Qiao SJ, Yang GP, Han N *et al.* Cardinality estimator: Processing SQL with a vertical scanning convolutional neural network. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 36(4): 762–777 July 2021. DOI 10.1007/s11390-021-1351-7

Cardinality Estimator: Processing SQL with a Vertical Scanning Convolutional Neural Network

Shao-Jie Qiao[1], *Senior Member*, *CCF*, Guo-Ping Yang[1], Nan Han[2,*], Hao Chen[3]
Fa-Liang Huang[4], *Member*, *CCF*, Kun Yue[5], *Member*, *CCF*, Yu-Gen Yi[6], *Member*, *CCF*, *IEEE*, and Chang-An Yuan[7], *Member*, *CCF*

[1] *School of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China*

[2] *School of Management, Chengdu University of Information Technology, Chengdu 610225, China*

[3] *Beijing Huawei Digital Technologies Co., Ltd., Beijing 100085, China*

[4] *School of Computer and Information Engineering, Nanning Normal University, Nanning 530299, China*

[5] *School of Information Science and Engineering, Yunnan University, Kunming 650500, China*

[6] *School of Software, Jiangxi Normal University, Nanchang 330022, China*

[7] *Guangxi College of Education, Nanning 530007, China*

E-mail: {sjqiao, 3200706035, hannan}@cuit.edu.cn; chenhao220@huawei.com; hfl@nnnu.edu.cn; kyue@ynu.edu.cn yiyg510@jxnu.edu.cn; yca@gxtc.edu.cn

Received February 1, 2021; accepted July 1, 2021.

Abstract Although the popular database systems perform well on query optimization, they still face poor query execution plans when the join operations across multiple tables are complex. Bad execution planning usually results in bad cardinality estimations. The cardinality estimation models in traditional databases cannot provide high-quality estimation, because they are not capable of capturing the correlation between multiple tables in an effective fashion. Recently, the state-of-the-art learning-based cardinality estimation is estimated to work better than the traditional empirical methods. Basically, they used deep neural networks to compute the relationships and correlations of tables. In this paper, we propose a vertical scanning convolutional neural network (abbreviated as VSCNN) to capture the relationships between words in the word vector in order to generate a feature map. The proposed learning-based cardinality estimator converts Structured Query Language (SQL) queries from a sentence to a word vector and we encode table names in the one-hot encoding method and the samples into bitmaps, separately, and then merge them to obtain enough semantic information from data samples. In particular, the feature map obtained by VSCNN contains semantic information including tables, joins, and predicates about SQL queries. Importantly, in order to improve the accuracy of cardinality estimation, we propose the negative sampling method for training the word vector by gradient descent from the base table and compress it into a bitmap. Extensive experiments are conducted and the results show that the estimation quality of q-error of the proposed vertical scanning convolutional neural network based model is reduced by at least 14.6% when compared with the estimators in traditional databases.

Keywords cardinality estimation, word vector, vertical scanning convolutional neural network, sampling method

1 Introduction

Query optimization is a very important method to improve the query efficiency in database management systems. The optimization process of query optimizer is transparent to database developers. It automatically performs logical equivalent transformation and the selection of physical execution plans. The purpose of the

Regular Paper
Special Section on AI4DB and DB4AI
The work was supported in part by the CCF-Huawei Database System Innovation Research Plan under Grant No. CCF-HuaweiDBIR2020004A, the National Natural Science Foundation of China under Grant Nos. 61772091, 61802035, 61962006 and 61962038, the Sichuan Science and Technology Program under Grant Nos. 2021JDJQ0021 and 2020YJ0481, and the Digital Media Art, Key Laboratory of Sichuan Province, Sichuan Conservatory of Music, Chengdu, China under Grant No. 21DMAKL02.
*Corresponding Author

query optimizer is to obtain theoretically the best execution plan for an SQL query. However, previous studies have shown that traditional query optimizers are not capable of accurately estimating the cardinality of query results, leading to selecting poor execution plans, which directly results in slow query speed and large estimation error. The main reason for the large error of cardinality estimations is that the model cannot accurately capture the relations of multiple table join operations.

How to accurately capture the relationships across different tables or columns directly determines the quality of the target method. To cope with this problem, IBJS (Index-Based Join Sampling) [1] was proposed. The method relies on the sampling technique and existing index structures to obtain accurate results. Thus, it is straightforward that the quality of samples and indexes will have a huge impact on the query results.

Challenges and Motivations. In the past few decades, the machine learning techniques have developed rapidly, especially for deep learning and reinforcement learning approaches. The researchers do not only focus on the algorithm itself to make the proposed neural networks more powerful, but also on improving the computing performance of machines. The database team [2] has begun to use artificial intelligence technology to optimize the databases. MSCN (Multi-Set Convolutional Network) [3] uses the convolutional neural network to estimate the cardinality of query results. This method takes into full consideration the features of samples, multi-table join conditions, and predicates, thereby the MSCN model yields satisfying estimation results. However, this method has some limitations. Firstly, this method only considers simple equivalent join operation, and does not take into account left join, right join and other complex join operations across multiple tables. Secondly, the operator, e.g., "OR", "AND", and the predictors, e.g., "LIKE" cannot be supported. Thirdly, the model cannot be applied to handle complex queries.

To the best of our knowledge, neural networks have great potential in solving the cardinality estimation problems, because when the internal rules of solving the problems are unknown or difficult to describe, neurons can obtain the hidden functional relations between samples through learning and training the samples, and use the learned rules to predict the future data. The estimation problem can be viewed as supervised learning while the label is the real cardinality. Then, the problem is how to deal with SQL queries. The biggest challenge is how to capture the SQL semantics. Currently, the existing database optimization techniques based on learning methods do not work well, and sometimes the results are not satisfying. In this study, we aim to propose a new learning-based method to obtain more accuracy query results than the state-of-the-art models.

Contributions. In this paper, we propose a natural language processing model called VSCNN (vertical scanning convolutional neural network) to cope with the aforementioned challenges. The proposed model is actually a vertical scanning convolutional neural network. We train a CNN (convolutional neural network) with one layer of special convolution. Its input vector is the word vector of SQL queries obtained from an unsupervised method. We keep the word vectors unchanged and employ them to automatically learn the optimal values of other parameters in the model. Importantly, we encode the table names by one-hot encoding method and encode the samples into bitmaps, respectively, and then merge them. This model can capture the semantics of each word in a complex SQL query, and also the join relationship between tables.

In addition, we conduct experiments on the real IMDB datasets to evaluate the performance of the VSCNN model and show that our approach is more powerful than the state-of-the-art models, i.e., IBJS [1] and MSCN [3]. The experimental results are promising, which also shows the feasibility of the learning-based cardinality estimation method in current database management systems.

The remainder of this paper is organized as follows. Section 2 summarizes the existing cardinality estimation methods. Section 3 introduces the working mechanism of the proposed cardinality estimator. Section 4 evaluates the proposed vertical scanning convolutional neural network based cardinality estimation model experimentally. Section 5 concludes this paper and discusses future research directions.

2 Related Work

2.1 Cardinality Estimation Methods

After several years of development on database query optimization techniques, the traditional cardinality estimation method has obtained good query performance, which performs well in some specific scenarios. In this study, we classify the traditional cardinality estimation methods into three categories.

1) *Histogram Techniques*[4]. The most important application of histogram techniques in databases is selectivity estimation. When the database uses histogram analysis to calculate the cardinality, the column data to be analyzed will be partitioned into several parts with the same number, and each part is called a bucket. CBO (Cost-Based Optimization)[2] can easily obtain the distribution of the size of columns. However, the histogram-based techniques cannot effectively estimate the correlation between different columns.

2) *Probability Algorithms*[5–9]. These algorithms are designed based on the theory of probability and statistics. They can overcome the disadvantages of the existing cardinality estimation methods, i.e., excessive memory requirements or being difficult to merge, and can control the error within the required range by averaging the results of the query.

3) *Sampling Methods*[1,10–13]. These methods use data samples to estimate the cardinality, which can improve the accuracy of cardinality estimation to a certain extent. But, it will cause other problems, such as inappropriate indexes and samples, space explosion and the 0-tuple problem. The sampling methods can improve the accuracy of cardinality estimation, but they will bring in space overhead[14].

2.2 Learning-Based Cardinality Estimation Methods

Currently, many experts and scholars are devoted to applying the artificial intelligence techniques to improve the performance of databases. For cardinality estimation, the machine learning techniques especially for the deep learning models can be used to solve this problem. The framework of learning-based cardinality estimation is given in Fig.1, which is motivated by the fact that modern join enumeration algorithms can find the optimal join order for queries with dozens of relations[15]. Cardinality estimation plays an important role in query optimization[16]. If the cardinality estimator is not accurate, it will generate many bad query results[17].

