CoLSE: A Lightweight and Robust Hybrid Learned Model for Single-Table Cardinality Estimation using Joint CDF

Lankadinee Rathuwadu, Guanli Liu, Christopher Leckie, Renata Borovica-Gajic
lrathuwadu@student.unimelb.edu.au, {guanli.liu1, caleckie, renata.borovica}@unimelb.edu.au
School of Computing and Information Systems
University of Melbourne

***Abstract*—Cardinality estimation (CE)—the task of predicting the result size of queries—is a critical component of query optimization. Accurate estimates are essential for generating efficient query execution plans. Recently, machine learning techniques have been applied to CE, broadly categorized into query-driven and data-driven approaches. Data-driven methods learn the joint distribution of data, while query-driven methods construct regression models that map query features to cardinalities. Ideally, a CE technique should strike a balance among three key factors: accuracy, efficiency, and memory footprint. However, existing state-of-the-art models often fail to achieve this balance.**

To address this, we propose CoLSE, a hybrid learned approach for single-table cardinality estimation. CoLSE directly models the joint probability over queried intervals using a novel algorithm based on copula theory and integrates a lightweight neural network to correct residual estimation errors. Experimental results show that CoLSE achieves a favorable trade-off among accuracy, training time, inference latency, and model size, outperforming existing state-of-the-art methods.

I. Introduction

Cardinality estimation (CE) is a fundamental yet challenging component of cost-based query optimizers. The optimizer relies on cardinality estimates to evaluate alternative execution plans for a given query and select the most efficient one. Formally, CE refers to the task of estimating the result size of queries with multiple predicates, based on data statistics and assumptions about data distributions, column correlations, and join relationships. A closely related problem is selectivity estimation, which computes the fraction of tuples that satisfy the query predicates. CE has been described as the "Achilles' heel" of query optimization [1], as it is responsible for many of the optimizer's performance issues. Inaccurate cardinality estimates can lead to poor plan choices, resulting in significant performance degradation [2], [3]. Consequently, selectivity and cardinality estimation have been active areas of research for several decades, with numerous studies examining the effectiveness of various techniques.

Existing CE techniques can be broadly categorized into *synopsis-based*, *sampling-based*, and *learning-based* methods [4], each developed for either *single-table* or *join cardinality estimation*.

Single-table CE aims to estimate the number of records returned from a single relation, given one or more local predicates. It plays a foundational role in query optimization, as these estimates influence early-stage decisions such as index usage, access paths, and the ordering of operations within a plan. Inaccuracies at this level can be particularly detrimental, as they propagate upward through the query plan, leading to cumulative estimation errors and often resulting in poor join orderings, unnecessary table scans, or misused indexes [5]–[8]. In contrast, join CE estimates intermediate result sizes across multiple relations and typically builds upon single-table estimates. As a result, inaccuracies in single-table CE can significantly impair join size estimation [9], [10]. Recent work increasingly integrates refined single-table estimates into join CE models [11], highlighting their foundational role. We, thus, focus on the advancement of single-table CE, recognizing its centrality in driving both local and global plan quality.

An ideal CE method should balance three factors: **accuracy** (in generating optimal plans), **efficiency** (in terms of inference and training latency), and **compactness** (in memory usage) [12], [13]. Commercial database systems predominantly employ synopses or sampling techniques for CE [14]. Histogram-based methods, for instance, approximate joint distributions using independence and uniformity assumptions [15], which are frequently violated in real-world data. Consequently, estimation errors in the range of 10^4 to 10^8 have been observed in both open-source and commercial DBMSes [2], [7]. Despite this, synopses and sampling remain widely used due to their low cost and simplicity [9].

Recent advances in machine learning have enabled *learned cardinality estimators*, which significantly outperform traditional methods in accuracy [5], [16], [17]. However, they often incur high training costs, long inference times, and require substantial hyperparameter tuning [5]. These practical limitations hinder adoption in production systems. This work aims to design a learned CE approach that more effectively balances accuracy, efficiency, and memory footprint.

State-of-the-art learned CE methods mainly fall into two paradigms: *query-driven* and *data-driven*. Query-driven models learn from features of training queries [6], [17]–[19], offering fast inference times comparable to traditional methods. However, they typically require large, diverse training query sets to generalize well, and often suffer when the distribution of test queries diverges from that of the training set [20]. In contrast, data-driven models learn a representation of the data's

joint distribution [7], [20]–[22], yielding higher accuracy and better generalization, but at the cost of slower inference, often due to Monte Carlo sampling [7], [23]. A major bottleneck shared by both paradigms is *scalability*. As the data set grows, the size and training time of the model increase - more so for data-driven models, which must learn increasingly complex joint distributions. In contrast, query-driven models maintain manageable complexity given a fixed query workload [24]. This trade-off motivates a hybrid approach.

Our Approach. We propose **CoLSE**, a novel hybrid learned method for single-table cardinality estimation that leverages both data and query workload. CoLSE combines the accuracy of data-driven models with the inference efficiency of query-driven approaches. While a few hybrid approaches exist, most [17], [25] treat data and queries separately, lacking unified modeling of the joint data distribution. UAE [22] addresses this limitation but still models joint probability density functions (PDFs) and relies on progressive Monte Carlo sampling, similar to autoregressive models like Naru [7]. In contrast, we reformulate the problem as direct joint CDF computation over query ranges, thereby avoiding expensive Monte Carlo sampling during inference. To achieve this, we introduce a new copula-based algorithm. Copulas [26] are well-suited for constructing a joint cumulative distribution function (CDF) from the marginal CDFs of individual attributes, offering both accuracy and interpretability.

In summary, we make the following contributions.

1) We propose a hybrid approach for single-table cardinality estimation that combines a novel D-vine copula-based algorithm as the data-driven component with a lightweight neural network as the query-driven component. The neural network learns and corrects estimation errors introduced by the data-driven model.
2) We introduce a novel and interpretable algorithm for modeling joint data distributions via marginal CDFs, grounded in D-vine copula theory. To the best of our knowledge, this is the first architecture to incorporate CDF-based joint distribution modeling into single-table cardinality estimation. We further demonstrate that vine copulas can be effectively applied to this task.
3) We present a new evaluation metric by modifying the PostgreSQL source code to better assess the impact of cardinality estimation methods. Traditional metrics such as Q-error focus solely on local estimation accuracy and overlook broader execution outcomes. Inspired by recent studies [16], [27], [28], our metric is based on the number of optimal query plans,which has been shown to strongly correlate with runtime, thereby providing a more holistic measure of practical performance.
4) We conduct extensive experiments on both real-world and synthetic datasets. Results show that our model achieves a favorable trade-off among accuracy, training time, inference latency, and model size, outperforming state-of-the-art methods across multiple benchmarks.

II. Related Work

Cardinality estimation has led to a wide range of approaches, from traditional statistical techniques to modern learned models. Here, we review classical methods, along with recent query- and data-driven learning-based estimators.

Traditional methods. Multidimensional histograms [29]–[32] are among the most well-studied techniques for capturing attribute correlations [6]. However, they often require significant storage space to maintain accuracy. Sampling-based approaches [33] can better capture complex correlations and dependencies among attributes. Nonetheless, samples may become stale as the underlying data changes, and sampling methods can incur high storage and retrieval overhead, especially on large datasets [4].

Query-driven learned CE methods. These methods treat CE as a supervised regression task, training models to map queries to estimated result sizes using features extracted from query structures. MSCN [19] uses a multi-set convolutional network that represents each query as a feature vector composed of table, join, and predicate modules—each implemented as a two-layer neural network. It also leverages a materialized sample to improve learning. LW-XGB and LW-NN [6] propose lightweight models based on XGBoost and neural networks, respectively. Their input features include both range features and CE features, derived from heuristics such as histograms and domain knowledge. DQM-Q [17] introduces a custom featurization method for training a neural CE model.

Data-driven learned CE methods. Data-driven approaches model CE as a joint probability distribution estimation task, aiming to learn the full joint distribution $P(A_1, A_2, \ldots, A_n)$ over table attributes, from which selectivity can be inferred. These models are typically unsupervised and rely on deep autoregressive models or probabilistic graphical models (PGMs). Naru [7] and DQM-D [17] utilize deep autoregressive architectures such as MADE [34] and Transformers [35] to approximate conditional probabilities between attributes. However, these models suffer from high training and inference times, limiting their applicability in real-world DBMSs. DeepDB [20] employs relational sum-product networks (RSPNs), a type of PGM, to capture both marginal and joint distributions. A key limitation of SPNs is that they retain local independence assumptions [5]. Other PGM-based methods use Bayesian networks [36], [37] to model conditional independencies, but structure learning is NP-hard [13], [38], making them expensive to train on large datasets.

Complimentary to the above, there is a line of work which improves query performance by tolerating CE errors via learned steering—learning hint sets or ranking/forcing plans from runtime feedback [39]–[42]. In contrast, direct CE estimation improves the estimates themselves, yielding system-wide, interpretable gains without exploration or cold-start overhead and ensuring predictable behavior within standard optimizer abstractions.

III. Problem Statement

We formally define the task of cardinality estimation for single-table queries involving range and equality predicates.

Consider a relation T consisting of n columns (or attributes), denoted as $\{A_1, A_2, \ldots, A_n\}$. A tuple $x \in T$ is an n-dimensional vector. A query q is defined as a conjunction of predicates, where each predicate imposes a constraint on a single attribute. Predicates may take the form of an equality constraint ($A_k = c$), an open range constraint ($lb_k \leq A_k$), or a closed range constraint ($lb_k \leq A_k \leq ub_k$).

Cardinality. The *cardinality* of query q, denoted by $|q(T)|$, is the number of tuples in T that satisfy all conditions specified in q:

$$|q(T)| = \sum_{x \in T} \prod_{k=1}^{n} I_k(x) \tag{1}$$

where $I_k(x)$ is an indicator function that evaluates to 1 if tuple x satisfies the predicate on attribute A_k, and 0 otherwise.

