Database Workload Characterization with Query Plan Encoders

VLDB2022

Debjyoti Paul, Jie Cao, Feifei Li, Vivek Srikumar
deb, jcao, lifeifei, svivek @cs.utah.edu
The University of Utah
InitialDLab and Utah NLP

Introduction

Each database running a different workload, demands different resources and database configuration settings to achieve optimal performance, which prompts us to study workload features in detail.
We define a database workload as

$$W = \Big\{(p_1, \theta_1), (p_2, \theta_2), \ldots, (p_m, \theta_m)\Big\},$$

where p_i is the database query-plan, and θ_i is a normalized weight of importance of p_i in workload W. For understanding workloads comprehensively it is necessary to perform feature engineering on query plans.

Key Contributions

- ↪ We propose query plan encoder models capturing structure and computational performance resource requisites as distributed feature representations.
- ↪ We keep structure, and computational performance representation separate that enables downstream tasks to weigh each representation independently in their model.
- ↪ We propose a taxonomy for operator types for learning diverse structure of query plans with self-attentive transformers.
- ↪ We find performance of query plans are best characterized by encoders when plan task nodes are classified under scan, join, sort and aggregate; each having an encoder of its type.
- ↪ Latency prediction and query classification downstream tasks performing well with our pretrained encoders suggests efficacy of our modeling strategy.
- ↪ In depth domain adaptation evaluation and ablation studies on various datasets signifies pretrained encoders adapts to new domain quickly, whereas encoders trained from scratch overfits.
- ↪ In this work, we open-sourced an automated workload execution tool for cloud, a crowd-sourced plan dataset and revised two spatial benchmarks.

