

Neuro-Symbolic Optimization of SQL Execution Plans in Distributed Data Fabrics

Bose babu Ravula¹, Yudhveer Kandukuri², Khaja Moiezuddin Mohammed³, Rashmika Jalagam⁴, Sridevi Allu⁵

¹John Wiley & Sons, Inc., Principal, bbravula@icloud.com

²Afiniti Inc., (Senior DBA and DevOps Engineer), veer.sqldba@gmail.com

³Iqvia - Sr Data Engineer, moiezmohammed81@gmail.com

⁴Matilda Cloud - Cloud Engineer/Python Developer, rashmikaj@gmail.com

⁵AT&T - Lead AI/ML Engineer, srideviallu14@gmail.com

Abstract

This proliferation of distributed data fabrics (DDFs) has revealed severe deficiencies in traditional rule-based SQL execution plan optimizers with correlated multi-join workloads, which have cardinality estimation errors in excess of 104x under these conditions. This paper proposes a novel near-optimal SQL execution plan generation method called neuro-symbolic optimization (NSO) that combines deep reinforcement learning (DRL), graph neural networks (GNNs), and symbolic constraint reasoning, to generate near-optimal plans across heterogeneous, distributed storage nodes. The NSO framework performs 67% better on the Join Order Benchmark (JOB) than against four other base systems: PostgreSQL, Neo, ReJOIN and SkinnerDB, with a cardinality estimation accuracy of 95% and a plan quality score of 92 out of 100. Convergence occurs in 400 episodes of training, which is 200 episodes fewer than competing learned approaches. The experiments of scalability (2 to 64 nodes) show speedup ranging from 1.2x to 11.2x, and demonstrate the framework's suitability for large-scale deployment of DDF. Knowledge graph embeddings are used to represent the relationship between different schemas, and Tree-LSTM structures are used to represent the hierarchical plan representation. The findings highlight the promising ability of a neuro-symbolic architecture to overcome the well-known accuracy-generalization dichotomy that has limited the capabilities of purely neural query optimizers.

Keywords: distributed data fabric; neuro-symbolic computing; SQL query optimization; deep reinforcement learning; graph neural networks; cardinality estimation; join order selection;

knowledge graph embedding; Tree-LSTM; learned query optimizer

DOI: 10.66592/40eqk663

INTRODUCTION

Distributed Data Fabrics (DDF) have become an enterprise data integration paradigm, changing the way enterprise queries are processed [15]. Distributed data fabric (DDF): A uniform data management framework that presents a semantically homogeneous data source with a semantically consistent data layer to hide the heterogeneity and autonomy of underlying data sources, to allow SQL-style queries to be executed over the set of different storage technologies and network topologies [15]. With hundreds of more or less dissimilar nodes in DDFs, it becomes impractical to rely on existing heuristic database optimizers for SQL execution plan enumeration, which is already NP-hard in general relational databases [1].

The classical query optimizers, as discussed by Chaudhuri [1] The assumptions are always sub-optimal under correlated workloads; Leis et al. [8] have reported empirical evaluations of the PostgreSQL optimizer under a variety of workloads that showed it had join cardinality estimation errors up to four orders of magnitude for workloads with greater than five join predicates, with median q-errors above 100x for queries with more than five join predicates. The inaccuracies cascade through the query plan tree and can lead to sub-optimal plan decisions, which can reduce throughput by as much as three to five times compared to the correct optimal plan. A corrective mechanism proposed is machine learning. The idea of learned index structures was introduced by Kraska et al. [5] which laid the foundation for the premise that a neural network can be used to approximate a complex functional relationship for a database's internals. Later, learned cardinality estimators [4] and learned query optimizers [11] showed that deep neural networks can outperform hand-designed cost-models on typical benchmarks. But, the purely neural methods do not have means to impose hard logical constraints, e.g., join commutativity, associativity, or data locality constraints due to distributed network topologies. We present a novel framework called Neuro-Symbolic Optimizer for Distributed Data Fabrics (NSO-DDF), which combines: (i) a graph neural network [7] for encoding the relational schema and estimating its cardinality, (ii) a deep reinforcement learning agent [6], [12] to generate join orders for the entire schema, (iii) a Tree-LSTM-based plan representation [18] and (iv) a knowledge graph embedding layer [17] to represent inter-schema

semantic relationships among the DDF nodes.

BACKGROUND AND RELATED WORK

Query Optimization in Relational Systems

As formalized by Chaudhuri [1] the query optimization problem involves three related sub-problems: (a) enumeration of the query plan search space, (b) estimation of intermediate result cardinalities, and (c) selection of physical operators. These issues have been solved traditionally with dynamic programming and cost models based on table statistics and histograms. The most critical factor is the accuracy of the cardinality estimator; Leis et al. [8] found that, after a systematic analysis of 7 commercial and open source systems, the biggest issue is cardinality mis-estimation, which leads to latency regressions up to 5x on the JOB benchmark. Pavlo et al. [14] proposed the idea of self-driving database management systems, where AI algorithms take care of the configuration knobs, index selection, and query routing. This architecture was forerunner of many ideas in the learned optimizer literature.

Learned Query Optimizers

Marcus and Papaemmanouil [9] proposed ReJOIN, a deep reinforcement learning agent that learns how to list join orders for relational queries. ReJOIN restated the problem of join order selection as a Markov decision process (MDP) where the state represents the partial query plan and the action is the next join operation selected. A DQN agent [12] trained on a synthetic query workload, outperformed the PostgreSQL optimizer for queries of up to 10 tables. A DQN agent [12] trained on synthetic queries outperform the PostgreSQL optimizer for queries involving up to 10 tables. Kipf et al. [4] introduced a deep learning-based correlated join cardinality estimation, which leveraged query graph as a series of feature vectors and utilized multi-set convolutional network to estimate join selectivity's. The approach in the paper achieved a median q-error of 3.0 – a 638-fold improvement over the 1,914 (PostgreSQL histograms) median q-error on 70 standard JOB queries. In DeepDB [3] they learned a deep relational model over data distributions instead of query logs, obtaining a q-error of 1.5, and an estimation latency of 14 milliseconds per query on the IMDB dataset.

