IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 14 August 2023, accepted 2 September 2023, date of publication 5 September 2023,
date of current version 13 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3312312

RESEARCH ARTICLE

ConvMADE: Convolution Makes Cardinality Estimation Stronger

CHAO GAO, JIONG YU, ZHENZHEN HE, AND XIAOQIAO XIONG
School of Information Science and Engineering, Xinjiang University, Ürümqi 830017, China

Corresponding author: Chao Gao (renfei@stu.xju.edu.cn)

This work was supported in part by the National Natural Science Foundation of China Project under Grant 62262064, Grant 62266043, and Grant 61966035; in part by the Key Research and Development Projects in Xinjiang Uygur Autonomous Region under Grant XJEDU2016S106; and in part by the Natural Science Foundation of Xinjiang Uygur Autonomous Region of China under Grant 2022D01C56.

ABSTRACT Cardinality estimation is critical for optimizing database queries, and accurate results are essential for a good query plan. Traditional models use statistical principles but struggle with complex data associations. Learning-based methods solve these problems but need to improve accuracy and reduce parameter size, and adapt to multi-table training. Therefore, we propose the **Conv**olutional **M**asked **A**utoencoder for **D**istribution **E**stimation(ConvMADE) model, which uses the **Re**-**p**arameterization **Conv**olution(RepConv) structure, which enhances the ability of the model to obtain data features, thereby improving the accuracy of cardinality estimation. At the same time, the DepthWise Multilayer Perceptron (DWMP) structure is added to reduce the number of model parameters, and each table is explicitly trained to improve the ability to capture multi-table data features. We compare the ConvMADE model with traditional and learning-based methods on the DMV and IMDB datasets. The results show that the performance of the ConvMADE model in both single-table and multi-table models is superior to other models, and the parameter amount of the ConvMADE model is much lower than that of the baseline model. The single table can be as low as 18% of the baseline model, the multi-table can be as low as 81%, and the multi-table average q-error is 27.2% lower than the baseline model.

INDEX TERMS Autoregression model, cardinality estimation, convolution, database management system, query optimizer.

I. INTRODUCTION

A query optimizer forms an integral component of any relational database management system (RDBMS) [1], [2], [3], [28]. Its primary task is to identify the optimal execution plan for SQL queries. Typically, the query optimizer comprises three constituent parts: Cardinality Estimation(CE), Cost Model, and Plan Enumeration. Cardinality Estimation predicts the selectivity of query statements that are not executed; Cost Model computes query overheads based on the output of Cardinality Estimation, while Plan Enumeration finds the optimal join order. Both the cost model and plan enumeration depend on Cardinality Estimation; thus, the accuracy of the latter impacts query optimizer performance. With accurate cardinality estimation, the optimizer will output optimal query plans consistently. On the other hand, incorrect cardinality estimates render the query plan several orders of magnitude slower than optimal [30]. Therefore, we investigated cardinality estimation variations in-depth to foster query optimizer performance.

Cardinality estimation [22] can be divided into two methods: traditional CE [4], [5], [6], [7], [8], [9], [10] and learned-based CE [11], [12], [13], [14], [15], [16], [17], [23]. Traditional cardinality estimation relies on statistical principles and probability distribution assumptions, which cannot comprehensively consider the connections between data and are prone to errors in cardinality values. The relationships between data become more complex when the number of predicate columns increases, making it more difficult to obtain the best execution plan. Therefore, the traditional method is suitable for scenarios with small data scales and relatively stable data distributions. The learning-based

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

cardinality estimation uses deep learning methods that can capture the connection relationships between tables and columns and is suitable for scenarios with large data sizes and complex distributions. Various methods have been proposed in learning-based cardinality estimation, such as the method of directly obtaining query cardinality and the method of obtaining query selectivity, which outperform traditional cardinality estimation methods both on single-table and multi-tables. However, these models have a high number of parameters themselves, and the number of model parameters and training and inference time increases rapidly when continuing to improve the accuracy of cardinality estimation. Based on this drawback, we propose an autoregressive model-based cardinality estimation method, which can obtain more accurate estimation results and can also significantly reduce the number of model parameters.

Designing a good cardinality estimation model faces several challenges. Firstly, the model's estimation accuracy is often related to the parameter size. Although increasing the number of parameters can improve the accuracy, a linear increase in parameter quantity does not necessarily lead to significant improvements in accuracy. Secondly, existing models still have significant optimization potential regarding reducing parameter count without compromising estimation quality. Lastly, in the case of multi-table joins, existing models encounter considerable difficulty. The connections between different tables are challenging to capture, and the internal characteristics of each table often need to be more adequately defined.

To address the above challenges, we propose the ConvMADE model, which can effectively improve the accuracy of single-table and multi-table cardinality estimation and reduce the parameter scale. Furthermore, we experiment on real datasets and show that our method performs well. In this study, our main contributions are as follows:

1) We propose the RepConv structure optimization model, which successfully improves the cardinality estimation accuracy on single-table and multi-table datasets without increasing the number of parameters. At the same time, we also designed the ConvMADE-S model to improve the accuracy of cardinality estimation further while reducing the number of parameters.

2)We have designed a series of ConvMADE-num models that leverage the DWMP structure to reduce the number of model parameters significantly. This structure is particularly effective for multi-table datasets as it allows for better capturing of each table's internal characteristics, thereby substantially decreasing the average q-error.

3)We conducted experiments on two real datasets, DMV and IMDB, and the results demonstrate that our method outperforms traditional and other learning-based methods.

This paper is divided into five main parts. The first chapter introduces the research background and our research content. The second chapter provides a summary of the existing cardinality estimation methods. In the third chapter, we describe the mechanism and principle of our proposed cardinality estimation method. The fourth chapter elaborates on the experimental details of the dataset, evaluation index, and ConvMADE model. Finally, the fourth chapter summarizes the content of the full text and discusses future research directions.

II. RELATED WORK

A. TRADITIONAL CARDINALITY ESTIMATION

Traditional cardinality estimation: There are two types: synopsis-based and sampling, which are widely used in various databases. Synopsis-based approaches assume data independence and utilize statistics collected by the database to quickly calculate query cardinality. Among them, the histogram [5] summarizes each column in the data table into a histogram of equal width or equal depth and calculates the final estimation result based on the assumption that the columns are independent of each other. For data sketching such as loglog [9] and hyperloglog [10], use bitmap or hash method to estimate the number of different element elements, and pay attention to saving memory and improving the robustness of estimation. The sampling method [4], [6], [7], [8], [26] randomly selects a certain proportion or number of tuples from the original data table, calculates the size of the query result set, and calculates the cardinality estimation result of the query in the original database according to the corresponding scaling ratio.The traditional cardinality estimation methods have the advantage of being computationally efficient due to their reliance on simple calculations. By utilizing corresponding estimation methods to understand the data distribution, accurate cardinality estimation results can be obtained quickly. These traditional methods are also highly interpretable and allow for greater control over their behavior. As a result, they have been widely adopted by the industry [26], [36], [37], [38].Many databases that employ cost models utilize existing statistical information to estimate the size of a problem. They assign weights to different operations using cost models and ultimately derive the cardinality estimation result. However, these methods mainly use explicit attributes to obtain data characteristics, which are fixed singly, especially in the face of uneven data distribution, incremental data updates, and data heterogeneity. These methods also make it difficult to recognize the connections between data from a perspective that researchers cannot recognize. Therefore, although traditional methods are fast and have strong interpretability, they still have some limitations.