Selectivity. The *selectivity* of q, denoted as $sel(q)$, is the probability that a randomly selected tuple from T satisfies all predicates in q:

$$sel(q) = \frac{|q(T)|}{|T|} \tag{2}$$

Alternatively, in probabilistic terms, if each predicate defines a bounded range, the selectivity can be expressed as:

$$sel(q) = P(lb_1 \leq A_1 \leq ub_1, \ldots, lb_n \leq A_n \leq ub_n) \tag{3}$$

Objective. In this work, we focus on modeling *selectivity* rather than cardinality directly. Since the two are linearly related via the total number of tuples $|T|$, modeling selectivity provides a probabilistic foundation that more naturally captures predicate interactions and generalizes across queries. Our goal is to build a selectivity estimation technique that achieves an optimal balance between accuracy, memory efficiency, and inference/training time by leveraging both the underlying data and the observed query workload.

IV. Background on Copula Models

Selectivity estimation fundamentally requires modeling the joint distribution of query attributes. Many traditional approaches assume independence between attributes, which can lead to large estimation errors in the presence of statistical dependencies [4], [29]. To address this, we leverage **copula theory**—a mathematical function that captures the dependence between random variables. Its applicability to modeling joint distributions is grounded in **Sklar's theorem** [43], which states that any multivariate joint distribution can be constructed by first modeling the marginal distributions of individual attributes and then separately capturing their dependency structure. This decoupling is particularly advantageous for cardinality estimation, as it allows for flexible and accurate representation of complex inter-attribute relationships, including non-linear and asymmetric dependencies, which are common in real-world datasets. We refer readers to [44]–[46] for more detailed tutorials on copulas.

Formally, for a random vector $[X_1, ..., X_d]$ with marginal CDFs $F_i(x_i)$, the joint CDF can be written as:

$$P(X_1 \leq x_1, ..., X_d \leq x_d) = C(F_1(x_1), ..., F_d(x_d)), \tag{4}$$

where C is the copula function. This transformation is enabled by the **Probability Integral Transform** [47], which states that applying the CDF of a continuous random variable X to itself yields a uniformly distributed variable $U = F_X(X)$ over $[0, 1]$. This allows copulas to operate in a normalized domain, making them broadly applicable to multivariate modeling.

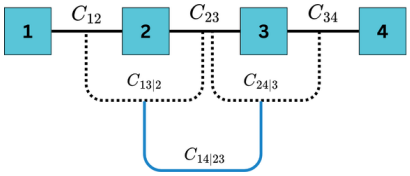


Fig. 1: D-vine structure for four variables: dependencies are modeled sequentially.

Several families of copulas exist, including Gaussian, t-copulas, and Archimedean copulas [26]. While effective in low dimensions, these classical models often struggle to scale due to increasing complexity and rigidity.

To overcome these limitations, **vine copulas** were introduced [48], enabling the construction of high-dimensional copulas using hierarchical compositions of bivariate (pairwise) copulas. This modular design enhances flexibility and interpretability. Among vine copulas, we adopt the **D-vine copula** due to its natural alignment with the structure of conjunctive predicates in SQL queries. In a D-vine, variables are arranged in a sequence, and dependencies are modeled step-by-step between adjacent variables (Fig. 1). This sequential structure simplifies training, mitigates overfitting, and allows the model to scale to higher dimensions while maintaining tractability [49], [50].

Therefore, we incorporate D-vine copulas into our model, using them to accurately and efficiently capture the joint cumulative distribution of query attributes. However, classical vine copulas are primarily defined for conditional densities. This requires designing a novel algorithmic adaptation that transforms the traditional density-based formulation into a CDF-based computation suitable for multi-attribute range queries. The details of these algorithmic adaptations and their integration with the query-driven components are presented in the following sections.

V. CoLSE Framework

This section presents the design and core components of the CoLSE framework, a hybrid selectivity estimator that integrates data-driven and query-driven learning. CoLSE models complex dependencies via a copula-based decomposition and refines estimates with a learned error-correction module. We first outline the architecture, then describe marginal distribution construction, the D-vine-based joint probability estimator, the error-compensation network, and support for categorical/discrete attributes and joins.

A. An Overview

CoLSE (Copula based Learned Selectivity Estimator) is a hybrid cardinality estimator that models the joint data distribution using copula-based decomposition. It integrates two key components: (1) a joint probability estimator (JPE) that captures the joint data distribution using copula-based decomposition (data-driven), and (2) an error compensation network

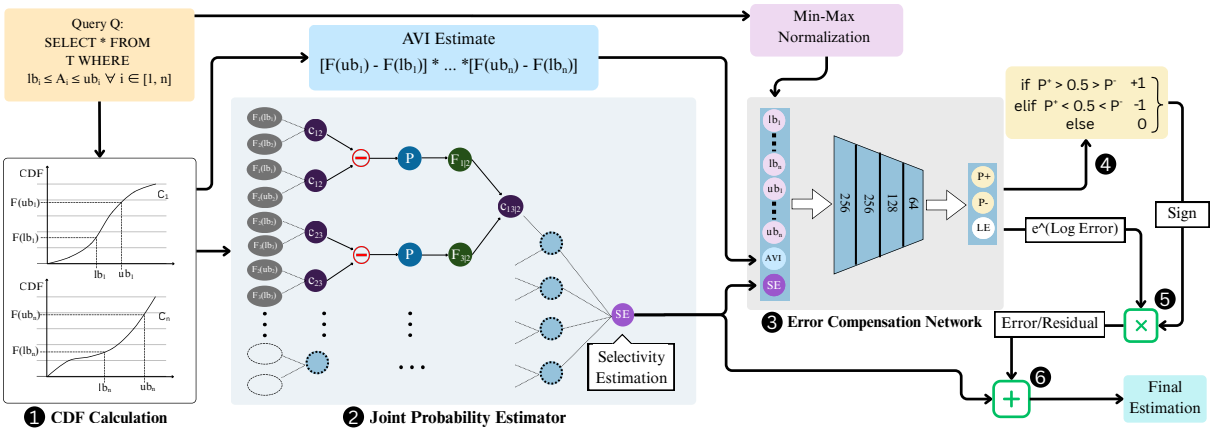


Fig. 2: CoLSE: The Overall Architecture

(ECN) that uses query workloads to refine the estimates for higher accuracy (query-driven). CoLSE is categorized as a hybrid approach due to its ability to learn from both data and observed query workloads.

Figure 2 illustrates the CoLSE architecture. Given a query Q, we begin by extracting the lower and upper bounds (lb_i, ub_i) for each selection predicate x_i, then evaluate their marginal CDFs $F(lb_i)$ and $F(ub_i)$ (Step ❶). These values are passed to the JPE, a tree-structured module based on D-vine copulas (Step ❷). Unlike conventional data-driven methods that approximate the joint probability density function (PDF), the JPE directly estimates the joint probability over the query ranges. This avoids costly sampling procedures and leads to significantly lower inference latency.

Next, the output of the JPE, along with the input CDFs, is passed to the ECN, a lightweight neural network trained on past query workloads (Step ❸). The ECN provides targeted corrections to the initial estimate, compensating for inaccuracies arising from modeling limitations or distributional shifts.

Residual correction is computed based on the ECN output (Steps ❹–❺) and applied to the initial estimate, yielding the final selectivity prediction (Step ❻). This architecture combines the strengths of data- and query-driven methods, resulting in robust and accurate selectivity estimation.

B. Marginal CDF Distribution Modelling

Marginal CDFs are essential inputs to the JPE, forming the basis for copula decomposition. To construct each marginal, we partition the data into B bins (e.g., $B = 5{,}000$) and compute the empirical CDF at bin edges. We then fit a Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) to the (x_j, CDF_j) pairs, yielding a smooth, strictly monotonic spline bounded in $[0, 1]$. At inference time, the marginal CDF of a query point x is quickly retrieved using the PCHIP spline.

C. Novel Algorithm based on D-vine Copula

At the core of CoLSE's JPE is a novel algorithm that estimates multi-dimensional selectivities directly over query ranges by combining principles from probability theory and D-vine copulas. Unlike prior methods that model the full joint PDF, our approach entirely avoids high-dimensional density estimation by relying solely on *pairwise copula functions*, which capture dependencies between attribute pairs by first transforming each attribute to its marginal CDF (i.e., uniform scores), and then modeling their joint behavior in the uniform space. This formulation significantly reduces model complexity while preserving essential dependency structures.

Input : Upper and lower bounds $[(lb_i, ub_i)]$ for queried attributes $[i_1, ..., i_n]$;
Output: Selectivity Estimate (SE)
Uses : $\{F_{i_1}, F_{i_2}, \ldots, F_n\}$; // CDF functions

$n \leftarrow$ number of queried attributes;
if $n = 1$ **then**
  return $F_i(ub_i) - F_i(lb_i)$;
end
if $n = 2$ **then**
  return $C(F_i(lb_i), F_j(lb_j)) - C(F_i(lb_i), F_j(ub_j))$
    $- C(F_i(ub_i), F_j(lb_j)) + C(F_i(ub_i), F_j(ub_j))$;
else
  $B \leftarrow [(lb_1, lb_n), (lb_1, ub_n), (ub_1, ub_n), (ub_1, lb_n)]$;
  $res \leftarrow 0$;
  for $k = 0$ **to** 3 **do**
    $(b^x, b^y) \leftarrow B[k]$;
    $res \leftarrow res + (-1)^k \cdot Recursive(b^x, [i_1, ..., i_n], b^y)$;
  end
  return res;
end

Algorithm 1: D-vine Copula Estimation. Note that θ is needed in all the copula function (C) calculations; we omit it due to space limitations.

In practice, we instantiate each pair-copula using the Gumbel [51] copula, a member of the Archimedean family known for its simple closed-form expression and single parameter that controls dependence strength. Compared to alternatives such as the Clayton or Frank copulas, the Gumbel copula offers greater flexibility in capturing asymmetric dependencies, which commonly arise in real-world datasets. The dependence parameter θ is estimated using Kendall's rank correlation [52], computed from the data points in each attribute pair.

JPE requires a predefined attribute sequence to model dependencies sequentially. The natural schema order is used as a fixed global ordering for all queries. When a query includes only a subset of attributes, their relative positions in the global order are preserved, and intermediate attributes are skipped (e.g., [City, Year, Make, Model] → [Make, Model]). Further justification for this choice is provided in Section VII-G.

