Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction

Benjamin Hilprecht
Technical University of Darmstadt
benjamin.hilprecht@cs.tu-darmstadt.de

Carsten Binnig
Technical University of Darmstadt & DFKI
carsten.binnig@cs.tu-darmstadt.de

ABSTRACT

In this paper, we introduce zero-shot cost models, which enable learned cost estimation that generalizes to unseen databases. In contrast to state-of-the-art workload-driven approaches, which require to execute a large set of training queries on every new database, zero-shot cost models thus allow to instantiate a learned cost model out-of-the-box without expensive training data collection. To enable such zero-shot cost models, we suggest a new learning paradigm based on pre-trained cost models. As core contributions to support the transfer of such a pre-trained cost model to unseen databases, we introduce a new model architecture and representation technique for encoding query workloads as input to those models. As we will show in our evaluation, zero-shot cost estimation can provide more accurate cost estimates than state-of-the-art models for a wide range of (real-world) databases without requiring any query executions on unseen databases. Furthermore, we show that zero-shot cost models can be used in a few-shot mode that further improves their quality by retraining them just with a small number of additional training queries on the unseen database.

PVLDB Reference Format:
Benjamin Hilprecht and Carsten Binnig. Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction. PVLDB, 15(11): 2361 - 2374, 2022.
doi:10.14778/3551793.3551799

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/DataManagementLab/zero-shot-cost-estimation.

1 INTRODUCTION

Motivation. Accurate physical cost estimation (i.e., estimating query latencies) is crucial for query optimization in DBMSs. Classically, cost estimation is performed using models that make several simplifying assumptions. As a result, such models often over- or underestimate runtimes, leading to suboptimal planning decisions that degrade the overall query performance [17]. Recently, machine learning (ML) has thus been used for learned cost models that do not need to make such simplifying assumptions [30].

While it was shown that the cost estimates of such learned cost models are significantly more accurate than those of the traditional cost models, the existing approaches rely on *workload-driven* learning where models have to observe thousands of queries on the same database[1] for which the cost prediction should be performed. This workload execution is required to gather the training data, which can take hours (or days) since tens of thousands of queries need to be executed on potentially large databases.

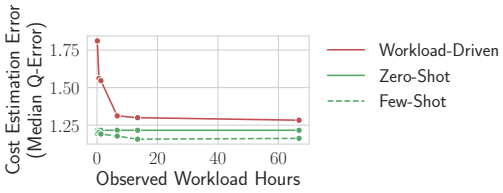


Figure 1: Cost Estimation Errors on the IMDB database. While workload-driven approaches [30] require many hours of workload executions as training data, our zero-shot cost model supports the unseen IMDB database *out-of-the-box* and provides highly accurate cost estimates. If a workload is observed however, the zero-shot model can be fine-tuned, which further improves the performance.

In Figure 1, we show the cost estimation accuracy depending on how many hours we allow for gathering the training data for a workload-driven model. As we can see, even for a medium-sized database such as IMDB, it takes more than 5 hours of running queries on this database to gather enough training data such that the cost estimation model can provide a decent accuracy.

Unfortunately, collecting training data by running queries is not a one-time effort. In fact, the training data collection has to be repeated for every new database a learned model should be deployed for. This is due to the fact that current model architectures for workload-driven learning tie a trained model to a particular database instance. Consequently, for every (new) unseen database we not only have to train a model from scratch but also gather training data in the form of queries. And even for the same database, in case of changed data characteristics due to updates, training data collection needs to be repeated. Overall, these repeated high costs for obtaining training data for unseen databases render workload-driven learning unattractive for many practical deployments.

Contributions. In this paper, we thus suggest a new learning paradigm for cost estimation called zero-shot cost models that reduces these high efforts. The general idea behind zero-shot cost models is motivated by recent advances in transfer learning of machine learning models. While a wide spectrum of methods have been proposed already to tackle zero-shot learning in domains such as NLP [3] or computer vision [15], no approaches for zero-shot learning exist for learned DBMS components and in particular also for cost models. To enable this, as a core contribution in this paper, we propose a new query and data representation that allows zero-shot cost models to be pre-trained across databases and thus be

[1]Throughout this paper, we use the term database to refer to a particular dataset with certain data characteristics.

Proceedings of the VLDB Endowment, Vol. 15, No. 11 ISSN 2150-8097.
doi:10.14778/3551793.3551799

used out-of-the-box (or with minimal fine-tuning only) on unseen databases.

In fact, as depicted in Figure 1 zero-shot cost models can thus provide a high accuracy and even outperform existing workload-driven approaches that have been trained on large sets of training queries. One could now argue that it might be a significant effort to collect sufficient training data across databases for pre-training a zero-shot model. However, in contrast to workload-driven models, which require training data for every unseen database, training data collection is a one-time effort; i.e., once trained the zero shot model can be used for any new unseen database. In fact, in our evaluation we show that zero-shot models can provide high accuracies for a wide variety of real-world databases. Moreover, historical traces can be used, which eliminates the need to collect any training data. For example, cloud providers such as AWS, Microsoft, or Google, typically anyway keep logs of their customer workloads, which could directly be used as training data for zero-shot learning without collecting any further training data.

A key aspect to enable zero-shot learning is that a cost model can be transferred to new (unseen) databases, i.e., the models leverage observed query executions on a variety of different databases to predict runtimes on new (unseen) databases. However, state-of-the-art model architectures used for workload-driven learning do not support this training and inference mode since they are tied to a particular database. As a core novel contribution for zero-shot cost models we thus devise a new model architecture based on a representation of queries that generalizes across databases using a *transferable* representation with features such as the tuple width that can be derived from any database. Moreover, zero-shot models *separate concerns*; i.e., data characteristics of a new database (e.g., rows of tables) are not implicitly learned as in classical workload-driven learning (which hinders generalization), but are provided as input to the model.

Another core question for zero-shot models is at which point a sufficient amount of different training databases and workloads was observed to generalize robustly to unseen databases. To answer this question, as a second contribution in this paper we derive a method to estimate how accurate the runtime estimations of zero-shot models will be for unseen databases. We also discuss how to address cases of workload drifts where the zero-shot models are expected to generalize less robustly. Furthermore, we also show that zero-shot models are widely applicable beyond cost models for query optimizers for single-node DBMSs, which is the main focus of this paper. For instance, we have initial results that zero-shot cost models can be naturally extended to distributed DBMS or even other use cases such as providing cost estimates for design advisors where the goal is to automatically find a suitable database design (e.g., a set of indexes) for a given workload. Due to space constraints, we defer these results to an extended technical report.

Finally, in our extensive experimental evaluation, we verify that zero-shot cost models generalize robustly to unseen databases and workloads while providing cost estimates which are more accurate than those of workload-driven models. As part of this evaluation, we also provide a new benchmark (beyond JOB), which is necessary to evaluate cost estimation models more broadly on a variety of (real-world) databases. We will make this benchmark including query executions for training cost models publicly available and hope that it will benefit future research in learned cost estimation and potentially beyond.

Outline. In Section 2, we give an overview of our approach and describe the model architecture in more detail in Section 3. We then derive formal methods to recognize when sufficient training data is available for the model to generalize in Section 4. Before discussing the evaluation in Section 6, we describe the design decisions for our proposed benchmark to evaluate cost models. Finally, we present related work (Section 7) and conclude in Section 8.

2 OVERVIEW

In this section, we introduce the problem of zero-shot cost estimation and then present an overview of our approach.

2.1 Problem Statement

The overall goal of zero-shot cost estimation is to predict query latencies (i.e., runtimes) on an unseen database without having observed any query on this unseen database. Throughout this paper we use the term database to refer to a particular dataset (i.e., a set of tables with a given data distribution). Note that the problem of zero-shot cost estimation is thus in sharp contrast to the problem addressed by state-of-the-art workload-driven cost models, which train a model per database. Finally, while we believe that zero-shot learning for DBMSs is more generally applicable, we restrict ourselves in this paper to cost estimations for relational DBMSs. In particular, zero-shot cost models for other types of systems such as graph-databases or streaming systems are interesting avenues of future work.

2.2 Our Approach

A key challenge for developing zero-shot cost models is the question how to design a model that allows to generalize across databases. Here, we draw inspiration from the way classical cost models in DBMSs are designed. Typically, these consist of two models: a database-agnostic model to estimate the runtime cost and a database-dependent model (e.g., histograms) to capture data characteristics. When predicting the cost of a query, the estimated cardinalities and other characteristics (i.e., outputs of the database-dependent models) serve as input to the general database-agnostic cost model, which captures the general system behavior (e.g., the costs of a sequential scan grows linearly w.r.t. the number of rows). While the classical models are lightweight, they often largely under- or overestimates the true costs of a query since models are too simple to capture complex interactions in the query plan and data.

Hence, in our approach, we also separate concerns but use a much richer learned model, which similarly takes data characteristics of the unseen database as input to predict query runtimes in a database-agnostic manner. As depicted in Figure 2 (upper part), for training such a zero-shot cost model we provide different query plans along with the runtime as well as the data characteristics of the plan (such as tuple width as well as intermediate cardinalities) to the zero-shot cost model. Once trained, the model can be used on unseen databases to predict the query runtime as shown in Figure 2 (lower part).