Regression-based model[18] has been used for cardinality estimation. BBA (Black-Box Approach)[19] applies the idea of classifying query statements according to structure of queries, but this method does not work for unknown structure of queries. A deep likelihood model[20] can capture the distribution of data between multiple columns, but not for multiple tables. Liu *et al.*[21] proposed to use neural networks to estimate the cardinality of query results. It is only effective at processing the simplest SQL queries, because it does not consider join queries and complex queries. We use the MLP (MultiLayer Perceptron) as a model for processing vectors. Since 1990s, MLP has been widely used in vowel classification[22], and the best classification effect was achieved. Recently, MLP has been applied to the classification and prediction of DNA (DeoxyriboNucleic Acid) sequences[23]. MSCN[3] introduces a multi-set convolutional network. Different components of SQL queries are encoded into vectors, and then these vectors are put into MLPs. Then, different vectors are merged to form a new vector. Hereafter, the vector is put into another MLP to figure out the estimated cardinality. The method works well for simple queries, but not for queries with complex semantics. Sun and Li[24] designed an end-to-end learning-based cost estimator. This model can deal with complex SQL queries and estimate the cost of queries, but it needs physical execution plans. It is difficult to obtain the physical execution plan, and the model is too complex.

Nowadays, the combination of machine learning and database techniques has become a new research direction, including many aspects, for example, learned join order selection[25], knob tuning[26,27], which prove that applying AI4DB (artificial intelligence for databases) techniques to improve the performance of databases is very promising. For crowdsourcing databases, Li *et al.*[28] proposed a crowd-powered database system CDB that supports crowd-based query optimization, which focuses on join and selection operations, and no longer rigidly adheres to traditional relational databases. Fan *et al.*[29] proposed a template ranking model to suggest templates relevant to query keywords, generating SQL queries from the suggested templates.

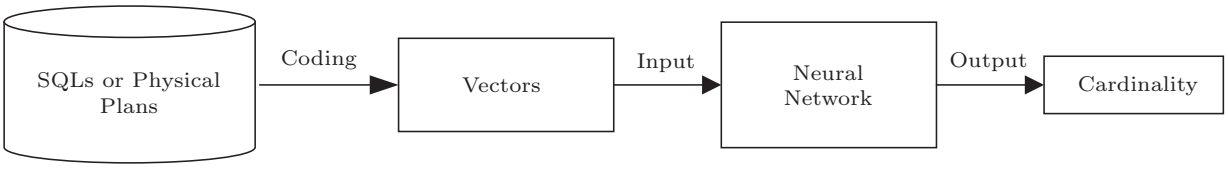


Fig.1. Framework of learning-based cardinality estimation methods.

3 Cardinality Estimator

Basically, our work is to encode SQL by the proposed method, and then input it into the prediction model to generate the cardinality of queries, and train the model by comparing it with the real cardinality. Straightforwardly, we aim to build a supervised learning model. Consequently, the following challenges will affect the success of the work. Firstly, how to encode SQL queries into vectors or matrices to obtain meaningful semantic information? Secondly, how to design an efficient model for supervised learning? Thirdly, how to obtain the training data? Here, we will introduce the proposed framework for encoding SQL queries in detail, as presented in Fig.2.

Fig.2 shows framework for encoding SQL queries. In Fig.2, there are six tables in total, and we encode the table names to a one-hot vector. Then, we sample the tables involved in the query and use bitmaps to represent them. Lastly, we merge the table name vector with the bitmap to obtain the table information vector representing the query. We transform the content contained in the "WHERE" clause into a word matrix by the SQL embedding technique in Subsection 3.2.

3.1 Table Information Encoding

3.1.1 Representation of Table Names

We record all the queries in a database represented by Q, and all the tables in the database as T. Given a query $q \in Q$, we represent the tables involved in the query q as $T_q \subset T$. For query q, the semantics of table names is relatively independent for join operations and predicates, thereby the one-hot encoding method can be adopted. We count the number of tables in T, and the number is $|S_T|$. For a specific query involved in table T_q, we transform it into a one-hot matrix $\boldsymbol{m}_t$. According to the aforementioned operation, we can obtain the matrix of the table names in the query.

3.1.2 Encoding of Sampling Tuples

In order to make the model learn more information from the table, in addition to encoding the table name, we also need to encode the sampling tuples from the related columns, which can help know the distribution of data in the table. For each table, we only need to know whether the samples meet the query conditions in the table. If they meet the query conditions, it is represented by 1; otherwise 0. Adding sampling features makes the proposed model more likely to perform join estimations. We represent the sample of a specific table T_q involved in query q as B_q.

If we only use the sampling method to estimate the cardinality, we will encounter an empty base table (0-tuple). It will have a big impact on the accuracy of estimation results. In the proposed VSCNN model, we adopt the deep learning technique by integrating the sampling method, which can well handle the 0-tuple problem.

We represent the table information E_q by (1):

$$E_q = \frac{\sum_{q \in Q}^{Q} MLP_T([T_q, B_q])}{|S_T|}. \tag{1}$$

The basic idea of (1) can be explained as: if a query contains $|S_T|$ tables, the query should be represented by two vectors. Each vector (T_q, B_q) is composed of one-hot encoding of a table name and a bitmap encoding of sampling tuples. Then, $|S_T|$ vectors are put into the $|S_T|$ MLPs, and the $|S_T|$ vectors obtained from the MLPs are summed up. Lastly, the average value E_q is calculated and the table information of the query will be obtained.

3.2 SQL Embedding

3.2.1 Skip-Gram for SQL Embedding

We will face two main challenges in cardinality estimation: the disappearance of semantic propagation and the explosion of searching space. For one single table,

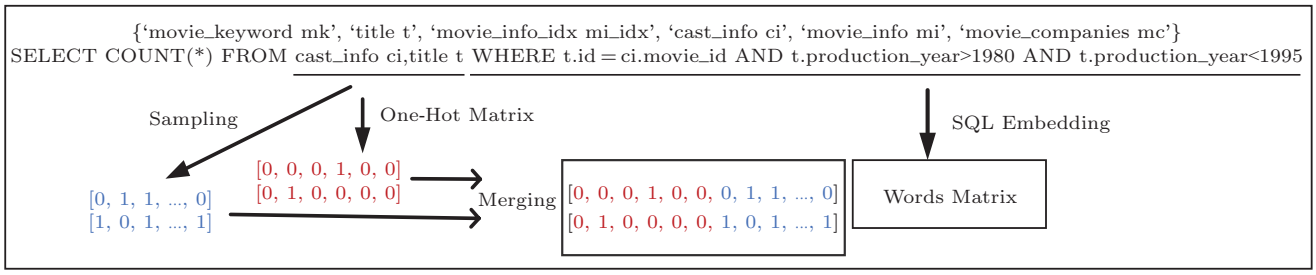


Fig.2. Proposed framework for encoding SQL queries.

given a filter condition, it is straightforward to figure out the correct cardinality. However, for the join query, if the query is very complex, the searching space will become very large, and then the problem of semantic disappearance will occur. In order to maintain more information of multi-table joins, we should choose a kind of vector having memory function or being capable of mapping words to context sensitive vectors.

We express the meaning of each word in the "WHERE" clause with a vector, and we use the Skip-Gram model [30] to perform SQL embedding method. Fig.3 shows the detail of the word vector by one-hot encoding is inputted into the single-layer neural network, in which the number of neuron nodes in the input layer should correspond to the dimension of the word vector via one-hot encoding. Specifically, we first encode the contents after the "WHERE" clause as a one-hot vector. Because the information in the one-hot encoding vector is very sparse and cannot represent the similarity between words, we update the values of vectors by the learning-based strategy. In this fashion, we can encode the statement into a dense vector.

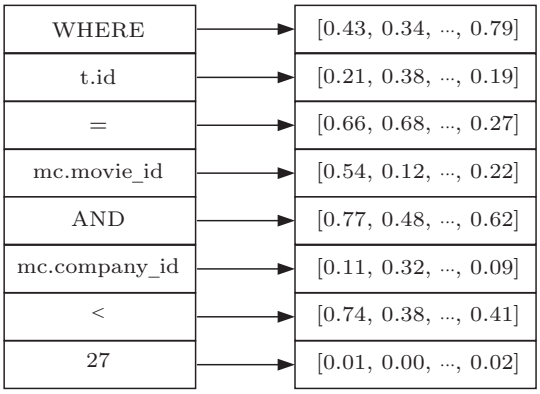


Fig.3. Example of an SQL query embedding matrix.