Input : b^x, $[i_1, \ldots, i_n]$, b^y
Output: Conditional Copula $C_{i_1,i_n|i_2 \ldots i_{n-1}}$
Uses : $\{F_{i_1}, F_{i_2}, \ldots, F_n\}$

`Recursive` (b^x, $[i_1, \ldots, i_n]$, b^y):

if $n = 3$ **then**
  $U_1 \leftarrow C_{i_1,i_2}\big(F_{i_1}(b^x_{i_1}),\ F_{i_2}(ub_{i_2})\big)$;
  $L_1 \leftarrow C_{i_1,i_2}\big(F_{i_1}(b^x_{i_1}),\ F_{i_2}(lb_{i_2})\big)$;
  $P(i_{1(b^x)}, i_2) \leftarrow U_1 - L_1$;
  $U_2 \leftarrow C_{i_2,i_3}\big(F_{i_3}(b^y_{i_3}),\ F_{i_2}(ub_{i_2})\big)$;
  $L_2 \leftarrow C_{i_2,i_3}\big(F_{i_3}(b^y_{i_3}),\ F_{i_2}(lb_{i_2})\big)$;
  $P(i_2, i_{3(b^y)}) \leftarrow U_2 - L_2$;
  $P(i_2) \leftarrow F_{i_2}(ub_{i_2}) - F_{i_2}(lb_{i_2})$;
  $F_{i_1|i_2} \leftarrow P(i_{1(b^x)}, i_2)/P(i_2)$;
  $F_{i_3|i_2} \leftarrow P(i_2, i_{3(b^y)})/P(i_2)$;
  return $C_{i_1,i_3|i_2}(F_{i_1|i_2}, F_{i_3|i_2})$;
end
else
  $C^{lb}_{i_1,i_{n-1}} \leftarrow$ ***Recursive***$(b^x, [i_1, \ldots, i_{n-1}], lb)$;
  $C^{ub}_{i_1,i_{n-1}} \leftarrow$ ***Recursive***$(b^x, [i_1, \ldots, i_{n-1}], ub)$;
  $F_{i_1|i_2,\ldots,i_{n-1}} \leftarrow \dfrac{(C^{lb}_{i_1,i_{n-1}} - C^{ub}_{i_1,i_{n-1}}) \cdot P(i_2, \ldots, i_{n-2})}{P(i_2, \ldots, i_{n-1})}$;
  $C^{lb}_{i_2,i_n} \leftarrow$ ***Recursive***$(lb, [i_2, \ldots, i_n], b^y)$;
  $C^{ub}_{i_2,i_n} \leftarrow$ ***Recursive***$(ub, [i_2, \ldots, i_n], b^y)$;
  $F_{i_n|i_2,\ldots,i_{n-1}} \leftarrow \dfrac{(C^{lb}_{i_2,i_n} - C^{ub}_{i_2,i_n}) \cdot P(i_3, \ldots, i_{n-1})}{P(i_2, \ldots, i_{n-1})}$;
  $P(i_2, i_3, \ldots i_n) = P(i_2, i_3, \ldots i_{n(ub)}) - P(i_2, i_3, \ldots i_{n(lb)})$
  return $C_{i_1,i_n|i_2 \ldots i_{n-1}}(F_{i_1|i_2,\ldots,i_{n-1}}, F_{i_n|i_2,\ldots,i_{n-1}})$;
end

Algorithm 2: Recursive Conditional Copula Estimation

Algorithm 1 presents the generalized procedure for selectivity estimation over a table with n query attributes. The algorithm is dynamic and adapts its structure based on the number of queried attributes. We distinguish three cases: (i) single-attribute queries, (ii) two-attribute queries, and (iii) multi-attribute queries ($n > 2$).

Case 1. For a single predicate on attribute X, the selectivity reduces to the difference of marginal CDF values (line 3 in Algorithm 1): $P(lb \leq X \leq ub) = F(ub) - F(lb)$.

Case 2. For two-attribute queries, we apply the inclusion–exclusion principle [53] (line 6 in Algorithm 1). This is because range queries define axis-aligned rectangles in the attribute space, while the joint CDF provides cumulative probabilities from the origin to a point. To compute the probability of a bounded region, we must combine CDF values at the rectangle's corners while correcting for overlaps. Let F_1 and F_2 be the marginal CDFs of X_1 and X_2, respectively. Their joint CDF, F_{12}, is approximated via a bivariate copula function C_{12} as: $F_{12}(x_1, x_2) = C_{12}(F_1(x_1), F_2(x_2))$.

The final selectivity estimate is obtained by applying inclusion–exclusion over the rectangular query range (as in Eq. (5)).

$$\begin{aligned} &P(\mathrm{lb}_1 \leq X_1 \leq \mathrm{ub}_1, \mathrm{lb}_2 \leq X_2 \leq \mathrm{ub}_2) \\ &= F_{12}(\mathrm{lb}_1, \mathrm{lb}_2) - F_{12}(\mathrm{lb}_1, \mathrm{ub}_2) - F_{12}(\mathrm{ub}_1, \mathrm{lb}_2) + F_{12}(\mathrm{ub}_1, \mathrm{ub}_2) \\ &= C_{12}(F_1(\mathrm{lb}_1), F_2(\mathrm{lb}_2)) - C_{12}(F_1(\mathrm{lb}_1), F_2(\mathrm{ub}_2)) \\ &\quad - C_{12}(F_1(\mathrm{ub}_1), F_2(\mathrm{lb}_2)) + C_{12}(F_1(\mathrm{ub}_1), F_2(\mathrm{ub}_2)) \end{aligned} \tag{5}$$

In Eq. (5), $F_{12}(\mathrm{ub}_1, \mathrm{ub}_2)$ represents the probability that both X_1 and X_2 are less than or equal to the upper bounds ub_1 and ub_2. $F_{12}(\mathrm{lb}_1, \mathrm{ub}_2)$ subtracts the region where $X_1 < \mathrm{lb}_1$, and $F_{12}(\mathrm{ub}_1, \mathrm{lb}_2)$ subtracts the region where $X_2 < \mathrm{lb}_2$. $F_{12}(\mathrm{lb}_1, \mathrm{lb}_2)$ adds back the region that was subtracted twice by the previous two terms.

Case 3. For queries involving more than two attributes, we apply the inclusion–exclusion principle to extract the four corner points of the query attributes (line 8 in Algorithm 1), and derive four signed copula terms corresponding to the outermost attribute pair in the D-vine structure. For each copula term, we recursively compute conditional copula values (line 13 in Algorithm 1) on the remaining dimensions using Algorithm 2. The recursion progresses by conditioning on intermediate attributes and terminates when it reaches the base case (i.e., $n = 3$).

To clarify the recursive computation in Algorithms 1 and 2, we include a worked example for 4 attributes with variable order A_1, A_2, A_3, A_4. Consider the example query: `SELECT * FROM T WHERE 10 ≤ A1 ≤ 20 AND 5 ≤ A2 ≤ 15 AND 100 ≤ A3 ≤ 200 AND 50 ≤ A4 ≤ 120;` with marginal CDFs F_1:[0.2,0.4], F_2:[0.3,0.6], F_3:[0.1,0.5], F_4:[0.25,0.55]. We aim to compute $P(10 \leq A_1 \leq 20, 5 \leq A_2 \leq 15, 100 \leq A_3 \leq 200, 50 \leq A_4 \leq 120) = P(A_1, A_4 \mid A_2, A_3) \times P(A_2, A_3)$.

Step 1 (Level 1): Compute pairwise probabilities using inclusion-exclusion on C_{12}, C_{23}, C_{34}. Example: $P(A_1, A_2) = C_{12}(0.4, 0.6) - C_{12}(0.4, 0.3) - C_{12}(0.2, 0.6) + C_{12}(0.2, 0.3)$.

Step 2 (Level 2): Compute conditional CDFs and copulas: $F_{1|2}, F_{3|2}, F_{4|3} \rightarrow$ then evaluate $C_{13|2}$ and $C_{24|3}$. Using these, derive $F_{1|23}$ and $F_{4|23}$.

Step 3 (Level 3): Apply inclusion-exclusion on (A_1, A_4) corners using $C_{14|23}$ to obtain $P(A_1, A_4 \mid A_2, A_3)$, thus the final probability.

D. Error Compensation Network

To further refine the selectivity predicted by the JPE, CoLSE employs a lightweight neural network trained to estimate the residual error. This network corrects systematic biases that are not captured by the copula-based model.

Inputs. The network takes as input: (i) normalized lower and upper bounds of the queried predicates, (ii) the JPE output, and (iii) a heuristic estimate computed via the Attribute Value Independence (AVI) assumption—that is, the product of marginal CDF differences. Following prior work [6], we find that including AVI improves both robustness and accuracy.

Outputs. The model predicts: (i) the log absolute residual, (ii) the probability that the residual should be added (P^+), and (iii) the probability that it should be subtracted (P^-). Residual correction is applied only when one probability exceeds the other and is greater than 0.5. ECN architecture separates magnitude and sign predictions to stabilize training by mitigating oscillations. Independent sign heads act as a confidence-driven gate, suppressing uncertain corrections to produce a more reliable and conservative adjustment process.

Architecture. The model consists of four fully connected layers with 256, 256, 128, and 64 neurons, each followed by

TABLE I: Dataset Characteristics

Dataset	Size (MB)	Rows	Cols/Cat	Domain
Census	4.8	49K	13/8	10^{16}
Forest	44.3	581K	10/0	10^{27}
Power	110.8	2.1M	7/0	10^{17}
DMV	972.8	11.6M	11/10	10^{15}

ReLU activation. The output layer has 3 neurons. This compact architecture enables efficient inference while capturing non-linear patterns between query features and estimation errors.

Training. The model is trained using a custom loss function that combines Mean Squared Error (for residual magnitude) and Binary Cross-Entropy with logits (for residual sign). The target residual is computed as the log difference between the ground truth and the JPE estimate.

E. Handling Categorical and Discrete Variables

Copula models and CDFs are inherently defined over continuous domains. To support categorical and discrete variables, we adopt a dequantization technique inspired by [3], which maps discrete values into a continuous space.