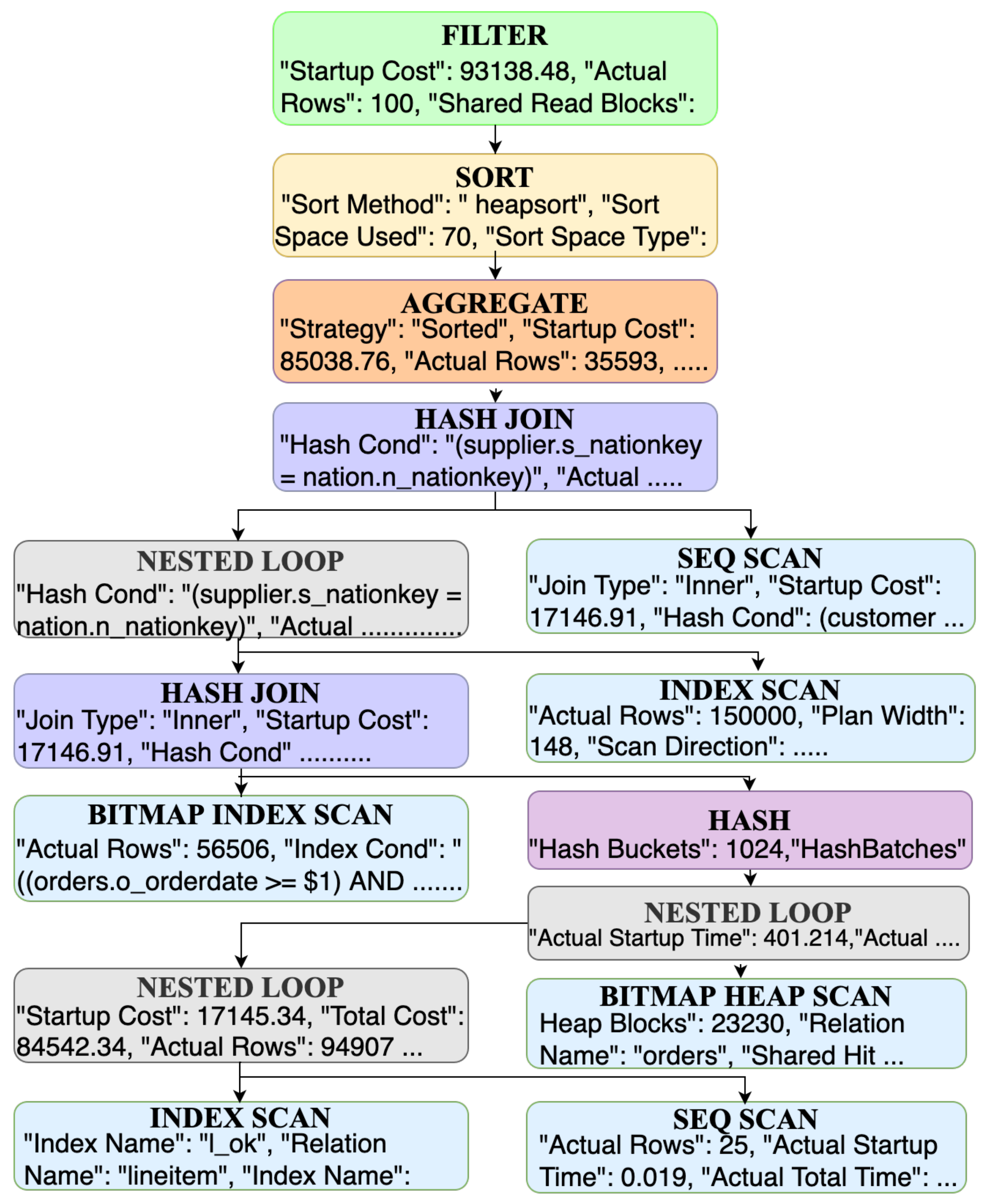
FILTER
"Startup Cost": 93138.48, "Actual Rows": 100, "Shared Read Blocks":
SORT
"Sort Method": " heapsort", "Sort Space Used": 70, "Sort Space Type":
AGGREGATE
"Strategy": "Sorted", "Startup Cost": 85038.76, "Actual Rows": 35593, ...
HASH JOIN
"Hash Cond": "(supplier.s_nationkey = nation.n_nationkey)", "Actual .....
NESTED LOOP
"Hash Cond": "(supplier.s_nationkey = nation.n_nationkey)", "Actual ....
SEQ SCAN
"Join Type": "Inner", "Startup Cost": 17146.91, "Hash Cond": (customer ...
HASH JOIN
"Join Type": "Inner", "Startup Cost": 17146.91, "Hash Cond" ....
INDEX SCAN
"Actual Rows": 150000, "Plan Width": 148, "Scan Direction": .....
BITMAP INDEX SCAN
"Actual Rows": 56506, "Index Cond": " ((orders.o_orderdate >= $1) AND ....
HASH
"Hash Buckets": 1024,"HashBatches"
NESTED LOOP
"Actual Startup Time": 401.214,"Actual ....
NESTED LOOP
"Startup Cost": 17145.34, "Total Cost": 84542.34, "Actual Rows": 94907 ...
BITMAP HEAP SCAN
Heap Blocks": 23230, "Relation Name": "orders", "Shared Hit ...
INDEX SCAN
"Index Name": "l_ok", "Relation Name": "lineitem", "Index Name":
SEQ SCAN
"Actual Rows": 25, "Actual Startup Time": 0.019, "Actual Total Time": ...

Fig 1. An example of query plan tree with different types of task/operators nodes. It is to note that many properties are associated with each task node. This query plan is from TPC-H Query Template 5.