B. LEARNED CARDINALITY ESTIMATION

Learned cardinality estimation: Aims at learning the joint data distribution. According to different training objects, learning-based cardinality estimation can be divided into two categories: query-driven [13], [14], [15], [23], [25] and data-driven [11], [12], [16], [17].

Query-driven cardinality estimation [13], [14], [15], [23], [25] uses SQL query statements as training objects and the cardinality corresponding to query statements as labels,

which can directly learn the corresponding relationship between query and cardinality, so it is called supervised cardinality estimation. However, the training data for this method needs to be representative. Otherwise, it will not be able to learn the correspondence from common query patterns in the working environment to cardinality, thus affecting prediction accuracy. MultiSet Convolutional Network [13] (MSCN) expresses the query as a triplet form of query table, join condition and predicate selection. It uses the neural network to estimate the cardinality of the query result. In Ji Sun's work [25], the authors combined cardinality estimation and query plan cost estimation in the same end-to-end learned estimation model. The model relies on the optimizer of the database system to generate a physical plan for the input query, and the training data is composed of the query plan, exact cardinality, and exact plan cost triplet form. Since the query plan is a tree structure, its model uses a tree Long Short-Term Memory(LSTM). In short, each node in the plan tree receives information from its child nodes and integrates it with its information, and finally sends the output of the query root node to the two models of predicting base and predicting cost to get the desired estimate.

The data-driven cardinality estimation [11], [12], [16], [17], [18], [19] is to fit the joint data distribution through the deep learning model and obtain the probability through the generated model reasoning to finally generate the cardinality. The final selectivity can be obtained without labels. It is a kind of unsupervised method. It does not need to assume the form of the data, nor does it need to make constraints on the data; it can be trained directly on the data, and the cardinality estimation results of this type are also more stable than other methods. Among all learning-based cardinality estimates, Naru [12] has the best performance on a single table, and the Neuro [11] model is a multi-table form of the Naru model, which uses autoregressive model training data and uses progressive sampling to infer the estimated cardinality on the model; The DeepDB model [17] uses the sum-product network to learn the data distribution. The model is sensitive to changes in the underlying data and performs better on dynamic data. However, it is not accurate to divide cardinality estimation into supervised and unsupervised methods. In works [18], [19], the model takes the SQL query cardinality as the label and is still driven by the data in the database for training. For example, the UAE model [19] uses the Gumbel-Softmax method and the existing label supervision training stage based on the Naru and Neuro models to make the data-driven cardinality estimation more accurate. However, the model greatly increases the training time and requires more GPU storage and GPU computing power than the Naru and Neuro models.

III. METHOD

We proposed the ConvMADE model to solve the problems that the model cannot mine the features in the noise, multi-table data cannot be adaptively trained for each table, and the number of parameters is too large. The model has two structures, the RepConv structure and the DWMP structure. The RepConv structure uses a small and compact one-dimensional 1×1 convolution structure to solve the problem that the baseline model cannot separate features from noise to fully acquire features without increasing the number of parameters. The DWMP structure integrates the divide-and-conquer idea and divides the fully-connected structure into several smaller fully-connected structures to process model features in parallel, which solves the problem of being unable to acquire features adaptively and too many parameters for a single table.

A. REPCONV

The q-error of certain queries in the inference phase is high and remains constant for multiple epochs. These q-error cardinality estimates are 1, while the actual cardinality values are large. The reason is that the model weight cannot be effectively updated, resulting in the low selectivity of the query, and the final cardinality value is estimated to be 1. During the training process, it is difficult for the model to capture the features represented by these queries. In the inference stage, it is considered that these features do not exist or the probability of existence is extremely low, making the final selection degree far from the actual one. A similar situation exists in the rest of the well-behaved q-error queries, summarizing that the cardinality estimates for common queries fluctuate around the true cardinality. However, when both the estimated and true cardinality values are large, the q-error performance of these queries is good due to the nature of the q-error itself. This phenomenon is because some features are too closely combined with noise to be found, which affects the final selection. Our ConvMADE model uses the RepConv structure to increase the feature channel, discover the features in the noise, and let the weight and offset value be updated effectively so that the accuracy of the cardinality estimation of the model is higher.

The RepConv structure is inspired by RepVGG [21]. The RepVGG model is a design paradigm of the convolutional neural network, which can improve the image feature extraction ability in the process of training data of a pure convolutional structure model. In the image classification task [29], [31], [32], [33], [34], the one-dimensional convolution structure can be used to replace the fully-connected structure for the final classification. From this, it can be seen that convolution and fully-connected have certain replaceability. The principle of the RepConv structure is to change the convolution structure in the RepVGG model and put it into the ResMADE model. Using one-dimensional convolution can be used to assist the fully-connected structure to improve the accuracy of cardinality estimation.

Other fields sometimes substitute fully-connected structures and convolutional structures for each other. The data has many features in the image domain. For example, the input of a picture is $3 \times 224 \times 224$. After processing, the number of features of the picture in a module of the backbone network of the model is far greater than 150,528. Using a fully-connected

structure will consume a lot of GPU resources, so replacing the fully-connected structure with a convolutional structure or other structures as much as possible is necessary. The data used in the cardinality estimation task contains few features; the maximum number of features in a record will be at most 10,000. The number of features in the backbone network can achieve the desired effect as long as 128 or 256. Therefore, using a fully-connected structure is more appropriate than a convolutional one. However, due to the small convolution kernel parameters and flexible calculation, it is very suitable as an auxiliary structure of the fully-connected model.

Assumptions: In the training stage of cardinality estimation, the input is a data table with each record in the format of $\{col_1, col_2, \ldots, col_m\}$, and after encoding as well as processing, the input is converted to feature $= \{feature_1, feature_2, \ldots, feature_n\}$, where n is the number of features. We set where S_{th} layer features are $X = \{x_1, x_2, \ldots, x_n\}$, $(S+1)_{th}$ layer features are $Y = \{y_1, y_2, \ldots, y_n\}$ and both layers have the same number of feature elements. Where n is the number of data X of the input, which is the same as the number of features, a denotes the weight value, let A be the weight matrix, b denotes the offset value, let B be the offset value matrix, the value of Y of the $(S+1)_{th}$ layer is related to the value of all X of the S_{th} layer. The activation function is set to the ReLU function.

