Journal of Machine Learning Research 24 (2023) 1-9 Submitted 9/22; Published 1/23

SQLFlow: An Extensible Toolkit Integrating DB and AI

Jun Zhou[1,2] JUN.ZHOUJUN@ANTGROUP.COM
Ke Zhang[2] YINGZI.ZK@ANTGROUP.COM
Lin Wang[2] FRED.WL@ANTGROUP.COM
Hua Wu[2] WUHUA.WH@ANTGROUP.COM
Yi Wang[2] YI.W@ANTGROUP.COM
ChaoChao Chen[1,*] ZJUCCC@ZJU.EDU.CN
[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[2]Ant Group, Beijing, China
[*]Corresponding Author

Editor: Sebastian Schelter

Abstract

Integrating AI algorithms into databases is an ongoing effort in both academia and industry. We introduce SQLFlow, a toolkit seamlessly combining data manipulations and AI operations that can be run locally or remotely. SQLFlow extends SQL syntax to support typical AI tasks including model training, inference, interpretation, and mathematical optimization. It is compatible with a variety of database management systems (DBMS) and AI engines, including MySQL, TiDB, MaxCompute, and Hive, as well as TensorFlow, scikit-learn, and XGBoost. Documentations and case studies are available at `https://sqlflow.org`. The source code and additional details can be found at `https://github.com/sql-machine-learning/sqlflow`.

Keywords: Machine Learning, AI Tasks, Model Training

1. Introduction

Nowadays, typical data science workflows consist of separate steps of processing data in a DBMS, exporting the processed data, and developing models in an Artificial Intelligence (AI) platform. This de facto standard workflow has several shortcomings: the data transfer incurs overhead and makes it hard to trace data, which is essential for compliance; development is slowed by imperative details in languages such as R and Python. From a practitioner point of view, the ideal system should efficiently execute AI algorithms in-place, be user-friendly and preserve the usability of exiting algorithms. And the integration of DBMS and AI algorithms provides the solution to meet the desired attributes(Makrynioti and Vassalos, 2019; Zhou et al., 2020; Villarroya and Baumann, 2020).

The integration of DBMS and AI algorithms could be realized by utilizing or extending DBMS to support AI algorithms. This could be done by direct SQL implementation (Ordonez, 2004; Ordonez and Pitchaimalai, 2009; Nagarjuna and Reddy, 2012; Marten and Heuer, 2017; Du, 2020; Giesser et al., 2021; Kaufmann et al., 2021; Blacher et al., 2022) or by extending DBMS (Luo et al., 2018; Karanasos et al., 2019; Jankov et al., 2020; Schüle

