**International Academic Institute for Science and Technology**

International Academic Journal of Science and Engineering
Vol. 12, No. 3, 2025, pp. 447-457.

ISSN 2454-3896

**International Academic Journal of Science and Engineering**

www.iaiest.com

# Structured Data-Based Query Analysis Processing by Deep Learning Algorithms and Meta Heuristic Optimization

R. Jeevitha [a], L. Ramesh [b]

*[a] Research Scholar, Department of Computer Applications, School of Computing Sciences, Vels Institute of Science, Technology & Advanced Studies (VISTAS), Chennai, Tamil Nadu, India. E-mail: jeevitharesearchwork@gmail.com, Orcid: https://orcid.org/0009-0002-0504-9252*
*[b] Assistant Professor, Department of Computer Applications, School of Computing Sciences, Vels Institute of Science, Technology & Advanced Studies (VISTAS), Chennai, Tamil Nadu, India. E-mail: lramesh.scs@vistas.ac.in*

## Abstract

The paper is concerned with the issue of query performance prediction in a database management system in the specific context of estimating query latency in interactive applications such as data visualization. It proposes a new algorithm of structured data query processing incorporating deep learning with optimization algorithms. In particular, the paper uses a multilayer convolutional graph transfer gradient encoder neural network to classify input queries and is thereafter optimized with the help of the enhanced glowworm flower pollination virtual optimization algorithm. The test methodology was done on different query types such as aggregate, join and basic queries. Experimental outcomes demonstrate a great improvement in performance measures, as the proposed method is characterized by a training accuracy of 93, an average precision of 90, and 41% decrease in query processing time on aggregate queries. The convergence was optimized at 78%, which surpasses the traditional methods on all categories tested. The system has proved to be much higher than the traditional query performance prediction techniques because it is scalable and can adapt to dynamic data volumes and query complexities. These results indicate that the application of AI-based models may significantly improve real-time query processing, especially when it comes to dynamic settings. The method has great potential for use in healthcare systems and financial industries where fast access to data is essential. The research finds the hybrid approach to deep learning and optimization algorithms as a potential solution to more efficient query performance prediction in contemporary database systems.

**Keywords:** Structured Data, Query Processing, Deep Learning, Classification Model, Optimization Algorithm, Database Management, Query Performance.



DOI: 10.71086/IAJSE/V12I3/IAJSE1280
Received: 23 July 2025/Revised: 27 August 2025/Accepted: 15 September 2025/Published: 30 September 2025

## Introduction

Non-relational databases are used to store data in practically every industry in the current digital era. Search engines, real-time web apps, and massive data sets are all using Non-Structured Query Language (NoSQL) databases more and more. Relational databases and other traditional databases can now be replaced with NoSQL databases. NoSQL databases contain a lot of data in document stores, key-value data stores, wide-column stores, graph stores as a result of technological advancements. MongoDB, CouchDB, Cassandra, and other distributed systems are built on architecture of distributed systems to store large amounts of data, in contrast to conventional databases. Numerous organisations are progressively investigating methods to comprehend and examine this massive amount of unstructured data. "Big Data" is transforming the way that data is currently organised, managed, and stored (Shaikhha et al., 2023). Specifically, "Big Data," an open-source platform for storing enormous volumes of semi-structured, unstructured, and structured data. It can be difficult to find a trustworthy method for creating NoSQL queries from natural language (English). Amateur users can interact with database method by using NoSQL technique. Without having to remember query syntactic technique for non-relational databases, the paradigm makes it easier for people and computers to communicate. A subfield of computer science, linguistics, information engineering, AI called natural language processing (NLP) examines how people and computers interact with natural language (Katsogiannis-Meimarakis & Koutrika, 2023). Natural language processing (NLP) uses traditional machine translation to translate text between languages. The design of hand-derived metrics, training models solely on plan-level data, putting forward mathematical models of relational operators, or ad hoc combinations of plan-level and operator-level data have been the main strategies utilised in the past for query performance prediction. In order to choose or derive bits of information from a query plan or query operator that may correlate with its latency, all of these approaches rely on clever human feature engineering (Li et al., 2022). As a manual process, feature engineering typically takes a lot of work from human expertise. More significantly, though, it does not scale well as database management systems get more sophisticated. Intelligent maintenance solutions are becoming more and more popular among industrial equipment manufacturers and operators. This is caused by a number of factors, including recent development of computationally efficient ML methods and steadily expanding number of commercial solutions for data gathering, transmission, storage (Mosqueira-Rey et al., 2023). From basic condition monitoring to problem detection to fault diagnosis as well as prognosis, topic of decision support for intelligent maintenance covers it all. Due to the great degree of operating state fluctuation and the scarcity of data that is representative of all of these situations, the latter, commonly referred to as "predictive maintenance," is rarely feasible these days. Algorithms for defect diagnosis and detection, however, aim at prediction and are used in a number of industrial systems (Li et al., 2024). A "plug-and-play" generic method is typically insufficient to identify defects or deterioration in complicated systems. The physical systems' distinctiveness and possible critical failure modes are to blame for this. Furthermore, the massive volumes of diverse data streams generated by these systems make it extremely difficult to store and analyse data effectively at scale (Kumar et al., 2025).

### Research Objectives

to suggest a new method for processing structure data queries that is based on an optimisation algorithm and a deep learning classification model. Here, a multilayer convolutional graph transfer gradient encoder neural network has classed the input query, and an enhanced glowworm flower pollination virtual optimisation technique has been used to optimise the classified query.