Plan Encoders

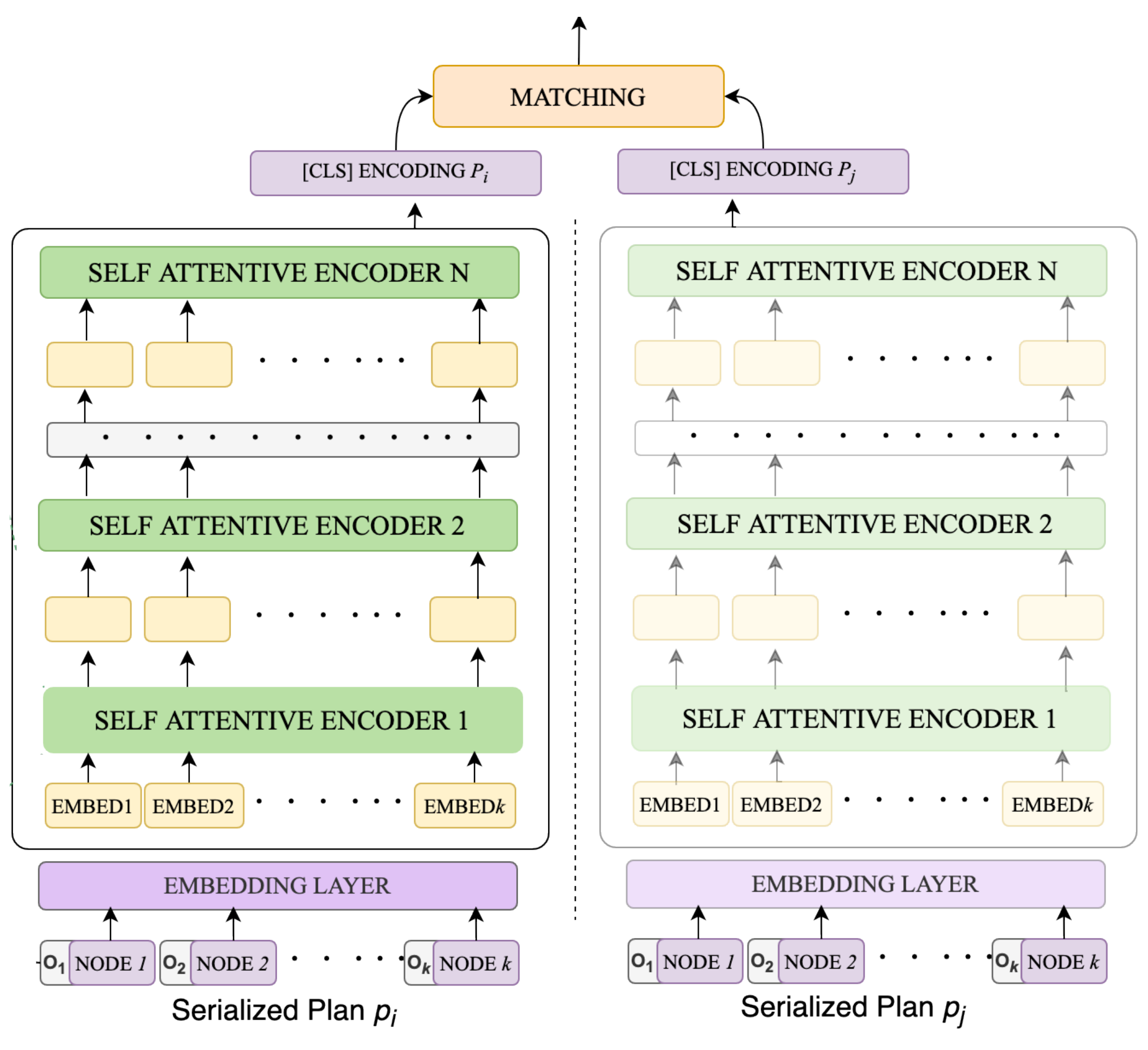

Fig 2. Structure Plan Encoder Modeling.

