Uncertainty-Aware Benefit Estimation: A Reliable Framework for Index Tuning

Tao Yu
Harbin Institute of Technology
Harbin, Heilongjiang, China
21B903056@stu.hit.edu.cn

Zhaonian Zou
Harbin Institute of Technology
Harbin, Heilongjiang, China
znzou@hit.edu.cn

Hao Xiong
Harbin Institute of Technology
Harbin, Heilongjiang, China
xionghao@stu.hit.edu.cn

ABSTRACT

Index tuning is a critical task for optimizing database performance, involving the selection of optimal indexes based on workload and constraints. Traditional methods for benefit estimation in index tuning rely on what-if tools, which suffer from low efficiency and accuracy. Learning-based models offer a promising alternative but are hindered by issues such as poor stability, lack of interpretability, and complex management. Research on quantifying uncertainty to validate the reliability of model results has received widespread attention in safety-critical fields such as medical research and autonomous driving. Therefore, this paper investigates the question: Can we utilize uncertainty to achieve high-quality and reliable index tuning based on learning-based BE models?

We propose Beauty, an uncertainty-aware framework that integrates learning-based models with what-if tools. Our approach leverages uncertainty quantification techniques to improve model reliability and reduce management complexity. Specifically, we introduce a novel method combining AutoEncoder and Monte Carlo Dropout to quantify uncertainty, tailored to the characteristics of benefit estimation tasks.

In our experiment, we compared 16 different models, demonstrating that our approach outperforms existing uncertainty quantification methods in almost all cases. To address the research question, we conducted tests on six datasets. By employing the Beauty framework, we were able to reduce the number of worst cases to zero and increase the number of best cases more than threefold.

PVLDB Reference Format:
Tao Yu, Zhaonian Zou, and Hao Xiong. Uncertainty-Aware Benefit Estimation: A Reliable Framework for Index Tuning. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/HIT-DB-Group/Beauty.

1 INTRODUCTION

Indexes are a critical means of accelerating database queries. The challenge of automatically creating suitable indexes for databases has long been a significant research focus, known as index tuning [10, 12, 39]. The component that addresses this problem is referred to as the index advisor. This problem assumes a workload composed of a set of queries and budget constraints, such as maximum storage space. Existing solutions typically employ a component called the index configuration enumerator to continuously propose potential index configurations based on pre-designed heuristic methods or reinforcement learning algorithms. Concurrently, the query optimizer uses what-if tools to estimate the benefits these index configurations would bring to the workload if they were created. Upon reaching the stopping condition, the index advisor provides the optimal index configuration identified during the exploration.

Benefit estimation (**BE** for short) is crucial for effective index tuning. Using what-if tools has two major drawbacks: (1) *Low inference efficiency*: The optimizer needs to sequentially compare the costs of almost all plans for a query. This process accounts for nearly 90 percent of the entire index tuning time [42]. (2) *Low accuracy*: The optimizer's cardinality and cost estimators usually yield significant errors especially when evaluating complex queries. Inaccurate BE can even lead to a decrease in system performance after tuning.

To address these issues, researchers have introduced learning-based models, such as in [14, 37, 41]. However, these methods have remained confined to academic research and have not been applied to real-world systems. Despite the high accuracy that learning-based models can exhibit in certain situations, there are concerns about the following drawbacks: (1) *Poor stability*: While learning-based methods may outperform what-if tools for queries and index configurations similar to their training datasets, they often produce highly erroneous results due to model design flaws and insufficient training data. (2) *Lack of interpretability*: As black-box models, learning-based methods make it difficult to understand when and why they perform poorly in real-world applications. (3) *Complex management*: These models require extensive data collection before deployment and continuous monitoring to determine if updates are necessary, introducing additional overhead. These factors are the main barriers to the adoption of learning-based methods.

This paper focuses on bridging the gap between the research and practical application of learning-based models. We note that the three aforementioned issues are present in many scenarios involving neural networks. In fields such as medical image detection [29, 32, 34], autonomous driving [2, 9, 17], and drug analysis [28, 35, 45], the tolerance for model errors is even lower than in BE, as incorrect results can lead to severe consequences. To quantify a model's confidence in its results and detect potential errors, uncertainty is categorized by its source into model uncertainty and data uncertainty . Methods such as Bayesian inference and ensemble approaches have been proposed to perform this quantification process, known as uncertainty quantification (**UQ** for

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

short). This naturally leads us to the question: Can we utilize uncertainty to achieve high-quality and reliable index tuning based on learning-based BE models?

To address this fundamental question, we conducted a systematic study on the sources of uncertainty in learning-based BE models. One natural approach to improving these models is to tackle the causes of uncertainty by designing more robust models that reduce output variability. However, considering factors such as deployment cost and compatibility, we proposed an alternative framework that leverages uncertainty. This framework, called Beauty (**b**enefit **e**stim**a**tion with **u**ncertain**ty**), is the first uncertainty-aware BE framework. It utilizes UQ techniques to combine the advantages of learning-based BE models with what-if tools, ensuring compatibility with existing systems. Additionally, it reduces model management costs and guides model design.We believe this framework can enable the practical application of learning-based models.

Although many existing methods for UQ are designed for general tasks, their accuracy and efficiency are suboptimal when applied directly to BE tasks. Therefore, we further categorized the uncertainty in BE and proposed a method that combines the use of AutoEncoder and Monte Carlo Dropout for quantification. This method exploits the characteristics of BE tasks and is compatible with the structure of existing learning-based estimators, outperforming general methods in almost all scenarios. Additionally, we provide a scheme to achieve a trade-off between quantification accuracy and inference efficiency based on model characteristics.

In summary, this paper makes the following contributions:

(1) We conducted a systematic study and classification of the sources of uncertainty in learning-based BE models within the context of index tuning tasks (Section 3). Based on the characteristics of these tasks, we proposed an accuracy UQ method that allows for a flexible trade-off between efficiency and accuracy, depending on model characteristics (Section 4). Our experiments demonstrate that the proposed UQ method outperforms all general-purpose UQ methods in five out of six scenarios.

(2) We introduced Beauty, the first uncertainty-aware BE framework, which combines the advantages of learning-based models and what-if tools and realizes uncertainty-driven model updating. We conducted experiments across six benchmarks with varying budgets. The experimental results demonstrate that the Beauty framework significantly improves index tuning outcomes (Section 6).

(3) In our experiments, we applied existing UQ methods to various learning-based BE models, resulting in a total of 16 different models (Section 5). We thoroughly studied the impact of these methods on model accuracy and efficiency, confirming the superior performance of our proposed UQ method.

2 BACKGROUND

2.1 The Index Tuning Problem

Let D be a relational database. A *query workload* W (workload for short) on D comprises a set of queries $\{q_1, q_2, \ldots, q_n\}$ on D, where each query $q_i \in W$ is weighted by $w_i \in \mathbb{R}$ according to the frequency and the significance of q_i [6, 12, 25, 38]. For a query q on D and a set $\mathcal{I}$ of indexes created on D, the cost of evaluating q with the support of $\mathcal{I}$ is denoted by $c(q, \mathcal{I})$. Let $\mathcal{I}_0$ be the current set of indexes created on D. The *benefit* of replacing $\mathcal{I}_0$ with another set $\mathcal{I}$ of indexes with respect to q is $B(q, \mathcal{I}_0, \mathcal{I}) = c(q, \mathcal{I}_0) - c(q, \mathcal{I})$. The

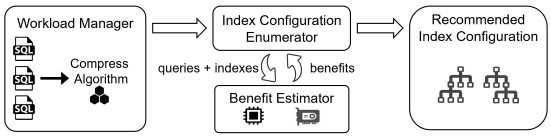


Figure 1: The typical structure of an index advisor.

index tuning problem, also known as the *index selection* problem, is often formalized as follows. Given a database D, the current set $\mathcal{I}_0$ of indexes on D, a recent workload W, and the maximum number k of indexes that can be created, find an *index configuration* $\mathcal{I}$ (i.e., a set of candidate indexes) that has $|\mathcal{I}| \leq k$[1] and can maximize the *total benefit* if $\mathcal{I}_0$ is completely replaced by $\mathcal{I}$:

$$B(W, \mathcal{I}_0, \mathcal{I}) = \sum_{i=1}^{n} w_i \cdot B(q_i, \mathcal{I}_0, \mathcal{I}). \tag{1}$$

Obviously, maximizing $B(W, \mathcal{I}_0, \mathcal{I})$ is equivalent to minimizing the *total cost* $c(W, \mathcal{I}) = \sum_{i=1}^{n} w_i \cdot c(q_i, \mathcal{I})$.

2.2 Index Advisors

Index advisor is a component in a DBMS specially designed for index tuning. An index advisor is typically composed of the following modules:

(1) *Workload Manager.* The workload manager collects queries executed recently under the current index configuration $\mathcal{I}_0$. For each collected query, the SQL statement and the execution plan are acquired. In order to reduce space overhead and subsequently improve the efficiency of index tuning, the workload manager can also compress the collected workload by eliminating redundant or less informative queries [12, 38].

(2) *Index Configuration Enumerator.* The index configuration enumerator continually generates hypothetical index configurations $\mathcal{I}$ that meet the specified budget constraint, i.e., $|\mathcal{I}| \leq k$.

(3) *Benefit Estimator.* For each enumerated index configuration $\mathcal{I}$, the benefit estimator computes the estimated benefit of sequentially executing all queries in the workload W given that the indexes in $\mathcal{I}_0$ are all replaced by the indexes in $\mathcal{I}$.

The typical working process of an index advisor is depicted in Figure 1. The benefit estimator plays an essential role in index tuning because the efficiency of BE determines the efficiency of index tuning, and the accuracy of BE also has a direct impact on the behavior of the index configuration enumerator [4, 8, 11, 26, 33]. Therefore, it is crucial to estimate the total benefit $B(W, \mathcal{I}_0, \mathcal{I})$ of replacing $\mathcal{I}_0$ with $\mathcal{I}$ as efficiently and accurately as possible without actually building the indexes in $\mathcal{I}$. The traditional approach is to use *what-if tools* to create virtual indexes and call the query optimizer to estimate the total cost $c(W, \mathcal{I})$. However, as verified in [14, 37], this approach is not always efficient and accurate especially when queries are complex. In recent years, machine learning is used to induce fast and accurate BE models based on historical query execution information under different index configurations.

2.3 Uncertainty in Machine Learning

Uncertainty is the quality or state of being uncertain, which arises in every stage of a machine learning task, including data acquisition, feature encoding, model design, model training, and model

[1] An alternative budget constraint is specified on the maximum size of all indexes in $\mathcal{I}$.

