Prediction Intervals for Learned Cardinality Estimation: An Experimental Evaluation

Saravanan Thirumuruganathan
QCRI, HBKU
sthirumuruganathan@hbku.edu.qa

Suraj Shetiya
UT Arlington
suraj.shetiya@mavs.uta.edu

Nick Koudas
University of Toronto
koudas@cs.toronto.edu

Gautam Das
UT Arlington
gdas@cse.uta.edu

2022 IEEE 38th International Conference on Data Engineering (ICDE) | 978-1-6654-0883-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDE53745.2022.00274

***Abstract*—Cardinality estimation is a fundamental and challenging problem in query optimization. Recently, a number of learned models have been proposed for this task. Often, these models significantly outperform traditional approaches in terms of accuracy. One of the stumbling blocks that prevents their increased adoption is that the learned models do not quantify the uncertainty of their estimates. It is desirable to associate each cardinality estimate of the model with a *prediction interval* that will contain the true cardinality with an user-specified probability. The size of the prediction interval encodes the uncertainty allowing the query optimizer to make an informed decision. For example, knowing that the cardinality of a query q lies between $1-3\%$ of the relation size with high probability is more informative than a single point estimate of 2%. While there has been some prior work on deriving bounds for traditional methods (such as sampling or histograms), they are not directly applicable for the learned models for cardinality estimation.**

In this paper, we conduct a systematic investigation of potential approaches for obtaining prediction intervals. We enumerate the list of desirable properties such as the ability to wrap around a learned model without significant internal modification and providing bounds with theoretical guarantees in a distribution agnostic manner among others. Based on an extensive literature survey, we identify *four* practical and high quality approaches for uncertainty quantification that satisfies these criteria. They span a wide spectrum in terms of theoretical guarantees, width of prediction interval and time taken for computing the prediction intervals. We conduct extensive experimental analysis of the efficacy of these approaches over *three* diverse and representative cardinality estimation algorithms. Our experiments covers diverse workloads involving both point and range queries and highlights the inherent trade-offs. Our results show that it is possible to obtain accurate prediction intervals in an efficient manner thereby opening up new avenues for future research.

I. Introduction

Cardinality estimation – the problem of estimating the number of tuples that satisfy the predicates of a given query – is an important and challenging problem in query optimization. The cardinality estimates could be used in diverse applications including query optimization, database tuning, approximate query processing etc. Inaccurate cardinality estimates could result in the selection of sub-optimal plans resulting in poor performance [24]. Hence, there has been extensive work by the database community using diverse traditional and machine learning based techniques. Recently, there has been increasing interest in replacing database components with *learned models* that are often based on deep learning (DL). Several recent works [34], [51], [16], [45] have shown that it is possible to develop learned (DL) models that achieve significantly better accuracy than traditional methods such as histograms.

The learned models for cardinality estimation could be broadly partitioned into two categories – learned *query* models [22], [8], [53], [17], [21], [38], [54] and learned *data* models [57], [17], [56], [19], [62]. Given a query q, the learned query model seeks to learn a mapping between the features of q and the cardinality of q. Typically, this is achieved by training a *supervised* non-linear regression model over a training workload of queries and their cardinalities. The learned data models formulate cardinality estimation as the problem of joint probability distribution estimation. Typically, these models are directly trained over the data in an *unsupervised* manner using appropriate architectures such as autoregressive models [57], [17], [56], sum-product networks [19], [62] among others.

Limitations of Learned Models. Despite superior performance over traditional methods, the learned models could still produce sub-optimal results for queries with complex predicates involving multiple tables. Recent work such as [51], [33] have critically examined their promises and pitfalls. The learned models are not a silver bullet and can be quite fallible. They could have high estimation errors due to diverse reasons including model capacity, quality/informativeness of query featurization, quality of training data [33], and high inter-column and inter-table correlations [27]. The performance is also affected by dataset and workload characteristics such as correlation, skewness, or domain size [51]. These limitations affect both supervised and unsupervised learned models. The supervised models are affected by the divergence between training and testing workloads [51], [33]. The data-driven unsupervised estimators are affected by skewed data [51], [27]. Furthermore, they often underestimate cardinalities for range queries [61] as they rely on Monte-Carlo integration over the learned distribution [57], [17]. These estimation errors are extenuated by the black-box nature of the learned models that makes analyzing their behavior much more challenging [51].

Quantifying Uncertainty of Cardinality Estimates. Traditional methods such as sampling, often provide some measure of uncertainty through variance or confidence intervals [31]. In contrast, the learned models output a single scalar as the cardinality estimate. The lack of uncertainty quantification associated with the estimate makes the job of query optimizer much more challenging as it has to select a specific plan

2375-026X/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDE53745.2022.00274

based on the estimate that could turn out to be incorrect. However, if the learned model could output some additional information (such as an interval), then the query optimizer could make an informed decision. For example, if the interval is sufficiently tight, then the optimizer could choose the plan with higher confidence. Uncertainty quantification could improve the reliability of the cardinality estimates and increase the adoption of learned components.

Prediction Intervals. Given a query q, the learned model outputs an estimate $Est(q)$. A prediction interval $[low_q, high_q]$ ensures that

$$P(low_q \leq Card(q) \leq high_q) \geq (1 - \alpha)$$

for some user specified coverage level $1 - \alpha$. Here $Card(q)$ denotes the true cardinality of query q. We provide a formal definition in Section III. To be useful, prediction intervals should have some desirable properties such as providing tight bounds with rigorous guarantees without making any strong assumptions about the model or distribution of the data or query workload. Furthermore, it is desirable that we should be able to *wrap* any arbitrary black-box learned model without requiring any internal modifications. Ideally, the prediction interval would subsume the uncertainties caused by different sources including data, model and workload.

Contributions. We conduct a systematic analysis of various techniques that could provide accurate prediction intervals. We identify *four* practical and high quality approaches for uncertainty quantification that share a number of desirable properties – (a) jackknife+ with cross validation [4], (b) split conformal inference [35], [23], (c) locally weighted conformal inference [23], [2] and (d) conformalized quantile regression [40] Broadly, they can be partitioned into two categories. Jackknife+ belongs to the class of resampling based approaches that has been widely used to obtain bounds for complex queries for sampling based approximate query processing (AQP) [31] . Jackknife+ extends these traditional approaches to provide prediction intervals for the learned models. The remainder are based on the framework on conformal prediction [3], [50] that provides rigorous bounds in an distribution free manner without making any strict assumptions. These four approaches span a wide spectrum in terms of theoretical guarantees, width of prediction interval and time taken for training and estimating the prediction intervals.

We evaluate these methods for obtaining prediction intervals on *three* diverse and representative cardinality estimation algorithms – Naru [57], MSCN [22] and LW-NN [8], [51]. Naru is an unsupervised approach while MSCN and LW-NN are supervised. LW-NN is a lightweight model that is especially optimized for range queries. We conduct extensive experiments over diverse workloads involving both point and range queries. Our results show that it is possible to obtain accurate prediction intervals in an efficient manner. We also provide some guidance about choosing an appropriate approach based on the trade-offs between accuracy of prediction intervals and the required inference time.

II. Learned Models for Cardinality Estimation

Cardinality Estimation. Let R be a relation with N tuples and M attributes $A = \{A_1, A_2, \ldots, A_M\}$. The domain of the attribute A_i is given by $Dom(A_i)$. If A_i is a numerical attribute, then $Dom(A_i)$ is specified by the range $[\min_i, \max_i]$. If A_i is categorical, then $Dom(A_i)$ is $\{a_{i,1}, \ldots, a_{i,|Dom(A_i)|}\}$. We denote the value of attribute A_i of an arbitrary tuple t as $t[A_i]$. We consider conjunctive queries on R of the form

```
SELECT COUNT(*) FROM R
WHERE P1 AND P2 AND ...
```

P_i could be either a point or range predicate. Point predicates are of the form $A_i = v_i$ where $v_i \in Dom(A_i)$ while range predicates are specified as an interval $lb_i \leq A_i \leq ub_i$. The goal of cardinality estimation is to accurately estimate the number of tuples $Card(q)$ that satisfy the predicates of query q. We use the normalized selectivity between $[0, 1]$ obtained by dividing $Card(q)$ by the number of tuples, N.

Performance Measures. Given a query q, let the estimate provided by selectivity estimation algorithm be $Est(Q)$. We use *q-error* which measures the quality of estimates that as the factor by which the estimate differs from true cardinality. This metric is widely used for evaluating cardinality estimation approaches [22], [53], [21], [17], [42].

$$\text{q-error} = \max\left(\frac{Est(Q)}{Card(Q)}, \frac{Card(Q)}{Est(Q)}\right) \tag{1}$$

Taxonomy of Learned Models. Recently, a number of papers have successfully applied deep learning (or machine learning in general) for cardinality estimation. Broadly, they could be partitioned into two major categories. The *query-driven* approaches are often based on supervised learning and are trained on a workload of queries and their true cardinalities. Typically, this involves three stages [51]. The *query featurization* stage represents each query as a set of features through diverse techniques such as one-hot encoding [21], [17], binary encoding [17], basic statistics [8], data samples [21] and even heuristic estimators [8]. In the *training* stage, a (non-linear) regression model is trained over the workload to estimate the cardinality of a query given its features. In the *inference* stage, the regression model is applied on the features of the input query. In contrast, the *data-driven* approaches formulate cardinality estimation as the problem of joint probability distribution (i.e. $P(A_1, A_2, \ldots, A_m)$) estimation. Typically, this approach consists of two stages – *training* and *inference*. The training is conducted on the data without the need for training data such as the query workload. In the inference phase, the data-driven models estimate the cardinality of a query q through diverse techniques including Monte-Carlo integration over the learned distribution [57], [17].

