MLSkip: Data Skipping for ML Filters via Lightweight Metadata

Mihail Stoian*
University of Technology Nuremberg
Nuremberg, Germany
mihail.stoian@utn.de

Mark Gerarts*
Hasselt University
Hasselt, Belgium
mark.gerarts@uhasselt.be

Pascal Ginter
Technical University of Munich
Munich, Germany
pascal.ginter@tum.de

Andreas Zimmerer
University of Technology Nuremberg
Nuremberg, Germany
andreas.zimmerer@utn.de

Jan Van den Bussche
Hasselt University
Hasselt, Belgium
jan.vandenbussche@uhasselt.be

Andreas Kipf
University of Technology Nuremberg
Nuremberg, Germany
andreas.kipf@utn.de

ABSTRACT

Database vendors recently released AI functions that can be used in filter predicates. As such functions often rely on costly, black-box ML models, they unveil new data management challenges. Concretely, traditional data skipping techniques for integer and string data fail to be applicable to the new filter type. Indeed, there is no known mechanism for pruning non-qualifying row groups, e.g., when reading files from blob storage.

In this work, we initiate the study of data skipping techniques for ML filters. We make the case that Parquet's default min-max metadata is enough to enable pruning. To this end, we draw connections to two lines of research: (i) the recently proposed query language for ML models and (ii) neural network verification.

Our preliminary results on ReLU architectures show that on tables from TPC-H and TPC-DS, the average pruning effectiveness for filters of selectivity below 0.1% amounts to 27.4%. Finally, inspired by research on spatial joins, we propose an enhanced metadata structure: a size-bounded 2D convex hull that verification tools can make better use of, increasing the pruning effectiveness to 38.31%, while occupying at most 45 bytes per row group and column pair. We observe an end-to-end speedup of 1.07× over PyTorch in DuckDB.

VLDB Workshop Reference Format:
Mihail Stoian, Mark Gerarts, Pascal Ginter, Andreas Zimmerer, Jan Van den Bussche, and Andreas Kipf. MLSkip: Data Skipping for ML Filters via Lightweight Metadata. VLDB 2026 Workshop: Applied AI for Database Systems and Applications (AIDB 2026).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/mlskip/mlskip.

1 INTRODUCTION

Support for executing ML models over relational data has become commonplace in modern database systems; examples include Google BigQuery's `ML.PREDICT` [15], Databricks' `ai_query` [10], Microsoft T-SQL's `PREDICT` [29], Snowflake's Model Registry [37], Amazon Redshift ML [3], Oracle's `OML4SQL` [30], and many others.

ML Filters. Indeed, integrating ML inference into the query engine increases flexibility and encourages customers to keep both their data and ML workloads within the database system [22]. The following example shows how a neural network, `sanity_score_nn`, can be used as a scoring function inside a table scan:[1]

```
SELECT city_name FROM cities WHERE
  sanity_score_nn(population_density, housing_cost) > 0.9;
```

As the complexity of the filter increases, the query engine loses the ability to effectively prune non-qualifying row groups, a capability that major vendors optimize for traditional filters [47].

Research Question. Unlike traditional equality and range table predicates, ML filters are difficult for query engines to reason about. Traditional predicates benefit from metadata-aware file formats such as Parquet, which maintain min-max statistics that enable data skipping. For example, a predicate `housing_cost > 3k` can exclude a row group with (`min`, `max`) = (1k, 2k). Given the recent adoption of ML filters, it is natural to ask whether it is possible to enable data skipping by leveraging the built-in metadata, available on both managed and open file formats.

⏩ **MLSkip.** Somewhat surprisingly, we show in this work that, *even with the already available min-max metadata*, one can obtain measurable pruning effects for standard ML models such as ReLU architectures. To this end, we draw connections to two lines of research, namely: (i) the recently proposed query language for ML models [16], and (ii) the established line of research on neural network verification, featuring tools such as Marabou [24] and α, β-Crown [26]. Our contributions are as follows:

(1) We introduce MLSkip, the first framework for row group pruning for ML filters.
(2) We evaluate the effectiveness of the two approaches with min-max metadata on the TPC-H [38] and TPC-DS [39] benchmarks (sf = 1), using ReLU architectures trained on 2-4 features (columns), and obtain an average pruning effectiveness of 27.4% for filters of $\leq 0.1\%$-selectivity.[2]
(3) We propose enhanced, geometry-aware metadata that boosts the pruning effectiveness to 38.31%, reaching an end-to-end speedup of 1.07× over PyTorch in DuckDB across all filters.

