

# KELM: Knowledge-Enhanced Learning Methodology for Cardinality Estimation

Yu Yan (✉ [yuyan@hit.edu.cn](mailto:yuyan@hit.edu.cn))

Hongzhi Wang (✉ [wangzh@hit.edu.cn](mailto:wangzh@hit.edu.cn))

Jian Geng (✉ [1161906117@qq.com](mailto:1161906117@qq.com))

Jian Ma (✉ [21S103266@stu.hit.edu.cn](mailto:21S103266@stu.hit.edu.cn))

Yuzhuo Wang (✉ [wangyuzh@hit.edu.cn](mailto:wangyuzh@hit.edu.cn))

---

## Research Article

## Keywords:

DOI: <https://doi.org/>

**License:** © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

---

# KELM: Knowledge-Enhanced Learning Methodology for Cardinality Estimation

Yu Yan<sup>a</sup>, Hongzhi Wang<sup>a</sup>, Jian Geng<sup>a</sup>, Jian Ma<sup>a</sup>, Yuzhuo Wang<sup>a</sup>

<sup>a</sup>*Harbin Institute of Technology, Harbin, China*

---

## Abstract

Recent years, the database community has attempted to develop learned cardinality estimator for improving the estimation accuracy. Although some researches show that the applying deep learning to cardinality estimation is a significant and promising direction, there still exists many problems in implementing these techniques to real applications (**long preparation time, unstable performance and poor model transition**). We comprehensively analyze the characteristics of statistical estimators which have the potential ability to enhance the learned estimators for solving these problems. After carefully design, we propose a knowledge-enhanced learning methodology for cardinality estimation, called KELM. To our knowledge, KELM is the first work to regard the statistical estimators as knowledge for enhancing the learned estimators. Specifically, we propose a knowledge enhanced preparation phase which could accelerate the model pretraining by reducing queries execution. For online phase, we design stable model inference algorithm and a two-stage transition method for efficiently adapting to dynamic workloads. Experimental results in four open source datasets demonstrate that KELM outperforms existing works in model preparation, model inference and model transition. Our work brings a new development direction for the application of learned estimators in real database environments.

---

**keywords:** Cardinality Estimation, Deep Learning, Database, Knowledge

## 1. Introduction

The cardinality estimation is an essential and important research for many database management services, like cost estimation [27, 26], query optimization [13, 11], index selection [24], etc. Typically, researchers propose many methods [4] to estimate the cardinality, such as the Equi-Height Histograms [3], Conditional Bayes Model [22], etc. Unfortunately, these statistic-based methods perform badly in complex workloads due to their simple structure [8].

In order to improve the efficiency of cardinality estimation, researchers attempt to utilize the powerful learning ability of neural network to learn the relationships between original data and cardinalities. For example, the Naru [27] proposes a deep regression model for cardinality estimation. Many experimental results [21, 23] have shown that learned estimators have great improvements for the accuracy of the cardinality estimation.

However, some important problems influence learned estimators to apply in real database environment, i.e. (i) **Unstable Performace** Existing learned estimators are based on the black box, neural network model. Thus, they may perform badly in irregular data distributions resulting in inefficient workload execution [13]. (ii) **Long Preparation Time** Learned estimators require a large amount of data to train before model online. Especially, complex situations are too time-consuming to afford for model pretraining. (iii) **Poor Model Transition** Changes in the environment (like data, workload and database) will cause the learned estimator to fail. It takes too long time for learned estimators to transfer the trained model from one case to another because of multiple retraining [23].

Generally speaking, these problems are caused by the complex structure of the neural network. Neurons sacrifice training time and transition cost to provide better accuracy. While a core observation is that statistical estimators do not face these problems due to its math structure, like the LogLog [2] and Min-Count [16]. A natural question is that if we can enhance the learned estimators by integrating the statistical estimators? To answer this question, we firstly

analyze some characteristics of statistical estimators.

(i) Fast preparation. The statistical estimators complete the cardinality estimation by gathering some statistical values from the database [3]. Therefore,  
35 they could quickly complete the estimator preparation.

(ii) Stability. Even though the statistical estimators have weaker learning ability than the learned estimators. Due to their statistical basis, statistical estimators have a stable error bound which guarantees the error of the worst-case, like the MinCount [16].

40 (iii) Easily Transition. Like the preparation phase of statistical estimators, the transition propose could also complete with gathering some statistical values [23]. The transition cost of statistical estimators has no influence on the online workload execution.

Inspired by the above characteristics of statistical estimators, we firstly pro-  
45 posed a novel learning methodology for cardinality estimation. Our key idea is to design a knowledge-enhanced learning which could combine the statistical estimators and learned estimators to overcome the shortcomings of them, such as low accuracy of statistical estimators, long training time of learned estimators, etc. Specifically, the main contributions are as follows:

- 50 • Our work is the first one to propose a knowledge-enhanced learning methodology for cardinality estimation, which could make full use of the advantages of two kinds of cardinality estimators.
- We carefully design a knowledge enhanced preparation phase by integrating the statistical estimators, which could save time in query execution to  
55 accelerate the training process of learned estimators.
- We guarantee an error bound for learned estimators by limiting the confidence between statistical estimators and learned estimators.
- We propose two-stage transition method which utilizes the statistical method as transition to quickly adapt to dynamic workloads.

- 60 • Extensive experimental results in four open source datasets show that our KELM outperforms existing works in preparation speed, stability and transition efficiency.