We evaluate the performance of the algorithms by selecting three learned cardinality estimation models – MSCN [21], Naru [57] and LW-NN [8]. Independent evaluation such as [51], [16], [45] have found that these approaches achieve competitive performance over various datasets and workloads.

III. Prediction Intervals for Cardinality Estimation

In this section, we describe four techniques that could be used for quantifying the error of the cardinality estimates produced by the learned models.

Problem Setup. Consider the query-driven approaches that train a regression model for estimating cardinalities. Let $\mathcal{D} = \{(X_1, y_1), \ldots, (X_n, y_n)\}$ be a labeled dataset drawn i.i.d. from an unknown distribution that was used to train the model. Here, X_i is the featurized version of query q_i while y_i is the correct cardinality of q_i. Suppose that X_{n+1} is the feature of a *new* query q_{n+1} drawn i.i.d. from the same distribution. Let the true (but unknown) and estimated cardinalities for X_{n+1} be y_{n+1} and $\hat{y}_{n+1}$ respectively. We desire an interval, denoted as $C(X_{n+1}) = [low_{n+1}, high_{n+1}]$, such that

$$P(y_{n+1} \in C(X_{n+1})) \geq (1 - \alpha) \tag{2}$$

for some user specified coverage level $1-\alpha$. α is also referred to as miscoverage level.

A. Quantifying Errors of Learned Models

As mentioned in Section I, the learned models could have large estimation errors due to various reasons. For example, the data could be skewed or the workload used for training the supervised models is not representative of the workloads used for inference. Alternatively, the learned model might not have sufficient capacity to learn the relevant characteristics of the data or workload. Hence, it is important to holistically incorporate these uncertainties in data, model and workload when quantifying the estimation error. For tasks like query optimization, aggregate metrics such as average test accuracy are insufficient. Instead, it is desirable to quantify the uncertainty in the cardinality estimation for each query by providing a range for prediction with some guarantee that the *true* cardinality estimate will fall within the bounds with high probability such as 0.95. This additional information could enable the query optimizer to make more informed decisions.

Uncertainty Framework. Typically, cardinality estimation is formulated as a regression problem by assuming that there exists an underlying data generation function $f(X)$ that generates the target values y as

$$y = f(\mathbf{X}) + \epsilon$$

In the cardinality estimation context, $\mathbf{X}$ is the feature for a query q while y is the true cardinality. ϵ corresponds to the data noise. The supervised learned models seek to learn a non-linear function $\hat{f}$ (such as through deep learning) that can map the features to an estimate. Specifically, given the feature $\mathbf{X}$ for query q, the regression generates a *point* estimate $\hat{f}(\mathbf{X})$. In order to generate the interval, we should have a mechanism to quantify the uncertainty.

Intuitively, there are two sources of uncertainty [20] – model and data. The *aleatoric* uncertainty that captures the inherent noise in the data while the *epistemic* uncertainty captures the uncertainty in the model parameters. Aleatoric uncertainties are data dependent and could be improved by improving the data. In contrast, epistemic uncertainties can be improved with better models and could become negligible in the presence of infinite data. We can measure the overall uncertainty as

$$\sigma_y^2 = \sigma_{\text{model}}^2 + \sigma_{\text{noise}}^2$$

σ_{model}^2 quantifies the model uncertainty that could arise from various sources [39] such as model misspecification (bias), uncertainty in training data/workload (variance) and parameter uncertainty. Additional discussion can be found in [39], [20].

Confidence and Prediction Intervals. There are two approaches that are widely used to quantify the uncertainty. Suppose that we are given an input $\mathbf{X}$ and a desired coverage level $1 - \alpha$. The confidence interval (CI) focuses on the distribution $P(f(\mathbf{X})|\hat{f}(\mathbf{X}))$. In the context of regression, CIs provide a bound that contains the estimated conditional mean value $E[y|\mathbf{X}]$ with confidence $1 - \alpha$. In contrast, prediction intervals (PI) focus on the distribution $P(y|\hat{f}(\mathbf{X})) = P(y|\hat{y})$. We can see that PIs are necessarily wider than CIs as the former only needs to handle the model uncertainty while the latter also has to tackle the data uncertainty [18], [39], [47]. In this paper, we focus on prediction intervals.

Desiderata for Prediction Intervals. We enumerate a list of desirable properties for identifying a promising set of techniques for computing prediction intervals for learned models.

- *Coverage Validity:* The prediction intervals should provide rigorous theoretical guarantees.
- *Interval Efficiency:* We require that the width of interval be as tight as possible. Otherwise, it is possible to provide trivial bounds such as $[0, 1]$ for the normalized cardinality.
- *Minimal Assumptions:* The approach should not make any assumptions about the data or workload distributions.
- *Generality:* The approach should work for arbitrary learned models and arbitrary data and workload distributions. The PIs should subsume the uncertainties caused by different sources including data, model and workload.
- *Minimal to No Changes to Underlying Model.* There has been a wide variety of learned models for cardinality estimation. The approach for PIs should be able to accept any blackbox method without requiring any internal changes. In other words, it should be possible to wrap an arbitrary model and produce meaningful error estimates.

Computing Quantiles. We describe *four* practical approaches that satisfy the desiderata. Each of them relies on computing quantiles. Suppose we are given a set of values $\{v_1, v_2, \ldots, v_n\}$. Then, we denote the $(1 - \alpha)$ quantile as

$$q_{n,1-\alpha}\{v_i\} = \lceil (1-\alpha)(n+1) \rceil\text{-th smallest value of } v_1, \ldots, v_n$$

B. JackKnife+ based Prediction Intervals

Jackknife [4] is a *resampling* based technique that has been used for estimating metrics such as bias and variance of an estimator without making any parameteric assumptions. This method could be extended to obtain prediction intervals for learned models. Recall that $\mathcal{D} = \{(X_1, y_1), \ldots, (X_n, y_n)\}$

is the labeled dataset containing query features and true cardinalities. The goal is to obtain a prediction interval for a new query X_{n+1}. Jackknife tackles this through leave-one-out approach. Instead of building a single model $\hat{f}$ for $\mathcal{D}$, we build n models. Let $\hat{f}_{-i}$ be the model trained over $D \setminus \{(X_i, y_i)\}$ by removing the i-th data point. The leave-one-out residual for that model is computed as $r_i = |y_i - \hat{f}_{-i}(X_i)|$. Let $r = \{r_1, r_2, \ldots, r_n\}$ be the set of residuals computed for models $\hat{f}_{-1}, \ldots, \hat{f}_{-n}$. Finally, we train the model $\hat{f}$ over the entire dataset $\mathcal{D}$. Given a new query X_{n+1}, the prediction interval is computed as the $1-\alpha$-th quantile over the set of leave-one-out residuals r. Specifically,

$$C_{JK}(X_{n+1}) = \Big[q_{n,1-\alpha}\{\hat{f}(X_{n+1}) - r_i\}, q_{n,\alpha}\{\hat{f}(X_{n+1}) + r_i\}\Big] \tag{3}$$

While this approach often works well empirically in practice, this suffers from two key limitations. First, it requires training of n models that is infeasible in our setting where n is large and training even a single learned model takes significant amount of time. More seriously, it does not provide any *universal* theoretical guarantees [4]. Instead, it only provides guarantees under an *asymptotic* setting when the learned model is *stable* [43]. Stability requires that $\hat{f}$ and each of the leave-one-out models $\hat{f}_{-i}$ provides similar predictions for X_{n+1}. In our experiments, we found that learned models do not always satisfy this property.

We rely upon a recent approach dubbed Jackknife+ [4] that extends Jackknife to provide non asymptotic coverage guarantees. By default, Jackknife+ proceeds in the same manner as Jackknife by training n models and computing their corresponding leave-one-out residuals r_i. The key difference is in how the prediction interval is computed as $C_{JK+}(X_{n+1}) =$

$$\Big[q_{n,1-\alpha}\{\hat{f}_{-i}(X_{n+1}) - r_i\}, q_{n,\alpha}\{\hat{f}_{-i}(X_{n+1}) + r_i\}\Big] \tag{4}$$

Often, Jackknife and Jackknife+ provide very similar predication intervals in practice [4]. This is especially true when the learned model for the entire data and the corresponding leave-one-out model produces similar outputs for an arbitrary query. However, while Jackknife might not provide a rigorous PI under certain circumstances, Jackknife+ always provides a bound of $1-2\alpha$ [4]. While this solves the problem of guarantees, the problem of efficiency still remains. However, it is possible to achieve a trade-off between number of trained models and the theoretical guarantees. Specifically, we can reduce the number of models trained by using K-fold cross validation for a user specified K (such as 10). We partition $\mathcal{D}$ into K disjoint subsets $S_1, S_2, \ldots, S_K$ of equal size. Next, we train K models where the model $\hat{f}_{-i}$ is trained by excluding subset S_i. We next compute the K-fold residual as

$$r_i = |y_i - \hat{f}_{S_{k(i)}}(X_i)|$$

$k(i) \in \{1, \ldots, K\}$ denotes the subset containing X_i. We can compute the PI as $C_{CV+}(X_{n+1}) =$

$$\Big[q_{n,1-\alpha}\{\hat{f}_{-S_{k(i)}}(X_{n+1}) - r_i\}, q_{n,\alpha}\{\hat{f}_{-S_{k(i)}}(X_{n+1}) + r_i\}\Big] \tag{5}$$