For categorical variables, we first order them alphabetically and then apply label encoding (e.g., $E(\text{Cook}) \rightarrow 0$) to convert them into discrete numeric form (see Section VII-G for further discussion). Both categorical and inherently discrete variables are then dequantized using a spline-based continuous distribution. This process involves two steps: (i) constructing a smooth CDF using PCHIP splines, and (ii) sampling from this CDF and inverting the spline via a fast, precomputed lookup table to produce continuous-valued representations.

After transformation, equality predicates on categorical attributes are translated into range queries. For example, if "Cook" is mapped to 0, the corresponding predicate becomes $0 \leq x < 1$, covering the continuous interval assigned to that category. This reformulation ensures compatibility with the copula-based estimation process.

F. Extension to Joins

Extending beyond single tables, we handle equi-joins by treating the join domain as a common key space across all joined tables and estimate the contribution of each key value to the join result. Instead of materializing per-key frequencies, which is infeasible for large domains, we partition the key space into bins and approximate within-bin frequencies using averages derived from our copula-based single-table estimator.

For each bin, we compute an approximate density of tuples per relation, and then estimate the join size by combining these densities across relations. Summing the contributions of all bins yields the final join cardinality estimate. This strategy preserves the advantages of our single-table approach, while scaling naturally to multi-relation equi-joins.

VI. Experimental Setup

The experiments are conducted on two Linux servers. Model training is performed on a server equipped with 16 Intel Xeon Platinum 8562Y+ CPUs @ 2.80GHz, one NVIDIA L40S-24Q GPU, and 116 GB of memory. End-to-end evaluation on PostgreSQL is carried out on another server with 4 AMD EPYC 7763 CPUs @ 2.50GHz.

A. Datasets

We evaluate all baselines on **four** real-world datasets, following the benchmark in [5], and **two** synthetic datasets. Table I summarizes the real-world dataset characteristics: "Cols/Cat" denotes the total and categorical column counts, while "Domain" refers to the product of the distinct values in each column.

The synthetic datasets are designed to test robustness and scalability. The first consists of data with uniform pairwise correlation coefficients between 0.2 and 0.8, aimed at evaluating how query-driven methods tolerate changes in data correlation. The second is derived from the TPC-H `lineitem` table, where we vary the Zipfian skew from 1 to 4 and dataset sizes from 0.1GB to 20GB (0.1, 1, 10, 20), following [54]. This setup assesses how data-driven methods scale in training time as dataset size increases. To test robustness further, we modify the DMV dataset by incrementally inserting 20% of random, correlated, and skewed data.

B. Query Workloads

The queries are generated using the workload generator from [5]. Each query is created in three steps: selecting predicate attributes, choosing query centers, and assigning operators and widths. The query center distribution (e.g., uniform or skewed) is varied to evaluate baseline robustness under different workloads.

Each dataset includes 100,000 training, 10,000 validation, and 10,000 test queries with ground truth cardinalities. For training query-driven baselines, we use 100,000 training and 10,000 validation queries. In contrast, the proposed model's error compensation network is trained with only 80,000 queries. All models are evaluated on the same 10,000-query test set.

C. Baseline Methods

We compare with eight baselines including three traditional methods and five learned methods.

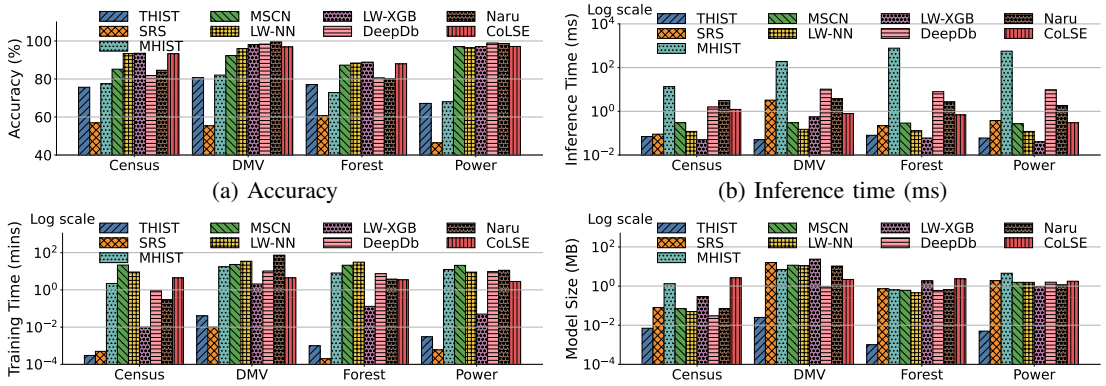
Traditional methods:

1. THIST [55]: a one-dimensional histogram estimator assuming full independence across attributes.
2. SRS [4]: a simple random sampling-based estimator that evaluates selectivity by probing sampled tuples, empirically capturing correlations.
3. MHIST [29]: a multidimensional histogram that partitions the joint data space into buckets to approximate correlated selectivity.

Query-driven Models:

4. MSCN [19]: a supervised neural estimator using multi-set convolutional networks with separate embeddings for tables, joins, and predicates.
5. LW-NN [6]: a lightweight neural network model trained on handcrafted query features, including predicate ranges and auxiliary CE estimates.
6. LW-XGB [6]: a gradient-boosted tree model using the same features as LW-NN.

Data-driven Models:



(a) Accuracy (b) Inference time (ms)

(c) Training time (min) (d) Model size (MB)

Fig. 3: Comparison against State-of-the-Art across four real-world datasets

7. DeepDb [20]: a data-driven estimator based on relational sum-product networks (SPNs) that model joint and marginal distributions.
8. Naru [7]: a deep autoregressive model that learns the full joint distribution using Residual Masked Autoencoder for Distribution Estimation (ResMADE) style networks.

LW-NN and LW-XGB use heuristic features from histograms and domain knowledge to improve estimation. We exclude DQM [17] as its data-driven model performs similarly to Naru, while its query-driven model does not support range queries, making it incompatible with our workloads [5].

D. Evaluation Metrics

Accuracy: We measure accuracy as the proportion of queries whose plans, generated using estimated cardinalities, are identical to those generated with true cardinalities in PostgreSQL over 10,000 test queries. We also observed in our experiments that this metric is strongly correlated with end-to-end execution times, further supporting its validity as a practical performance indicator.

Inference Latency: Since cardinality estimators are invoked repeatedly during query planning, high inference latency slow optimization and negate the benefits of accurate estimates.

Model Size: Models are often loaded into memory during planning; smaller ones reduce memory use and support deployment in resource-limited settings.

Offline Training Time: Shorter training times reduce computational costs, particularly for large-scale databases, and facilitate timely model updates in response to data changes.

Discussion: Q-error is a common metric for evaluating cardinality estimates, but it raises the question of whether numerical precision is necessary. Since the optimizer's goal is to minimize execution time–not to predict exact cardinalities–prior work [16], [27] has argued for metrics that better reflect query performance, shifting focus from estimate accuracy to overall effectiveness.

E. Modifications to PostgreSQL

To evaluate estimation accuracy, we modified PostgreSQL [56] to integrate externally predicted cardinalities. Specifically, we added a function to `costsize.c`, which computes the cost of potential access paths [57], to load selectivities. During query planning, PostgreSQL invokes `set_baserel_size_estimates()` to estimate base relation cardinalities. We extended this function to populate a new field, `custom_selectivity`, in the `PlannerInfo` structure with external values. We also modified `clauselist_selectivity()`, used throughout `costsize.c`, to prioritize `custom_selectivity` when available. This ensures consistent use of external estimates across all access path evaluations and avoids conflicts between internal and injected values [57].

VII. Experimental Results

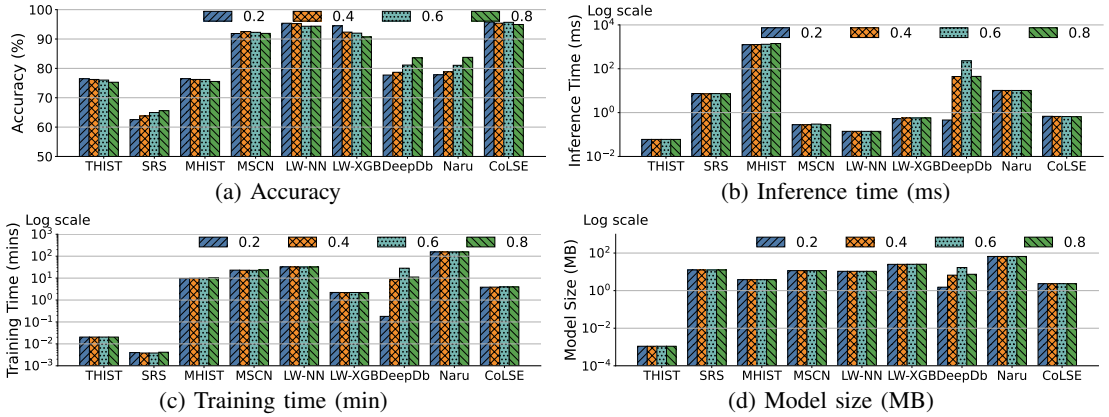
To evaluate the effectiveness of CoLSE, we conduct a comprehensive set of experiments addressing the following research questions:

Q1: How does CoLSE compare to traditional, data-driven, and query-driven baselines in terms of accuracy, inference time, training time, and model size? (Section VII-A) **Q2:** How do varying column correlations affect these core metrics across all methods? (Section VII-B) **Q3:** How do dynamic data updates impact model performance and robustness? (Section VII-C) **Q4:** How well do different methods adapt to workload shifts? (Section VII-D) **Q5:** How does data skew influence accuracy and latency across methods? (Section VII-E) **Q6:** How sensitive are different methods to changes in dataset cardinality? (Section VII-F)

A. Comparison against State-of-the-Art

Fig. 3 presents the (a) accuracy, (b) inference time, (c) training time, and (d) model size of CoLSE and baseline methods across four real-world datasets.