Supplementally, our paper clarifies a novel development direction for learned solutions, which claim that statistical methods could be regarded as knowledge  
65 to enhance the learned estimators. We hope that our paper could spark more promising practical results.

## 2. Related work

Because of the powerful learning ability of neural network [12, 9], database community has proposed many learning methods for database management,  
70 like learned index [15, 6]. Cardinality estimation is an essential task of database management system for which researchers has proposed many methods [10, 17, 27, 26, 21]. Existing methods could be generally divided into two main classes, statistics-based and learning-based methods. Next, we introduce some popular works for above two classes separately.

### 75 2.1. *Statistic-based Estimators*

Statistic-based methods [10, 19] utilize the statistic theory to estimate the cardinalities. For example, Lipton et al. [10] propose an adaptive sampling cardinality estimator that uses partial samples to estimate the overall. And with the partial independence assumption, Deshpande et al. [3] propose a histogram  
80 model to the store data distribution for cardinality estimation. Tzoumas et al. [22] design a probabilistic relational model which relies on the Bayes theory to estimate the conditional probabilities of relational tables. We can observe that these statistic-based estimators rely on probability theory rather than complex models. As a result, they cost less resources on model building and maintaining,  
85 but are less capable of handling complex workloads. In this paper, we consider to construct a learned model to improve the accuracy of cardinality estimation with limited resources.

## 2.2. Learning-based Estimators

With the powerful learning ability of the neural network, many researchers  
90 have attempted to utilize the deep learning to estimate the cardinality for higher  
accuracy. For example, MSCN [8] encodes the queries and tables as input and  
establishes an deep learning model to estimate the cardinality. With the theory  
of the conditional density, Naru [27] proposes a deep autoregressive model to  
achieve cardinality estimation for single table. Zhao et al. design a lightweight  
95 deep model [7] for the cardinality estimation in approximate complex work-  
loads. In general, the utilization of deep learning did improve the accuracy of  
cardinality estimation, but it also brought many key problems, such as unstable  
performance, long training time and clumsy model transition. In this paper, dif-  
ferent from existing learned estimators, we propose a novel learning structure,  
100 which integrates the statistical estimators to solve the above problems. Our  
method supports potential capabilities for applying learning-based estimators  
to database in real environments.

## 3. Motivations

As shown in Section 1, existing learned cardinality estimators [8, 13] have  
105 some significant shortcomings in applying to real applications. Surprisingly,  
traditional cardinality estimators (like histogram-based methods, sample-based  
methods, etc.) which have simple and general structure provide some poten-  
tial opportunities to enhance the learned cardinality estimators. In this paper,  
our core sight is to utilize the strengths of traditional cardinality estimators to  
110 enhance the weaknesses of learned cardinality estimators. Detailed discussions  
are as follows:

(i) Stable Performance. Traditional cardinality estimators have solid statistic  
theory basic which guarantees the statistical bound for cardinality estimation.  
But learned estimators [13] is a black-box and may get poor performance in some  
115 special data distribution [13]. In this paper, we firstly consider to construct a

theoretical bound for learned cardinality estimators by integrating statistical estimators.

(ii) Quickly Preparation. Due to the simple principles [23], traditional cardinality estimators do not need large time consumption in model preparation. For example, the estimator in Postgres<sup>1</sup> could start in several seconds. Conversely, the learned cardinality estimators need a large amount of sample data to train the model which costs large time resource in query execution. In this paper, we consider to combine the statistical estimators and learned estimators to achieve the quickly pre-training for learned estimators.

(iii) Efficiently Model Transition. Traditional cardinality estimators could efficiently transfer to dynamic workload due to its universal structure [18]. For example, QuickSel [18] models the data distribution by uniform mixture model which has good adaptability. However, it is very costly for the well trained learned cardinality estimators [27] to achieve model transition in changed environment. In order to achieve efficiently model transition for dynamic workload, we consider to utilize the traditional cardinality estimators as the transition to help learned estimators.

Based on the above three motivates, we propose an knowledge-enhanced learned cardinality estimator called KELM. Our goal is to improve the shortcomings of learned estimators by utilizing the strengths of statistical estimators. In this next section, we will show the methodologies of KELM in detail.

#### 4. Methodologies

In this section, we introduce the basic methodologies of our KELM and clarify how KELM integrates the traditional methods as knowledge. Figure 1 demonstrates the workflow of KELM. Similar to other learned cardinality estimator, our KELM also has two main phases: Preparation Phase, Online Phase. The preparation phase is in charge of model preparation. The online phase is re-

---

<sup>1</sup><https://www.postgresql.org>

sponsible for cardinality inference and model transition. Next, we will introduce the two phases separately in Section 4.1 4.2.