Note that we now only train K models instead of n models as in Jackknife+. Each of the models are trained using $n\left(1-\frac{1}{K}\right)$ instead of $n-1$ data points. The increased efficiency comes at the cost of wider prediction intervals. It also results in reduced coverage guarantee[4] that is at least

$$1 - 2\alpha - \min\left\{\frac{2(1-1/K)}{n/K+1}, \frac{1-K/n}{K+1}\right\}$$

Algorithm 1 JackKnife+ with Cross Validation [4], [23]

Input: $\mathcal{D}$, K, α
Split D into K disjoint partitions $S_1, \ldots, S_K$
for i = 1 to K **do**
  Train model $\hat{f}_{-i}$ using data $\mathcal{D} \setminus S_i$
for i = 1 to n **do**
  Compute residual $r_i = |y_i - \hat{f}_{S_{k(i)}}(X_i)|$
$\delta = \lceil (n+1)(1-\alpha) \rceil$-th smallest value in $r = \{r_1, \ldots, r_n\}$
Train model $\hat{f}$ over entire dataset $\mathcal{D}$
return $[\hat{f}(X) - \delta, \hat{f}(X) + \delta]$ for any query feature X

C. Conformal Prediction based Approaches

Conformal Prediction (CP) is a powerful and flexible approach for outputting prediction intervals. CP falls under the paradigm of distribution free uncertainty quantification [50], [2], [58], [3] where the goal is to create prediction intervals with rigorous statistical guarantees. The prediction intervals (PI) produced by CP are *valid* – i.e. they are guaranteed to contain the correct cardinality with the user specified coverage level. An appealing property of CP is that it produces the PI using a finite number of data points *without* making any assumptions about the model or the data. We first provide a high level overview of CP while the next three subsections describe concrete instantiations with differing trade-offs.

Assumptions of CP. CP can be used whenever the data are independent and identically distributed (i.i.d.). This is a standard assumption widely used in ML for both regression and classification. In fact, CP can work even with the weaker assumption of *exchangeability*, where the ordering of data points does not affect their joint distributions. A sequence $Z = (z_1, z_2, \ldots, z_n)$ is exchangeable if the joint probability distribution of the permuted sequence Z' is the same as that of the original sequence, i.e. $P(z_1, z_2, \ldots, z_n) = P(\pi(z_1, z_2, \ldots, z_n))$ where $\pi(\cdot)$ is the permutation operator [58]. We can see that exchangeability is a weaker assumption than i.i.d. Every i.i.d. data is exchangeable but the vice versa need not be true. CP do not make any assumption about the ML model and can work for arbitrary models. Hence, CP could be used for supervised DL models such as MSCN [22], unsupervised autogressive models [57], [17] and even models not based on deep learning.

Conformal Prediction. Next, we provide a simplified exposition tailored for the task of cardinality estimation. Additional details can be found in [2], [23], [36], [58], [3]. Intuitively, CP consists of three key steps: fitting, calibration and inference.

Given a labeled dataset $\mathcal{D}$, we split it into two parts: $\mathcal{D}_T$ and $\mathcal{D}_C$. First, we fit a model $\hat{f}(\cdot)$ using the training dataset $\mathcal{D}_T$. Second, we calibrate the output of the model using $\mathcal{D}_C$. The calibration is done through a *scoring function* $s(X, y)$ that produces a *conformal score*. For example, the scoring function could be $|y - \hat{f}(X)|$ that computes the residual error. Intuitively, the score is a proxy for the quality of the prediction with a smaller value indicating a relatively accurate prediction. We compute the scoring function for each $X \in \mathcal{D}_C$. We cannot directly use the conformal scores to produce a prediction interval. The goal of CP is to *calibrate* the conformal scores so that they could be used for computing prediction intervals. It achieves this by computing the $\delta = \lceil (n+1)(1-\alpha) \rceil$-th smallest value of the conformal scores. Suppose $\alpha = 0.1$. Then, at least 90% of the conformal scores in the calibration set are less than δ. Intuitively, by the exchangeability (or the i.i.d.) assumption, we would expect this fact to hold for any new X with probability of at least $1 - \alpha$. Please refer to [2], [23] for a formal proof. Finally, the inference step produces a valid interval for a new point X' using $\hat{f}(X')$ and δ. For example, the prediction interval could contain all y such that score $s(X, y) \leq \delta$ for all $(X, y) \in \mathcal{D}_C$.

Importance of Scoring Functions. CP does not make any model or distributional assumptions as the scoring function abstracts the classifier performance allowing it to be used for arbitrary classifiers. The prediction interval is valid as long as exchangeability assumption holds. It is important to note that the *validity* of the prediction interval holds for arbitrary scoring functions. If the scoring function is informative – giving low scores for accurate estimates and large scores for inaccurate ones – then the prediction interval will provide meaningful and non-trivial bound. The use of different scoring functions could potentially produce different prediction intervals.

D. Split Conformal Prediction

Split conformal prediction (S-CP) is one of the simplest and widely used variant of conformal prediction. It broadly follows the three steps described in Section III-C. It begins by partitioning the labeled dataset $\mathcal{D}$ into two *equal* and disjoint subsets: a training dataset ($\mathcal{D}_T$) and a calibration dataset ($\mathcal{D}_C$). This partitioning is analogous to that of training and validation datasets used for training DL models where the validation dataset is used for tasks such as tuning the model hyper-parameters. Hence, we can use the validation dataset for calibration purposes. Next, we apply the scoring function for each $(X_i, y_i) \in \mathcal{D}_C$ producing a set of conformal scores $r = \{r_i\}$. Given a coverage $1-\alpha$, we compute the appropriate quantile and use it to compute the prediction intervals for any new data point.

Split-CP is computationally efficient as only a single model need to be trained over $\mathcal{D}_C$ instead of n models (in Jackknife+) or K models (in CV-Jackknife+). However, this efficiency comes at a cost. Suppose we did a 50-50 split of $\mathcal{D}$ to get $\mathcal{D}_T$ and $\mathcal{D}_C$. Then, the model $\hat{f}$ is only trained on $n/2$ data points in contrast to $n-1$ and $n - n/K$ in the Jackknife+ variants. Reducing the size of $\mathcal{D}_T$ could potentially result in an inaccurate model. However, while increasing the size of $\mathcal{D}_T$ increases the accuracy of $\hat{f}$, it also increases the variability of the prediction interval. We study this tradeoff in the experiments and show that typical sizes of validation datasets in DL training provides sufficiently accurate prediction intervals.

Algorithm 2 Split Conformal Prediction [35], [23]

Input: $\mathcal{D}$, α
Split $\mathcal{D}$ to two disjoint subsets $\mathcal{D}_T$ and $\mathcal{D}_C$.
Train model $\hat{f}$ using $\mathcal{D}_T$
$r = \{|y_i - \hat{f}(X_i)|\} \qquad \forall (X_i, y_i) \in \mathcal{D}_C$
$\delta = \lceil (n_1+1)(1-\alpha) \rceil$-th smallest value in r where $n_1 = |\mathcal{D}_C|$.
return $[\hat{f}(X) - \delta, \hat{f}(X) + \delta]$ for any query feature X

E. Locally Weighted Split Conformal Prediction

While Split-CP works well in practice, it suffers from a subtle issue. From Algorithm 2, we can see that the size of the prediction interval is a fixed constant δ for all queries. This property is useful if the residual error of the learned model does not vary significantly with X. However, an empirical analysis shows that this assumption does not hold for most learned models for cardinality estimation. For example, the errors for queries with predicates containing highly correlated attributes is often higher than that of non correlated attributes. Hence, it is desirable for the prediction interval to take this non-uniformity of residual errors into account. There are many different ways to formulate the heterogeneous prediction intervals [23], [2]. We focus on normalized non-conformity scores. Intuitively, the idea is to require a prediction interval proportional to the 'difficulty' of the query – narrower PIs for easy queries and wider PIs for challenging ones.

As mentioned above, the key insight is that it is possible to alter the scoring function $S(X, y)$ without losing the coverage guarantees as long as the new scoring function has the exchangeability property. We leverage this flexibility by replacing the residual error based scoring function (i.e. $S(X, y) = |y - \hat{f}(X)|$) with a *normalized* scoring function based on the difficulty of the query thereby enabling *adaptive* PIs. Let $U(X)$ be a function that quantifies the difficulty (or in general uncertainty) associated with X. Hence, a smaller value indicates an easier example. Then, we can replace the *absolute* residuals with *scaled* residuals as

$$\hat{r}_i = \frac{|y_i - \hat{f}(X_i)|}{U(X_i)} = \frac{r_i}{U(X_i)}$$

Then, the PI for a new query X is computed as

$$C(X_i) = [\hat{f}(X_i) - \delta\, U(X_i), \hat{f}(X_i) + \delta\, U(X_i)] \qquad (6)$$

where δ is the prediction interval for the original scoring function (see Algorithm 2). The adaptiveness of the PI now relies on appropriate design of the normalization function. There are many ways to design $U(X)$. For example $U(X)$ could measure the conditional mean absolute deviation (MAD)

of $|Y - \hat{f}(X)|$ for a given X. We can see that the deviation will be higher for harder X while it will be narrow for easier X. Other approaches include setting $U(X)$ as the variance of $\hat{f}(X)$ for an ensemble of learned models all initiated with different random seeds or hyper-parameters. Alternatively, we could generate various 'perturbations' of X such as $X_1, X_2, \ldots$ and measure the variance in predictions i.e. $Var(\{\hat{f}(X_1), \hat{f}(X_2), \ldots\})$.

