Enhancing Query Optimization in DBMS ThroughIntegration of Machine Learning: A Survey

ABSTRACT

Optimizing queries within databases plays a pivotal role in enhancing the efficiency and performance of database management systems (DBMS). Traditional query optimization methods often rely on rule-based strategies and heuristics, which may not fully account for the intricate interplay among query attributes, data distribution, and system resources. Managing large volumes of data poses significant challenges when striving for high-performance storage and retrieval within database systems. The query optimizer bears the responsibility for achieving such high performance. However, classical optimizers encounter challenges, including inaccuracies in cardinality estimation, errors in cost modeling, and the vast search space, which hinder the generation of optimal execution plans for complex queries. In recent times, as scientific advancements across various domains burgeon, machine learning (ML) is profoundly reshaping the landscape of data management. Integration of machine learning techniques with database management systems holds promise for augmenting query optimizers and, consequently, optimizing queries. Consequently, numerous ML techniques have been proposed or employed to enhance query performance, encompassing cardinality estimation, cost estimation, and query plan enumeration. This survey paper critically reviews existing works that leverage machine learning methodologies to address the query optimization challenge, while also delineating future directions for each approach.

KEYWORDS: Machine learning; Database management system; Query Optimization; Cardinality

estimation; Cost model; Plan enumeration.

INTRODUCTION

These days, AI research has a significant impact on both academic and industry domains, with machine learning (ML) emerging as a rapidly growing branch. ML is the process of compiling a dataset and algorithmically creating a statistical model to solve real-world problems, categorized into supervised, semi-supervised, unsupervised, and reinforcement learning [2]. In the data science field, ML plays a crucial role in classifying trends without direct programming, starting from data analysis to expected conclusions [3]. Database management systems (DBMS) are used for storing and retrieving data effectively across various applications. So, the integration of DBMS and ML has become a major trend in managing data, analytics, and decisionmaking processes. This integration has the potential to transform conventional patterns in database systems and improve data processing and extraction capabilities [1]. The research on Machine Learning Integration in Database Management Systems delves into the complex terrain of incorporating ML technology within DBMS. This ML integration is used to handle difficult problems such as in query optimization, which improves query execution times, resource consumption, and overall system efficiency and workload by applying ML algorithms to query optimization. Query optimization faces a lot of challenges in working with classical optimizers that are very complex when avoiding selecting inefficient plans, they might automatically slow down query run time. This complexity causes many issues and to reduce the complexity of the optimization problem, optimizers use a wide range of heuristics. Although, the heuristics sacrifice performance. Also, depending on that, developing the optimizer is very costly. Machine learning is an innovative technique to handle these issues by replacing algorithms that are manually created with patterns that are automatically discovered from data and using the reinforcement technique that can learn doing tasks by experimentation [4]. A lot of researchers focus on improving the efficiency of the optimizer.

The purpose of the research outlined in the provided sources is to explore the integration of machine learning techniques in query optimization to address challenges related to accuracy problems in cardinality estimation and cost modeling. The research aims to leverage machine learning approaches to enhance the efficiency of query execution plans by improving cardinality estimation, cost estimation, and plan enumeration within query optimization frameworks. By utilizing machine learning algorithms such as auto-regressive unsupervised learning, deep neural networks, and deep reinforcement learning, the research seeks to overcome limitations in traditional query optimizers and enhance the accuracy, adaptability, and performance of query optimization processes. Additionally, the research aims to investigate future directions in cardinality estimation, cost modeling, and plan enumeration to further improve the effectiveness, scalability, and applicability of machine learning techniques in optimizing database query performance.

This survey paper aims to provide a comprehensive overview of the current state of research in using ML techniques to enhance the query optimizer by improving the ability of its components.

The paper presents the following contributions:

1. Review the studies that use machine learning to estimate cardinality more accurately.
2. Review the studies that work on enhancing the cost model.
3. Review machine learning methods used in plan enumeration.

This paper is organized to be as the following sections: Section 2 presents the background, Section 3 covers the optimizing ml integration in DBMS, Section 4 discusses open research challenges, Section 5 presents discussion, and Section 6 shows the conclusions.

BACKGROUND

In this section, we give an overview of the integration of machine learning into a database management system and some background information on machine learning techniques, database management systems, and query optimizers to better understand the rest of the paper.