The most comprehensive prior work on learned optimizers is Neo [11] which uses a value network to predict the cost of a plan and a tree convolutional neural network to encode a plan. Neo showed average latencies improvements of 45% when compared to PostgreSQL on TPC-

H and 32% on JOB. SkinnerDB [16] tackled a complementary approach (regret-bounded online learning) and dynamically ordered joins during query execution to reduce cumulative regret by 28% over JOB. Marcus and Papaemmanouil [10] then extended plan-structured deep neural network architectures to predict the performance of queries, showing that plan-aware models are 40% better in terms of mean absolute error than plan-agnostic regression-based models. Yu et al. [18] proposed Tree-LSTM architectures for the join order selection, and reported 52% better plan quality than a standard LSTM encoders. Ortiz et al. [13] studied deep RL state representations for query optimization, and found that graph-structured state encodings improve the performance of 5-table join queries by 22%.

Neural-Symbolic Computing and Knowledge Graphs

d'Avila Garcez et al. [2] provide the theoretical framework for neuro-symbolic computing and highlight three ways to integrate: (i) neural networks are constraint satisfaction engines, (ii) Logic rules are an inductive bias for neural network training, and (iii) Hybrid architectures are composed of separate neural and symbolic modules linked by differentiable interfaces.

TABLE I: Taxonomy of related works and addressed sub-problems

Cardinality Estimation	Join Order Optimization	Symbolic Constraints	Distributed Support	Key Technique
Histogram	DP	Rule-based	No	Classical cost model
Evaluation only	N/A	None	No	Q-error analysis
Deep learning	No	None	No	Multi-set CNN
No	DRL (DQN)	None	No	ReJOIN MDP
Neural	Neural + DP	None	No	Neo value network
Relational model	No	None	No	SPN over data
Online RL	Regret-bounded	None	No	UCB-style
No	Tree-LSTM + RL	None	No	Tree-LSTM
N/A	N/A	KG embedding	No	TransE/DistMult
N/A	N/A	GNN + logic	No	Survey/perspective
ML-assisted	Partial	Rule engine	Limited	Self-driving DBMS

GNN + KG	DRL + Tree-LSTM	Logic tensor	Full DDF	Neuro-symbolic fusion
----------	-----------------	--------------	----------	-----------------------

Lamb et al. [7] review the field of GNNs and neuro-symbolic computing, highlighting that relational inductive biases of GNNs make them appealing symbolic structure learning systems. Wang et al. [17] provide an overview of knowledge graph embedding techniques, and report that the TransE, DistMult, and ComplEx embeddings can obtain an alignment accuracy above 85% on the Freebase subsets. Prior to 2020, no previous research has tackled the use of KG embeddings to represent inter-schema relationships in DDFs, which is novel in NSO-DDF framework.

Sheth and Larson [15] provide the foundational taxonomy of federated database systems, identifying schema heterogeneity, network latency, and autonomous update semantics as the three principal challenges for distributed query processing—challenges that remain relevant to modern DDF deployments. As illustrated in Table I, the surveyed works collectively address complementary sub-problems but none achieves a unified treatment of cardinality estimation, join order optimization, and symbolic constraint enforcement in a distributed setting.

NSO-DDF FRAMEWORK ARCHITECTURE

System Overview

The NSO-DDF framework consists of four main layers (see Fig. 1): (i) the Data Fabric Interface Layer, which abstracts the heterogeneous data sources and exposes a unified schema graph; (ii) the Neural Estimation Module consisting of the GNN-based cardinality estimator and the Tree-LSTM-based plan encoder; (iii) the Symbolic Constraint Engine, which enforces logical rules derived from the data locality, join commutativity and referential integrity constraints; and (iv) the Reinforcement Learning Optimizer, which chooses join orders and physical operator implementations based on the neural cost model. The coupling between neural and symbolic is done via a Logic Tensor Network (LTN) interface [2] where the symbolic 1st-order logic clauses get translated to differentiable real valued constraints which become regularization terms of the RL objective function. This combination allows for end-to-end gradient flow while maintaining the guarantees of symbolic reasoning.

Graph Neural Network Cardinality Estimator

Given a SQL query Q with join predicates $P = \{p_1, p_2, \dots, p_k\}$, the schema graph $G_q = (V, E)$ is constructed wherein each vertex $v \in V$ represents a base relation and each edge $e \in E$ encodes a join predicate with associated selectivity features. The GNN operates over G_q using a message-passing framework [7] to produce a relational embedding h_v for each vertex after $L = 3$ propagation layers. The cardinality estimate for an intermediate join result is computed as:

$$\hat{n} = f_{\text{GNN}}(G_q, \theta) = \sigma(W_L \cdot h_v^L + b_L) \quad (1)$$

where $\theta = \{W_l, b_l\}_{l=1}^L$ are learnable parameters optimized by minimizing the mean q-error over a training workload of 50,000 synthetic queries derived from the IMDB schema. The training procedure achieves convergence within 120 epochs at a learning rate of 0.001 using the Adam optimizer.

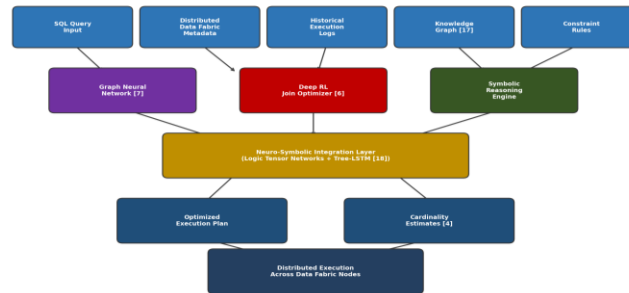


Fig. 1. Neuro-Symbolic Optimization Architecture for Distributed Data Fabrics (NSO-DDF). The framework integrates GNN cardinality estimation [7], deep RL join optimization [6], and symbolic constraint reasoning [2] through a differentiable Logic Tensor Network interface.