The principle of the RepConv structure is shown in Equation1:

$$y_i = x_i + \mathrm{f}(x_i) + \mathrm{g}(x_i) \tag{1}$$

x_i is the i_{th} feature of the S_{th} layer, which is the original input and is not processed input to the $(S+1)_{th}$ layer. $\mathrm{f}(x_i)$ is the processing structure of the original input x_i, which in this experiment refers to a layer in the MADE model. Each layer of the MADE model is a multilayer perceptron structure (MLP), and each MLP structure in this paper is composed of two fully-connected structures and activation functions. When the fully-connected structure of the MLP structure has the same input size and appears in the MADE model in parallel with the residual structure [24], this model is the ResMADE model. $\mathrm{g}(x_i)$ is the processing function of the one-dimensional 1×1convolution structure on the input, as shown in Figure[1], in parallel with the backbone network structure and the residual structure. y_i is the final output of the RepConv structure, which is the $(S+1)_{th}$ layer i_{th} feature.

The process of obtaining the feature Y set by the elements in the feature X set through the MLP structure is equivalent to multiplying the features in the X set by the weight and adding the offset value. Here the model structure is explained using equation 2:

$$y_i = (\sum_{j=1}^{n} a_{ij} x_j) + b_i \tag{2}$$

Here, x_j is the feature of the S_{th} feature layer at j, which is the input of the above formula, a_{ij} represents the weight value, b_i represents the offset value, and y_i is the feature of the $(S+1)_{th}$ feature layer at i, is the output of the above formula.

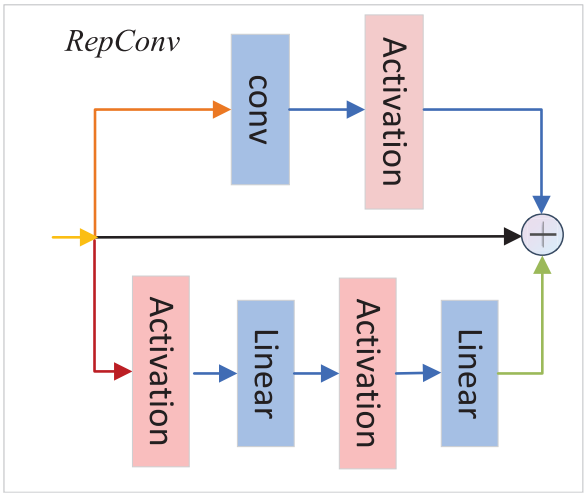


FIGURE 1. RepConv Structure.

x_i is activated after passing through the 1×1 convolution structure, which can be drawn this:

$$y_i = \begin{cases} (\sum_{j=1}^{n} a_{ij} x_j) + (1+\rho) x_i + d_i & \rho x_i > 0 \\ (\sum_{j=1}^{n} a_{ij} x_j) + x_i + d_i & \text{otherwise} \end{cases} \tag{3}$$

Here, ρ represents the weight value of one-dimensional convolution, and d_i is the sum of the convolution offset value and the offset value at fully-connected structure i. The model processes the feature set X of the S_{th} layer to obtain the feature set Y of the $(S+1)_{th}$ layer. The element y_i in Y consists of three parts:

1. Unprocessed x_i features;
2. A combination of all feature values in the X feature set with a specific weight plus an offset value;
3. Features processed by the convolutional structure.

According to Equation 3, when x_i reaches the threshold of the ReLU activation function after being processed by the convolutional structure, x_i will increase by ρ times and accumulate at y_i. If the threshold is not met, the features are not increased. However, due to the characteristics of the ReLU activation function, the feature value will not decrease but will continue to propagate forward from x_i to y_i.

This convolution operation tends to increase the feature values in the final feature set Y. The operation reduces noise for negative eigenvalues, which enables the model to detect features better in the noise. Furthermore, the increase of positive eigenvalues in the active state results in a better recognition degree due to an increase in the difference between eigenvalues. The primary reason for the large q-error is the failure of the model to capture features. Even if the scale of records corresponding to features is small compared to the total data, it may still result in a large final q-error. Failure to

capture features leads to a probabilistically small inference output, which may even be smaller than the probability of a tuple occupying the whole. Some queries with small q-errors are also impacted by the model's incorrect distinction of features. Failure to distinguish features indicates indistinct recognition, which may result in scaling issues. Convolution operation weakens the noise, learns the features, and increases the feature value, enabling the model to distinguish different features effectively. In the backpropagation process, increasing eigenvalues of the $(S+1)_{\text{th}}$ layer makes gradient changes when updating the A matrix and the B matrix more adaptable to feature variations, thereby achieving better weight parameters.

The RepConv structure enhances the model's feature acquisition ability by adding feature channels without disturbing the final output probability like non $-$ 1×1 convolution. It is a supplement to the residual structure. In the DWMP structure, the RepConv structure not only considers the above advantages but also gives a global view, improving the feature exchange between different fully-connected structures.

B. DWMP

The RepConv structure can enhance the feature acquisition ability of the model without increasing the number of model parameters. However, the model's parameters are still large; moreover, the structure cannot be adaptively trained for the single table in the multi-table. To this end, we propose the DWMP structure, which takes inspiration from the divide-and-conquer idea. This structure splits the original fully-connected structure into multiple small fully-connected structures and processes the original data features in parallel. Therefore, it can significantly reduce the number of parameters, adaptively train every single table in multiple tables, and improve the cardinality estimation accuracy of the model in multiple tables.

Data tables consist of data and hidden connection information. The relationships in multi-table data can be categorized into the internal data relationship of a single table, the relationship between data tables directly connected by the primary key and the foreign key, and the relationship between data tables indirectly connected by the foreign keys. However, in dealing with multi-table data, the original method treats all tables as a whole to obtain more comprehensive data characteristics but neglects the characteristics of each data table. On the other hand, the deep fully-connected structure can divide data tables into several parts. While considering the overall characteristics, different fully-connected structures focus on different features according to table relationships. This structure allows the model to learn the characteristics of single table data thoroughly. Thus, in the DWMP structure, the number of fully-connected structures exceeds the number of data tables. In past models, all data blocks were concatenated and trained together. When updating a specific parameter block benefits the overall network, the whole network would pay more attention to this parameter

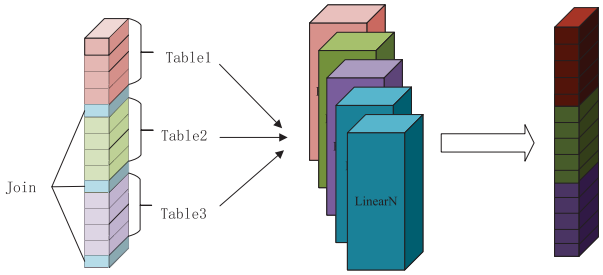


FIGURE 2. Schematic diagram of the structure of DWMP.