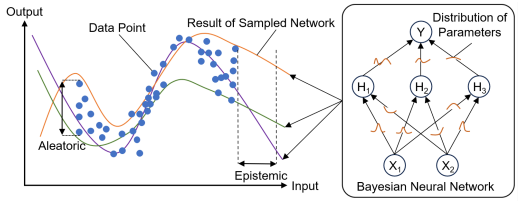


Figure 2: A set of data points and a Bayesian neural network (BNN) trained based on them. Three curves on the left-hand side represent the fitting curves corresponding to three sampled NNs of the BNN, which illustrate the difference between aleatoric uncertainty and epistemic uncertainty.

inference [19]. Naturally, uncertainty is inherent in learning-based index tuning, which will be explained in the next section. As illustrated in Figure 2, there are two different types of uncertainty, often referred to as *aleatoric uncertainty* and *epistemic uncertainty* [24].

Aleatoric uncertainty, also known as *data uncertainty* or *statistical uncertainty*, refers to the notion of *randomness*, that is, the variability in the outcome of an experiment (i.e., random errors) caused by inherently random effects. For instance, the data collected by sensors usually has aleatoric uncertainty because the same input target might correspond to different sensing outputs at random. In principle, aleatoric uncertainty cannot be reduced by incorporating additional information such as collecting more data or using better models. Aleatoric uncertainty can be further divided into *homoscedastic* and *heteroscedastic* depending on whether the inherent randomness varies with the input target.

Epistemic uncertainty, also known as *model uncertainty* or *systematic uncertainty*, refers to uncertainty caused by the *ignorance* of knowledge (about the best model). Epistemic uncertainty may result in errors generated in the design, training and inference of a model. For example, if the inference of a model is made on a dataset that has a distribution significantly different from the model's training dataset, the model might provide inaccurate outputs due to ignorance. As opposed to uncertainty due to randomness, uncertainty caused by ignorance can in principle be reduced based on additional information. In other words, epistemic uncertainty refers to the reducible part of the (total) uncertainty, whereas aleatoric uncertainty refers to the irreducible part.

3 MOTIVATION

3.1 Limitations of Learning-based Benefit Estimation

As discussed in subsection 2.2, improving the efficiency and the accuracy of BE is crucial for index tuning. Learning-based BE is currently the state-of-the-art approach. However, the learning-based BE methods are faced with a number of difficulties which limit the accuracy of BE. Here, we explain these limitations from the perspective of uncertainty in BE models.

U1. Epistemic uncertainty due to insufficient information modeling. Estimating the index benefit is complex, requiring the query optimizer to compare execution plan costs before and after index creation. The optimizer must evaluate nearly all possible plans and choose the one with the lowest cost. In this process, many factors influence the cost, and BE will model only the directly related information, such as query plans and index configurations. However, other factors also affect the cost of query plans. One such factor is data distribution, which determines the intermediate results produced by queries, known as cardinality estimation. The existing learning-based estimator either does not model data distribution or models it in a simplified manner. Cardinality estimation itself is a challenging task, which is a key focus in the AI4DB field. More accurate data-driven methods often require building complex and heavy models, which are far more complex than those used in existing BE models. Another factor is the database's system environment. Database knobs such as "work_mem", "shared_buffers", etc., all influence the conditions during query execution, yet the current BE model does not address this aspect.

U2. Epistemic uncertainty caused by flawed feature extraction. The model requires encoding various types of information, such as queries, their plans, indices, and database statuses, into vectors. This encoding process can lead to information loss. Moreover, flawed feature extraction techniques may encode diverse training data into identical inputs, resulting in a training dataset where the same input yields different outputs, thus contributing to epistemic uncertainty. For example, [37] highlights how Index Optimizable Operations are extracted in query plans. If two similar queries under the same configuration share the same Index Optimizable Operations—usually the leaf nodes in the query plan tree—they become indistinguishable after extraction.

U3. Epistemic uncertainty due to model design and training. The structure and mechanisms of the model, along with settings for hyperparameters and training processes such as batch size, optimizer, learning rate, and stopping criteria, can limit the model's expressive power. As a result, even on the training data, the model generally cannot achieve complete accuracy.

U4. Epistemic uncertainty caused by inadequate data collection. The training data for the BE model requires recording the execution costs and plans of the same query under different index configurations, which places a burden on the system. Additionally, the data might not cover all future scenarios, leading to Out-of-Domain (OOD) issues during inference. For instance, if the model is initially trained on OLAP queries, but receives OLTP queries during inference, its ability to generalize effectively to these new queries may be compromised.

U5. Aleatoric uncertainty due to the randomness of query execution time. Although we can precisely record the time from when a query is issued to when the results are obtained, factors such as cache, system resource scheduling, and hardware conditions cause the execution time of the same query to vary with each execution. This variability is related to the characteristics of the query itself. Therefore, this scenario represents a heteroscedastic aleatoric uncertainty.

3.2 Research Objective and Issues

The six different sources of uncertainty listed in the previous subsection contribute to the uncertainty in a learning-based BE model. To realize more reliable BE, a natural approach is to reduce uncertainty inherent in the learning-based BE model. Among the uncertainty listed above, the aleatoric uncertainty is in principle irreducible, but the epistemic uncertainty can be reduced by acquiring more training data, enriching the information used in modeling, improving the model's feature encoding, or adopting a more powerful model

such as a large language model (LLM). These methods may result in higher training costs, higher deployment costs and longer inference latency. However, as a key module in the index advisor, the benefit estimator has a stringent requirement on its efficiency.

In this paper, we explore a new way to improve the reliability of BE by identifying the situations in which the learning-based BE model is highly inaccurate, that is, the model has higher uncertainty. As opposed to the learning-based benefit estimators, the what-if tools are more reliable. Therefore, we aim to take advantages of the strengths of both the learning-based benefit estimator and the what-if tools. In particular, *when the learning-based BE model is highly uncertain about the estimated benefit output by itself, the what-if tool is used instead to compute the estimated benefit.*

Therefore, we have the following problems to be investigated in the rest of the paper:

Q1. How to quantify the uncertainty inherent in a learning-based BE model? In section 4, we will review the existing general-purpose UQ methods and propose a new quantification method that is more suitable for BE by taking advantages of the characteristics of the BE problem.

Q2. What impact does UQ have on a BE model? In section 5, we will carry out an empirical study on the impact of UQ on a BE model by evaluating the accuracy, the training cost and the inference cost of the BE model extended with UQ as well as the precision of the quantified uncertainty. In subsection 4.3, we will also explore a new way to determine when a BE model needs to be updated according to the quantified uncertainty and design a complete uncertainty-aware benefit estimator.

Q3. What impact does Beauty have on the index advisor? In section 6, we will evaluate the performance of eight models with UQ modules across 36 scenarios within six workloads. This evaluation aims to explore the impact of the accuracy and efficiency of UQ models on the index advisor, as well as the relationship between UQ models and the index configuration enumeration algorithms.

4 QUANTIFYING UNCERTAINTY IN BENEFIT ESTIMATION MODELS

In this section, we review the general-purpose UQ methods and propose our UQ method tailored for BE models which is more accurate and efficient than the existing general-purpose UQ methods.

4.1 General-Purpose UQ Methods

A multitude of *general-purpose* methods have been proposed for quantifying uncertainty in deep learning models. These methods can be categorized into two different types: *model-independent* and *model-dependent*.

Model-Independent UQ Methods. This category of methods include *Bayesian neural networks (BNNs)* [13, 20], *Monte Carlo dropout (MCD) methods* [18] and *ensemble methods* [5, 21, 22] which are independent of the models in which the uncertainty is quantified.

Bayesian Neural Networks (BNNs). BNNs have been extensively studied for UQ. Let $\mathcal{N}$ be a neural network (NN). To quantify the uncertainty in $\mathcal{N}$, a BNN $\mathcal{B}$ is induced, which has the same structure as $\mathcal{N}$ but regards each parameter w in $\mathcal{N}$ as a random variable X_w following a certain probability distribution. Let W be the set of all parameters in $\mathcal{N}$ and $\mathbf{X}_W = \{X_w \mid w \in W\}$ be the set of all corresponding random variables in $\mathcal{B}$. Given a training set $D = \{(\mathbf{x}_i, y_i) \mid 1 \le i \le n\}$, the goal of inducing $\mathcal{B}$ is to learn the posterior probability density function $p(\mathbf{X}_W|D)$ which is given by

$$p(\mathbf{X}_W|D) = \frac{p(D|\mathbf{X}_W)p(\mathbf{X}_W)}{p(D)} = \frac{p(D|\mathbf{X}_W)p(\mathbf{X}_W)}{\int \cdots \int p(D|\mathbf{X}_W)p(\mathbf{X}_W)d\mathbf{X}_W}.$$

The prior distribution $p(\mathbf{X}_W)$ should be defined to reflect the preconceptions about the distribution of the weights W before observing any data. Solving this equation directly can be challenging, so *variational inference (VI)* is often employed in the literature [20, 23, 30]. Blundell et al. [3] proposes the renowned *Bayes by Backprop (BBB)* algorithm which is compatible with backpropagation for learning a probability distribution. The BBB algorithm is selected as the baseline in the subsequent experimental evaluation.

The trained BNN $\mathcal{B}$ and its inference process are shown in Figure 2. The uncertainty in the NN $\mathcal{N}$ with regards to a test record $\mathbf{x}$ can be assessed as follows. First, a collection of settings $\{W_1, W_2, \ldots, W_m\}$ for W can be obtained using Monte Carlo sampling independently and identically according to the posterior probability distribution $p(\mathbf{X}_W|D)$. By setting the parameters W to each sampled configuration W_i, a specific instance $\mathcal{N}_i$ of $\mathcal{N}$ is obtained. Given the test record $\mathbf{x}$ as input, let y_i be the output of $\mathcal{N}_i$. The mean of $\hat{y_1}, \hat{y_2}, \ldots, \hat{y_m}$ is the prediction for $\mathbf{x}$, and the variance of $\hat{y_1}, \hat{y_2}, \ldots, \hat{y_m}$ is the uncertainty of $\mathcal{N}$ with regards to $\mathbf{x}$. In the implementation, the sampling process and the inference process described above can be merged into one process and performed layer by layer on $\mathcal{N}$ in an interleaved manner.

Monte Carlo Dropout (MCD) Methods. Dropout layers are often used in deep NNs to prevent overfitting. During training, a dropout layer randomly sets some elements of the input tensor to 0 with probability $p \in [0, 1]$ and scales up all elements by $1/(1-p)$ so that the sum over all input elements is unchanged. Moreover, the zeroed elements are chosen independently for each forward call. During inference, dropout layers are usually bypassed. However, to represent model uncertainty, dropout layers are also enabled during inference. Gal and Ghahramani [18] proves that dropouts can approximate Bayesian approximations in BNNs. Given a test record $\mathbf{x}$, model inference is carried out many times independently, yielding the outputs $\hat{y_1}, \hat{y_2}, \ldots, \hat{y_m}$. The mean of $\hat{y_1}, \hat{y_2}, \ldots, \hat{y_m}$ is the prediction for $\mathbf{x}$, and the variance of $\hat{y_1}, \hat{y_2}, \ldots, \hat{y_m}$ represents the model uncertainty with respect to $\mathbf{x}$.