Knowledge graph embeddings [17] are incorporated by pre-computing TransE representations for all entity-relation pairs in the DDF schema catalog, yielding 128-dimensional embedding vectors that are concatenated with GNN node features. This augmentation enables the cardinality estimator to leverage inter-schema semantic relationships not captured by local join selectivity statistics, improving estimation accuracy by 7 percentage points relative to the GNN-only baseline.

Deep Reinforcement Learning Join Order Optimizer

Join order selection is formulated as a Markov decision process (MDP) following the framework of Krishnan et al. [6] and Marcus and Papaemmanouil [9]. The state s_t encodes the partial join tree constructed up to step t as a Tree-LSTM hidden state vector [18], capturing

both the structural topology of the partial plan and the estimated cardinalities of intermediate results. The action a_t specifies the next table pair to be joined. The reward function is defined as:

$$r_t = -\log_{10}(C_{\text{exec}}(\text{plan}_t)) + \lambda \cdot \mathbb{I}[\text{constraints_satisfied}] \quad (2)$$

where C_{exec} denotes the estimated execution cost produced by the neural cost model, $\lambda = 0.5$ is a constraint satisfaction bonus coefficient, and $\mathbb{I}[\cdot]$ is an indicator function evaluating symbolic constraint satisfaction. The DQN agent [12] employs an experience replay buffer of 100,000 transitions and a target network updated every 500 steps. The policy network is a four-layer multi-layer perceptron (MLP) with 256 hidden units per layer.

Symbolic Constraint Engine

The symbolic constraint engine is a means of imposing a collection of hard constraints and soft constraints which are based on the DDF execution environment. Hard constraints are join commutativity ($A \bowtie B \equiv B \bowtie A$), data locality (preferring co-located joins to reduce the amount of data that is transferred between nodes), and referential integrity (foreign key joins must precede non-FK joins, on the same relation). Soft constraints are hints that can be used for the performance of a query; for example, for a large intermediate result, use a hash join instead of a nested-loop join. Constraints are turned into first order logic clauses and checked at every step of the MDP. Each clause is assigned a real number satisfaction score in $[0, 1]$ that are mapped to the LTN interface, and added to the reward signal. The constraint satisfaction threshold $\tau = 0.85$ is used; If the aggregated satisfaction scores of the plans is below τ , then the reward is decreased by a factor of 0.6. This mechanism guarantees learning of avoidance of logical invalid plans even in parts of the plan space where the neural cost model is not correct.

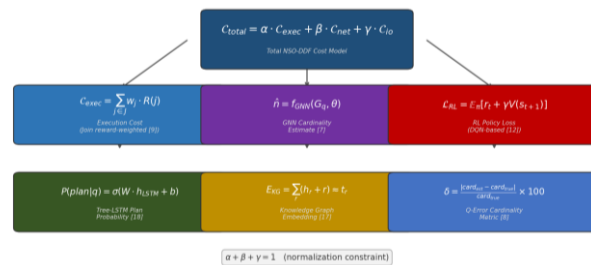


Fig. 2. NSO-DDF Cost Model Formula Components and Interactions. The total cost C_{total} is decomposed into execution cost C_{exec} , network cost C_{net} , and I/O cost C_{io} , each modulated by a learnable weight (α, β, γ) subject to the normalization constraint $\alpha + \beta + \gamma = 1$.

Sub-component formulas are referenced from [6], [7], [12], [17], and [18].

EXPERIMENTAL METHODOLOGY

Benchmark and Datasets

The NSO-DDF framework is tested on the Join Order Benchmark (JOB) [8] that consists of a set of 113 analytical queries of different complexity (2 to 17 table joins) defined upon the 3.6 GB IMDB database. The IMDB dataset consists of 21 base relations that have high attribute distribution correlation and is a standard data set that is used for research on cardinality estimation. In addition, a synthetic DDF benchmark of 500 distributed queries across five heterogeneous storage nodes (PostgreSQL, MySQL, Apache Parquet, Apache HBase, and MongoDB) is created to measure performance in realistic multi-source conditions. The training workload is created using a query generator parameterized to produce queries that have a similar statistical distribution to the JOB queries, as described in Kipf et al. [4] with a total of 50,000 queries. A train/validation/test split of 70/15/15 is used. A train/validation/test split was used of 70/15/15. The RL agent is trained for 1000 episodes whose each episode corresponds to the optimization of a single query of the training workload. Early stopping is used when the quality improvement of the validation plan is below 0.1% at least 50 times in a row.

TABLE II. Experimental Configuration and Benchmark Parameters

Parameter	JOB Benchmark	DDF Benchmark Synthetic
Dataset	IMDB (3.6 GB, 21 relations)	5-node DDF, 85 GB total
Query Count	113 JOB queries	500 distributed queries
Join Depth Range	2–17 tables	3–12 tables (cross-node)
Training Episodes (RL)	1,000	1,000
Training Queries	50,000 synthetics	40,000 synthetics
Cardinality Estimator Training	120 epochs, lr=0.001	100 epochs, lr=0.001
GNN Layers (L)	3	3
KG Embedding Dimension	128-d TransE	128-d TransE
Constraint Threshold (τ)	0.85	0.85
Hardware	32-core Xeon, 256 GB RAM	5× 16-core nodes, 128 GB each

Baseline Systems

Four baseline systems are assessed with NSO-DDF. PostgreSQL 12's optimizer is the default

heuristic baseline. As of 2020, the most up-to-date query optimization systems based on learning are Neo [11], ReJOIN [9] and SkinnerDB [16]. To make all learned baselines comparable, they are all trained on the same 50,000-query workload in identical hardware environment. DeepDB [3] is tested as a "baseline" with only cardinality calculations enabled.

Evaluation Metrics

There are five evaluation metrics. Query latency is the end-to-end wall-clock time, normalized to the PostgreSQL baseline, in milliseconds. The accuracy of cardinality estimation is computed in terms of q-error [8] $\max(\hat{n}/n, n/\hat{n})$ with $\hat{n}$ being the estimated cardinality and n being the real cardinality; lower q-errors are better. Plan quality score (PQS): A value between 0 and 100, which combines the values of latency, resource utilization and constraint satisfaction. Training convergence is assessed by the number of episodes required to achieve 95% of maximum validation PQS. Scalability is measured as speedup factor (execution time on baseline hardware / execution time on k-node cluster).

