A ADDITIONAL MICROBENCHMARKS

A.1 Inconsistent Features.

We showed the examples above in a simple setting. Similar spurious correlates can be observed for joins, and correlation effects from different tables. But joins also introduce another major challenge for workload shifts: the features in the new workload can be inconsistent with features from the training regime. This is because the two keys in a join refer to the same semantic column. Past featurization approaches, such as sample bitmaps, treat these as independent columns.

Setup. We consider two tables, primary, and foreign, where primary is a table with $10K$ unique $primary.id$ values, and $foreign.pid$ is a foreign key reference to $primary.id$. $foreign$ has $30K$ values, because several pid values will be repeated. The number of repetitions of pid follows a Zipf distribution with $a = 1.1$, so the repetitions range from 1 to a few hundred.

Figure 20 presents query templates showing a workload drift, and the result of MSCN and Robust-MSCN model trained on the first template. The filters in the new template would be ignored by the MSCN model since it was not seen in training; this leads to a clear degradation of performance in the new template. Robust-MSCN recognizes that these two filters are on an equivalent column, and creates the same unified bitmap representation for both these filters.

A.2 Analyzing Performance on Query Plans.

Workload drift is defined precisely in §4.1, and our main experiments in Section 6 explore several differences between the training and evaluation (testing) queries. Here we will use a carefully selected training / evaluation query set where it is possible to understand exactly what is being learned by the query driven model, and develop intuition about our proposed training framework.

Training / Evaluation workload. We take two templates in CEB that are very similar to each other — one has filters on the column keyword.keyword (template 2b), and one doesn't (template 2a). Each template has a few hundred queries, with filters on several other columns which are generated using the same rules for both the templates. The join graph for queries from this template is shown in Figure 19a. The goal of a cardinality estimation model is to estimate cardinalities of each subplan, i.e. join query involving a subset of the tables in the full query, which are then used by the query optimizer to find the join order to execute the query. There are 290 such subplan estimates in this template.

Effect of the different filter. Intuitively, the training workload and the evaluation workload are very close to each other since almost all the filters are from the same distribution. There is an additional filter on the table 'keyword' in the evaluation queries. This selects for titles with particular keywords, such as 'marvel', or 'superheroes'. Any subplan that involves 'keyword' will naturally get smaller in the evaluation queries. This represents a consistent change from the training workload in all the queries from the new template. Note that the data features — PostgreSQL estimates of the cardinalities — will reflect this change: i.e., the estimate for any subplan with a filter on keyword will have lower cardinality than the estimate for the same subplan without a filter on keyword.

MSCN model is brittle. On new queries from 2b, the training template, MSCN outperforms PostgreSQL. However, on the unseen template, this same MSCN model has a lot of variance across three runs — going from being similar to PostgreSQL to being many times worse than PostgreSQL.

Robust-MSCN model stably improves in both scenarios. Robust-MSCN uses the same training data and features as the MSCN model, but is trained with the query masking technique described in §4.2. Intuitively, it puts more importance on the data features while training. Across all three runs, it improves clearly over PostgreSQL on both the 2b (training template), and 2a (workload drift template).