*The author contributed equally to this work.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

[1] The model evaluates the "sanity score" of a city; see Ref. [9] for a formal analysis.
[2] pruning effectiveness [%] = `#pruned_row_groups` / `#prunable_row_groups`.

Related Work. Integrating ML inference in a query engine is a rather recent idea, with origins in many proposals in the last decade [17, 21, 22]. After the arrival of LLMs, the simple ML operators have been replaced by the more flexible semantic operators [1, 7, 11, 19, 28, 32, 36]. Notably, recent work on optimized semantic filters showed that traditional ML models, such as logistic regression, can replace expensive LLMs for query processing for certain cases [7]. The idea is to use some initial sample of the table—which is processed with an expensive LLM—as training data for a proxy model over pre-computed row-embeddings (in this case a logistic regression model, but other models might be used as well). After the proxy model has been trained, its accuracy is compared against the LLM and, if deemed good enough, execution is continued with the proxy model instead of the LLM. We can imagine that MLSkip can be used for such optimizations of semantic filters via proxy models to further speed up query processing by allowing to skip evaluation of entire row groups.

Research in the context of traditional ML-native engines includes Raven [22], Smart [17], Umbra's PyTorch extension [34], and LingoDB's MLIR-empowered model compilation [20]. However, despite much research on this topic, we are unaware of any work on data skipping techniques for this new filter type.

2 PRELIMINARIES

In this section, we briefly present file format metadata such as that of Parquet's §2.1, neural network verification §2.2, and the recent paradigm of query languages for ML models §2.3.

2.1 File Format Metadata

The quest for data skipping led modern file formats like Parquet to integrate metadata column statistics, including the min-max values of the column, the number of `NULL` values, and the distinct count [4]; these are missing in traditional file formats like CSV. Given Parquet's widespread adoption, query engines highly optimized for this (limited) metadata structure [47]. Noteworthy, Parquet supports adding custom metadata, such as our enhanced 2D metadata (§3.1), while preserving the readability compatibility [46].

2.2 Neural Network Verification

Neural networks (NNs) are indispensable tools in today's software landscape. When used in mission-critical systems, one must have guarantees on how the networks operate. The field of neural network verification (NNV) develops tools to formally verify properties such as robustness, consistency, and reachability [2, 27]. We focus on the latter: given a neural network and some input constraints, we want to verify if the network's output maps to a given value.

There are several tools to verify NNs. In this work, we consider Marabou [24, 44], due to its easy setup and fast verification time. The state-of-the-art tool is α, β-Crown [25, 26], which scales to larger models by sacrificing completeness for efficiency [42],[3] and has a GPU-optimized implementation, making it compelling for the rise of GPU-enabled databases [18, 43].

[3]That is, a property proven correct by α, β-Crown is always correct, but a property that α, β-Crown fails to verify might still be correct in reality. In the context of data skipping, this is acceptable: we may read some row groups that could have been pruned, but never miss any that are required.

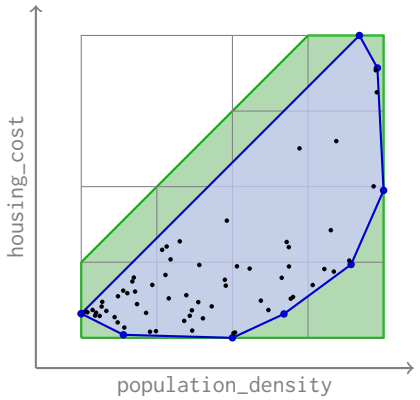


Figure 1: Row group metadata variants: (i) vanilla min-max ranges induce a 2D rectangle, (ii) `ConvexHull` creates the optimal convex polygon around the row group's data points, while (iii) `BoundedConvexHull` bounds the number of vertices in the hull, by working on a grid of depth 2.