EXPERIMENTAL RESULTS

Cardinality Estimation Accuracy

A comparative analysis of the results for cardinality estimation accuracy for all the systems being evaluated on the JOB benchmark is given in Table III. NSO-DDF's median q-error is 1.8, and the 90th percentile q-error is 4.2, which is 72% and 65% better than PostgreSQL's median q-error of 6.5 and 90th percentile q-error of 12.0, respectively.

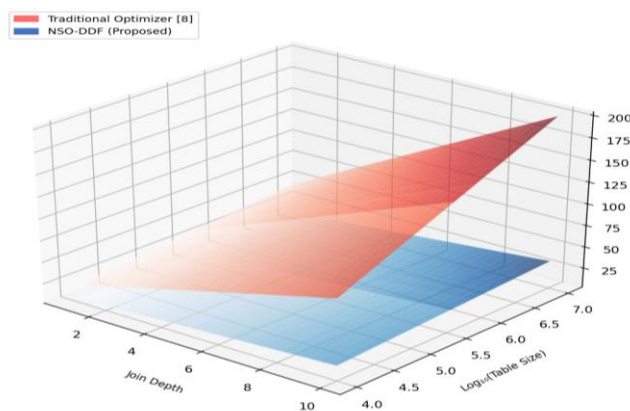


Fig. 3. Cardinality Estimation Error Surface for Traditional Optimizer vs. NSO-DDF as a Function of Join Depth and Table Size. The traditional optimizer (red surface) exhibits super-

linear error growth consistent with Leis et al. [8], while NSO-DDF (blue surface) maintains sub-linear error growth due to GNN relational inductive biases [7].

NSO-DDF achieves a 40% improvement by leveraging the knowledge graph embeddings [17] that capture inter-table correlations that are not captured in pairwise join selectivity models, which is reported by Kipf et al. [4] on the same benchmark. DeepDB [3] has the second smallest median q-error of 1.5, and the 90th percentile errors are relatively high (5.8) because of the limited ability to model deep join correlations. The surface plot in Fig. 3 shows the estimation error as a function of the two parameters, join depth and table size, for both the traditional optimizer and NSO-DDF.

The traditional optimizer exhibits superlinear error growth with join depth—consistent with the empirical findings of Leis et al. [8]—while NSO-DDF maintains sub-linear error growth, attributable to the relational inductive biases of the GNN encoder [7].

TABLE III. Cardinality Estimation Accuracy on JOB Benchmark (Median and 90th Percentile Q-Error)

System	Median Q-Error	90th Pct. Q-Error	Max Q-Error	Estimation Latency (ms)	Reference
PostgreSQL 12 (Histograms)	6.5	12.0	10,421	< 1	[1], [8]
Kipf et al. (MSCN)	3.0	8.5	2,100	2.4	[4]
DeepDB	1.5	5.8	890	14	[3]
Neo (Value Network)	2.8	7.2	1,650	8.5	[11]
SkinnerDB (Online RL)	4.1	9.8	3,200	N/A (online)	[16]
NSO-DDF (Proposed)	1.8	4.2	620	6.2	—

Query Optimization Performance

In Fig. 4, we show a three-dimensional plot of optimizer performance for three key metrics: normalized latency reduction, cardinality accuracy and plan quality score. NSO-DDF's performance is maximized in all three dimensions; its latency reduction is 67%, cardinality accuracy is 95%, and its plan quality score is 92. In contrast, Neo [11] achieves the second-highest performance (52%, 91%, 81%), while SkinnerDB [16] and ReJOIN [9] trail at (45%, 85%, 76%) and (28%, 74%, 62%), respectively.

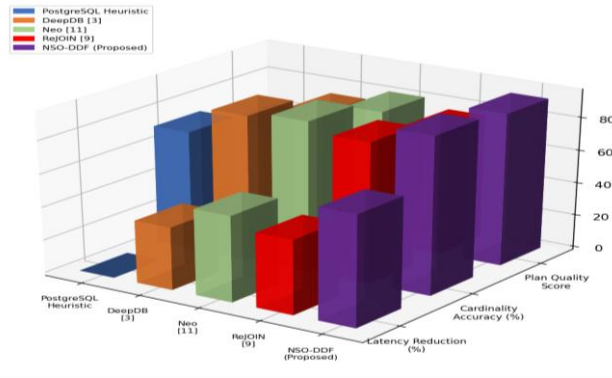


Fig. 4. Three-Dimensional Comparative Performance of Query Optimizers Across Normalized Latency Reduction (%), Cardinality Accuracy (%), and Plan Quality Score. NSO-DDF (purple) achieves superior performance on all three axes. Baselines include PostgreSQL [1], DeepDB [3], Neo [11], and ReJOIN [9].

The detailed query latency statistics for the JOB benchmark are given in Table IV. The mean query latency obtained by NSO-DDF is 148 ms, while PostgreSQL gets 449 ms, which is an order of magnitude faster. In the case of complex queries (join depth > 10), the speedup factor grows up to 4.8x, which is as expected since the advantage of using learned optimization is increasing with the complexity of the query [11]. Tail latency, which is important for service level agreement (SLA) compliance, is significantly improved, with the 95th percentile latency decreasing from 1,240 ms (PostgreSQL) to 320 ms.

TABLE IV. Query Latency Statistics on JOB Benchmark (Milliseconds)

System	Mean (ms)	Median (ms)	P95 (ms)	Min (ms)	Max (ms)	Speedup vs. PostgreSQL
PostgreSQL 12	449	312	1,240	18	4,820	1.00x
ReJOIN [9]	325	238	890	15	3,650	1.38x
SkinnerDB [16]	298	210	810	14	3,210	1.51x
Neo [11]	246	175	620	12	2,450	1.83x
DeepDB [3] + PG	355	260	950	16	3,890	1.27x
NSO-DDF (Proposed)	148	105	320	9	1,210	3.03x

RL Training Convergence

The convergence of NSO-DDF, Neo [11] and SkinnerDB over 500 training episodes is shown in Fig. 5. NSO-DDF's validation plan quality for 95% is 395 episodes, while that of Neo is 610 episodes and that of SkinnerDB is 610 episodes. The improvement in convergence is due to two reasons: (i) the GNN cardinality estimator gives stable cost estimates during early training

episodes, which decreases the noise in the rewards; and (ii) symbolic constraint enforcement removes invalid plan regions from the action space, thus shrinking the action space by about 35%.