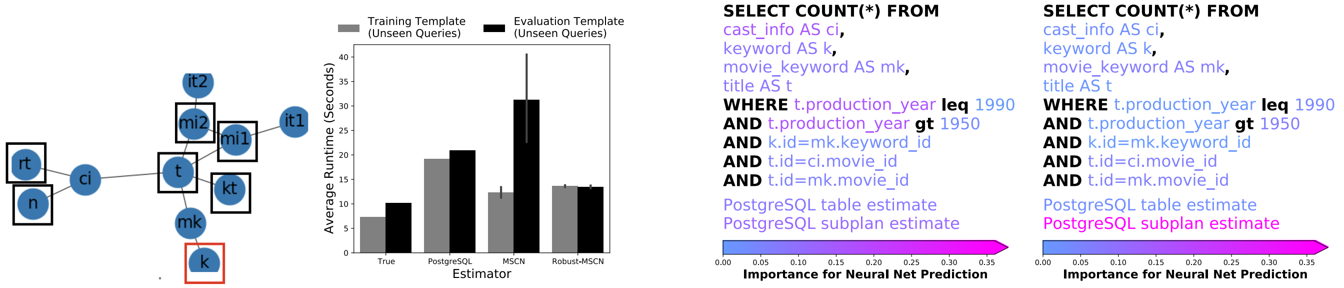
How different are the MSCN and Robust-MSCN models? We apply interpretablity techniques developed to understand what a deep neural network learns to the MSCN and Robust-MSCN model, and find that they use the input features very differently. We use the integrated gradients algorithm [36] implemented in the Captum library [19]. For a given input, the algorithm considers a neural network's gradient activations for several inputs that interpolate from all zeros to the actual input. Then, it analyzes the gradients to quantify how important each feature was for the model's observed output. We show this analysis for an example subplan of a query from the unseen template; the importance of the input features for the MSCN model is shown in Figure 19c, and the importance of features for the Robust-MSCN model in Figure 19d.

This gives us insight into what the two models learned — and we find that they are actually quite different despite having the same training workload and input features. With the query masking approach, the importance for the PostgreSQL estimate is the highest — i.e., it relies more on that feature. In the standard approach, it relies more on the query features, such as the tables and joins, which is less robust when the evaluation workload changes. Importance attributions over other subplans also show the same pattern. As we discussed above, the effect of the workload drift in this example is coarsely captured by the lower DBMS estimates — and relying more on this feature lets the Robust-MSCN model adapt better to this change. At the same time, the Robust-MSCN model clearly improves over PostgreSQL — thus, it is effectively utilizing the information it learned from the training workload, and not just using PostgreSQL estimates.

B ADDITIONAL EXPERIMENTS

B.1 Job-Light experiments

Analyzing JOBLight-train Q-Errors. Even though we see consistent improvements over PostgreSQL in Q-Errors, typically there is still a long tail of Q-Errors. This is expected: there are new tables, and new join graphs in these workloads that were not in JOBLight-train; Figure 21a shows that the Q-Errors are much worse if no table in the subplan query is present in JOBLight-train; But, when one or more tables from the training workload are present, the Q-Errors improve. This suggests that the model is learning to incorporate information from the simple JOBLight-train workload in a sensible way over more complex queries. This is particularly important in JOB queries because several JOB queries have very extreme filters, for instance: '17f.sql' has 'n.name LIKE "%B%"', or '6e.sql' has 'n.name LIKE "%Downey%Robert%"'. JOBLight-train contains

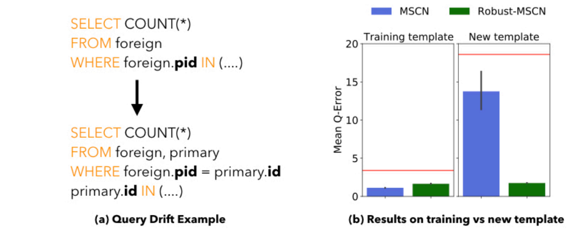


```
SELECT COUNT(*) FROM
cast_info AS ci,
keyword AS k,
movie_keyword AS mk,
title AS t
WHERE t.production_year leq 1990
AND t.production_year gt 1950
AND k.id=mk.keyword_id
AND t.id=ci.movie_id
AND t.id=mk.movie_id
```

```
SELECT COUNT(*) FROM
cast_info AS ci,
keyword AS k,
movie_keyword AS mk,
title AS t
WHERE t.production_year leq 1990
AND t.production_year gt 1950
AND k.id=mk.keyword_id
AND t.id=ci.movie_id
AND t.id=mk.movie_id
```

(a) Join graph for all queries. (b) Query performance. (c) MSCN's importance. (d) Robust-MSCN's importance.

Figure 19: The MSCN and Robust-MSCN model trained on CEB template 2b, and evaluated on unseen queries from 2b or template 2a, that only differs by one filter on column, keyword, highlighted in red. (b) Shows the runtime performance of baselines, MSCN and Robust-MSCN models. (c-d) Shows the importance of each feature to each model's output when applied on an example subplan.