Instantiating $U(X)$. For concreteness, let $U(X)$ be the conditional mean absolute deviation (MAD) of $|Y - \hat{f}(X)|$. Locally weighted S-CP (LW-S-CP) proceeds as follows. As before, we split $\mathcal{D}$ to $\mathcal{D}_T$ and $\mathcal{D}_C$. We train $\hat{f}$ on $\mathcal{D}_T$ and compute the residual absolute deviation $r_i = |y_i - \hat{f}(X_i)|$ for each $(X_i, y_i) \in \mathcal{D}_T$ (not the calibration set). Next we train a model $\hat{g}$ based on the labeled dataset $\{X_i, r_i\}$ for each $X_i \in \mathcal{D}_T$. In other words, we train a second model $\hat{g}$ to predict the residual for X produced by $\hat{f}$. Next, we compute the scaled residuals for each $X_i \in \mathcal{D}_C$ and use it compute the quantiles. Finally, we apply the modified formula for computing prediction interval provided by Equation 6.

Algorithm 3 Locally Weighted Split Conformal Prediction [23], [2]

Input: $\mathcal{D}$, α
Split $\mathcal{D}$ to two disjoint subsets $\mathcal{D}_T$ and $\mathcal{D}_C$.
Train model $\hat{f}$ using $\mathcal{D}_T$
Train model $\hat{g}$ using $\{X_i, |y_i - \hat{f}(X_i)|\}$ $\forall (X_i, y_i) \in \mathcal{D}_T$
$r = \{|y_i - \hat{f}(X_i)|\}$ $\forall (X_i, y_i) \in \mathcal{D}_C$
$\hat{r} = \{\frac{r_i}{U(X_i)}\}$ $\forall X_i \in \mathcal{D}_C$
$\delta = \lceil (n_1 + 1)(1 - \alpha) \rceil$-th smallest value in r where $n_1 = |\mathcal{D}_C|$.
return $[\hat{f}(X) - \delta\, U(X), \hat{f}(X) + \delta\, U(X)]$ for any X

F. Conformalized Quantile Regression

The locally weighted variant of S-CP (LW-S-CP) works well in practice and has appealing properties. It provides adaptive PIs instead of the fixed ones from S-CP. Both S-CP and LW-S-CP can provide valid PI for arbitrary $1-\alpha$ by recomputing the appropriate δ. Finally, they can be used as a wrapper for any pretrained model without requiring any major modification. However, it is possible to obtain even tighter PIs. By default, the supervised learned regression models often rely on deep neural networks to estimate the conditional mean. It is well-known [40] that over-parameterized models could overfit to the training data. This could result in very low residuals for the training data that is an under-estimate of errors in test set. The use of normalized residuals $\hat{r}$ ameliorates this issue but also results in the loss of some adaptivity.

In this subsection, we describe conditional quantile regression (CQR) [40] that allows us to obtain tighter bounds under two additional restrictions. First, the miscoverage level α has to be known and fixed (alternatively, we have to train a different CQR model for each α). Second, we must be able to modify the *loss* function of the learned model. Specifically, we will train an *additional* uncertainty quantification model that is identical to learned model except for the loss function where it uses a quantile loss function.

The goal of a traditional regression models is to learn the conditional mean of $Y|X$. In contrast, an τ-quantile regression function seeks to learn the τ-quantile of $Y|X$ which makes it robust to outliers. For example, $\tau = 0.5$ estimates the conditional median of $Y|X$. Suppose we require a coverage of 0.9. We train two quantile regression models with $\tau = 0.05$ and $\tau = 0.95$ dubbed as $\hat{Q}_l$ and $\hat{Q}_u$ respectively. Then for a new X, the true value y is *likely* to be in the interval $[\hat{Q}_l(X), \hat{Q}_u(X)]$. Intuitively, this is due to the fact that the model $\hat{Q}_l$ strives to ensure that $Y|X$ falls below the estimate $\hat{Q}_l(X)$ with 5% probability. Similarly, $\hat{Q}_u$ seeks that $Y|X$ has a value above $\hat{Q}_u(X)$ with 5% probability. Hence, this provides a *nominal* coverage of 0.9 as desired. Of course, the intervals provided by quantile regression does not provide any validity guarantees. Hence, it is necessary to *conformalize* the quantile regression outputs as originally proposed in [40]. Note that quantile regression is naturally adaptive to the data and typically provides an asymmetric interval i.e. $|\hat{f}(X) - \hat{Q}_l(X)| \neq |\hat{f}(X) - \hat{Q}_u(X)|$. In contrast, the interval is symmetric and fixed to δ for both S-CP and LW-S-CP.

The goal of CQR is to convert the heuristic uncertainty estimate provided by quantile regression into a rigorous uncertainty estimate through the conformal inference framework [2], [40]. For a given α, we train two quantile regression models $\hat{Q}_l$ and $\hat{Q}_u$ using the training dataset $\mathcal{D}_T$. Specifically, $\hat{Q}_l$ and $\hat{Q}_u$ are trained with $\tau = \alpha/2$ and $\tau = 1 - \alpha/2$ respectively. We define the scoring function as

$$S(X, Y) = \max\{\hat{Q}_l(X) - y, \hat{Q}_u(X) - y\}$$

and compute it for all (X, y) in the calibration set $\mathcal{D}_C$. We then use this set to compute δ as the $\lceil (n_1 + 1)(1 - \alpha) \rceil$-th smallest value. The prediction interval is now provided by

$$[\hat{Q}_l(X) - \delta, \hat{Q}_u(X) + \delta]$$

Note that we do not directly use $\hat{f}(X)$ and instead used the quantile functions $\hat{Q}_l(X)$ and $\hat{Q}_u(X)$ that is guaranteed to cover $\hat{f}(X)$. To achieve this, we replace the loss function of $\hat{f}(X)$ (such as MSE or average q-error) with the *quantile loss* function. The remainder of the learned model architecture does not require any modification.

Algorithm 4 Conformalized Quantile Regression [40]

Input: $\mathcal{D}$, α
Split $\mathcal{D}$ to two disjoint subsets $\mathcal{D}_T$ and $\mathcal{D}_C$.
Train model $\hat{f}$ using $\mathcal{D}_T$
Train $\hat{Q}_l$ with $\tau = \alpha/2$ and $\hat{Q}_u$ with $\tau = 1 - \alpha/2$ on $\mathcal{D}_T$
$r = \{\max\{\hat{Q}_l(X_i) - y_i, \hat{Q}_u(X_i) - y_i\}\}$ $\forall (X_i, y_i) \in \mathcal{D}_C$
$\delta = \lceil (n_1 + 1)(1 - \alpha) \rceil$-th smallest value in r where $n_1 = |\mathcal{D}_C|$.
return $[\hat{Q}_l(X) - \delta, \hat{Q}_u(X) + \delta]$ for any query feature X

IV. Discussion

Incorporating Workload Information. So far, we have discussed the use of PI techniques in an abstract setting. In a production setting, it is often possible to achieve better results for each of four algorithms described in the paper by leveraging additional workload information. In general, the width of PI is dependent on the $1-\alpha$-th quantile of the residual error. If the queries in the training/calibration dataset are representative of the workload, then the learned model would have smaller residual errors. This in turn results in smaller δ and thereby tighter prediction interval. It is possible to further improve the prediction intervals for the conformal prediction based variants. Let $\mathcal{D}_C = \{(X_1, y_1), (X_2, y_2), \ldots, (X_l, y_l)\}$ be the calibration set. By default, the PI are obtained by computing the residual error over the queries in the conformal set. However, conformal prediction has a natural 'online' property [50], [58] whereby one could augment $\mathcal{D}_C$ at runtime. Consider a scenario where we get a query X_{l+1} and we compute the prediction interval using $\mathcal{D}_C$. After the query is executed, we would have the correct selectivity y_{l+1}. We can now augment the calibration set as $\mathcal{D}_C = \mathcal{D}_C \cup \{(X_{l+1}, y_{l+1})\}$. Since the queries are exchangeable, this does not violate the assumptions of conformal prediction. As the system processes more and more queries, the calibration set becomes more attuned to the latest workload. One could also use a 'window' approach where the calibration set consists of the queries for a given time interval (such as the last 24 hours). Each of these approaches transparently encode the workload information into the calibration set resulting in tighter intervals.

Overhead for Prediction Intervals. Each of the algorithms has two phases – preprocessing phase and an inference phase. The preprocessing phase consists of two key steps: model building and estimating relevant parameters such as δ. Let us consider the model building step. JackKnife+ with cross validation (JK-CV+) requires the training of K learned cardinality models. While split conformal prediction (S-CP) does not require training of any additional model, the locally weighted variant (LW-S-CP) requires the training of a model $\hat{g}$ that can estimate the difficulty of predicting denoted as $U(X) = \hat{g}(X)$. Typically, $\hat{g}$ is simpler and more lightweight than the learned model $\hat{f}$. In our experiments, we used xgboost for instantiating $\hat{g}$. Finally, conformalized quantile regression (CQR) requires the training of two models $\hat{Q}_l, \hat{Q}_u$ for obtaining the upper and lower quantile estimates. These are identical to the learned model $\hat{f}$ except that they use the quantile loss. Once the relevant models have been trained, they can be used to obtain the set of residuals r for the entire training dataset (JK-CV+) or the calibration set (S-CP, LW-S-CP and CQR). Additionally, LW-S-CP also requires the scaling of residuals using $\hat{g}$. Once the (scaled) residuals r are computed, we can compute δ as the $1-\alpha$-th quantile of r. Note that δ is pre-computed and need not be estimated for each query.