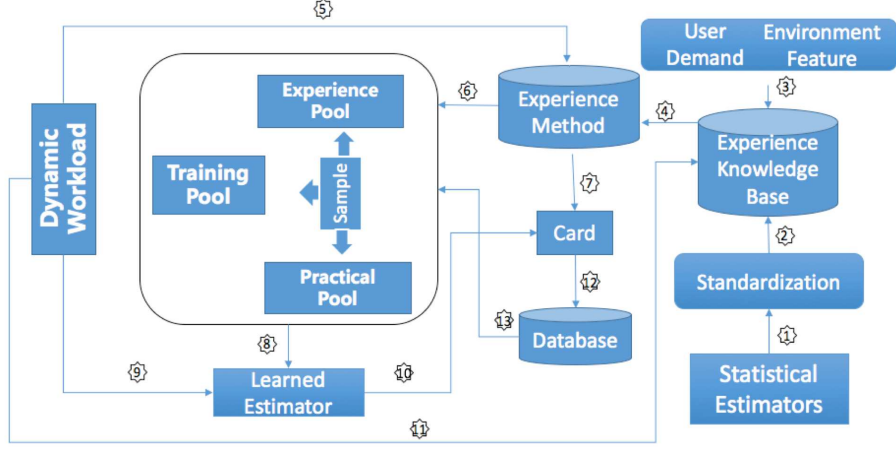


Figure 1: The Architecture of Knowledge-Enhanced Learned Cardinality Estimator

#### 4.1. Preparation Phase

In this section, we present the process of model preparation. The deep learning models require well training before they could work for cardinality estimation [13]. The key sight of model preparation is to obtain enough training dataset. Due to the complexity of database system, it is very time consuming to directly obtain samples by execution queries.

Different from existing works [23] which directly gather the training set from database, we regard the statistical estimators as a knowledge base and directly gather training data from the knowledge rather than real query executions. Thus, our method could avoid large time consumption in the preparation phase. As shown in Figure 1, the preparation of KELM consists of steps 1-2-3-4-5-6-8. Our preparation contains three main stages: experience knowledge base preparation (steps 1-2), experience method preparation (steps 3-4) and model pretraining (steps 5-6-8).

Firstly, we introduce a key component of our method, the experience knowledge base (EKB) which consists of the standard statistical estimators, like His-

togram [3], MinCount [16], etc. Specifically, the EKB could be standardized as a triple  $(\mathcal{F}(\text{Method}), \mathcal{H}(\text{Task}, \text{Env}), \mathcal{K})$ .  $\mathcal{F}$  is a function to standardize these traditional methods to a unified expression  $k_i$ .  $\mathcal{H}$  is function to find the experience method which is suitable for current database.  $\mathcal{K} = \{k_1, k_2, \dots\}$  is the set of standard statistical methods, called experience methods. The schema of the experience methods could be formally defined as  $k_i = (\text{input}, \text{output}, \text{parameter}, \text{feature})$ . For example, MHist [19] which establishes multi-dimensional histogram could be standardized to  $k = ([\text{table}, \text{workload}], \text{cardinality}, [\text{samplenum}, \text{updateinterval}], [\text{dependence}, \text{pretrain}])$ .

In steps 1-2, KELM firstly completes the preparation of experience knowledge base. Each statistical method selected for the current database could be standardized as  $k = (\text{input}, \text{output}, \text{parameter}, \text{feature})$  and stored in EKB. After the preparation of EKB, steps 3-4 complete the preparation of the experience method. In step3, KELM vectorize the user demand and environment as  $v$ . In step 4, KELM return the suitable experience method  $e$  which is most close to  $v$ . For example, the vector  $v$  of index selection  $[\text{cost}, \text{multi}, \text{relational}]$  is closer to  $k_{\text{index}}$  than  $[\text{time}, \text{multi}, \text{graph}]$ .

Finally, in steps 5-6-8, KELM completes the learned model preparation. In steps 5-6 KELM could quickly gather a large amount of training data by utilizing the experience method instead of practical query execution. And KELM store these training data to training pool for model training. In step 8, KELM utilizes the training pool to complete the model retraining. To sum up, our KELM constructs an EKB which standardizes some statistical estimators for model preparation. Compared with the traditional learned estimators [27], KELM could quickly gathers enough training data in any situation with no query execution to support the quick model preparation.

#### 4.2. Online Phase

In this section, we introduce the online phase of KELM in detail. Considering the dynamic workloads, our KELM has two important processes in online phase. One is to implement the cardinality inference in learned estimators. The

other is to efficiently achieve model transition for dynamic workloads. Next, we introduce the two processes respectively.

#### 4.2.1. Model Inference

Due to the black-box nature, the learned cardinality estimators [27, 26, 21] may obtain unstable cardinality in different data distributions. However, the DBMSs require high availability. The unstable performance may cause vital influence for real applications.

To ensure the stability of cardinality estimation, we design novel inference process which has two paths to execute the cardinality estimation. Our key idea is to construct the credibility between experience method and learned model. Then, we utilize the theoretical bound of experience method to construct a theoretical bound for learned cardinality estimators. In the remaining of this section, we firstly introduce the detailed process of cardinality inference. The specific theoretical illustration is shown in Section 6.

As we can observe in Figure 1, KELM has two paths to obtain the cardinalities for workloads. One is the ‘Learned Estimator’ path, steps 9-10. In step 9, KELM delivers a estimation request to the ‘Learned Estimator’. Then, step 10 delivers the result of learned estimator to the ‘Card’. Similar to the ‘Learned Estimator’ path, the other is ‘Experience Method’ path, steps 5-7. The ‘Experience Method’ path utilizes the learned estimator to gather the cardinality and also delivers the result of cardinality to the ‘Card’.

After obtaining two cardinality results,  $C_R$  from ‘Experience Method’ path and  $C_L$  from ‘Learned Estimator’ path, KELM counts a credibility between  $C_R$  and  $C_L$  which represents the applicability of  $C_L$ . According to the credibility, ‘Card’ chooses to utilize  $C_L$  or  $C_R$  for database execution in step 12. Finally, KELM delivers the execution log to ‘Practical Pool’ in steps 13. The unique data in ‘Practical Pool’ will be used to model re-training for improving the accuracy of learned estimators.