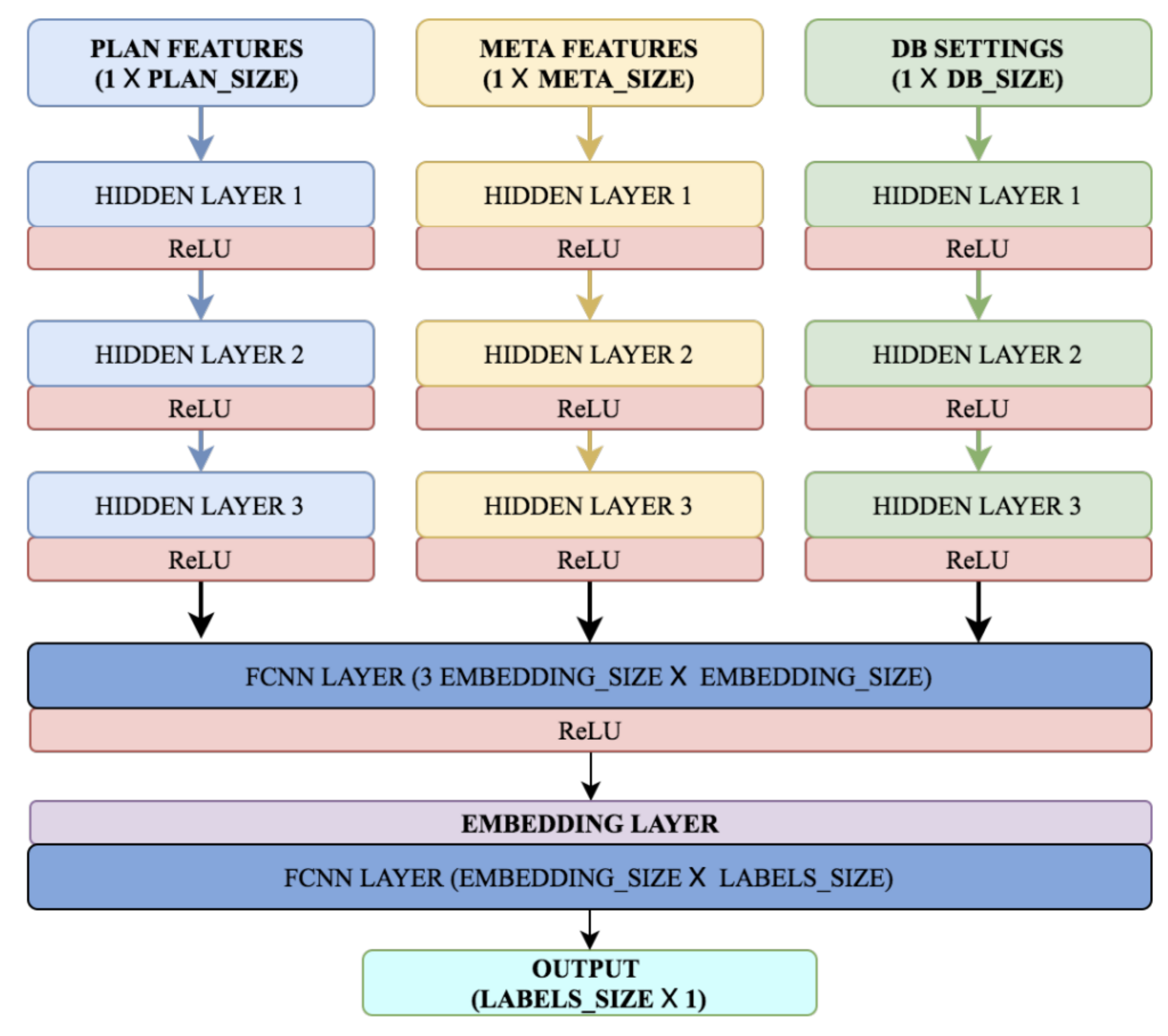
PLAN FEATURES (1 X PLAN_SIZE)
META FEATURES (1 X META_SIZE)
DB SETTINGS (1 X DB_SIZE)
HIDDEN LAYER 1
ReLU
HIDDEN LAYER 2
ReLU
HIDDEN LAYER 3
ReLU
FCNN LAYER (3 EMBEDDING_SIZE X EMBEDDING_SIZE)
ReLU
EMBEDDING LAYER
FCNN LAYER (EMBEDDING_SIZE X LABELS_SIZE)
OUTPUT (LABELS_SIZE X 1)

Fig 3. Computational Performance Encoder Modeling.