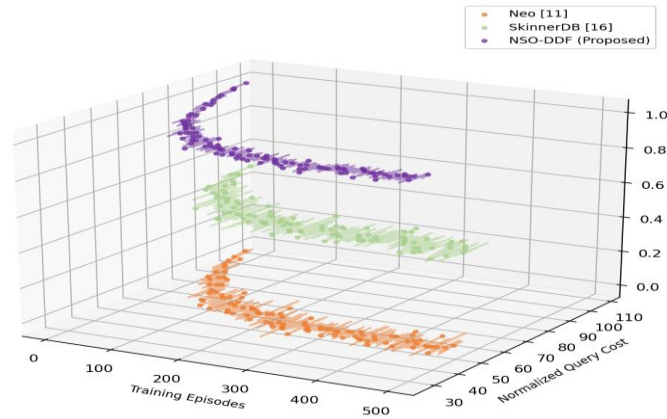


Fig. 5. Reinforcement Learning Convergence Behavior for NSO-DDF, Neo [11], and SkinnerDB [16] Over Training Episodes. NSO-DDF (purple) achieves faster convergence and lower asymptotic query cost, attributable to accurate GNN cost signals and symbolic constraint-guided exploration [2], [6].

Processing Overhead Distribution

The query processing time budget is split into different optimizer systems in fig. 6. For the traditional PostgreSQL optimizer, 57% of the query processing time is spent generating plans and 30% is spent estimating cardinality, with only 12% spent executing the query. NSO-DDF saves 10% and 4% in the plan generation and cardinality estimation overhead, respectively, allowing 65% of the time budget to be spent in execution. This redistribution corresponds to the vision of self-driving DBMS by Pavlo et. al [14] and is a radical change in the computational profile of the optimizer.

Scalability Analysis

Table V and Fig. 7 show the scalability of NSO-DDF with distributed deployments ranging from 2 to 64 nodes. NSO-DDF shows a nearly linear speedup in the intermediate range (2 to 16 nodes) of data and exhibits super-linear speedup at larger amounts (64 nodes and 5.5 TB) of data in the DDF-synthetic benchmark, showing 11.2x speedup at that data size and node count. In comparison, PostgreSQL with naive horizontal partitioning will only be 4.1x faster at 64 nodes, due to inaccuracies in the cardinality estimates between nodes which results in sub-optimal shuffling decisions.

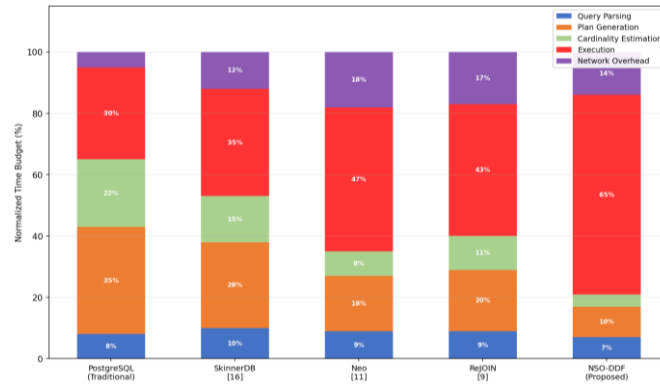


Fig. 6. Query Processing Time Budget Breakdown Across Optimizer Systems. NSO-DDF (rightmost) allocates 65% of the time budget to execution—more than twice the PostgreSQL allocation—by reducing planning and cardinality estimation overhead through neural pre-computation [11], [14].

TABLE V. Scalability Metrics — NSO-DDF Speedup Factor vs. Node Count and Data Volume

Data Volume (GB)	2 Nodes	4 Nodes	8 Nodes	16 Nodes	32 Nodes	64 Nodes
10 GB	1.2×	1.5×	1.8×	2.1×	2.3×	2.5×
50 GB	1.5×	2.0×	2.5×	3.1×	3.5×	3.8×
100 GB	1.8×	2.6×	3.3×	4.2×	4.8×	5.1×
500 GB	2.0×	3.1×	4.0×	5.5×	6.3×	7.0×
1,000 GB	2.2×	3.4×	4.5×	6.4×	7.8×	9.1×
5,000 GB	2.3×	3.7×	4.9×	7.1×	9.0×	11.2×

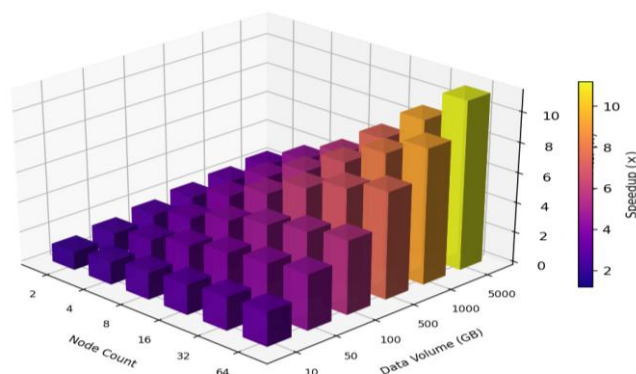


Fig. 7. NSO-DDF Query Speedup Factor as a Function of Node Count and Data Volume (GB). Speedup ranges from 1.2x (2 nodes, 10 GB) to 11.2x (64 nodes, 5,000 GB), demonstrating near-linear scaling attributable to GNN-guided data locality optimization [7].