To conclude, KELM guarantees the error bound for learned estimators by limiting the confidence between  $C_L$  and  $C_R$ . Our KELM is the first work to

clarify the theoretical bound of learned estimators. More detailed theoretical proof is presented in Section 6.

#### 4.2.2. Model Transition

In this section, we introduce the transfer process of KELM. Model transition  
225 is a significant part for learned estimators to adapt to the dynamic workload [17]. Before they finish the transfer, estimators could not perform well in changed workload. However, a comprehensive transfer of learned cardinality estimators needs consume a lot of time [26] which database could not afford.

In order to improve the efficiency of model transition, we integrate the ex-  
230 perience method as a transition to help learned estimators. The key idea is that KELM do not directly transfer the learned estimator for the changed workload and instead firstly transfer the experience method. Then, KELM could quickly work well in changed workload by utilizing the experience method. Next, KELM utilizes experience method as a transition to gradually achieve the learned es-  
235 timator transfer. To sum up, the transfer of KELM consists of two processes. One is the experience method transition. The other is learned estimator transfer. Next, we express the specific steps of the two processes.

After reaching the condition of transfer, KELM implements steps 3-4 to  
complete the experience method transfer as shown in Figure 1. Step 3 regathers  
240 the current feature which consists of user demand and environment feature. After retrieving the experience knowledge base, step 4 refresh the experience method. Then the experience method completes the adaptability for current situation. Based on the transfer of experience method, KELM starts to model transition. As shown in Figure 1, KELM firstly utilizes step 5-6 and steps 12-13  
245 to gradually update the ‘Experience Pool’ and ‘Practical Pool’. Then, step 8 implements the model retraining to achieve model transition by utilizing the training data sampled from ‘Experience Pool’ and ‘Practical Pool’.

Overall, KELM could efficiently adapt to dynamic workloads by utilizing  
two-stage transfer. In the first stage, KELM quickly updates the experience  
250 methods, which only requires once retrieving from experience knowledge base.

In the second stage, based on the updated experience KELM gradually updates the training pool and implements model retraining. We utilize the training set to improve the ability of the learned estimator. After some model retraining, the learned model could gradually adapt to changed workloads.

255     **Futher Discussion** So far, the database management systems (like MySQL <sup>2</sup>) have not been able to abandon statistical cardinality estimation methods, because of the fast preparation, solid error bound and quick transition. In this paper, we propose a novel learning methodology for cardinality estimation, which regards the traditional estimators as knowledge to enhance the learned estimators. Our methodology could solve the key problems of learned estimators and maintain high estimation accuracy. We hope that our work can promote the application of learning estimators in real database environments.

## 5. Training Algorithm

In this section, we present the training algorithm for KELM. Different from existing cardinality estimators [27, 26], KELM combines the learned cardinality estimators and the traditional cardinality estimators. Thus, the general training algorithms are no longer applicable to our approach. In this paper, we design a formal algorithm which integrates the traditional methods to train the KELM. Next, we present the specific introduction of our algorithm.

270     As shown in Algorithm 1, the input of this algorithm consists of seven parameters.  $W$  is the real-time workload.  $T$  is the list of given traditional methods.  $M$  is the learned model for cardinality estimator. *Sample* is an algorithm for sampling training data.  $I$  is the interval of re-training which could balance the adjustment cost and performance improvement.  $d$  is the bound of credibility which guarantees the theoretical bound of the learned estimator. *Env* is the current environment for obtaining the features of the environment. This algorithm runs along the entire life cycle of the system for ensuring the high suitability of

---

<sup>2</sup><https://www.mysql.com/>

---

**Algorithm 1:** The Training Algorithm of KELM

---

**input:**  $W$  is the real-time workload

$T$  is the list of traditional methods

$M$  is the learned model

$Sample$  is the sampling algorithm

$I$  is the interval of re-training

$d$  is the bound of credibility

$Env$  is the current environment

```
1   $EP, PP \leftarrow \emptyset, \emptyset$  // initialize the experience pool and practical pool
2   $EKB \leftarrow Standardize(T)$  // construct the experience knowledge base
3   $v \leftarrow Env.gather()$  // gather the environment features
4   $E \leftarrow EKB.retrieve(v)$  // retrieve the most proper experience method
5   $M \leftarrow M.pretrain(E, W)$  // implement pretraining
6  while true do
7       $q = W.pop()$ 
8       $C_R \leftarrow E(q)$ 
9       $EP \leftarrow EP \cup \{q, C_R\}$  // store the result of E to experience pool
10      $C_L \leftarrow M(q)$ 
11      $c \leftarrow \frac{|C_L - C_R|}{C_R}$  // calculate the credibility
12     if  $c \leq d$  then
13          $Execute(q, C_L)$  // execute t with learning model
14     else
15          $Execute(q, C_R)$  // execute t with rule-based methods
16     end
17      $PP \leftarrow PP \cup gatherLable()$  // obtain real label from execution log
18     if reach I then
19          $v \leftarrow Env.gather()$  // gather the environment features
20          $E \leftarrow EKB.retrieve(v)$  // update the experience method
21          $M.retrain(Sample, EP, PP)$  // implement model retraining
22     end
23 end
```

---

real environment.