## Literature Survey

Database administration tasks can be greatly aided by machine learning approaches like computer vision, deep learning (DL), reinforcement learning (RL), and natural language processing (NLP) (Wang et al., 2021). NLP, for example, can be used to better comprehend the context of queries, enabling the system to provide more tailored and pertinent responses. By identifying the most effective tactics based on feedback from prior executions, RL can be utilised to optimise query execution in the interim. Additionally, deep learning can be used to identify patterns in data and more accurately forecast query outcomes. Because they rely on static execution plans that do not adjust to changing workload factors, traditional query processing approaches frequently fail when applied to large datasets. The limitations of traditional query optimisation techniques in Big Data contexts were highlighted in a noteworthy study by (Alanazi, 2022), which showed that existing techniques are unable to handle the complexity and dynamic of large-scale data. The authors argued that since the non-stable algorithm of execution does not consider the dynamic data distribution and access pattern, the algorithm is inefficient. It was this observation that was the basis of the investigation of more flexible and clever approaches. Incorporation of AI into query optimisation has been an interesting method of enhancing performance in recent years. One of the groundbreaking studies offered a machine

learning-based approach towards query optimisation, and demonstrated that the cost of executing a query could be predicted using previous query performance data (Dang et al., 2021). They found that the AI could transform the traditional methods of database management by indicating that the application of machine learning models could reduce the time of query response by up to 30 %. The importance of continuing the research based on the patterns of data access was also brought out in their research, as it is essential that AI systems evolve along with the constantly changing requirements of the workload of Big Data. On the basis of this, (Bhatti et al., 2023) benchmarked several ML methods, including decision tree and neural network, to query optimisation. In their results, deep learning models were more effective than the traditional ones in accuracy and flexibility, particularly in complex query situations. The authors note that deep learning will be a superior choice to maximise the query performance of the Big Data systems due to its ability to detect intricate patterns in the data access behaviour. Also, (Abbasiantaeb & Momtazi, 2021) explored the application of the reinforcement learning to adaptive query optimisation where an agent can be trained to optimise execution plans to feedback provided by the database system. By using a reinforcement learning framework that modifies execution strategies in real-time, their study showed a substantial decrease in average query execution time. A query-response paradigm that can react to a range of queries, including imperative, compound, complex, assertive, and interrogative forms, was shown in Work (Lavanya & Sasikala, 2021). A Deep Learning-based method for translating English queries into MongoDB queries was provided by the author (Nassiri & Akhloufi, 2023). For this conversion, they used an encoder-decoder machine translation technique. NLQ text input is converted into a vector by encoder and then sent to decoder. Decoder predicts NoSQL queries using a deep neural network. English to SQL conversion was suggested by Work (Chiche & Yitagesu, 2022). Their method, for instance, translates text queries or English questions into SQL queries. Databases will be used to operate it later. Their strategy and technique are fluid and generic. It can handle generic NLIDB systems for both small and large applications (Zhu et al., 2021). Any input token of appropriate terminals found in input NLQ will be used in place of the related attribute in relational table or pertinent SQL operators. User may quickly and automatically configure interface. For tables and attributes, it depends on Metadata as well as Semantic sets. Ambiguities in the input NLQ can be handled by it. A discrete artificial bee colony (dABCSPARQL) technique for reordering SPARQL queries was proposed in Work (Dogra et al., 2022). It is based on a novel heuristic method. After classifying the query statements into chained, star, cyclic, chain-star queries using syntax tree parsing, the method turns statements into pre-ordered vectors according to various query types (Samant et al., 2022). A collection of reasonably optimal query execution plans is initially obtained by utilising the genetic algorithms fast convergence properties. The execution plan is then converted into the multi-ant colony algorithm's initial routing pheromone value. A system called "POLYPHONET" has been created by Work (Choi et al., 2022) to extract information from social networks. It can identify links between individuals and groups of individuals and retrieve keywords for an individual. As noted by (Liang et al., 2024), social networks and semantics are the dualism sides of the coin in context of semantic web. A major social user aggregation based on the FOAF ontology has been provided by the authors (Bernius et al., 2022). Nevertheless, neither the provenance nor the time of any information has been recorded by the model. Additionally, a particular property may have conflicting values, and the user is responsible for determining whether or not to keep the information. Under the set of user attributes, such as sexual orientation, birthday, and city, Work (Qi & Shabrina, 2023) has presented a mechanised coordinating computation that can register the potential comparability between these initial qualities and creep information from social sites. The more information that is gathered, the more accurate the computation. At any rate, the proximity of most of the key OSNs makes ambiguous information easily available. In addition to gathering social data from many networks, another OSN aggregator suggested by similarly groups, rates, and alerts users to friends' actions.

## Proposed Model

In order to create the training dataset for a particular database backend, schema graph abstraction that depicts the entities and relationships found in the database was derived. Then, as explained below, run random walks on it to produce a number of subgraphs that describe various database queries. To create the training set, these are further converted to synthetic natural language. As a result, the training data consists of computationally-based searches that mimic human queries rather than queries created by humans. An outline of the stages taken by the deep learning classification procedure-based structural data query processing is shown in fig.1.