GENERAL OVERVIEW OF MACHINE LEARNING, DATABASES, AND

RELEVANT HISTORY

Traditional database administration heavily relies on human effort for configuring and maintaining databases, often following experimental methods and specifications. Nevertheless, conventional methods are constrained by inherent limitations, which can be alleviated by incorporating machine learning (ML) techniques into database management systems (DBMS). ML offers advanced features like picture identification, anomaly detection, and natural language processing, transforming traditional DBMS approaches [5]. In early 2000, The integration of AI and ML into DBMS was a significant trend and allowed sophisticated insights and analytics. So, there is a lengthy history of ML integration with DMBSs. The author Wu [6] determined the integration of machine learning (ML) with database management systems (DBMS) brings up a world of opportunities for efficiency and performance optimization. A closer look at this integration process indicates several strategies and procedures that might be employed to achieve these goals. To fully utilize ML within the DBMS, it must explore these important issues, which range from minimizing data storage to expediting query execution, and from optimizing algorithms to utilizing parallel processing capabilities. difficulties that lie ahead in integrating database technology and machine learning to develop Intelligent Learning Database (ILDB) solutions. Research distinguished two primary areas: machine learning approaches that can handle enterprise-scale database systems and effective data representation. Also, added that real-world databases are frequently noisy and contain inaccurate or missing data, which is expected to present difficulties for learning systems.

MACHINE LEARNING TECHNIQUES (MLT)

The machine learning methodology learns from data to design an algorithmic solution. The aim is to program computers to learn from past experiments or data to solve the problem that is given. For more explanation the machine learning models into two parts as follows: techniques which has classical methods for machine learning and deep learning [1][2].

Classical machine learning (CML)

Machine learning tasks can be organized depending on the output that is obtained during the learning into four main categories known as the type of learning.

1. **Supervised learning,** the dataset in supervised learning is made up of all the labeled samples. A feature vector is any one of the elements x in a set N. A feature vector has a value describing each in it. The goal it is to create a model using the dataset that accepts a feature vector x as input and returns data that makes it possible to determine the label for this feature vector [2].

2. **Unsupervised learning,** on the other hand, uses a dataset made up of unlabeled instances. X is a feature vector again at this stage. Developing a model that accepts a feature vector x as input and either converts it into another vector or a value that can be utilized to address a real-world issue is the aim of an unsupervised learning algorithm [2].
3. **Semi-supervised learning,** both labeled and unlabeled samples are present in the dataset. In most cases, there are a lot more unlabeled samples than labeled examples. The aim of a supervised learning algorithm and a semi-supervised learning algorithm are the same. It is hoped that by employing many unlabeled instances, the learning algorithm will be able to find a better model [2].
4. **Reinforcement learning,** which is another branch of machine learning, allows a computer to "live" in an environment and interpret its current state as a vector of features. The equipment is capable of operating in all states. Different behaviors result in different rewards and can change the machine's environment. Learning a policy is the aim of a reinforcement learning algorithm. A policy is a function f that, like the model in supervised learning, accepts a state's feature vector as input and produces the best course of action to take while in that state. If an activity maximizes the expected average benefit, then it is considered optimal [2].

Deep learning (DL)

Deep learning (dl) is a machine learning model that refers to learning neural networks with multiple non-output layers. It recognized the complex pattern by working on learning the computer to process data from multiple layers as well as human brine [2]. They use different kinds of layers like fully connected, and ReLU to learn a hierarchy of parametric characteristics. Back-propagation serves as the methodology employed for training all parameters. Recently, deep learning methodologies have demonstrated superior performance over conventional machine learning techniques, particularly in achieving heightened accuracy across demanding tasks like natural language processing and computer vision [1].

DATABASE MANAGEMENT SYSTEM (DBMS)

DBMS is a software tool used in building multiple different types of databases. Each one stores a specific group of data and is used for one purpose. It has evolved into a common tool for insulating computer users from secondary storage management specifics. It offers many benefits over file system management since it allows for most of the issues with data anomalies, data inconsistencies, data dependence, and structural dependence to be resolved. Even better, the DBMS software of the modern era keeps all the data structures, relationships between them, and access paths to them in one central area. All necessary access pathways to such components are defined, stored, and managed by the DBMS software of the present generation. There are many types of DBMSs such as RDBMS, Centralized databases, Distributed databases, Cloud databases, Object-oriented databases, and NoSQL databases [7].