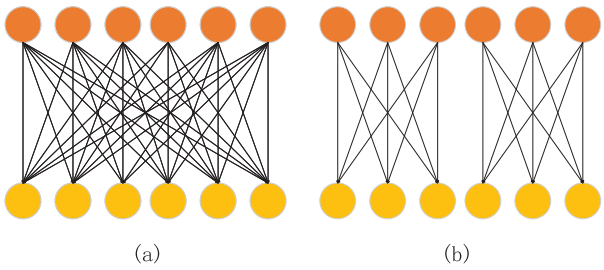


FIGURE 3. (a) Fullly-connected structure (b)DWMP structure.

block and overlook other parts. In contrast, as illustrated in Figure [2], the DWMP structure disperses the parameters, and each fully-connected structure emphasizes its parameters, allowing for better identification of optimal values. The DWMP structure captures features of different tables and connections through various fully-connected structures to learn these relationships. Since the input data is replicated at the entrance of the DWMP structure and features are relayed through the residual and RepConv structures backward, the structure retains a global view and can accommodate global information.

The main parameters for this model comprise the parameters of the fully-connected structure, including three sections: the input structure, the backbone network, and the output structure. The input structure's primary purpose is to convert encoded data into a format required by the backbone network, with data input and model output sizes designated as e and k, respectively. The backbone network structure consists of n RepConv structures, comprising a DWMP structure, a residual structure, and a one-dimensional convolution structure. The DWMP structure replaces the original fully-connected structure in parallel, containing m small fully-connected structures with input and output sizes of m/k. Figure [3] shows the situation when the m of the DWMP structure is 2, where the straight line represents the weight. The final section is the output structure, which generates the final output with an output length of L.

The total weight parameter quantity of the three sections is designated as $ek/m + n(2k^2/m + m^2) + Lk/m$, while the original parameter quantity of this section is $ek + 2nk^2 + Lk$. Since the convolution operation parameters are minimal, the parameter amount in the three sections represents roughly $1/m$ of the amount in the original model. Beyond the three sections, other parameters exist. These parameters retain the same number between the original and ConvMADE

networks, so the number of parameters in the final model is greater than $1/m$ of the original model.The number of floating-point multiplication operations is also about $1/m$ of the original structure, but the number of matrix multiplication operations increases. In the DWMP structure, the large matrix is divided into m parts for separate operations, so the number of matrix operations is m times the original. Since there is no corresponding optimization at the bottom layer, the small matrix is treated as a complete matrix, and the GPU cannot be fully utilized, thus affecting the final calculation speed.

The DWMP structure splits a large fully-connected structure into several fully-connected structures of the same size, acknowledges different table features with a comprehensive global perspective, and substantially reduces the number of model parameters required.

C. OVERVIEW OF CONVMADE

The ConvMADE model is a probabilistic model that employs probabilities to describe data distribution. It is a classification model that outputs the probability of each value of each data column under certain conditions. The shape and size of the model's output for each data column correspond to the number of unique values for that column. Unlike other classification tasks, the cardinality estimation task pursues accurate probability values based on correct classification. It cannot use stacks of modules to increase the channels of feature extraction, as done in image classification tasks, to enhance classification accuracy.

The model training stage is introduced in Figure [4]. First, the data in the table is discretized and encoded, then input to the model for training. Finally, the loss value is calculated with the discretized original data. Each distinct value after data discretization is deemed a category, and the final output shape is determined based on the category count. The output indicates the probability of each category or serves as a basis to derive the probability matrix. One-hot, binary, and embedding encoding are currently the most commonly used methods. One-hot encoding and binary encoding yield output shapes in line with the category count, with probabilities directly calculated from the output during inference. The embedding encoding output achieves its encoding shape, and the word embedding matrix [35] is inverted during inference to obtain the final probability matrix. The input data is directed to the InputLinear module, where the output of the fully-connected structure is M-times repeated according to settings, then input into the RepConv structure. The model's loss function is the cross-entropy function, which measures the distribution difference between different elements in a column in the data table in the data set. During the training stage, the model's output features are used to calculate the individual loss of each column through cross-entropy, which is then summated to obtain the final loss value according to Equation 4.

$$\text{Loss} = \sum_{i=1}^{m} H_i(p, q) = \sum_{i=1}^{m} \sum_{j=1}^{\text{bs}} p(j)\log(1/\text{softmax}(j)) \quad (4)$$

TABLE 1. Sampling rate of job-light tables.

Table	Sampling Rate
cast_info	0.87%
movie_info	2.12%
movie_companies	10.85%
movie_info_idx	17.39%
movie_keyword	4.65%
title	12.47%

Here, m refers to the number of columns in the data table, while bs pertains to the batch size used for the loss function calculation. The cross-entropy function is denoted by $H_i(p, q)$.

For multi-table data, it needs to go through data cleaning and format conversion and performs full outer join through the primary key or foreign key to merge the target table data in the data set. After sampling processing, the multi-table data can be trained, as shown in Figure [4]. It is worth noting that the multi-table training data is obtained by multiple unbiased sampling. As shown in Table 1, only part of the data is used for training. In contrast, the single-table training process is to input all data into the model for training, which is one reason why multi-table performance is less than single-table.

The process of the inference stage is shown in Figure [5]. The model uses the conditional probability formula to progressively sample the high-probability density area. It then uses importance weighting to correct the deviation caused by it, obtains the probability of the query statement, and uses the probability formula to obtain the final selectivity. Among them, progressive sampling is a Monte Carlo integration technique used in the reasoning stage to efficiently estimate the sampling range in high-dimensional data and query discrete variables that meet the conditions to obtain the selectivity of the query statement.

In the inference stage, there is the query statement "SELECT COUNT(*) FROM FROM T WHERE X, Y, Z". T is the target table of the query, X, Y, and Z are the query conditions, respectively, for a range of queries on a column. In the inference, after converting X, Y, and Z into discrete sets of coded values that satisfy the conditions, the discrete sets are input to the trained model in a particular order to get the probability values and then multiplied in a certain way to get the final cardinality estimation results. For example, if the X condition is $\text{col}_1 <= 4$, the column involved is col_1. The column takes five unique integers from 1 to 5, so the discrete values involved in the X condition are 1 to 4. The set size is 4, where the i in the last x_i in the Figure [5], and the corresponding probabilities $\hat{\text{P}}(x_1 = 1)$, $\hat{\text{P}}(x_2 = 2)$, $\hat{\text{P}}(x_3 = 3)$ and $\hat{\text{P}}(x_4 = 4)$ are obtained, and the probability of $\hat{\text{P}}(X)$ is obtained by summing. Then, the values in the set of X are input to the model by sampling, and the model is used again to find $\hat{\text{P}}(Y|X)$, and after repeating the above operation, $\hat{\text{P}}(Z|Y, X)$ is derived in the same way. Finally, using the conditional probability formula $\hat{\text{P}}(X, Y, Z) = \hat{\text{P}}(X)\hat{\text{P}}(Y \mid X)\hat{\text{P}}(Z \mid Y, X)$ multiplies the sought probabilities to obtain the selectivity of the sought query $\hat{\text{P}}(X, Y, Z)$.