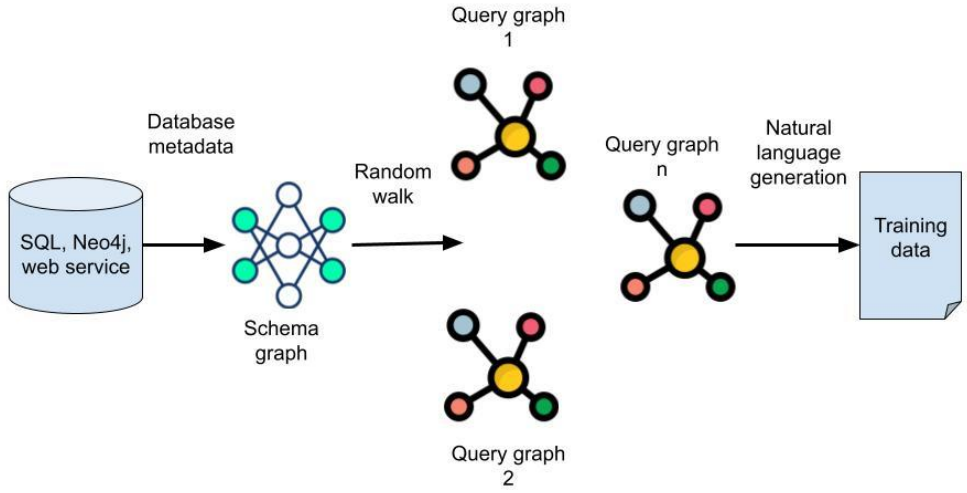



**Fig. 1: Structure Data Query Processing Based on a Deep Learning Classification Model**

The first phase is called query decomposition, which entails taking user-submitted query requests, parsing them, and then figuring out the syntactic structure and semantics of the question. Next, use a query statement to generate a matching parse tree. Data retrieval is the second stage, which entails creating a storage site matrix, utilising a metadata table to find sites where required data are housed, and looking at the parse tree to determine which data tables are part of the query. The query optimiser develops the query execution plan in the third stage, referred to as global optimisation, using the parse tree and the data storage matrix. Based on a query-optimized execution plan, the query is then broken down into a number of subqueries, which are then spread to different distributed sites for execution. Local optimisation is the fourth step: Following the upper layer's assignment of the query request to the local processing site, it is comparable to a centralised database environment. Consequently, query optimisation may now be done using the centralised database approach. The user receives the final merged query results once the results of subqueries run on each site have been combined. The third stage of global optimisation is the main topic of this work. Execution cost of the query plan is determined utilizing the cost function; the optimal query plan is the one with the lowest query cost.

**Query Processing Based Classification Using Multilayer Convolutional Graph Transfer Gradient Encoder Neural Network (MCGTGENN):**

According to this model, the cardinality of the query Q is determined as follows for a join operation between relations T1 and T2, represented as T2 on the join predicate k1 = k2, where k1 and k2 are characteristics of relations T1 and T2, by equation (1)

$$|\mathrm{Q}| = \mathrm{P_{T1}} * \mathrm{P_{T2}} * |\ \mathrm{T_1} \bowtie_{\ \mathrm{k1}-k2}\ \mathrm{T_2}| \qquad (1)$$

where the cardinality of the joint operation on relations T1 and T2 is computed as follows: where PT1 is the selectivity on the relation T1 for a given predicate (filter) in Q, and PT2 is for T2 by equation (2)

$$|\mathrm{T_1} \bowtie \mathrm{k_{k1-k2}}\ \mathrm{T_2}| = \frac{|\mathrm{T1}|\times|T2|}{\max(\mathrm{dom}(k1),\mathrm{dom}(k2))} \qquad (2)$$

In this relationship, Xi represents input to the i-th node (input layer), θj represents biases of the j-th node (hidden layer), m represents the number of input nodes, and Wij represents the weight of the edge from the i-th node (input layer) to the j-th node (hidden layer). Output of every hidden node is obtained as an equation (3) to a sigmoid function.

$$U_j = \mathrm{sigmoid}\big(U_j\big) = \frac{1}{\left(1+\exp\left(-s_j\right)\right)} \cdot j = 1.2.\cdots.h$$

$$Z_k = \sum_{j=1}^{l} \left(W_{jk} \cdot U_j\right) - \theta_k' \cdot k = 1.2.\cdots.s$$

$$Z_k = \mathrm{sigmoid}(Z_k) = \frac{1}{(1+\exp\left(-Z_k\right))}, k = 1.2, \cdots.s \qquad (3)$$

The final outputs can be defined as follows once the hidden nodes have been calculated. This connects node j -th (hidden layer) to node k -th (output layer), where Wjk is the weight of the edge and θ_k^' is the bias of node k -th. Atom and bond data of multigraph G is prepared as discrete representations consisting of binary digits that characterise constituent elements and bonds, such as the distance between atoms and their group and periodic number.