1) Accuracy Comparison: CoLSE consistently delivers strong performance across all datasets, matching the accuracy of top-performing models while maintaining robustness. In contrast, while learned methods generally outperform traditional ones, their performance drops on the Census and Forest datasets. This decline is primarily due to increased randomness in these datasets, which makes it harder for models to learn meaningful patterns. The effect is especially pronounced for data-driven methods, whose accuracy is more dependent on the underlying data structure.



(a) Accuracy (b) Inference time (ms)

(c) Training time (min) (d) Model size (MB)

Fig. 4: Varying correlation levels

2) Inference Latency Comparison: CoLSE maintains low inference latency (under 1.5 ms), making it suitable for real-time workloads. Although slightly slower than query-driven models due to its two-stage design, it remains significantly faster than data-driven methods. Among learned models, data-driven approaches show notably higher latency. DeepDB scales poorly, with latency increasing sharply on larger datasets. Naru remains moderate (2–4 ms) but is still slower than query-driven methods like MSCN, LW-NN, and LW-XGB, which all sustain sub-millisecond latencies. Among traditional methods, THIST offers the lowest latency but limited accuracy; SRS shows moderate and dataset-dependent latency; MHIST, while more accurate, is too slow for real-time use (up to 700 ms).

3) Training Time Comparison: CoLSE achieves one of the fastest training times among learned models, staying under 5 minutes even on large datasets like DMV. While not as fast as LW-XGB, it outperforms all other query-driven and data-driven baselines in training efficiency, making it practical for deployment. Among learned baselines, MSCN and LW-NN generally train slower ($>$ 25 minutes) due to workload encoding and model complexity. LW-XGB is a notable exception, completing training in 2–5 minutes across datasets. Data-driven models exhibit high variability: Naru takes over 70 minutes on DMV but only 0.5 minutes on Census. Traditional methods like THIST and SRS are extremely fast (a few seconds), while MHIST, though more accurate, becomes slower on large datasets (15+ minutes on DMV).

4) Model Size Comparison: CoLSE maintains a compact representation—under 3 MB—even on large datasets like DMV, while also achieving low training and inference times. Its model size is mainly determined by the marginal CDFs and parameters of the D-vine and error compensation network. As shown in Fig. 3d, model sizes rise significantly for the largest dataset (DMV) across all approaches except DeepDb and CoLSE. THIST remains the most lightweight among traditional methods. In contrast, MHIST and SRS exhibit model sizes comparable to learned models. Interestingly, DeepDb maintains the smallest model size across all datasets despite higher training and inference costs. LW-XGB, while efficient in training and inference, incurs a steep model size increase—reaching ~24 MB on DMV—indicating high memory usage and limited portability.

B. Impact of Column Correlation

Fig. 4 shows the accuracy, training and inference time, and model size of all under varying column correlations (ρ).

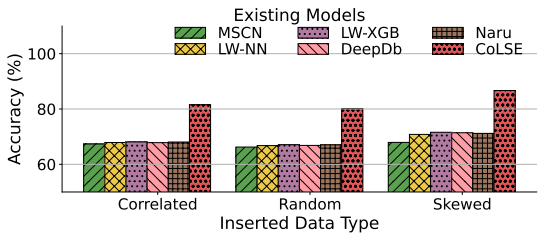
1) Accuracy Comparison: CoLSE is best across all correlations (peak 95.79%), maintaining top-tier performance. Learned baselines beat traditional methods but generally trail CoLSE in robustness. Among them, query-driven models outperform data-driven ones. DeepDB and Naru show improved accuracy as correlation increases (∼77% to 83.5% and 77.8% to 83.76%), indicating these models benefit from more predictable attribute patterns. MSCN and LW-NN stay consistently high, whereas LW-XGB degrades at higher correlation (94.5% at 0.2 to 90.69% at 0.8), indicating reduced robustness.

2) Inference Latency Comparison: CoLSE maintains sub-millisecond inference times (<1 ms) across all correlations, comparable to query-driven methods. Data-driven models are slower: Naru holds steady around 10 ms; while DeepDB is highest among learned approaches. Among traditional baselines, MHIST is slowest (>1,200 ms). THIST is relatively fast, and SRS is moderate (∼7 ms), still above most learned models.

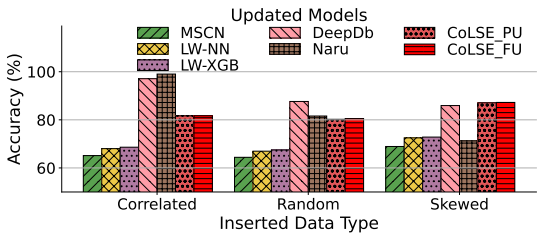
Most models show stable latency across correlations because hyperparameters were fixed to isolate correlation effects. DeepDB is the exception: despite identical settings, its latency is non-monotonic—rising to (∼220 ms) at $\rho = 0.6$ and then falling to (∼45 ms) at $\rho = 0.8$—likely due to how its RSPN ensemble is constructed from attribute correlations.

3) Training Time Comparison: Training time trends generally mirror inference latency patterns. CoLSE is among the fastest learned models (≤5 min), second only to LW-XGB (≤5 min). Naru is slowest (∼160 min). DeepDB spikes at $\rho = 0.6$ (∼30 min), mirroring its latency pattern. MSCN and LW-NN are in mid-range (20–30 min). As expected, traditional methods incur minimal training: MHIST completes in under 10 minutes, while THIST and SRS build within seconds.

4) Model Size Comparison: CoLSE is most compact (∼2 MB), enabling memory-constrained deployment. Naru is largest (∼67 MB) due to its deep autoregressive design. DeepDB varies non-monotonically, ∼2 MB at $\rho = 0.2$, peaking ∼17 MB at $\rho = 0.6$, then ∼8 MB at $\rho = 0.8$, mirroring its



(a) Accuracy of models trained on past data



(b) Accuracy of models trained on new data

Model	DMV
MSCN	3.97
LW-XGB	1.43
LW-NN	6.83
DeepDB	0.19
Naru	1.29
CoLSE_PU	0.05
CoLSE_FU	1.55

(c) Updating times of models (minutes)

Fig. 5: Performance of models under different conditions of data updates on DMV data

latency/training trends. Among query-driven models, MSCN and LW-NN are stable between 10–13 MB, while LW-XGB is moderately larger (∼25 MB) owing to its ensemble model.

C. Impact of Data Updates

Fig. 5 illustrates how state-of-the-art models respond to different types of data updates before and after retraining, along with their respective update times. Each model is retrained following their original implementations. Specifically, MSCN, LW-XGB, and LW-NN are retrained using newly generated workloads comprising 10,000, 8,000, and 16,000 queries, respectively. Naru is updated by performing one additional training epoch, while DeepDB is incrementally updated by inserting a small sample (1%) of the newly appended data into its tree-based model. To handle data updates, CoLSE can be retrained in two steps: 1) updating the marginal CDF distributions along with the dependency parameters needed for copula calculations-referred to as "CoLSE_PU" (Partially Updated) and 2) additionally retraining the error compensation network-referred to as "CoLSE_FU" (Fully Updated).

The Fig. 5a illustrates that all models experience some performance degradation following the insertion of new data. However, CoLSE exhibits the least deterioration, demonstrating greater robustness compared to other baselines across various types of data updates. After retraining (Fig. 5b), all models show improved accuracy, with data-driven models benefiting the most. While CoLSE does not consistently outperform all models post-retraining, it remains the most effective—particularly when the updated data is skewed. Notably, the performance of CoLSE_PU and CoLSE_FU is nearly identical, suggesting that retraining the error compensation network is not always necessary.

The Fig. 5c presents the model updating times. Query-driven models generally require longer update times due to the overhead of query generation. Among them, LW-XGB exhibits the shortest update time. When CoLSE is fully updated, its update time remains comparable to that of LW-XGB, which is 1.5 minutes. However, CoLSE_PU achieves smallest updating time across all the models, which is 0.05 minutes.

We evaluate robustness on DMV in update-heavy settings by inserting 5% new tuples after each query batch for four rounds (20% total), with skewed or correlated drift (Fig. 6). Here, CoLSE-old and Naru-old refer to existing models without retraining. Overall, CoLSE-Old remains accurate, while CoLSE-PU provides small, consistent gains over the existing model. In contrast, Naru degrades as drift accumulates, indicating that CoLSE is robust under continual updates.

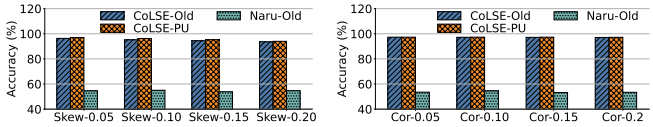


Fig. 6: Performance of CoLSE vs. Naru on DMV under continual updates

D. Impact of Workload Shifts

Fig. 7 illustrates how state-of-the-art models respond to shifts in query distributions. We simulate workload shifts by incrementally replacing 25% of the original workload with queries drawn from a different distribution at each step.

As the shift ratio increases, CoLSE demonstrates robustness comparable to data-driven methods, consistently outperforming all query-driven baselines. This indicates that CoLSE not only captures complex data correlations but also generalizes well to previously unseen query patterns. As expected, data-driven models exhibit greater resilience to distributional drift due to their explicit modeling of the joint data distribution. In contrast, query-driven methods tend to overfit to the observed workload, resulting in noticeable performance degradation as the distribution shift increases. For example, MSCN suffers a substantial drop in accuracy when the original workload is fully replaced.

These findings highlight the limitations of purely query-driven estimators and underscore CoLSE's effectiveness in maintaining stable accuracy under evolving workloads.