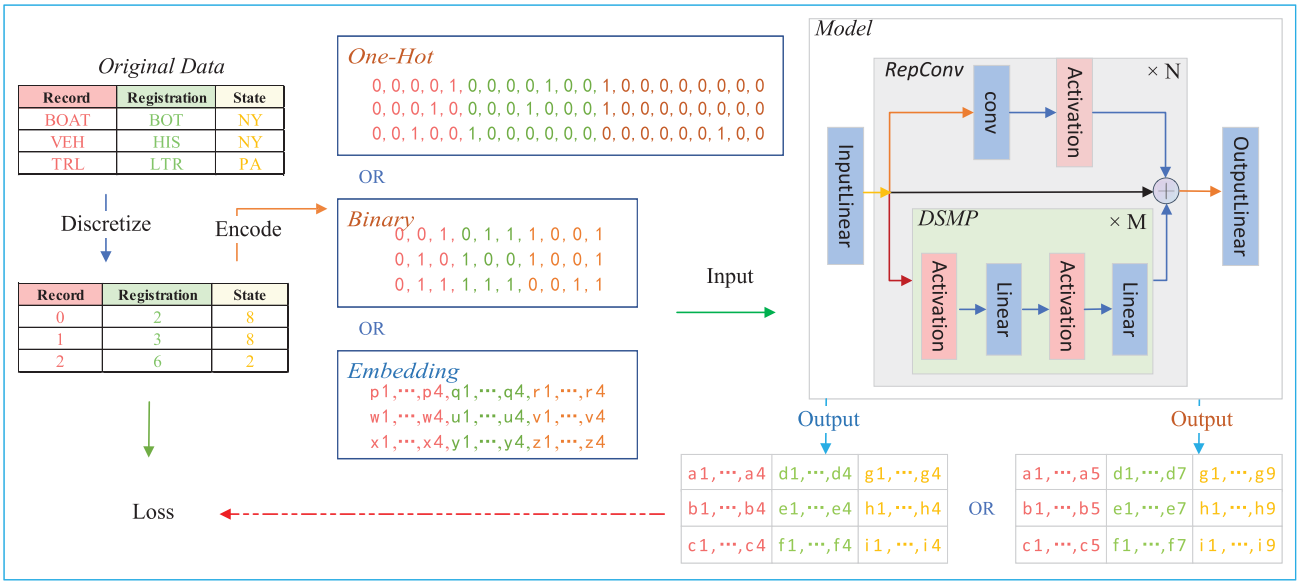


FIGURE 4. Training stage.

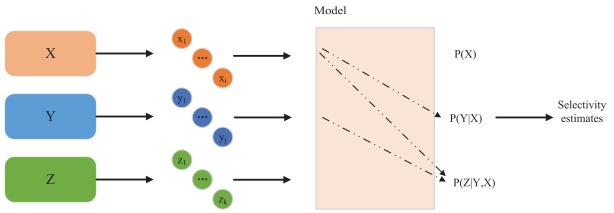


FIGURE 5. Inference stage.

Cardinality estimation is divided into a training stage and an inference stage. In the training stage, the model uses the RepConv and DWMP structures to better and more accurately fit the data distribution. In the inference stage, the probability of the model output is used to obtain the final selectivity.It can be seen that this paper only involves a small part of query optimization. Unfortunately, there is currently no query optimizer that matches the deep learning-based cardinality estimation to judge the degree of query improvement of the model.

IV. EXPERIMENT

A. DATASETS

We used real-world datasets commonly used in cardinality estimation studies, which are data-rich and easily accessible.

DMV [20]. The DMV dataset is a widely used single-table dataset containing information about owners of various types of vehicles, such as cars, sleds, and boats. The dataset includes personal information of car and boat owners and basic information about the vehicles and boats, such as model, color, engine type, and so on. This dataset is large, providing data containing about 12.5 million rows, and 11 columns of DMV table were used in this experiment.

IMDB [11]. The IMDB database, an online database used to collect information, reviews, and ratings of movies, is one of the most widely covered and popular databases. Each movie has its unique identifier in this dataset, making it easier to study the correlation and relevance between movies. Linking all the tables in this order gives 2.8×10^{14} of data volume. This experiment uses the job-light and job-m series tables. Job-ligth is used to test the performance of the model under ordinary table joins, while job-m tests the scalability of the model under larger and more complex table join modes.

B. COMPARISON METHOD

We used state-of-the-art learning-based models as a comparison method. Among them, ResMADE is an autoregressive model and one of the models used by Naru and Neuro methods. This model uses historical data to predict the probability of the current value and performs this operation several times to obtain the final query selection degree. However, this method requires many additional parameters when improving the accuracy of cardinality estimation. Also, it is not easy to train adaptively for each table when training on multiple tables, so there is more room for improvement in this model. We set this model as baseline. In addition, we also used other learning-based methods for comparisons, such as MSCN, DeepDB, and the Transformer method in Naru. In addition, we also chose traditional methods, such as histogram, Bayesian, and database PostgreSQL [26], to compare with our model.

C. ENVIRONMENT AND PARAMETER SETTING

The model environment is Win10 with NVIDIA GeForce RTX 3060 Laptop GPU (6 GB), 32G RAM, and Python version 3.8. Multi-table training follows the ray framework used by baseline, version 0.8.7, which is not stable on Win 10. Set the batch size to 2048, the number of model layers to 5, the single-table hidden unit to 256, the multi-table hidden unit

TABLE 2. Ablation experiments on the single-table DMV.

Baseline	✓	✓	✓	✓
RepConv		✓		✓
DWMP			✓	✓
q-error	1.064	**1.061**	1.074	1.073
Size	6.1 MB	6.1 MB	3.4 MB	3.4 MB

to 128, the activation function to use ReLU function is used for activation function, one-hot encoding is used for single-table, Embedding encoding is used for multi-table, 20 epochs are trained for single-table, 100 epochs are trained for multi-table, and the size of psample [27] is set to 2000 for testing.

D. EVALUATION INDICATOR

We use the traditional evaluation index q-error of cardinality estimation as a measure of accuracy to measure the quality of the estimation results. It calculates the coefficient of difference between the estimated value and the actual cardinality, as follows. The closer the value of q-error is to 1, the more accurate the model estimate is.

$$\text{q-error} = \max(\text{est/true}, \text{true/est}) \tag{5}$$

Here, est is the estimated cardinality of the query statement, and true is the real cardinality of the query statement. The accuracy of the cardinality estimator will be finally evaluated using the q-error of all queries at the four percentiles at Max,99_{th}, 95_{th}, 50_{th} and the average q-error.