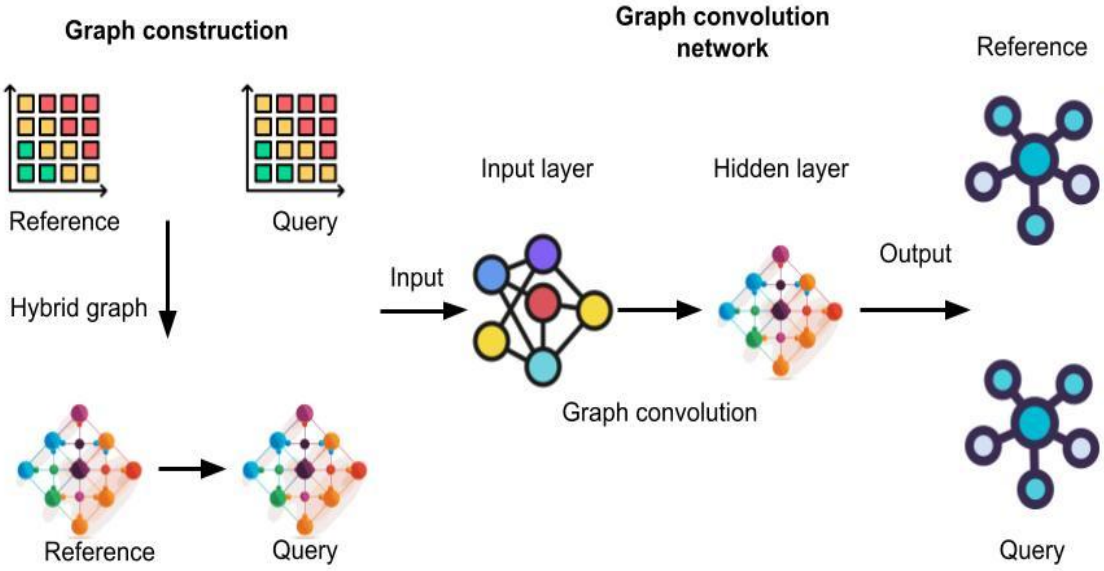



**Fig. 2: Convolutional Graph Transfer Model in Structured Data Query Processing**

To identify and spread shared data between reference and query data, fig. 2 displays a hybrid graph of intra-dataset and inter-dataset cell mappings utilizing mutual nearest neighbours of canonical correlation vectors that project various datasets onto a correlated low-dimensional space. Cells from reference as well as new datasets are then projected onto the same latent space based on the generated graph using semi-supervised GCN, which ensures that cells with the same labels are present in the same population.

The embedding layer converts discrete representations into continuous representations. After that, the convolutional layer receives continuous representations.

In the (t+1)-th convolutional layer by equation (4)

$$v_i^{(t+1)} = g\left[\left(\sum_{j,k} v_j^{(t)} \oplus u_{(i,j))_k}\right)W_c^{(t)} + v_i^{(t)}W_s^{(t)} + b^{(t)}\right] \qquad (4)$$

where $\oplus$ indicates that the atom and bond feature vectors of the i-th atom's neighbouring atoms have been concatenated. Nevertheless, distinct neighbour interactions are not discernible in the convolution formulation since every neighbour shares the weight matrix. Consequently, the following formulation of a novel kind of convolution can be made using a conventional edge-gating technique: Using a neighbour feature vector z(i,j)k by equation (5)

$$z_{(i,j)k}^{(t)} = v_i^{(t)} \oplus v_j^{(t)} \oplus u_{(i,j)k}$$

$$v_i^{(t+1)} = v_i^{(t)} + \sum_{j,k} \sigma\left(z_{(i,j)k}^{(t)}W_c^{(t)} + b_c^{(t)}\right) \circ g\left(z_{(i,j)k}^{(t)}W_s^{(t)} + b_s^{(t)}\right) \qquad (5)$$

The bottleneck layer teaches auto encoders the underlying data structure, which is frequently compressed. The encoder layers encode the data into the compressed state, while the decoder layers decode it back to its original form. Despite their effectiveness, AEs, like artificial neural networks, do not take advantage of the information found in the vicinity of the input data. Unlike the stacking process used by other NN architectures, the convolution operation makes use of the spatial and temporal data structure to discover correlations more readily. Consider an in-context learning problem with N context tokens and an extra query token, indexed by N + 1. Regarding the linear regression issue, N+1-th token eN+1 = (xN+1, yN+1) = (xtest, ˆ ytest) = etest to the test input xtest and the related prediction ˆ ytest corresponds to N context tokens ej = (xj,yj) ∈ RNx+Ny, which correspond to N training points in D. Now establish their equivalence, refer to query and test token/data, as well as training and in-context data, interchangeably.

**Algorithm of MCGTGENN**

**Input : source data** $\{G^s = (V^s, A^s, X^s), Y^{sl}\}$**, target data** $\{G^t = (V^t, A^t, X^t)\}$**, domain critic training step** $n_d$**, coefficients** $\gamma, \lambda$**, learning rates** $\alpha_1, \alpha_2$

**Initialize parameters** $\theta_g$ **for representation learner** $f_g$

$\theta_c$ **for label classifier** $f_c$**, and** $\theta_d$ **for domain critic** $f_d$**;**

**while not converge do**

**for** $t = 1, \dots, n_d$ **do**

$H_g^s \leftarrow f_g(A^s, X^s; \theta_g), H_g^t \leftarrow f_g(A^t, X^t; \theta_g)$;

$N \leftarrow \min\{N^*, N^t\}$;

**Construct** $H = \{h_i\}_{i=1}^N$ **with**

$h_i \leftarrow \varepsilon h^* + (1-\varepsilon)h^t$**, where** $\varepsilon$ **is a random number sampled from** $U[0,1]$**,** $h^s$ **and** $h^t$ **are sampled from** $H_g^s$ **and** $H_g^t$**, respectively;**

$\hat{H} \leftarrow \{H_g^s, H_g^t, H\}$;

$\theta_d \leftarrow \theta_d + \alpha_1 \cdot \nabla_{\theta_d}\{\mathcal{L}_d - \gamma \mathcal{L}_{\text{grad}}(\hat{H})\}$

**End**

$\theta \leftarrow \{\theta_g, \theta_c\}$

$\theta \leftarrow \theta - \alpha_2 \cdot \nabla_\theta\{\mathcal{L}_c + \lambda \mathcal{L}_d\}$

**End**

**Query Optimization Using Enhanced Glowworm Flower Pollination Virtual Optimisation (EGFPVO) Technique**

The level, which represents the quantity of shining, is used by the agents to calculate fitness. The agent moves and broadcasts a higher luciferin value after locating a neighbouring position that is randomly selected. An issue that can cause load imbalance in parallel processing is agent's ability to become trapped without moving if it has no other neighbours.