Ensemble Methods. Ensemble methods refer to machine learning techniques such as Bagging and Boosting [1] that create multiple models and then combine them to make improved predictions. Ensemble methods usually produce more accurate results than a single model would. Lakshminarayanan et al. [27] utilizes ensemble methods to quantify model uncertainty. A set of models are induced either independently (e.g., in Bagging) or dependently (e.g., in Boosting). Given a test record $\mathbf{x}$ as input, these models give the outputs $\hat{y_1}, \hat{y_2}, \ldots, \hat{y_m}$, respectively. The mean of $\hat{y_1}, \hat{y_2}, \ldots, \hat{y_m}$ is the prediction for $\mathbf{x}$, and the variance of $\hat{y_1}, \hat{y_2}, \ldots, \hat{y_m}$ represents the uncertainty of the ensemble model with regards to $\mathbf{x}$.

Model-Dependent UQ Methods. Deep NNs often adopt some effective mechanisms such as self-attention [40] and Graph Neural Network (GNN) [43]. The model-dependent UQ methods combine UQ with these mechanisms. For example, the hierarchical stochastic self-attention (HSTO) can substitute the Multi-head Attention

and can quantify uncertainty during model inference [31]. HSTO utilizes the Gumbel-Softmax to cast the deterministic attention distribution for values in each self-attention head to be stochastic. The prediction and the uncertainty for a given test record $\mathbf{x}$ are obtained in the same way as in the MCD method.

Limitations of General-Purpose UQ Methods. Although the general-purpose methods can be directly applied to quantify uncertainty in BE models, they have following two limitations:

(1) *High Inference Overhead*: All of the methods require either multiple rounds of model inference or inference of multiple models to quantify uncertainty, which consumes a large amount of computational resources and inference time. Therefore, applying any general-purpose method to a learning-based BE model reduces the model's inference efficiency which was regarded as one of the main advantages of learning-based BE models over the what-if tools.

(2) *Low Accuracy*: The general-purpose methods are designed for a wide range of machine learning tasks such as image recognition and sequence prediction, without accounting for the special characteristics of learning-based BE models. Consequently, as will demonstrated in subsection 5.3, the general-purpose methods attain lower accuracy than our proposed UQ method.

(3) *Low Stability*: No single general-purpose method consistently performs well across all models and all datasets. When the dataset is out-of-distribution, these methods can sometimes yield results that are less reliable than random guesses, significantly decreasing the usability of the UQ methods.

4.2 UQ for Learning-based BE Models

Motivated by the above limitations, we design a more accurate and efficient UQ method that considers the structure of a learning-based BE model and the origins of uncertainty identified in subsection 3.1.

4.2.1 **Structure of Learning-based BE Model.** Let us first review the typical structure of a learning-based BE model. Given an index configuration $\mathcal{I}$ and a query q, the model predicts $c(q, \mathcal{I})$ or $B(q, \mathcal{I}_0, \mathcal{I})$. As depicted in Figure 3, the typical architecture of a learning-based BE model is composed of three modules.

(1) *Feature Extractor.* The feature extractor represents the information related to BE into a set of vectors that the model can process. These information includes the index configuration $\mathcal{I}$, the SQL statement of q, the query plan of q on the current index configuration $\mathcal{I}_0$, the database schema, the statistics of the database, the hardware environment, and so on. According to the data type, the features in these information can be classified into numerical features and categorical features which are encoded into a set of vectors $V^1 = \{\mathbf{v}_1^1, \mathbf{v}_2^1, \ldots, \mathbf{v}_t^1\}$. Depending on the model design, the number of vectors in V^1 can be fixed (e.g., in AMA [14]) or vary for different queries and environments (e.g., in LIB [37]). Subsequently, for each vector $\mathbf{v}_i^1$, the elements in $\mathbf{v}_i^1$ that encode the categorical features are mapped to higher-dimensional vectors using one-hot or learnable word embeddings, thereby obtaining a vector denoted as $\mathbf{v}_i^2$ which has more dimensions than $\mathbf{v}_i^1$. Thus, we have the model's input vector set $V^2 = \{\mathbf{v}_1^2, \mathbf{v}_2^2, \ldots, \mathbf{v}_t^2\}$.

(2) *Encoder.* Each vector in V^2 independently characterizes a facet of the information related to BE. The vectors in V^2 are typically quite sparse and need to be aggregated into one shorter and information-rich hidden vector $\mathbf{v}^h$ through various model structures. If the number of vectors in V^2 is variable for different queries, Recurrent Neural Network (RNN), GNN or Multi-head Attention can be used as the encoder that allows each vector in V^2 to extract related information from other vectors in V^2, thereby forming a vector set $V^h = Encoder(V^2) = \{\mathbf{v}_1^h, \mathbf{v}_2^h, \ldots, \mathbf{v}_t^h\}$ as an intermediate representation. Then, the model aggregates the vectors in V^h into a single vector $\mathbf{v}^h$ using max-pooling, average-pooling or aggregation through a super node, that is, $\mathbf{v}^h = Pool(V^h)$. If the number of vectors in V^2 is fixed, a Multi-Layer Perceptron (MLP) is typically used as the encoder to form the hidden vector $\mathbf{v}^h = MLP(V^2)$.

(3) *Predictor.* The predictor, which typically employs a MLP, receives the hidden vector $\mathbf{v}^h$ as input and generates the final output $\hat{y} = MLP(\mathbf{v}^h)$, that is, an estimation of $c(q, \mathcal{I})$ or $B(q, \mathcal{I}_0, \mathcal{I})$.

4.2.2 **Our Uncertainty Quantification Method.** In subsection 3.1, we have shown five origins of uncertainty in a learning-based BE model. A single UQ method is incapable of capturing various types of uncertainty due to either insufficient quantification capability or an overemphasis on one type of factor. To improve the accuracy and the stability of UQ, our approach employs different methods to quantify various types of uncertainty.

Quantifying Uncertainty U3 and U4. Uncertainty U3 and U4 is often due to insufficient fitting capabilities of the BE models or inadequate training data. In the three modules of the BE model, the Encoder is usually the key part, responsible for mapping the original rich information to a low-dimensional space, directly determining the performance of the final result. Thus, we use the model's ability to complete this process as a measure of its performance, to quantify the model's uncertainty about a particular input.

Inspired by the well-known concept of *AutoEncoder* [15], we add a Decoder network structure to the BE model which behaves reversely against the Encoder module. The Decoder tries to reconstruct the input V^1 of the Encoder from the intermediate representation V^h, i.e., $\hat{V}^1 = Decoder(V^h) = \{\hat{\mathbf{v}}_1^1, \hat{\mathbf{v}}_2^1, \ldots, \hat{\mathbf{v}}_t^1\}$. The quality of the reconstruction is assessed by the total mean squared error (MSE) between all pairs of vectors $\mathbf{v}_i$ and $\hat{\mathbf{v}}_i$, that is,

$$\mathcal{U}_1 = \frac{1}{t} \sum_{i=1}^{t} MSE(\mathbf{v}_i^1, \hat{\mathbf{v}}_i^1). \tag{2}$$

$\mathcal{U}_1$ quantifies the uncertainty in the BE model with respect to V^1.

The purpose of adding the Decoder to the BE model is as follows. For the Decoder to accurately reconstruct V^1 from V^h, two essential conditions must be met. First, the structure of the Encoder must be appropriate. The network structure of the Decoder should be symmetric to that of the Encoder. Only if the Encoder's structure can gradually eliminate unimportant features and reduce dimensionality, the Decoder can reverse this process and reconstruct V^1 based on the essential features. Second, the quality of the latent representation must be high. Given the correct structure, the quality of the Decoder's input also determines the quality of the reconstruction. Therefore, when the divergence between V^1 and $\hat{V}^1$ is minimal, the Encoder has processed V^1 effectively, indicating low uncertainty in this part of the model.

As depicted in Figure 4, the Encoder transforms V^2 to $\mathbf{v}^h$, but the Decoder transforms V^h to V^1. Obviously, the Encoder and the

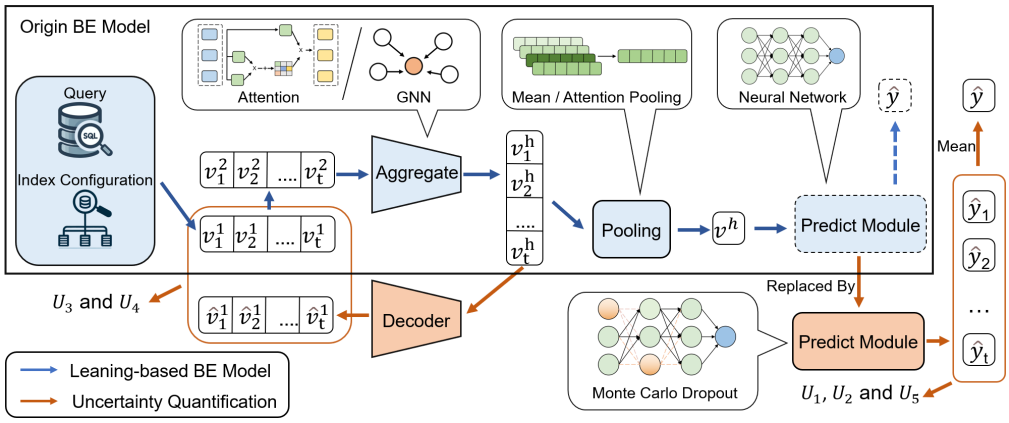


Figure 3: The BEAUTY framework for joint benefit estimation and uncertainty quantification.

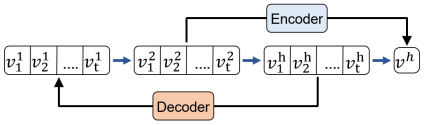


Figure 4: The transformation of vectors performed by the Encoder and the Decoder.

Decoder do not carry out strictly inverse processes. First, V^h is used as the input of the Decoder rather than $\mathbf{v}^h$. This is because the pooling in the Encoder that transforms V^h to $\mathbf{v}^h$ irreversibly compresses most information, e.g. the number of vectors in V^h. The loss of information makes subsequent reconstruction in the Decoder impossible. Second, V^1 is set as the target of the reconstruction process carried out by the Decoder instead of V^2. This is because after transforming the categorical features in the vectors in V^1 to embeddings, the sparsity of the vectors increases, and the richness of information decreases, which significantly increases the difficulty in reconstruction carried out by the Decoder. Our experiments verify that the design of the Decoder is reasonable and effective.

In Eq. (2), the MSE between $\mathbf{v}_i^1$ and $\hat{\mathbf{v}}_i^1$ is computed, where all dimensions of $\mathbf{v}_i^1$ and $\hat{\mathbf{v}}_i^1$ are considered as equally important. However, in practice, different features can have different impacts on BE, and important features can be assigned higher weights. In the engineering, MSE can be replaced by weighted MSE, and the practitioners can properly set weights to all dimensions.