In this paper, the SQL embedding is used to map an SQL into context sensitive vectors, which aims to deal with "WHERE" clauses based on natural language processing. It is worth to note that we employ the basic idea of natural language processing to process the "WHERE" clauses. Because one-hot vectors cannot accurately describe the similarity between words or symbols from various dimensions, including ("AND", "OR"), (">", ">="), ("<", "<="), ("%XX", "%XX%") and so on, we map the "WHERE" clause into a context sensitive vector in order to perform the natural language processing by using the convolutional neural network.

This paper mainly uses the basic idea of natural language processing to deal with SQLs, and the SQL statement itself contains semantics. "WHERE" and "<" in the same SQL sentence naturally have context relationship, but this relationship is not so strong as that in the natural language; therefore when we scan the word vector, we set three convolution kernels with different sizes to scan at the same time, and the model can recognize the semantics of SQLs from different aspects. The three convolution kernels can help improve the recognition accuracy of the semantics of SQLs by the convolution neural network.

3.2.2 *Negative Sampling for Accelerating Gradient Descent*

There are 100 000 sampling queries in the training set. In order to accelerate the speed of gradient descent during training, we use the negative sampling method. For a query statement, we select a central word, such as "AND", and mark it as w. The words around it are denoted by $context(w)$. If this central word is correlated to $context(w)$, it is viewed as a real positive sample. When we use the negative sampling method, we can obtain some center words w_i different from w where $i = 1, 2, ..., |neg|$, and $|neg|$ represents the total number of those center words. Therefore, we will make up $|neg|$ negative samples that do not really exist based on $context(w)$ and w_i. We use this positive sample and $|neg|$ negative samples to do binary logistic regression. The model parameters θ_i of each word w_i corresponding to negative sampling and the word vector of each word are obtained. In order to keep the expressions consistent, we define the positive sample as w_0. In the phase of logistic regression, the positive sample should be expected to meet the following requirements, and the probability of the center word w_0 and its surrounding word appearing at the same time is calculated by (2).

$$P(context(w_0), w_i) = \sigma\left(\boldsymbol{x}_{w_i}^{\mathrm{T}} \theta_{w_i}\right). \tag{2}$$

The negative example expectation should obey (3), and the probabilities of the center word w_0 and the word w_i not appearing at the same time are calculated by (3).

$$P(context(w_0), w_i) = 1 - \sigma\left(\boldsymbol{x}_{w_i}^{\mathrm{T}} \theta_{w_i}\right). \tag{3}$$

Then, by using the knowledge from logistic regression, we can obtain the likelihood function of the model

as shown in (4) and we need to figure out the maximum probability.

$$L = \sum_{i=0}^{|neg|} \left(y_i \log\left(\sigma\left(\boldsymbol{x}_{w_i}^{\mathrm{T}}\theta_{w_i}\right)\right) + (1-y_i)\log\left(1-\sigma\left(\boldsymbol{x}_{w_i}^{\mathrm{T}}\theta_{w_i}\right)\right)\right), \tag{4}$$

where $y_0 = 1$ represents a positive sample, $y_i = 0$ represents a negative sample, $i = 1, 2, ...,$ and $|neg|$ represents the label of a negative sample. $\sigma(*)$ is the logistic regression function, $\boldsymbol{x}_{w_i}^{\mathrm{T}}$ represents the word vector corresponding to the i-th word and θ_{w_i} represents the weight corresponding to the i-th word.

We use the random gradient ascent method to update the gradient of only one sample at a time, and finally we can obtain the proper $\boldsymbol{x}_{w_i}^{\mathrm{T}}$ and θ_{w_i} values.

3.3 Vertical Scanning Convolutional Neural Network Model

The standard neural network structure is not good at dealing with vectors or matrices representing different information, thereby we use different neural networks to deal with different parts of a specific query. Fig.4 shows the framework for cardinality estimation, which consists of three stages: 1) the convolutional neural network vertically scans the "WHERE" clause semantics, 2) the MLPs process table information, and 3) the vector from the first two stages is transferred to a new MLP. The details of these three steps are presented in the following subsections.

3.3.1 *Learning Table Information*

We choose MLP [22] as the model of learning table information. The new vector that combines the table name vector and the sample vector is represented by (T_q, B_q), and then we input this vector into the MLP represented by $MLP(T_q, B_q)$. However, we encounter a challenge, that is, the dimension of each query's table information is not uniform. For example, query q_1 contains one table, while query q_2 contains two tables. Then, query q_1 contains one row vector, while query q_2 contains two row vectors. Therefore, we use zero to fill in the row vector. We take query q_n with the most row vectors as the benchmark, assuming that q_n contains

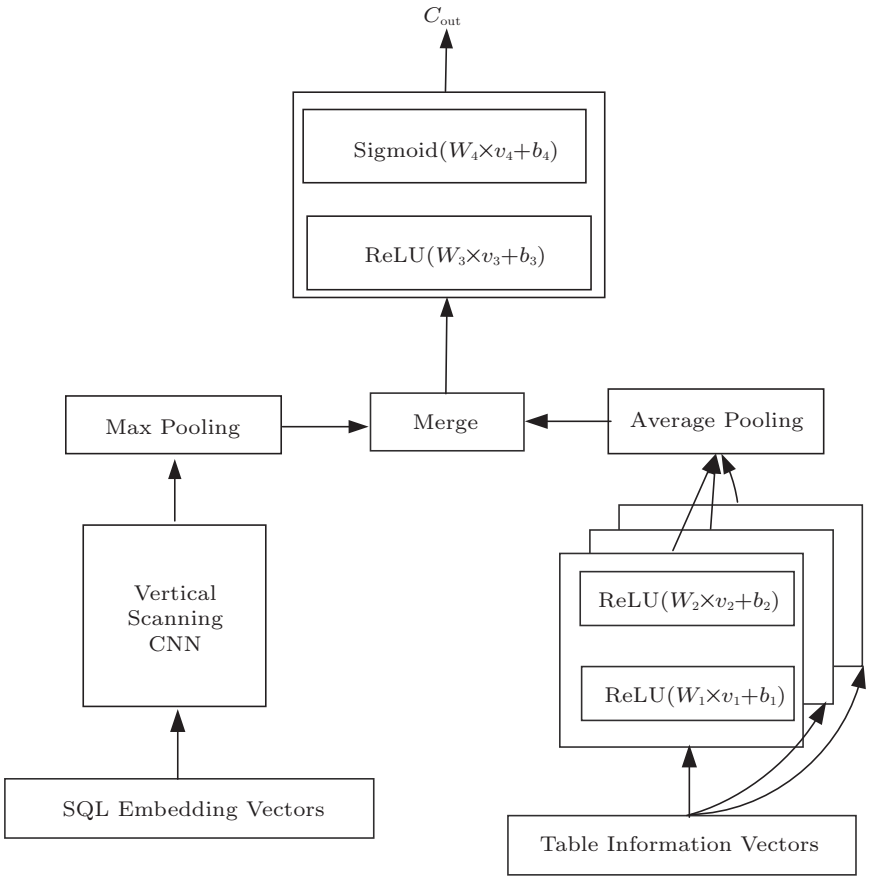


Fig.4. Framework for cardinality estimation.

n tables. For example, if query q_1 contains only one table, that is, a row vector, then we fill in $n-1$ zero in this row vector.

In order to generalize query features, we design an average pool, which is used to add all MLPs and average the result by $|S_T|$. These MLPs consist of two fully-connected layers. The first layer is $\mathrm{ReLU}(W_1 \times v_1 + b_1)$, where W_1, v_1 and b_1 represent the weight, the table information vector and the bias of linear regression in the first layer, respectively, and the second layer is $\mathrm{ReLU}(W_2 \times v_2 + b_2)$. We use the ReLU (Rectified Linear Unit) activation functions because they show strong empirical performance and are fast to converge. This point is discussed in detail in Subsection 4.2.

3.3.2 *Convolutional Neural Networks*

Our inspiration comes from sentence classification. In general, we use recurrent neural networks to complete natural language processing tasks, while the image recognition task is achieved by convolution neural networks. In this study, we use a novel convolution neural network to convolute the "WHERE" clauses. The first step is to transform the clause into a word matrix. Each word in a query is composed of an n-dimensional word vector, that is, the size of the input matrix is $m \times n$, where m is the length of the sentence. Fig.5 shows the working mechanism of the proposed CNN convoluting the matrix. For the embedding matrix, the filter no longer slides horizontally, but just moves downward, which is similar to extracting the local correlation between words in the n-gram model. In this task, we define three types of filters with different widths: [2, 3, 4] (as shown in Fig.5, in the embedding matrix, there are three kinds of one-hot vectors connected by different curves). The proposed CNN model contains three filters with different widths. The length of each filter is equivalent to the dimension of the word vector. It is worth noting that there will be a lot of filters in training the real samples. Then, nine convoluted vectors are obtained by applying different filters on different word windows. Then, each vector is maximized and pooled, and each pooled value is spliced. Finally, the feature representation F_q of the "WHERE" clause is obtained.

