Opening The Black-Box: Explaining Learned Cost Models For Databases

Roman Heinrich
TU Darmstadt & DFKI
roman.heinrich@dfki.de

Oleksandr Havrylov
TU Darmstadt
sashka.havr@gmail.com

Manisha Luthra
TU Darmstadt & DFKI
manisha.luthra@dfki.de

Johannes Wehrstein
TU Darmstadt
johannes.wehrstein@tu-darmstadt.de

Carsten Binnig
TU Darmstadt & DFKI
carsten.binnig@tu-darmstadt.de

ABSTRACT

Learned Cost Models (LCMs) have shown superior results over traditional database cost models as they can significantly improve the accuracy of cost predictions. However, LCMs still fail for some query plans, as prediction errors can be large in the tail. Unfortunately, recent LCMs are based on complex deep neural models, and thus, there is no easy way to understand where this accuracy drop is rooted, which critically prevents systematic troubleshooting. In this demo paper, we present the very first approach for opening the black box by bringing AI explainability approaches to LCMs. As a core contribution, we developed new explanation techniques that extend existing methods that are available for the general explainability of AI models and adapt them significantly to be usable for LCMs. In our demo, we provide an interactive tool to showcase how explainability for LCMs works. We believe this is a first step for making LCMs debuggable and thus paving the road for new approaches for systematically fixing problems in LCMs.

PVLDB Reference Format:
Roman Heinrich, Oleksandr Havrylov, Manisha Luthra, Johannes Wehrstein, and Carsten Binnig. Opening The Black-Box: Explaining Learned Cost Models For Databases. PVLDB, 18(12): 5255 - 5258, 2025.
doi:10.14778/3750601.3750645

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/DataManagementLab/demo-explain-lcms.

1 INTRODUCTION

The Potential and Risk of Learned Cost Models. *Learned Cost Models* (LCMs) for databases were proposed to predict query execution costs [3, 5, 6]. Overall, LCMs demonstrate high prediction accuracy, typically outperforming classical approaches [2]. However, LCMs can still produce large errors for individual queries, which is reflected by high tail prediction errors [3]. Critically, such bad estimates of LCMs lead to sub-optimal plan selections during query optimization, which in turn cause prolonged execution runtimes and thus reduced performance [2].

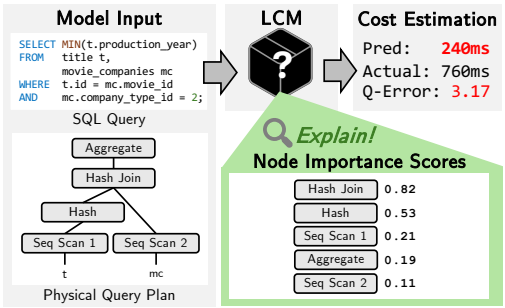

Figure 1: LCMs predict query execution costs, but they can be off in their predictions. Unfortunately, with current LCMs, it is unclear how they came to their prediction. We leverage explanation algorithms to provide information on which nodes in a query plan contributed most significantly to the runtime prediction, which allows users to understand the model's internal behavior better and contrast it with the actual runtime per operator to identify potential problems of the model as we show in Figure 2.

Black-Box Behavior of LCMs. Unfortunately, many recent LCMs that provide high precision are based on deep learning models. Thus, the predictions are produced by models that are a black-box to database administrators, making it hard to understand their prediction behavior systematically. In fact, if LCMs provide erroneous predictions, it is unclear *why* they mispredicted the costs for a given plan. Consequently, a critical downside of LCMs is that we cannot debug them, i.e., trace why they mispredict the costs of a given query plan, which critically prevents systematic troubleshooting.