When I t is represented by xi (t) during Luciferin-update stage, glow-worm's position is established, and J (xi (t)) is its precise primary function. Glow-worm I's luciferin level at time t is determined utilizing equation (6).

$$l_i(t) = (1-\rho)l_i(t-1) + \gamma J(x_i(t)) \quad (6)$$

Two luciferin-related variables are involved in the above equation: consistent luciferin improvement, denoted by γ, luciferin decay constant, represented by ρ (0 < ρ < 1) by equation (7)

$$j \in N_i \text{ iff Dis tance } e_{ij} < r\ d_i(t) \text{ and } l_j(t) > l_i(t) \quad (7)$$

Likelihood that glowworm i will travel in the direction of its neighbour j is computed utilizing equation (8).

$$p(t) = 1_j(t) - 1_i(t)$$

$$ij \sum l(t) - l(t)$$

$$k \in N_i(t)N^k \quad (8)$$

Equation (9) is utilised to update the glow-worm I's position.

$$rd_i(t) = \min\{rs, \max[0, rd_i(t-1) + \beta(nt - |N_i(t-1))]\} \quad (9)$$

Equation (10) is used to update local-decision range, where stands for neighbourhood, where st is step size.

$$x_i(t+1) = x_i(t) + st * \left\{\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}\right| \quad (10)$$

Global and local pollination search processes are the two distinct search methods used by the algorithm. The following equation could be used to quantitatively depict the global pollination process in accordance with first and third rules of FP (flower pollination) by equation (11)

$$x_i^{t+1} = x_i^t + \gamma \cdot L(\lambda) \cdot (\boldsymbol{g}^* - x_i^t)$$

$$x_i^{t+1} = x_i^t + \varepsilon \cdot \left(x_j^t - x_k^t\right) \quad (11)$$

Fourth rule states that whether flower pollination is local or global is determined by a switch probability p ∈ [0, 1]. As a result, this problem does not consider the distance between food sources. It is equal to rerunning a random search at each iteration when using the conventional search method, which results in an algorithm without directional optimisation and a very slow convergence speed.

## Experimental Analysis

Several tests that are used to assess performance of proposed model are described. A PC with the following specs was utilized to test the proposed hybrid model: Windows 7, SciPy, NumPy, Pandas, Keras, and Matplotlib frameworks, Python 2.7, Intel(R) Core (TM) i5-7500 CPU, 32-bit OS, and 4 GB RAM.

1. Real-World Datasets: The image vectors in the Deep1B public dataset were taken from a deep neural network. One billion 96-dimensional vectors are contained in it. The SIFT1B public dataset consists of SIFT features that have been collected manually. There are a billion vectors of 128 dimensions. Ali Commodity consists of 830 million 512-dimensional vectors that were sampled by Alibaba commodity photos. There are also 21 organised columns, including the colour, sleeve kind, style, creation time, etc., with the largest table of 6 million rows simulating heavy query demands.
2. Synthetic Workload Generation: To generate fake query workloads that can simulate different user requests to the database, a query generating tool was developed. With adjustable criteria such as the number of tables, the depth of a query, and the presence of filters, the tool would generate queries of varying complexity, such as simple select, join, aggregation, and sub-queries, to ensure a realistic simulation of user behaviour. The conditions are randomly generated using statistical distributions.
3. Statistics of Historical execution: o Execution statistics, including execution time, CPU and memory utilization, as well as I/O activities, are collected through the previous executions of the query. These are all kept in a special logging system, and these statistics provide valuable information about the performance of some searches in the long run.

The distribution of the artificial dataset is uniform. To properly control the tests, fixed the attribute cardinalities for unpartitioned attributes to 103 unless otherwise indicated. Because each partition creates a folder, partitioned properties are handled differently. This results in a huge number of folders and subfolders in order of product of all partitioned attribute cardinalities when there are several partitioned attributes. Using one, two, four partitioned characteristics with cardinalities of 104, 102, and 101, to lessen this effect. This fixes file systems' inability to list millions of files quickly and produces a comparable number of folders across various partition configurations.

## Result Analysis

Query performance with these three settings is displayed in fig. 3. Performance of one-dimensional point queries (1D-PQ) is displayed in the upper section. One-dimensional range enquiries (1D-RQ) are displayed at the bottom. y-axis is logarithmically scaled to base 2. It is evident that partitioning results in quicker response times less than a second for both range and point searches. Depending on the cluster size, Z-ordering is slower by a factor of 3 for point searches and 16–32 for range queries. In contrast to point searches, range queries typically require a greater number of files to be examined. Partitioning by a factor of 20 to 32 is faster than without using any optimisations. Keep in mind that Z-ordering exhibits the greatest sensitivity to the particular values that are searched, as well as the most fluctuation in response times. Additionally, Z-ordered range queries are two times faster than non-partitioned attribute queries. Despite being the slowest, searches against non-partitioned attributes exhibit nearly linear scalability in relation to the cluster size. However, because of network latencies and the resulting higher communication costs between four or eight worker nodes, it is difficult to further reduce query response times on partitioned attributes, which are already less than one second.