OPTIMIZING THE QUERY

Query optimizations are the process that is used to select an efficient way of running a SQL command. SQL is a nonprocedural language; therefore, the optimizer can combine, rearrange, and process in any sequence. The purpose is to turn an initial relational algebra statement into the most efficient query evaluation plan. To get to this point the conventional query optimizers go through potential query evaluation plans to find the one with the lowest overall cost [8]. Lately, there's been much research focused on integrating machine learning techniques into query optimization efforts. In figure (1) can see the query process and see the optimizer translation work.

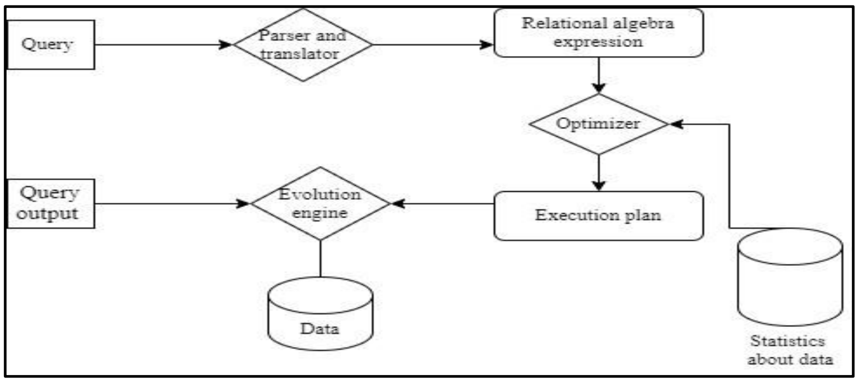

Figure 1 Query process steps [9]

THE INTEGRATION OF MACHINE LEARNING IN QUERY OPTIMIZER

The query optimizer is integrated database software that chooses the fastest way for a SQL query to get the needed data which contains three main components as follows: Cardinality estimation, Cost model, and Plan space enumeration shown in figure (2) [10][11].

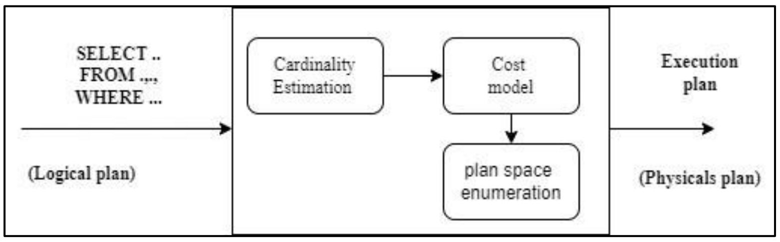

Figure 2 The Optimizer components architecture [12]

Figure (3) shows the classification of query optimizer in this paper.

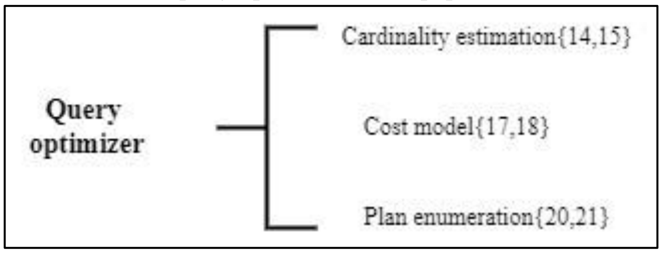

Figure 3 Query optimizer classification.

CARDINALITY ESTIMATION

Cardinality estimation stands as a crucial task within query processing and optimization in the realm of database management systems. It's a statistical model that can estimate row numbers that are created by the operators and use it in the cost model to calculate the final cost of operators. The cardinality estimation is the basis of optimizer work. The cost model and join enumeration will be nothing while the cardinality estimation is not correct, as Leis et al [12] concluded the errors of cardinality estimation in their experiment. They are focusing on the physical quality of the cardinality estimation through multi-join queries and summarize the causes of errors. The cases can be in a base(single) table with prediction, multi-join query errors, and userdefined functions. They agree with the Lohman [13] results, that is the cardinality estimation introduced the error on multiple magnitude order. There are three main approaches to estimating cardinality as follows: Synopsis-based methods, Sampling-based methods, and learning-based methods [10][11]. Recent papers show that machine learning (ML)--based algorithms can yield more accurate cardinality estimates than older methods. There are delineated distinct approaches for training machine learning models used in cardinality estimation as follows:

Using the deeply unsupervised Cardinality Estimation [14], For years, cardinality estimation has relied on statistical tools for density estimation. To capture the complex multivariate distributions found in relational tables, the suggestion is to utilize a novel high-capacity statistical model: deep autoregressive models. When range or wildcard predicates are used, using these models directly results in a constrained estimator that is excessively expensive to evaluate. To generate a highly functional estimator, the Monte Carlo integration approach is designed for autoregressive models that can handle query range with multi-dimensional effectively. Similar to typical synopses, the estimator summarizes data without supervision. The joint data distribution is approximated by an estimator without generating any independent assumptions. When tested on real-world datasets and compared to actual systems and leading methodologies, it achieves single-digit multiplicative error rates at the tail, resulting in up to a 90x accuracy improvement over the next best method. Furthermore, it is space- and time-efficient.

NaruCard, [15] is based on a straightforward idea: understand the correlations that exist across all database tables without depending on any assumptions of independence. It uses existing techniques from join sampling and deep self-supervised learning to address cardinality estimation, which is an important part of query optimization. Unlike typical data-driven estimators, learns from data and captures all possible inter-table correlations within a probabilistic model indicated as $p\theta$. To the best of our understanding, this is the initial cardinality estimator to accomplish probabilistic modeling devoid of assumptions across over a dozen tables. Its performance is exceptional, with state-of-the-art accuracy for join cardinality estimation 4 to 34 times better than earlier techniques. Furthermore, it accomplishes this using a single per-schema model that is both compact and efficient in learning.

COST MODEL

The main function of the cost model is the ability to expect from alternative plans which one is the fastest query plane depending on the cardinality estimation. It works as a plane selection from search [12]. It uses cost-based estimation to produce the sup-query cost. The plan cost is a summation of all operator's cost work on it [11]. The cost model depends on the implementation of the operators, data volume, and the number of process executions in each operator. The accuracy of the cost model shows in the ability to distinguish between sequential and random I/O operations [16]. Notice that the error of cardinality estimation affects cost model accuracy. One of the proposed solutions for the cost model is to build a new one with a supervised machine-learning approach as follows:

Ji et al [17] proposed a method to build an estimator by presenting an effective learning-based, end-to-end cost estimation technique built on a tree-structured model. The goal of this methodology is to accurately estimate cost and cardinality at the same time. They advocate for proficient feature extraction and encoding algorithms that account for both queries and physical activities throughout the extraction process. Subsequently, these extracted features are integrated with our tree-structured model. Furthermore, it presents an effective method for encoding string values, which improves the model's generalization capacity for predicate matching. Due to the impracticality of exhaustively enumerating all string values, we have devised a pattern-based approach to identify patterns containing string values. Subsequently, these patterns are utilized to effectively embed the string values. Extensive findings using real datasets show that the technique outperforms.

DeepCm framework [18], was presented and developed for building a strong database cost model based on min-max optimization. This framework is built on deep neural networks to guarantee higher performance whatever the different configurations of software and workload quality. It faces several experiments to evaluate the robustness in generating the cost model and the result demonstrates that this framework achieves high accuracy in cost model estimation and stability in performance.

PLAN ENUMERATION

In addition to cardinality estimation and cost modeling, query optimization includes a plan enumeration technique that identifies semantically ideal join orders. It Minimizes query costs by selecting semantically similar join orders [11]. The traditional method that based on cardinality estimation and cost model search for solutions for the possible join orders. In the traditional database, there are three numeration methods: bottom-up joining by dynamic programming, top-dawn joins by memorization and randomized method. The restriction of plan enumeration can be classified as cardinality estimation and cost model Errors, working with queries that contain huge amounts of data and multiple tables, and the algorithms used in search space cutting. The cardinality estimation error effect in the cost model as a result leads to a suboptimal plan [16][19]. As a solution for these limitations, the researcher using different reinforcement learning techniques proposes to enhance the join order selection performance by learning from the last query.