Instead of splitting SQL embedding into three kinds, we divide convolution kernels into three kinds with the height of 2, 3 and 4, respectively. Then, we use these three convolution kernels to scan downward in the vertical direction from the word vector of the "WHERE"

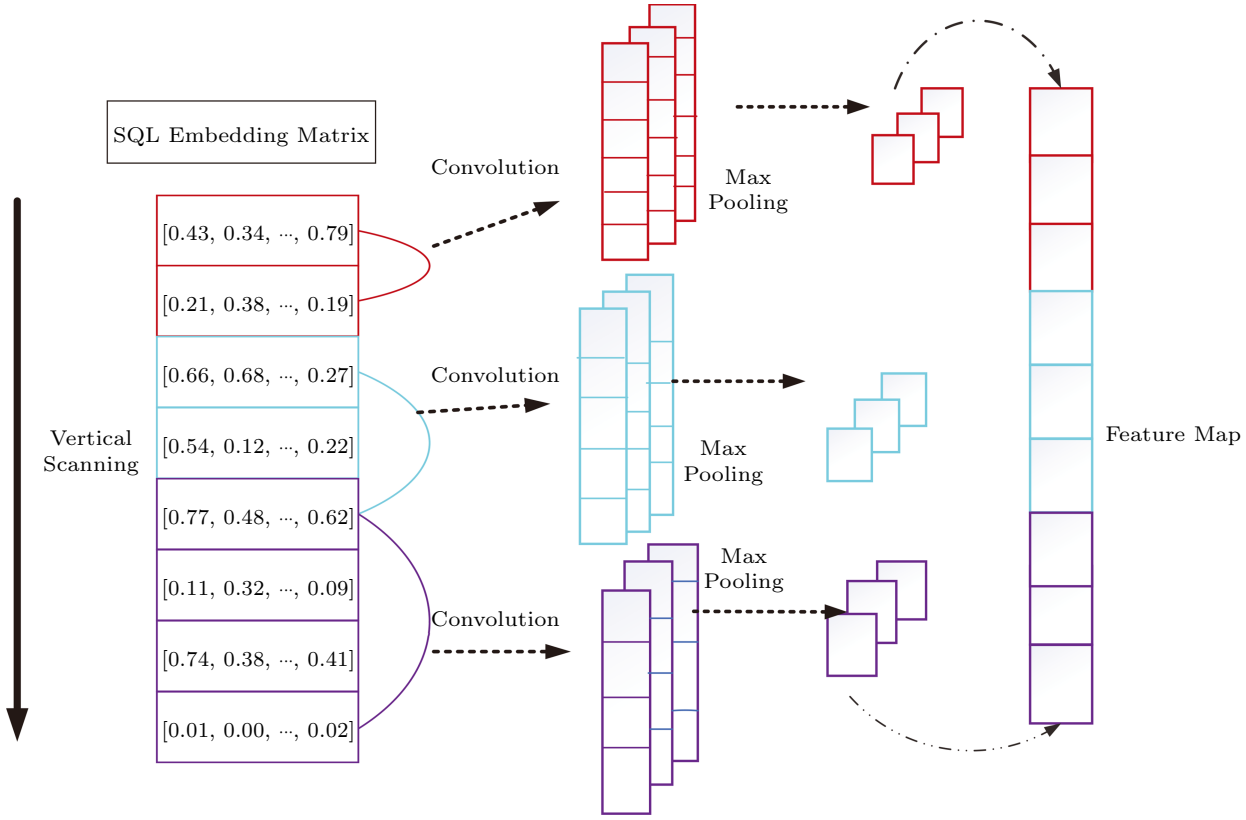


Fig.5. Working mechanism of a vertical scanning convolutional neural network.

clause at the same time. We set three convolution kernels with the height of 2, 3 and 4, respectively. After the convolution operation, we can obtain the MaxPooling layer and lastly get the feature map.

Specifically, we specify the convolution kernel to 300×2 and the stride is 1. The word vector is set to 300 dimensions in this study, and the convolution kernel can only slide (scan) down the matrix. Furthermore, we set three convolution kernels of 300×2 with different weights. As shown in Fig.5, when the convolution kernel on top with the height of 2 scans downward in the vertical direction, three feature maps will be generated. Similarly, we set up two convolution kernels of 300×3 and 300×4, respectively, and generate the feature map in the same fashion.

3.3.3 Merge and Output Operations

When we obtain E_q and F_q, we need to merge them as (E_q, F_q). As shown in Fig.4, we put the vector into the final neural network. The neural network consists of two fully-connected layers. The first layer is $\mathrm{ReLU}(W_3 \times v_3 + b_3)$, and the second layer is $\mathrm{Sigmoid}(W_4 \times v_4 + b_4)$.

The cardinality estimation C_{out} of the final prediction is a number of numeric type, thereby we have to choose the Sigmoid function. And we use the ReLU activation function, because it shows strong empirical performance and is fast to converge. The final estimated results $C_{\mathrm{out}} \in [0, 1]$. Due to the large cardinality of the samples, we cannot directly compare C_{out} with C_{label}. Then we normalize the tag cardinality $C_{\mathrm{label}} \in [0, 1]$, and C_{out} and C_{label} can be compared.

3.4 Cold Start Problem

Many websites or portals want to provide the personalized recommendation services, how to make the most effective recommendation without knowing the user (i.e., having no user behavior data)? This leads to the problem of cold start. In addition, the cold start problem arises when a new user or new item enters the personalized recommendation system, because such user/items system does not have enough information to make a decision. For example, a new user has not rated any items and not yet viewed any items, and then it would be difficult for the system to build a model on that basis.

In the research of AI4DB (Artificial Intelligence for Database), one key challenge of learning-based algorithms is the "cold start problem", i.e., how to train the model before having concrete information about the query workload. Our approach is to obtain an initial training corpus by generating random queries based on the schema information and extracting the literalness from actual values in the database.

The information of data samples in the training set is mainly composed of table names, join mode, a varying type of predicates and real cardinality. The real cardinality is obtained by the SQL analysis tool. In this study, we use IMDB (Internet Movie Database) to select several candidate tables and find the relationships of the primary and foreign keys. We use these relationships to randomly generate join conditions. It is meaningless to blindly increase the number of join conditions, because the join form we choose is $(T_q^1.Col_q^1{=}T_q^2.Col_q^2$ AND $T_q^2.Col_q^2{=}T_q^3.Col_q^3)$, which is represented by $(A \bowtie B) \bowtie C$ or $A \bowtie (B \bowtie C)$. No matter how many join conditions are added, the performance of the model will not be affected. Therefore we set the number of join operations in each query to 0, 1 or 2, which makes the combination space smaller. In terms of the predicates, we randomly select the columns of numeric type in the related tables. These operators "$>$,$<$,$=$,$!=$" are selected for the logical relation. Then, we aggregate the predicate expressions with "AND/OR" operators. Lastly, we produce the training data of queries.

4 Experiments and Discussions

4.1 Workload and Comparing Methods

We use the real IMDB database because it contains several real-world correlations between tables. The IMDB database is rich in contexts and covers a wide range of movies, and IMDB lays a solid foundation for coping with the challenge of cardinality estimation. We conduct comparison experiments on three different query workloads: 1) a synthetic workload: this workload contains 5 000 unique queries between 0–2 joins; 2) a scale workload: queries of this workload contain more joins; and 3) a job-light workload, called JOB (Join Order Benchmark) workload. This workload is a lightweight workload extracted from the JOB without any predicates on strings, and it contains 70 queries between 1–4 joins.

It is a commonly-used method to obtain the initial word vector by unsupervised learning methods before performing large-scale supervised learning. For word vectors, we use pre-trained word vectors for the training samples. In order to demonstrate that our model

can handle complex predicates with strings, we generate 50 000 queries with strings based on the JOB workload.

Table 1 shows the characteristics of different cardinality estimators on JOB workloads. We choose some of the most popular relational databases to compare, i.e., PostgreSQL v9.6.20, MySQL v5.7.32, SQL Server 2008 R2 and Oracle 12c Release2. We choose the state-of-the-art models as the comparison methods. BBA (Black-Box Approach) [19] was proposed based on the idea of classifying query statements according to query structures, but this method cannot be applied on unknown structures of queries. IBJS [1] is a cardinality estimation based on indexes, which relies too much on the index structure. MSCN [3] is a multi-set convolutional network, but it does not have the function of semantic extraction, thereby it cannot be used to process complex SQL or the string type of columns. It is straightforward that the cardinality estimator in traditional databases does not use the complex network structures and the encoding techniques. The learning-based cardinality estimation methods use complex neural networks and complex encoding methods. We use 90% of the records in the database as the training samples and the remaining samples as the validation set.