Downstream Task Modeling

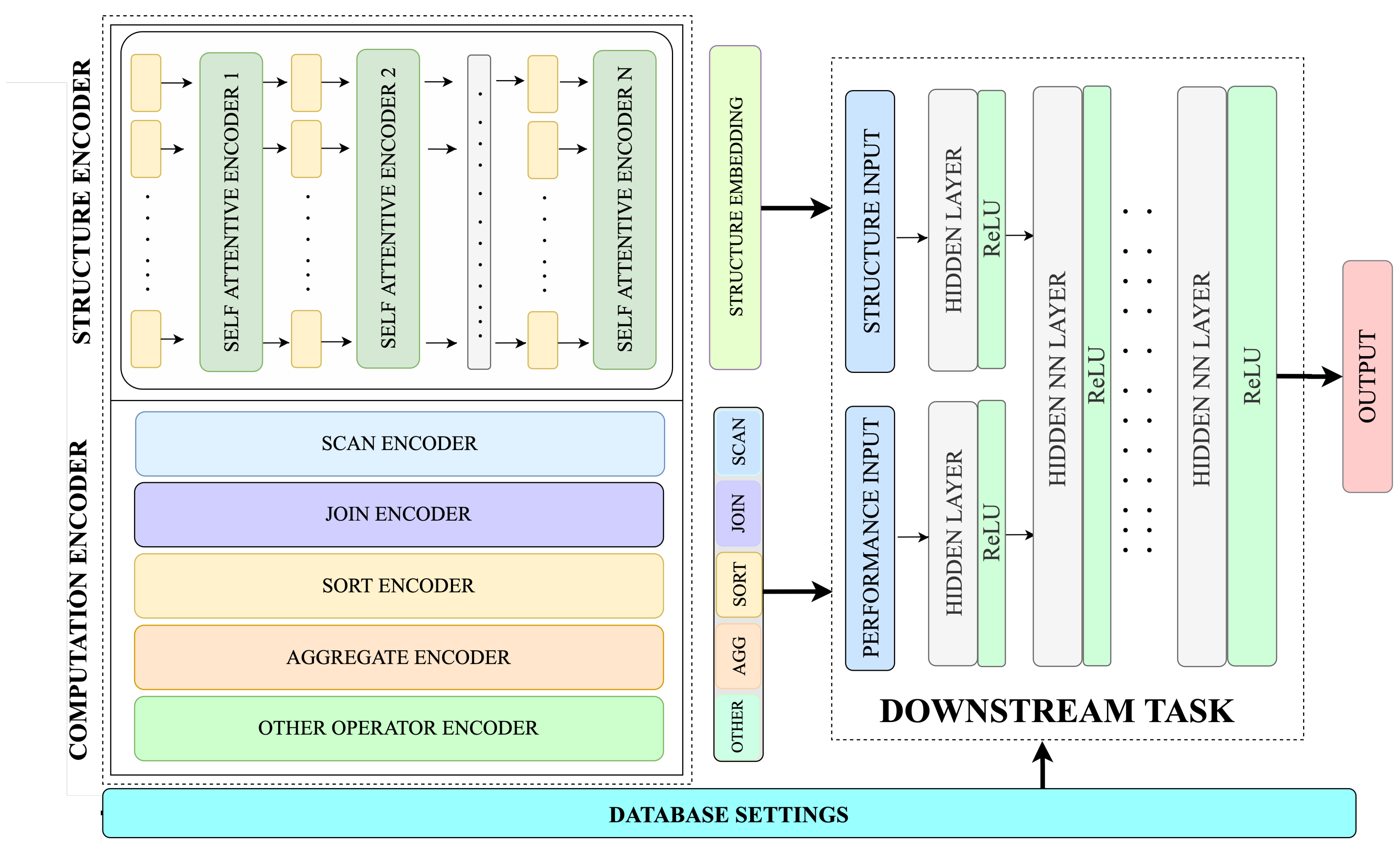

Fig 4. A bird-view architecture diagram, showing the role of plan encoders for downstream tasks. For example, latency prediction and query classification tasks.