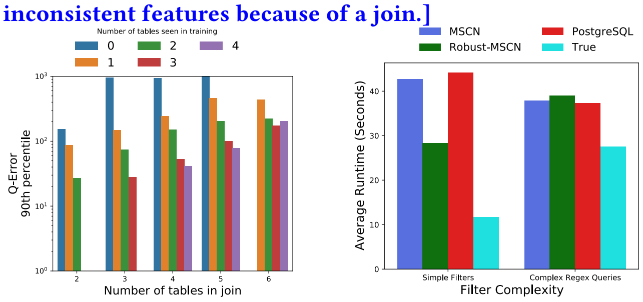


```
SELECT COUNT(*)
FROM foreign
WHERE foreign.pid IN (....)
```

```
SELECT COUNT(*)
FROM foreign, primary
WHERE foreign.pid = primary.id
primary.id IN (....)
```

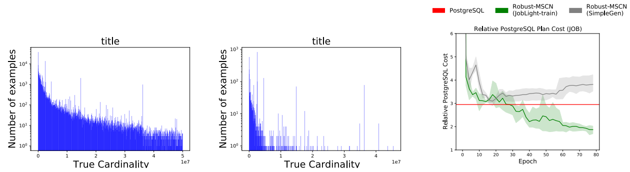
(a) Query Drift Example **(b) Results on training vs new template**

Figure 20: [Simple example of workload drift leading to inconsistent features because of a join.]



(a) Robust-MSCN Q-Errors. **(b) Simple vs. complex filters.**

Figure 21: Exploring where Robust-MSCN trained on JOBLight-train improves.



(a) JOBLight-train. **(b) SimpleGen.** **(c) Learning curves.**

Figure 22: Distribution of cardinalities in JOB-light, and another synthetically generated workload, SimpleGen. (c) compares the relative plan cost of the Robust-MSCN model as its trained on JOBLight-train or SimpleGen.

thousands of queries with filters on cast_info.person_id, thus, it could learn to correct the DBMS estimate upward for filters that select a large amount of table, or correct the estimate downward for filters that select for just one movie or actor.

What doesn't improve using just JOBLight-train queries? CEB has many more automatically generated queries than JOB, and samples filters based on their correlations across tables; therefore, it has fewer such extreme filters, which explains why we see much less improvement on it when we train on the simple JOBLight-train. Figure 21b shows that we see no improvement over a subset of the more complex regex templates on CEB, and a slight improvement over simpler templates that involve categorical filters over attributes such as 'genre', 'language' etc.

Importance of training workload's smooth cardinality distribution. We compare the performance of the Robust-MSCN model trained on two different workloads that are superficially similar: JOBLight-train, and a synthetic query generator used in [14] (SimpleGen) that also has joins only up to 3 tables. We find that the Robust-MSCN model trained on JOBLight-train does significantly better (Figure 22c). Figure 22 shows the true cardinalities of all queries that involve the table 'title' in the two workloads. This suggests that the important thing to learn robust models is to have a workload sampling process that is smooth (i.e., covers a lot of points in the space, rather than just a few discrete points that could be memorized). This ensures that the model will need to learn a smooth and continuous function that should generalize to dynamic scenarios better.

B.2 Flow-Loss comparison.

In §2, we discussed the trade-offs between Q-Error or Flow-Loss as the loss function. Figure 23 shows the runtime performance on JOB from training MSCN or Robust-MSCN on CEB using Flow-Loss.

Robust-MSCN is complementary to Flow-Loss. The standard MSCN model also improves over PostgreSQL when trained on CEB; recall from Figure 10 that MSCN with Q-Error had actually got worse. Thus, Flow-Loss also improves robustness to workload drift when the cost model is well tuned. However, using the Robust-MSCN model with Flow-Loss improves the performance further — showing that these are complementary techniques.