Quantifying Uncertainty U1, U2 and U5. During training and inference, the model receives data with identical features but different labels, which prevents it from making correct distinctions and thus leads to erroneous outputs, including uncertainties U1, U2, and U5.

In the BE model, the features that determine $B(q, \mathcal{I}_0, \mathcal{I})$ are captured by the Encoder, and the Predictor maps these features to $B(q, \mathcal{I}_0, \mathcal{I})$. Therefore, we quantify the uncertainty U1, U2 and U5 by applying the MCD method to the Predictor. In particular, we add Monte Carlo dropouts to the original MLP of the predictor to obtain an approximation of BNNs. After the Encoder produces the hidden representation $\mathbf{v}^h$, a set of predictions $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m\}$ are generated by the approximated BNNs, that is, $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m\} = MLP_{MCD}(\mathbf{v}^h)$. The mean of $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m$ is returned as the prediction of $B(q, \mathcal{I}_0, \mathcal{I})$, and the uncertainty is quantified as

$$\mathcal{U}_2 = \mathrm{Var}(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m). \tag{3}$$

In the BE scenario, we prefer the MCD method to BNNs for two reasons. First, using BNNs requires making more changes to the BE model than using MCD. Second, the training cost of a BNN is usually higher because a BNN often uses multiple parameters to characterize the distribution of a single parameter of a neuron. For example, if a weight of a neuron follows the normal distribution $N(\mu, \sigma)$, it requires two parameters μ and σ to characterize this distribution. Therefore, a BNN requires multiple times of parameters than a normal NN, which significantly increases the training cost.

Model Training. After adding the Decoder module to the original BE model, we obtain the BE-UQ model that simultaneously completes the BE and the UQ tasks. Training the BE-UQ model must be different from training the original BE model and needs to be designed accordingly. In our index advisor, the BE-UQ model is trained in two phases. First, the BE model including the Feature Extractor, the Encoder and the Predictor is trained using the training method of the original BE model. Second, we freeze the parameters of the Feature Extractor and the Encoder and train the Decoder by minimizing the MSE loss. The rationale for this training method is as follows. Note that BE is the primary goal of the BE-UQ model, and UQ is the secondary goal. Our training method ensures that, no matter UQ is used or not, the accuracy of BE will not be compromised. Guaranteeing the accuracy of the Encoder also ensures that the useful input information is reasonably encoded in the intermediate representation V^h, which is the foundation for the Decoder to accurately reconstruct the input vector set V^1 and thus correctly quantify uncertainty.

Actually, there are many other ways to train the BE-UQ model. For example, the Encoder and the Decoder can be trained together first, followed by training the Predictor. However, this method causes the Encoder module to focus more on UQ rather than BE, which contradicts our goal. Therefore, our proposed training method is the most suitable for the index tuning task.

4.2.3 **Improving Efficiency.** Our UQ method inevitably introduces additional overhead. Here, we present a more efficient version.

In terms of quantification overhead, $\mathcal{U}_2$ is greater than $\mathcal{U}_1$: The Autoencoder requires an extra Decoder inference, while MCD necessitates dozens of inferences of the Prediction Module to ensure result stability [16].

In terms of the importance of the quantification, $\mathcal{U}_1$ is greater than $\mathcal{U}_2$: (1) Among the various uncertainties affecting $\mathcal{U}_2$, U1 and U2 arise from flaws in the model design. These can be resolved by increasing the variety of encoded information and designing a more effective feature extraction module. U5 belongs to aleatoric uncertainty, which cannot be eliminated due to the inherent randomness of query execution. However, in practical scenarios, if the same query is executed multiple times, the average time can be used as the estimation target. (2) The uncertainties affecting $\mathcal{U}_1$, U3 and U4, cannot be completely resolved: it is challenging for the system to gather all possible combinations of queries and index configurations, and the model cannot achieve complete accuracy on the training data. Therefore, when the trained model is applied, it will always encounter workloads that invalidate it.

Therefore, when the model's feature extraction module is effective and the encoded information is sufficient, the quantification of $\mathcal{U}_2$ can be turned off. Although this slightly reduces the accuracy of UQ, it can significantly improve the inference efficiency of the BE-UQ model. In cases where the time budget for index tuning is small, allowing the index advisor to enumerate more index configurations can sometimes improve the tuning results.

4.3 The Beauty Framework

Based on our UQ method presented in the previous subsection, we devise the Beauty framework that integrates the BE-UQ model into the index advisor to improve the quality of the tuning results.

4.3.1 **Uncertainty-Driven Benefit Estimation.** The benefit estimator in Beauty takes advantages of both the BE-UQ model and the what-if tools to improve the accuracy and the reliability of BE. Given a query q, the current index configuration $\mathcal{I}_0$ and a new index configuration $\mathcal{I}$ generated by the index configuration enumerator, the benefit estimator works as follows: First, the BE-UQ model produces the estimated benefit $\hat{B}(q, \mathcal{I}_0, \mathcal{I})$ and the quantified uncertainty $\mathcal{U}_1$ and $\mathcal{U}_2$. If $\mathcal{U}_1$ exceeds the threshold θ_1 specified on uncertainty U3 and U4, it indicates that $\hat{B}(q, \mathcal{I}_0, \mathcal{I})$ is likely to be significantly divergent from $B(q, \mathcal{I}_0, \mathcal{I})$. If $\mathcal{U}_2$ exceeds the threshold θ_2 specified on uncertainty U1, U2 and U5, it indicates that $\hat{B}(q, \mathcal{I}_0, \mathcal{I})$ attains a high variance. Therefore, if $\mathcal{U}_1 \leq \theta_1$ and $\mathcal{U}_2 \leq \theta_2$, the benefit estimator returns $\hat{B}(q, \mathcal{I}_0, \mathcal{I})$ as the result; otherwise, $B(q, \mathcal{I}_0, \mathcal{I})$ is estimated using the what-if tools.

The domain of θ_1 and θ_2 is $[0, +\infty)$. The setting of θ_1 and θ_2 has a great impact on the accuracy and the reliability of BE. However, as we observed in our experimental study, there is no universal setting for θ_1 and θ_2 that is effective for various databases, query workloads and system status. Instead, the optimal setting of θ_1 and θ_2 depends on the BE-UQ model structure, the hyperparameters of the model, the query workload, the database instance, and so on. Based on this observation, we devise a context-dependent method to set θ_1 and θ_2. In particular, after training the BE-UQ model, we make model inference for each instance $(q, \mathcal{I}_0, \mathcal{I})$ in the training set and collect the quantified uncertainty $\mathcal{U}_1$ and $\mathcal{U}_2$ for the training instance. The 90th percentile of $\mathcal{U}_1$ (or $\mathcal{U}_2$) for all training instances is designated as θ_1 (or θ_2).

4.3.2 **Uncertainty-Driven Model Updating.** The performance of the BE-UQ model may decline when the database or the query workload is changed. To keep the BE-UQ model accurate and stable, it must be updated timely once its performance tends to decrease. The existing methods for determining whether a machine learning model needs to be updated generally fall into two categories.

The first category of methods sample the predictions produced during model inference and compare them with the actual results. Model updating is required when the error is significantly large. There are two main disadvantages of this method. First, the actual benefits $B(q, \mathcal{I}_0, \mathcal{I})$ for the sampled instances $(q, \mathcal{I}_0, \mathcal{I})$ must be known. To obtain $B(q, \mathcal{I}_0, \mathcal{I})$, the indexes in $\mathcal{I}$ must be created, and the query q is actually executed based on the indexes in $\mathcal{I}$. This additionally incurs significant time and space costs. Second, this method might result in a misperception of good model accuracy due to random sampling errors.

The second category triggers model updates when a decrease in system operational efficiency is observed. However, these methods cause model updating to lag behind the occurrence of performance regression. Despite the accuracy of this method, the system performance has already decreased before the model is updated.

In our Beauty framework, if the what-if tool estimates benefits for a significant portion test instances $(q, \mathcal{I}_0, \mathcal{I})$, it likely indicates the BE-UQ model no longer fits the current dataset, workload, or system conditions and needs updating. Furthermore, the uncertainty $\mathcal{U}_1$ and $\mathcal{U}_2$ returned by the model can guide model updating.

(1) If the benefits of a significant portion of test instances $(q, \mathcal{I}_0, \mathcal{I})$ are estimated using the what-if tool due to $\mathcal{U}_1 > \theta_1$, this indicates that the BE-UQ model cannot provide effective hidden vectors under the current workload. For instance, if there have been substantial changes in query patterns and index configurations, the BE-UQ model fails to yield accurate results. Therefore, more data needs to be collected under the current workload to retrain or fine-tune the existing model.

(2) If the benefits of a significant portion of test instances $(q, \mathcal{I}_0, \mathcal{I})$ are estimated using the what-if tool due to $\mathcal{U}_2 > \theta_2$, this indicates that there are too many queries and index configurations in the current workload that the BE-UQ model cannot distinguish. Due to insufficient modeling information or defects in feature extraction, the model estimates a wide range of benefits for the same test instance, leading to unreliable results. The designer must identify the missing information or defects in feature extraction and redesign an appropriate model based on the current workload.

5 EMPIRICAL STUDY ON UNCERTAINTY QUANTIFICATION

In this section, we compare different UQ methods from three aspects, namely the UQ capability, the efficiency and the impact on the accuracy of BE. The main results are illustrated in Figure 5.

5.1 Datasets Construction

Three famous database benchmarks are adopted in the empirical study. Table 1 summarizes the databases and the workloads.

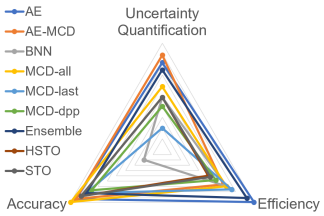


Figure 5: The performance of uncertainty quantification methods across various dimensions.

Table 1: Summary of databases and workloads and the maximum runtime for index tuning algorithm DTA.

Benchmark	DB SF/Size	# Queries	# Templates	# Synthetic Queries	Avg. # Plan Nodes	Maximum Runtime (mins)
TPC-H	SF = 10	570	19	0	14	20
TPC-DS	SF = 1	2700	90	0	29	45
JOB	13GB	3390	113	0	24	60
TPC-H+	SF = 10	950	19	380	10	30
TPC-DS+	SF = 1	4500	90	1800	19	60
JOB+	13GB	5650	113	2260	16	75

TPC-H: Following Leis et al. [26, 44], we randomly generated 30 queries from each template in TPC-H, excluding templates 2, 17, and 20. These were excluded because their queries have significantly longer execution times, which would disproportionately affect the overall workload cost. Indexes accelerating these queries are typically preferred over those for other templates.