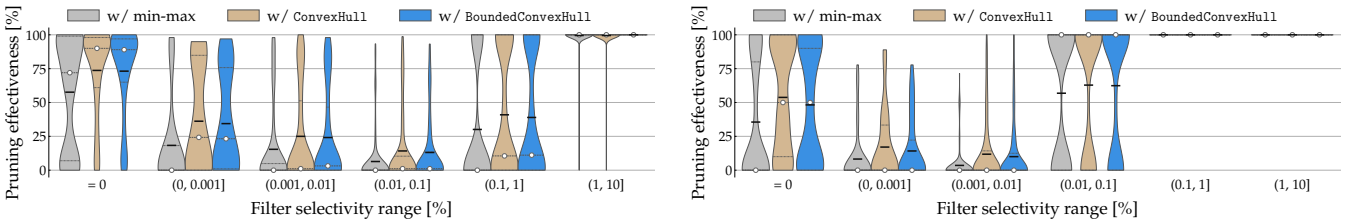
2.3 Query Languages for ML Models

Recent work has shown that SQL can act as a query language for neural networks [16]. By representing a network's structure and weights as a database instance, SQL can reason about the network and verify its behavior. We implemented this paradigm as a two-step approach, which we call ML-QL. First, ML-QL compiles the network to a geometric representation and stores it in a database. We then use this precomputed representation to execute many verification queries in SQL relatively cheaply, unlike traditional tools that run entirely at query time [23].

We consider feed-forward networks with ReLU activation functions and a single output node. Such networks represent piecewise linear functions. Geometrically, the ReLU layers partition the continuous input space into a hyperplane arrangement: a collection of polyhedral regions where the network behaves as an affine function within each distinct region.

Prior work reconstructs the output functions of networks with a single input node and a single hidden layer [14]; we generalize this approach to networks with an arbitrary number of inputs and layers. Our compilation step is based on Cylindrical Algebraic Decomposition (CAD), mirroring the proof of the theoretical foundations [16]. However, CAD imposes restrictions on the size of networks we can practically verify because it scales doubly exponential [8] in the number of input features of the network, and the number of inputs for the algorithm grows exponentially in the number of hidden layers due to ReLU activation functions. Future work is needed to replace CAD with more efficient algorithms.

We use the geometric representation to answer reachability queries: determining whether the network's output can fall within a specific range given a set of input constraints. While many alternative techniques exist for this particular problem [41, 45], viewing the problem through a database lens provides a distinct advantage: it shifts the heavy lifting entirely to the offline phase, allowing the online verification phase to benefit directly from database query optimization, e.g., we can batch all table metadata in the same



(a) Row group size = 1K

(b) Row group size = 10K

Figure 2: Data skipping behavior for varying filter selectivity under two row group sizes $\in \{1K, 10K\}$ and the metadata types. The models are neural networks with 2 hidden layers trained on 2-4 table columns of the TPC-H and TPC-DS benchmarks.

SQL query. The verification queries currently use approximation techniques and are thus incomplete, just like α, β-Crown. This is a practical constraint; in future work, we plan to use exact techniques.

To compare against other verifiers, we evaluate ML-QL as a standalone tool. However, we envision that a sufficiently advanced query planner might use the geometric representation directly to optimize data skipping. This remains a topic for future work.

3 MLSKIP

MLSkip is the first framework to enable data skipping for ML filters, using lightweight metadata, such as Parquet's. MLSkip works as follows. Consider the query used in the introduction. Assuming already available min-max metadata for each of the two columns, we specify the input space as the rectangle spanned by the min-max ranges, namely: $[\texttt{min1}, \texttt{max1}] \times [\texttt{min2}, \texttt{max2}]$.[4] Next, we specify the input constraints, along with the output constraint $(0.9, +\infty)$ from the filter, i.e., `sanity_score_nn(...) > 0.9`, and obtain the result whether the NN's function, on that input, takes values in that output constraints. If the result is negative, we can safely skip the respective row group.

In this work, our goal is to study both (i) the verification tools (NNV, ML-QL) and (ii) the metadata required to enable the skipping. Next, we detail on the latter point.