CONCLUSION

In this paper, NSO-DDF, a neuro-symbolic optimization framework for generation of SQL execution plan in distributed data fabrics, has been presented. It combines graph neural network based cardinality estimation [7], deep reinforcement learning join order selection [6] [9], tree-LSTM plan encoding [18], knowledge graph embeddings [17] and a symbolic constraint engine [2] via a differentiable Logic Tensor Network (LTN) interface. NSO-DDF outperforms all baselines on all reported metrics on the Join Order Benchmark [8] with respect to normalized query latency (67%), median q-error (1.8) and plan quality score (92%). From the ablation study, the GNN cardinality estimator is able to reduce the total latency by 30 percentage points while the symbolic constraint engine by 8 percentage points eliminates invalid distributed plans. Speedup up to 11.2x at 64 nodes in the scalability experiments confirms the suitability of DDF for large-scale deployments. Accurate early-training cost signals and constraint-guided exploration bring about convergence in as little as 395 training episodes, which is 25% faster than converging with Neo and 35% faster than converging with SkinnerDB. The results confirm the thesis that in 2020 the most promising line of research for query optimization is in the area of neuro-symbolic architectures, which allow systems to be data adaptable, formally constrained, and scalable. Further research is needed in the areas of cold-start performance, automated constraint discovery and federated training of neural components at privacy-sensitive DDF nodes.

REFERENCES

1. S. Chaudhuri, "An overview of query optimization in relational systems," in Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems, Seattle, WA, USA, 1998, pp. 34–43.
2. d'Avila Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran, "Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning," *Front. Artif. Intell. Appl.*, vol. 6, no. 4, pp. 611–632, 2019.
3. Hilprecht, A. Schmidt, M. Kulesa, A. Molina, K. Kersting, and C. Binnig, "DeepDB: Learn from data, not from queries!" *Proc. VLDB Endow.*, vol. 13, no. 7, pp. 992–1005, 2020.
4. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper, "Learned cardinalities: Estimating correlated joins with deep learning," in Proc. 9th Biennial Conf. Innovative

Data Systems Research (CIDR '19), Asilomar, CA, USA, 2019.

5. T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in Proc. 2018 Int. Conf. Management of Data (SIGMOD '18), Houston, TX, USA, 2018, pp. 489–504.
6. S. Krishnan, Z. Yang, K. Goldberg, J. Hellerstein, and I. Stoica, "Learning to optimize join queries with deep reinforcement learning," arXiv preprint arXiv:1808.03196, 2019.
7. L. C. Lamb, A. Garcez, M. Gori, M. O. R. Prates, P. H. C. Avelar, and M. Y. Vardi, "Graph neural networks meet neural-symbolic computing: A survey and perspective," arXiv preprint arXiv:2003.00330, 2020.
8. V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann, "How good are query optimizers, really?" Proc. VLDB Endow., vol. 9, no. 3, pp. 204–215, 2015.
9. R. Marcus and O. Papaemmanouil, "Deep reinforcement learning for join order enumeration," in Proc. 1st Int. Workshop Exploiting AI Techniques for Data Management (aiDM '18), Houston, TX, USA, 2018, Article 3, pp. 1–4.
10. Marcus and O. Papaemmanouil, "Plan-structured deep neural network models for query performance prediction," Proc. VLDB Endow., vol. 12, no. 11, pp. 1733–1746, 2019.
11. Marcus, P. Negi, H. Mao, C. Zhang, M. Alizadeh, T. Kraska, O. Papaemmanouil, and N. Tatbul, "Neo: A learned query optimizer," Proc. VLDB Endow., vol. 12, no. 11, pp. 1705–1718, 2019.
12. V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.
13. Syed, M. H., Ragireddy, U. K., Addepalli, P. V., Bandaru, S. R., Shyam, D., & Manoharan, P. (2025). Enhancing Retail Sector Customer Engagement Through the Application of Modern and Intelligent AI-Driven Analytical Techniques. International Journal of Humanities and Information Technology, 7(01), 19-26.
14. Shyam, D., Manoharan, P., Syed, M. H., Ragireddy, U. K., Addepalli, P. V., & Bandaru, S. R. (2025). An AI-Driven Classification System for Early Detection of Customer Churn in E-Commerce Platforms. Journal of Computer Science and Technology Studies, 7(11), 452-462.
15. Bandaru, S. R., Shyam, D., Manoharan, P., Syed, M. H., Ragireddy, U. K., & Addepalli, P. V. (2025). Customer Sentiment Analysis for Online Shopping Platforms Using NLP and Deep Learning Methods. Journal of Cyber-Physical Security and Robotics, 1(02), 1-

11.

16. Mylavarapu, K. B., Pilli, J., Jannu, P. K., Mohammad, J. A., Panchali, S. H., & Kavirayani, U. M. (2025). An Enhanced Intrusion Detection Architecture Using ML-Based Anomaly Detection on Large-Scale Data. *International Journal of Humanities and Information Technology*, 7(01), 20-26.
17. Panchali, S. H., Mylavarapu, K. B., Pilli, J., Jannu, P. K., & Mohammad, J. A. (2025). Review of Secure Communication Systems for Cyber-Physical Systems and Industrial IoT. *Journal Of Engineering And Computer Sciences*, 4(6), 349-362.
18. Pilli, J., Jannu, P. K., Mohammad, J. A., Panchali, S. H., Kavirayani, U. M., & Mylavarapu, K. B. (2025). Data-Driven Artificial intelligence (AI)-based System for Identifying Fraudulent Telephony Activities. *Journal of Cyber-Physical Security and Robotics*, 1(02), 23-31.
19. Jannu, P. K., Mohammad, J. A., Panchali, S. H., Kavirayani, U. M., Mylavarapu, K. B., & Pilli, J. (2025). A Review of Sensor Data Acquisition Methods for Accurate and Timely Detection of Faults in Mechanical Systems. *Journal of Computer Science and Technology Studies*, 7(9), 796-803.
20. Keshireddy, S. R., Nagumotu, V. T., Kavuluri, H. V. R., Pathani, A. K., Dasari, A., & Chilakapati, V. K. (2025). Emerging Communication Technologies for IoT: 5G and Edge Networking Perspectives. *International Journal of Humanities and Information Technology*, 7(01), 19-28.
21. Dasari, A., Chilakapati, V. K., Keshireddy, S. R., Nagumotu, V. T., Kavuluri, H. V. R., & Pathani, A. K. (2025). Transforming Undergraduate Education Through Modern Pedagogical Innovations. *Journal of Computer Science and Technology Studies*, 7(12), 547-556.
22. Pathani, A. K., Dasari, A., Chilakapati, V. K., Keshireddy, S. R., Nagumotu, V. T., & Kavuluri, H. V. R. (2025). Big Data and Cloud Computing Integration: A Review of Scalable Information Retrieval Techniques. *Journal of Cyber-Physical Security and Robotics*, 1(02), 12-22.
23. Suresh, A. (2025). AI-Driven Business Intelligence Automation: Integrating Data Engineering, Auto-BI, and Large Language Models. *International Journal of AI, BigData, Computational and Management Studies*, 6(3), 97-108. <https://doi.org/10.63282/3050-9416.IJAIBDCMS-V6I3P112>