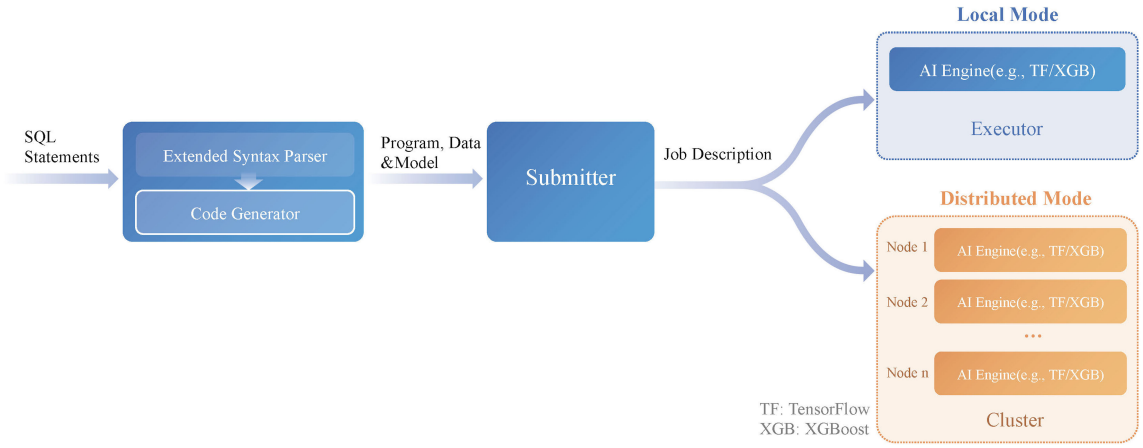


Figure 1: Basic Components and Workflows of SQLFlow

et al., 2019; Dolmatova et al., 2020; Schüle et al., 2022). The progresses on implementation abstraction(Yuan et al., 2020), programming model(Jasny et al., 2020), and execution engine(Schleich et al., 2019) could significantly outperform existing AI frameworks. In addition, typical AI workloads such as model selections(Zhang et al., 2021) and even the whole AI pipeline(Schule et al., 2021) could be expressed and executed in SQL. This approach is user-friendly, performs AI tasks in the DBMS which avoids the data transfer and makes it easy to meet compliance requirement. Nevertheless, it is hard to reuse existing AI tools such as TensorFlow(Abadi et al., 2016), scikit-learn, and XGBoost(Chen and Guestrin, 2016).

An alternative approach for the integration of DBMS and AI algorithms is implementing AI algorithms in DBMS, mostly via user-defined functions (UDFs)(Hirn and Grust, 2021). Apache MADlib (Hellerstein et al., 2012) is the representative open-source example of this approach. Oracle Machine Learning(Oracle, 2022), Microsoft SQL Server AI Services(Microsoft, 2022), and Google BigQuery(Google, 2022) are notable commercial platforms adopting this approach. While this approach performs data processing in the DBMS and preserves the usability of existing AI tools, it is not as user-friendly since users still need to handle low-level details in UDFs.

Works above are contrasting paradigms of SQLFlow and similar platforms such as IntegratedAI(De Boe et al., 2020), Vertica-AI(Fard et al., 2020), Amazon RedShift AI(Amazon, 2022) to integrate DBMS and AI algorithms and each paradigm represents different tradeoffs(Kumar et al., 2017). SQLFlow is also very user-friendly and preserves the usability of existing AI tools, at the cost of transferring data from DBMS to external platforms. While data transfer is needed in SQLFlow, SQLFlow could support data tracing since it builds an end-to-end workflow and is able to manage data throughout the workflow.

2. Overview of SQLFlow

SQLFlow is developed so practitioners could build end-to-end AI pipelines with just a few lines of SQL code. To this end, SQLFlow extends SQL syntax to support AI tasks including training, prediction, model evaluation, model explanation, custom jobs, and mathematical

programming. It also offers data processing and feature engineering capabilities aside from AI functionalities. SQLFlow supports various database systems like MySQL, MariaDB, TiDB(Huang et al., 2020), Hive, MaxCompute(Alibaba, 2023) and many machine learning toolkits like TensorFlow, Keras, XGBoost. SQLFlow has been adopted in practical applications by DiDi(Wang et al., 2020), Alibaba, and Ant Group over the years. For example, Xie et al. (2019) developed a SQLFlow-based deep neural network (DNN) classification model and applied it to taxi coupon distribution.

Key Design Considerations SQLFlow adheres to the following design principles: 1) Rather than being limited to specific SQL engines, it should be compatible with a wide range of DBMS. 2) Rather than only prototyping, it should democratize ML by letting SQL practitioners build models using existing AI tools and SQL skills. 3) Rather than embedding Python or R code in SQL statements, it should allow users to construct and execute ML workflows using SQL queries. 4) Rather than being imperative and expressing how the computation is to be achieved, it should use the declarative paradigm to describe what the outcome of the computation will be.

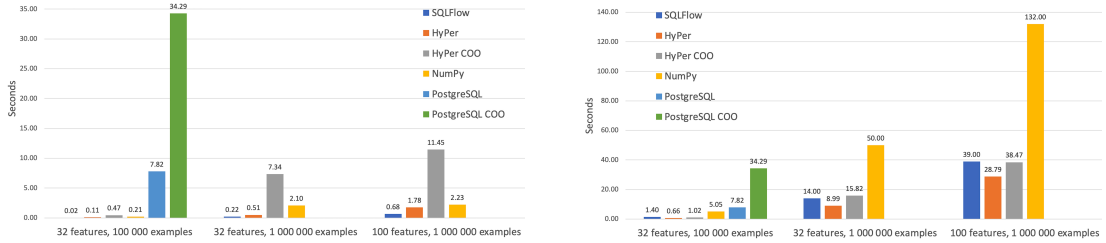
SQLFlow Architecture SQLFlow turns lines of SQL code into a complete AI pipeline by invoking *Extended Syntax Parse (ESP)*, *Code Generator (CG)* and *Submitter*, shown in Figure 1. First, the ESP decodes the input SQL program using a collaborative parsing algorithm. For standard SQL statements, ESP forwards them to corresponding DB engines. For extended SQL statement supporting AI functionalities, such as TRAIN, PREDICT, EXPLAIN, and MAXIMIZE, ESP uses SQLFlow extension parser to handle them. As a result, SQLFlow will understand "normal statements" of SQL dialects, as well as those SELECT statements followed by SQLFlow extension clauses. After the input SQL statements are parsed, the CG verifies data schema with the help of SQL engine and produces the Python program, which includes information required by AI engines such as IO, dataset, and model structure. Finally, the Submitter acts as a pluggable translator, generating Job Descriptions that are tailored to various AI engines. The AI job is either run locally or submitted to a distributed computing service through it.