3.1 Enhanced Metadata

Observing the poor, yet non-zero pruning performance in the presence of only min-max metadata, we propose in the following a more enhanced metadata design, inspired from research on spatial joins [5] and linear optimization queries [6]. We gradually build towards a compact metadata, that, interestingly, fits well with the input structure expected by the verifiers. We first consider the case of a 2-feature model, as per our sample model `sanity_score_nn`. The following ideas are visualized in Fig. 1.

`ConvexHull`. The first idea is to build a convex hull of the two-dimensional data points. The intuition is that regions around the corners induced by the 2D min-max ranges are unoccupied, leading to a smaller input domain. However, the disadvantage of vanilla convex hulls is that we cannot bound the number of stored vertices, leading to unpredictable metadata size (and verification time).

`BoundedConvexHull`. The second idea, which does guarantee bounded metadata size, is to instead recursively split the 2D rectangle induced by the min-max ranges into a grid of given depth; unoccupied grid cells are not split. As performing verification on the grid cells would be particularly expensive,[5] we instead fit a convex hull on top of the occupied cells. This comes with two clear advantages: (i) the verification becomes single-pass, as we simply constraint the input with the linear constraints of the convex hull,[6] and (ii) the metadata size is bounded, as the number of hull vertices is, by grid's construction, bounded. We store a $2d + 2$ bit per vertex, capturing the cell's index ($2 \times d$ bit) and the involved corner (2 bit).

Integration. Our 2D metadata can also be used when using d-dimensional models, without requiring the presence of metadata on more than two dimensions. Namely, we will constraint the input space via all $\binom{d}{2}$ (bounded) convex hulls. Currently, we build the 2D metadata on all column pairs. In a production setting, one should analyze the workload to understand which pairs are useful for pruning so that the build overhead pays off.

4 EVALUATION

Setup. To train the NNs, we use the first 2k table rows of the TPC-H and TPC-DS benchmarks (sf = 1). The functions are learning a column using 2-4 other table columns; in total, we have 10 regression function templates, proposed by OpenAI's GPT5.5 model: 2 on TPC-H and 8 on TPC-DS. The trained NNs have either 1 or 2 hidden layers, each with 32 nodes; hence, the largest neural network has 1317 parameters. We generate the filters with varying selectivity, by gradually increasing the width of the queried filter range and picking a random range left bound, following the structure:

```
where model(col1, ..., coln) between a and b;
```

We obtain a total of 1376 such filters of varying selectivity. The `BoundedConvexHull` metadata comes with a default grid depth of 4. We conduct the experiments on a single node Intel® Xeon® Gold 5318Y CPU (24 cores, 48 hyper-threads). The machine is equipped with 128GB DDR4 main memory and runs Ubuntu 24.04.

Tools. We use Marabou v2.0 [44] and our implementation of ML-QL for neural networks for multiple inputs (§2.3). ML-QL is currently limited to models with a single hidden layer. Due to its slow model

[4] In our example: `min1 := select min(population_density) from cities`.

[5] Indeed, one would have to perform a verification call for each occupied grid cell.

[6] This property also holds for min-max metadata, allowing for fast verification time.

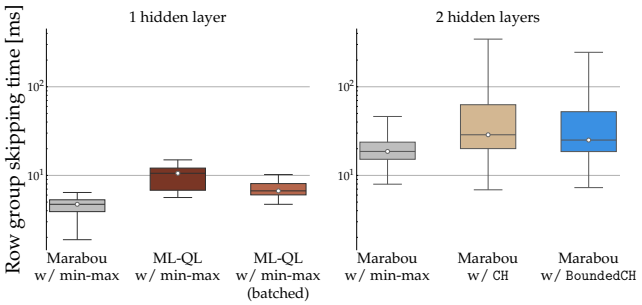


Figure 3: Data skipping time for models on TPC-H and TPC-DS with a row group size of 1K.

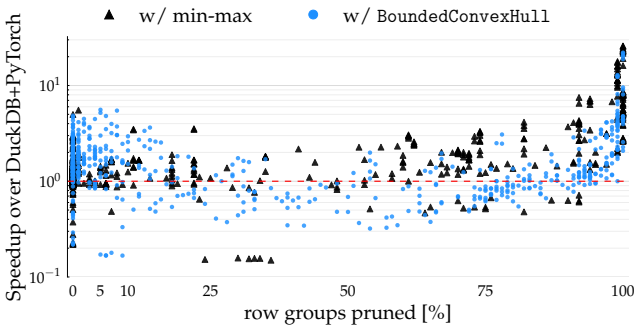


Figure 4: End-to-end speedup on DuckDB with PyTorch across all benchmark filters, using a row group size of 1K.

compilation phase, we only benchmark it on two-dimensional models; it takes around 0.8s to compile the model with CAD, stored then in 3 tables, the largest having 32k rows and 7 columns.