During the inference phase, we have to estimate the prediction interval for a given query X. Given the output of learned model $\hat{f}(X)$, the prediction interval for JK-CV+ and S-CP can be computed by just performing a single addition and subtraction as $[\hat{f}(X) - \delta, \hat{f}(X) + \delta]$. For LW-S-CP, we calculate $U(X)$ and compute the prediction interval as $[\hat{f}(X) - \delta U(X), \hat{f}(X) + \delta U(X)]$. Here, the cost is dependent on the ML model $\hat{g}$ that produces $U(X)$. Using a lightweight model such as xgboost for $\hat{g}$ requires less than 0.1 milliseconds in CPU for producing $U(X)$. Finally, CQR requires the invocation of two models $\hat{Q}_l, \hat{Q}_u$. Since $\hat{Q}_l, \hat{Q}_u$ are identical to the learned model $\hat{f}$ except for the loss function, the cost is exactly twice that of obtaining the selectivity $\hat{f}(X)$ from the learned model. The PI is produced as $[\hat{Q}_l(X) - \delta, \hat{Q}_u(X) + \delta]$. One can invoke $\hat{f}(X), \hat{Q}_l(X)$ and $\hat{Q}_u(X)$ in parallel so that the PI is available immediately after $\hat{f}(X)$ is computed.

Obtaining Calibration Set. A practical issue in applying conformal prediction based approaches is the need for calibration set. As we shall demonstrate later, these approaches work best when they satisfy the exchangeabilty property. Every learned selectivity model (and in general, almost all ML models) make the i.i.d. assumption. Furthermore, if the labeled dataset $\mathcal{D}$ has the i.i.d. property than any arbitrary partition of them into training ($\mathcal{D}_T$) and calibration ($\mathcal{D}_C$) data sets still satisfies the i.i.d. property. Hence, a typical setup of the learned model already ensures that the calibration set is representative (i.e. exchangeable and i.i.d.) with the test set. When this assumption is violated (such as due to covariate shift), then the performance of BOTH the ML and the prediction interval algorithm deteriorates. There are some prior theoretical research for testing whether the dataset has the exchangeability property. A common approach is through martingales [9]. Almost all of the prior work on learned selectivity models produce the queries using a generator. These methods often already have the i.i.d. and thereby exchangeability property. We empirically confirmed that workload of common benchmarks such as LeCard [51], TPC-DS and JOB benchmark datasets have the exchangeability property. Another practical heuristic would be to augment the calibration set with queries from the test set so that it becomes eventually representative of the test set.

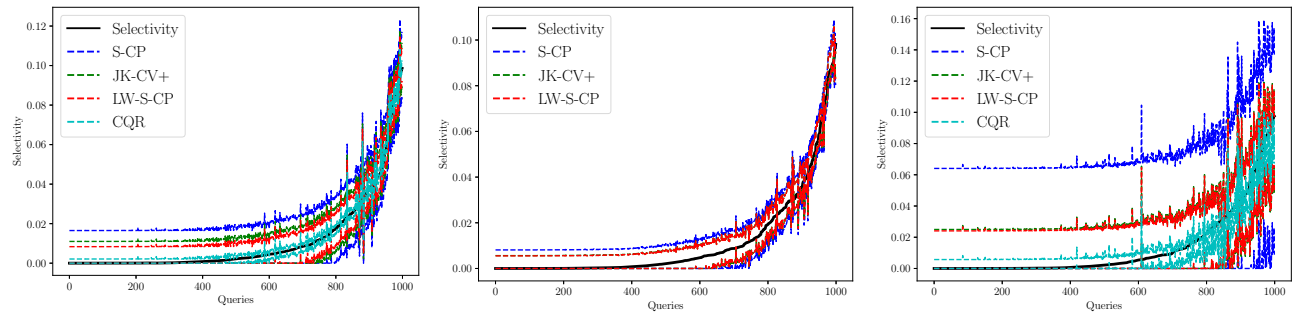
V. Experimental Evaluation

In this section, we conduct extensive experiments to evaluate the performance of the uncertainty quantification algorithms described in Section III.

A. Experimental Setup

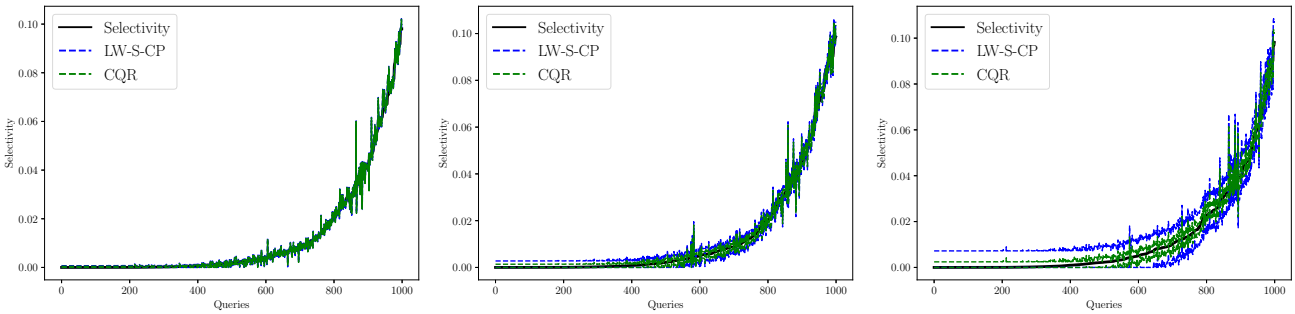
Hardware and Platform. All our experiments were performed on a NVidia V100 GPU. The CPU is a quad-core 2.2 GHz machine with 32 GB of RAM. We implemented the algorithms for uncertainty quantification in Python.

Datasets. All of our experiments are conducted on the DMV dataset that has been widely used for evaluating learned cardinality estimation models. DMV consists of 11.6M tuples and 11 columns of which 10 are categorical. We also conducted experiments on other datasets such as Census, Forest and Power that were recently used in [51] for a rigorous evaluation of learned models.



(a) MSCN (b) Naru (c) LW-NN

Fig. 1: Prediction Intervals for Residual Error as Scoring Function.



(a) Census (b) Forest (c) Power

Fig. 2: Prediction Intervals for three single table datasets with residual error as scoring function (MSCN).

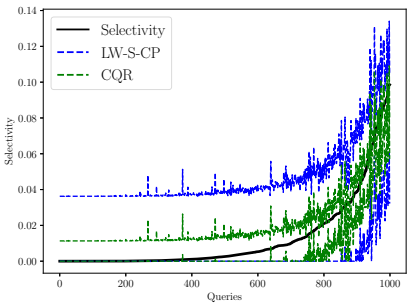


Fig. 3: Join queries in TPC-DS (MSCN)

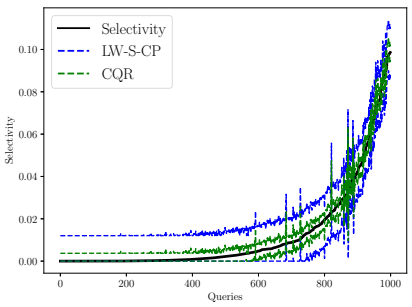


Fig. 4: Join queries in JOB (MSCN)

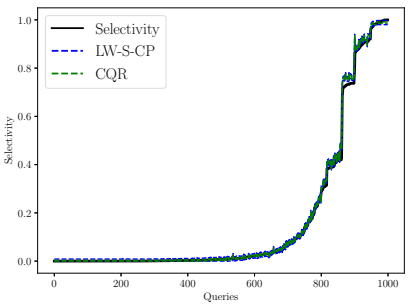


Fig. 5: PI for queries with larger selectivities (MSCN)

Performance Metric. We used q-error defined in Section II for measuring the estimation quality. Recall that q-error of 1 corresponds to perfect estimate. This metric has been extensively used in the design of learned cardinality estimation models. If the estimated or true cardinality is 0, then we modify it to 1.

Cardinality Estimation Algorithms. Our experiments are conducted over three representative approaches – MSCN, Naru and LW-NN. MSCN is an exemplar of supervised query-driven approaches while Naru is a data-driven unsupervised method. LW-NN is a lightweight model targeted towards range predicates. It uses heuristic features such as estimates from simpler models achieving good performance. Each of these methods obtain competitive results in recent and independent experimental evaluations [51], [16], [45]. Please refer to Section II for further details. For each of these models, we reuse the best hyper-parameters found in a recent work [51].

Uncertainty Quantification Algorithms. We use the four approaches described in Section III for computing the prediction intervals. They are Jackknife+ with cross validation (JK-CV+), split conformal prediction (S-CP), locally weighted split conformal prediction (LW-S-CP) and conformal quantile regression (CQR). We used K=10 for Jackknife+ – ie we

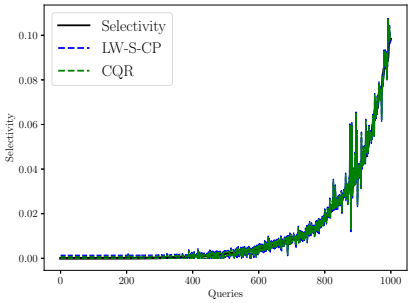


Fig. 6: PI for Q-Error as Scoring Function.