As mentioned before, to predict the runtime of a query plan on a new (unseen) database, we feed the query plan together with its data

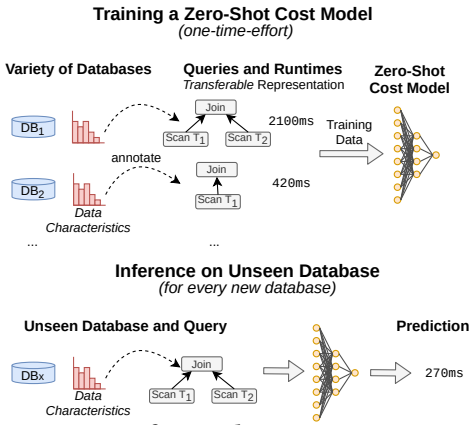


Figure 2: Overview of Zero-Shot Cost Estimation. The zero-shot cost model is trained *once* using a variety of queries and databases. At inference time, the model can then provide cost estimates for an unseen database and queries without requiring additional training queries. Enabling zero-shot cost estimation is based on two key ideas: (1) a new *transferable* query representation and model architecture is used to enable cost predictions on unseen databases and (2) we separate concerns, i.e., a zero-shot model learns a general database-agnostic cost model, which takes database-specific characteristics as input.

characteristics into a zero-shot model. While data characteristics such as the tuple width can be derived from the database catalogs, other characteristics such as intermediate cardinalities require more complex techniques. To derive intermediate cardinalities of a query plan in our approach we thus make use of data-driven learning [10, 34] that can provide exact estimates on a given database. Note that this does not contradict our main promise of zero-shot learning since data-driven models to capture data characteristics can be learned without queries as training data.

Another core challenge of enabling zero-shot cost models that can estimate the runtime of a plan given its data characteristics is how to represent query plans, which serve as input to the model. While along with workload-driven cost models, particular representation methods for query plans have already been proposed, those are not applicable for zero-shot learning. The reason is that the representations are not *transferable* across databases. For instance, literals in filter predicates are provided as input to the model (e.g., `2021` for the predicate `movie.production_year=2021`). However, the selectivity of literals will vary largely per database since the data distribution will likely be different (e.g., there might not even exist movies produced in 2021 in the test database).

Hence, as a second technique in this paper, we propose a new representation for queries that completely relies on features that can be derived from any database to allow the model to generalize to unseen databases. For example, predicates for filter operations in a query are encoded by the general predicate structure (e.g., which data types and comparison operators are used in a predicate) instead of encoding the literals. In addition, data characteristics of a filter operator (e.g., input and output cardinality to express the selectivity) are provided as additional input to a zero-shot model. That way, a zero-shot model can learn the runtime overhead of a filter operation based on database-agnostic characteristics. We present details of our query representation in Section 3.

Finally, a last important aspect of zero-shot cost models is that they can easily be extended to *few-shot learning*. Hence, instead of using the zero-shot model out-of-the box (which already can provide good performance), one can fine-tune the model with only a few training queries on an unseen database.

2.3 Assumptions and Limitations

While we expect zero-shot cost models to support a variety of different databases and workloads out-of-the-box, we next discuss the assumptions for a successful generalization.

In this paper, the main assumption is that we only focus on the transfer of learned cost models across databases for a *single database system* on a *fixed hardware*. We think that this is already challenging and allows for many interesting use cases. For instance, with zero-shot cost models cloud DBMSs (such as Redshift or Snowflake) can use learned cost models for new customer databases and workloads with significantly lower training overhead compared to the existing workload-driven models that require that a model is trained per new database. While we believe that zero-shot cost models can be extended to support also the transfer of cost models between different hardware setups and DBMSs by adding additional transferable features, we leave this to future work and assume a fixed hardware and DBMS in this paper.

Furthermore, while zero-shot cost models can generalize to unseen query patterns as we show in our experiments, it is clearly required that the training queries have a certain *coverage*, i.e., come with a diverse set of workloads and databases. For instance, it is a minimum requirement that every physical operator is observed in the training data s.t. the model can internalize the overall characteristics. Moreover, if there are extreme differences between training and test workloads, we expect the zero-shot model accuracy to degrade. We discuss how to detect and mitigate such cases by fine-tuning a zero-shot model in Section 4.

3 ZERO-SHOT COST MODELS

As mentioned in Section 2, a zero-shot cost model (once trained) is able to predict the runtime of a query on an entirely new database without retraining. A core building block needed to enable a zero-shot model is a new representation of queries that can generalize across databases. In the following, we thus first explain how we devised such a transferable query representation and then discuss how inference and training of a zero-shot model that uses this representation works.

3.1 Query Representation

State-of-the-art workload-driven models [13, 30] for cost estimation do not use a transferable query representations and can thus only be used on the database they were trained on. To better understand why current query representations are not transferable, we first explain how they typically encode queries.

3.1.1 Query Representation for Workload-Driven Models. At the core, query representations used for workload-driven approaches hard-code the model against a single database. For example, column names (e.g., those used in filter predicates) are typically encoded

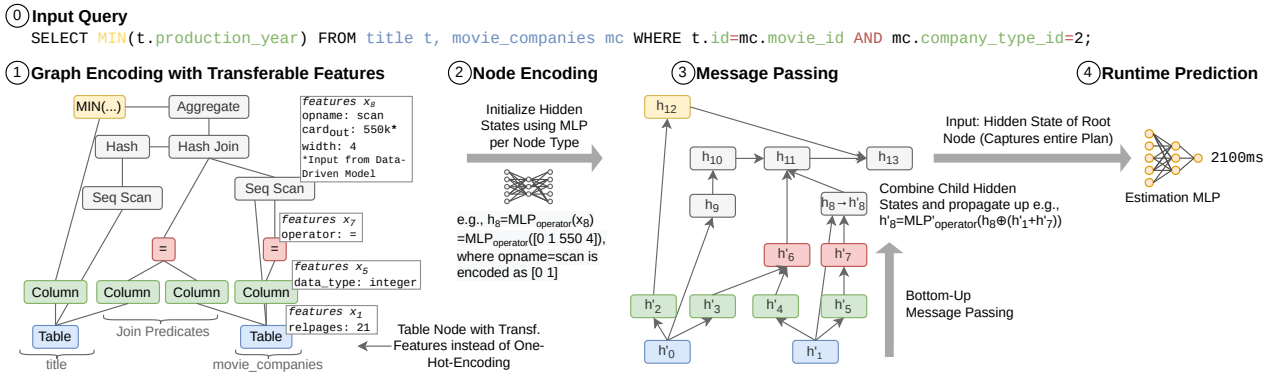


Figure 3: Using Zero-Shot Models for Cost Estimation (i.e., for Inference) on an unseen Database. (1) A query is represented as a graph with different node types (to represent plan operators, predicates, tables, columns etc.) and nodes are annotated with *transferable* features, which generalize across databases. (2) Afterwards, the resulting feature vectors of the nodes are fed into node-type-specific Multi-Layer-Perceptrons (MLPs) to obtain a hidden state, which is (3) propagated through the query-tree using bottom-up message passing to account for interactions among connected nodes. (4) Finally, the hidden state of the root node (encoding the entire graph) is fed into a final model — the estimation MLP — which predicts the query runtime.

using a one-hot encoding assigning each column present in the database a specific position in a feature vector. For instance, the column `production_year` of the IMDB dataset might be encoded using the vector $(0, 1, 0)$ (assuming that there are only three columns in total). If the same model should now be used to predict query costs for the SSB dataset, some columns might not even exist or even worse they might exist but have very different data distributions or even a different data type. In fact, non-transferable feature encodings are not only used for columns but in various places of the query representation such as encoding table names or literals in filter predicates.

3.1.2 Query Representation for Zero-Shot Cost Models. Hence, for zero-shot cost models we require a new query representation that is transferable across databases. The main idea of the transferable representation we suggest in this paper is shown in Figure 3. At the core, a query plan and the involved tables and columns are represented using a graph where graph nodes use transferable features ① (i.e., features that provide meaningful information to predict runtime on different databases). This representation then serves as input for the training and inference process of zero-shot cost models ②-④ that we explain in the subsequent sections. In the following, we discuss the graph encoding of the transferable featurization in detail.

Graph Encoding of Query Plans. While graph-based representations have been already used to represent query operators of a query plan [30], our representation has significant differences. First, as shown in Figure 3 ①, our representation not only encodes physical plan operators as nodes (gray) in the graph as in previous work [30], but it also covers all query plan information more holistically using different nodes types for input columns (green), tables (blue) as well as predicate information (red). Second, as discussed before, previous approaches also covered such information, however, they used one-hot-encodings (which are non-transferable) while our representation captures the query complexity in a transferable way.