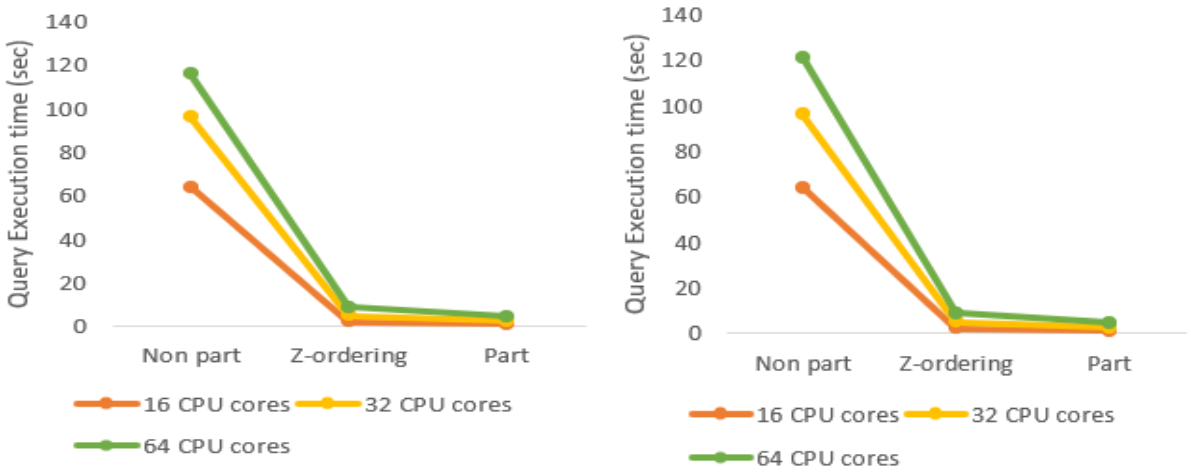



**Fig. 3: Response Times For 1D Point Queries: 1D-PQ (A) And 1D-RQ Range Queries (B)**

A blue histogram displaying the dataset's word and text distribution is shown in fig. 4. It is a semantic parsing dataset with hand annotations that includes both conventional and logical forms. The information in the dataset was taken from the internet. Spark's ability to use fle-level statistics, which show which attribute ranges are included in partition as well as thereby limit amount of data access during query processing, explains partitioned table's high query performance. However, Z-ordering has a drawback because rows cannot be chosen by merely examining partitions that contain the requested attribute ranges, particularly when range queries are involved.

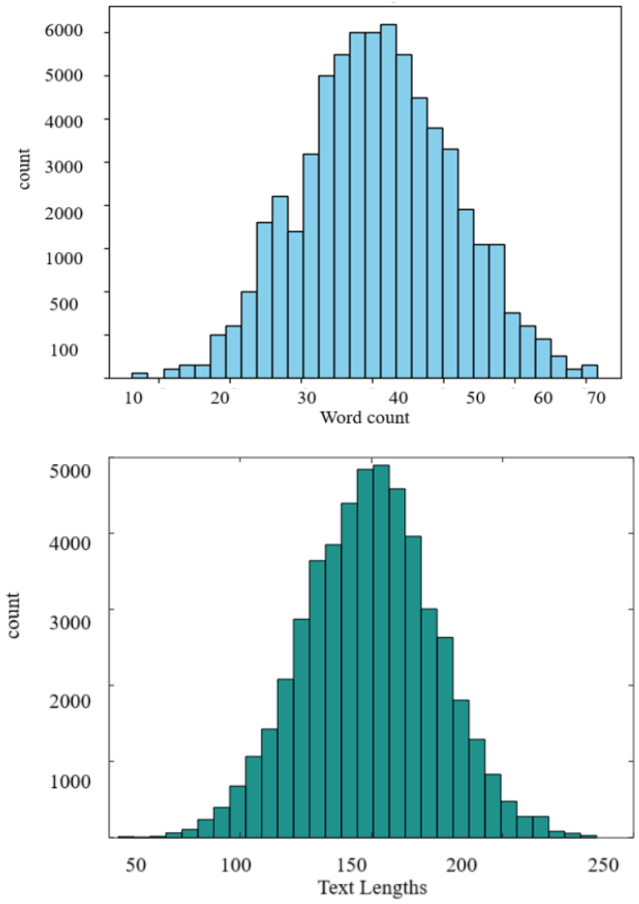
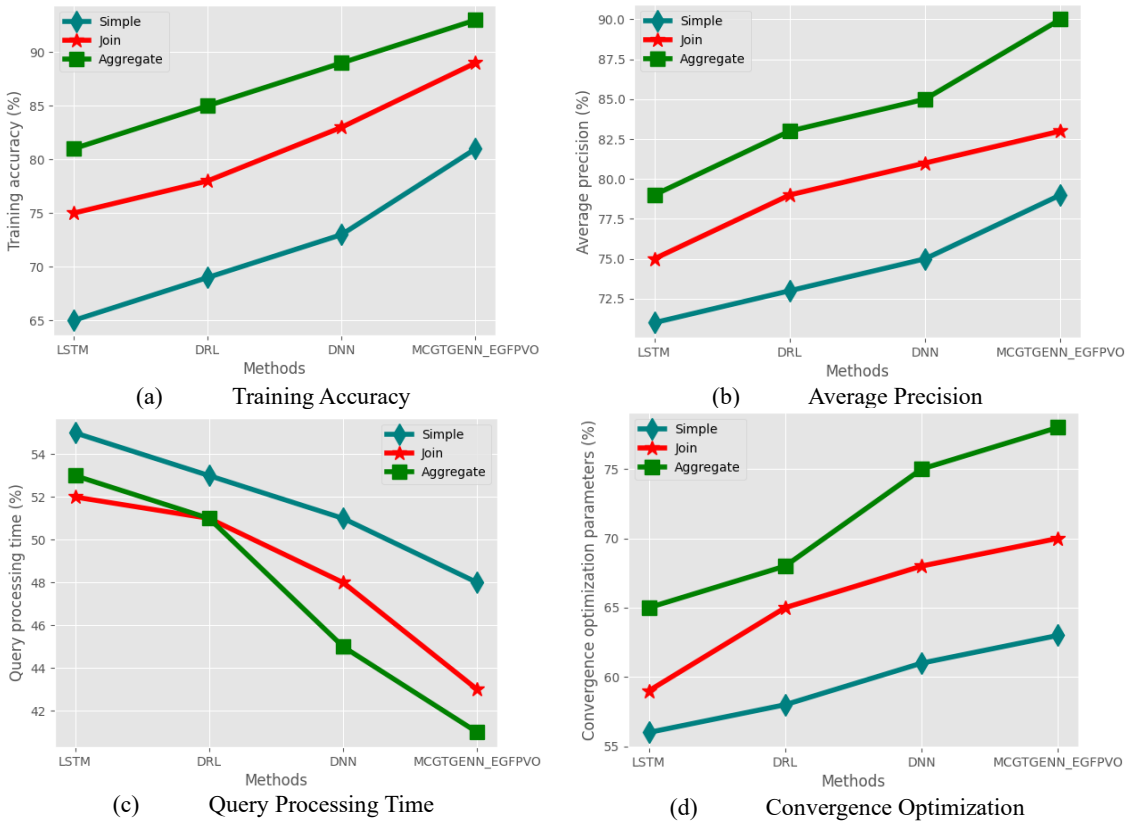