Table 1. Characteristics of Cardinality Estimators on JOB Workloads

Estimator	Represent Network	Predicate Network	String Encoding	Simple Bitmap
PostgreSQL	No	No	No	No
MySQL	No	No	No	No
Oracle	No	No	No	No
SQL Server	No	No	No	No
BBA	No	No	No	No
IBJS	No	No	No	Yes
MSCN	Yes	Yes	Yes	Yes or No
VSCNN	Yes	Yes	Yes	Yes or No

We develop the proposed model with PyTorch framework [31] and CUDA [32]. All experiments are conducted on a machine with Intel® Xeon® CPU i7-6700k, 16 GB memory, 128 G SSD and Ubuntu 18.04 operating systems.

4.2 Selection of Activation Function

Firstly, when using Sigmoid or other functions to calculate the activation function, the cost of calculation (i.e., the exponential operation) is quite high. When calculating the gradient of errors through the back propagation method, the derivation calculation involves division operation, and then the amount of calculation is very large.

Secondly, when using the ReLU activation function, the amount of calculation can be reduced. This is because, for the deep neural network, when the Sigmoid function propagates backward, it is easy for gradient to disappear, i.e., when the Sigmoid function approximates to saturation, the transformation becomes too slow and the derivative value tends to be 0, which will cause the loss of information, and then it is unable to complete the training of deep neural networks. The derivative of Tanh is larger than that of the Sigmoid function, the gradient changes fast, and the convergence speed is faster in the phase of training. But it will cause the problem of gradient disappearing. However, the ReLu function does not have the gradient vanishing problem.

Thirdly, the ReLU operation will make the output of some neurons be zero, which makes the network become sparse and can help reduce the interdependence of parameters and alleviate the over-fitting problem. The ReLU function is actually a piecewise linear function, which changes all negative values to zero while keeping the positive values unchanged. This operation is called unilateral suppression that plays an important role in deep learning. Because of this unilateral inhibition, the neurons in the neural network also have sparse activation. Especially, in deep neural network models (such as CNN), theoretically, when n layers are added to the model, the activation rate of ReLU neurons will be reduced by 2^n where n is the number of layers [33].

4.3 Regularization

In order to avoid the phenomenon of over-fitting, we train the proposed neural network by the dropout technique. Dropout is used in the phase of deep learning network training, and the neural network unit is temporarily discarded from the network according to a certain probability. It is worth noting that the dropout is only temporary. For random gradient descent, each mini-batch trains a different network and it is randomly discarded. When the dropout probability is applied in training samples, each neuron has a 50% probability of being removed, which makes the training of one neuron independent of another neuron and weakens the correlation of features. For fitting the new data, L_1-norm is not stable and the loss by applying L_1-norm changes a lot, while L_2-norm changes little. L_1-norm tends to make coefficients sparse, but L_2-norm has no sparse coefficients. In a word, L_2-norm is more stable than L_1-norm, thereby we choose L_2-norm. For regulari-

zation, we use the dropout technique on the penultimate layer with a constraint on L_2-norm of the weight vectors [34]. At the same time, we also use early stopping to prevent the over-fitting problem. When we train deep learning neural networks, we aim to obtain the best generalization performance (that is, it can fit the data well). But all the standard deep learning neural network structures, such as fully connected multi-layer perceptron, suffer from substantial over-fitting. That is, when the network performs well in training, the prediction error is low. However, in reality, at a certain moment, its performance in the test set has begun to deteriorate. The regularization helps keep the number of parameters in the model smaller.

4.4 Hyperparameters Tuning

In order to make the model reach an ideal state, we have to tune the hyperparameters including the number of epochs, the size of mini-batch, the weight of filters, the width of filters, the number of hidden units and the learning rate in experiments. We compare many gradient descent algorithms and find that Adam [35] achieves the best performance, thereby we apply the Adam gradient descent algorithm [35] in this study.

In experiments, we try the following settings as shown in Table 2.

We use 60% of the data to train the samples. The training set contains 54 000 queries, and the validation set contains 6 000 queries. We find that the comparison models can obtain good performance when the parameters are set to 100 epochs, 1 000 batch size, the weight of filters [128, 128, 128], the width of filters [2, 3, 4] and 256 hidden units.

After several settings of parameter combinations, we observe that the comparison models perform better when training for 100 epochs than when training for 200 epochs. This is a phenomenon of over-fitting. As for the width of filters, in general, the training sample contains three words, which produce some specific semantics, for example, the predicate "production_year", "=", "2021". Therefore, the filter window is set to [2, 3, 4] as necessary. Fig.6 shows the influence of different filter combinations on the loss defined in (6).

Furthermore, in this set of experiments, we discuss the influence of the number of windows on training. Fig.7 shows the training performance comparison between three windows and four windows. We set the three windows to [2, 3, 4], and set two different groups of four windows [1, 2, 3, 4] and [2, 3, 4, 5], respectively. We set up 30 epochs for training.

According to Fig.7, we can see that [2, 3, 4] has the best performance, because four windows are easy to fall into the over-fitting phenomena. In addition, it is time-consuming to specify too many windows.

4.5 Cardinality Estimation Quality

We use the following q-error [36] defined in (5) and the loss function given in (6) to evaluate the performance of distinct models. q-error can be used to measure the deviations of cardinality estimates from ground-truth cardinalities. In (5) and (6), C_{out} represents the estimated value and C_{label} represents the ground-truth value.

$$q\text{-error} = \max\left(\frac{C_{\text{out}}}{C_{\text{label}}}, \frac{C_{\text{label}}}{C_{\text{out}}}\right). \tag{5}$$

$$Loss = \log \sum_{i=0}^{n} \max\left(\frac{C_{\text{out}}}{C_{\text{label}}}, \frac{C_{\text{label}}}{C_{\text{out}}}\right). \tag{6}$$

Fig.8 demonstrates the convergence performance of the proposed VSCNN model by comparing it with other three models on the validation set. We use 54 000 queries as the training set and the remaining 6 000 as the validation set. We find that NoSamplingMSCN (MSCN without sampling) shows the trend of fluctuation as the number of training epochs grows, having the slowest convergence speed. MSCN works the best because it converges the fastest. In addition, our model VSCNN is better than NoSamplingMSCN without sampling. We can see that the convergence speed of the proposed VSCNN model will be significantly improved after using the proposed negative sampling method and it is more accurate with the lowest loss. Furthermore, we also find that MSCN and VSCNN algorithms converge

Table 2. Parameter Setting of Experiments

Number of Epochs	Batch Size	Weight of Filters	Width of Filters	Hidden Units	Learning Rate
50	500	[128, 128, 128]	[1, 2, 3]	256	0.000 1
100	1 000	[256, 256, 256]	[2, 3, 4]	512	/
/	/	[512, 512, 512]	[3, 4, 5]	1 024	/

when the epochs is 100. After 100 epochs, the curve of our model almost coincides with that of MSCN. As for NoSamplingMSCN and NoSamplingVSCNN, the performance of q-error is improved by 40.4% and 23.5%, respectively.

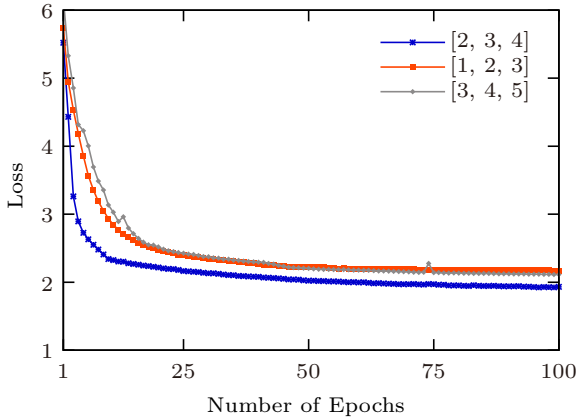


Fig.6. Impact of different filter combinations on the loss.

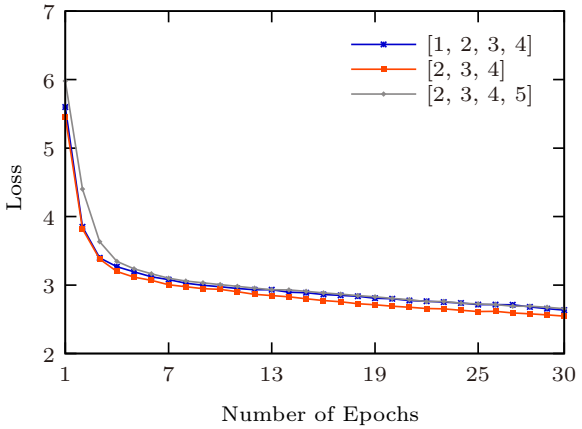


Fig.7. Impact of different numbers of windows on training.