Making LCMs Debuggable. In this demo paper, we aim to open the black-box of LCMs and achieve debuggability by making their predictions *explainable*. The reason is that the knowledge learned by LCMs is hidden in their model parameters. Thus, their predictions are inexplicable. To this end, we propose a novel approach to explain the internalized, learned knowledge of LCMs. The main idea is that we apply ideas of node importance for LCMs. Overall, we believe our approach thus helps to improve the design of future approaches, as it enables the identification of biases in the models, learning strategies, or training data.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750645

An Example. We illustrate our approach on a high level in Figure 1. The task of a LCM (middle) is to estimate query execution costs. Thus, it takes a SQL query as input (left), typically in the form of the physical plan, as this provides information about the planned execution. The LCM predicts a query runtime (right) for the given plan, which ideally is close to the actual predicted costs, reflected by a small prediction error. However, in the example, the LCM predicts a very low runtime of `240ms` that significantly ($\approx 3.17\times$) deviates from the actual runtime of `760ms`, which is clearly an undesirable behavior. Given such a query plan, our demo allows to *explain* the prediction of the LCM. In particular, we leverage and adapt various explainer algorithms, like `GNNExplainer` [8], to indicate how important every node of the query plan was for the prediction. In our example, the explainer returns a high importance score of `0.82` for the Hash Join, indicating that this node was highly important during the prediction. Overall, this correlates with the actual importance, as the Hash Join had the highest runtime during the execution. However, the explainer assigned a feature importance of `0.19` for the Aggregate, which is higher than the score for Seq Scan 2 of `0.11`. This is unexpected, as it is not reflected in the actual execution runtimes in this query plan and indicates that the LCM is not yet able to rank the costs of both operators correctly.

Our Contribution: Explainability for LCMs. At the core, our novel approach identifies which operators of a query plan were most important for the runtime prediction according to the LCM. To achieve this, we significantly adapted existing explainability approaches as follows (cf. Section 3.2): First, we adapted explainers for regression, which is required for cost estimation, as existing explainers typically only support classification. Second, we introduce a new masking strategy for query operators, which allows us to determine which operators have high relevance for predictions. Third, we propose novel metrics to explain LCMs, such as the correlation between the actual runtime and the node importance scores (cf. Section 2.3). We also integrated our approach into an interactive tool (cf. Section 3.3) that allows an interactive comparison between the internalized model knowledge and the actual dependencies of a query plan.

2 EXPLAINING LCMS

In the following, we first provide an overview of LCMs to provide the required context before we describe our methodology and the proposed explanation metrics.

2.1 Learned Cost Models

The core idea of a LCM is to learn query execution costs from previously executed query traces. As neural networks can learn arbitrarily complex functions, this approach is promising to outperform classical models, which often misestimate query execution costs due to simplifying assumptions. Importantly, many LCMs rely on a graph-based representation designed to capture the properties of the physical query graph. As shown in [2], recent graph-based models typically achieve the best performance for cost estimation, and thus, we pick the state-of-the-art `Zero-Shot` approach for this demo as a representative example [3]. Importantly, this approach employs *Graph Neural Network*s (GNNs) to learn the query execution costs from a graph representing the query plan, filter conditions and tables. However, due to the black-box nature of GNNs, it remains unclear how the model came to its prediction, which hinders debuggability and thus troubleshooting. Finally, our ideas can also be applied beyond GNNs, as explainers for node importance also exist for other model architectures.

2.2 Methodology

The fundamental question for this work is: *How to explain the predictions of a LCM?* In particular, we want to understand why the model predicts a runtime for a given plan but was highly off in its prediction. Clearly, the predictions depend both on the trained model and the model input, which is the featurized query plan. Thus, to answer this question, our approach provides local post-hoc explanations for a given query plan and a trained LCM. In particular, we focus on the *node importance*, which specifies how important the presence of a given node (i.e., a query operator in a cost model) was for the prediction. The node importance thus highlights the importance of given query operators for the runtime prediction. For instance, if a query plan employs an expensive join operator that largely affects the runtime, this should be reflected in its node importance. Mismatches between the operator runtimes and the provided importance scores indicate an incorrect model assumption of the LCM about the query plan. To quantify this, we propose novel metrics in Section 2.3, which we also discuss in a more detailed example in Section 3.1. To obtain node importance scores, we adapted recent explanation algorithms (cf. Section 3.2).