4.1 Pruning Effectiveness

We report the *pruning effectiveness* of Marabou for all trained ML filters, computed as:

$$\frac{\texttt{\#pruned\_row\_groups}}{\texttt{\#prunable\_row\_groups}} \cdot 100.$$

In Fig. 2, we report the pruning effectiveness achieved across both benchmarks for row groups of 1K and 10K rows, clustering the filters in selectivity buckets. (Note that highly non-selective filters will also have a high effectiveness by default.) We observe that for zero-selectivity filters, the pruning is quite effective, especially when enabling the enhanced metadata (§3.1). For highly selective filters ($\leq 0.1\%$ selectivity), with min-max metadata, we obtain an average effectiveness of 27.4%. Notably, `BoundedConvexHull` metadata improves this number to 38.31%, while `ConvexHull` gets to 39.3%. Naturally, when running on groups consisting of 10K rows, the pruning effect is reduced on the same selectivity buckets, as the information captured by the metadata becomes coarser.

E2E Speedup. Being able to prune row groups makes model inference cheaper. Indeed, when running PyTorch 2.8.0+cpu [31] in DuckDB 1.5 [33], we observe a total end-to-end speedup of 1.07× across *all* filters, when using models with 2 hidden layers. We plot in Fig. 4 the relationship between the pruning ratio and the speedup. Where MLSkip is unable to prune a substantial fraction of row groups, e.g., $\leq 5\%$, the (noisy) overhead of verification tends to dominate.

4.2 Pruning Time

We report the time to prune a row group via both Marabou and ML-QL. We differentiate between using models with 1 and 2 hidden layers, respectively. First, ML-QL is around 2x slower than Marabou on models with one hidden layer, with our batched variant narrowing this gap. Second, the time increases by almost one magnitude for Marabou with min-max metadata on models with two hidden layers. Noteworthy, the effect of having a bounded number of vertices in the convex hull can be observed for `BoundedConvexHull`, inducing a faster verification time than with `ConvexHull`.

4.3 Metadata Overhead

While building min-max metadata is rather cheap, our enhanced metadata requires more compute (and more space). We show this overhead in terms of both compute and space in Tab. 1, for table column pairs across 100 row groups. Naturally, min-max metadata requires $2 \times (8 + 8) = 32$ byte to store a pair of doubles per column; the build time is that of a DuckDB table scan.

The regular `ConvexHull` requires two doubles for each convex hull vertex. Hence, without having control on the number of vertices, we observe that the overhead can get up to 304 byte, making this metadata type rather impractical; note that the lightweight HyperLogLog sketch [12] is usually optimized to occupy 64 byte [13]. Our `BoundedConvexHull` achieves this goal, by fitting the convex hull on a grid support. In this case, we empirically observed a maximum size of 45 byte (for a grid depth of 4). Note that the space footprint can be reduced by choosing a smaller grid depth. Both metadata types are built via `scipy.spatial.ConvexHull` [35]. To support a d-dimensional input, the metadata has to be built on all $\binom{d}{2}$ pairs; see §3.1 for a discussion on how to optimize this step.

Table 1: Metadata space footprint and build time per row group for column pairs; the default grid depth is 4.

Metadata type	Size [byte]		Build time [ms]	
	avg	max	avg	max
min–max	32.00	32.00	12.09	20.09
`ConvexHull`	161.12	304.00	31.73	41.22
`BoundedConvexHull`	40.51	45.00	31.74	41.59

5 CONCLUSION & FUTURE WORK

With ⏩ MLSkip, we showed that it is indeed possible to enable pruning for the rather complex ML filters, with focus on ReLU architectures, even when using only lightweight metadata, while richer metadata can further improve pruning. Given the rise of semantic query processing and its use of proxy models [7], MLSkip is a necessary tool to enable faster query times and save I/O from blob storage. We plan to test MLSkip with larger models, e.g., Transformers [40], study verification on string data, and integrate a Batch API into current neural network verification tools.