E. Impact of Data Skew

1) Accuracy Comparison: CoLSE maintains strong and consistent accuracy across varying levels of data skew. While it does not always outperform every baseline in absolute

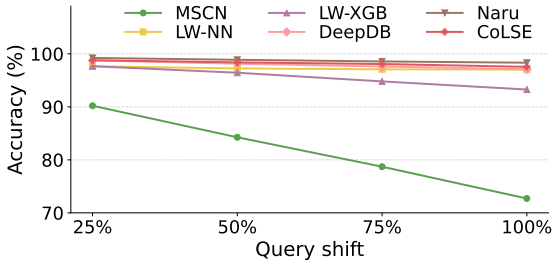
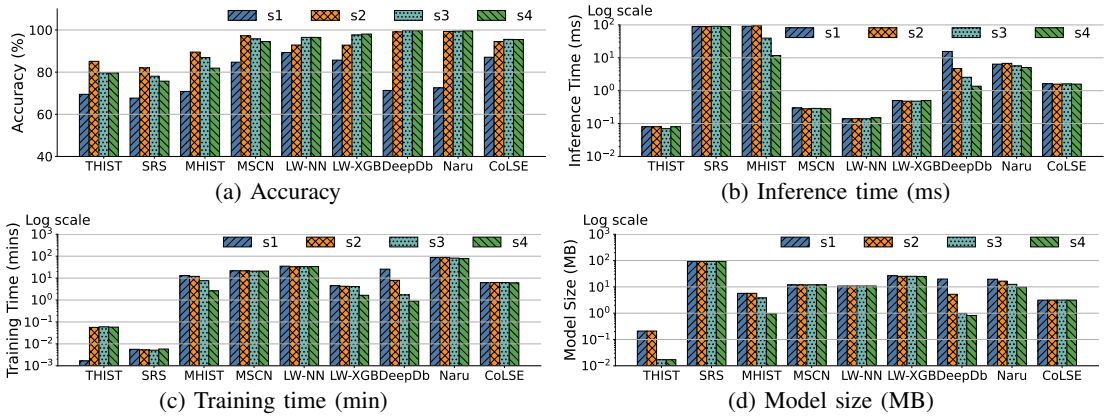


Fig. 7: Performance of models with dynamic workload distribution shifts on the DMV dataset



(a) Accuracy (b) Inference time (ms)

(c) Training time (min) (d) Model size (MB)

Fig. 8: Varying skew levels

accuracy, its performance remains competitive and stable, even under challenging low-skew scenarios, making it a reliable choice across diverse distributions.

As skew increases, most learned models—especially data-driven ones like DeepDb and Naru—show improved accuracy, benefiting from their ability to learn structured patterns in non-uniform data. For instance, DeepDb improves from roughly 77% to 83.5% as skew increases. Conversely, traditional methods degrade under skew due to their limited modeling capacity. Notably, MSCN, a query-driven model, fails to capitalize on skewness, likely due to its reliance on sampling-based features. Meanwhile, LW-XGB, though highly accurate at low skew, shows reduced robustness at higher skew levels.

Data skewness clearly has a notable impact on estimation performance. Interestingly, all models, including learned ones, perform worst on the dataset with minimal skew. This dataset contains a high number of distinct values with low true cardinalities, leading the optimizer to favor index scans, which are highly sensitive to estimation errors. Even minor inaccuracies in such settings can result in suboptimal plan choices and significant deviations.

2) Inference Latency Comparison: As shown in Fig. 8b, CoLSE remains under 2 ms across skew levels, positioning it near query-driven methods, which are most consistent at ≤0.5 ms. DeepDB is highest at low skew (∼16 ms at s1) but drops below 2 ms by s4, as fewer distinct values simplifies the distribution and yields smaller, more efficient models. Naru follows the same trend with moderately higher latency (∼7 to 5 ms). Traditional methods generally exhibit higher inference times compared to learned approaches.

3) Training Time Comparison: Fig. 8c shows that CoLSE trains quickly and consistently (∼6 min across skews), making it one of the most efficient learned models. Expectedly, traditional methods are faster overall. Among learned baselines, Naru is the slowest (∼90 to 75 min from s1 to s4), while DeepDB drops sharply (28 min to ≤5 min). LW-NN, LW-XGB, and MSCN show moderate, stable times (20–35 min).

4) Model Size Comparison: As shown in Fig. 8d, CoLSE is the most compact, remaining ≤4 MB across all skew levels. Conversely, LW-XGB is largest and stays high regardless of skew. DeepDB and Naru shrink as skew increases, mirroring their training-time trends.

F. Impact of Dataset Size

Section VII-A shows that larger datasets inflate training/inference time and model size for learned methods (though they remain more accurate than traditional baselines). Hence, this evaluation focuses solely on learned models to assess their performance on large-scale datasets. Naru is omitted for 10 and 20 GB datasets since one epoch on 10 GB took ∼22 h, indicating poor scalability beyond 10 GB.

1) Accuracy Comparison: Fig. 9a shows that CoLSE maintains over 95% accuracy even on larger datasets, matching or surpassing other learned methods. On the 10 GB and 20 GB datasets, it outperforms MSCN and DeepDb.

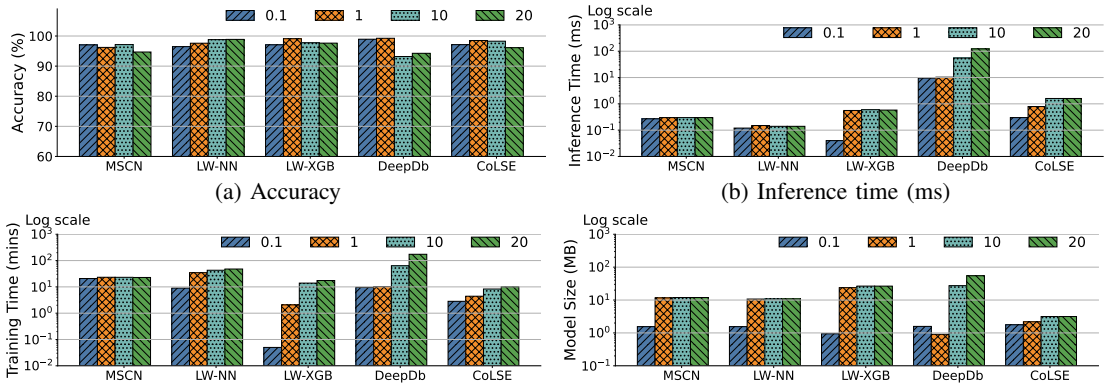
2) Inference Latency Comparison: Fig. 9b compares inference times. Even with larger datasets, query-driven models remain near-zero. DeepDB escalates with size, reaching ∼125 ms at 20 GB, reflecting high computational complexity during predication and limiting practicality. CoLSE offers a favorable trade-off—much lower latency than data-driven methods while remaining competitive with query-driven ones.

3) Training Time Comparison: Training time generally increases with dataset size, except for MSCN, which remains roughly constant at ∼20 minutes across all scales (Fig. 9c). DeepDB scales poorly, rising sharply to ∼175 minutes (∼3 hours) at 20 GB. In contrast, LW-NN grows gradually from ∼10 to ∼50 minutes. LW-XGB and CoLSE also increase with size but remain moderate, reaching ∼20 minutes and ∼10 minutes, respectively, at 20 GB.

4) Model Size Comparison: As Fig. 9d depicts, model sizes generally grow with dataset size. CoLSE remains smallest and stable (∼2–3 MB), DeepDB expands steeply (>55 MB at 20 GB), and among query-driven methods LW-XGB grows most (∼27 MB at 10–20 GB) despite its speed advantages.

G. Sensitivity Analysis

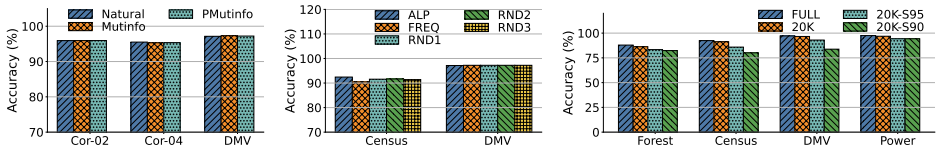
Our model includes several parameters associated with its two components. We first focus on those related to the joint probability estimator. As explained in Section V-C, the Gumbel copula is selected as the copula family for all pair-copula modeling. Table II supports this choice, as



(a) Accuracy (b) Inference time (ms)

(c) Training time (min) (d) Model size (MB)

Fig. 9: Varying dataset sizes



(a) (b) (c)

Fig. 10: Sensitivity study: Impact of (a) attribute ordering and (b) label encoding variants on JPE performance, and (c) Impact of sparse training workloads on ECN

Model	E2E Time (s)	Infer Time (ms)	Training Time (min)	Model size (MB)
ASM	3.97	9.84	38	42
PRICE	1.43	20.13	10	42
CoLSE	6.83	1.25	05	28

TABLE III: Performance comparison on joins

TABLE II: Performance of different copula types on different correlation structures

Column groups	Frank	Clayton	Gumbel
[0, 1, 2]	1.03E-03	1.18E-04	**8.28E-05**
[2, 3, 4]	8.82E-04	8.29E-04	**7.68E-04**
[5, 6, 7]	9.35E-04	1.40E-04	**1.16E-04**
[8, 9, 10]	6.47E-04	3.51E-04	**3.26E-04**
[4, 5, 6]	0.001	0.001	**0.001**
[7, 8, 9]	0.001	0.001	**0.001**

Gumbel consistently yields the lowest error across different correlation structures compared to other popular Archimedean copulas [51], namely Frank and Clayton.

Attribute Ordering in JPE. We assess the impact of attribute order on JPE performance using three strategies from [7]: MutInfo (maximizing mutual information with the selected set), PMutInfo (mutual information with only the most recent attribute), and the Natural ordering (schema order). Experiments on synthetic datasets (pairwise correlations 0.2 and 0.6) and the DMV dataset show only minor differences (see Fig. 10a). Interestingly, the natural ordering is also effective, likely reflecting a human bias to place key columns early, thereby reducing the uncertainty of subsequent attributes [7]. Since MutInfo and PMutInfo are expensive and scale poorly, while Natural incurs no extra computation, we adopt Natural as the practical default.

Categorical Attribute Encoding Schemes. Constructing a CDF for categorical variables requires an imposed order, which is arbitrary for nominal attributes –a known limitation [58], [59]. To mitigate bias from fixed numeric codes, our spline-based dequantization (Section V-E) uses the order only to define CDF steps, not for direct embedding, thus avoiding artificial ordinal distances [60]. We assessed the effect of ordering using alphabetical, frequency-based, and random permutations on two real-world datasets (Census and DMV). Results were consistent, showing negligible impact on performance. For practicality and reproducibility, we thus default to alphabetical ordering.