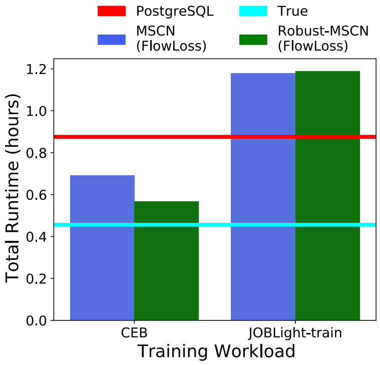


Figure 23: Total latency on JOB for models trained with FlowLoss instead of Q-Error.

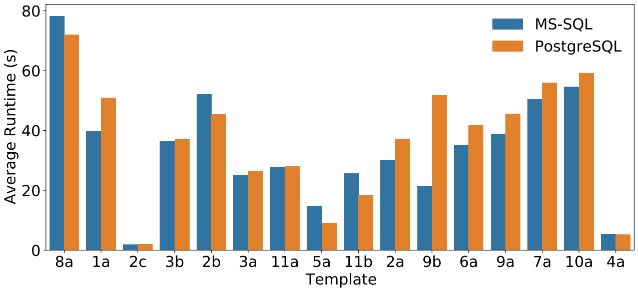


Figure 24: Comparing MS-SQL vs PostgreSQL estimates on CEB templates.

Flow-Loss does not work well without complex query workloads. Flow-Loss optimizes for an approximate plan cost; JOBLight-train contains very simple queries, and thus, there are not many query plans. So a Flow-Loss trained model may not be able to learn much there — and its performance would be unpredictable.

B.3 MS-SQL vs PostgreSQL comparison

Figure 24 compares MS-SQL vs. PostgreSQL estimates on CEB templates. In the experiments, we showed that overall, PostgreSQL estimates are worse, but there is still variance among the templates, including a few where PostgreSQL is better than MS-SQL as well.

REFERENCES

[1] 2018. PostgreSQL Cardinality Estimation Techniques. https://www.postgresql.org/docs/current/row-estimation-examples.html [Online;].

[2] 2018. SQL Server's Join Cardinality Estimation. https://www.sqlshack.com/join-estimation-internals/ [Online;].

[3] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999. The Aqua Approximate Query Answering System. In SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA, Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh (Eds.). ACM Press, 574–576. https://doi.org/10.1145/304182.304581

[4] Pierre Baldi and Peter J Sadowski. 2013. Understanding dropout. Advances in neural information processing systems 26 (2013).

[5] Anshuman Dutt, Chi Wang, Vivek R. Narasayya, and Surajit Chaudhuri. 2020. Efficiently Approximating Selectivity Functions using Low Overhead Regression Models. Proc. VLDB Endow. 13, 11 (2020), 2215–2228. http://www.vldb.org/pvldb/vol13/p2215-dutt.pdf

[6] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek R. Narasayya, and Surajit Chaudhuri. 2019. Selectivity Estimation for Range Predicates using Lightweight Models. Proc. VLDB Endow. 12, 9 (2019), 1044–1057. https://doi.org/10.14778/3329772.3329780

[7] Ergast. 2021. Ergast F1 Database Schema. https://relational.fit.cvut.cz/assets/img/datasets-generated/ErgastF1.svg [Online;].

[8] Lise Getoor, Benjamin Taskar, and Daphne Koller. 2001. Selectivity Estimation using Probabilistic Models. In Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001, Sharad Mehrotra and Timos K. Sellis (Eds.). ACM, 461–472. https://doi.org/10.1145/375663.375727

[9] Max Halford, Philippe Saint-Pierre, and Franck Morvan. 2020. Selectivity correction with online machine learning. arXiv preprint arXiv:2009.09884 (2020).