Machine learning can improve selection plans efficiently. For this task, FOOP [20] was proposed using deep reinforcement learning based on a fully observed optimizer, which is the general framework for query optimization that allows multi-machine learning methods to plug in. The concept behind this approach is to use data-adaptive learning query optimization, which is quicker than standard dynamic programming, to eliminate the laborious enumeration of join orders. As the findings of these experiments show, the proposed method is outperformed by a deep Q-network in join order optimization. In addition, they suggest combining deep reinforcement learning with ensemble learning to reduce the issues of stability in the training process because the results show the improvement of reinforcement learning performance while using it with ensemble learning.

Additionally, Ji et al. [21] worked with a dynamic double Q network selection technique for join optimization.

The technique using the Markov decision process as a join of query and integrating the network DQN with DDQN weighting to solve the estimation error problem of the DRL model in query joining will enhance the developing query plans. To improve the randomness and depth of exploration and accumulate a high information gain of exploration, using a dynamic progressive search strategy in selected actions. by comparing the proposed method with a dynamic progressive search strategy, it gets high accuracy in join order and learns the effect strategy in the query plan. The result of these methods shows that the performance of the query outperforms others in multi-join queries.

OPEN RESEARCH CHALLENGES FOR USING MACHINE LEARNING IN QUERY OPTIMIZATION

Addressing the accuracy problems in the cost model and cardinality estimation by using machine learning in the query optimizer to choose the best plan in the plan enumeration. A lot of the papers proposed this integration with a different ML approach to close the gap in classical optimizers.

This section discusses open research challenges in utilizing machine learning for query optimization, focusing on cardinality estimation, cost models, and plan enumeration. By addressing these challenges, there is a potential to significantly improve query optimization performance and tackle prevailing accuracy concerns in traditional optimizers.

CARDINALITY ESTIMATION

Cardinality estimation is a crucial component of query optimization, as it directly impacts the efficiency of query execution plans. Recent studies have explored the integration of machine learning techniques to enhance cardinality estimation, particularly through the use of auto-regressive unsupervised learning. This approach has shown promise in capturing the joint data distribution, leading to more accurate estimations of cardinality. However, challenges persist, including the potential complexity and computational cost associated with modeling the full joint distribution, especially for high-dimensional datasets.

Two notable studies employing machine learning techniques in cardinality estimation have been surveyed: Naru [14] and NeuroCard [15]. Both propose data-driven unsupervised learning strategies specifically, autoregressive models for enhanced cardinality estimation, adapting to shifting workloads through retraining. However, they differ in their approaches to overcoming limitations inherent in assuming attribute independence.

In [14], Naru, the provided estimator is trained without supposing column independence to capture the entire joint data distribution. It can give more precise estimations of cardinality by taking into consideration all potential interactions between characteristics by approximating the combined data distribution in its entirety. This approach allows for a more comprehensive understanding of the data and can lead to improved accuracy in estimating selectivity for various types of queries. The gap described in the paper is the potential complexity and computational cost associated with modeling the full joint. Estimating the joint distribution in its entirety can be computationally intensive, especially for high-dimensional datasets with many attributes.

On the other hand, NeuroCard [15] leverages deep autoregressive models along with neural density estimators, boasting exceptional handling of cross-table correlated data without imposing restrictive independence constraints. Nevertheless, several limitations hinder NeuroCard's practical utility: dependency on training data accuracy, intricate model opacity hampering interpretability, potential scalability difficulties in dealing with massive databases, the substantial computational expense involved in deep neural network training, and the necessity for dependable generalization during adaptation to novel workloads. Addressing these shortcomings holds paramount importance for realizing NeuroCard's maximum efficacy and practicability in genuine query optimization situations.

The NeuroCard offers groundbreaking advancements in cardinality estimation, but it is important to acknowledge its limitations. These include the dependencies on the accuracy of the training data, the complicated models' black-box nature that hinders interpretability, potential scalability issues with very large databases, the significant computational cost of deep neural network training, and the requirement for reliable generalization to new workloads are a few of these. Addressing these limitations will be crucial for maximizing the practical applicability and effectiveness of NeuroCard in real-world query optimization scenarios.

To address these limitations, future research should focus on:

1. The Integration of Reinforcement Learning: Adjusting cardinality estimators dynamically based on feedback from query execution results could improve adaptability to changing workloads and query patterns.
2. Employing Graph Neural Networks (GNNs): Modeling complex relationships and dependencies in database schemas and queries using graph neural networks could lead to more accurate cardinality estimates, especially for graph-structured data.