2.3 Explanation Metrics

As a core contribution, we propose new explanation metrics relying on node importance scores that are included in our demo:

(1) **Node Ranking:** *Which nodes did the model prioritize the most in its predictions?* After obtaining the importance scores, we sorted them in descending order to identify the nodes that contributed most to the prediction. This ranking offers valuable insights into the model's internalized knowledge, such as which operators are consistently overlooked, poorly considered, overemphasized, or fail to align with the expectations of database experts regarding their execution costs.

(2) **Runtime Correlation:** *How well does the node importance match the actual runtime?* In addition to the node ranking, we argue that the importance scores of given operators in the query plan should correlate with their actual runtime, i.e., longer-running operators should have a higher impact on cost estimation. To report this correlation, we visualize the normalized runtime and normalized importance scores to make the ratios easily comparable. A mismatch of runtime fraction and importance fraction again shows false assumptions and mistakes of the LCM. Note that we subtract the runtimes of the sub-plans of a given node to isolate its runtime. Similarly, we include the correlation with cardinalities, and we visualize correlation metrics like Spearman's rank.

(3) **Explanation Quality:** *How reliable are the explanations?* The quality of the explanations also plays an essential role when using them for making LCMs debuggable. Importantly, explanations can vary depending on the explainer, and not every

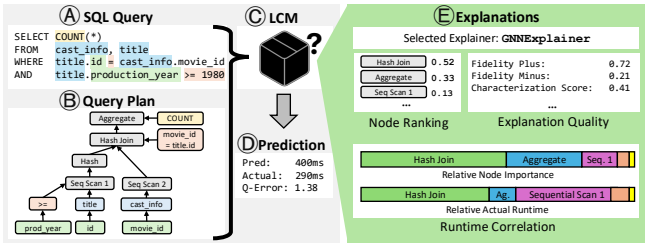

Figure 2: Example Query with a prediction and explanations. Execution costs for a given SQL query should be obtained Ⓐ. For this, it is translated to a query plan Ⓑ that employs additional nodes for filter conditions and join conditions [3] and passed to the LCM Ⓒ. The LCM predicts query execution costs Ⓓ, which, however, can be off. For this reason, our demo provides explanations of the prediction Ⓔ for a given explainer in the form of node ranking, runtime correlation and evaluation metrics.

explanation has the same quality. To ensure that the explanations are reliable, we thus adapted standard evaluation metrics in the demo to assess the quality of the explanations. We report `Fidelity+` and `Fidelity-` [9], which describe how close a varied prediction that is based on the explanation is to the prediction of the original input graph. For `Fidelity+`, we compare the prediction using the query graph without the most important nodes according to the explainer and compare it against the original prediction. We evaluate similarly for `Fidelity-`, omitting the least important nodes. To apply this metric for `Zero-Shot` models, we zeroed out the hidden states of the corresponding node to mask the node out but not affect message passing [3]. Finally, we report the `CharacterizationScore` as a harmonic mean of these metrics [4]. In our demonstration, we scale all metrics to 0 and 1, where 1 means a higher quality.

3 DEMONSTRATION OVERVIEW

In this section, we give an overview of our demo by presenting an example scenario and the novel explanation algorithms for LCMs, which are the main contributions of this work.

3.1 Example Scenario

In the following, we discuss an example scenario of our approach as shown in Figure 2. This example closely follows the interaction flow of our demo that is presented in detail in Section 3.3. Ⓐ A SQL query that operates on the IMDB dataset is used as input for cost estimation. Ⓑ Next, the query is converted into a query graph that contains the physical operators. In this work, we follow the `Zero-Shot` featurization [3], which, in addition to the physical operators, introduces graph nodes, e.g., for filter predicates (e.g., >=) or table columns (e.g., prod_year). Ⓒ The physical query plan is passed to the LCM, which uses it for cost prediction. Ⓓ The LCM predicts a runtime that deviates from the actual runtime. The prediction quality is typically expressed as Q-Error [5]. In this example, the Q-Error is 1.38, indicating a mis-prediction of 38%. Ⓔ Finally, our demo allows to *explain* the predictions the LCM by providing the explanation metrics from Section 2.3.