For instance, to encode filter predicates, different from previous approaches we encode the predicate structure as nodes (red) without literals. In particular, we encode information such as data types of the columns and operators used for comparisons. For example, the filter predicate `company_type_id=2` for the query ⓪ in Figure 3, is encoded using a *column* node (x_5) with the comparison node = (x_7). As such, a zero-shot cost model provided with the transferable features (e.g., intermediate cardinalities, which are given by the data-driven models) can infer the complexity of the predicates to estimate the query runtime.

Transferable Featurization. While our graph representation allows to flexibly encode query plans across databases, we similarly have to make sure that the features used to represent nodes in the graph ① (e.g., plan operators as shown in gray) are transferable. In particular, when used on different databases, features should not encode any implicit information that hinder the transfer of the model to a new unseen database.

The concrete set of such features used for the different node types in our graph representation is depicted in Table 1 specifically for zero-shot cost models on Postgres, which we use as DBMS in all our experiments. For instance, input column nodes (green) use features such as the data type or the width in bytes. Similarly, for tables (blue nodes), we use other transferable features (e.g., the number of rows as well as the number of pages on disk). However, note that the general class of features allows cost predictions also for other single-node DBMSs such as MySQL. The rationale for selecting these features is to include transferable features that cover very different aspects aspects regarding the query (e.g., involved operators, column types in predicates) as well as the data (e.g., queried tables, data distribution). As we will show in an ablation study in the evaluation, each such aspect with the corresponding individual features improves the cost estimation accuracy.

Importantly, transferable features can either characterize the query plan (e.g., operator types) or represent the data characteristics (e.g., intermediate cardinalities) and together allow a zero-shot cost model to generalize to an unseen database. For transferable features that represent data characteristics many can be derived from the metadata of a database (such as the the number of rows of a table node). However, some other features that represent data characteristics — e.g., the estimated output cardinality of an operator

Table 1: Zero-Shot Features. All features are transferable and have the same semantics for different databases.

Node Type/Category	Feature	Description
Operators/ Data Distribution	`workers`	Number of parallel workers
	`opname`	Name of physical operator
	$\texttt{card}_{\texttt{out}}$	Estimated output cardinality of operator
	`width`	Tuple width
	`card_prod`	Estimated product of children output cardinalities
Predicate	`operator`	Operator type (e.g., =)
	`literal_feat`	Feature capturing literal complexity, e.g., number of values for `IN` operator or regex complexity
Table	`relpages`	Number of pages
	`reltuples`	Number of rows
Input Column	`width`	Avg. number of bytes to represent a value
	`correlation`	Attribute correlation with row number
	`data_type`	Data type of column
	`ndistinct`	Number of distinct attribute values
	`null_frac`	Fraction of `NULL` values
Output Column	`aggregation`	Which aggregation type is used

node — require more involved techniques. In Section 3.4, we discuss alternatives of how we provide estimated output cardinalities to zero-shot cost models.

3.2 Inference on Zero-Shot Models

Once a query graph with the transferable features on an unseen database is constructed for a query plan, it can be used as input for a (trained) zero-shot cost model to predict the runtime. Predicting the runtime of a new query plan with a zero-shot cost model is executed in three steps, which we depict as pseudocode in Algorithm 1: First, we compute a hidden state for every node of the query graph ② given the node-wise input features. Second, the information of different graph nodes is combined using message passing ③ before a Multi-Layer-Perceptron (MLP) predicts the runtime of the query plan ④. The same steps are also reflected in Figure 3 ② to ④.

In particular, in step ②, the feature vectors x_v of each graph node v are encoded using a node-type specific MLP, i.e., nodes of the same type (e.g., all plan operators) use the same MLP to initialize their hidden state h_v (line 5). For instance, in Figure 3, the hidden state h_8 of the node representing the sequential scan on the `movie_companies` table is obtained by feeding the feature vector x_8 (containing transferable features) into an MLP, which is shared among all plan operators (gray nodes).

Afterwards, in step ③, a message passing scheme is applied, which is prominently used in graph neural networks (GNNs) [6] to model the interactions between nodes in graphs (i.e., to capture interactions of query operators in the plan such as effects of a pipelined query execution). Different from message passing schemes for general graph encodings, for the message passing in zero-shot models we can exploit the fact that queries can be represented as directed acyclic graphs (DAGs) since query-plans are tree-structured. We thus use a novel bottom-up message passing scheme through the graph (i.e., in topological ordering) to obtain an updated hidden state h'_v of a node v that contains all information of the child nodes. During this pass, the updated hidden states h'_u of the children u are combined by summation [35] and concatenated with the initial hidden state h_v of a node and fed into a node-type-specific MLP (line 7). For instance, in Figure 3, the updated hidden state h'_8 of the scan node is obtained by summing up the updated hidden states of the child nodes (representing the table and predicate operator of the scan) concatenated with the initial hidden state

Algorithm 1 Inference on Zero-Shot Models

1: **Input:** Query graph encoding with nodes v and input features x_v
2: **Output:** Cost estimate $\hat{c}$
3:
4: **for** $v \in$ graph encoding **do** ▷ Compute hidden state per node ②
5: $\quad h_v \leftarrow MLP_{T(v)}(x_v)$
6: **for** $v \in$ in topological ordering **do** ▷ Bottom-up pass in graph ③
7: $\quad h'_v \leftarrow MLP'_{T(v)}\left(\sum_{u \in children(v)} h'_u \oplus h_v\right)$
8: $\hat{c} \leftarrow MLP_{est}(h'_r)$ ▷ Estimate costs using root node state ④
9: **return** $\hat{c}$

(capturing properties of the scan), which is then fed into an MLP, which is again shared among all plan operators.

Finally, as a result of step ③ the updated hidden state h'_r of the root node r of a query plan captures the properties of the entire query. For the cost prediction in step ④, we thus feed this hidden state into a final estimation MLP to obtain the cost estimate $\hat{c} = MLP_{est}(h_r)$ (line 8). Hence, in Figure 3 ④, the updated hidden state h'_{13} is fed into the final estimation MLP to obtain the cost estimate since it captures information of the entire plan.

3.3 Training Zero-Shot Models

As mentioned before, a zero-shot cost model is trained on several databases and queries to learn the runtime complexity of query plans given the input features. To be more precise, a zero-shot cost model is trained in a supervised fashion using pairs (P, c) that consists of a plan P with the respective features and the actual runtime cost c. Importantly, all steps described in the inference procedure (node encoding, message passing and finally runtime estimation) are differentiable, which allows us to train the model parameters of the MLPs used for all zero-shot model components jointly in an end-to-end fashion. As loss function to compare the actual costs c of a featurized query plan P with the estimated costs $\hat{c}$, we use the Q-error loss $\max(\frac{c}{\hat{c}}, \frac{\hat{c}}{c})$ [13, 30] since this worked best for zero-shot models compared to other alternatives.

3.4 Deriving Data Characteristics

As discussed before, an important aspect of a zero-shot model is that the model is not tied to a particular data distribution of a single database. For enabling this, we provide data characteristics such as column widths in bytes, number of pages and tuples of tables but also output cardinalities of operators as input to those models. To be more precise, given a particular query plan for which the runtime should be estimated, those features have to be annotated for each graph node in the query encoding.

While the majority of those features can simply be derived from the database catalog, intermediate cardinalities in a query plan are notoriously hard to predict and simple statistics are known to be often imprecise [17]. Hence, learned approaches to tackle cardinality estimation have been proposed to derive accurate intermediate cardinalities. While in principle such learned approaches can be used to predict intermediate cardinalities, which are then used as input for the zero-shot models, there are important trade-offs when choosing which techniques are suitable for zero-shot learning. In the following, we discuss these aspects.

First, a zero-shot cost model should be able to predict query runtimes on databases that were not seen before without relying on an observed workload on that database. Since workload-driven models for cardinality estimation require such queries as training data, they are not suited for our purpose of predicting cardinalities for zero-shot models. Second, traditional histogram-based approaches have the advantage that no additional efforts are required since the query optimizers anyway have built-in techniques. However, they are often imprecise. Third, data-driven models are more precise but also need to be trained. However, the training does not rely on query executions and is thus usually just in the order of minutes. Unfortunately, state-of-the-art data-driven cardinality estimators do not yet support the same variety of different queries as traditional approaches.

Hence, we have two options to supply zero-shot cost models with intermediate cardinalities. First, we can train a data-driven model, which results in more accurate cardinality as input to a zero-shot model and thus also cost estimates. However, if the effort of training a data-driven model for a new database is not acceptable or the workload is not supported by data-driven learning, we can fall back to the cardinality estimates of the query optimizer. In our evaluation, we will demonstrate that zero-shot models can still produce reasonable estimates even if only cardinalities estimates from traditional models are available.

4 ROBUSTNESS OF ZERO-SHOT MODELS

An important question for zero-shot models is at which point a sufficient amount of different training databases (and workloads) was observed to generalize robustly to unseen databases. As discussed in Section 2.3, a minimum requirement is to have sufficient coverage of the training data for the expected queries in the evaluation workload. For instance, all operators should be observed at least once and the number of joins, group by attributes as well as databases used for pre-training etc. should be representative as well. However, it is still interesting to provide an estimate of how precise the zero-shot cost models will be for unseen databases and what can be done in cases of more severe workload drifts which we will discuss below.