TPC-DS: We also randomly generated 30 queries from each template in TPC-DS. Templates 4, 6, 9, 10, 11, 32, 35, 41, and 95 were excluded for the same reason given above.

JOB: This workload consists of 30 queries randomly generated from each template in JOB.

To increase the complexity of index tuning, we ingested synthetic queries into these workloads using the method devised by Yu et al. [44]. The enhanced workloads are referred to as **TPC-H+**, **TPC-DS+** and **JOB+**, respectively.

To the best of our knowledge, no systematic experiments have previously explored uncertainty involved in index tuning tasks. To investigate the impacts of UQ on the accuracy and the efficiency of learning-based benefit estimators, as well as their ability to quantify uncertainty, we designed the following experimental methodology:

We executed the Autoadmin and Extend algorithms on three workloads—**TPC-H+**, **TPC-DS+**, and **JOB+**—under various budgets to collect index configurations for index tuning. Upon materializing these configurations, we gathered query execution information, denoted as [(query, index config) → benefit], forming a comprehensive dataset. This dataset was then divided into three segments based on query templates and indexes:

Training Dataset: Used for model training, containing approximately 50% of the total data. This subset only includes some of the query templates and indexes.

Test Dataset: Comprises about 15% of the total data, where each query or index configuration appeared in the training set but was not simultaneously seen by the model. This segment tests the model's generalization capability within in-domain data.

Uncertainty Dataset: This dataset accounts for about 35% of the total data. It includes queries and indexes that are not present in the first two datasets, and no similar queries (from the same template) appear. For each model mentioned in subsubsection 5.2.1, we will test using these samples, excluding those with errors below the 90th percentile of the Training Dataset errors. This way, the remaining samples are those the model fails to predict correctly due to U1-U4, and the model should assign a high uncertainty value to them.These samples are designated as Uncertainty Samples.

There are two points to note. First, the samples that different models fail to predict accurately vary, resulting in an inconsistent number of Uncertainty Samples, but the difference does not exceed 10%. Second, the existing BE models in the index advisor use the average query costs rather than retaining all results, so we do not include U5-type uncertainty samples here.

5.2 Experimental Setup

5.2.1 Baseline. We have selected two well-known learning-based BE models as base models, and chosen five UQ methods. Together, these have allowed us to construct sixteen distinct models:

LIB: The BE model proposed in [37] encodes the operators and information in a query plan that can be affected by indexes, forming a variable-length Index Optimizable Operations Set. After encoding through Multi-head Attention, the Pooling by Multi-head Attention retrieves v^h, culminating in the output of the BE via a MLP. The associated UQ methods incorporated with this model are as follows:

(1) BNN: We have substituted the MLP in the Prediction Module with a BNN, referred to as LIB-BNN.

(2) MCD: Using the approach from [36], we created three models. LIB-MCD-all replaces all Dropout layers in LIB with MCD. LIB-MCD-last applies MCD only to the final layer. LIB-MCD-dpp, inspired by [16], uses determinantal point processes for sampling to enhance diversity with minimal inference time increase, incorporating MCD-dpp in the final layer.

(3) Ensemble: We selected the five best-performing models, each trained under different hyperparameters, to establish LIB-Ensemble.

(4) HSTO-Attention: We replaced the Multi-head Attention mechanism in LIB with two techniques introduced in [31], designated as LIB-STO and LIB-HSTO.

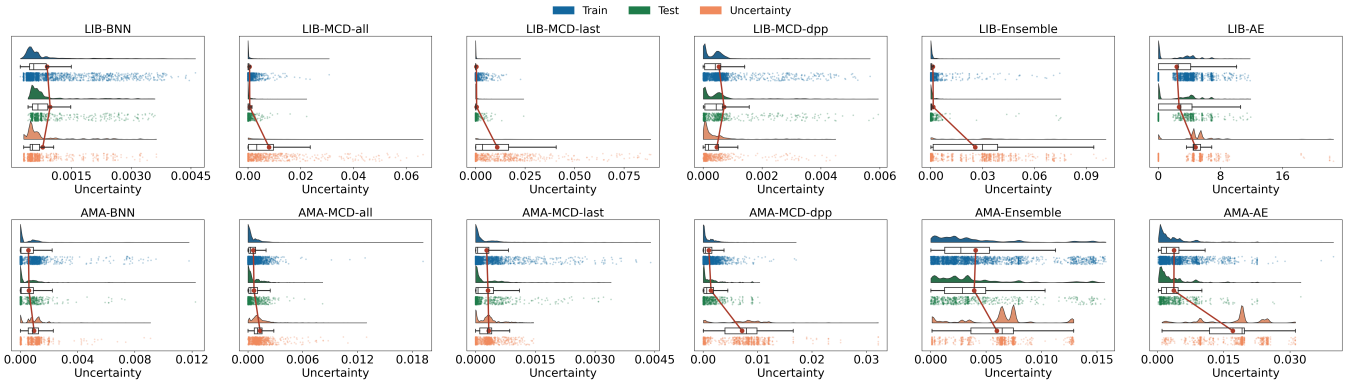
(5) Ours: The UQ method proposed in this paper results in two models: LIB-AE-MCD and LIB-AE, depending on whether MCD is enabled.

AMA: The model in [14] uses multiple channels to extract information and WeightedSum to compare query plans, selecting the faster one. Following [37], we replaced AMA's classification layer with a MLP for benefit estimation. Similarly, we integrated AMA with model-independent UQ methods, resulting in these models: AMA-BNN, AMA-MCD-all, AMA-MCD-last, AMA-MCD-dpp, AMA-Ensemble, AMA-AE, and AMA-AE-MCD.

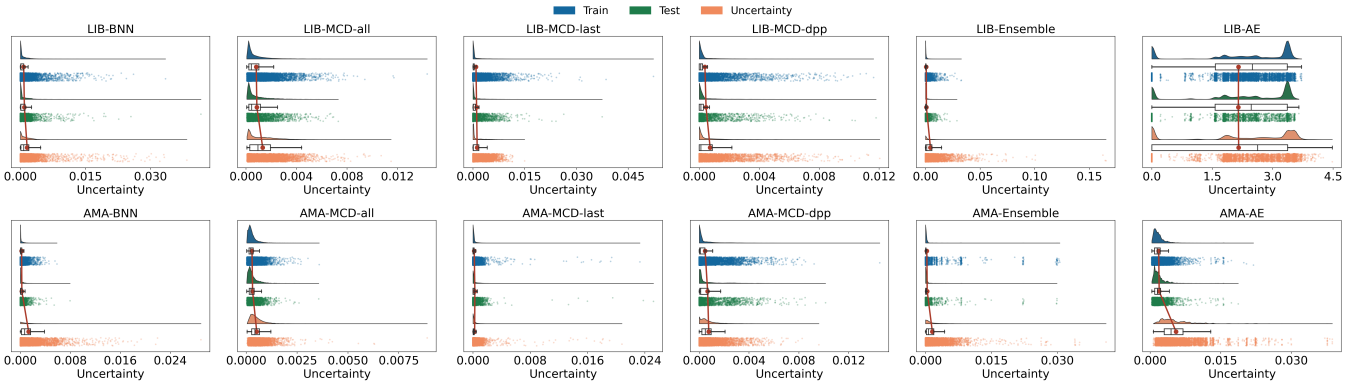
It is important to note that we will try to keep the two BE models consistent with the original paper, but complete consistency may not be achievable. Our main focus is on studying the impact of the UQ method, not comparing the two BE models.

Based on the models constructed with BNN, MCD, and HSTO-Attention, as well as the MCD module in ours, we perform inference 20 times [16] for each batch sample, using variance as the quantification of uncertainty. For those based on Ensemble, the variance of results from five models is used as the quantification of uncertainty.

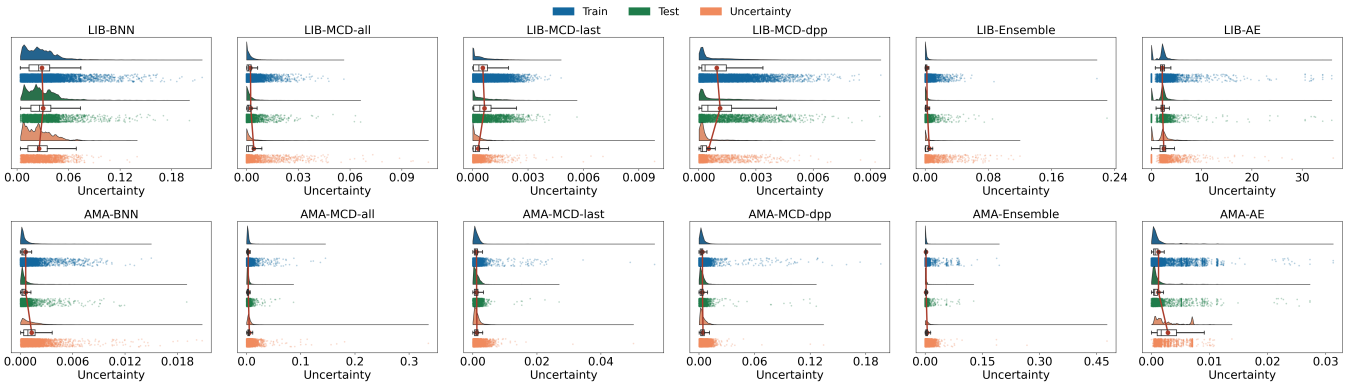
5.2.2 Implementation. We set the relative benefit $rb = \frac{B(q,\mathcal{I}_0,\mathcal{I})}{c(q,\mathcal{I}_0)}$ as the prediction target. To maintain compatibility with the LIB



(a) TPC-H+



(b) TPC-DS+



(c) JOB+

Figure 6: The distribution of quantified uncertainties across various benchmarks using different model-independent methods.

framework, we exclude data entries exhibiting negative benefits from our dataset. For each model discussed, we generate thirty sets of hyperparameters and conduct training on the training dataset. Subsequently, we retain the model that demonstrates the best performance based on evaluations using the test dataset.

5.2.3 Hardware & Software. The experiments were carried out on a Ubuntu server with two Intel Xeon 4210R CPUs (10 cores, 2.40GHz), 256GB of main memory, and one NVIDIA GeForce RTX 3060 GPU. The DBMS is PostgreSQL 12.13.