SQLFlow provides an interface for user interaction using a client-server architecture. The interaction with the end user is handled in the client by the "MagicCommand" module implemented in Go language. All modules in Figure 1 are included in "SQLFlow Server". "MagicCommand" and "SQLFlow Server" communicate via gRPC. The detailed architecture is provided here(Contributors, 2019).

3. SQLFlow Use Case and Performance

Classify Iris Dataset Using DNNClassifier There are four features and one label in the Iris dataset(Fisher, 1936). Each feature is represented by a float number. The label is an integer with the values 0, 1, and 2. The SQLFlow training statement is shown as Program 1. The trained DNN model is saved into table models.my_dnn_model. The following results are obtained: {'accuracy': 0.72, 'loss': 0.5695601}.

```
-- Program 1: Train a DNNClassifier on the Iris dataset --
SELECT * FROM iris.train
TO TRAIN DNNClassifier
```



(a) Average training time per iteration

(b) Average IO + training time per iteration

Figure 2: Performance for NumPy, HyPer, PostgreSQL, and SQLFlow

```
WITH
  model.n_classes = 3, model.hidden_units = [100, 100], train.epoch = 10
COLUMN sepal_length, sepal_width, petal_length, petal_width
LABEL class
INTO models.my_dnn_model;
```

Performance Comparisons We also compare the performance of NumPy, HyPer(Kemper and Neumann, 2011; Neumann et al., 2015), PostgreSQL, and SQLFlow (with MySQL) for batch gradient descent-based logistic regression (LR) using identical tests as (Blacher et al., 2022). We use a machine with an Intel Xeon E5-2650 32-core (64 threads) processor, 131 GB of RAM, and Ubuntu 20.04.4 LTS. We measure performance in terms of the average time per iteration when running the LR, and verify that all implementations achieve a training accuracy of 0.99. Figure 2(a) shows that SQLFLow, HyPer (database-friendly implementation), and NumPy achieve top three rankings on each of the three different synthetic datasets when only training time is considered, while the performance on PostgreSQL is low compared to others (out of memory when sample size $\geq 1\,000\,000$). This demonstrates the AI engine's good training performance. Figure 2(b) shows that when data IO is considered, HyPer's in-memory database feature and database-friendly SQL mappings make it the least time consuming in terms of IO and help it achieve the highest performance. SQLFlow and HyPer COO (coordinate format) are comparable. NumPy spends more than 90% of time parsing input texts, negatively impacting its performance. This shows that the optimizing data IO is still a promising topic in data systems.

We also test the multi-threaded (32 threads) performance with 100 features and 1 000 000 examples and we observe that HyPer outperforms SQLFlow by a margin of 25%. NumPy and SQLFlow have comparably performance and both outperform HyPer COO by 15%. Using SQL for ML algorithms puts computations near to the data and can thus achieve high performance, which echoes the findings of (Blacher et al., 2022).

4. Conclusion

In this paper, we introduced SQLFlow, an extensible toolkit integrating DB and AI so practitioners could build end-to-end AI pipelines with just a few lines of SQL code. As an open-source software, it contributes to the overall effort of integrating DB and AI. Future directions include accelerating data IO, supporting more frameworks and more functionalities such as privacy computing(Zhou et al., 2022; Fang et al., 2020).

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No.72192823 and No.62172362).

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *Osdi*, volume 16, pages 265–283. Savannah, GA, USA, 2016.