Specifically, Line 1 initializes the experience pool and practical pool. Line 2 standardize the traditional methods and construct the experience knowledge base (EKB). Line 3 gathers the user demand and environment features which is used to retrieve the proper experience method. Line 4 returns the most proper experience method for  $v$ . After determining the experience method, Line 5 utilizes the real-time workload and  $E$  to complete model pretraining. Our KELM could quickly complete pretraining because of using the  $E$  instead of practical execution.

Next, Lines 6-23 present the algorithm of online phase. With the dynamic workloads, KELM repeatedly execution the following steps. Lines 8-10 calculate two cardinality results from the experience method and the learned model. Then Line 11 counts the credibility of  $C_L$ . If the learned estimator is credible, KELM implements  $q$  with  $C_L$ . Otherwise, KELM utilize the  $C_R$ . Then, Line 17 gathers the execution log to ‘Practical Pool’ which is used to model retraining. Moreover, every once in a while, KELM will detect the suitability of the experience method and retrains learned model. Lines 18-22 present the update of the experience method and the model retraining.

To sum up, our approach utilizes the strengths of experience methods to enhance the learned methods. KELM could quickly complete model preparation and have good adaptability to dynamic workloads.

## 6. Theoretical Guarantee for KELM

Most learned methods do not have solid theoretical guarantee which may have problems to apply them in high available database system [28]. In this section, we theoretically clarify that our KELM has a theoretical bound which is significant for high available database system. The basic idea is that if the credibility between the result of experience methods  $C_R$  and the result learned method  $C_L$  has a bound, and  $C_R$  has some performance guarantee  $\epsilon$ , the performance guarantee of  $C_L$  could be derived as  $\epsilon$  and the credibility bound.

We define the solution of learned methods  $C_L$ , the solution of rule-based methods  $C_R$  and the optimal solution  $C_*$ . And the solution of experience rule-based method has a theoretical bound  $\varepsilon(n) \frac{|C_R - C_*|}{C_*}$ . The credibility between  $C_L$  and  $C_R$  is  $d = \frac{|C_L - C_R|}{C_R}$ . Then the performance guarantee of  $C_L$  could be estimated according to the following theorem.

Then we have a theorem as follows:

**Theorem 6.1.** *If  $\forall task, \varepsilon(n) \leq \epsilon$  &  $d$  is a small constant, then  $\frac{|C_L - C_*|}{C_*} \leq d(1 + \epsilon) + \epsilon$*

*proof:* We first consider the minimization scene. The bound of our methodology could be defined as follows:

$$\begin{aligned}
\frac{|C_L - C_*|}{C_*} &= \frac{C_L - C_*}{C_*} \\
&= \frac{(C_L - C_R) + (C_R - C_*)}{C_*} \\
&= \frac{C_L - C_R}{C_R - C_*} \times \frac{C_R - C_*}{C_*} + \frac{C_R - C_*}{C_*} \\
as \quad C_* &= \frac{C_R}{1 + \varepsilon(n)} \\
\frac{|C_L - C_*|}{C_*} &= \varepsilon(n) \times \left( \frac{C_L - C_R}{C_R} \times \frac{1 + \varepsilon(n)}{\varepsilon(n)} + 1 \right) \\
&\leq \epsilon \times \left( d \times \frac{1 + \epsilon}{\epsilon} + 1 \right) \\
&= d(1 + \epsilon) + \epsilon
\end{aligned} \tag{1}$$

Similarly, there also exists a bound for the maximization scene. Thus, we prove that our KELM has solid theoretical basic with the condition  $d$ .  $\square$

## 7. EXPERIMENT

In this paper, we propose a methodology called KELM which enhance the learned cardinality estimators by utilizing statistical estimators. For evaluation, we firstly introduce some specific algorithm settings for KELM in Section 7.1, containing the experience knowledge base, the learned estimator and the sampling algorithm. Then, we introduce the experimental setup, containing

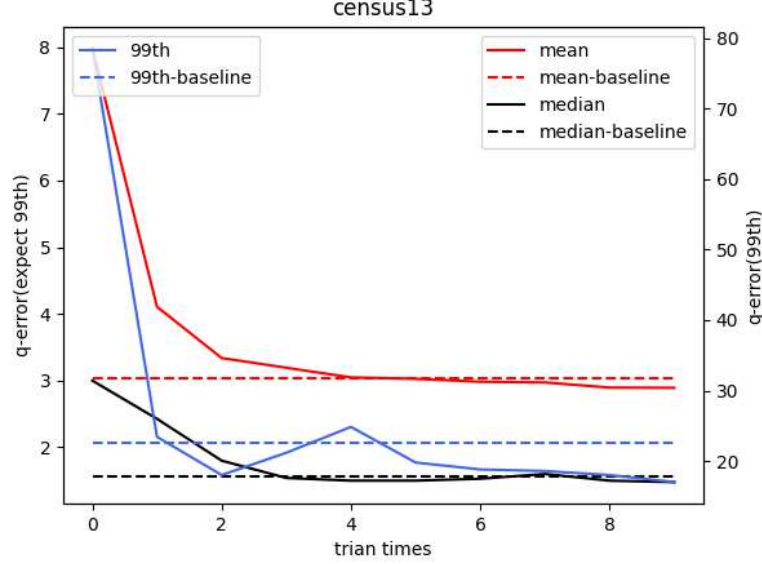


Figure 2: The Q-errors of MSCN (dotted line) and the knowledge enhanced MSCN (solid line) with the times of re-training in census13.

the hardware settings, dataset and metric in Section 7.2. Next, we compare the training efficiency of our KELM with existing learned estimator [8] in Section 7.3. Then, we evaluate the estimation accuracy of KELM of online phase in Section 7.4. Lastly, we test the robustness of the changes of re-training interval in section 7.5.