REFERENCES

[1] Paritosh Aggarwal, Bowei Chen, Anupam Datta, Benjamin Han, Boxin Jiang, Nitish Jindal, Zihan Li, Aaron Lin, Pawel Liskowski, Jay Tayade, et al. 2025. Cortex AISQL: A Production SQL Engine for Unstructured Data. *arXiv preprint arXiv:2511.07663* (2025).

[2] Aws Albarghouthi. 2021. *Introduction to Neural Network Verification.* verifieddeeplearning.com. arXiv:2109.10317 [cs.LG] http://verifieddeeplearning.com.

[3] Amazon Web Services. 2026. *Amazon Redshift ML.* https://docs.aws.amazon.com/redshift/latest/dg/machine_learning.html

[4] Apache Parquet. 2026. *Metadata.* https://parquet.apache.org/docs/file-format/metadata/

[5] Thomas Brinkhoff, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1994. Multi-Step Processing of Spatial Joins. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, USA, May 24-27, 1994*, Richard T. Snodgrass and Marianne Winslett (Eds.). ACM Press, 197–208. https://doi.org/10.1145/191839.191880

[6] Yuan-Chi Chang, Lawrence D. Bergman, Vittorio Castelli, Chung-Sheng Li, Ming-Ling Lo, and John R. Smith. 2000. The Onion Technique: Indexing for Linear Optimization Queries. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein (Eds.). ACM, 391–402. https://doi.org/10.1145/342009.335433

[7] Yeounoh Chung, Rushabh Desai, Jian He, Yu Xiao, Thibaud Hottelier, Yves-Laurent Kom Samo, Pushkar Kadilkar, Xianshun Chen, Sam Idicula, Fatma Özcan, et al. 2026. 100x Cost & Latency Reduction: Performance Analysis of AI Query Approximation using Lightweight Proxy Models. *arXiv preprint arXiv:2603.15970* (2026).

[8] G.E. Collins. 1975. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. *Lecture Notes in Computer Science* 33 (1975), 134–183.

[9] Wendy Cox. 2025. *Demographia International Housing Affordability.* Technical Report. Frontier Centre for Public Policy, Canada. https://policycommons.net/artifacts/21033541/demographia-international-housing/21933951/ Retrieved from https://coilink.org/20.500.12592/3hb8vjr on May 31, 2026. COI: 20.500.12592/3hb8vjr.

[10] Databricks. 2026. `ai_query` *Function.* https://docs.databricks.com/aws/en/sql/language-manual/functions/ai_query

[11] Anas Dorbani, Sunny Yasser, Jimmy Lin, and Amine Mhedhbi. 2025. Beyond Quacking: Deep Integration of Language Models and RAG into DuckDB. *Proc. VLDB Endow.* 18, 12 (Sept. 2025), 5415–5418. https://doi.org/10.14778/3750601.3750685

[12] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete mathematics & theoretical computer science* Proceedings (2007).

[13] Michael J. Freitag and Thomas Neumann. 2019. Every Row Counts: Combining Sketches and Sampling for Accurate Group-By Result Estimates. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*. www.cidrdb.org. https://vldb.org/cidrdb/2019/every-row-counts-combining-sketches-and-sampling-for-accurate-group-by-result-estimates.html

[14] Mark Gerarts, Juno Steegmans, and Jan Van den Bussche. 2025. SQL4NN: Validation and expressive querying of models as data. In *Proceedings of the Workshop on Data Management for End-to-End Machine Learning*. 1–5.

[15] Google Cloud. 2026. ML.PREDICT Function. https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-predict. Google Cloud Documentation, accessed 2026-05-31.

[16] Martin Grohe, Christoph Standke, Juno Steegmans, and Jan Van den Bussche. 2025. Query Languages for Neural Networks. In *28th International Conference on Database Theory, ICDT 2025, March 25–28, 2025, Barcelona, Spain (LIPIcs)*, Sudeepa Roy and Ahmet Kara (Eds.), Vol. 328. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 9:1–9:18. https://doi.org/10.4230/LIPICS.ICDT.2025.9

[17] Yunyan Guo, Guoliang Li, Ruilin Hu, and Yong Wang. 2025. In-database query optimization on SQL with ML predicates. *VLDB J.* 34, 1 (2025), 12.