4.1 Estimating the Generalization Performance

We first formalize the problem, before we derive a method to estimate the generalization error. For training a zero-shot model, we have observed n databases and workloads. In particular, for each of the databases D_i we have access to training data T_i in the form of query plans and their runtimes $T_i = \{(P_1, c_1), (P_2, c_2), \dots (P_m, c_m)\}$. We are now interested in how accurately the zero-shot cost model Z will predict the runtimes for plans T^* on some unseen database D^*. In particular, if the expected error is acceptable, we have observed a sufficient amount of databases and workloads. More formally, we will define some error metric $E(T_i)$ with which we can compare the true runtimes and model predictions for some database D_i. An example for such a metric could be the prominently used median Q-error. We are now interested in estimating this error metric for an unseen database $E(T^*)$, i.e., the expected generalization error.

We now make use of statistical techniques to estimate the generalization error. For instance, in ML it is standard practice to train the model on a subset of the data and then use the remaining samples to estimate the error for future unseen datasets. Analogously, we can train the zero-shot model on a subset of the training databases $T_1, T_2, \dots, T_i$ (i.e., for a subset of databases) and evaluate the trained model on the remaining databases $T_{i+1}, \dots, T_n$. Similar to cross validation, we can repeat this procedure with different splits and average the test errors to estimate the generalization error $E(T^*)$, i.e., how accurate the model is expected to be on an unseen database. Interestingly, this is an unbiased estimator of the test error $E(T^*)$ under the independent identically distributed (i.i.d.) assumption, which we will discuss shortly. Hence, using only the observed databases and queries, we can estimate how accurate the model predictions for unseen databases will be.

In order to now evaluate whether the model has observed a sufficient amount of databases and workloads, we can use two techniques. First, we can simply estimate the generalization error as described above and stop the training if it is sufficient. However, in this case we have to decide which generalization error is acceptable. A second technique (which we actually use in this paper) is to estimate if additional training databases will improve the generalization performance. For this, we train the model on subsets of all training databases. If the estimated generalization error $E(T^*)$ does not improve significantly for a larger number of training databases, we can conclude that additional databases will not improve the generalization capabilities of the zero-shot cost model and thus stop the training data collection.

4.2 Tackling Workload and Data Drifts

The performance of zero-shot cost models will deteriorate if the new database and workload is significantly different from the training data. While data drifts can be handled by providing up-to-date cardinality estimates (from simple or data-driven models) as we show in our experiments, workload drifts need a more careful handling. For instance, if there are significantly larger joins in the unseen database than for the training databases, the zero-shot model might not be able to predict the runtime with the same high accuracy. As we will show in our experimental evaluation, however, zero-shot cost models can often still generalize robustly in practice and can provide more accurate estimates than other baselines in case of workload drifts. In addition, we suggest a strategy to detect cases of workload drifts by monitoring the test error and propose to tackle workload-drifts using few-shot learning.

Note that in cases of workload drifts the i.i.d. assumption does not hold and the Q-error on the unseen database is larger than implied by the generalization error. More technically, the i.i.d. assumption is a common assumption in ML that requires that the training datasets and test datasets are independent samples of some distribution $\mathcal{D}$. Due to a workload drift, the samples are no longer independent and thus the generalization error $E(T^*)$ might be increased. A simple yet effective strategy to recognize those cases is thus to monitor the error for unseen databases during inference. In cases where the error exceeds a certain threshold, one could decide to fine-tune the zero-shot model using the additional observed queries as training data (resulting in few-shot models). We will demonstrate in the experimental evaluation that zero-shot cost models fine-tuned on a small number of additional queries can significantly improve the accuracy on the unseen database in such cases.

5 A NEW BENCHMARK

In order to properly train and evaluate cost models, we require both a diverse set of databases and executed workloads on these databases. Since currently there is no suitable benchmark with such properties, we created a new benchmark (that includes existing benchmarks such as JOB), which we discuss in this section. Furthermore, we will make this benchmark publicly available to foster future research in this area.

5.1 Design Decisions

For many years, DBMS systems were evaluated using synthetic benchmarks such as TPC-H [1], TPC-DS [26] or SSB [27]. While such benchmarks allow to evaluate the general system performance and scalability, they are in isolation insufficient to evaluate cost prediction models since the predicted cardinalities of the query optimizer are significantly more accurate than in practice. The reason is that the data is synthetic and thus no interesting correlations have to be captured making cardinality estimation challenging in practice. Hence, Leis et al. [17] suggested the JOB-workload on the IMDB dataset that comes with challenging correlations and has become the standard method (along with the simplified JOB-light workload [13]) to evaluate learned cost and cardinality models.

While the IMDB benchmark is useful to evaluate workload-driven cost estimators that need to work on a single database only, it cannot be used for the evaluation of zero-shot cost models since these have to be trained on a variety of different databases. Moreover, even for workload-driven cost estimators a benchmark that spans a more diverse set of databases would definitively be helpful to evaluate the prediction quality. Hence, we decided to create a new benchmark that covers established datasets such as IMDB but also additional datasets that have other characteristics.

5.2 Datasets

As discussed before, it is insufficient to just add synthetic datasets since correlations hardly resemble data distributions found in the real-world. We thus decided to leverage *publicly available real-world datasets* [25] together with the *datasets used in established benchmarks such as JOB*. Since certain databases were very small in size, we additionally scaled them to larger sizes to be interesting for cost estimation (s.t. a sample of queries takes a predefined threshold of time). In addition to the datasets mentioned before, we also include data and workloads of existing benchmarks such as SSB and TPC-H. As these benchmarks rely on synthetic data, this further increases the variety of data characteristics our benchmark covers for testing learned cardinality estimators. Overall, the benchmark comprises of 20 databases that vary largely in the number of tables, columns and foreign-key relationships.

5.3 Workloads and Traces

Furthermore, for benchmarking learned cost models, workloads are required for training and testing. To simplify the comparison with prior work we first include predefined benchmark queries for databases that come with such workloads (e.g., JOB for IMDB). However, since for the majority of the databases mentioned before no workloads are available, we implemented a workload generator that generates different types of queries. For creating the workload, the generator supports three modes: a *standard* mode where Select-Project-Aggregate-Join (SPAJ) queries with conjunctive predicates on numeric and categorical columns similar to the ones used by Kipf et al. [13] are generated, a more *complex* mode, which includes predicates involving disjunctions, string comparisons with regex predicates, `IS (NOT) NULL` comparisons and `IN` operators (resembling the complexity of the JOB-workload) and finally an *index* workload where random indexes (both foreign key and for predicate columns) are created throughout the execution of the standard workload, which is challenging due to the varying physical designs. Since the benchmark will be publicly available it can be easily extended in the future.

In addition to the datasets and the workload generator, the benchmark comes with workload traces (e.g., executions of the queries and their runtime) for all 20 databases that can be used directly by other researchers as training / testing data (which we also used in our evaluation). To be more precise, we generated 15,000 queries per database and executed those queries on a Postgres DBMS (v12) on `c8220` nodes on the cloudlab platform. Overall, this also allows for a better reproducibility since this platform can be used by other researchers as well. To limit the already excessive resource consumption required to produce this trace, we excluded queries running longer than 30 seconds from the benchmark for all workloads. In total, the execution of these more than 300k queries takes 10 days if executed on a single node. As part of the traces, we not only provide the runtime of the query but also the physical plan used to run the query along with actual cardinalities.

6 EXPERIMENTAL EVALUATION

In this Section, we evaluate zero-shot cost estimation with a set of different experiments:

- **Exp 1. Zero-Shot Accuracy on Unseen Databases.** We evaluate how accurately zero-shot cost models can predict costs for unseen databases.
- **Exp 2. Zero-Shot vs. Workload-Driven.** In addition, we compare the training overhead and accuracy with state-of-the-art workload driven approaches.
- **Exp 3. Generalization.** In this experiment, we study how our models generalize under workload drifts (i.e., under database updates and larger unseen joins).
- **Exp 4. Training and Inference Performance.** Furthermore, we evaluate the training and inference performance of zero-shot cost models and compare training efforts to workload-driven models.
- **Exp 5. Ablation Studies.** Finally, we show the effects of different design alternatives of zero-shot models as well as a study where we determine how many database are sufficient for zero-shot cost models to generalize.

For all experiments, we use the traces of the benchmark discussed before (for training and testing).