[10] Yuxing Han, Ziniu Wu, Peizhi Wu, Rong Zhu, Jingyi Yang, Liang Wei Tan, Kai Zeng, Gao Cong, Yanzhao Qin, Andreas Pfadler, Zhengping Qian, Jingren Zhou, Jiangneng Li, and Bin Cui. 2021. Cardinality Estimation in DBMS: A Comprehensive Benchmark Evaluation. Proc. VLDB Endow. 15, 4 (2021), 752–765. https://doi.org/10.14778/3503585.3503586

[11] Shohedul Hasan, Saravanan Thirumuruganathan, Jees Augustine, Nick Koudas, and Gautam Das. 2020. Deep Learning Models for Selectivity Estimation of Multi-Attribute Queries. 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, 1035–1050. https://doi.org/10.1145/3318464.3389741

[12] Rojeh Hayek and Oded Shmueli. 2020. Improved Cardinality Estimation by Learning Queries Containment Rates. In Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020, Angela Bonifati, Yongluan Zhou, Marcos Antonio Vaz Salles, Alexander Böhm, Dan Olteanu, George H. L. Fletcher, Arijit Khan, and Bin Yang (Eds.). OpenProceedings.org, 157–168. https://doi.org/10.5441/002/edbt.2020.15

[13] Axel Hertzschuch, Claudio Hartmann, Dirk Habich, and Wolfgang Lehner. 2021. Simplicity Done Right for Join Ordering. In 11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual Event, January 11-15, 2021, Online Proceedings. www.cidrdb.org. http://cidrdb.org/cidr2021/papers/cidr2021_paper01.pdf

[14] Benjamin Hilprecht and Carsten Binnig. 2021. One model to rule them all: towards zero-shot learning for databases. arXiv preprint arXiv:2105.00642 (2021).

[15] 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

[16] Zachary G Ives and Nicholas E Taylor. 2008. Sideways information passing for push-style query processing. (2008).

[17] Andreas Kipf, Michael Freitag, Dimitri Vorona, Peter Boncz, Thomas Neumann, and Alfons Kemper. 2019. Estimating filtered group-by queries is hard: Deep learning to the rescue. In 1st International Workshop on Applied AI for Database Systems and Applications.

[18] Andreas Kipf, Dimitri Vorona, Jonas Müller, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, Thomas Neumann, and Alfons Kemper. 2019. Estimating Cardinalities with Deep Sketches. 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, 1937–1940. https://doi.org/10.1145/3299869.3320218

[19] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. 2020. Captum: A unified and generic model interpretability library for pytorch. arXiv preprint arXiv:2009.07896 (2020).

[20] M. Seetha Lakshmi and Shaoyu Zhou. 1998. Selectivity Estimation in Extensible Databases - A Neural Network Approach. In VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA, Ashish Gupta, Oded Shmueli, and Jennifer Widom (Eds.). Morgan Kaufmann, 623–627. http://www.vldb.org/conf/1998/p623.pdf

[21] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? Proc. VLDB Endow. 9, 3 (2015), 204–215. https://doi.org/10.14778/2850583.2850594

[22] Beibin Li, Yao Lu, and Srikanth Kandula. 2022. Warper: Efficiently Adapting Learned Cardinality Estimators to Data and Workload Drifts. In Proceedings of the 2022 International Conference on Management of Data.

[23] Beibin Li, Yao Lu, Chi Wang, and Srikanth Kandula. 2021. Cardinality Estimation: Is Machine Learning a Silver Bullet. In 3rd International Workshop on Applied AI for Database Systems and Applications (AIDB).

[24] Jie Liu, Wenqian Dong, Qingqing Zhou, and Dong Li. 2021. Fauce: fast and accurate deep ensembles with uncertainty for cardinality estimation. Proceedings of the VLDB Endowment 14, 11 (2021), 1950–1963.

[25] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2022. Bao: Making learned query optimization practical. ACM SIGMOD Record 51, 1 (2022), 6–13.

[26] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A learned query optimizer. arXiv preprint arXiv:1904.03711 (2019).

[27] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. 2009. Preventing Bad Plans by Bounding the Impact of Cardinality Estimation Errors. Proc. VLDB Endow. 2, 1 (2009), 982–993. https://doi.org/10.14778/1687627.1687738

[28] Parimarjan Negi. 2022. Robust Query Driven Cardinality Estimation under Changing Workloads: Online Appendix. Retrieved 2022 from https://parimarjan.github.io/robust_cardinality_appendix.pdf [Online;].