[18] Dong He, Supun Chathuranga Nakandala, Dalitso Banda, Rathijit Sen, Karla Saur, Kwanghyun Park, Carlo Curino, Jesús Camacho-Rodríguez, Konstantinos Karanasos, and Matteo Interlandi. 2022. Query Processing on Tensor Computation Runtimes. *Proc. VLDB Endow.* 15, 11 (2022), 2811–2825. https://doi.org/10.14778/3551793.3551833

[19] Saehan Jo and Immanuel Trummer. 2024. ThalamusDB: Approximate Query Processing on Multi-Modal Data. *Proc. ACM Manag. Data* 2, 3 (2024), 186. https://doi.org/10.1145/3654989

[20] Michael Jungmair, André Kohn, and Jana Giceva. 2022. Designing an Open Framework for Query Optimization and Compilation. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2389–2401.

[21] Gaurav Tarlok Kakkar, Jiashen Cao, Aubhro Sengupta, Joy Arulraj, and Hyesoon Kim. 2025. Aero: Adaptive Query Processing of ML Queries. *Proc. ACM Manag. Data* 3, 3 (2025), 174:1–174:27.

[22] Konstantinos Karanasos, Matteo Interlandi, Doris Xin, Fotis Psallidas, Rathijit Sen, Kwanghyun Park, Ivan Popivanov, Supun Nakandala, Subru Krishnan, Markus Weimer, Yuan Yu, Raghu Ramakrishnan, and Carlo Curino. 2019. Extending Relational Query Processing with ML Inference. *CoRR* abs/1911.00231 (2019).

[23] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24–28, 2017, Proceedings, Part I (Lecture Notes in Computer Science)*, Rupak Majumdar and Viktor Kuncak (Eds.), Vol. 10426. Springer, 97–117. https://doi.org/10.1007/978-3-319-63387-9_5

[24] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljic, David L. Dill, Mykel J. Kochenderfer, and Clark W. Barrett. 2019. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15–18, 2019, Proceedings, Part I (Lecture Notes in Computer Science)*, Isil Dillig and Serdar Tasiran (Eds.), Vol. 11561. Springer, 443–452. https://doi.org/10.1007/978-3-030-25540-4_26

[25] Konstantin Kaulen, Tobias Ladner, Stanley Bak, Christopher Brix, Hai Duong, Thomas Flinkow, Taylor T. Johnson, Lukas Koller, Edoardo Manino, ThanhVu H. Nguyen, and Haoze Wu. 2025. The 6th International Verification of Neural Networks Competition (VNN-COMP 2025): Summary and Results. *CoRR* abs/2512.19007 (2025). https://doi.org/10.48550/ARXIV.2512.19007 arXiv:2512.19007

[26] Suhas Kotha, Christopher Brix, J. Zico Kolter, Krishnamurthy Dvijotham, and Huan Zhang. 2023. Provably Bounding Neural Network Preimages. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/fe061ec0ae03c5cf5b5323a2b9121bfd-Abstract-Conference.html

[27] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher A. Strong, Clark W. Barrett, and Mykel J. Kochenderfer. 2021. Algorithms for Verifying Deep Neural Networks. *Found. Trends Optim.* 4, 3-4 (2021), 244–404. https://doi.org/10.1561/2400000035

[28] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, et al. 2025. Palimpzest: Optimizing ai-powered analytics with declarative query processing. In *Proceedings of the Conference on Innovative Database Research (CIDR)*. 2.

[29] Microsoft. 2026. *PREDICT (Transact-SQL) - SQL Machine Learning.* https://learn.microsoft.com/en-us/sql/t-sql/queries/predict-transact-sql?view=sql-server-ver17

[30] Oracle. 2026. *Oracle Machine Learning for SQL (OML4SQL).* https://docs.oracle.com/en/database/oracle/machine-learning/oml4sql/index.html

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[32] Liana Patel, Siddharth Jha, Carlos Guestrin, and Matei Zaharia. 2024. LOTUS: Enabling Semantic Queries with LLMs Over Tables of Unstructured and Structured Data. *CoRR* abs/2407.11418 (2024). https://doi.org/10.48550/ARXIV.2407.11418 arXiv:2407.11418

[33] Mark Raasveldt and Hannes Mühleisen. 2019. DuckDB: An Embeddable Analytical Database. In *Proceedings of the 2019 international conference on management of data*. 1981–1984.