6.1 Exp 1: Zero-Shot Accuracy on Unseen Databases

First, in order to evaluate the accuracy of zero-shot cost models, we trained a zero-shot model using workloads on 19 out of the 20 datasets of the benchmark as training data and evaluated the model on the workload of the unseen (remaining) database. In particular,

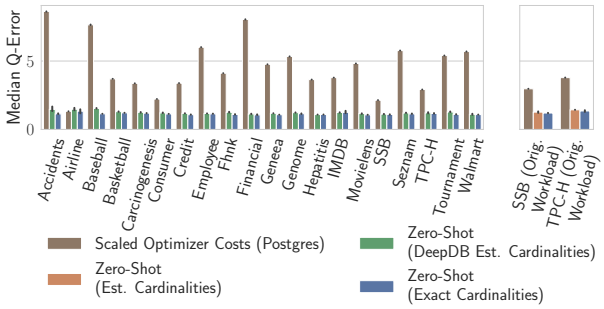


Figure 4: Zero-Shot Generalization across Databases. The zero-shot models are trained using workloads on 19/20 databases and tested on the remaining unseen database. Overall, zero-shot models are significantly more accurate than the scaled estimates of the optimizer cost model. In addition to using workloads as defined by our benchmark (left), we repeated this experiment with standard benchmark workloads (SSB and TPC-H on the right) to further show the generalization potentials of zero-shot cost models.

we use the trained model to predict the runtimes of the queries on the unseen database and report the median Q-error. In the first experiment, we focus on the *standard* workloads and defer the results of the *complex* and *index* workloads of our benchmark to follow-up experiments. For this experiment, we ran each setup three times using different seeds for the cost estimation for every unseen database.

For showing the performance of zero-shot cost models on unseen databases, we used two variants of providing intermediate cardinalities - we either used predictions of learned cardinality estimators or the actual cardinalities, which are not available in practice prior to execution but serve as an interesting upper baseline for zero-shot learning (i.e., how accurate the predictions become with perfect cardinality estimates). For the data-driven cardinality estimator, we trained DeepDB [10] models, which worked best in preliminary experiments. To the best of our knowledge, we are the first to propose zero-shot cost estimation and thus no other learned approaches are included as a direct baseline in this first experiment where we aim to analyze the accuracy on unseen databases. For instance, workload-driven approaches would need query executions on the unseen database, which we do not provide in the zero-shot setting. However, we compare our approach with workload-driven models in Section 6.2.

As a sanity check that zero-shot models provide better performance than classical cost estimation models that rely on simple (non-learned) techniques (and as such could also count as zero-shot cost models), we use cost estimates coming from the Postgres query optimizer as a baseline similar to previous work [30]. Moreover, for the distributed setup we later on also employ the cost estimates of a commercial cloud DBMS. Since Postgres cost estimates are provided as abstract cost units, we use a simple linear model on top of Postgres estimates (and hence the results are called *Scaled Optimizer*), which provides actual query runtimes. Different from [30], which directly take the cost units as runtime (in ms), using a linear model on top results in a much lower Q-error for Postgres. For training the simple linear model we are using the same training data from the other 19 databases as for zero-shot models to be fair.

The results can be seen in Figure 4. In general, the zero-shot models offer robust performances for all of the databases despite the varying complexity. In fact, all median Q-errors are below 1.54 for the version using DeepDB cardinality estimates (vs. 8.62 in the worst case for the *Scaled Optimizer* cost). Finally, we can see that zero-shot cost models using DeepDB cardinalities are almost matching the performance with perfect cardinalities. This suggests that the models can cope with partially inaccurate cardinalities. Indeed, as we will see in a follow-up experiment, this even holds when we use potentially inaccurate cardinality estimates coming from a classical optimizer instead.

Overall, we can see that the zero-shot cost models are significantly more accurate than the scaled optimizer estimates outperforming these on 18 out of 19 datasets and being on par for the last remaining dataset (Airline). The reason is that zero-shot cost models capture subtleties in operator performance and interactions of operators in the plan more accurately than simplistic cost models. The results are just on par for the remaining database since the optimizer costs are relatively accurate because it is merely a star schema, i.e., a relatively simple schema structure.

To demonstrate that zero-shot cost models also improve the estimates for workloads of traditional benchmarks, we repeat the previous experiment with the original benchmark queries of SSB and TPC-H.[2] Again we train on 19 out of 20 datasets (excluding either SSB or TPC-H) and show the median Q-errors of both the baseline and our approaches. Note that DeepDB does not support all operators in TPC-H and thus we use Postgres cardinality estimates (orange bar in Figure 4) instead. As we can see, the results are very similar to the results using our new benchmark for zero-shot cost estimation providing additional evidence that zero-shot cost estimation can improve the cost estimation accuracy on queries from standard benchmarks as well.

6.2 Exp 2: Zero-Shot vs. Workload-Driven

In the following, we contrast the performance of zero-shot cost models with workload-driven approaches.

Training Overhead. An interesting question is how many training queries are required for workload-driven learning on an unseen database to match the performance of zero-shot learning, which we will study next. In particular, in this experiment we evaluate the Q-errors for the scale, synthetic, and JOB-light workloads (IMDB). As before zero-shot models are not trained on IMDB at all (but on the other 19 databases) while workload-driven models are trained on a varying number of training queries on IMDB.

For the workload-driven approaches we use the E2E model proposed by Sun and Li [30] as well as the MSCN model by Kipf et al. [13]. The idea of the E2E models is to featurize the physical query plans and feed them into a neural model to predict the runtime. However, in contrast to zero-shot cost models the query plan representation is not transferable and thus the train and test databases have to be identical. The MSCN model, which was initially developed for cardinality estimation uses a more high level representation and encodes the sets of joins, predicates and tables

[2]For SSB, we used all queries as-is. For TPC-H, since our current implementation does not support the `subplan` operator of Postgres, we rewrote subqueries using joins. However, we believe that our approach can also be extended to support subqueries.

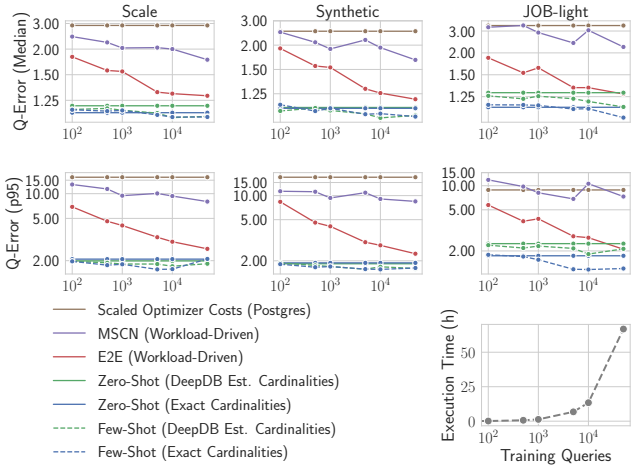


Figure 5: Estimation Errors of Workload-Driven Models for a varying Number of Training Queries compared with Zero-Shot Cost Models. Even the most accurate workload-driven model (E2E) requires approximately 50k query executions on an unseen database for a comparable performance with zero-shot models, which is roughly equivalent to 66 hours of executed workload. Since zero-shot models do not require any additional queries it is significantly cheaper to deploy them for a new database. However, zero-shot models can be fine-tuned to obtain few-shot models, which further improve the accuracy.

of a query, which are then fed into a neural architecture, which is thus oblivious of the physical plans used. Both models are trained on a varying number of training queries, which are generated for the IMDB dataset similar to the original training setup used by Sun and Li [30]. Furthermore, as a last baseline, we again employ the scaled costs of the Postgres query optimizer.

In Figure 5, we depict the median Q-error of comparing our zero-shot performance to the baselines as discussed before for the IMDB benchmark workloads for a varying number of training queries. As we can see the zero-shot cost models can estimate the runtimes accurately even though queries on the IMDB dataset were not observed in the training data. In particular, E2E requires about 50k training queries on the IMDB database to be on-par with zero-shot cost models. As we can see in the lower right plot in Figure 5 this amount of queries takes approximately 66 hours to run, which is a significant effort given that it has to be repeated for every new database. Another interesting comparison is to use the training queries also to fine-tune the zero-shot models on the IMDB database; i.e., we use zero-shot models in the few-shot mode discussed in the paper. As we can see, few-shot cost models that are fine-tuned on the IMDB database can further improve the cost estimation accuracy of zero-shot models. It is thus beneficial to also leverage fine-tuning in case training queries for the unseen database are available.

Finally, we can see that the MSCN models are not equally accurate, which is likely due to the fact that they do not consider the physical plan that was run to execute a given query. Still, all learned approaches are more accurate than the scaled optimizer in the median after only a few queries. Furthermore, we can observe that zero-shot and few-shot cost models not only outperform

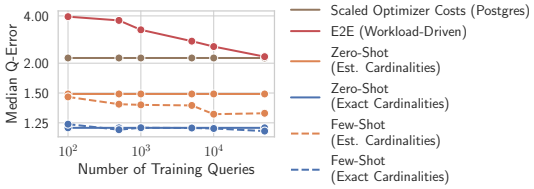


Figure 6: JOB-Full Workload. Zero-shot models are significantly more accurate than the workload-driven model (E2E) and the scaled optimizer estimates even for the complex JOB benchmark. Again few-shot learning can further improve the performance of zero-shot models.

workload-driven models in the median but also in the tail performance, i.e., on the 95th percentile Q-error. We can observe similar effects for the maximum Q-error.