In our demo, we include the importance scores and previously introduced explanation metrics of several explainers that we extended for LCMs (cf. Section 3.2). For example, `GNNExplainer` (which is one of our explanation algorithms) returns the following node importance scores and explanation metrics: *(1) Node Ranking*: According to the node ranking, the Hash Join is the most important node with an importance of 0.52. Interestingly, the Aggregate is the second most important node, having an importance score of 0.33. *(2) Runtime Correlation*: We see that the relative node importance and relative runtimes fractions have similarities but do not really match for all operators. While the Hash Join is most important in both the actual runtime and the provided node importance, this is not the case for the Aggregate, which gets a higher node importance score than it should have according to its short runtime. This indicates that the LCM overestimated the importance of aggregations in this query plan. *(3) Explanation Quality*: Finally, we also report on the explanation quality using a metric called `Fidelity+`, which is 0.72 in the example, indicating that the explanation successfully determined the most important nodes for the prediction. However, the least important nodes were less accurately estimated with a `Fidelity+`-Score of 0.21. Furthermore, the `CharacterizationScore` is 0.41, indicating an average explanation quality.

3.2 Explainer Algorithms

In the following, we introduce the explainer algorithms that we adapted for explaining LCMs. Importantly, we selected explainers that apply to GNN-based cost models, but the ideas could also be extended to other model architectures. All these explainers are also provided in our demo. Overall, the selected explanation algorithms provide the node importance, which is a score describing how important a given node was for a prediction. We normalize the scores to values between 0 and 1, where a high value indicates a high importance during the prediction. We focused on two classes of explanation algorithms:

Gradient-based approaches directly leverage the trained gradients of the neurons to explain the predictions of the models. In particular, they analyze how small changes in input affect the model's output. *Perturbation-based approaches* are derived from the idea that the output of a model changes differently with respect to various input perturbations like removed nodes or altered features. Removing an important input node or feature changes is expected to change the output significantly, while removing an unimportant node or feature should not impact the prediction. To apply this for `Zero-Shot` cost models [3], we zeroed out the hidden states of a given node to mask it out. To adapt these approaches for regression, we introduce a relative loss when comparing the masked with the actual predictions. We adopted the following explanation algorithms:

(1) `SensitivityAnalysis` [1] is a gradient-based explainer. It assumes that for a given query plan, higher absolute gradient values indicate that the corresponding features are more important than the others. Thus, it uses the values of the gradients and transforms them into importance scores for the input nodes.
(2) `GuidedBackprop` [7] is also a gradient-based method that, in contrast, additionally clamps negative gradients to zero during the backward pass. The intuition is that negative gradients are difficult to interpret, so only positive gradients are considered.

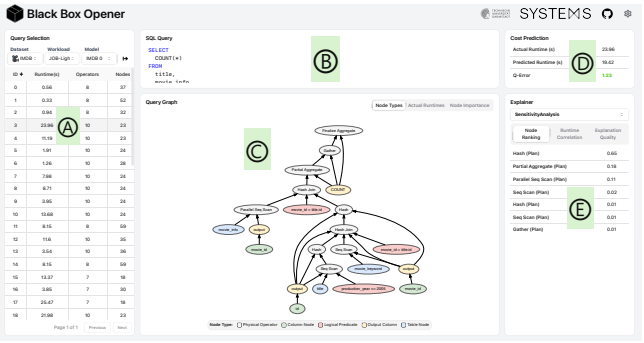

Figure 3: Overview of our Demo: Ⓐ First, a query plan is selected from given datasets and workloads. Ⓑ The SQL query is visualized. Ⓒ The query plan consists of operator, table and predicate nodes. Ⓓ The predicted and actual runtime is shown together with the estimation error (Q-Error). Ⓔ Finally, an explainer can be selected, and the explanation metrics are displayed.

That way, `GuidedBackprop` guides the explanation of the features that affect the output, which is designed to improve the interpretability of explanations.

(3) `GNNExplainer` [8] is, in contrast to the other explainers, a perturbation-based method. To obtain importance scores for nodes and node features, it learns soft masks for nodes to explain predictions via mask optimization. First, it randomly initializes soft masks for both nodes and node features and treats them as trainable variables. Then, an optimizer is used to minimize the difference between the original prediction and the masked prediction while maximizing the sparsity of the mask.