Alibaba. Conduct large-scale data warehousing with maxcompute, 2023. `https://www.alibabacloud.com/en/product/maxcompute`, Last accessed on 2023-01-27.

Amazon. Amazon redshift ai, 2022. `https://docs.amazonaws.cn/en_us/redshift/latest/dg/machine_learning.html`, Last accessed on 2022-09-09.

Mark Blacher, Joachim Giesen, Sören Laue, Julien Klaus, and Viktor Leis. Machine learning, linear algebra, and more: Is sql all you need? In *11th Conference on Innovative Data Systems Research, CIDR*, pages 9–12, 2022.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

SQLFlow Contributors. System-architecture, 2019. `https://github.com/sql-machine-learning/sqlflow/blob/develop/doc/design/syntax.md#system-architecture`, Last accessed on 2022-09-09.

Benjamin De Boe, Tom Woodfin, Thomas Dyar, Dave McCaldon, Aleks Djakovic, Alex MacLeod, and Don Woodlock. Integratedml: Every sql developer is a data scientist. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, pages 1–4, 2020.

Oksana Dolmatova, Nikolaus Augsten, and Michael H Böhlen. A relational matrix algebra and its implementation in a column store. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2573–2587, 2020.

Len Du. In-machine-learning database: Reimagining deep learning with old-school sql. *arXiv preprint arXiv:2004.05366*, 2020.

Wenjing Fang, Chaochao Chen, Jin Tan, Chaofan Yu, Yufei Lu, Li Wang, Lei Wang, Jun Zhou, and Alex X Liu. A hybrid-domain framework for secure gradient tree boosting. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM'20), Galway, Ireland. ACM, New York, NY, USA*, 2020.

Arash Fard, Anh Le, George Larionov, Waqas Dhillon, and Chuck Bear. Vertica-ml: Distributed machine learning in vertica database. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 755–768, 2020.

R.A. Fisher. Iris data set, 1936. `https://archive.ics.uci.edu/ml/datasets/iris`, Last accessed on 2022-09-09.

Patrick Giesser, Gabriel Stechschulte, Anna da Costa Vaz, and Michael Kaufmann. Implementing efficient and scalable in-database linear regression in sql. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 5125–5132. IEEE, 2021.

Google. Google bigquery, 2022. `https://cloud.google.com/bigquery-ml/docs/introduction`, Last accessed on 2022-09-09.

Joe Hellerstein, Christopher Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleksander Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, et al. The madlib analytics library or mad skills, the sql. *arXiv preprint arXiv:1208.4165*, 2012.

Denis Hirn and Torsten Grust. One with recursive is worth many gotos. In *Proceedings of the 2021 International Conference on Management of Data*, pages 723–735, 2021.

Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, Li Shen, Liu Tang, Yuxing Zhou, Menglong Huang, et al. Tidb: a raft-based htap database. *Proceedings of the VLDB Endowment*, 13(12):3072–3084, 2020.

Dimitrije Jankov, Shangyu Luo, Binhang Yuan, Zhuhua Cai, Jia Zou, Chris Jermaine, and Zekai J Gao. Declarative recursive computation on an rdbms: or, why you should use a database for distributed machine learning. *ACM SIGMOD Record*, 49(1):43–50, 2020.

Matthias Jasny, Tobias Ziegler, Tim Kraska, Uwe Roehm, and Carsten Binnig. Db4ml-an in-memory database kernel with machine learning support. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 159–173, 2020.

Konstantinos Karanasos, Matteo Interlandi, Doris Xin, Fotis Psallidas, Rathijit Sen, Kwanghyun Park, Ivan Popivanov, Supun Nakandal, Subru Krishnan, Markus Weimer, et al. Extending relational query processing with ml inference. *arXiv preprint arXiv:1911.00231*, 2019.

Michael Kaufmann, Gabriel Stechschulte, and Anna Huber. Efficient and accurate in-database machine learning with sql code generation in python. *arXiv preprint arXiv:2104.03224*, 2021.

Alfons Kemper and Thomas Neumann. Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots. In *2011 IEEE 27th International Conference on Data Engineering*, pages 195–206. IEEE, 2011.

Arun Kumar, Matthias Boehm, and Jun Yang. Data management in machine learning: Challenges, techniques, and systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1717–1722, 2017.