[29] Parimarjan Negi, Ryan Marcus, Hongzi Mao, Nesime Tatbul, Tim Kraska, and Mohammad Alizadeh. 2020. Cost-Guided Cardinality Estimation: Focus Where it Matters. In 36th IEEE International Conference on Data Engineering Workshops, ICDE Workshops 2020, Dallas, TX, USA, April 20-24, 2020. IEEE, 154–157. https://doi.org/10.1109/ICDEW49219.2020.00034

[30] Parimarjan Negi, Ryan C. Marcus, Andreas Kipf, Hongzi Mao, Nesime Tatbul, Tim Kraska, and Mohammad Alizadeh. 2021. Flow-Loss: Learning Cardinality Estimates That Matter. Proc. VLDB Endow. 14, 11 (2021), 2019–2032. https://doi.org/10.14778/3476249.3476259

[31] Parimarjan Negi, Ryan C. Marcus, Andreas Kipf, Hongzi Mao, Nesime Tatbul, Tim Kraska, and Mohammad Alizadeh. 2021. Flow-Loss: Learning Cardinality Estimates That Matter. Proc. VLDB Endow. 14, 11 (2021), 2019–2032. https://doi.org/10.14778/3476249.3476259

[32] Andreas Kipf Hongzi Mao Nesime Tatbul Tim Kraska Mohammad Alizadeh Parimarjan Negi, Ryan Marcus. 2021. Cardinality Estimation Benchmark. https://github.com/learnedsystems/ceb [Online;].

[33] Yongjoo Park, Shucheng Zhong, and Barzan Mozafari. 2020. QuickSel: Quick Selectivity Learning with Mixture Models. 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, 1017–1033. https://doi.org/10.1145/3318464.3389727

[34] Suraj Shetiya, Saravanan Thirumuruganathan, Nick Koudas, and Gautam Das. 2020. Astrid: Accurate Selectivity Estimation for String Predicates using Deep Learning. Proc. VLDB Endow. 14, 4 (2020), 471–484. https://doi.org/10.14778/3436905.3436907

[35] Michael Stillger, Guy M. Lohman, Volker Markl, and Mokhtar Kandil. 2001. LEO - DB2's LEarning Optimizer. In VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy, Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard T. Snodgrass (Eds.). Morgan Kaufmann, 19–28. http://www.vldb.org/conf/2001/P019.pdf

[36] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In International conference on machine learning. PMLR, 3319–3328.

[37] Kostas Tzoumas, Amol Deshpande, and Christian S. Jensen. 2011. Lightweight Graphical Models for Selectivity Estimation Without Independence Assumptions. Proc. VLDB Endow. 4, 11 (2011), 852–863. http://www.vldb.org/pvldb/vol4/p852-tzoumas.pdf

[38] Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. 2021. Are We Ready For Learned Cardinality Estimation? Proc. VLDB Endow. 14, 9 (2021), 1640–1654. https://doi.org/10.14778/3461535.3461552

[39] Lucas Woltmann, Claudio Hartmann, Maik Thiele, Dirk Habich, and Wolfgang Lehner. 2019. Cardinality estimation with local deep learning models. 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, 5:1–5:8. https://doi.org/10.1145/3329859.3329875

[40] Peizhi Wu and Gao Cong. 2021. A Unified Deep Model of Learning from both Data and Queries for Cardinality Estimation. 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, 2009–2022. https://doi.org/10.1145/3448016.3452830

[41] Ziniu Wu, Amir Shaikhha, Rong Zhu, Kai Zeng, Yuxing Han, and Jingren Zhou. 2020. BayesCard: Revitilizing Bayesian Frameworks for Cardinality Estimation. arXiv preprint arXiv:2012.14743 (2020).

[42] 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

[43] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep Unsupervised Cardinality Estimation. Proc. VLDB Endow. 13, 3 (2019), 279–292. https://doi.org/10.14778/3368289.3368294

[44] Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, and Bin Cui. 2021. FLAT: Fast, Lightweight and Accurate Method for Cardinality Estimation. Proc. VLDB Endow. 14, 9 (2021), 1489–1502. https://doi.org/10.14778/3461535.3461539