3. Tight Integration with Query Optimization Frameworks: Closer integration of cardinality estimation techniques with query optimization frameworks could enable better query plan selection and performance tuning based on accurate cardinality estimates.

The future direction in cardinality estimation should be focused on improving the limitations of the previous method that was used. When handling the limitation on another side the cardinality estimation will be more accurate.

COST MODEL

The cost model is another critical aspect of query optimization, where machine learning has been applied to improve estimation accuracy. Deep neural networks have been proposed as a solution, capturing global cost information and overcoming challenges like information loss and space explosion. However, limitations include the need for large amounts of high-quality training data, the "black box" nature of deep neural networks, and the complexity of hyperparameter tuning. Comparative examination unfolds in the realm of cost models, contrasting contributions from [17] and [18]:

As in [17], the proposed end-to-end learning-based cost estimation framework addresses the limitations of traditional methods by introducing a tree-structured deep neural network solution. This framework captures global cost information from leaf nodes to the root, ensuring comprehensive estimation and overcoming challenges like information loss and space explosion. Key features include effective feature extraction techniques considering physical query operations, query predicates, metadata, and data, encoded into vectors to represent different aspects of the query plan. Additionally, a pattern-based method for encoding string values enhances the generalization ability for predicate matching by generating rules to generalize keywords in the query workload, enabling efficient storage and retrieval of keyword vectors for improved accuracy and efficiency in cost estimation for query optimization tasks. One limitation of this paper is its narrow focus on cost and cardinality estimation within query optimization, potentially overlooking broader aspects of database performance optimization. The proposed framework may face challenges in handling highly complex query structures or scenarios with diverse data distributions, impacting the model's generalization ability. Additionally, the effectiveness of the pattern-based method for encoding string values could be limited by the complexity and variability of string patterns in real-world datasets, potentially affecting the accuracy of predicate matching.

On the other hand [18], The development of accurate database cost models faces challenges due to the complexity of modern systems, the need for specific Quality of Service levels, and evolving hardware platforms. Traditional approaches like analytic models (AMs) require explicit system knowledge but are difficult to build and maintain, while Machine-Learning (ML) models rely on training data but may lack generalization.

Deep neural networks, as seen in DeepCM, offer a promising solution by capturing complex interactions for improved accuracy in cost model predictions. DeepCM's application extends to enhancing query performance, evaluating resource impact, testing model robustness, assessing computational overhead, ensuring portability, and comparing performance with analytical models. Its deep learning approach automates parameter analysis, extraction, prediction, and evaluation, potentially offering a more efficient and accurate solution for database management tasks.

One limitation of DeepCM and deep neural network-based approaches, in general, is the need for a large amount of high-quality training data to effectively train the models. Gathering and preparing such datasets can be time-consuming and resource-intensive. Additionally, deep neural networks are often considered "black box" models, meaning that the internal workings and decision-making processes of the model may not be easily interpretable or explainable, which can be a drawback in certain applications where transparency is crucial. Furthermore, the complexity of deep neural networks can lead to longer training times and higher computational requirements compared to simpler models, which may not be feasible in all practical scenarios. Lastly, the performance of deep neural networks can be sensitive to hyperparameter tuning, requiring expertise and careful optimization to achieve optimal results.

To further enhance the accuracy, efficiency, and applicability of cost models, future research could focus on:

1. Explainable Artificial Intelligence (XAI): Integrating XAI techniques into deep neural network models like DeepCM could improve model interpretability and transparency.
2. Automated Machine Learning (AutoML): Leveraging AutoML techniques to automate feature selection, hyperparameter tuning, and model optimization for cost models could streamline the model development process.
3. Scalability and Real-time Performance: Addressing scalability challenges to ensure cost models can handle large-scale datasets and complex systems efficiently, optimizing model performance and computational efficiency for real-time or near-real-time cost predictions.
4. Platform Expansion: Utilizing DeepCM in different database platforms, such as centralized, cloud, parallel, and distributed, to broaden its applicability.

PLAN ENUMERATION

Plan enumeration is a critical step in query optimization, where machine learning has been applied to find the optimal query plan. Deep reinforcement learning has been proposed as a solution, avoiding exhaustive join order enumeration, and increasing efficiency. However, limitations include the lack of thorough assessment of generated query plan quality and the need for further exploration of generalization and robustness. Researchers working on [20] and [21] explore the frontier of incorporating deep reinforcement learning (DRL) for identifying optimal query plans.