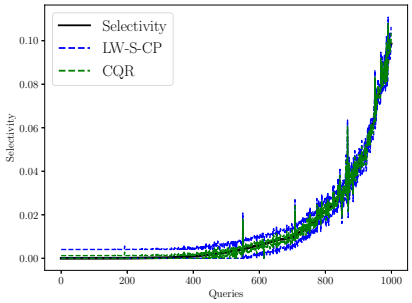


Fig. 7: PI for Relative Error as Scoring Function.

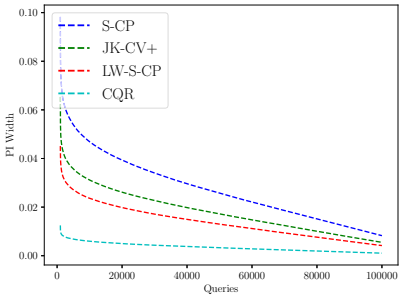


Fig. 8: PI Reduction with increasing calibration set (MSCN)

train 10 models by splitting the dataset into 10 disjoint partitions. For the conformal methods, we used a 50-50 split of training and calibration datasets consisting of 10K queries each. We evaluate the prediction intervals on another workload consisting of 10K queries. We evaluate the impact of the training-calibration split later. Unsupervised approaches such as Naru does not require a training workload. Here, we just use the 10K queries for calibration alone. We modified the supervised approaches MSCN and LW-NN for CQR based prediction intervals. This is achieved by training two quantile regression models $\hat{Q}_l$ and $\hat{Q}_u$ by altering the loss function (mean q-error for MSCN and MSE for LW-NN) with pinball quantile loss function. However, this approach does not work for unsupervised approaches such as Naru. Hence, we compute CQR PI only for supervised models. By default, we use the residual error as the scoring function. The default coverage level is set to 0.9. We do a common sense post-processing of the prediction interval by clipping it to 0 and N respectively which corresponds to the minimum and maximum possible values for cardinality of a query.

Query Workload. We use the unified workload generator from [51] that subsumes all the workload settings of prior work on learned cardinality estimation. The generator is a principled approach for obtaining a wide variety of queries including point and range queries. In our plots, we focus on queries with low selectivity (less than 0.1). Queries with higher selectivity are often answered accurately by almost all learned models. In fact, even a simple sampling based approach would provide acceptable answer in this setting. Illustrating the low cardinality queries allow us to showcase the key trends of prediction intervals without cluttering the plot with high selectivity queries that obfuscate them.

B. Experimental Analysis

Feasibility of Prediction Intervals. In our first set of experiments, we investigate the feasibility of prediction intervals – whether it is possible to compute tight prediction intervals with rigorous theoretical guarantees in an efficient manner without requiring extensive modifications to the model. Our experimental results from Figure 1 for the DMV dataset shows that it is indeed feasible! We compute prediction interval for three learned models – MSCN, Naru and LW-NN – using the residual error as scoring function. We plot the normalized cardinality along with the prediction intervals from the algorithms.

We can make the following observations. First, each of the algorithms for computing PIs comfortably satisfied the coverage requirements. Empirically, the prediction interval contained the true cardinality estimate for more than 90% of the queries in the test set. Second, roughly speaking, the four algorithms demonstrate a consistent ranking based on the tightness of the prediction interval. S-CP is the widest followed by Jackknife+, localized S-CP and CQR. While S-CP and JK-CV+ provide a constant PI, the PI of localized S-CP and CQR could vary based on the query. This explains the comparatively higher level of noise in the PI of localized S-CP and CQR. Finally, the width of PI is dependent on the accuracy of the cardinality estimation algorithm. Naru is one of the most-accurate learned models and hence also benefits from tighter PIs. LW-NN is comparatively less accurate resulting in wider PIs. The results of the feasibility experiments over three other datasets – Census, Forest and Power can be found in Figure 2. We can see that the major trends and relative ranking between the methods are similar to the DMV dataset. In order to reduce the visual clutter, we only show the results of LW-S-CP and CQR in the remaining plots.

We do not plot the PI for high selectivity queries as the learned models are usually accurate and to avoid visual clutter. Empirically, we found that it is easier to discern the relative performance of PI algorithms for queries with smaller selectivities. For example, the width of prediction interval δ for S-CP is the same for all queries. Hence, a prediction interval of say width 500 is more notable when the selectivity of a query is 100 or 1000 than when it is 100K or 1 million. As can be seen from Figure 5, the prediction intervals of all the algorithms in the paper become indistinguishable.

Multi-Table Datasets. The prediction interval algorithms are agnostic to single/multi table setting. Recall from Algorithms 1 to 4 that they operate on the list of residual errors. These errors could have been obtained from a workload containing diverse types of queries such as point/range, single-table/multi-table.

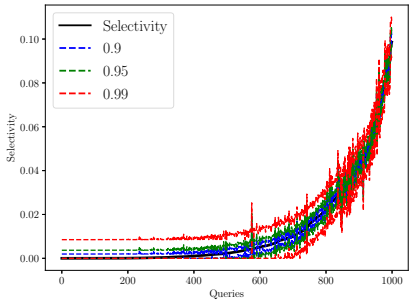


Fig. 9: Varying Coverage Level for CQR and MSCN

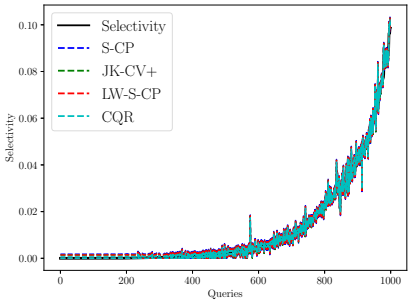


Fig. 10: PI when the calibration and test sets are exchangeable (MSCN)

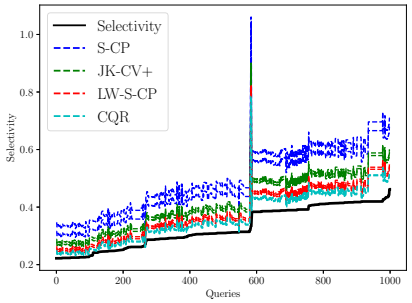


Fig. 11: PI when the calibration and test sets are non-exchangeable (MSCN)

As long as underlying learned model can handle the query type (such as joins), the PI wrapper can transparently provide interval estimates with coverage guarantees. We demonstrate this property by conducting the experiments using the MSCN model for two benchmark datasets – DSB (TPC-DS) and JOB. We conducted experiments on the TPC-DS dataset where the query workload was generated based on the default settings of DSB. Specifically, DSB has 15 templates for SPJ queries and we generated 1000 queries for each template. We also ensured that there were no duplicate queries. We split the query workload into training calibration and testing set in the proportion 50:25:25. We used the JOB dataset and the workload from [22]. We trained the MSCN model and estimated the prediction interval for each query in the testing set. The results can be seen in Figures 3 and 4. We can see that the overall trends and relative rankings of the PI algorithms are consistent with that of the single-table scenario.

Integrating Prediction Intervals in Postgres. We conducted a feasibility experiment based on a setting from [5] where the authors modified the query optimizer of Postgres 9.6.6 to obtain *upper bounds* of join query selectivity from an external module. No other changes were made to the rest of the execution engine. We conducted our experiment in three steps. First, we partitioned the queries from JOB benchmark into equally sized calibration and testing set. There is no need for training data as we used the default Postgres selectivity estimator instead of any learned model. The estimator uses multidimensional histograms, hand-written rules and assumptions on the underlying data [5]. Next, we issued the queries from calibration set and obtained the selectivity estimates of the default Postgres optimizer (with no modifications). We calculated the residual error for each query in the calibration set using their respective correct and estimated selectivites. We estimated the parameter δ as per the split conformal algorithm of Algorithm 2. Finally, each query from the testing set in invoked on a modified version of Postgres from [5]. Specifically, we *replaced* the Postgres' default estimate $Est(Q)$ by the upper bound of the prediction interval (i.e. $Est(Q)+\delta$). Even this simple approach of injecting prediction intervals into the optimizer results in 11% reduction in the cumulative running time of executing all queries in the testing set. We report the average results after repeating the experiment 5 times by randomly partitioning JOB benchmark into calibration and testing set. Our analysis shows that the modified upper bound especially benefits queries involving correlated columns that are often underestimated by Postgres. A rigorous investigation of other algorithms is a promising future work. Table I shows the descriptive statistics of the Q-error for both the approaches. We can see that our proposed approach achieves a meaningful reduction of Q-error and also reduces the running time through the selection of better execution plans.

	Q-Error (Percentile)		
	90	**95**	**99**
Postgres	5.89	143	1822
Postgres with PI	4.62	102	1686

TABLE I: Performance of Postgres with and without Prediction Intervals.

C. Factors Impacting Efficacy of Prediction Intervals

Impact of other scoring functions. By default, we use residual error as the scoring function for obtaining the PI. However, the algorithms described in the paper, are agnostic to the specific scoring function used. Intuitively, the scoring function $S(X, y)$ seeks to incorporate information about the model performance. Hence, it returns smaller values for relatively accurate estimates and larger values otherwise. While relative error function $|y-\hat{f}(X)|$ follows this philosophy, it is possible to do better. For example, consider two queries q_1 and q_2 with cardinalities 100 and 1000 respectively. We get a residual error of 1000 if the learned model predicts $Est(q_1) = 1100$ and $Est(q_2) = 2000$. However, the error for q_1 is much more serious than that of q_2. For example, the q-error for q_1 and q_2 will be 10 and 2 respectively. This motivates us to study the impact of other scoring functions. Specifically, we focus on two of them – q-error and relative error. Relative error is defined as $|Card(q) - Est(q)|/Card(q)$. Both have been widely used for evaluating the quality of cardinality estimators.