**Fig. 4: Word and Text Distribution**

**Table 1: Comparative Analysis Based on Query Type Between the Proposed and Existing Models**

| Query Type | Methods | Training accuracy (%) | Average precision (%) | Query processing time (%) | Convergence optimization parameters (%) |
|---|---|---|---|---|---|
| Simple | LSTM | 65 | 71 | 55 | 56 |
| | DRL | 69 | 73 | 53 | 58 |
| | DNN | 73 | 75 | 51 | 61 |
| | MCGTGENN_EGFPVO | 81 | 79 | 48 | 63 |
| Join | LSTM | 75 | 75 | 52 | 59 |
| | DRL | 78 | 79 | 51 | 65 |
| | DNN | 83 | 81 | 48 | 68 |
| | MCGTGENN_EGFPVO | 89 | 83 | 43 | 70 |
| Aggregate | LSTM | 81 | 79 | 53 | 65 |
| | DRL | 85 | 83 | 51 | 68 |
| | DNN | 89 | 85 | 45 | 75 |
| | MCGTGENN_EGFPVO | 93 | 90 | 41 | 78 |

As indicated in table 1, conventional query optimisation methods were used as a reference point for comparison. These techniques comprised cost-based optimisers that assess execution plans using predicted costs without adaptive learning capabilities and static query optimisation methodologies. Conventional optimisation techniques frequently rely on static heuristics and pre-set rules, notwithstanding their effectiveness in many situations. According to this study, these approaches could not work well in intricate settings with dynamic and unpredictable query patterns. AI models are flexible and can learn from past execution data to make well-informed choices about query preparation and execution tactics. The descriptive variables are Q, the target variable, xi, the response variable, and e0, the sum of the word squared error, where sound is cautious. Use a regression known as a function to minimise this error term to the potential moment in order to obtain a more accurate estimate. Therefore, as seen in fig 5(a)-(d), understanding the procedure to be monitored entails learning a model using academic statistics and evaluating a sample of invisible test data to determine reform accuracy.




(a) Training Accuracy (b) Average Precision

(c) Query Processing Time (d) Convergence Optimization

**Fig. 5: Comparative Analysis Between Proposed and Existing Model in Terms Of (A) Training Accuracy, (B) Average Precision, (C) Query Processing Time, (D) Convergence Optimization**

For example, NN learns on 60 query templates, the network's performance is evaluated on instances of 10 randomly chosen query templates that are "held out" of the training set. 10% of randomly chosen queries are "held out" of training set because there aren't enough query templates to employ the same technique. Modern neural network libraries use vectorization, which is the simultaneous application of mathematical operators to complete vectors, to speed up their models and cut down on the amount of time needed. Neural networks are thought to accomplish this by employing a fixed architecture, that is, a design that remains constant regardless of the specific input. In many applications, including computer vision, this is true. Libraries can vectorise their computation by assuming a fixed architecture, which allows them to assume that each weight's symbolic gradient will be the same for every item in the batch.

## Conclusion

This paper presents a new approach to query processing of structured data that can combine deep learning classification models with optimization algorithms and enhance query processing in database management systems. The combination of a multilayer convoluted graph transfer gradient encoder neural network (MCGTGENN) and an improved version of glowworm flower pollination virtual optimization (EGFPVO) technique enhances the effect of query processing considerably when proposed. Experimenting with the model showed that it had a training accuracy of 93% and an average precision of 90%, which illustrates the strength of the model in managing the various types of queries. It is important to note that the processing time of an aggregate query was decreased by 41 %, and convergence optimization was achieved at 78 %, which means that the model can optimize query processing. These findings indicate the possibility of the AI-driven models, particularly deep learning and optimization algorithms, being useful in the dynamic query optimization tasks. It can be seen that the query response time (decreasing the latency by half with all types of queries) indicates that the suggested approach is effective when working with large data sets and complicated queries. The importance of real-time adjustment to workload patterns and data distributions is also highlighted in this piece of work and is frequently not considered by classic methods of optimization. The results of the study can be especially applied in the areas of use where the quick response rate of queries is necessary, e.g., healthcare and financial systems. These industries require the use of high-performance query processing because they rely on the real-time data retrieval process in making high-impact decisions. To conduct future research, would recommend considering using this hybrid deep learning and optimization in the distributed database systems to improve scalability and efficiency. Also, further exploration into a way to combine the reinforcement learning process with the adaptive query optimization approach would enhance the system itself in terms of being adaptive to changing query dynamics. More investigations may also be done on how the model can be applied in other complicated areas, including massive social networks or live analytics systems.

