is currently an Assistant Professor with the Department of Computer Science, COMSATS University at Islamabad (CUI), Islamabad, Pakistan. He has authored several articles in refereed journals. His research interests include database management systems, data mining, data warehousing, machine learning, deep learning, and artificial intelligence. He has been serving as a Reviewer for prestigious journals, such as *Applied Soft Computing*, *Swarm and Evolutionary Computation*, *Swarm Intelligence*, *Applied Intelligence*, *IEEE ACCESS*, and *Future Generation Computer Systems*.



AHMAD RAZA SHAHID received the Ph.D. degree in computer science in York, U.K., in 2012. During his Ph.D. studies, he worked on automatically building a WordNet for four languages, namely English, German, French, and Greek. Since his Ph.D., he has been working in the areas of computer vision and pattern recognition, machine learning, and natural language processing. He is currently working as an Assistant Professor with the COMSATS Institute of Information Technology, Islamabad, Pakistan. A few of the problems that he has worked on include cancer detection, pedestrian detection, driver fatigue detection, and data mining.



AHMAD KAMRAN MALIK received the Ph.D. degree from the Vienna University of Technology (TU-Wien), Vienna, Austria. He is currently an Assistant Professor with COMSATS University at Islamabad (CUI), Islamabad, Pakistan. He has published a book and many research papers in reputed international journals and conferences. His current research interests include data science, social network analysis, and access control.

• • •