[34] Maximilian Rieger, Moritz Sichert, and Thomas Neumann. 2022. Integrating deep learning frameworks into main-memory databases. In *Proceedings of the VLDB 2022 Applied AI for Database Systems and Applications Workshop co-located with (VLDB 2022)(AIDB Workshop Proceedings). https://drive. google. com/file/d/1GfZH3Y1sQKgplnnpTEM_E4skWdhmyrfe/edit.*

[35] SciPy Developers. 2025. `scipy.spatial.ConvexHull`. https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.ConvexHull.html. Accessed: 2026-05-31.

[36] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2025. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. *Proc. VLDB Endow.* 18, 9 (Sept. 2025), 3035–3048. https://doi.org/10.14778/3746405.3746426

[37] Snowflake. 2026. *Snowflake Model Registry.* https://docs.snowflake.com/en/developer-guide/snowflake-ml/model-registry/overview

[38] Transaction Processing Performance Council. [n.d.]. TPC Benchmark H (Decision Support) Standard Specification. https://www.tpc.org/tpch/. Version 3.0.1.

[39] Transaction Processing Performance Council. [n.d.]. TPC-DS Benchmark Standard Specification. https://www.tpc.org/tpcds/. Version 4.0.0.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[41] Joseph A. Vincent and Mac Schwager. 2025. Reachable Polyhedral Marching (RPM): An Exact Analysis Tool for Deep-Learned Control Systems. *IEEE Trans. Neural Networks Learn. Syst.* 36, 10 (2025), 19225–19239. https://doi.org/10.1109/TNNLS.2025.3571720

[42] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. 2021. Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Neural Network Robustness Verification. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 29909–29921. https://proceedings.neurips.cc/paper/2021/hash/fac7fead96dafceaf80c1daffeae82a4-Abstract.html

[43] Bowen Wu, Wei Cui, Carlo Curino, Matteo Interlandi, and Rathijit Sen. 2025. Terabyte-Scale Analytics in the Blink of an Eye. *Proc. VLDB Endow.* 19, 2 (2025), 141–155. https://www.vldb.org/pvldb/vol19/p141-sen.pdf

[44] Haoze Wu, Omri Isac, Aleksandar Zeljic, Teruhiro Tagomori, Matthew L. Daggitt, Wen Kokke, Idan Refaeli, Guy Amir, Kyle Julian, Shahaf Bassan, Pei Huang, Ori Lahav, Min Wu, Min Zhang, Ekaterina Komendantskaya, Guy Katz, and Clark W. Barrett. 2024. Marabou 2.0: A Versatile Formal Analyzer of Neural Networks. In *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part II (Lecture Notes in Computer Science)*, Arie Gurfinkel and Vijay Ganesh (Eds.). Springer, 249–264. https://doi.org/10.1007/978-3-031-65630-9_13

[45] Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson. 2017. Reachable Set Computation and Safety Verification for Neural Networks with ReLU Activations. *CoRR* abs/1712.08163 (2017). arXiv:1712.08163 http://arxiv.org/abs/1712.08163

[46] Qi Zhu, Jigao Luo, and Andrew Lamb. [n.d.]. *Embedding User-Defined Indexes in Apache Parquet Files.* https://datafusion.apache.org/blog/2025/07/14/user-defined-parquet-indexes/ Apache DataFusion Blog.

[47] Andreas Zimmerer, Damien Dam, Jan Kossmann, Juliane Waack, Ismail Oukid, and Andreas Kipf. 2025. Pruning in Snowflake: Working Smarter, Not Harder. In *Companion of the 2025 International Conference on Management of Data, SIGMOD/PODS 2025, Berlin, Germany, June 22-27, 2025*, Volker Markl, Joseph M. Hellerstein, and Azza Abouzied (Eds.). ACM, 757–770. https://doi.org/10.1145/3722212.3724447