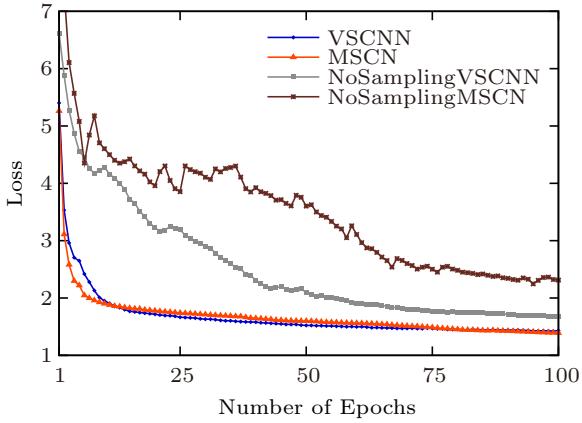


Fig.8. Convergence of the loss on the validation set as the number of epochs grows.

Table 3 shows the value of the median (denoted by mid), percentile, maximum (denoted by max), and the mean (denoted by mean) q-errors of distinct cardinality estimators in the synthetic workload. The lowest q-error values are emphasized in bold. We can see that IBJS achieves the best median estimation. VSCNN wins the other methods in the 90th and 95th percentile of q-errors. Although IBJS performs well on the median q-error, it relies too much on the index structures. If there is no proper index or even no index, its performance will become very poor. In terms of robustness, VSCNN and MSCN outperform IBJS. The mean q-error of VSCNN is 104.4 times less than that of the average q-error on cardinality estimation in traditional databases, and 7.7 times less than that of NoSamplingVSCNN. When compared with other cardinality estimation methods, our model has powerful semantic recognition capability. The SQL contained in the synthetic workload is simple, thereby the result is very good. In particular, VSCNN applying the sampling method can utilize the information of samples in the base table to learn the distribution of records.

Table 4 shows the detail of the median, percentile, maximum, and the mean q-errors in the scale workload. It is straightforward to find that all the q-errors are larger than those on the synthetic workload. This is because the Scale workload contains more SQL joins, thereby the q-errors of all methods increase. We can see that MSCN achieves the best median estimation, and its maximum q-error and mean q-error are the minimum. VSCNN wins other methods in the 90th, 95th and 99th percentile of q-errors. The mean q-error of our model is 11.6 times less than that of the average q-error of cardinality estimation methods in traditional databases, and 3.7 times less than that of NoSamplingVSCNN. The proposed VSCNN model performs well, because we use different convolution kernels to scan the features of words, which can well identify the relationship of different join operations. Similarly, VSCNN with the sampling method can make full use of the information of samples in the base table to learn the the distribution of records.

To show that the proposed model can be generally applied to query workloads which are not generated by the query generator, we use the JOB query workload. Table 5 shows the value of the median, percentile, maximum, and the mean q-errors of distinct cardinality estimators in the JOB workload. In order to validate the generalization capability of processing the queries, we conduct experiments on the job-light workloads that

Table 3. *Q*-Error of Cardinality Estimators on Synthetic Workload

Method	Mid	90th	95th	99th	Max	Mean
PostgreSQL	1.85	10.03	25.10	487.00	399 274	172.00
MySQL	2.27	25.80	55.70	642.00	458 835	367.00
Oracle	2.03	15.10	47.40	497.00	545 925	385.00
SQL Server	2.15	17.40	50.10	296.00	512 591	379.00
BBA	2.33	20.01	49.30	544.00	483 216	303.00
IBJS	**1.11**	10.36	37.10	299.00	293 290	127.00
NoSamplingMSCN	2.39	8.94	14.60	120.00	1 903	25.71
NoSamplingVSCNN	2.16	8.37	13.90	108.00	1 842	24.03
MSCN	1.26	4.03	7.32	**33.67**	**1 545**	**3.08**
VSCNN	1.32	**3.99**	**7.29**	33.95	1 564	3.12

Table 4. *Q*-Error of Cardinality Estimators on Scale Workload

Method	Mid	90th	95th	99th	Max	Mean
PostgreSQL	2.79	259.0	577.0	1 956.0	243 863	581.0
MySQL	3.29	95.4	339.0	7 795.0	56 512	441.0
Oracle	2.71	121.0	491.0	3 601.0	104 912	405.0
SQL Server	2.81	242.0	446.0	3 905.0	64 321	423.0
BBA	2.67	102.4	281.0	1 829.0	58 219	332.0
IBJS	2.54	91.3	252.1	1 711.0	49 306	291.0
NoSamplingMSCN	2.39	97.1	261.0	1 121.0	4 113	137.0
NoSamplingVSCNN	2.31	95.4	254.0	1 107.0	4 098	135.0
MSCN	**1.71**	38.5	146.0	795.0	**3 687**	**36.3**
VSCNN	1.76	**38.1**	**144.8**	**793.2**	3 690	36.5

Table 5. *Q*-Error of Cardinality Estimators on JOB Workload

Method	Mid	90th	95th	99th	Max	Mean
PostgreSQL	8.04	169.0	1 110.0	2 921.0	3 492.0	176.0
MySQL	10.01	307.0	691.0	2 263.0	2 589.0	156.0
Oracle	8.76	379.0	981.0	2 770.0	3 342.0	163.0
SQL Server	9.03	364.0	701.0	2 550.0	3 421.0	171.0
BBA	8.03	157.2	621.1	2 311.0	3 006.0	151.0
IBJS	**1.92**	157.0	3 198.0	14 309.0	15 775.0	590.0
NoSamplingMSCN	5.49	131.0	981.0	1 315.0	2 023.0	104.0
NoSamplingVSCNN	5.41	129.2	976.0	1 301.0	1 999.0	103.0
MSCN	3.91	81.7	364.1	**931.7**	1 119.0	**58.8**
VSCNN	3.82	**79.4**	**362.5**	935.1	**1 118.2**	60.1

contain the queries with a predicate in closed interval, e.g., $1997 < production\_year < 2021$, while the traditional training data only contain the predicates in open interval, e.g., $production\_year > 1997$. According to the experimental results, VSCNN performs the best on the 90th, and the 95th of percentile and the maximum q-error respectively.

In addition, the results show our model can be generalized to the workloads with distributions different from the training data. The mean q-error of the VSCNN model is 2.7 times less than that of the average q-error of cardinality estimation models in traditional databases, and 1.5 times less than that of NoSamplingVSCNN. This can be explained by the reason that the proposed model will learn a lot of SQL features from the training samples. But, the word vector we use is trained in advance, and the value of each field is very sparse in the phase of training. Therefore in terms of the validation samples, it is very likely to encounter the vector that is not in the word vector. We set this kind of word vector to zero vector. Because the proposed model has learned all kinds of SQL structures through the training set, although it is only the deviation from one data field, it has little negative effect on the cardinality estimation of the whole SQL queries.

We train the models on 80 000 queries with multiple joins and take 90% of multi-table join queries as training samples and 10% of them as validation samples. We train the model until the loss converges. The estimation q-errors are shown in Table 6. From the ex-

Table 6. Q-Error of Cardinality Estimators on JOB Workloads with Strings

JOB-String	Mid	90th	95th	99th	Max	Mean
PostgreSQL	184.0	8 303.0	34 203.0	106 000.0	670 000.0	10 416.0
MySQL	104.0	28 157.0	213 471.0	1 630 689.0	2 487 611.0	60 229.0
Oracle	119.0	55 446.0	179 106.0	697 790.0	927 648.0	34 493.0
SQL Server	174.0	60 432.0	231 045.0	552 190.0	432 609.0	52 700.0
MSCN	Null	Null	Null	Null	Null	Null
MSCN(string)	**14.6**	**85.4**	207.3	792.7	851.1	73.5
VSCNN	15.2	89.1	**199.5**	**760.2**	**803.7**	**69.6**

perimental results, traditional database systems tend to overestimate the cardinality, and the q-error is very large, even up to hundreds of thousands. The q-error of learning-based cardinality estimation models can be reduced to at least 14.6% when compared with the estimators in traditional databases. MSCN can handle simple queries, but for complex queries in JOB workloads, including strings, MSCN does not work, thereby the results are represented by "Null" in Table 6. In order to increase the generalization capability of MSCN, that is, the compatibility of handing strings, we extend the value to a multi-dimensional vector representing string. We can find that our model works well to achieve good results in all experiments. MSCN cannot handle the string type field, because it does not have a neural network structure that can recognize strings. MSCN (string) can handle strings, but its q-error is larger than VSCNN. The VSCNN model can handle strings well, because it embeds the string as a word vector by the embedding technique introduced in Subsection 3.2 and it captures the semantic information in SQL statements.