Error Bounds for JPE. We estimate empirical error bounds for JPE using a two-stage approach: bootstrapped confidence intervals of log-scale multiplicative errors per dataset, followed by a random-effects meta-analysis across datasets. The resulting bounds–1.09 to 2.89–indicate that, on average, estimates may vary by 1.1× to 2.9× from true values.

We evaluated several architectures for the error compensation network, 512_256, 256_256_256, 256_256_128_128, 256_256_64, and 256_256_128_64, by comparing validation loss on the Power and Forest datasets. The 256_256_128_64 configuration yielded the lowest average loss using a batch size of 32, learning rate of 0.001, and 25 training epochs. We selected this architecture for all datasets without further tuning, as it balances accuracy, model size, and training time. This is also aligned with the task's nature: correcting residual errors from the joint estimator generally requires less model complexity than learning the full joint distribution.

To evaluate ECN under sparse workloads, we retrain it using (i) 25% of the queries (20K) and (ii) dimensionality-aware sparsification, keeping a fraction p^d of queries with dimensionality d. As shown in Fig. 10c, accuracy drops only slightly with 20K queries and moderately under sparsification (especially at $p = 0.9$), indicating that ECN generalizes well and remains robust even with limited training data.

H. Evaluation of Join Extension

The join extension was evaluated on the IMDB Job-light workload (70 queries over 6 tables) [61], with comparisons to two state-of-the-art baselines, ASM [10] and PRICE [62].

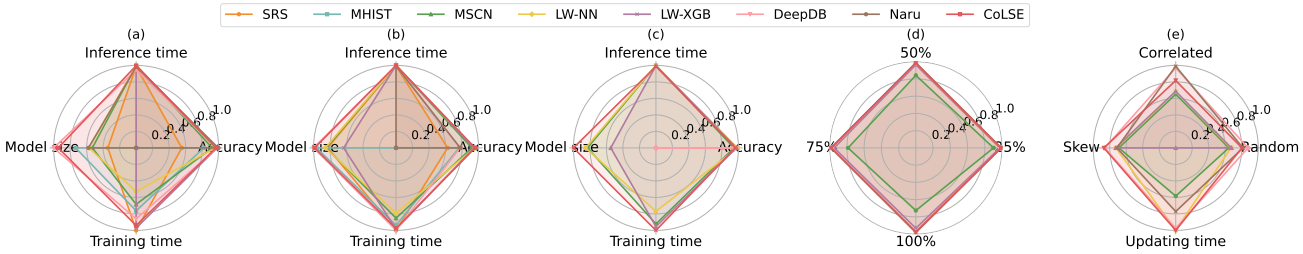


Fig. 11: Summary of experimental evaluation: (a) on a real-world dataset (DMV), (b) on a correlated dataset with a pairwise correlation coefficient of 0.2, (c) on a large-scale dataset of size 20 GB, (d) under query workload shift (using DMV) and (e) with different data updates applied to DMV dataset

According to the results in Table III, the extended CoLSE achieves the lowest end-to-end execution time (356 s) compared with ASM (374 s) and PRICE (366 s). Inference latency was markedly lower at 1.25 ms per query, versus 9.84 ms for ASM and 20.13 ms for PRICE. Training was also faster, completing in 5 min compared to 38 min for ASM and 10 min for PRICE, while the model size remained compact (28 MB vs. 42 MB for both baselines). Overall, these results demonstrate strong efficiency and scalability without compromising accuracy. Future work will consider extensions to multi-attribute and non-equi joins.

The latest work on join cardinality estimation, LpBound [63], computes upper bounds for cardinality. While not a direct competitor, we evaluate CoLSE with the q-error–style metric suggested in the LpBound paper. CoLSE consistently outperforms LpBound: median error drops from 5.25 to 1.62 ($\sim$3.2×), 75th percentile from 19.05 to 2.12 ($\sim$9.0×), and max from 63.1 to 10.33 ($\sim$6.1×), with lower variability (IQR: 1.39–2.12 vs. 2.63–19.05). These gaps reflect design choices: LpBound always overestimates; whereas CoLSE occasionally underestimates but achieves substantially higher accuracy and stability overall.

I. Evaluation Insights

To comprehensively compare CoLSE with baselines, we evaluate accuracy, training time, inference latency, and model size. Since these metrics differ in scale and optimization direction, we normalize all to [0,1] for fair spider chart visualization, applying min-max normalization for metrics where higher is better (e.g., accuracy) and inverting the scale for those where lower is better (e.g., inference and training time).

The spider charts in Fig. 11 illustrate CoLSE's performance trade-offs and robustness across a range of scenarios. In both Fig. 11a and Fig. 11b, CoLSE achieves faster training, smaller model size, and lower inference latency than all baselines, while sacrificing only a small amount of accuracy. In contrast, other models typically degrade significantly in at least one of these dimensions when trying to match CoLSE's accuracy.

Under correlated data distributions, CoLSE shows only a slight dip in accuracy while maintaining resource efficiency, unlike most baseline models. When the query workload distribution shifts (Fig. 11d), CoLSE continues to perform consistently, demonstrating robustness on par with data-driven methods—an area where query-driven models tend to falter.

On large datasets, CoLSE maintains, or even improves, its accuracy with only modest increases in training time and model size. In contrast, data-driven baselines face significantly higher resource costs. Beyond aggregate metrics, CoLSE remains robust under dynamic workloads: it preserves strong performance after data updates even without retraining and, while not always best post-retraining, consistently outperforms query-driven methods and leads all models on skewed data appends. Overall, CoLSE offers a uniquely balanced mix of accuracy, scalability, and robustness.

These findings also underscore the limitations of current learned estimators. Data-driven models often achieve high accuracy but struggle with high inference latency and unpredictable memory usage. Query-driven methods are inference-efficient but depend on pre-collected workloads and lack generalizability. For example, LW-XGB trains quickly but uses significant memory, while Naru delivers high accuracy but is extremely slow to train. Moreover, the need for extensive hyperparameter tuning limits their practicality in DBMSs.

Overall, CoLSE bridges data- and query-driven approaches by combining their strengths, which helps it to deliver competitive accuracy with sub-2 ms inference, $\leq$15 min training, and $\leq$4 MB model size—without hyperparameter tuning. Crucially, CoLSE scales cleanly to large datasets (tens of GB) with only modest resource growth and remains robust under correlations, workload shifts, and skewed data appends, often without retraining, making it a practical deployable choice for modern DBMSs.

VIII. Conclusion

CoLSE presents a novel hybrid architecture for single-table cardinality estimation, integrating data-driven modeling with query-driven correction. Its core innovation lies in approximating joint distributions through CDFs and pairwise D-vine copulas, avoiding the complexity of high-dimensional PDFs while preserving key attribute dependencies.

This is complemented by an error compensation network that adjusts estimates based on query workload signals, enhancing accuracy without the need for hyperparameter tuning. Together, these components allow CoLSE to achieve a strong balance of accuracy, efficiency, and scalability, outperforming existing methods in diverse scenarios including skewed, correlated, and shifting workloads.

ACKNOWLEDGMENTS

This work is partially supported by the Australian Research Council (ARC) via Discovery Early Career Researcher Award DE230100366.

IX. AI-GENERATED CONTENT ACKNOWLEDGEMENT

The authors used ChatGPT to assist with English language editing, improving the grammar, and clarity of text originally written by the authors. Furthermore, ChatGPT was used during code development to assist with basic syntax and implementation details. No experimental results or novel research content were generated solely by AI tools. The authors take full responsibility for the accuracy, originality, and integrity of all content presented in this paper.

REFERENCES

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

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

[3] J. Wang, C. Chai, J. Liu, and G. Li, "Face: A normalizing flow based cardinality estimator," *Proceedings of the VLDB Endowment*, vol. 15, no. 1, pp. 72–84, 2021.

[4] H. Lan, Z. Bao, and Y. Peng, "A survey on advancing the dbms query optimizer: Cardinality estimation, cost model, and plan enumeration," *Data Science and Engineering*, vol. 6, pp. 86–101, 2021.

[5] X. Wang, C. Qu, W. Wu, J. Wang, and Q. Zhou, "Are we ready for learned cardinality estimation?" *arXiv preprint arXiv:2012.06743*, 2020.

[6] A. Dutt, C. Wang, A. Nazi, S. Kandula, V. Narasayya, and S. Chaudhuri, "Selectivity estimation for range predicates using lightweight models," *Proceedings of the VLDB Endowment*, vol. 12, no. 9, pp. 1044–1057, 2019.

[7] Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, X. Chen, P. Abbeel, J. M. Hellerstein, S. Krishnan, and I. Stoica, "Deep unsupervised cardinality estimation," *arXiv preprint arXiv:1905.04278*, 2019.

[8] K. Lee, A. Dutt, V. Narasayya, and S. Chaudhuri, "Analyzing the impact of cardinality estimation on execution plans in microsoft sql server," *Proceedings of the VLDB Endowment*, vol. 16, no. 11, pp. 2871–2883, 2023.

[9] Z. Wu, P. Negi, M. Alizadeh, T. Kraska, and S. Madden, "Factorjoin: a new cardinality estimation framework for join queries," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–27, 2023.

[10] K. Kim, S. Lee, I. Kim, and W.-S. Han, "Asm: Harmonizing autoregressive model, sampling, and multi-dimensional statistics merging for cardinality estimation," *Proceedings of the ACM on Management of Data*, vol. 2, no. 1, pp. 1–27, 2024.

[11] R. Zhu, T. Zeng, A. Pfadler, W. Chen, B. Ding, and J. Zhou, "Glue: Adaptively merging single table cardinality to estimate join query size," *arXiv preprint arXiv:2112.03458*, 2021.

[12] G. Cormode, M. Garofalakis, P. J. Haas, C. Jermaine *et al.*, "Synopses for massive data: Samples, histograms, wavelets, sketches," *Foundations and Trends® in Databases*, vol. 4, no. 1–3, pp. 1–294, 2011.