Specifically, [20] circumvents exhaustive join order enumerations, thereby diminishing the necessity for laborious search routines and increasing efficiency. Demonstrating Proximal Policy Optimization (PPO) supremacy over alternative Q-learning approaches, FOOP underscores DRL prominence in query optimization. Regrettably, insufficient coverage exists examining FOOP's output quality, limiting appraisals of practical applicability. Questions linger about the method's capacity to scale and retain stability across assorted query workloads. It focused on developing a data-adaptive learning query optimizer that shows off the promise of DRL in query optimization by reducing the need for laborious join order enumerations and increasing efficiency. The paper lacks a thorough assessment of the quality of the generated query plans, which is essential for determining the DRL-based approach's practical applicability. Additionally, the generalization and robustness of the DRL-based approach to handle a variety of query workloads are not thoroughly covered, potentially leaving gaps in understanding the method's scalability.

On the other hand [21] proposed a dynamic double DQN (DDQN) order selection method for join order optimization, modeling join queries as a Markov decision process (MDP) and integrating DQN and DDQN weighting into the DRL model to improve query plan development. The method aims to address the overestimation of action values in query optimization by using DDQN, enhancing the quality of query plans. It employs a dynamic progressive search strategy for action selection to improve exploration and information gain in query optimization. Focuses on enhancing query performance by using a dynamic DRL strategy that performs better than existing DRL optimization techniques based on the Join Order Benchmark (JOB), dynamic programming, and heuristic algorithms. Specific limitations related to the performance, scalability, or robustness of the Dynamic Double Deep Q-Network method in handling complex join queries and diverse data environments are not explicitly discussed in the available information.

To address these limitations, future research could focus on:

1. Expert knowledge derived through bootstrapping: Integrating bootstrapped expert knowledge into the pretraining phase of certain models to mitigate computational intensity and facilitate faster learning.
2. Generalization Capabilities: Investigating the generalization capabilities of the DRL-based approach across various query workloads and data scenarios to ensure robustness and scalability in real-world applications.

DISCUSSION

Integrating machine learning techniques into query optimization to address accuracy challenges in cost modeling, cardinality estimation, and plan enumeration presents a promising avenue for enhancing database query performance. By leveraging advanced machine learning algorithms such as auto-regressive unsupervised learning, deep neural networks, and deep reinforcement learning, researchers aim to improve the efficiency, accuracy, and adaptability of query execution plans. This paper highlights the potential of machine learning approaches like NeuroCard and DeepCM in revolutionizing cardinality estimation and cost modeling within query optimization frameworks. While these approaches offer significant advancements in accuracy and scalability, they also pose challenges such as the need for high-quality training data, interpretability issues due to the "black box" nature of deep neural networks, and scalability concerns with large datasets. Future research directions emphasize the integration of reinforcement learning for dynamic cardinality estimation adjustment, the exploration of graph neural networks for modeling complex relationships in database schemas, and the closer integration of cardinality estimation techniques with query optimization frameworks. Additionally, the research suggests incorporating explainable artificial intelligence (XAI) techniques, leveraging Automated Machine Learning (AutoML), addressing scalability challenges, and expanding the application of DeepCM to different database platforms to enhance the accuracy, efficiency, and applicability of cost models.

This paper underscores the importance of further advancements in machine learning-driven query optimization to overcome existing limitations, improve model interpretability, streamline model development processes, ensure scalability, and enhance the practical applicability of these techniques in real-world database management scenarios. By addressing these challenges and exploring future research directions, the field of machine learning in query optimization is poised to make significant strides in optimizing database performance and query execution efficiency.

CONCLUSION

Utilizing machine learning (ML) in cardinality estimation, cost modeling, and plan enumeration presents significant opportunities for enhancing the efficiency and accuracy of query optimization in database systems. ML techniques offer the potential to adaptively learn from data patterns and query characteristics, leading to improved cardinality estimates, more precise cost models, and optimized query plans. By leveraging ML algorithms, database systems can better handle complex queries, varying workloads, and evolving data distributions. However, challenges such as model interpretability, training data quality, and computational complexity must be carefully addressed to realize the full benefits of ML integration. Overall, the integration of ML into cardinality estimation, cost modeling, and plan enumeration represents a promising avenue for advancing the capabilities of database query optimization systems in handling modern dataintensive applications.