Results

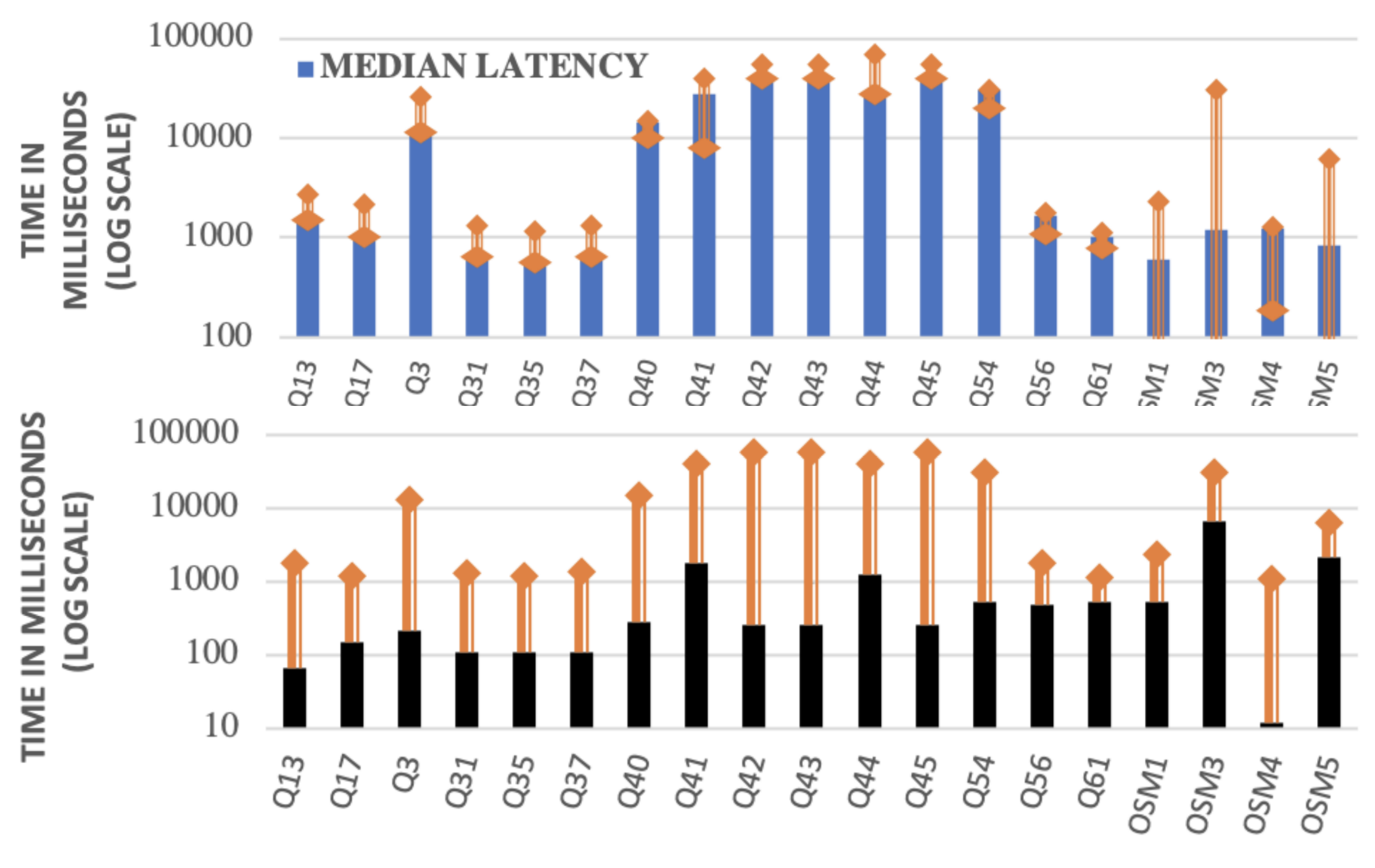

Fig 5. Blue bars are median query latency, Orange lines are 5th-95th percentile range variations, and mean abs. error marked with black bar for spatial queries. A smaller black bar on a larger orange-line bar means better results.