Complex Queries. In this experiment, we next focus on the performance for complex queries. For this, we again train on 19 datasets and test on the IMDB database (this time using the *complex* benchmark queries) using the JOB-Full benchmark, which (different from the other workloads on IMDB) contains also queries with a higher number of joins and more complex predicates including pattern-matching queries on strings. Note that data-driven models do not support complex predicates and we thus resort to the cardinality estimates of the query optimizer (Postgres) to inform the zero-shot model. As baselines, we again compare to the scaled optimizer costs and E2E, which in contrast to MSCN supports complex predicates. To be fair, we use training queries with complex predicates on IMDB for the workload-driven models. In addition, we also report the accuracy of zero-shot models fine-tuned on the IMDB database using the few-shot learning.

As we can see in Figure 6, again zero-shot models outperform the other approaches. In particular, even the version using just optimizer cardinality estimates is more accurate than E2E using 50k queries, which emphasizes that zero-shot cost models are robust w.r.t. imprecise cardinality estimates. The E2E models in this case need 50k queries just to match the performance of the scaled optimizer costs, which is inferior to the previous experiment with a lower query complexity. The reason is that the E2E model has to learn the data distribution of strings as well and support complex predicates including wildcards while only observing queries. We hope that in the future, data-driven models support string predicates and disjunctions as well to be used in conjunction with zero-shot cost models also for complex queries. Similar to the previous experiment, few-shot learning can further improve the accuracy.

6.3 Exp 3: Generalization

In this experiment, we investigate how robustly zero-shot cost models react to changes in the data characteristics and workload.

Generalization to Updates. For the first aspect, we analyze the effects of updates on the accuracy of cost estimation. For this, we only train on a fraction of the full data and then update the database (without retraining the prediction models). After the update of the database, we then predicted the query runtimes using zero-shot cost models as well as the other baselines (workload-driven models and the scaled optimizer). Note, that workload-driven models are expected to result in inferior performance for a higher fraction of updates since they cannot capture database updates without

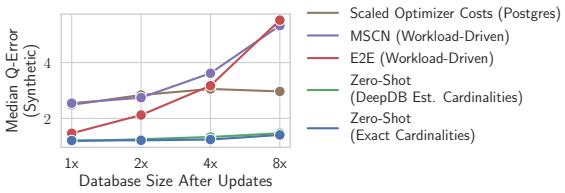


Figure 7: Zero-Shot Models are robust w.r.t. Updates. Without any retraining we do not see regressions in cost estimation accuracy even for massive update rates (up to 8 times the size of the original database using rescaling). In contrast, workload-driven models require additional training queries.

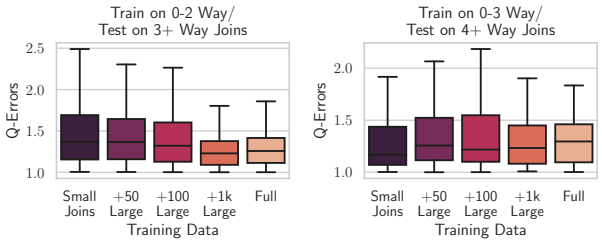
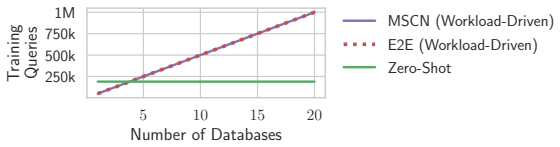


Figure 8: Zero-shot cost models generalize robustly to larger Joins. Compared to zero-shot models trained also on larger joins (Full), the zero-shot models trained only on smaller joins (Small Joins) have only minor regressions in accuracy. In addition, fine-tuning the zero-shot cost models on a low number of additional queries with larger joins (resulting in few-shot models) further improves the performance.

collecting new training data. This is very different from zero-shot models that get informed by data-driven models that can thus adjust to data updates without the need to retrain. In particular, the data-driven models from DeepDB [10] as well as classical statistics such as histograms that are compatible with zero-shot cost models are directly updateable with low overhead and hence can provide also accurate estimates after the update.

We depicted the results in Figure 7. As we can see, there is almost no performance degradation for the zero-shot cost models with a higher update fraction. Note that we did not retrain the zero-shot cost models at all to achieve the performance but simply relied on the ability to generalize to different data characteristics. In contrast, for workload-driven models we observe a performance degradation since those models would require additional training queries on the updated database to be adapted. The reason is that the models also internalize the data distribution (i.e., table sizes and correlations) implicitly during the training and can only be informed about changes by observing additional query runtimes. This is especially problematic for more update-heavy workloads were frequently additional training queries have to be run to update the models. Note that the scaled optimizer costs do not experience such a degradation but are again less accurate than zero-shot models.

Generalization to Workload Drifts. In this experiment, we investigate how zero-shot models react to workload drifts, in particular to larger joins that appear after training a cost prediction model. To this end, we trained the zero-shot models using only queries with up to 2 or 3-way joins on the 19 training datasets and evaluate the model using 3-way or 4-way joins (or larger) on the IMDB dataset, respectively. Since we suggest to address workload-drifts using few-shot learning, we also introduce variants that are fine-tuned on a small amount of large joins on the IMDB database. As we can see in Figure 8, the performance of the model with a training set constrained to small joins does not degrade heavily compared to the model that was also trained on larger joins on the remaining 19 datasets (Full) indicating a robust generalization to larger joins. In addition, few-shot models fine-tuned on a small amount of larger joins ($\approx$ 50) observed on the IMDB dataset is sufficient to achieve the same median Q-error. An even larger amount of retraining queries allows to outperform the original zero-shot model, which is consistent with previous experiments showing that few-shot learning further improves the accuracy.



(a) Required Training Queries.

(b) Train and Test Throughput.

Figure 9: Training and Inference and Performance. Even though zero-shot models generalize across databases they almost match the inference and training throughput of the most accurate workload-driven alternative (E2E) and quickly amortize in terms of required training queries.

6.4 Exp 4: Efficiency of Training and Inference

In this experiment, we evaluate the efficiency of training and inference of zero-shot models compared to workload-driven models.

Training Overhead. In a first experiment, we compare the number of training queries required for zero-shot models as well as for workload-driven models. Importantly, workload-driven models need to be trained on every single database while zero-shot models can (once trained) be applied to many different databases out-of-the-box. For showing this effect we analyze how many training queries would be required for supporting a varying number of unseen databases for which new cost estimates are required The results are shown in Figure 9a. As we can see since workload execution is a one-time effort for zero-shot models (since they generalize across databases) this quickly amortizes compared to workload-driven learning since for workload-driven models, we need to collect training data for every new database.

Training and Inference Throughput. In a second experiment, we compare the training and inference throughput of zero-shot cost models with state-of-the-art workload-driven approaches. In this experiment, we aim to show that zero-shot models are not imposing higher overhead for training and inference and thus can be used efficiently in real DBMSs. As we can see in Figure 9b, zero-shot models achieve a comparable throughput and thus do not impose higher overhead compared to workload-driven models. As we can see, the MSCN models achieve higher throughput compared to all other models (zero-shot and E2E). The reason is that these models

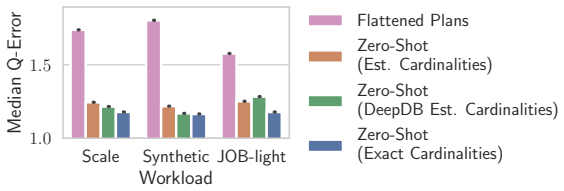


Figure 10: Ablation Study. Using a flattened representation of the plans instead of our graph-based encoding yields less accurate models. Zero-shot models using the cardinality estimates of the query optimizer are still reasonably accurate.

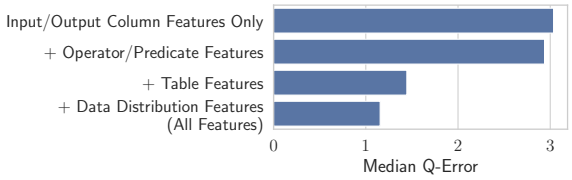


Figure 11: Feature Ablation Study. All groups of query graph features used in zero-shot cost models individually improve the accuracy. Table and data distribution features have the largest impact since they determine the scan cost and size of intermediate joins, respectively.

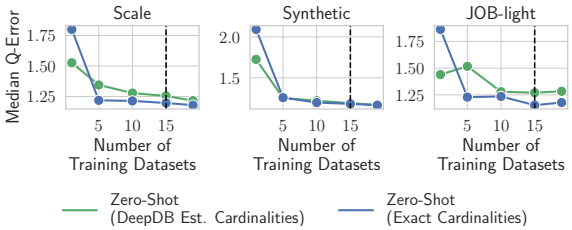


Figure 12: Zero-Shot Generalization by Number of Training Databases. If we use more than 15 training databases we start to see diminishing returns in accuracy suggesting that the variety of databases in the benchmark is sufficient.

featurize the physical query plan resulting in larger graphs compared to MSCN models, which only encode the joins, tables and predicates in a query. However, this comes at the cost of an inferior predictive performance as shown before.