E. EXPERIMENTAL RESULTS AND ANALYSIS

The experiment is divided into four parts. The first part is the ablation experiment to verify the effect of the RepConv and DWMP structures on the ConvMADE model. The second part is to verify that the ConvMADE model improves the cardinality estimation results after using the RepConv structure. The third part tests the ConvMADE model using the DWMP structure based on RepConv, the parameter change, and the average q-error and q-error change trend at 50_{th}. The last part evaluates the performance of the model on more joined tables.

1) ABLATION EXPERIMENT

In order to reflect the advantages of each structure, we conducted ablation experiments. The q-error here is the average q-error, and the DWMP structure adopts the group with the best results.

The Baseline+RepConv+DWMP experiment and Baseline+DWMP experiment in Table 2, the DWMP structure divides the fully connected structure into two groups to achieve the best average q-error of the series structure. It can be seen that the Baseline+RepConv experiment obtained the best results, and the effect of the DWMP structure declined. Since the single table contains all the data, the certainty is very high. Hence, the DWMP structure sometimes makes the model fall into a locally optimal solution, which affects the final result. Therefore, it is only necessary to add the RepConv structure and increase the feature channel to allow the model to capture features more accurately. However, when the ConvMADE model uses the DWMP structure, the number of parameters is reduced to 3.4 MB, and the all-around performance is higher than that of the benchmark model.

TABLE 3. Ablation experiments on the multi-tables job-light.

Baseline	✓	✓	✓	✓
RepConv		✓		✓
DWMP			✓	✓
q-error	28.28	26.40	28.00	**20.58**

TABLE 4. q-error on the DMV dataset.

Method	Max	99_{th}	95_{th}	50_{th}	Size
PostgreSQL	3×10^5	810.209	28.734	1.311	/
Bayes	1×10^5	12.9	1.85	1.03	/
Histogram	1685	385.001	87.951	3.004	/
DeepDB	5086	5.88	1.86	1.020	/
Transformer	814.674	6.292	2.385	1.061	/
ResMADE	**3.000**	1.700	1.230	1.024	6.1 MB
ResMADE-S	**3.000**	1.714	1.236	1.024	**5.1 MB**
ConvMADE	8.000	1.890	**1.214**	**1.019**	6.1 MB
ConvMADE-S	3.333	1.695	1.236	1.022	**5.1 MB**

The experiments of Baseline+RepConv+DWMP and Baseline+DWMP in Table 3, the DWMP structure divides the fully-connected structure into eight groups and achieves the best effect in this series. When the Baseline added the RepConv structure and the DWMP structure, respectively, the average q-error decreased. However, the reduction brought about by adding the RepConv structure is more significant because RepConv does not change the parameter amount of the model, and the model in the Baseline+RepConv experiment can view all features from a more global perspective. In contrast, the models in the Baseline+DWMP experiments are restricted to their respective groupings. In combining the RepConv structure and the DWMP structure, the RepConv structure promotes the fusion of parallel features in the DWMP structure, resulting in better results.

In summary, the performance of the RepConv structure and the DSMP structure on single-table and multi-table is different. Baseline+RepConv, a structure that increases feature channels to mine features from noise, can get the optimal value on a single table. On the other hand, Baseline+RepConv+DWMP adaptively learns the features of a single table on multiple tables and promotes the structure of feature fusion of the divide-and-conquer divide-and-conquer structure to obtain the optimal value.

2) REPCONV

We compare with PostgreSQL 14.6, histogram, Bayesian methods, and learning-based DeepDB methods on the single table. The experimental results on the DMV dataset are shown in Table4.

Table4 shows the model's performance on the percentile q-error of different cardinality estimators on the DMV dataset and the number of parameters of some models. First of all, the results of the traditional cardinality estimation method

TABLE 5. q-error on the job-light tables.

Method	Max	99_{th}	95_{th}	50_{th}	Size
PostgreSQL	4.5×10^6	34200	1814.96	13.9	/
MSCN	607.39	458.33	88.17	2.43	/
Transformer	810	207.07	30.035	**1.45**	20.3 MB
ResMADE	8169	375.09	58.30	1.62	4.2 MB
ConvMADE	8169	373.02	46.10	1.50	4.2 MB

TABLE 6. The number of parameters of the models on the DMV table.

Method	ResMADE	Conv	Conv-2	Conv-4	Conv-8	Conv-16
Size	6.1 MB	6.1 MB	3.4 MB	2.1 MB	1.4 MB	1.1 MB

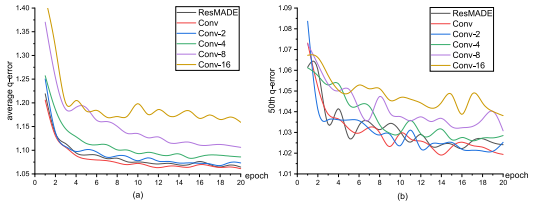


FIGURE 6. Single-table's q-error.

TABLE 7. The number of parameters of the models on the job-light tables.

Method	ResMADE	Conv	Conv-2	Conv-4	Conv-8	Conv-16
Size	4.2 MB	4.2 MB	3.8 MB	3.5 MB	3.4 MB	3.4 MB

are unstable. The Max value is five orders of magnitude higher than our model because its calculation relies on statistical data and cannot capture the complex relationship between columns. Histogram is at the bottom of the list because the statistical information is less detailed than mature databases, and the algorithm is relatively simple, so it lags behind various models in all aspects. In the learning-based cardinality estimation method, the ConvMADE model surpasses the benchmark method ResMADE in 95_{th} and 50_{th} and is also better than other methods because the RepConv structure is used in the ConvMADE model, which increases the feature channel without increasing the number of parameters so that the model can better find the optimal value. There will be no disturbance similar to the convolution of other shapes. ConvMADE-S removes a fully-connected structure from each MLP structure in ConvMADE. Even if some fully-connected structures are reduced, it is still ahead of ResMADE on the 50_{th} q-error. It is slightly lower than the ResMADE model on Max and 95_{th} because the RepConv structure deepens the model's understanding of a small part of the noise, interfering with the final result. 50_{th} reflects the model's comprehensive ability, so the RepConv structure's auxiliary function is excellent because the RepConv structure helps the model learn features from noise, making the model more likely to discover helpful information.

On multiple tables, we compare with PostgreSQL 14.6 and the learning-based MSCN approach. The experimental results on the job-light dataset are shown in Table 5.