### 7.1. ELCE Setup

In this section, we firstly introduce the configurations of our experiments. (i) For experience knowledge base, we choose three kinds of universal cardinality estimators, containing the cardinality estimator in *Postgres*<sup>3</sup>, the QuickSel estimator [18] and the Bayes estimator [1]. (ii) For learned model, we use a popular existing work MSCN [8] which is carefully designed for the cardinality estimation. (iii) For the sampling method, we utilize the simple random

<sup>3</sup><https://www.postgresql.org>

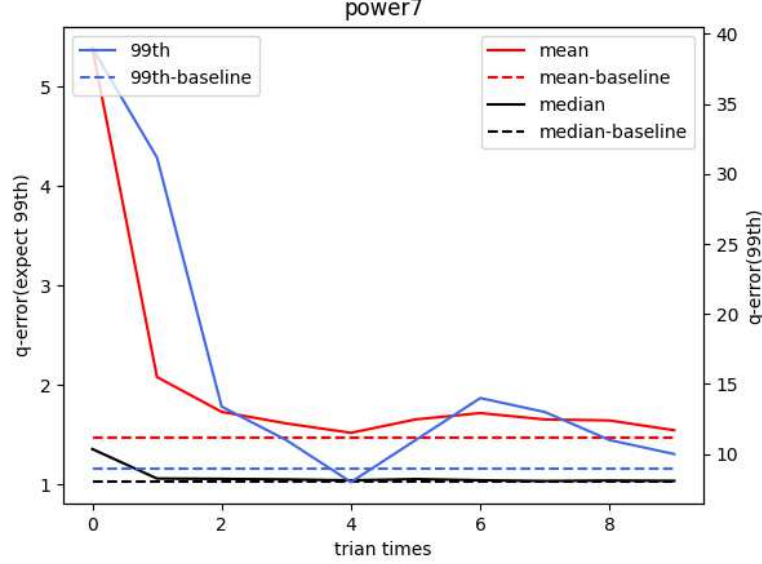


Figure 3: The Q-errors of MSCN (dotted line) and the knowledge enhanced MSCN (solid line) with the times of re-training in power7.

approach from the ‘Practical Pool’.

## 7.2. Environment Setup

340 **Hardware and Platform:** Our experiments conduct in Windows 10 system by using NVIDIA GeForce RTX 2060 GPU, 32 GB memory and Intel Core i7-10857H CPU.

345 **Dataset:** We employ the benchmark generated in the survey [23], contains four real-world dataset, Census [5], Forest [5], Power [5] and DMV [14]. These datasets have difference size (4.8MB, 44.3MB, 110.8MB, 972.8MB) and distribution in columns and workloads. Also, these datasets are used to test at least one existing learned cardinality method [23]. In our experiment, we choose 20K queries as the preparation training samples and use the remaining queries as the execution query queue of testing online phase.

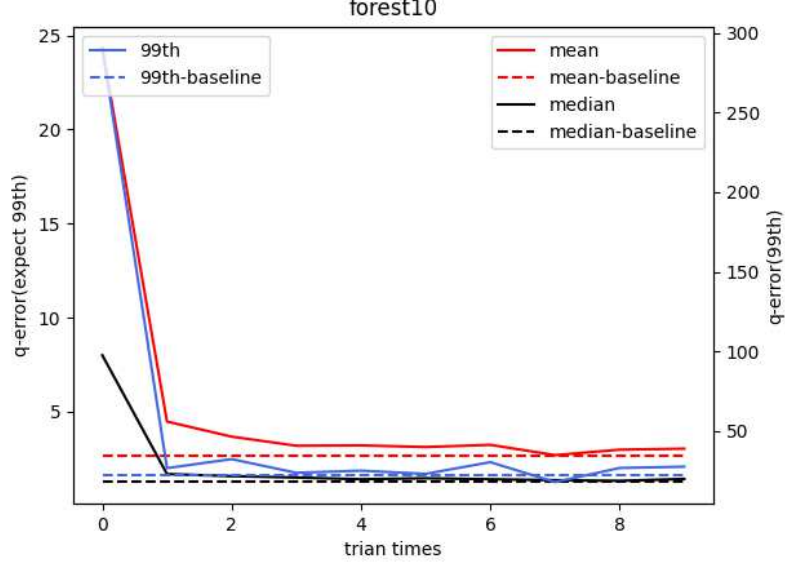


Figure 4: The Q-errors of MSCN (dotted line) and the knowledge enhanced MSCN (solid line) with the times of re-training in forest10.