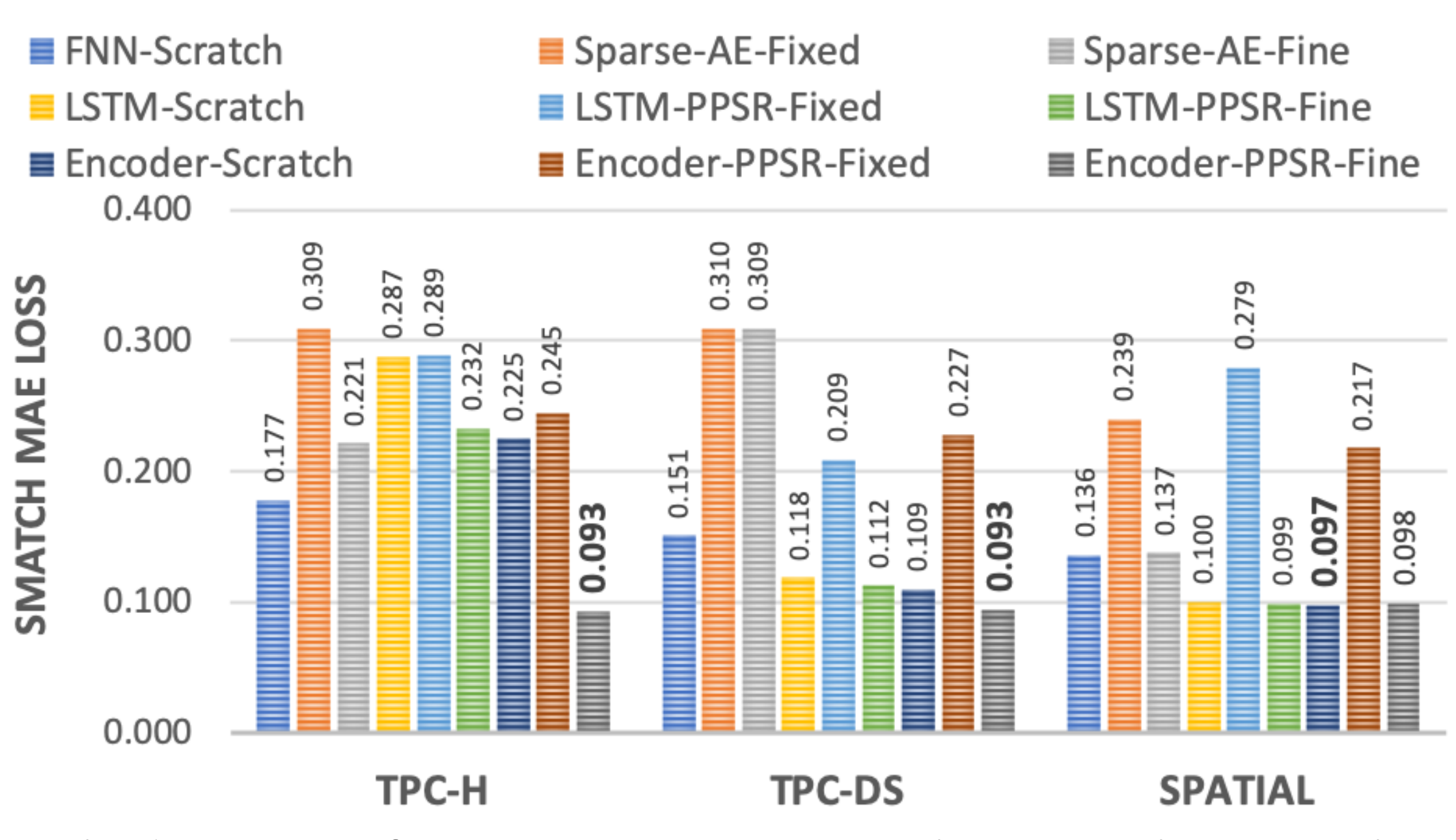

Fig 6. Results of structure encoder domain adaptation analysis on TPC-H, TPC-DS, and SPATIAL datasets. Notations: `Scratch` is Untrained encoder weights initialized; `Fixed` is Pretrained encoder weights freeze. `Fine` is Pretrained+Finetuned Encoder.

$$R(q) = \max\left(\frac{predicted(q)}{original(q)}, \frac{original(q)}{predicted(q)}\right)$$

Models	$R \le 1.5$	$1.5 < R \le 2.0$	$R > 2.0$
TAM [4]	51%	22%	27%
SVF [5]	68%	15%	17%
RBF [6]	85%	6%	9%
QPPNet [7]	89%	**7%**	4%
Plan Encoder	**91%**	**7%**	**2%**

Table 1. Percentage of queries from TPC-DS SF-100 testset binned based on R-factor for all the models in evaluations. Pretrained Plan Encoder performed well with 91% queries within 1.5R and only 2% queries above 2.0R.

Models	Development		Test	
	template	cluster	template	cluster
Structure only	0.2452	0.4670	0.1946	0.3847
Performance only	0.1645	0.2973	0.0977	0.1769
Both encoders	**0.2783**	**0.5573**	**0.2518**	**0.4647**
Both encoders 10% data	0.2000	0.4927	0.151	0.334
Both encoders 30% data	0.2555	0.5228	0.1843	0.3855

Table 2. F1-scores of models for template and cluster query classification task on development and test dataset.

https://linkmix.co/11389156
Scan this QR Code for open-source resources.