Figure 6 shows the PI when q-error is used as the scoring function. We can immediately see that the PIs are much tighter than the one in Figure 1 for residual error. Even though it is not

clearly visible in the plot, the coverage guarantees still hold empirically and theoretically for q-error (and relative error). Figure 7 presents the PI when relative error is used as the scoring function. We can see that the PIs are tighter than that of Figure 1 but wider than that of Figure 6. We can conclude that among the three popular metrics for estimating the cardinality estimation quality, q-error provides the best PIs. Of course, it is possible to obtain better intervals through other custom scoring functions which is orthogonal to our work.

Impact of Coverage Level. In the next set of experiments, we study how varying the coverage level $1-\alpha$ impacts the PIs. We consider three commonly used coverage levels of $0.9, 0.95$ and 0.99. Figure 9 depict the results for the MSCN classifier with CQR. The results for other classifiers and PI algorithms were quite similar. As expected, increasing the coverage level also increases the size of the prediction interval. However, the relative increase in width of PI is dependent on the uncertainty quantification algorithm and the accuracy of the learned model. If the model is relatively accurate, then the increase in PI is relatively small. For example, changing the coverage level from 0.9 to 0.95 results in minimal change for MSCN and Naru but a relatively higher change for LW-NN which is a noisier model. This is due to the fact that the 90-th percentile q-error for MSCN and Naru are very close to that of 95-th percentile q-error. However, when we increase the coverage level from 0.95 to 0.99, we then observe a large PI for MSCN while the relative impact for Naru is minimal. For example, an independent work [51] found that for DMV dataset, the 95-th and 99-th percentile q-error for Naru are 1.09 and 1.35. Whereas, the corresponding values for MSCN and LW-NN are (5.3, 25.0) and (3.29, 22.1) respectively. This is reflected in the proportional increase in PI for these classifiers at the coverage levels 0.99 vis-a-vis 0.95 and 0.9.

Impact of Calibration Set. All of the four algorithms for uncertainty quantification rely on a separate calibration set for obtaining the quantile. In the next set of experiments, we study how varying the calibration set impacts the prediction interval. Specifically, we consider two extreme cases – when the calibration set is representative of the testing dataset and when it is not. The former scenario can occur when the system has a good idea about the testing query workload and used queries similar to the workload in the calibration set resulting in exchangeable sets. In the other extreme, the test queries could be arbitrarily different from that of the calibration set. Figures 10 and 11 illustrate the PI for these two scenarios. When the calibration and test sets are very similar, one obtains tight PI for each of the four algorithms. When they are dissimilar, the exchangeability assumption is violated resulting in inaccurate prediction intervals. In fact, this even results in the loss of coverage guarantees as seen in Figure 11 where the ground truth (black solid line) is not covered by the PIs for significant number of queries. Admittedly, Figure 11 happens only under an extreme situation and a cherry picked set of queries where the true cardinality does not fall into the estimated prediction intervals.

Impact of Training-Calibration Split. Each of the uncertainty quantification algorithm described in the paper relies on the calibration set to choose an appropriate δ based on the desired coverage level. This is especially important for over-parameterized deep neural networks that often have negligible error on the data points in the training dataset due to overfitting. Hence, calibration set provides a better estimate of the performance of the classifier over unseen queries. Almost all of the prior work on conformal inference use a 50-50 split of the dataset $\mathcal{D}$ into training $\mathcal{D}_T$ and calibration $\mathcal{D}_C$ datasets. However, this results in a tradeoff. It might seem that increasing the size of the calibration set could potentially result in tighter prediction intervals. However, the corresponding reduction in the size of the training dataset results in a less accurate classifier. The lack of accuracy of the classifier in turn makes the prediction intervals larger. On the other hand, increasing the size of the training set makes the classifier more accurate. However, using too little calibration set results in reduced calibration accuracy resulting in tighter prediction intervals and a higher variance. Note that as long as the exchangeability assumption holds, the calibration set still provides the coverage guarantees. However, the variance of the prediction intervals (specifically δ) will be higher based on which queries fall in the smaller calibration set.

In order to reduce the clutter in the plots, we consider three discrete settings where the training dataset $\mathcal{D}_T$ is set to 25%, 50% and 75% of the labeled dataset $\mathcal{D}$. The remaining is allocated to calibration set $\mathcal{D}_C$. Figure 12 depict the result of this experiment for MSCN classifier for LW-S-CP algorithm. We can observe that using a smaller training dataset results in a larger prediction interval. Furthermore, increasing the training dataset size to 75% results in the tightest prediction interval among the three splits. We empirically found that using a calibration set between 20-33% of the labeled dataset provides a good balance between classifier accuracy and calibration accuracy. This is also in line with the typical sizes of the split between training and validation datasets. This finding is corroborated by a prior work [25] that noted that unequal splits could potentially achieve better performance than equal splits.

Impact of Classifier Accuracy. In the previous set of experiments, the size of training dataset indirectly impacts the classifier accuracy and thereby the prediction intervals. In the final set of experiments, we study how the accuracy of the classifier impacts each of the uncertainty quantification algorithms. We conduct an experiment where we fix all the hyper-parameters that provided the best results for MSCN and Naru except for the number of epochs. Let E_M be the number of epochs that provided best results for MSCN. Then we train three variants of MSCN where the model is trained for $0.5E_M$, $0.75E_M$ and E_M epochs respectively. Then we apply the prediction interval algorithms over these classifier variants. The training and calibration set are fixed for each of these settings. We repeat this process for Naru classifier. Figures 13 and 14 illustrate the behavior for MSCN and Naru classifiers for the split conformal prediction algorithm. As expected,

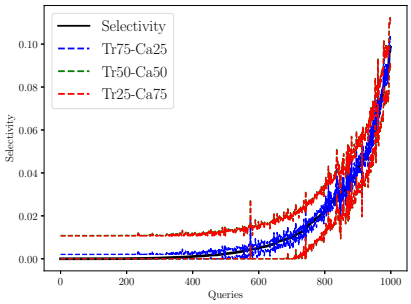


Fig. 12: Varying Training-Calibration Split (MSCN and LW-S-CP)

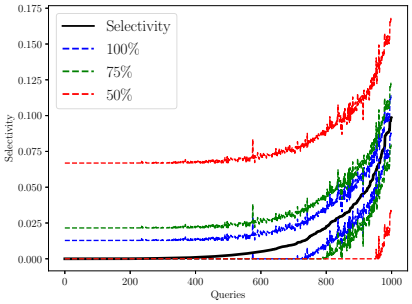


Fig. 13: Impact of Classifier Accuracy (MSCN, S-CP)

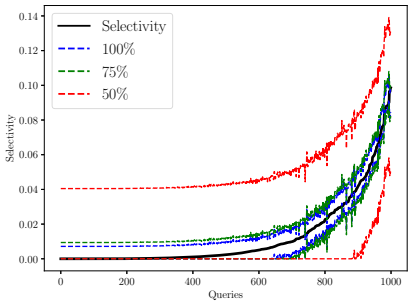


Fig. 14: Impact of Classifier Accuracy (Naru, S-CP)

we can see that S-CP provides valid coverage guarantees regardless of the accuracy of the classifier. However, the tightness of the bounds is influenced by the accuracy of the classifier. Hence, the MSCN/Naru variant that was trained for the entirety has a tighter PI than the variants that were trained for lesser number of epochs. This phenomenon holds across classifiers too with the PI for a Naru variant being tighter than that of the corresponding MSCN variant.

Online Conformal Prediction. Recall from Section IV that one could improve upon the PI by augmenting calibration set. We conduct an experiment over the DMV dataset and MSCN classifier. We begin with a calibration set of size 1000 and 100K queries in the testing dataset. After the PI for each query q_j is obtained, we augment the calibration set using $(q_j, Sel(q_j))$. The PI estimation for every future query q_k will include q_j in the calibration set. Figure 8 shows that the prediction intervals become progressively tighter as the calibration set becomes reflective of the workload.

D. Guidance for Practitioners.

S-CP is a simple and efficient algorithm that can be used to compute the PI. While it provides the widest PI among the four algorithms, it also requires almost no additional steps other than computing the quantile of the residual errors. In contrast, JK-CV+ requires the training of K models, localized S-CP requires the training of another model for estimating the uncertainty $U(X)$ and CQR requires altering the loss function. JK-CV+ provides a tighter PI than S-CP but at the cost of training and maintaining K models. We found that the PI of JK-CV+ were on average 83%-96% that of S-CP. If the accuracy of PI is paramount then JK-CV+ is a viable approach though it imposes a large computational cost. However, it is possible to do much better through LW-S-CP and CQR. LW-S-CP uses a normalization function $U(X)$ (such as the residual error) as the normalization function to provide tighter PI than either S-CP or JK-CV+. However, the best PI is obtained through CQR that requires the most intrusive changes in terms of training two separate quantile regression models by modifying the loss function. This may not always be possible. To summarize, LW-S-CP is an appropriate first choice as it balances efficacy of PI and the inference time. However, if the efficacy is paramount and the learned model allows modifying the loss function, then CQR is appropriate.