It can be seen from Table 5 that the cardinality of PostgreSQL is estimated to be too large in Max, 99_{th}, 95_{th}, and 50_{th}, while the MSCN model performs better in Max. At the same time, the ConvMADE model is better than the ResMADE model on the 50_{th}, which is reduced by 3% on the 50_{th}, and the general q-error uses the 50_{th} to indicate the overall reasoning situation. At the same time, it can be found in Figure 7(a) that the average q-error of the ConvMADE model is also better than ResMADE. The RepConv structure improves the overall capability of the model by adding feature channels without increasing the number of parameters. Although q-error is still not as good as Transformer, the model parameter size of Transformer is 20.3 MB, while the parameter size of ResMADE and ConvMADE models is only 4.2 MB. Therefore, the overall performance of ConvMADE is higher than that of Transformer.

In summary, the RepConv structure does not reduce the q-error significantly due to the small number of parameters added by itself. However, the structure can stably increase the reasoning ability of the model by increasing the feature channel, weakening the noise, and enhancing the feature extraction ability.

3) DWMP

We use the DWMP structure to reduce the number of parameters and the average q-error while maintaining the same output features.

In table6, Conv refers to the ConvMADE model, Conv-number refers to the ConvMADE model using a deep fully-connected structure, and number refers to how many parts the original fully-connected structure is divided into.

It can be seen from Table6 and Figure6 that: The DWMP structure can reduce many parameters on a single table. In Conv-16, the parameter amount is 18.0% of the ConvMADE and ResMADE models. At the 20th epoch, the average q-error only increased by 9.7%. ConvMADE-2/4/8 performed well, with only 1.1%, 2.3%, and 4.2% increase in the average q-error, while reducing the number of parameters by 44.2%, 65.6%, and 77.0%. The more parts of the DWMP structure, the fewer parameters, and the greater the average q-error because the DWMP structure in the RepConv structure reduces the intermediate weight parameter of the structure to the original 1/M, significantly reducing the total parameter amount. However, it can be seen that the more parameters are reduced, the less the average q-error increases. Here, M equals the number, the part where the DWMP structure divides the original fully-connected structure. Because the data represents a single table, there is no need for an adaptive training table, so dividing it into multiple parts does not reduce q-error. However, even if the number of parameters drops significantly, the remaining parameters are still enough to fit the entire data distribution, so the average q-error does not increase too much.

Embedding encoding is adopted on the multi-table, and the parameters are mainly concentrated in the word embedding matrix used in the encoding. Hence, the decrease in the number of parameters is not apparent. In Table7, when the ConvMADE model does not use the DWMP structure, the average q-error and 50_{th} q-error are smaller than the ResMADE model, showing that the RepConv structure

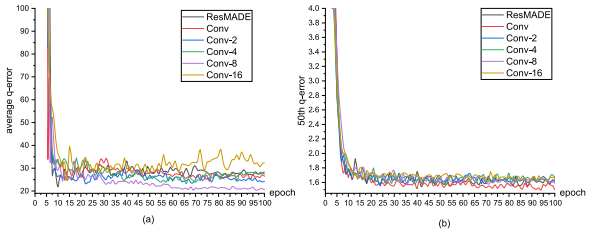


FIGURE 7. Multi-tables' q-error.

can effectively improve the model's ability. The model parameters of ConvMADE-2/4/16 are 90.5%, 83%, and 81.0% of the original ResMADE model. At the same time, it can be seen that if there are too many fully-connected structures in the DWMP structure, such as 16, its average q-error is inferior to the benchmark model because job-light has a total of 6 tables. The parallel processing of too many fully-connected structures interferes with the final result. In the ConvMADE-2/4 model, although the 50_{th} and ResMADE models are difficult to distinguish in 100 epochs, the average q-error is usually better than the ResMADE model. In ConvMADE-8, the 50_{th} effect on the percentile map is also challenging to distinguish. However, the average q-error performance is relatively good, and the average q-error on the 100_{th} epoch is reduced by 27.2% compared with the ResMADE model. Each DWMP structure of the ConvMADE-8 model makes eight fully-connected structures parallel, covering six tables, primary keys and foreign keys, and the relationship between foreign keys and foreign keys, and the optimal result is obtained.

To sum up, the DWMP structure utilizes the divide-and-conquer idea. In the single-table model, the parameters are significantly reduced. For the multi-table model, the structure adaptively trains all the tables to capture the features better and improve the comprehensive performance of the model.

4) JOB-M

Job-M is used to evaluate the performance of the estimator in a complex and large number of multi-table joins. Job-M consists of 16 tables involving 66 columns, and the relationship between them is more complex than job-light, so other learning models do not support this experiment. In this experiment, the ConvMADE model combines the two structures of DWMP and RepConv.

As can be seen from Table 8, PostgreSQL cannot obtain accurate cardinality values after multi-table joins through simple statistics, which makes q-error several orders of magnitude higher than our model. The ConvMADE model is based on the idea of divide and conquer, which pays attention to local features while taking into account the overall features, and can still maintain good accuracy under multi-table connection. Compared with ResMADE, the average q-error of our model is lower than about 18%. However, as the number of tables increases, whether it is Resmade or ConvMADE, it is difficult to obtain multi-table features to a certain extent. This is because the multi-table training

TABLE 8. q-error on the job-m.

Method	Max	99_{th}	95_{th}	50_{th}	Average
PostgreSQL	5×10^7	3×10^7	1.7×10^7	1×10^4	2.4×10^6
ResMADE	17054	2407	432	3.76	227
ConvMADE	17054	863	190	2.74	186

sampling rate decreases as the number of tables increases, and more data is difficult to be sampled so that certain features cannot be fully learned, which is currently insurmountable for the learning model on multiple tables. Moreover, due to the increase in the number of tables involved in the data, the resources consumed by each piece of data acquisition and reasoning are also increasing, and our model increases the number of matrix operations on this basis, and the speed of the model is also affected to a certain extent.

V. SUMMARY

In this study, the ConvMADE model handles the cardinality estimation problem by introducing the RepConv structure and the DSMP structure and improves the feature value by increasing the feature channel through the one-dimensional 1×1 convolution structure to promoting the update of model parameters and stably reducing the average q-error of the model. The DWMP structure adaptively fits every table in multiple tables, significantly reducing the model's average q-error in multiple tables. The DWMP structure can also significantly reduce the number of parameters in a single table. We conduct experiments on real data sets, and the experimental results show that our method has significantly improved more than other advanced methods on single-table and multi-table. Compared with the benchmark model, the single-table parameter amount can be reduced to 18% while the model still maintains a high level of capability. However,although the model divides a large matrix into multiple small matrices, the parameter scale is reduced, but the number of matrix operations is increased. The current operator cannot simultaneously calculate these multiple matrices in the GPU. It can be found that the model performs poorly in some indicators, which indicates that for some features, the ConvMADE model further ignores them based on the baseline model. Furthermore, the number of model parameters is still large. No matter which encoding method is used, the number of parameters will increase with the increase of the unique value of a column in the data table at a certain point. Therefore, we will pay attention to these issues in the following work.