5.3 Results on Uncertainty Quantification

We present Raincloud plots of uncertainty distributions for sixteen models across various benchmarks. Figure 6 shows the plots for model-independent and AE methods. Figures 7, and 8 show the plots for model-dependent and AE-MCD methods, respectively. Each subplot depicts the uncertainty distribution of samples as calculated by the model for the Training, Test Dataset, and Uncertainty Samples. Consequently, the greater the uncertainty of the orange samples representing Uncertainty Samples in the graph and the higher their differentiation from other samples, the more precisely

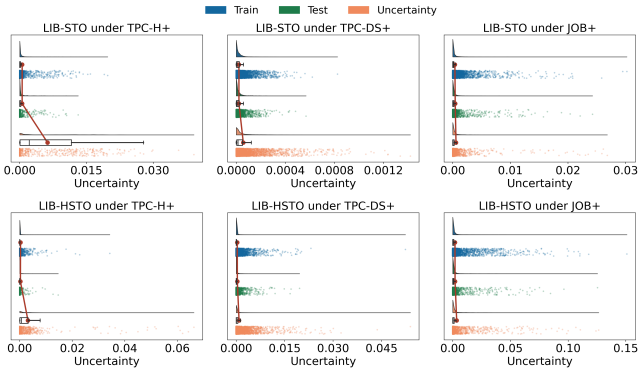


Figure 7: The uncertainty distribution of model-dependent methods.

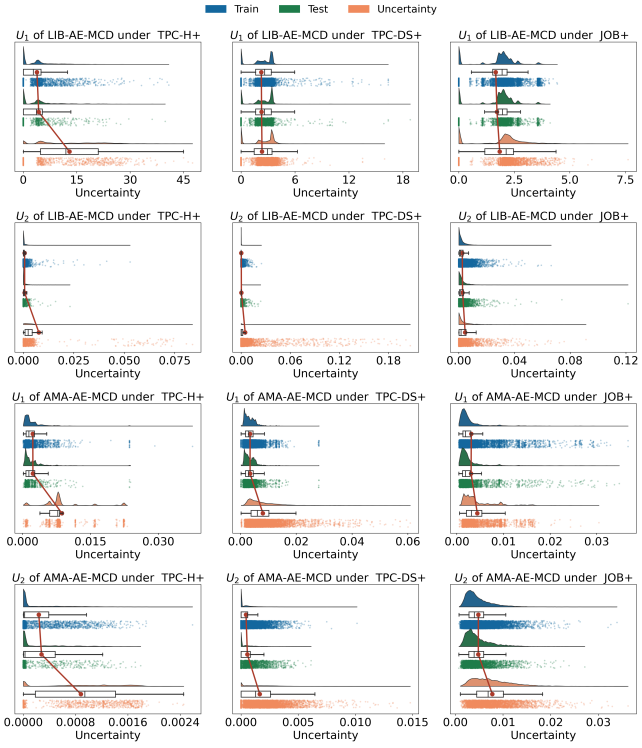


Figure 8: The uncertainty distribution of AE-MCD.

the model's uncertainty modeling is demonstrated, enabling it to effectively distinguish uncertainty data. The red lines in the graphs represent the average uncertainties across the three datasets.

Table 2 further illustrates the filtering capabilities of various methods for Uncertainty Samples. Since AE-MCD includes two UQ modules, we adopt the following approach to ensure comparability with other models: for $\mathcal{U}_1$ and $\mathcal{U}_2$, we select the 90th percentile of the uncertainty distribution from the Training Dataset as their corresponding thresholds. We calculate the proportion P of samples in the Training Dataset with uncertainty values lower than these thresholds. If $\mathcal{U}_1$ and $\mathcal{U}_2$ are probabilistically independent, this proportion should be 81%. For other models with only one UQ module, we employ the P-th percentile of the uncertainty distribution from the Training Dataset as a threshold. The table shows

Table 2: Percentage of Uncertainty Samples exceeding the uncertainty threshold set by the Training Dataset.

LIB-based Model	TPC-H+	TPC-DS+	JOB+	AMA-based Model	TPC-H+	TPC-DS+	JOB+
BNN	13.03	40.20	16.04	BNN	45.26	65.32	53.67
MCD-all	68.08	40.98	26.22	MCD-all	51.73	54.17	45.61
MCD-last	61.79	29.32	5.92	MCD-last	21.76	21.27	22.11
MCD-dpp	16.02	36.91	8.35	MCD-dpp	78.28	34.83	30.76
Ensemble	79.01	53.44	29.32	Ensemble	38.73	77.62	60.22
AE	71.85	31.13	30.93	AE	92.68	84.19	61.00
AE-MCD	81.80	57.65	32.88	AE-MCD	91.74	78.13	47.82
$\mathcal{U}_1$	51.88	27.21	20.68	$\mathcal{U}_1$	88.87	54.21	20.00
$\mathcal{U}_2$	59.70	43.46	20.76	$\mathcal{U}_2$	54.93	47.50	34.87
STO	68.41	34.86	23.51				
HSTO	63.64	35.13	24.58				

the proportion of samples in the Uncertainty Samples that exceed this threshold. The higher the proportion, the stronger the model's ability to identify uncertain samples. For each benchmark in the table, we highlight the model with the most effective filtering in green and the second most effective in blue. Note that $\mathcal{U}_1$ and $\mathcal{U}_2$, as parts of the AE-MCD, are excluded from this ranking.

From these results, we can draw the following conclusions:

First, the AE and AE-MCD models consistently perform the best, followed by the Ensemble method. The other methods lag significantly behind in both stability and optimal performance in UQ. The best performance is seen with the AMA-AE model under the TPC-H+, which can identify 92.68% of Uncertainty Samples while misclassifying about 19% of training samples, significantly outperforming general-purpose methods. In more than two-thirds of the benchmarks, the AE and AE-MCD models consistently rank among the top two, demonstrating better stability than the Ensemble method.

Second, the quality of uncertainty quantification is related to the base model and dataset. Across three benchmarks, the uncertainty modeling quality of the AMA base model consistently surpasses that of the LIB model, particularly evident in the proposed AE and AE-MCD models. This may be attributed to the fact that the model's ability to fit the data also determines its accuracy and capability in UQ. As illustrated in Table 4, AMA's accuracy is significantly higher than LIB's. Accurate outcomes necessitate effective encoding of inputs into a hidden vector, the foundational principle of our methodology, which accounts for these observations.

Overall, these 16 models demonstrate superior performance on the TPC-H+ bencamark compared to TPC-DS+, with the poorest performance observed on the JOB+ benchmark, where the most effective UQ methods filter only 61% of Uncertainty Samples.

Third, the settings for uncertainty thresholds significantly impact the ability to detect Uncertainty Samples. Uncertainty distribution within a dataset is usually confined to a narrow range, and the relationship between the threshold and this range dictates the model's detection capabilities. For example, in the AMA-Ensemble model under TPC-H+, setting a filtering threshold at the 81th percentile of the training dataset excludes only 38.73% of Uncertainty Samples, whereas a threshold at the 75th percentile increases exclusion to 73.34%.

Fourth, model-dependent methods of UQ did not demonstrate improved performance in this task. Neither model achieved optimal performance across the three benchmarks.

Table 3: Time for model inference per batch (ms).

Model	TPC-H+	TPC-DS+	JOB+	Model	TPC-H+	TPC-DS+	JOB+
LIB	**2.76**	**2.30**	**5.58**	**AMA**	**0.18**	**0.18**	**0.25**
BNN	61.85	44.46	50.75	BNN	10.77	7.41	10.54
MCD-all	184.59	45.03	155.14	MCD-all	6.19	2.94	4.90
MCD-last	60.00	89.63	133.04	MCD-last	5.23	4.13	5.23
MCD-dpp	60.01	147.86	83.63	MCD-dpp	260.82	78.39	51.45
Ensemble	22.72	14.15	29.90	Ensemble	1.82	0.80	1.71
AE	3.08	4.76	6.37	AE	0.52	0.51	0.51
AE-MCD	60.74	103.95	98.16	AE-MCD	10.28	9.82	9.26
STO	151.40	137.33	157.01				
HSTO	64.46	109.25	327.03				

Fifth, among the three MCD-based methods, MCD-all outperforms the others, achieving optimal results in five out of six scenarios. MCD-dpp provides a richer sampling diversity compared to MCD-last and outperforms it in the majority of scenarios.

5.4 Results on Prediction Accuracy and Inference Efficiency

Table 3 shows the time taken by different models to process a single batch, which includes predicting the relative error $\hat{rb}_{predicted}$ and estimating uncertainty, in milliseconds. Models in which the inference time with UQ is on the same order of magnitude as the base model are marked in green.

The primary factors influencing inference efficiency are the base model structure, model hyperparameters, and methods for UQ. LIB, utilizing heavier Multi-head Attention, requires significantly more inference time, up to an order of magnitude higher than AMA.For the models based on the same BE model, the optimal hyperparameters vary across different datasets, which can also lead to several times difference in inference efficiency. Methods like BNN, MCD, and HSTO necessitate multiple inferences on the same model, resulting in inference times that are proportionally related to the number of inferences, significantly exceeding the base model and AE by one to two orders of magnitude. Specifically, MCD-dpp introduces extra computational overhead due to sampling processes, notably affecting the faster base model AMA, and to a lesser extent, the heavier LIB model where it is less of a determining efficiency factor. The inference time for Ensemble models scales with the number of models involved; selecting five models during hyperparameter tuning has resulted in higher overall efficiency compared to other multiple-inference UQ methods. Of all methods evaluated, AE proves the most efficient, necessitating only an additional inference for the decoder network, which keeps its inference times close to those of the base model on all six benchmarks.

Table 4 evaluates the impact of incorporating UQ methods into BE models. We assess performance using the absolute difference between predicted and actual values, i.e. $|rb_{actual} - \hat{rb}_{predicted}|$. The table shows the 50th, 95th, and 99th percentiles of error distributions for LIB and AMA. For models to which UQ methods have been added, the table shows the change in error relative to the base model, where models that have the most negative impact are marked in red, and those with the most positive impact are marked in green.

Among all UQ methods, only BNN affected the accuracy of the BE model, while other methods did not cause significant impact. In the BNN structure, each neuron is replaced with a value sampled from a distribution, which doubles the number of network parameters (each value in a deterministic neuron corresponds to mean, variance, etc., in BNN) and expands the hyperparameter space (for instance, requiring a prior distribution), making the training process different. Hence, training becomes more challenging, requiring more hyperparameter tuning.

Comparing to the MCD-last, MCD-dpp had a higher effect on accuracy when AMA was used as the base model, more inference burden while more UQ capability, making it better suited for integration with heavier base models. , which might be related to the changes it made in the sampling process during inference. However, more research is needed to provide a definitive conclusion, which is beyond the scope of this study.

Among the other methods, since the AE method did not alter the original structure and parameters of the BE model after training, it theoretically should not affect the accuracy of the BE model.

6 EMPIRICAL STUDY ON INDEX TUNING

6.1 Experimental Setup

In this section, we explore whether modeling uncertainty can enhance index tuning results. Therefore, we integrated the aforementioned BE model into the Beauty framework and performed index tuning on six benchmarks under different budget constraints.

6.1.1 Implementation. Leis et al. [26] show that no single index configuration enumeration algorithm delivers optimal performance in all scenarios. In our experiments, we used DTA [7], which balances runtime and solution quality well across various scenarios. This algorithm has a set maximum runtime and stops after enumerating all configurations or reaching this limit. However, following Leis et al.'s [26] implementation, the algorithm completes the current seed's configuration even if it exceeds the time limit, occasionally surpassing the maximum runtime. We adjust the maximum runtime based on workload size, as shown in Table 1.