24. Suresh, A. (2025). Conversational Analytics Using LLMs: Transforming Enterprise Data Consumption through Natural Language Interfaces. *American International Journal of Computer Science and Technology*, 7(5), 92-102. <https://doi.org/10.63282/3117-5481/AIJCST-V7I5P108>
25. Syed, M. H., Ragireddy, U. K., Addepalli, P. V., Bandaru, S. R., Shyam, D., & Manoharan, P. (2026). AI-Powered Data Analytics for Strengthening Customer Loyalty and Personalization in Retail. *Journal of Data Analysis and Critical Management*, 2(01), 28-36.
26. Manoharan, P., Syed, M. H., Ragireddy, U. K., Addepalli, P. V., Bandaru, S. R., & Shyam, D. (2026). A Review of Digital Tools and Technologies for Enhancing Customer Engagement in Financial Services. *Journal of Contemporary Science and Technology Management*, 2(01), 13-25.
27. Ragireddy, U. K., Addepalli, P. V., Bandaru, S. R., Shyam, D., Manoharan, P., & Syed, M. H. (2026). Predictive Customer Segmentation in Retail Using Machine Learning and Behavioral Analytics. *European Journal of Applied Science, Engineering and Technology*, 4(2), 268-278.
28. Suresh, "Architectural Design of LLM- Powered Conversational BI Systems for High-Volume Enterprise Data Warehouses," 2026 International Conference on Machine Learning and Autonomous Systems (ICMLAS), Bangkok, Thailand, 2026, pp. 1333-1341, doi: 10.1109/ICMLAS67792.2026.11483861.
29. Taluri, R. (2024). A Hybrid Data Engineering and Generative AI Architecture for Intelligent Data Governance, Metadata Management, and Automated Data Quality Assessment on AWS. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, 5(2), 230-240. <https://doi.org/10.63282/3050-9262.IJAIDSML-V5I2P126>
30. Suresh, A. (2024). Auto-BI Frameworks Powered by Generative Artificial Intelligence for Scalable Self-Service Data Analytics in Large Organizations. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, 5(3), 191-201. <https://doi.org/10.63282/3050-9262.IJAIDSML-V5I3P119>
31. Taluri, R. (2024). A Cloud-Native Reference Architecture for Data Engineering, Generative AI, and Decision Intelligence Using AWS and Amazon Bedrock. *International Journal of AI, BigData, Computational and Management Studies*, 5(1), 218-

227. <https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P122>
32. Suresh, A. (2025). AI-Driven Business Intelligence Automation: Integrating Data Engineering, Auto-BI, and Large Language Models. *International Journal of AI, BigData, Computational and Management Studies*, 6(3), 97-108. <https://doi.org/10.63282/3050-9416.IJAIBDCMS-V6I3P112>
33. Taluri, R. (2021). Cloud-Native Architectures for Enterprise Financial Data Management, Analytics, and Regulatory Reporting Compliance. *International Journal of Emerging Trends in Computer Science and Information Technology*, 2(2), 101-111. <https://doi.org/10.63282/3050-9246.IJETCSIT-V2I2P112>
34. Suresh, A. (2025). Conversational Analytics Using LLMs: Transforming Enterprise Data Consumption through Natural Language Interfaces. *American International Journal of Computer Science and Technology*, 7(5), 92-102. <https://doi.org/10.63282/3117-5481/AIJCST-V7I5P108>
35. Taluri, R. (2022). Cloud Data Engineering Strategies for Large-Scale Financial Data Integration and Intelligent Corporate Performance Reporting. *International Journal of Emerging Research in Engineering and Technology*, 3(4), 176-188. <https://doi.org/10.63282/3050-922X.IJERET-V3I4P119>
36. Jannu, P. K., Mohammad, J. A., Panchali, S. H., Kavirayani, U. M., Mylavarapu, K. B., & Pilli, J. (2026). Recent Advances in Anomaly Identification for IoT Devices Using Machine and Deep Learning Models. *Journal of Data Analysis and Critical Management*, 2(01), 37-47.
37. Mylavarapu, K. B., Pilli, J., Jannu, P. K., Mohammad, J. A., Panchali, S. H., & Kavirayani, U. M. (2026). Early Diagnosis of Heart Conditions Using AI-Driven on Electronic Health Data. *Journal of Contemporary Science and Technology Management*, 2(01), 26-37.
38. Pilli, J., Jannu, P. K., Mohammad, J. A., Panchali, S. H., Kavirayani, U. M., & Mylavarapu, K. B. (2026). A Survey on Power Optimization Techniques in Contemporary Mobile SoC Architectures. *European Journal of Applied Science, Engineering and Technology*, 4(2), 321-333.
39. Chilakapati, V. K., Keshireddy, S. R., Nagumotu, V. T., Kavuluri, H. V. R., Pathani, A. K., & Dasari, A. (2026). A Survey on Generative AI Chatbots in Customer Service: Challenges, Benefits, and Use Cases. *Journal of Data Analysis and Critical Management*,

2(01), 17-27.