Promising approaches for improving PI. There are a number of intriguing open research problems. While we evaluated a simple heuristic of augmenting calibration set, it is possible to design better mechanisms to incorporate workload information. An emerging area of research is that of localized conformal prediction (LCP) [10], [15] where a subset of 'local' queries from the calibration set are used for estimating the PI instead of the entire set. It is possible that using such 'local' queries could provide a tighter interval. If a query is representative of a given workload, it will find sufficient 'local' queries from the workload allowing for tighter prediction interval. A related problem is to detect potential shift in workload so that we could take preventive steps before losing the coverage guarantees. While there are some work on exchangeability tests such as [9], identifying an optimal one that is effective and efficient is important. Finally, there is a need for more research on the design of appropriate scoring and normalization function for S-CP and LW-S-CP respectively. While we used residual, Q- and relative error with promising results, it might be possible to do better.

VI. Conclusion

While learned models for cardinality estimation have achieved tremendous successes in the recent years, accurate cardinality estimation remains a challenging task. In this paper, we investigated an orthogonal problem – how can we quantify the uncertainty associated with the cardinality estimate of a learned model through prediction intervals. We enumerated a series of desiderata and identify four promising approaches for this problem. We provided a self-contained introduction to these ideas and conducted extensive experiments to understand their tradeoffs. The experiments show that it is possible to obtain accurate prediction intervals in an efficient manner without intrusive changes to the learned model.

VII. Acknowledgments

The research of Nick Koudas is supported by NSERC COHESA. The research of Gautam Das is supported in part by grants 1261007500 and 1261007210 from NSF.

REFERENCES

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 2021.

[2] A. N. Angelopoulos and S. Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[3] V. Balasubramanian, S.-S. Ho, and V. Vovk. *Conformal prediction for reliable machine learning: theory, adaptations and applications*. Newnes, 2014.

[4] R. F. Barber, E. J. Candes, A. Ramdas, and R. J. Tibshirani. Predictive inference with the jackknife+. *The Annals of Statistics*, 49(1):486–507, 2021.

[5] W. Cai, M. Balazinska, and D. Suciu. Pessimistic cardinality estimation: Tighter upper bounds for intermediate join cardinalities. In *Proceedings of the 2019 International Conference on Management of Data*, pages 18–35, 2019.

[6] S. Chaudhuri, B. Ding, and S. Kandula. Approximate query processing: No silver bullet. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 511–519, 2017.

[7] B. Ding, S. Chaudhuri, J. Gehrke, and V. Narasayya. Dsb: a decision support benchmark for workload-driven and traditional database systems. *Proceedings of the VLDB Endowment*, 14(13):3376–3388, 2021.

[8] 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*, 12(9):1044–1057, 2019.

[9] V. Fedorova, A. J. Gammerman, I. Nouretdinov, and V. Vovk. Plug-in martingales for testing exchangeability on-line. In *ICML*, 2012.

[10] R. Foygel Barber, E. J. Candes, A. Ramdas, and R. J. Tibshirani. The limits of distribution-free conditional predictive inference. *Information and Inference: A Journal of the IMA*, 10(2):455–482, 2021.

[11] M. N. Garofalakis and P. B. Gibbons. Approximate query processing: Taming the terabytes. In *VLDB*, volume 10, pages 645927–672356, 2001.

[12] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

[13] M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

[14] 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*, pages 461–472, 2001.

[15] L. Guan. Localized conformal prediction: A generalized inference framework for conformal prediction. *arXiv preprint arXiv:2106.08460*, 2021.

[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 *SIGMOD*, pages 1035–1050, 2020.

[18] T. Heskes. Practical confidence and prediction intervals. *NIPS*, pages 128–135, 1997.

[19] 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.

[20] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NIPS*, 2017.

[21] A. Kipf, M. Freitag, D. Vorona, P. Boncz, T. Neumann, and A. Kemper. Estimating filtered group-by queries is hard: Deep learning to the rescue. In *1st International Workshop on Applied AI for Database Systems and Applications*, 2019.

[22] 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.

[23] J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018.

[24] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? *Proceedings of the VLDB Endowment*, 9(3):204–215, 2015.

[25] H. Linusson, U. Johansson, H. Boström, and T. Löfström. Efficiency comparison of unstable transductive and inductive conformal classifiers. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 261–270. Springer, 2014.

[26] R. J. Lipton, J. F. Naughton, and D. A. Schneider. Practical selectivity estimation through adaptive sampling. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 1–11, 1990.

[27] J. Liu, W. Dong, Q. Zhou, and D. Li. Fauce: Fast and accurate deep ensembles with uncertainty for cardinality estimation. *PVLDB*, 2021.

[28] Q. Ma, A. M. Shanghooshabad, M. Almasi, M. Kurmanji, and P. Triantafillou. Learned approximate query processing: Make it light, accurate and fast. In *CIDR*, 2021.

[29] Q. Ma and P. Triantafillou. Dbest: Revisiting approximate query processing engines with machine learning models. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1553–1570, 2019.

[30] B. Mozafari. Approximate query engines: Commercial challenges and research opportunities. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 521–524, 2017.

[31] B. Mozafari and N. Niu. A handbook for building an approximate query engine. *IEEE Data Eng. Bull.*, 38(3):3–29, 2015.

[32] R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

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

[34] J. Ortiz, M. Balazinska, J. Gehrke, and S. S. Keerthi. An empirical analysis of deep learning for cardinality estimation. *arXiv preprint arXiv:1905.06425*, 2019.

[35] H. Papadopoulos, K. Proedrou, V. Vovk, and A. Gammerman. Inductive confidence machines for regression. In *European Conference on Machine Learning*, pages 345–356. Springer, 2002.

[36] H. Papadopoulos, V. Vovk, and A. Gammerman. Regression conformal prediction with nearest neighbours. *Journal of Artificial Intelligence Research*, 40:815–840, 2011.

[37] Y. Park, B. Mozafari, J. Sorenson, and J. Wang. Verdictdb: Universalizing approximate query processing. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1461–1476, 2018.

[38] 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*, pages 1017–1033, 2020.

[39] T. Pearce, A. Brintrup, M. Zaki, and A. Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *International Conference on Machine Learning*, pages 4075–4084. PMLR, 2018.

[40] Y. Romano, E. Patterson, and E. Candes. Conformalized quantile regression. *Advances in Neural Information Processing Systems*, 32:3543–3553, 2019.

[41] F. Savva, C. Anagnostopoulos, and P. Triantafillou. Ml-aqp: Query-driven approximate query processing based on machine learning. *arXiv preprint arXiv:2003.06613*, 2020.

[42] S. Shetiya, S. Thirumuruganathan, N. Koudas, and G. Das. Astrid: accurate selectivity estimation for string predicates using deep learning. *Proceedings of the VLDB Endowment*, 14(4):471–484, 2020.

[43] L. Steinberger and H. Leeb. Conditional predictive inference for high-dimensional stable algorithms. *arXiv preprint arXiv:1809.01412*, 2018.

[44] H. Su, M. Zait, V. Barrière, J. Torres, and A. Menck. Approximate aggregates in oracle 12c. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1603–1612, 2016.

[45] J. Sun, J. Zhang, Z. Sun, G. Li, and N. Tang. Learned cardinality estimation: A design space exploration and a comparative evaluation. 2022.

[46] S. Thirumuruganathan, S. Hasan, N. Koudas, and G. Das. Approximate query processing for data exploration using deep generative models. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1309–1320. IEEE, 2020.

[47] R. Tibshirani. A comparison of some error estimates for neural network models. *Neural Computation*, 8(1):152–163, 1996.

[48] K. Tzoumas, A. Deshpande, and C. S. Jensen. Lightweight graphical models for selectivity estimation without independence assumptions. *Proceedings of the VLDB Endowment*, 4(11):852–863, 2011.

[49] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[50] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

[51] X. Wang, C. Qu, W. Wu, J. Wang, and Q. Zhou. Are we ready for learned cardinality estimation? *PVLDB*, 2021.

[52] Z. Wang, D. Cashman, M. Li, J. Li, M. Berger, J. A. Levine, R. Chang, and C. Scheidegger. Neuralcubes: Deep representations for visual data exploration. *arXiv preprint arXiv:1808.08983*, 2018.

[53] L. Woltmann, C. Hartmann, M. Thiele, D. Habich, and W. Lehner. Cardinality estimation with local deep learning models. In *Proceedings of the second international workshop on exploiting artificial intelligence techniques for data management*, pages 1–8, 2019.

[54] 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*, pages 2009–2022, 2021.

[55] Z. Wu and A. Shaikhha. Bayescard: A unified bayesian framework for cardinality estimation. *arXiv e-prints*, pages arXiv–2012, 2020.

[56] Z. Yang, A. Kamsetty, S. Luan, E. Liang, Y. Duan, X. Chen, and I. Stoica. Neurocard: one cardinality estimator for all tables. *arXiv preprint arXiv:2006.08109*, 2020.

[57] 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. *Proceedings of the VLDB Endowment*, 13(3):279–292, 2019.

[58] G. Zeni, M. Fontana, and S. Vantini. Conformal prediction: a unified review of theory and new challenges. *arXiv preprint arXiv:2005.07972* 2020.

[59] M. Zhang and H. Wang. Approximate query processing for group-by queries based on conditional generative models. *arXiv preprint arXiv:2101.02914*, 2021.

[60] M. Zhang and H. Wang. Laqp: Learning-based approximate query processing. *Information Sciences*, 546:1113–1134, 2021.

[61] K. Zhao, J. X. Yu, Z. He, and H. Zhang. Uncertainty-aware cardinality estimation by neural network gaussian process. *SIGMOD*, 2022.

[62] 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.