Based on the performance of various models mentioned in Section section 5, we selected nine models in our experiment: LIB, LIB-Ensemble, LIB-AE, LIB-AE-MCD, AMA, AMA-Ensemble, AMA-AE, AMA-AE-MCD, and what-if. The thresholds for each model were set as described in Section subsection 5.3. When the uncertainty value of the benefits estimated by the BE-UQ model for a test instance exceeds the threshold, we use what-if tools to estimate the benefits of that test instance.

6.1.2 Evaluation Metrics. We use specific metrics to assess the performance of index tuning. Given a workload W, $\mathcal{I}_0$ represents the initial set of indexes, and $\mathcal{I}$ the set of selected indexes. We measure the quality of replacing $\mathcal{I}_0$ with $\mathcal{I}$ by the relative improvement in W's cost: $\max\left(0, \frac{B(W,\mathcal{I}_0,\mathcal{I})}{c(W,\mathcal{I}_0)}\right) = \max\left(0, 1 - \frac{c(W,\mathcal{I})}{c(W,\mathcal{I}_0)}\right)$. On some occasions, if the selected indexes increase the cost, they must be discarded, and thus the improvement in cost is zero.

6.2 Results on Index Tuning

Figure 9 illustrates the index tuning results of nine methods under various budgets across six benchmarks. Table 5 presents detailed statistics of cost improvements, highlighting the best performances in green and the second best in blue. From these results, we can draw the following conclusions:

First, integrating the UQ module into the two base models significantly improved index tuning results. The LIB model has seen a significant reduction in the number of worst-case scenarios, while

Table 4: The impact of uncertainty quantification methods on the prediction errors. Models that have the most negative impact are marked in red, while those with the most positive impact are marked in green.

Dataset		LIB	BNN	MCD-all	MCD-last	MCD-dpp	Ensemble	STO	HSTO	AE	AE-MCD	AMA	BNN	MCD-all	MCD-last	MCD-dpp	Ensemble	AE	AE-MCD
TPC-H+	50th	**0.13**	+0.10	+0.01	+0.02	+0.04	+0.00	+0.03	+0.00	+0.09	+0.05	**0.01**	+0.02	+0.00	+0.00	+0.02	+0.02	+0.00	+0.00
	95th	**0.47**	+0.02	+0.00	+0.00	-0.04	+0.00	+0.00	+0.00	+0.05	+0.01	**0.05**	+0.06	+0.01	+0.03	+0.00	+0.03	+0.00	+0.02
	99th	**0.65**	-0.11	-0.05	-0.04	-0.07	-0.01	+0.00	-0.05	-0.12	-0.01	**0.11**	+0.07	+0.00	+0.00	+0.03	+0.01	-0.03	+0.02
TPC-DS+	50th	**0.06**	+0.01	+0.00	-0.01	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00	**0.01**	+0.01	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00
	95th	**0.44**	+0.03	+0.00	-0.02	+0.01	+0.00	+0.00	-0.02	+0.00	-0.04	**0.20**	+0.06	+0.02	+0.00	+0.09	+0.03	+0.01	+0.01
	99th	**0.71**	+0.00	+0.00	+0.03	+0.00	+0.00	+0.00	+0.00	+0.02	+0.00	**0.34**	+0.08	+0.06	+0.05	+0.07	+0.04	+0.05	+0.07
JOB+	50th	**0.19**	+0.13	+0.01	+0.03	+0.04	+0.00	+0.04	+0.05	+0.03	+0.01	**0.03**	+0.02	+0.00	+0.00	+0.05	+0.00	+0.00	+0.01
	95th	**0.57**	+0.04	+0.00	+0.00	+0.00	+0.00	+0.02	+0.00	+0.03	+0.00	**0.19**	+0.06	-0.01	+0.00	+0.05	+0.00	+0.00	+0.04
	99th	**0.76**	+0.16	-0.02	+0.02	+0.00	-0.02	+0.07	+0.09	+0.04	+0.04	**0.37**	+0.17	+0.00	+0.00	+0.11	+0.00	+0.08	+0.10

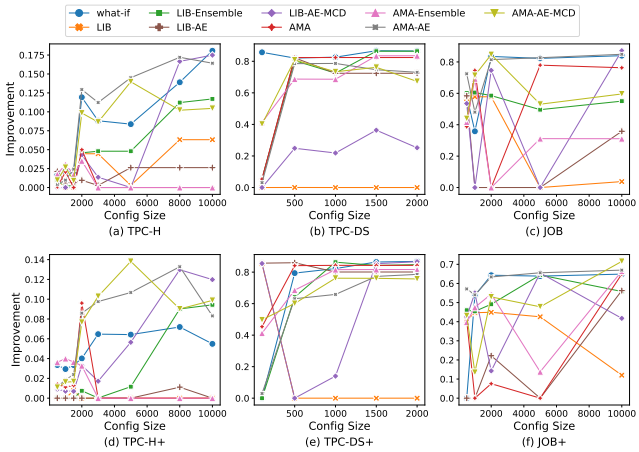


Figure 9: Cost improvements achieved by different benefit estimation models.

Table 5: Rankings of cost improvements achieved by models.

Method	# 1st Place	# 2nd Place	# 3rd Place	# Worst
What-if	8	8	11	1
LIB	0	0	1	10
LIB-Ensemble	1	4	4	3
LIB-AE	3	1	1	4
LIB-AE-MCD	3	7	0	4
AMA	3	3	3	6
AMA-Ensemble	4	2	6	7
AMA-AE	10	7	5	0
AMA-AE-MCD	4	4	5	0

the AMA model has experienced a substantial increase in the number of optimal configurations. Additionally, when integrating AE and AE-MCD into the AMA model, the number of worst-case scenarios decreased from six to zero.

Second, the performance of the integrated AE-MCD model is superior to that of the Ensemble-based model. As shown in Table 2, AE-MCD's ability to quantify uncertainty surpasses that of Ensemble, allowing it to more accurately identify the parts of the learning-based BE model that it cannot accurately estimate, thereby improving BE's accuracy. The DTA algorithm requires a specified maximum runtime, and usually, it cannot enumerate all index configurations by the end. Therefore, the greater the delay in BE-UQ inference, the fewer configurations the index advisor can explore. As seen in Table 3, AE-MCD has a higher delay. Thus, by providing more accurate uncertainty quantifications, AE-MCD finds better results with fewer configurations.

Third, AE's results are slightly better than AE-MCD's. The accuracy of the LIB base model is not high, and in this situation, AE-MCD's UQ accuracy is higher than AE's, enabling it to identify as many uncertain data points as possible, resulting in better performance than AE. Conversely, when using AMA as the base model, AE achieves higher identification accuracy and lower inference delay, allowing for more accurate enumeration of more index configurations. Therefore, it achieves the highest number of optimal configurations and surpasses the what-if tools. For index tuning tasks, AE-MCD is preferable when there is sufficient time available for tuning. If time is limited and the BE model demonstrates high accuracy, selecting AE can result in a better index configuration.

Fourth, in scenarios abundant with Uncertainty Samples, the what-if approach remains a stable and precise method, achieving the best or nearly the best results across all six benchmarks. This consistency forms the foundation of this study, which aims to model the uncertainty of learning-based models and combine the accuracy of these models with the reliability of what-if tools.

7 CONCLUSION

In this paper, we address an important yet unexplored question: Can uncertainty enable better learning-based index tuning? To answer this question, we focus on the core module of index tuning and prioritize the improvement of the learning-based BE model. We then divide the investigation into two sub-questions: (1) Can UQ methods accurately measure the reliability of model results? (2) How can a BE model that provides uncertainty estimates be integrated into an index advisor, and what effects will this integration produce?

To address these two questions, we first analyze the origins and characteristics of model uncertainty in BE tasks. Based on existing general UQ methods, we design a new quantification framework better suited to the task's characteristics and propose an alternative version to improve efficiency. Experimental validation shows that our method outperforms existing quantification methods in five out of six scenarios. The alternative version significantly improves the efficiency of UQ, albeit with a slight sacrifice in quantification accuracy. Next, our proposed Beauty framework leverages the stability of what-if tools as a supplement to the learning-based BE model. In end-to-end experiments, this framework significantly improves the quality of index tuning results.

Our findings affirm that integrating uncertainty can improve learning-based index tuning. Our framework focuses on the BE problem, marking an initial exploration in this area. Future research will explore the role of uncertainty in other index advisor modules.

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China (Grant No. 62072138).

REFERENCES

[1] Omer Achrack, Ouriel Barzilay, and Raizy Kellerman. 2020. Multi-Loss Sub-Ensembles for Accurate Classification with Uncertainty Estimation. *CoRR* abs/2010.01917 (2020). arXiv:2010.01917 https://arxiv.org/abs/2010.01917

[2] Alexander Amini, Ava Soleimany, Sertac Karaman, and Daniela Rus. 2018. Spatial Uncertainty Sampling for End-to-End Control. *CoRR* abs/1805.04829 (2018). arXiv:1805.04829 http://arxiv.org/abs/1805.04829

[3] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight Uncertainty in Neural Networks. *CoRR* abs/1505.05424 (2015). arXiv:1505.05424 http://arxiv.org/abs/1505.05424

[4] Nicolas Bruno and Surajit Chaudhuri. 2005. Automatic Physical Database Tuning: A Relaxation-based Approach. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, Fatma Özcan (Ed.). ACM, 227–238. https://doi.org/10.1145/1066157.1066184

[5] George D. C. Cavalcanti, Luiz S. Oliveira, Thiago J. M. Moura, and Guilherme V. Carvalho. 2016. Combining diversity measures for ensemble pruning. *Pattern Recognit. Lett.* 74 (2016), 38–45. https://doi.org/10.1016/J.PATREC.2016.01.029

[6] Surajit Chaudhuri, Ashish Kumar Gupta, and Vivek R. Narasayya. 2002. Compressing SQL workloads. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002*, Michael J. Franklin, Bongki Moon, and Anastassia Ailamaki (Eds.). ACM, 488–499. https://doi.org/10.1145/564691.564747

[7] Surajit Chaudhuri and Vivek Narasayya. 2020. Anytime Algorithm of Database Tuning Advisor for Microsoft SQL Server. (June 2020). https://www.microsoft.com/en-us/research/publication/anytime-algorithm-of-database-tuning-advisor-for-microsoft-sql-server/

[8] Surajit Chaudhuri and Vivek R. Narasayya. 1997. An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld (Eds.). Morgan Kaufmann, 146–155.