6.5 Exp. 5: Ablation Study

In this experiment, we present the results of our ablation study showing the effects of the different design choices as well as the efficiency of our estimator to determine how many different databases are needed for training a zero-shot model.

Zero-Shot Design Space. We first explore the different design space options of zero-shot cost estimation. In particular, we focus on the questions how different cardinality estimation techniques impact the model accuracy and whether our new model architecture using graph encodings is actually required or a simpler architecture suffices.

To address the latter question, we implemented a different version of zero-shot cost estimation that represents a single query plan as a flat vector (instead of using a graph). In particular, the chosen representation is similar to Ganapathi et al. [5] that represents a query plan using a vector where each physical operator corresponds to two entries in the vector: one that counts how often the operator appears in the plan and one that sums up the cardinality estimates for that operator. For instance, if we only had sequential scans and nested loop joins in the query plans and one plan would scan two relations of 1M tuples each and join them resulting in 1M tuples, the vector representing the query plan could be $(2, 2M, 1, 1M)$. Given this representation, we train a state-of-the-art regression model [12] to predict the runtime given a vector. Similar to the zero-shot models, we train on the remaining 19 datasets and evaluate the performance on the IMDB benchmarks.

As we can see in Figure 10, the flattened version of zero-shot cost models is significantly less accurate than our proposed transferable graph-based representation. The reason is that the interactions of physical operators in the plan can only be modeled approximately if represented as a vector while our graph-based encoding allows the neural model to capture such interactions more accurately. Second, regarding cardinality estimates, we can see that data-driven cardinality estimates improve the accuracy of zero-shot cost models compared to models using optimizer cardinality estimates. However, the estimates are still very accurate even if cardinality estimates are annotated from simple cardinality estimation models that are used in DBMSs today. This is especially useful for query types that data-driven models do not support as of today and where the optimizer cardinality estimates hence serve as a fallback.

Query Graph Featurization. In addition, we also study the impact of different featurizations of the query graph representations used by zero-shot cost models. In particular, instead of training zero-shot cost models using all the features introduced in Table 1, we will gradually include more groups of features (e.g., all features related to scanned tables). We then report the median Q-error achievable with this set of features. As we can see in Figure 11, each group of features individually improves the performance of the models. The most significant improvement is due to features characterizing the tables as well as the data distribution (e.g., cardinalities) as these features influence the runtime overhead of a query most significantly. However, we can conclude that all features are worth incorporating in zero-shot models as long as they are transferable as described in Section 3. The rationale is to include as many aspects that could impact the query runtime as possible in the query representation.

Number of Training Databases. As described in Section 4.1, in order to assess whether a zero-shot cost model has seen a sufficient number of training databases and workloads, we estimate the expected generalization error for a varying number of training databases. The generalization error is estimated by computing the test error on an unseen holdout database. If the model performance plateaus for a certain number of training databases, we can conclude that the number of training databases is sufficient.

In this experiment, we show how the generalization error develops for a growing number of training databases (i.e., from just using one up to all 19 databases). For estimating the generalization error, we use the standard benchmark queries as defined on the IMDB dataset (i.e., we use the synthetic, scale and JOB-light [13] workloads). As we can observe in Figure 12, as expected the generalization errors reduce with a growing number of databases. This is the case because with an increased number of databases the model can observe a larger variety of different data characteristics and can thus more robustly predict the runtimes for an unseen database,

i.e., IMDB in this case. Interestingly, we can already achieve a reasonably small generalization error after just five different databases indicating that a moderate number of databases can be sufficient for zero-shot learning. Moreover, we clearly observe diminishing returns between 15 to 19 databases.

We can thus conclude that the number of training datasets from the benchmark is indeed sufficient to allow a zero-shot model to generalize robustly to unseen databases from the benchmark and that further datasets will likely not improve the model performance.

Workload & Data drifts. While zero-shot cost models generalize to some extent under workload and data drifts as demonstrated in Section 6.3, there is clearly a point where the evaluation workload differs too severely from any workload observed at training time by the model and where the performance will degrade as discussed Section 2.3. In a final ablation study, we thus want to investigate further how quickly the performance degrades in such cases and thus intentionally create evaluation workloads that differ largely from the training workload (due to significantly more joins, group by attributes, number of predicates, aggregations or larger dataset sizes). As we can see in Figure 13, the more different the test database and workload is from the training workload and data, the more the performance degrades. However, while the zero-shot model can still predict costs relatively accurately for cases close to training data examples (e.g., five aggregates instead of one) the performance degrades as the characteristics are further changed. Moreover, not all aspects have an equal impact on the accuracy: for example, the number of joins seems to be the most severe factor, which is expected, since more joins can result in large intermediate join sizes the model has not experienced before. Note, that in this experiment we report the results of zero-shot models using cardinality estimates of the Postgres optimizer but we could observe similar effects for the other methods as well.

7 RELATED WORK

Learned Cost Estimation. Closest to our work are workload-driven approaches for cost estimation. Neural predictions models [24, 30] have been proposed for cost estimation by featurizing the physical query plan as a tree. However, the models are workload-driven and thus require thousands of query executions for an unseen database. Recently, a framework has been proposed to efficiently gather this training data [32]. In contrast, zero-shot learning completely alleviates the need to run a representative workload for new databases. Moreover, workload-driven models were extended by improving inference and training performance [11] and to concurrent query latency prediction [37]. These ideas are orthogonal and could potentially be applied to zero-shot learning as well. An alternative to reduce the required training queries for cost estimation is DBMS fitting [8] where the idea is to model the operator complexity and adjust this basic model by fitting the parameters using differentiable programming. However, the operator complexity has to be modeled explicitly, which can be impossible for complex queries.

Earlier work proposes to use statistical methods to predict the costs of queries. For instance, it was proposed to learn models at a per-operator level [2, 18] to predict the overall query runtime. However, since interactions of operators cannot be learned and the models are thus too simplistic, the performance is inferior to

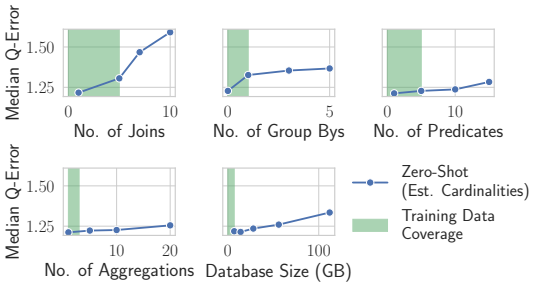


Figure 13: Accuracy under Workload and Data Drifts. Training data coverage is shown in green (e.g., the training data contains 0-5-way joins and we generalize up to 10-way joins). While zero-shot cost models generalize reasonably well to unseen workload and data characteristics, we see an increase in estimation errors for more severe drifts as expected.

workload-driven approaches [24]. An alternative idea is to represent query plans as flat vectors [5] to treat cost estimation using supervised regression, which we have shown to be less accurate than zero-shot cost estimation (cf. Section 6.1). In addition, it was suggested to leverage query executions on smaller data samples [4, 31] or queries sharing common subexpressions [29, 33] for cost estimation. In both cases, the test workload needs to closely resemble the train workload for the models to be effective.

Learned DBMS components and Design Advisors. Machine learning has been applied more broadly to optimize DBMS systems by replacing traditional approaches for tasks such as query optimization [14, 21–23] or query scheduling [20, 28]. In addition, it was applied to knob tuning [36], materialized view selection [7, 19], index selection [16] or partitioning [9]. Note that all these approaches are workload-driven since query executions on the test database are required to train the models.

8 CONCLUSION AND FUTURE WORK

In this paper, we have demonstrated that it is possible to accurately and robustly predict query runtimes on entirely unseen databases, i.e., in a zero-shot setting. In addition, fine-tuning the zero-shot models to obtain few-shot models can further improve the performance if training queries are available on the new database. We enabled this by deriving a transferable representation of queries that generalizes across databases and a specialized model architecture.

As a future direction, we argue that zero-shot learning even has a much broader applicability and could be applied to a large set of learned DBMS components including design advisors etc. Furthermore, we believe that the underlying principles can be applied to an even broader set of data systems (e.g., data streaming systems).

ACKNOWLEDGMENTS

We thank the reviewers for their feedback. This research is funded by the BMBF project within the “The Future of Value Creation – Research on Production, Services and Work” program, the Hochtief project *AICO* (AI in Construction), the HMWK cluster project *3AI* (The Third Wave of AI), as well as the DFG Collaborative Research Center 1053 (MAKI). Finally, we want to thank hessian.AI at TU Darmstadt as well as DFKI Darmstadt.