**Metric:** In order to avoid the influence of the scale of queries, we choose the widely used metric, q-error [27, 26, 21] to measure our methods. Q-error is a ratio between the max and the min in estimation cardinality and actual cardinality. The formal formula is as follows:

$$q - error = \frac{\max\{estimation(q), actual(q)\}}{\min\{estimation(q), actual(q)\}} \quad (2)$$

### 350 7.3. Time Efficiency

Firstly, we evaluate the time consumption of preparation phase between KELM and general learned estimator [8]. As shown in Table 1, we test the time consumption of training process in four datasets. We can find that our KELM which integrates the statistical method to obtain labels only costs some seconds  
355 to get these labels, such as census13 3.4s, forest10:3.49, power7: 3.95s and dmv11:4.20s. While the general learned model (MSCN) which need to obtain labels from query execution takes more than 8 hours to gather cardinality labels

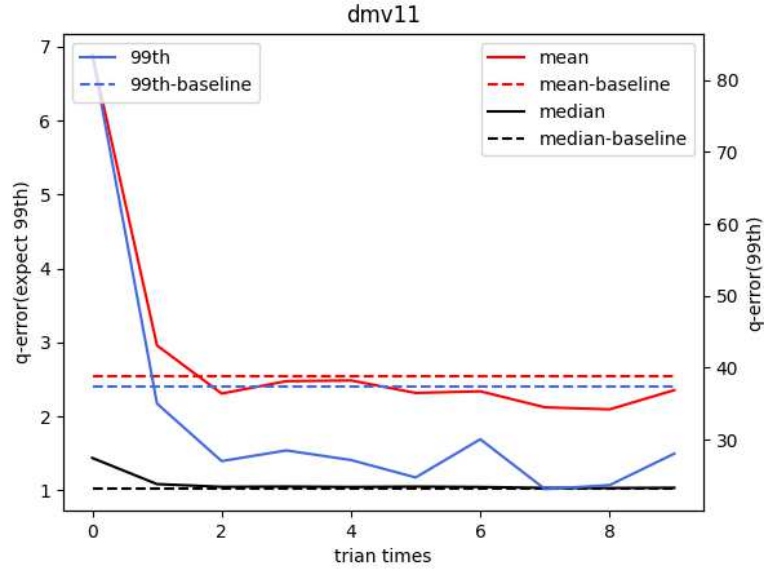


Figure 5: The Q-errors of MSCN (dotted line) and the knowledge enhanced MSCN (solid line) with the times of re-training in dmv11.

in all dataset. Experimental results clarify that KELM could largely reduce the time consumption in model pre-training. Moreover, if we need to implement  
 360 the learned model to real-world applications, the KELM could quickly establish a usable model by embedding the statistical methods.

| Dataset  | KELM model (second) | general learned model |
|----------|---------------------|-----------------------|
| census13 | 3.4048              | BAN                   |
| power7   | 3.9522              | BAN                   |
| forest10 | 3.4965              | BAN                   |
| dmv11    | 4.1986              | BAN                   |

Table 1: The time consumption in preparation phase (BAN presents that model could not obtain the labels within 8 hours).

#### 7.4. Q-error

In this section, we evaluate the Q-error changes of KELM in online phase. For fair comparison, in all dataset, we implement the model re-training with  
365 interval = 10K queries. Because the performance of KELM will change with the model re-training, we choose the re-training times as the abscissa. After every re-training, we evaluate the Q-error of KELM with real-time 2K queries.

Figure 2 3 4 5 show the Q-error changes in four datasets and compare KELM with the MSCN [8]. The dotted lines represent the performance of general  
370 MSCN. And the solid lines represent the performance of KELM based MSCN. We observe that after the preparation phase, the performance of KELM based MSCN is relatively poor on all indicators (99th, median, mean). That's because KELM utilize the experience method as knowledge to implement pretraining. Then, with the implementation of re-training, learned model start to learn the  
375 practical execution cardinalities from the database log. Thus, in all dataset, after two re-training, KELM based MSCN start to steadily tend to the general MSCN. Specifically, in dmv11, our method outperforms the general MSCN on the 99th after the one re-training. That's because that our model could adapt to current workload by using the training data from the 'Practical Pool' in  
380 time while the MSCN pay more attention to the whole workload. To sum up, our KELM could efficiently adapt to the dynamic workloads with integrating statistical methods.

#### 7.5. Robustness of the Re-training interval

In our KELM, retraining interval  $I$  is a significant parameter which controls  
385 the balance of model accuracy and retraining cost. In this section, we test the robustness of the re-training interval parameter in dataset power7. As shown in Figure 6, we evaluate the changes of mean Q-error with re-training interval = 2.5K, 5K, 7.5K, 10K in online phase. And we find that there is only 0.12 higher on average between the performance with 2.5K and the 10K interval. So, an  
390 appropriate interval is enough and we do not need to set the interval as small as possible.

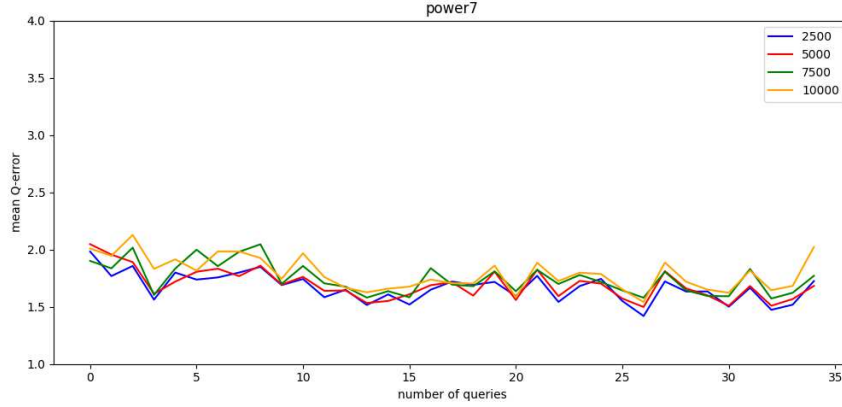


Figure 6: The mean Q-errors of re-training interval = 2.5K, 5K, 7.5K, 10K.