[9] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee. 2019. Gaussian YOLOv3: An Accurate and Fast Object Detector Using Localization Uncertainty for Autonomous Driving. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 502–511. https://doi.org/10.1109/ICCV.2019.00059

[10] Douglas Comer. 1978. The Difficulty of Optimum Index Selection. *ACM Trans. Database Syst.* 3, 4 (1978), 440–445. https://doi.org/10.1145/320289.320296

[11] Debabrata Dash, Neoklis Polyzotis, and Anastasia Ailamaki. 2011. CoPhy: A Scalable, Portable, and Interactive Index Advisor for Large Workloads. *Proc. VLDB Endow.* 4, 6 (2011), 362–372. https://doi.org/10.14778/1978665.1978668

[12] Shaleen Deep, Anja Gruenheid, Paraschos Koutris, Jeffrey Naughton, and Stratis Viglas. 2020. Comprehensive and Efficient Workload Compression. *Proc. VLDB Endow.* 14, 3 (2020), 418–430. https://doi.org/10.14778/3430915.3430931

[13] John S. Denker, Daniel B. Schwartz, Ben S. Wittner, Sara A. Solla, Richard E. Howard, Lawrence D. Jackel, and John J. Hopfield. 1987. Large Automatic Learning, Rule Extraction, and Generalization. *Complex Syst.* 1, 5 (1987). http://www.complex-systems.com/abstracts/v01_i05_a02.html

[14] Bailu Ding, Sudipto Das, Ryan Marcus, Wentao Wu, Surajit Chaudhuri, and Vivek R. Narasayya. 2019. AI Meets AI: Leveraging Query Executions to Improve Index Recommendations. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 1241–1258. https://doi.org/10.1145/3299869.3324957

[15] Ganggang Dong, Guisheng Liao, Hongwei Liu, and Gangyao Kuang. 2018. A review of the autoencoder and its variants: A comparative perspective from target recognition in synthetic-aperture radar images. *IEEE Geoscience and Remote Sensing Magazine* 6, 3 (2018), 44–68.

[16] Kirill Fedyanin, Evgenii Tsymbalov, and Maxim Panov. 2021. Dropout Strikes Back: Improved Uncertainty Estimation via Diversity Sampling. In *Recent Trends in Analysis of Images, Social Networks and Texts - 10th International Conference, AIST 2021, Tbilisi, Georgia, December 16-18, 2021, Revised Supplementary Proceedings (Communications in Computer and Information Science)*, Evgeny Burnaev, Dmitry I. Ignatov, Sergei Ivanov, Michael Yu. Khachay, Olessia Koltsova, Andrei Kutuzov, Sergei O. Kuznetsov, Natalia V. Loukachevitch, Amedeo Napoli, Alexander Panchenko, Panos M. Pardalos, Jari Saramäki, Andrey V. Savchenko, Evgenii Tsymbalov, and Elena Tutubalina (Eds.), Vol. 1573. Springer, 125–137. https://doi.org/10.1007/978-3-031-15168-2_11

[17] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. 2018. Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network For Lidar 3D Vehicle Detection. In *21st International Conference on Intelligent Transportation Systems, ITSC 2018, Maui, HI, USA, November 4-7, 2018*, Wei-Bin Zhang, Alexandre M. Bayen, Javier J. Sánchez Medina, and Matthew J. Barth (Eds.). IEEE, 3266–3273. https://doi.org/10.1109/ITSC.2018.8569814

[18] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 (JMLR Workshop and Conference Proceedings)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.), Vol. 48. JMLR.org, 1050–1059. http://proceedings.mlr.press/v48/gal16.html

[19] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna M. Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiaoxiang Zhu. 2023. A survey of uncertainty in deep neural networks. *Artif. Intell. Rev.* 56, S1 (2023), 1513–1589. https://doi.org/10.1007/S10462-023-10562-9

[20] Alex Graves. 2011. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger (Eds.). 2348–2356. https://proceedings.neurips.cc/paper/2011/hash/7eb3c8be3d411e8ebfab08eba5f49632-Abstract.html

[21] Huaping Guo, Hongbing Liu, Ran Li, Chang-an Wu, Yibo Guo, and Mingliang Xu. 2018. Margin & diversity based ordering ensemble pruning. *Neurocomputing* 275 (2018), 237–246. https://doi.org/10.1016/J.NEUCOM.2017.06.052

[22] Lars Kai Hansen and Peter Salamon. 1990. Neural Network Ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.* 12, 10 (1990), 993–1001. https://doi.org/10.1109/34.58871

[23] José Miguel Hernández-Lobato and Ryan P. Adams. 2015. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR Workshop and Conference Proceedings)*, Francis R. Bach and David M. Blei (Eds.), Vol. 37. JMLR.org, 1861–1869. http://proceedings.mlr.press/v37/hernandez-lobatoc15.html

[24] Eyke Hüllermeier and Willem Waegeman. 2021. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach. Learn.* 110, 3 (2021), 457–506. https://doi.org/10.1007/S10994-021-05946-3

[25] Piotr Kolaczkowski. 2008. Compressing Very Large Database Workloads for Continuous Online Index Selection. In *Database and Expert Systems Applications, 19th International Conference, DEXA 2008, Turin, Italy, September 1-5, 2008. Proceedings (Lecture Notes in Computer Science)*, Sourav S. Bhowmick, Josef Küng, and Roland R. Wagner (Eds.), Vol. 5181. Springer, 791–799. https://doi.org/10.1007/978-3-540-85654-2_71

[26] Jan Kossmann, Stefan Halfpap, Marcel Jankrift, and Rainer Schlosser. 2020. Magic Mirror in My Hand, Which Is the Best in the Land?: An Experimental Evaluation of Index Selection Algorithms. *Proc. VLDB Endow.* 13, 12 (2020), 2382–2395. https://doi.org/10.14778/3407790.3407832

[27] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 6402–6413. https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html

[28] Lewis H Mervin, Simon Johansson, Elizaveta Semenova, Kathryn A Giblin, and Ola Engkvist. 2021. Uncertainty quantification in drug design. *Drug discovery today* 26, 2 (2021), 474–489.

[29] Tanya Nair, Doina Precup, Douglas L. Arnold, and Tal Arbel. 2020. Exploring uncertainty measures in deep networks for Multiple sclerosis lesion detection and segmentation. *Medical Image Anal.* 59 (2020). https://doi.org/10.1016/J.MEDIA.2019.101557

[30] Manfred Opper and Cédric Archambeau. 2009. The Variational Gaussian Approximation Revisited. *Neural Comput.* 21, 3 (2009), 786–792. https://doi.org/10.1162/NECO.2008.08-07-592

[31] Jiahuan Pei, Cheng Wang, and György Szarvas. 2022. Transformer Uncertainty Estimation with Hierarchical Stochastic Attention. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 11147–11155. https://doi.org/10.1609/AAAI.V36I10.21364

[32] Abhijit Guha Roy, Sailesh Conjeti, Nassir Navab, and Christian Wachinger. 2019. Bayesian QuickNAT: Model uncertainty in deep whole-brain segmentation for structure-wise quality control. *NeuroImage* 195 (2019), 11–22. https://doi.org/10.1016/J.NEUROIMAGE.2019.03.042

[33] Rainer Schlosser, Jan Kossmann, and Martin Boissier. 2019. Efficient Scalable Multi-Attribute Index Selection Using Recursive Strategies. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (Macao, Macao, 2019-04). IEEE, 1238–1249. https://doi.org/10.1109/ICDE.2019.00113

[34] Philipp Seeböck, José Ignacio Orlando, Thomas Schlegl, Sebastian M. Waldstein, Hrvoje Bogunovic, Sophie Klimscha, Georg Langs, and Ursula Schmidt-Erfurth. 2020. Exploiting Epistemic Uncertainty of Anatomy Segmentation for Anomaly Detection in Retinal OCT. *IEEE Trans. Medical Imaging* 39, 1 (2020), 87–98. https://doi.org/10.1109/TMI.2019.2919951

[35] Luciana Separovic, Renan S Simabukuro, Aldo R Couto, Maria Luiza G Bertanha, Francielle RS Dias, Adriano Y Sano, Arthur M Caffaro, and Felipe R Lourenco.

2023. Measurement uncertainty and conformity assessment applied to drug and medicine analyses–a review. *Critical Reviews in Analytical Chemistry* 53, 1 (2023), 123–138.

[36] Artem Shelmanov, Evgenii Tsymbalov, Dmitry Puzyrev, Kirill Fedyanin, Alexander Panchenko, and Maxim Panov. 2021. How Certain is Your Transformer?. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty (Eds.). Association for Computational Linguistics, 1833–1840. https://doi.org/10.18653/V1/2021.EACL-MAIN.157

[37] Jiachen Shi, Gao Cong, and Xiaoli Li. 2022. Learned Index Benefits: Machine Learning Based Index Performance Estimation. *Proc. VLDB Endow.* 15, 13 (2022), 3950–3962. https://www.vldb.org/pvldb/vol15/p3950-shi.pdf

[38] Tarique Siddiqui, Saehan Jo, Wentao Wu, Chi Wang, Vivek Narasayya, and Surajit Chaudhuri. 2022. ISUM: Efficiently Compressing Large and Complex Workloads for Scalable Index Tuning. In *Proceedings of the 2022 International Conference on Management of Data* (New York, NY, USA, 2022-06-10) *(SIGMOD '22)*. Association for Computing Machinery, 660–673. https://doi.org/10.1145/3514221.3526152

[39] Tarique Siddiqui, Wentao Wu, Vivek Narasayya, and Surajit Chaudhuri. 2022. DISTILL: Low-Overhead Data-Driven Techniques for Filtering and Costing Indexes for Scalable Index Tuning. *Proc. VLDB Endow.* 15, 10 (2022), 2019–2031. https://doi.org/10.14778/3547305.3547309

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2017-12-04) *(NIPS'17)*. Curran Associates Inc., 6000–6010.

[41] Zilong Wang, Qixiong Zeng, Ning Wang, Haowen Lu, and Yue Zhang. 2023. CEDA: Learned Cardinality Estimation with Domain Adaptation. *Proc. VLDB Endow.* 16, 12 (2023), 3934–3937. https://doi.org/10.14778/3611540.3611589

[42] Wentao Wu, Chi Wang, Tarique Siddiqui, Junxiong Wang, Vivek R. Narasayya, Surajit Chaudhuri, and Philip A. Bernstein. 2022. Budget-aware Index Tuning with Reinforcement Learning. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1528–1541. https://doi.org/10.1145/3514221.3526128

[43] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[44] Tao Yu, Zhaonian Zou, Weihua Sun, and Yu Yan. 2024. Refactoring Index Tuning Process with Benefit Estimation. *Proc. VLDB Endow.* 17, 7 (2024), 1528–1541. https://www.vldb.org/pvldb/vol17/p1528-zou.pdf

[45] Yao Zhang and Alpha A. Lee. 2019. Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning. *CoRR* abs/1902.00925 (2019). arXiv:1902.00925 http://arxiv.org/abs/1902.00925