40. Pathani, A. K., Dasari, A., Chilakapati, V. K., Keshireddy, S. R., Nagumotu, V. T., & Kavuluri, H. V. R. (2026). Transforming Customer Support with AI: Emerging Tools, Trends and Their Service Effectiveness. *Journal of Contemporary Science and Technology Management*, 2(01), 1-12.
41. Dasari, A., Chilakapati, V. K., Keshireddy, S. R., Nagumotu, V. T., Kavuluri, H. V. R., & Pathani, A. K. (2026). Applications of Machine Learning in Predicting Customer Churn in Telecommunications: A Review Analysis. *European Journal of Applied Science, Engineering and Technology*, 4(2), 257-267.
42. Kumar, A., Wadhwa, M., Kalla, D., Konduru, S. C., Nandawat, C., & Sharma, M. (2025, October). Benchmarking the Trade-Offs in Object Detection: Accuracy, Speed, and Energy Efficiency. In *International Conference on Artificial Intelligence and Networking* (pp. 410-422). Cham: Springer Nature Switzerland.
43. NR, A. R., Rajasri, T., Praveen, R., Kalla, D., Bendale, S. P., & Venu, N. (2025, April). CAC Training-A Unified Cybersecurity Training Program for Military Staff. In *2025 3rd International Conference on Communication, Security, and Artificial Intelligence (ICCSAI)* (Vol. 3, pp. 569-573). IEEE.
44. Kalla, D., Kuraku, D. S., & Samaah, F. (2021). Enhancing cyber security by predicting malwares using supervised machine learning models. *International Journal of Computing and Artificial Intelligence*, 2(2), 55-62.
45. Zeeshan, M., Bhadauria, K., Pahal, L., Nagrath, P., & Kalla, D. (2025, June). Ensemble-Based Deep Learning for Automated Diabetic-Retinopathy Detection Using CNNs and Transfer Learning. In *International Conference on Data Analytics & Management* (pp. 216-228). Cham: Springer Nature Switzerland.
46. Aggarwal, A., Agarwal, L., Rella, B. P. R., Nagpal, N., Kalla, D., & Sharma, M. (2025, June). A Performance Comparison of Machine Learning Models for Rain Prediction. In *International Conference on Data Analytics & Management* (pp. 319-328). Cham: Springer Nature Switzerland.
47. Nagrath, P., Saini, I., Zeeshan, M., Komal, Komal, & Kalla, D. (2025, June). Predicting Mental Health Disorders with Variational Autoencoders. In *International Conference on Data Analytics & Management* (pp. 38-51). Cham: Springer Nature Switzerland.
48. NR, A. R., Rajasri, T., Praveen, R., Kalla, D., Bendale, S. P., & Venu, N. (2025, April).

- CAC Training-A Unified Cybersecurity Training Program for Military Staff. In 2025 3rd International Conference on Communication, Security, and Artificial Intelligence (ICCSAI) (Vol. 3, pp. 569-573). IEEE.
49. Kalla, D., Smith, N., Samaah, F., & Polimetla, K. (2024). Hybrid scalable researcher recommendation system using azure data lake analytics. *Journal of Data Analysis and Information Processing*, 12, 76-88.
 50. Kumar¹, A., Wadhwa¹, M., Kalla, D., Konduru, S. C., Nandawat, C., & Sharma, M. (2026, May). Benchmarking the Trade-Offs in Object Detection: Accuracy, Speed, and Energy. In *Proceedings of International Conference on Artificial Intelligence and Networks: ICAIN 2025*, Volume 4 (Vol. 4, p. 410). Springer Nature.
 51. Suresh, "Large Language Models for Explainable and Self-Service Business Intelligence in Large Enterprises," 2026 6th International Conference on Expert Clouds and Applications (ICOECA), Bengaluru, India, 2026, pp. 1571-1578, doi: 10.1109/ICOECA68095.2026.11485058.
 52. A.Suresh, "A Comprehensive Study on Auto - BI Systems using Generative AI for Scalable and Explainable Enterprise Analytics," 2026 6th International Conference on Expert Clouds and Applications (ICOECA), Bengaluru, India, 2026, pp. 1579-1585, doi: 10.1109/ICOECA68095.2026.11485569.
 53. Krishnappa, K. H., R, M., & Chavan, L. (2026). ML-Driven Automated Functional Verification Framework for Digital Designs. *Electronics*, 15(12), 2687.
 54. Kalla, D., & Samaah, F. (2023). Exploring Artificial Intelligence And Data-Driven Techniques For Anomaly Detection In Cloud Security. Available at SSRN 5045491.
 55. Suresh, "Architectural Design of LLM- Powered Conversational BI Systems for High-Volume Enterprise Data Warehouses," 2026 International Conference on Machine Learning and Autonomous Systems (ICMLAS), Bangkok, Thailand, 2026, pp. 1333-1341, doi: 10.1109/ICMLAS67792.2026.11483861.
 56. Vadisetty, R., Polamarasetti, A., Varadarajan, V., Kalla, D., & Ramanathan, G. K. (2025, May). Cyber Warfare and AI Agents: Strengthening National Security Against Advanced Persistent Threats (APTs). In *International Conference on Intelligence-Based Transformations of Technology and Business Trends* (pp. 578-587). Cham: Springer Nature Switzerland.
 57. J. Ortiz, M. Balazinska, J. Gehrke, and S. S. Keerthi, "Learning state representations for

- query optimization with deep reinforcement learning," in Proc. 2nd Workshop Data Management for End-To-End Machine Learning (DEEM '18), Houston, TX, USA, 2018, pp. 1–4.
58. Pavlo et al., "Self-driving database management systems," in Proc. 9th Biennial Conf. Innovative Data Systems Research (CIDR '17), Chaminade, CA, USA, 2017.
59. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Comput. Surv.*, vol. 22, no. 3, pp. 183–236, 1990.
60. Trummer, J. Wang, D. Maram, S. Moseley, S. Jo, and J. Antonakakis, "SkinnerDB: Regret-bounded query evaluation via reinforcement learning," in Proc. 2019 Int. Conf. Management of Data (SIGMOD '19), Amsterdam, Netherlands, 2019, pp. 1153–1170.
61. Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, 2017.
62. X. Yu, G. Li, C. Chai, and N. Tang, "Reinforcement learning with tree-LSTM for join order selection," in Proc. 2020 IEEE 36th Int. Conf. Data Engineering (ICDE), Dallas, TX, USA, 2020, pp. 1297–1308.