## References

Shaikhha, A., Kelepeshis, M., & Ghorbani, M. (2023, February). Fine-tuning data structures for query processing. In *Proceedings of the 21st ACM/IEEE International Symposium on Code Generation and Optimization* (pp. 149-161). https://doi.org/10.1145/3579990.3580016

Katsogiannis-Meimarakis, G., & Koutrika, G. (2023). A survey on deep learning approaches for text-to-SQL. *The VLDB Journal*, *32*(4), 905-936. https://doi.org/10.1007/s00778-022-00776-8

Li, I., Pan, J., Goldwasser, J., Verma, N., Wong, W. P., Nuzumlalı, M. Y., ... & Radev, D. (2022). Neural natural language processing for unstructured data in electronic health records: a review. *Computer Science Review*, *46*, 100511. https://doi.org/10.1016/j.cosrev.2022.100511

Mosqueira-Rey, E., Hernández-Pereira, E., Alonso-Ríos, D., Bobes-Bascarán, J., & Fernández-Leal, Á. (2023). Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, *56*(4), 3005-3054. https://doi.org/10.1007/s10462-022-10246-w

Li, B., Jiang, G., Li, N., & Song, C. (2024, August). Research on large-scale structured and unstructured data processing based on large language model. In *Proceedings of the International Conference on Machine Learning, Pattern Recognition and Automation Engineering* (pp. 111-116). https://doi.org/10.1145/3696687.3696707

Kumar, A., Chilluri, V. S. B., Sharma, C., & Singh, A. (2025, February). A Comparative Study of Data Structure for Efficient Query Processing. In *2025 First International Conference on Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT)* (pp. 1111-1115). IEEE. https://doi.org/10.1109/CE2CT64011.2025.10941242

Wang, J., Yi, X., Guo, R., Jin, H., Xu, P., Li, S., ... & Xie, C. (2021, June). Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 international conference on management of data* (pp. 2614-2627). https://doi.org/10.1145/3448016.3457550

Alanazi, R. (2022). Identification and prediction of chronic diseases using machine learning approach. *Journal of healthcare engineering*, *2022*(1), 2826127. https://doi.org/10.1155/2022/2826127

Dang, H. V., Tatipamula, M., & Nguyen, H. X. (2021). Cloud-based digital twinning for structural health monitoring using deep learning. *IEEE transactions on industrial informatics*, *18*(6), 3820-3830. https://doi.org/10.1109/TII.2021.3115119

Bhatti, U. A., Tang, H., Wu, G., Marjan, S., & Hussain, A. (2023). Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence. *International Journal of Intelligent Systems*, *2023*(1), 8342104. https://doi.org/10.1155/2023/8342104

Abbasiantaeb, Z., & Momtazi, S. (2021). Text-based question answering from information retrieval and deep neural network perspectives: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *11*(6), e1412. https://doi.org/10.1002/widm.1412

Lavanya, P. M., & Sasikala, E. (2021, May). Deep learning techniques on text classification using Natural language processing (NLP) in social healthcare network: A comprehensive survey. In *2021 3rd international conference on signal processing and communication (ICPSC)* (pp. 603-609). IEEE. https://doi.org/10.1109/ICSPC51351.2021.9451752

Nassiri, K., & Akhloufi, M. (2023). Transformer models used for text-based question answering systems. *Applied Intelligence*, *53*(9), 10602-10635. https://doi.org/10.1007/s10489-022-04052-8

Chiche, A., & Yitagesu, B. (2022). Part of speech tagging: a systematic review of deep learning and machine learning approaches. *Journal of Big Data*, *9*(1), 10. https://doi.org/10.1186/s40537-022-00561-y

Zhu, A., Wang, Z., Li, Y., Wan, X., Jin, J., Wang, T., ... & Hua, G. (2021, October). Dssl: Deep surroundings-person separation learning for text-based person retrieval. In *Proceedings of the 29th ACM international conference on multimedia* (pp. 209-217). https://doi.org/10.1145/3474085.3475369

Dogra, V., Verma, S., Kavita, Chatterjee, P., Shafi, J., Choi, J., & Ijaz, M. F. (2022). A complete process of text classification system using state-of-the-art NLP models. *Computational Intelligence and Neuroscience*, *2022*(1), 1883698. https://doi.org/10.1155/2022/1883698

Samant, R. M., Bachute, M. R., Gite, S., & Kotecha, K. (2022). Framework for deep learning-based language models using multi-task learning in natural language understanding: A systematic literature review and future directions. *IEEE Access*, *10*, 17078-17097. https://doi.org/10.1109/ACCESS.2022.3149798

Choi, S., Lee, H., Park, E., & Choi, S. (2022). Deep learning for patent landscaping using transformer and graph embedding. *Technological Forecasting and Social Change*, *175*, 121413. https://doi.org/10.1016/j.techfore.2021.121413

Liang, Y., Gao, E., Ma, Y., Zhan, Q., Sun, D., & Gu, X. (2024, August). Contextual analysis using deep learning for sensitive information detection. In *2024 International Conference on Computers, Information Processing and Advanced Education (CIPAE)* (pp. 633-637). IEEE. https://doi.org/10.1109/CIPAE64326.2024.00121

Bernius, J. P., Krusche, S., & Bruegge, B. (2022). Machine learning based feedback on textual student answers in large courses. *Computers and Education: Artificial Intelligence*, *3*, 100081. https://doi.org/10.1016/j.caeai.2022.100081

Qi, Y., & Shabrina, Z. (2023). Sentiment analysis using Twitter data: a comparative application of lexicon-and machine-learning-based approach. *Social network analysis and mining*, *13*(1), 31. https://doi.org/10.1007/s13278-023-01030-x