4.6 Prediction Time Comparison

Fig.9 shows the prediction time among different cardinality estimators. We can see that the prediction time (per sample) of the proposed VSCNN model on the validation set is only about 0.2 ms. In Fig.9, we demonstrate the prediction time on the training as well as validation samples, respectively. We find that the cardinality estimation models (i.e., TLSTM [24] and TPOOL [24]) with NLP (natural language processing) have high prediction time cost for training and validation. The proposed tree structure can deal with complex predicates and has powerful functions. The prediction time of TLSTM on the validation set is 3.1 times higher than that of VSCNN, and the prediction time of TPOOL on the validation set is about two times higher than that of VSCNN. The prediction time of VSCNN on the validation set is similar to that of traditional database (i.e., PostgreSQL and MySQL). In general, MSCN has the least prediction time, but the prediction time on the validation set is higher than that on the training set. The prediction time of VSCNN on the validation set is lower than that on the training set. The structures of TLSTM and TPOOL are very complex, which leads to costly prediction time. The method of cardinality estimation applied in traditional databases (i.e., PostgreSQL and MySQL) is very simple, thereby the prediction time is short.

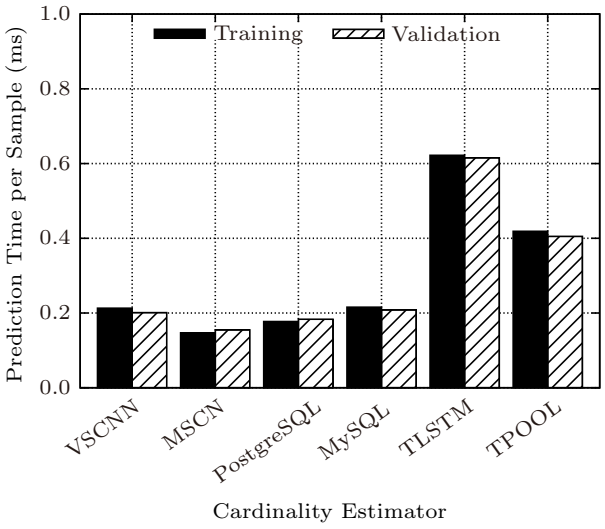


Fig.9. Prediction time comparison of different cardinality estimators.

Fig.10 shows the prediction time of different filters in VSCNN. We specify three categoris of filters: [1, 2, 3], [2, 3, 4] and [3, 4, 5], and we observe that when the filters is [1, 2, 3], the prediction time is the least, and the prediction time of filters [3, 4, 5] is the maximum.

4.7 Data Update

To cope with data and schema changes, we can apply some modifications to our model that supports incremental training, and we can also completely re-train the model. But complete re-training the model will consume a lot of computing resources, because we need to

re-execute SQLs to get the latest cardinality and then re-train the model. For the columns that do not involve the change of data range in the training set, re-training is not necessary. Complete re-training is allowed to use different data encoding methods in these operations, e.g., creating a new table, and then we need to use a long one-hot vector to represent the table name and we could re-normalize values in case of new minimum or maximum values.

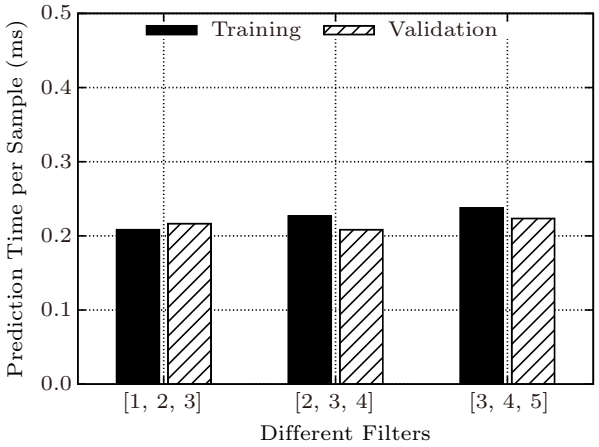


Fig.10. Prediction time comparison of different filters.

In this study, we assume that the database state is stable without changing, that is to say, the cardinality estimation is achieved on the snapshot of the database. When training the data, we learn the data distribution by the proposed sampling method in Subsection 3.1.2. However, the state of the database cannot be constant, and there are operations such as creating new tables, inserting data, deleting data, and updating data, which may make the state of the database no longer constant, resulting in the change of data distribution. At this time, if we work on the trained model directly without considering the change of data distribution, it will cause large loss values. Then, it is straightforward to think of incremental training, which does not require us to re-train all the data in advance. We could re-use the model state and only apply it in the new samples. But this will encounter a problem, how to adapt to the change of data encoding, including one-hot encodings and the normalization of values (i.e., cardinality). For the cardinality, we can specify a threshold, such as the maximum value. No matter whether it is re-training or incremental training, the proposed VSCNN model can adapt the change of data distribution, because there is no memory mechanism like that applied in LSTM (Long Short-Term Memory) networks.

5 Conclusions

In this study, we introduced the vertical scanning convolutional neural network model VSCNN in coping with the cardinality estimation problem. VSCNN can convert SQL queries from a sentence level to a word vector, and then use a vertical scanning convolutional neural network to capture the relationships between words in the word vector. It works very well when dealing with multiple table join queries and can handle the string type of predicates. In particular, we used the proposed sampling method based on the base table and compressed it into a bitmap to improve the accuracy of cardinality estimation. In addition, we conducted extensive experiments on the real IMDB datasets, and the experimental results showed that the estimation quality of q-error of the proposed vertical scanning convolutional neural network-based model is reduced by at least 14.6% when compared with the existing estimators in traditional databases.

In order to apply the proposed model on more types of queries, we will improve it from the following aspect: in terms of the string type of data, although we can handle it, the results are not satisfying. When compared with the numerical type of data, the fields of numeric data have the maximum and the minimum values in the table, and all values are between the maximum and the minimum values. The string type is difficult to process because the numbers of characters in strings are different, and the characters themselves are different. In this way, the string type of data is more sparse than the numeric type. In future, we will design a better method to deal with the string type of data.

References

[1] Leis V, Radke B, Gubichev A, Kemper A, Neumann T. Cardinality estimation done right: Index-based join sampling. In *Proc. the 8th Biennial Conference on Innovative Data Systems Research*, Jan. 2017.

[2] Li G, Zhou X, Li S. XuanYuan: An AI-native database. *IEEE Data Eng. Bull.*, 2019, 42(2): 70-81.

[3] Kipf A, Kipf T, Radke B, Leis V, Boncz P A, Kemper A. Learned cardinalities: Estimating correlated joins with deep learning. In *Proc. the 9th Biennial Conference on Innovative Data Systems Research*, Jan. 2019.

[4] Ioannidis Y E. The history of histograms (abridged). In *Proc. the 29th International Conference on Very Large Data Bases*, Sept. 2003, pp.19-30. DOI: 10.1016/B978-012722442-8/50011-2.

[5] Giroire F. Order statistics and estimating cardinalities of massive data sets. *Discret. Appl. Math.*, 2009, 157(2): 406-427. DOI: 10.1016/j.dam.2008.06.020.

[6] Flajolet P, Martin G N. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 1985, 31(2): 182-209. DOI: 10.1016/0022-0000(85)90041-8.

[7] Durand M, Flajolet P. Loglog counting of large cardinalities. In *Proc. the 11th Annual European Symposium*, Sept. 2003, pp.605-617. DOI: 10.1007/978-3-540-39658-1_55.

[8] Flajolet P, Fusy É, Gandouet O, Meunier F. HyperLogLog: The analysis of a near-optimal cardinality estimation algorithm. In *Proc. the 2007 Conference on Analysis of Algorithm*, Jun. 2007, pp.137-156.

[9] Whang K, Zanden B T V, Taylor H M. A linear-time probabilistic counting algorithm for database applications. *ACM Trans. Database Syst.*, 1990, 15(2): 208-229. DOI: 10.1145/78922.78925.

[10] Wu W, Naughton J F, Singh H. Sampling-based query re-optimization. In *Proc. the 2016 International Conference on Management of Data*, June 26–July 1, 2016, pp.1721-1736. DOI: 10.1145/2882903.2882914.

[11] Lipton R J, Naughton J F, Schneider D A. Practical selectivity estimation through adaptive sampling. In *Proc. the 1990 ACM SIGMOD International Conference on Management of Data*, May 1990, pp.1-11. DOI: 10.1145/93605.93611.

[12] Olken F, Rotem D. Random sampling from database files: A survey. In *Proc. the 5th International Conference on Statistical and Scientific Database Management*, Apr. 1990, pp.92-111. DOI: 10.1007/3-540-52342-1_23.