[13] R. Zhu, Z. Wu, Y. Han, K. Zeng, A. Pfadler, Z. Qian, J. Zhou, and B. Cui, "Flat: fast, lightweight and accurate method for cardinality estimation," *arXiv preprint arXiv:2011.09022*, 2020.

[14] V. Poosala, *Histogram-based estimation techniques in database systems*. The University of Wisconsin-Madison, 1997.

[15] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita, "Improved histograms for selectivity estimation of range predicates," *ACM Sigmod Record*, vol. 25, no. 2, pp. 294–305, 1996.

[16] Y. Han, Z. Wu, P. Wu, R. Zhu, J. Yang, L. W. Tan, K. Zeng, G. Cong, Y. Qin, A. Pfadler *et al.*, "Cardinality estimation in dbms: A comprehensive benchmark evaluation," *arXiv preprint arXiv:2109.05877*, 2021.

[17] S. Hasan, S. Thirumuruganathan, J. Augustine, N. Koudas, and G. Das, "Deep learning models for selectivity estimation of multi-attribute queries," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1035–1050.

[18] Y. Park, S. Zhong, and B. Mozafari, "Quicksel: Quick selectivity learning with mixture models," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1017–1033.

[19] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper, "Learned cardinalities: Estimating correlated joins with deep learning," *arXiv preprint arXiv:1809.00677*, 2018.

[20] B. Hilprecht, A. Schmidt, M. Kulessa, A. Molina, K. Kersting, and C. Binnig, "Deepdb: Learn from data, not from queries!" *arXiv preprint arXiv:1909.00607*, 2019.

[21] M. Heimel, M. Kiefer, and V. Markl, "Self-tuning, gpu-accelerated kernel density models for multidimensional selectivity estimation," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015, pp. 1477–1492.

[22] P. Wu and G. Cong, "A unified deep model of learning from both data and queries for cardinality estimation," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2009–2022.

[23] Y. Lin, Z. Xu, Y. Zhang, Y. Li, and J. Zhang, "Cardinality estimation with smoothing autoregressive models," *World Wide Web*, vol. 26, no. 5, pp. 3441–3461, 2023.

[24] K. Kim, J. Jung, I. Seo, W.-S. Han, K. Choi, and J. Chong, "Learned cardinality estimation: An in-depth study," in *Proceedings of the 2022 international conference on management of data*, 2022, pp. 1214–1227.

[25] P. Li, W. Wei, R. Zhu, B. Ding, J. Zhou, and H. Lu, "Alece: An attention-based learned cardinality estimator for spj queries on dynamic workloads," *Proc. VLDB Endow.*, vol. 17, no. 2, p. 197–210, 2023. [Online]. Available: https://doi.org/10.14778/3626292.3626302

[26] W. contributors, "Copula (statistics)," 2025, accessed: 2025-05-03. [Online]. Available: \url{https://en.wikipedia.org/wiki/Copula_(statistics)}

[27] P. Negi, R. Marcus, H. Mao, N. Tatbul, T. Kraska, and M. Alizadeh, "Cost-guided cardinality estimation: Focus where it matters," in *2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW)*, 2020, pp. 154–157.

[28] P. Negi, R. Marcus, A. Kipf, H. Mao, N. Tatbul, T. Kraska, and M. Alizadeh, "Flow-loss: Learning cardinality estimates that matter," *arXiv preprint arXiv:2101.04964*, 2021.

[29] V. Poosala and Y. E. Ioannidis, "Selectivity estimation without the attribute value independence assumption," in *VLDB*, vol. 97, 1997, pp. 486–495.

[30] M. Muralikrishna and D. J. DeWitt, "Equi-depth multidimensional histograms," in *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, 1988, pp. 28–36.

[31] A. Aboulnaga and S. Chaudhuri, "Self-tuning histograms: Building histograms without looking at data," *ACM SIGMOD Record*, vol. 28, no. 2, pp. 181–192, 1999.

[32] N. Bruno, S. Chaudhuri, and L. Gravano, "Stholes: A multidimensional workload-aware histogram," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, pp. 211–222.

[33] Y.-L. Wu, D. Agrawal, and A. El Abbadi, "Applying the golden rule of sampling for query estimation," *ACM SIGMOD Record*, vol. 30, no. 2, pp. 449–460, 2001.

[34] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "Made: Masked autoencoder for distribution estimation," in *International conference on machine learning*. PMLR, 2015, pp. 881–889.

[35] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *Advances in neural information processing systems*, vol. 34, pp. 15 908–15 919, 2021.

[36] L. Getoor, B. Taskar, and D. Koller, "Selectivity estimation using probabilistic models," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, pp. 461–472.

[37] K. Tzoumas, A. Deshpande, and C. S. Jensen, "Lightweight graphical models for selectivity estimation without independence assumptions," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 852–863, 2011.

[38] M. Scanagatta, A. Salmerón, and F. Stella, "A survey on bayesian network structure learning from data," *Progress in Artificial Intelligence*, vol. 8, no. 4, pp. 425–439, 2019.

[39] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska, "Bao: Making learned query optimization practical," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1275–1288.

[40] L. Woltmann, J. Thiessat, C. Hartmann, D. Habich, and W. Lehner, "Fastgres: Making learned query optimizer hinting effective," *Proceedings of the VLDB Endowment*, vol. 16, no. 11, pp. 3310–3322, 2023.

[41] X. Xu, Z. Zhao, T. Zhang, R. Kang, L. Sun, and J. Chen, "Coool: A learning-to-rank approach for sql hint recommendations," *arXiv preprint arXiv:2304.04407*, 2023.

[42] R. Zhu, W. Chen, B. Ding, X. Chen, A. Pfadler, Z. Wu, and J. Zhou, "Lero: A learning-to-rank query optimizer," *arXiv preprint arXiv:2302.06873*, 2023.

[43] C. K. Ling, F. Fang, and J. Z. Kolter, "Deep archimedean copulas," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1535–1545, 2020.

[44] Hudson & Thames Quantitative Research. (2024) A practical introduction to vine copula. ArbitrageLab documentation, version 1.0.0. [Online]. Available: https://hudson-and-thames-arbitragelab.readthedocs-hosted.com/en/latest/copula_approach/vine_copula_intro.html

[45] R. Wicklin. (2021) An introduction to simulating correlated data by using copulas. *The DO Loop* (SAS Blogs), SAS Institute Inc. [Online]. Available: https://blogs.sas.com/content/iml/2021/07/05/introduction-copulas.html

[46] C. Czado and T. Nagler, "Vine copula based modeling," *Annual Review of Statistics and Its Application*, vol. 9, pp. 453–477, 2022, first published as a Review in Advance on November 2, 2021. [Online]. Available: https://tnagler.github.io/vine-arisa.pdf

[47] Wikipedia contributors, "Probability integral transform," May 2025, accessed on 2025-05-03. [Online]. Available: \url{https://en.wikipedia.org/wiki/Probability_integral_transform}

[48] K. Aas, C. Czado, A. Frigessi, and H. Bakken, "Pair-copula constructions of multiple dependence," *Insurance: Mathematics and economics*, vol. 44, no. 2, pp. 182–198, 2009.

[49] E. C. Brechmann and U. Schepsmeier, "Modeling dependence with c- and d-vine copulas: the r package cdvine," *Journal of statistical software*, vol. 52, pp. 1–27, 2013.

[50] C. Czado and T. Nagler, "Vine copula based modeling," *Annual Review of Statistics and Its Application*, vol. 9, no. 1, pp. 453–477, 2022.

[51] C. Genest and L.-P. Rivest, "Statistical inference procedures for bivariate archimedean copulas," *Journal of the American statistical Association*, vol. 88, no. 423, pp. 1034–1043, 1993.

[52] J. Górecki, M. Hofert, and M. Holeňa, "An approach to structure determination and estimation of hierarchical archimedean copulas and its application to bayesian classification," *Journal of Intelligent Information Systems*, vol. 46, no. 1, pp. 21–59, 2016.

[53] S. S. Sane, "The inclusion-exclusion principle," in *Combinatorial Techniques*. Springer, 2013, pp. 57–79.

[54] F. Zirak, "Tpch-skew for linux," https://github.com/fzirak/tpch-skew-linux, 2025, accessed: 2025-05-03.

[55] PostgreSQL Global Development Group. (2024) Row estimation examples. PostgreSQL. [Online]. Available: https://www.postgresql.org/docs/current/row-estimation-examples.html

[56] The PostgreSQL Global Development Group, *PostgreSQL 13.20 Documentation*, PostgreSQL Global Development Group, 2025, accessed: 2025-05-03. [Online]. Available: https://www.postgresql.org/docs/13/index.html

[57] H. Suzuki, "3.2. cost estimation in single-table query," https://www.interdb.jp/pg/pgsql03/02.html, 2024, accessed: 2025-05-03.

[58] T. Mizuno and C. Deutsch, "Sequential indicator simulation (sis)," in *Geostatistics Lessons*, J. Deutsch, Ed. Centre for Computational Geostatistics, 2022. [Online]. Available: http://www.geostatisticslessons.com/lessons/sequentialindicatorsim

[59] J. Bois, "Categorical distribution — probability distribution explorer," https://distribution-explorer.github.io/discrete/categorical.html?utm_source=chatgpt.com, 2025, last updated August 09, 2025.

[60] P. K. Dunn and G. K. Smyth, "Randomized quantile residuals," *Journal of Computational and graphical statistics*, vol. 5, no. 3, pp. 236–244, 1996.

[61] A. Kipf. Job-light workload. [Online]. Available: https://github.com/andreaskipf/learnedcardinalities/blob/master/workloads/job-light.sql

[62] T. Zeng, J. Lan, J. Ma, W. Wei, R. Zhu, P. Li, B. Ding, D. Lian, Z. Wei, and J. Zhou, "Price: a pretrained model for cross-database cardinality estimation," *arXiv preprint arXiv:2406.01027*, 2024.

[63] H. Zhang, C. Mayer, M. Abo Khamis, D. Olteanu, and D. Suciu, "Lpbound: Pessimistic cardinality estimation using ℓp-norms of degree sequences," *Proceedings of the ACM on Management of Data*, vol. 3, no. 3, pp. 1–27, 2025.