REFERENCES

[1] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, "Access path selection in a relational database management system," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1979, pp. 23–34.

[2] D. Karamyan, "Cardinality estimation of an SQL query using recursive neural networks," *Math. Problems Comput. Sci.*, vol. 54, pp. 41–52, Dec. 2020.

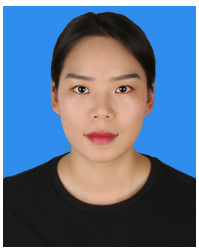
[3] J. Sun, J. Zhang, Z. Sun, G. Li, and N. Tang, "Learned cardinality estimation: A design space exploration and a comparative evaluation," *Proc. VLDB Endowment*, vol. 15, no. 1, pp. 85–97, 2021.

[4] V. Leis, B. Radke, A. Gubichev, A. Kemper, and T. Neumann, "Cardinality estimation done right: Index-based join sampling," in *Proc. CIDR*, 2017.

[5] Y. Ioannidis, "The history of histograms (abridged)," in *Proc. VLDB Conf.* Amsterdam, The Netherlands: Elsevier, 2003, pp. 19–30.
[6] R. J. Lipton, J. F. Naughton, and D. A. Schneider, "Practical selectivity estimation through adaptive sampling," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, May 1990, pp. 1–11.
[7] F. Olken and D. Rotem, "Random sampling from database files: A survey," in *Statistical and Scientific Database Management*. Charlotte, NC, USA: Springer, Apr. 1990, pp. 92–111.
[8] W. Wu, J. F. Naughton, and H. Singh, "Sampling-based query re-optimization," in *Proc. Int. Conf. Manage. Data*, Jun. 2016, pp. 1721–1736.
[9] M. Durand and P. Flajolet, "Loglog counting of large cardinalities," in *Algorithms—ESA 2003*. Budapest, Hungary: Springer, Sep. 2003, pp. 605–617.
[10] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm," in *Proc. Discrete Math. Theor. Comput. Sci.*, 2007, pp. 137–156.
[11] Z. Yang, A. Kamsetty, S. Luan, E. Liang, Y. Duan, X. Chen, and I. Stoica, "NeuroCard: One cardinality estimator for all tables," 2020, *arXiv:2006.08109*.
[12] Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, X. Chen, P. Abbeel, J. M. Hellerstein, S. Krishnan, and I. Stoica, "Deep unsupervised cardinality estimation," 2019, *arXiv:1905.04278*.
[13] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper, "Learned cardinalities: Estimating correlated joins with deep learning," 2018, *arXiv:1809.00677*.
[14] M. Heimel, M. Kiefer, and V. Markl, "Self-tuning, GPU-accelerated kernel density models for multidimensional selectivity estimation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, May 2015, pp. 1477–1492.
[15] M. Kiefer, M. Heimel, S. Breß, and V. Markl, "Estimating join selectivities using bandwidth-optimized kernel density models," *Proc. VLDB Endowment*, vol. 10, no. 13, pp. 2085–2096, Sep. 2017.
[16] S. Hasan, S. Thirumuruganathan, J. Augustine, N. Koudas, and G. Das, "Deep learning models for selectivity estimation of multi-attribute queries," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2020, pp. 1035–1050.
[17] B. Hilprecht, A. Schmidt, M. Kulessa, A. Molina, K. Kersting, and C. Binnig, "DeepDB: Learn from data, not from queries!" 2019, *arXiv:1909.00607*.
[18] J. Yang, P. Wu, G. Cong, T. Zhang, and X. He, "SAM: Database generation from query workloads with supervised autoregressive models," in *Proc. Int. Conf. Manage. Data*, Jun. 2022, pp. 1542–1555.
[19] P. Wu and G. Cong, "A unified deep model of learning from both data and queries for cardinality estimation," in *Proc. Int. Conf. Manage. Data*, Jun. 2021, pp. 2009–2022.
[20] State of New York. (2019). *Vehicle, Snowmobile, and Boat Registrations*. Accessed: Mar. 1, 2019. [Online]. Available: https://catalog.data.gov/dataset/vehicle-snowmobile-and-boat-registrations
[21] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "RepVGG: Making VGG-style ConvNets great again," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13733–13742.
[22] X. Wang, C. Qu, W. Wu, J. Wang, and Q. Zhou, "Are we ready for learned cardinality estimation?" 2020, *arXiv:2012.06743*.
[23] J. Wang, C. Chai, J. Liu, and G. Li, "FACE: A normalizing flow based cardinality estimator," *Proc. VLDB Endowment*, vol. 15, no. 1, pp. 72–84, Sep. 2021.
[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[25] J. Sun and G. Li, "An end-to-end learning-based cost estimator," 2019, *arXiv:1906.02560*.
[26] PostgreSQL Global Development Group. (2021). *PostgreSQL*. [Online]. Available: https://www.postgresql.org/
[27] B. Li, Y. Lu, C. Wang, and S. Kandula, "Q-error bounds of random uniform sampling for cardinality estimation," 2021, *arXiv:2108.02715*.
[28] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, and M. Zaharia, "Spark SQL: Relational data processing in spark," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, May 2015, pp. 1383–1394.
[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[30] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann, "How good are query optimizers, really?" *Proc. VLDB Endowment*, vol. 9, no. 3, pp. 204–215, Nov. 2015.
[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[33] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.
[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[35] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
[36] X. Y. Yang, C. G. Zeng, X. Zhou, and K. W. Wei, "Estimation of a filter factor used for access path optimization in a database," U.S. Patent 9 031 934, May 12, 2015.
[37] D. Wei, K. W. Wei, X. Y. Yang, and M. Zheng, "Optimizing query statements in relational databases," U.S. Patent 9 811 557, Nov. 7, 2017.
[38] D. S. Chen, S. H. Liu, X. Y. Yang, and M. Zheng, "Database query optimization," U.S. Patent 9 110 946, 2015.

CHAO GAO is currently pursuing the master's degree with the School of Information Science and Engineering, Xinjiang University. His research interest includes database query optimization.

JIONG YU received the Ph.D. degree from the School of Computer Science and Technology, Beijing University of Technology, China, in 2009. He was a Senior Visiting Scholar with the Information Computing Center, National Research Institute, Canada. He is currently a Professor and the Ph.D. Supervisor of computer science with the School of Information Science and Engineering, Xinjiang University. His research interests include grid computing, parallel computing, and deep learning.



ZHENZHEN HE is currently pursuing the Ph.D. degree with the School of Information Science and Engineering, Xinjiang University. Her research interest includes graph database query optimization.

XIAOQIAO XIONG is currently pursuing the Ph.D. degree with the School of Information Science and Engineering, Xinjiang University. His research interest includes database query optimization.