[13] Estan C, Naughton J F. End-biased samples for join cardinality estimation. In *Proc. the 22nd International Conference on Data Engineering*, Apr. 2006, Article No. 20. DOI: 10.1109/ICDE.2006.61.

[14] Neumann T, Leis V, Kemper A. The complete story of joins (in hyper). In *Proc. the Datenbanksysteme für Business, Technologie und Web*, Mar. 2017, pp.31-50.

[15] Neumann T, Radke B. Adaptive optimization of very large join queries. In *Proc. the 2018 International Conference on Management of Data*, Jun. 2018, pp.677-692. DOI: 10.1145/3183713.3183733.

[16] Zhang W E, Sheng Q Z, Qin Y, Taylor K, Yao L. Learning-based SPARQL query performance modeling and prediction. *World Wide Web*, 2018, 21(4): 1015-1035. DOI: 10.1007/s11280-017-0498-1.

[17] Leis V, Gubichev A, Mirchev A, Boncz P A, Kemper A, Neumann T. How good are query optimizers, really? *Proc. VLDB Endow.*, 2015, 9(3): 204-215. DOI: 10.14778/2850583.2850594.

[18] Lakshmi M S, Zhou S. Selectivity estimation in extensible databases—A neural network approach. In *Proc. the 24th International Conference on Very Large Data Bases*, Aug. 1998, pp.623-627.

[19] Malik T, Burns R C, Chawla N V. A black-box approach to query cardinality estimation. In *Proc. the 3rd Biennial Conference on Innovative Data Systems Research*, Jan. 2007, pp.56-67.

[20] Yang Z, Liang E, Kamsetty A, Wu C, Duan Y, Chen X, Abbeel P, Hellerstein J M, Krishnan S, Stoica I. Selectivity estimation with deep likelihood models. arXiv:1905.04278, 2019. http://arxiv.org/abs/1905.04278, Aug. 2020.

[21] Liu H, Xu M, Yu Z, Corvinelli V, Zuzarte C. Cardinality estimation using neural networks. In *Proc. the 25th Annual International Conference on Computer Science and Software Engineering*, Nov. 2015, pp.53-59.

[22] Knagenhjelm P, Brauer P. Classification of vowels in continuous speech using MLP and a hybrid net. *Speech Commun.*, 1990, 9(1): 31-34. DOI: 10.1016/0167-6393(90)90042-8.

[23] Mahmoud M A B, Guo P. DNA sequence classification based on MLP with PILAE algorithm. *Soft Comput.*, 2021, 25(5): 4003-4014. DOI: 10.1007/s00500-020-05429-y.

[24] Sun J, Li G. An end-to-end learning-based cost estimator. *Proc. VLDB Endow.*, 2019, 13(3): 307-319. DOI: 10.1477-8/3368289.3368296.

[25] Yu X, Li G, Chai C, Tang N. Reinforcement learning with tree-LSTM for join order selection. In *Proc. the 36th IEEE International Conference on Data Engineering*, Apr. 2020, pp.1297-1308. DOI: 10.1109/ICDE48307.2020.00116.

[26] Zhang J, Liu Y, Zhou K, Li G, Xiao Z, Cheng B, Xing J, Wang Y, Cheng T, Liu L, Ran M, Li Z. An end-to-end automatic cloud database tuning system using deep reinforcement learning. In *Proc. the 2019 International Conference on Management of Data*, Jun. 2019, pp.415-432. DOI: 10.1145/3299869.3300085.

[27] Li G, Zhou X, Li S, Gao B. QTune: A query-aware database tuning system with deep reinforcement learning. *Proc. VLDB Endow.*, 2019, 12(12): 2118-2130. DOI: 10.14-778/3352063.3352129.

[28] Li G, Chai C, Fan J, Weng X, Li J, Zheng Y, Li Y, Yu X, Zhang X, Yuan H. CDB: Optimizing queries with crowd-based selections and joins. In *Proc. the 2017 ACM International Conference on Management of Data*, May 2017, pp.1463-1478. DOI: 10.1145/3035918.3064036.

[29] Fan J, Li G, Zhou L. Interactive SQL query suggestion: Making databases user-friendly. In *Proc. the 27th International Conference on Data Engineering*, Apr. 2011, pp.351-362. DOI: 10.1109/ICDE.2011.5767843.

[30] Mikolov T, Sutskever I, Chen K, Corrado G S, Dean J. Distributed representations of words and phrases and their compositionality. In *Proc. the 27th International Conference on Neural Information Processing Systems*, Dec. 2013, pp.3111-3119.

[31] Zimmer R, Pellegrini T, Singh S F, Masquelier T. Supervised training of convolutional spiking neural networks with PyTorch. arXiv:1911.10124, 2019. https://arxiv.org/abs/1911.10124, Nov. 2020.

[32] Al-Mouhamed M A, Hasan Khan A, Mohammad N. A review of CUDA optimization techniques and tools for structured grid computing. *Computing*, 2020, 102(4): 977-1003. DOI: 10.1007/s00607-019-00744-1.

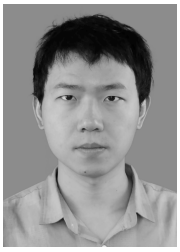
[33] Liu B, Liang Y. Optimal function approximation with ReLU neural networks. *Neurocomputing*, 2021, 435: 216-227. DOI: 10.1016/j.neucom.2021.01.007.

[34] Hinton G E, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov R. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580, 2012. https://arxiv.org/abs/1207.0580, May 2021.

[35] Kingma D P, Ba J. Adam: A method for stochastic optimization. In *Proc. the 3rd International Conference on Learning Representations*, May 2015.

[36] Moerkotte G, Neumann T, Steidl G. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proc. VLDB Endow.*, 2009, 2(1): 982-993. DOI: 10.1477-8/1687627.1687738.

Shao-Jie Qiao received his B.S. and Ph.D. degrees in computer application technology from Sichuan University, Chengdu, in 2004 and 2009, respectively. From 2007 to 2008, he worked as a visiting scholar in the School of Computing at the National University of Singapore, Singapore. He is a distinguished young scholar of Sichuan Province of China. He is currently a professor with the School of Software Engineering, Chengdu University of Information Technology, Chengdu. He has authored more than 150 high quality papers, including IEEE Transactions on ITS, IEEE TKDE, IEEE TNNLS, IEEE TCSVT, IEEE Transactions on SMC: Systems, ACM TKDD, ACM TIST, etc.. His research interests include databases, data mining and artificial intelligence. He is a senior member of CCF.



Guo-Ping Yang is currently a Master student in the School of Software Engineering, Chengdu University of Information Technology, Chengdu. His research interests include artificial intelligence for databases.

Nan Han received her M.S. and Ph.D. degrees in prescription of traditional Chinese medicine from Chengdu University of Traditional Chinese Medicine, Chengdu, in 2009 and 2012, respectively. She is currently an associate professor with the School of Management, Chengdu University of Information Technology, Chengdu. Her research interests include data mining and artificial intelligence. She is the author of more than 50 papers and she has participated in several projects supported by the National Natural Science Foundation of China.

Hao Chen received his B.S. degree in electronic information engineering from the Sichuan University, Chengdu, in 2013. He used to work in Alibaba and Baidu. He is currently an expert with the Database System Engineering in Huawei's Gauss Laboratory. His research interests include database optimizer and executor.

Fa-Liang Huang received his Ph.D. degree in computer science from the South China University of Technology, Guangzhou, in 2011. He is an associate professor with the School of Computer and Information Engineering, Nanning Normal University, Nanning. His research interests include data mining and deep learning. He is a member of CCF.

Kun Yue received his M.S. degree from Fudan University, Shanghai, in 2004, and his Ph.D. degree from Yunnan University, Kunming, in 2009, both in computer science. He is currently a professor and Ph.D. supervisor at School of Information Science and Engineering, Yunnan University, Kunming. He has authored more than 80 papers in peer-reviewed journals and conferences, such as IEEE Trans. Cybernetics, IEEE Trans. Service Computing, Information Sciences, Neurocomputing, Applied Soft Computing, and DASFAA. His current research interests include massive data analysis and uncertainty in artificial intelligence. He is a member of CCF.

Yu-Gen Yi received his Ph.D. degree in applied mathematics from Northeast Normal University, Changchun, in 2016. He is an associate professor with the School of Software, Jiangxi Normal University, Nanchang. His research interests include artificial intelligence, computer vision, and machine learning. He is a member of CCF and IEEE.

Chang-An Yuan received his Ph.D. degree in computer science from Sichuan University, Chengdu, in 2006. He is a professor at Guangxi College of Education and Nanning Normal University, Nanning. His research interests include computational intelligence and data mining. He is a member of CCF.