REFERENCES

[1] [n.d.]. *TPC-H benchmark*. Retrieved December 10, 2021 from http://www.tpc.org/tpch/

[2] Mert Akdere, Ugur Çetintemel, Matteo Riondato, Eli Upfal, and Stanley B. Zdonik. 2012. Learning-based Query Performance Modeling and Prediction. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, Anastasios Kementsietsidis and Marcos Antonio Vaz Salles (Eds.). IEEE Computer Society, 390–401. https://doi.org/10.1109/ICDE.2012.64

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[4] Andrew D. Ferguson, Peter Bodík, Srikanth Kandula, Eric Boutin, and Rodrigo Fonseca. 2012. Jockey: guaranteed job latency in data parallel clusters. In *European Conference on Computer Systems, Proceedings of the Seventh EuroSys Conference 2012, EuroSys '12, Bern, Switzerland, April 10-13, 2012*, Pascal Felber, Frank Bellosa, and Herbert Bos (Eds.). ACM, 99–112. https://doi.org/10.1145/2168836.2168847

[5] Archana Ganapathi, Harumi A. Kuno, Umeshwar Dayal, Janet L. Wiener, Armando Fox, Michael I. Jordan, and David A. Patterson. 2009. Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, Yannis E. Ioannidis, Dik Lun Lee, and Raymond T. Ng (Eds.). IEEE Computer Society, 592–603. https://doi.org/10.1109/ICDE.2009.130

[6] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, 1263–1272. http://proceedings.mlr.press/v70/gilmer17a.html

[7] Yue Han, Guoliang Li, Haitao Yuan, and Ji Sun. 2021. An Autonomous Materialized View Management System with Deep Reinforcement Learning. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 2159–2164. https://doi.org/10.1109/ICDE51399.2021.00217

[8] Benjamin Hilprecht, Carsten Binnig, Tiemo Bang, Muhammad El-Hindi, Benjamin Hättasch, Aditya Khanna, Robin Rehrmann, Uwe Röhm, Andreas Schmidt, Lasse Thostrup, and Tobias Ziegler. 2020. DBMS Fitting: Why should we learn what we already know?. In *10th Conference on Innovative Data Systems Research, CIDR 2020, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2020/papers/p34-hilprecht-cidr20.pdf

[9] Benjamin Hilprecht, Carsten Binnig, and Uwe Röhm. 2020. Learning a Partitioning Advisor for Cloud Databases. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 143–157. https://doi.org/10.1145/3318464.3389704

[10] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2020. DeepDB: Learn from Data, not from Queries! *Proc. VLDB Endow.* 13, 7 (2020), 992–1005. https://doi.org/10.14778/3384345.3384349

[11] Johan Kok Zhi Kang, Gaurav, Sien Yi Tan, Feng Cheng, Shixuan Sun, and Bingsheng He. 2021. Efficient Deep Learning Pipelines for Accurate Cost Estimations Over Large Scale Query Workload. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1014–1022. https://doi.org/10.1145/3448016.3457546

[12] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. 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.). 3146–3154. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[13] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. 2019. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2019/papers/p101-kipf-cidr19.pdf

[14] Sanjay Krishnan, Zongheng Yang, Ken Goldberg, Joseph M. Hellerstein, and Ion Stoica. 2018. Learning to Optimize Join Queries With Deep Reinforcement Learning. *CoRR* abs/1808.03196 (2018). arXiv:1808.03196 http://arxiv.org/abs/1808.03196

[15] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 951–958. https://doi.org/10.1109/CVPR.2009.5206594

[16] Hai Lan, Zhifeng Bao, and Yuwei Peng. 2020. An Index Advisor Using Deep Reinforcement Learning. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (Eds.). ACM, 2105–2108. https://doi.org/10.1145/3340531.3412106

[17] Viktor Leis, Bernhard Radke, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2018. Query optimization through the looking glass, and what we found running the Join Order Benchmark. *VLDB J.* 27, 5 (2018), 643–668. https://doi.org/10.1007/s00778-017-0480-7

[18] Jiexing Li, Arnd Christian König, Vivek R. Narasayya, and Surajit Chaudhuri. 2012. Robust Estimation of Resource Consumption for SQL Queries using Statistical Techniques. *Proc. VLDB Endow.* 5, 11 (2012), 1555–1566. https://doi.org/10.14778/2350229.2350269

[19] Xi Liang, Aaron J. Elmore, and Sanjay Krishnan. 2019. Opportunistic View Materialization with Deep Reinforcement Learning. *CoRR* abs/1903.01363 (2019). arXiv:1903.01363 http://arxiv.org/abs/1903.01363

[20] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. 2019. Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*, Jianping Wu and Wendy Hall (Eds.). ACM, 270–288. https://doi.org/10.1145/3341302.3342080

[21] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2021. Bao: Making Learned Query Optimization Practical. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1275–1288. https://doi.org/10.1145/3448016.3452838

[22] Ryan Marcus and Olga Papaemmanouil. 2018. Deep Reinforcement Learning for Join Order Enumeration. In *Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, aiDM@SIGMOD 2018, Houston, TX, USA, June 10, 2018*, Rajesh Bordawekar and Oded Shmueli (Eds.). ACM, 3:1–3:4. https://doi.org/10.1145/3211954.3211957

[23] Ryan C. Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A Learned Query Optimizer. *Proc. VLDB Endow.* 12, 11 (2019), 1705–1718. https://doi.org/10.14778/3342263.3342644

[24] Ryan C. Marcus and Olga Papaemmanouil. 2019. Plan-Structured Deep Neural Network Models for Query Performance Prediction. *Proc. VLDB Endow.* 12, 11 (2019), 1733–1746. https://doi.org/10.14778/3342263.3342646

[25] Jan Motl and Oliver Schulte. 2015. The CTU Prague Relational Learning Repository. *CoRR* abs/1511.03086 (2015). arXiv:1511.03086 http://arxiv.org/abs/1511.03086

[26] Raghunath Othayoth Nambiar and Meikel Poess. 2006. The Making of TPC-DS. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim (Eds.). ACM, 1049–1058. http://dl.acm.org/citation.cfm?id=1164217

[27] Patrick E. O'Neil, Elizabeth J. O'Neil, Xuedong Chen, and Stephen Revilak. 2009. The Star Schema Benchmark and Augmented Fact Table Indexing. In *Performance Evaluation and Benchmarking, First TPC Technology Conference, TPCTC 2009, Lyon, France, August 24-28, 2009, Revised Selected Papers (Lecture Notes in Computer Science)*, Raghunath Othayoth Nambiar and Meikel Poess (Eds.), Vol. 5895. Springer, 237–252. https://doi.org/10.1007/978-3-642-10424-4_17

[28] Yangjun Sheng, Anthony Tomasic, Tieying Zhang, and Andrew Pavlo. 2019. Scheduling OLTP transactions via learned abort prediction. In *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, aiDM@SIGMOD 2019, Amsterdam, The Netherlands, July 5, 2019*, Rajesh Bordawekar and Oded Shmueli (Eds.). ACM, 1:1–1:8. https://doi.org/10.1145/3329859.3329871

[29] Tarique Siddiqui, Alekh Jindal, Shi Qiao, Hiren Patel, and Wangchao Le. 2020. Cost Models for Big Data Query Processing: Learning, Retrofitting, and Our Findings. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 99–113. https://doi.org/10.1145/3318464.3380584

[30] Ji Sun and Guoliang Li. 2019. An End-to-End Learning-based Cost Estimator. *Proc. VLDB Endow.* 13, 3 (2019), 307–319. https://doi.org/10.14778/3368289.3368296

[31] Shivaram Venkataraman, Zongheng Yang, Michael J. Franklin, Benjamin Recht, and Ion Stoica. 2016. Ernest: Efficient Performance Prediction for Large-Scale Advanced Analytics. In *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*, Katerina J. Argyraki and Rebecca Isaacs (Eds.). USENIX Association, 363–378. https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/venkataraman

[32] Francesco Ventura, Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. 2021. Expand your Training Limits! Generating Training Data for ML-based Data Management. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1865–1878. https://doi.org/10.1145/3448016.3457286

[33] Chenggang Wu, Alekh Jindal, Saeed Amizadeh, Hiren Patel, Wangchao Le, Shi Qiao, and Sriram Rao. 2018. Towards a Learning Optimizer for Shared Clouds. *Proc. VLDB Endow.* 12, 3 (2018), 210–222. https://doi.org/10.14778/3291264.3291267

[34] Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. 2020. NeuroCard: One Cardinality Estimator for All Tables. *Proc. VLDB Endow.* 14, 1 (2020), 61–73. https://doi.org/10.14778/3421424.3421432

[35] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. 2017. Deep Sets. 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.). 3391–3401. https://proceedings.neurips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html

[36] Ji Zhang, Yu Liu, Ke Zhou, Guoliang Li, Zhili Xiao, Bin Cheng, Jiashu Xing, Yangtao Wang, Tianheng Cheng, Li Liu, Minwei Ran, and Zekang Li. 2019. An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning. 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, 415–432. https://doi.org/10.1145/3299869.3300085

[37] Xuanhe Zhou, Ji Sun, Guoliang Li, and Jianhua Feng. 2020. Query Performance Prediction for Concurrent Queries using Graph Embedding. *Proc. VLDB Endow.* 13, 9 (2020), 1416–1428. https://doi.org/10.14778/3397230.3397238