(4) `DiffMask`: Additionally, we propose `DiffMask` as a perturbation-based explainer. It is based on the idea that the prediction from the masked input graph is different from the prediction from the original graph. We derive node importance directly from the difference between the prediction with the corresponding node masked and the original prediction.

3.3 User Interface

In the following, we describe how users can interact with our demonstration as shown in Figure 3: Overall, our demo is an interactive web-based tool, allowing us to browse and explore datasets, query plans, models, and explanations interactively. Ⓐ At first, the dataset is selected. We include several datasets from standard cost estimation datasets, including IMDB, TPC-H, and Baseball. Moreover, queries from a selected workload can be chosen, like JOB-Light for IMDB. In addition, we include various synthetic workloads with varying query complexity. For each workload, a list of query plans is shown along with various metadata, such as the number of plan operators etc., such that users can select plans of different complexity. Once a query is selected in Ⓐ, the SQL query string is shown in Ⓑ, and in Ⓒ, users can see the query plan. Here, all nodes and edges of the GNN-based model are displayed. Ⓓ shows the predicted and actual runtime along with the Q-Error for the selected LCM. Ⓔ most importantly shows the result of our explanation. Here, the explainer algorithm can be selected with a drop-down bar. Below, the user can see the metrics discussed in Section 2.3. In addition, our demo colorizes the nodes in Ⓑ according to the importance scores or the actual runtimes to allow better comparisons.

4 SUMMARY AND OUTLOOK

Initial Insights. When using our demo to explain `Zero-Shot` cost models, it provided interesting initial insights. Often, we observed a high correlation between the actual runtimes and the node importance scores. However, we also observed accurate predictions where the node importance did not really match the actual runtimes. This shows that although the model did not always learn what it actually should, it still can be accurate. Still, we believe this mismatch must be overcome to provide more robust predictions. In addition, we observed that the aggregation nodes are often highly important, which contradicts their actual runtime. This potentially shows that the model puts too much emphasis on aggregation nodes.

Outlook. Overall, we showed that our demo provides valuable insights into the prediction behavior of LCMs and thus helps to make them explainable. This paves the road for future improvements for LCMs, affecting model and feature design, training strategies and data collection to improve reliability. We see further potential by exploring more dimensions, such as adding more recent explainer algorithms. Another interesting avenue is to not only use node importance but instead evaluate feature importance. Similarly, the scope can be extended to sub-graph importance, i.e., investigating which sub-plan of the query was most important for the prediction.

ACKNOWLEDGMENTS

This work is funded by the LOEWE program (Reference III 5 - 519/05.00.003-(0005)), etaGPT project under grant number 03EN4107, hessian.AI and Athene Young Investigator Programme at TU Darmstadt, as well as DFKI Darmstadt. It has benefited from the early stages of the funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy— EXC-3057; funding will begin in 2026.

REFERENCES

[1] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to Explain Individual Classification Decisions. *J. Mach. Learn. Res.* 11 (2010).

[2] Roman Heinrich, Manisha Luthra, Johannes Wehrstein, Harald Kornmayer, and Carsten Binnig. 2025. How Good are Learned Cost Models, Really? Insights from Query Optimization Tasks. *Proc. ACM Manag. Data* 3, 3, Article 172 (June 2025).

[3] Benjamin Hilprecht and Carsten Binnig. 2022. Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction. *Proc. VLDB Endow.* 15, 11 (2022).

[4] Jaykumar Kakkad, Jaspal Jannu, Kartik Sharma, Charu Aggarwal, and Sourav Medya. 2023. A Survey on Explainability of Graph Neural Networks. *IEEE Data Eng. Bull.* 46, 2 (2023).

[5] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. 2019. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *CIDR 2019*.

[6] Ryan Marcus and Olga Papaemmanouil. 2019. Plan-Structured Deep Neural Network Models for Query Performance Prediction. *Proc. VLDB Endow.* 12, 11 (2019).

[7] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2015. Striving for Simplicity: The All Convolutional Net. In *ICLR 2015*.

[8] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. In *NeurIPS 2019*.

[9] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2023. Explainability in Graph Neural Networks: A Taxonomic Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 5 (2023).