## 8. CONCLUSION

In this study, we propose a knowledge-enhanced learning methodology for cardinality estimation. Different from existing works, our learned cardinality estimator has three significant features: (i) Fast preparation. (ii) High stability. (iii) Quick transition. Our work brings a new development direction for cardinality estimation. On the one hand, we utilize the learned model to achieve high accuracy. On the other hand, we integrate the statistical estimators for availability. In the future, we will focus on the automatic construction of EKB for various scenarios and attempt to apply our methodology to real product environments. We hope that our work could bring more researches for the availability of cardinality estimators.

## References

- [1] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462-467, 1968.
- [2] M. Durand and P. Flajolet. LogLog counting of large cardinalities. In *European Symposium on Algorithms*, pages 605-617. Springer, 2003.

- [3] A. Deshpande, M. N. Garofalakis, and R. Rastogi. Independence is good: dependency-based histogram synopses for high-dimensional data. *acm sigmod record*, 2001.
- [4] Luo, C., Jiang, Z., Hou, WC. et al. A sampling approach for skyline query cardinality estimation. *Knowl Inf Syst* 32, 281-301 (2012). <https://doi.org/10.1007/s10115-011-0441-1>.
- [5] D. Dua and C. Graff. Uci machine learning repository. <http://archive.ics.uci.>, 2017.
- [6] Singh, A., Gupta, S. Learning to hash: a comprehensive survey of deep learning-based hashing methods. *Knowl Inf Syst* 64, 2565-2597 (2022). <https://doi.org/10.1007/s10115-022-01734-0>.
- [7] Zhao Kangfei, Xu Yu Jerey, He Zongyan, Li Rui, and Zhan Hao. Lightweight and accurate cardinality estimation by neural network gaussian process for approximate complex event processing. In *SIGMOD*, 2022.
- [8] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. Learned cardinalities: Estimating correlated joins with deep learning. In *CIDR*, 2019.
- [9] Hu, X., Chu, L., Pei, J. et al. Model complexity of deep learning: a survey. *Knowl Inf Syst* 63, 2585-2619 (2021). <https://doi.org/10.1007/s10115-021-01605-0>.
- [10] R. J. Lipton, J. F. Naughton, and D. A. Schneider. Practical selectivity estimation through adaptive sampling. *ACM SIGMOD Record*, 19(2):1-11, 1992.
- [11] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. Bao: Making learned query optimization practical. New York, NY, USA, 2021. Association for Computing Machinery.

- 435 [12] Catal, C., Giray, G., Tekinerdogan, B. et al. Applications of deep learning for phishing detection: a systematic literature review. *Knowl Inf Syst* 64, 1457-1500 (2022). <https://doi.org/10.1007/s10115-022-01672-x>.
- [13] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. Neo: A  
440 learned query optimizer. 12(11), 2019.
- [14] S. of New York. Vehicle and snowmobile. [catalog.data.gov/ dataset/vehicle-snowmobile-and-boat-registration](https://catalog.data.gov/dataset/vehicle-snowmobile-and-boat-registration). 2019.
- [15] Li, L., Cai, J. & Xu, J. A learned index for approximate kNN queries in high-dimensional spaces. *Knowl Inf Syst* 64, 3325-3342 (2022).  
445 <https://doi.org/10.1007/s10115-022-01742-0>.
- [16] F. Giroire. Order statistics and estimating cardinalities of massive data sets. *Discrete Applied Mathematics*, 157(2):406-427, 2009.
- [17] J. Ortiz, M. Balazinska, J. Gehrke, and S. S. Keerthi. An empirical analysis of deep learning for cardinality estimation. 2019.
- 450 [18] Yongjoo Park, Shucheng Zhong, and Barzan Mozafari. Quicksel: Quick selectivity learning with mixture models. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1017-1033, 2020.
- [19] Viswanath Poosala and Yannis E Ioannidis. Selectivity estimation without  
455 the attribute value independence assumption. In *VLDB*, volume 97, pages 486-495. Citeseer, 1997.
- [20] A. Sharma, F. M. Schuhknecht, and J. Dittrich. The case for automatic database administration using deep reinforcement learning. 2018.
- [21] Ji Sun and Guoliang Li. An end-to-end learning-based cost estimator. 2021.

- 460 [22] K. Tzoumas, A. Deshpande, and C. S. Jensen. Lightweight graphical models for selectivity estimation without independence assumptions. *Pvldb*, 4:2011, 2011.
- [23] Xiaoying Wang, Changbo Qu, Wu Weiyuan, Jiannan Wang, and Qingqing Zhou. Are we ready for learned cardinality estimation? *Proceedings of the VLDB Endowment*, 2021.
- 465 [24] Kossmann Jan, Halfpap Stefan, Jankrift Marcel, Schlosser Rainer. Magic mirror in my hand, which is the best in the land? An Experimental Evaluation of Index Selection Algorithms. *Proceedings of the VLDB Endowment*, 2020.
- 470 [25] Kyu-Young Whang. Index selection in relational databases. In *Foundations of Data Organization*, pages 487-500. Springer, 1987.
- [26] Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. Neurocard: One cardinality estimator for all tables. 14(1), 2020.
- 475 [27] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. Deep unsupervised cardinality estimation. 13(3), 2019.
- [28] X. Zhou, C. Chai, G. Li, and J. I. Sun. Database meets artificial intelligence: A survey. *IEEE Transactions on Knowledge and Data Engineering*, PP(99):1-1, 2020.
- 480