Shangyu Luo, Zekai J Gao, Michael Gubanov, Luis L Perez, and Christopher Jermaine. Scalable linear algebra on a relational database system. *IEEE Transactions on Knowledge and Data Engineering*, 31(7):1224–1238, 2018.

Nantia Makrynioti and Vasilis Vassalos. Declarative data analytics: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2392–2411, 2019.

Dennis Marten and Andreas Heuer. Machine learning on large databases: transforming hidden markov models to sql statements. *Open Journal of Databases (OJDB)*, 4(1): 22–42, 2017.

Microsoft. Microsoft sql server ai services, 2022. `https://docs.microsoft.com/en-us/sql/machine-learning/?view=sql-server-ver16/`, Last accessed on 2022-09-09.

K Venkat Nagarjuna and PV Subba Reddy. Bayesian classifiers programmed in sql using pca. *Global Journal of Computer Science and Technology*, 2012.

Thomas Neumann, Tobias Mühlbauer, and Alfons Kemper. Fast serializable multi-version concurrency control for main-memory database systems. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 677–689, 2015.

Oracle. Oracle machine learning, 2022. `https://docs.oracle.com/en/database/oracle/machine-learning/oml4sql/index.html/`, Last accessed on 2022-09-09.

Carlos Ordonez. Programming the k-means clustering algorithm in sql. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 823–828, 2004.

Carlos Ordonez and Sasi K Pitchaimalai. Bayesian classifiers programmed in sql. *IEEE Transactions on Knowledge and Data Engineering*, 22(1):139–144, 2009.

Maximilian Schleich, Dan Olteanu, Mahmoud Abo Khamis, Hung Q Ngo, and XuanLong Nguyen. A layered aggregate engine for analytics workloads. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1642–1659, 2019.

Maximilian Schüle, Frédéric Simonis, Thomas Heyenbrock, Alfons Kemper, Stephan Günnemann, and Thomas Neumann. In-database machine learning: Gradient descent and tensor algebra for main memory database systems. *BTW 2019*, 2019.

Maximilian Schule, Harald Lang, Maximilian Springer, Alfons Kemper, Thomas Neumann, and Stephan Gunnemann. In-database machine learning with sql on gpus. In *33rd International Conference on Scientific and Statistical Database Management*, pages 25–36, 2021.

Maximilian Emanuel Schüle, Alfons Kemper, and Thomas Neumann. Recursive sql for data mining. In *Proceedings of the 34th International Conference on Scientific and Statistical Database Management*, pages 1–4, 2022.

Sebastian Villarroya and Peter Baumann. On the integration of machine learning and array databases. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1786–1789. IEEE, 2020.

Yi Wang, Yang Yang, Weiguo Zhu, Yi Wu, Xu Yan, Yongfeng Liu, Yu Wang, Liang Xie, Ziyao Gao, Wenjing Zhu, et al. Sqlflow: a bridge between sql and machine learning. *arXiv preprint arXiv:2001.06846*, 2020.

Liang Xie, Yi Wang, and SQLFlow Contributors. didi-experience, 2019. `https://developpaper.com/didi-experience-sharing-how-can-sqlflow-let-operation-experts-use-ai`, Last accessed on 2022-09-09.

Binhang Yuan, Dimitrije Jankov, Jia Zou, Yuxin Tang, Daniel Bourgeois, and Chris Jermaine. Tensor relational algebra for machine learning system design. *arXiv preprint arXiv:2009.00524*, 2020.

Yuhao Zhang, Frank Mcquillan, Nandish Jayaram, Nikhil Kak, Ekta Khanna, Orhan Kislal, Domino Valdano, and Arun Kumar. Distributed deep learning on data systems: a comparative analysis of approaches. *Proceedings of the VLDB Endowment*, 14(10):1769–1782, 2021.

Jun Zhou, Longfei Zheng, Chaochao Chen, Yan Wang, Xiaolin Zheng, Bingzhe Wu, Cen Chen, Li Wang, and Jianwei Yin. Toward scalable and privacy-preserving deep neural network via algorithmic-cryptographic co-design. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–21, 2022.

Xuanhe Zhou, Chengliang Chai, Guoliang Li, and Ji Sun. Database meets artificial intelligence: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.