Future research should focus on reinforcement learning integration, graph neural networks, and closer alignment with query optimization frameworks. Enhancing cost models through XAI, AutoML, scalability improvements, and platform expansion is crucial. In plan enumeration, DRL methods show promise, but thorough assessment and generalization are needed for practical applicability. Future directions include integrating expert knowledge and investigating the generalization capabilities of DRL approaches across various scenarios.

REFERENCES

[1] S. Nannapaneni, "Machine Learning for Database Management Systems," International Journal of Engineering and Computer Science, vol. 9, pp. 25132-25147, 2020. DOI: 10.18535/ijecs/v9i08.4520.

[2] A. Burkov, "The Hundred-Page Machine Learning Book," 159, 2019.

[3] X. Zhou, C. Chai, G. Li, and J. Sun, "Database meets artificial intelligence: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 3, pp. 1096-1116, 2020.

[4] Z. Yang, "Machine Learning for Query Optimization," University of California, Berkeley, 2022.

[5] C. Chai, J. Wang, Y. Luo, Z. Niu, and G. Li, "Data management for machine learning: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 5, pp. 4646-4667, 2022.

[6] X. Wu, "Building intelligent learning database systems," AI magazine, vol. 21, no. 3, pp. 6161, 2000.

[7] C. Coronel and S. Morris, "Database systems: design, implementation and management," Cengage learning, 2019.

[8] S. J. Kamatkar, A. Kamble, A. Viloria, L. Hernández-Fernández, and E. G. Cali, "Database performance tuning and query optimization," in Data Mining and Big Data: Third International Conference, DMBD 2018, Shanghai, China, June 17–22, 2018, Proceedings 3, pp. 3-11, Springer International Publishing, 2018.

[9] G. R. Bamnote and S. S. Agrawal, "Introduction to query processing and optimization," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 7, pp. 53-56, 2013.

[10] V. Leis, B. Radke, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann, "Query optimization through the looking glass, and what we found running the join order benchmark," The VLDB Journal, vol. 27, pp. 643-668, 2018.

[11] H. Lan, Z. Bao, and Y. Peng, "A survey on advancing the dbms query optimizer: Cardinality estimation, cost model, and plan enumeration," Data Science and Engineering, vol. 6, pp. 86101, 2021.

[12] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann, "How good are query optimizers, really?," Proceedings of the VLDB Endowment, vol. 9, no. 3, pp. 204-215, 2015.

[13] G. Lohman, "Is query optimization a 'solved' problem," in Proc. Workshop on Database Query Optimization, vol. 13, p. 10, Oregon Graduate Center Comp. Sci. Tech. Rep., April 2014.

[14] Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, X. Chen, et al., "Deep unsupervised cardinality estimation," arXiv preprint arXiv:1905.04278, 2019.

[15] Z. Yang, A. Kamsetty, S. Luan, E. Liang, Y. Duan, X. Chen, et al., "NeuroCard: one cardinality estimator for all tables," arXiv preprint arXiv:2006.08109, 2020.

[16] S. Manegold, P. Boncz, and M. L. Kersten, "Generic database cost models for hierarchical memory systems," in VLDB'02: Proceedings of the 28th International Conference on Very Large Databases, pp. 191-202, Morgan Kaufmann, January 2002.

[17] J. Sun and G. Li, "An end-to-end learning-based cost estimator," arXiv preprint arXiv:1906.02560, 2019.

[18] A. Ouared, A. Chadli, and M. A. Daoud, "Deepcm: Deep neural networks to improve accuracy prediction of database cost models," Concurrency and Computation: Practice and Experience, vol. 34, no. 10, p. e6724, 2022.

[19] X. Zhou, C. Chai, G. Li, and J. Sun, "Database meets artificial intelligence: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 3, pp. 1096-1116, 2020.

[20] J. Heitz and K. Stockinger, "Join query optimization with deep reinforcement learning algorithms," arXiv preprint arXiv:1911.11689, 2019.

[21] L. Ji, R. Zhao, Y. Dang, J. Liu, and H. Zhang, "Query Join Order Optimization Method Based on Dynamic Double Deep Q-Network," Electronics, vol. 12, no. 6, p. 1504, 2023.