SOLAR: Scalable Distributed Spatial Joins through Learning-based Optimization — Appendix

Yongyi Liu*, Ahmed Abdelmaguid*, Ahmed R. Mahmood†, Amr Magdy*, Minyao Zhu†
*University of California, Riverside, USA
Email: {yliu786, aabde039}@ucr.edu, amr@cs.ucr.edu
†Google LLC, Mountain View, USA
Email: amahmoo@google.com, minyaozhu@gmail.com

APPENDIX A
TRAINING PROCESS

In this appendix, we describe the process and details for training the Siamese Network similarity model. For details on training the Random Forest decision model, see Section VI-C [1].

A. Environment Setup

As described in Section VIII [1], the model training and experiments were conducted on a Microsoft Azure HDInsight cluster consisting of two head nodes (E8 V3, 8 cores, 64 GB RAM) and eight worker nodes (E2 V3, 2 cores, 16 GB RAM). For the model training environment, we used Python 3.9, PyTorch 2.6, and Shapley 2.0.1.

B. Dataset Preparation

The dataset was split into 80% training and 20% test. Join pairs were generated for both the training set and the test set. When the pair generation process is based on a random function, in the scalability experiment, the random seed was fixed to `2025316` to ensure reproducibility. The preparation of training samples used in the training process is conducted during *SOLAR*'s offline phase. There are two main stages : (i) Partition generation and histogram calculation, and (ii) Dataset embeddings extraction and generation.

1) Partition Generation and Histogram Calculation: We first generate partitioners over the training set pairs. This provides an initial set of saved workload partitioners that can be reused during *SOLAR*'s online phase as described in Section VII [1].

After that, for each dataset, a high-resolution 2D data histogram of 8192×8192 is computed. These histograms are then flattened into a vector to be used for generating similarity scores using JSD, as mentioned in Section V [1]. These scores are used as *ground truth* labels assigned to each dataset pair.

2) Dataset Embeddings Extraction and Generation: As referenced in Section VI-A [1], the Siamese Network model is trained on embedding vectors representing the datasets. Each dataset embedding consists of a 9-dimensional vector encoded using its metadata. These metadata include the number of points, polygonal area, polygon centroid coordinates (centroid_x, centroid_y), bounding box ($\min_x$, $\min_y$, $\max_x$, $\max_y$), and the compactness measure. In our experiments, polygon-level metadata was computed using the `Shapely` library. Given a set of two-dimensional points (x, y), a convex hull polygon is constructed, and the required features are extracted to create the embeddings.

C. Hyperparameter Tuning and Model Training

The architecture of the Siamese network is well-suited for our task of generating similarity scores based on an input pair. To train the model, the training data consists of:

- Training samples, where each sample represents a pair of datasets, and each dataset is encoded as a 9-dimensional vector.
- Similarity labels, computed based on the JSD similarity score derived from each dataset's flattened 2D histogram.

The model was trained using the Adam optimizer and MSE as the loss function, minimizing the mean squared error between the predicted distance and the actual JSD score. *The learning rate* and *weight decay* used in the training were chosen based on a hyperparameter search. The search was conducted through 5-fold cross-validation on *learning rate* values selected from $0.0001, 0.0003, 0.001, 0.003, 0.01$ and *weight decay* values of 0 and 0.0001. The random seed for the cross-validation was set to `42`. After choosing the parameters, the model was trained for 50 epochs on the full training set using a batch size of 24, with early stopping (patience 10).

D. Offline Phase Cost

The offline phase of *SOLAR* incurred three main costs, all measured on the training dataset mentioned in Section VIII-A [1]. (i) The computation of dataset histograms, necessary for deriving ground-truth Jensen–Shannon divergence values, took about 19 minutes. (ii) The extraction and storage of datasets metadata features required approximately 7.7 hours. (iii) Finally, the hyperparameter search for the learning rate and weight decay, performed using 5-fold cross-validation, together with the final training of the Siamese network on the full dataset, required approximately 6 minutes.

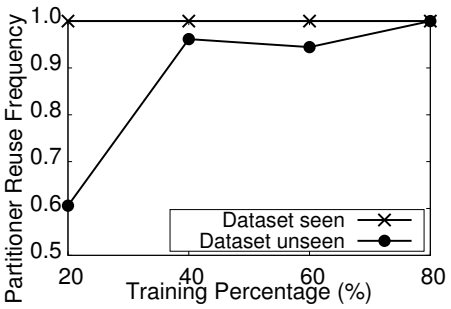


Fig. 1: Matching Frequency Vs Training Data Percentage

APPENDIX B
ADDITIONAL EXPERIMENTS

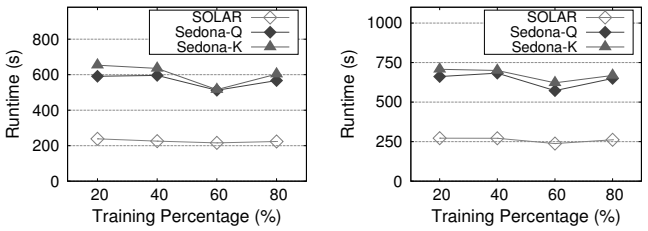
A. Partitioner Reuse Frequency Analysis

Figure 1 illustrates the frequency with which an existing partitioner is selected for incoming joins. We examine cases where upcoming joins are repeated (i.e., were seen during *SOLAR*'s training) or not repeated (i.e., were not seen during *SOLAR*'s training). In this experiment, we increase the percentage of datasets used in *SOLAR*'s training from 20% to 80% of our overall spatial datasets.

The figure highlights that when *SOLAR* is presented with incoming joins composed of datasets encountered during training, its model accurately identifies these as repeated operations and efficiently reemploys a pre-computed partitioner. This ability allows *SOLAR* to effectively detect and bypass redundant partitioning for recurring join tasks. The reason is that identical datasets yield identical embeddings, resulting in a feature-space distance of zero, indicative of maximum similarity, as explained in Section VI [1]. However, if we build joins from *test datasets* that have not been directly seen before during the training of *SOLAR*, as we increase the number of *training datasets* used in the training of *SOLAR*, the number of corresponding *training joins* grows proportionally. As a result, the system encounters and stores a larger variety of dataset pairs and their associated partitioners during the offline phase. Consequently, the probability of successfully matching a dataset from a new join to an existing partitioner increases significantly. This explains the upward trend observed in Figure 1, where the likelihood of partitioner reuse for new incoming joins improves with a larger set of *training datasets* and *training joins*. This further highlights the applicability and scalability of our architecture in industrial environments, where datasets are continuously integrated into production workflows. As the model accumulates more datasets, the repository of partitioners expands, thereby enhancing the system's capability to effectively reuse existing partitioners.

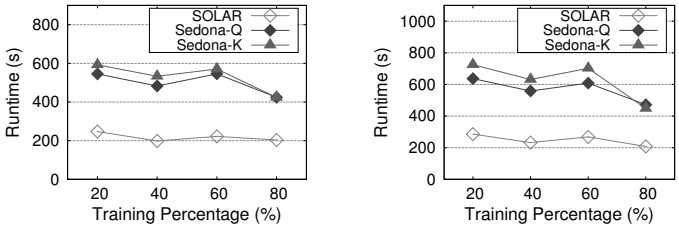
B. Supplementary plots

For completeness, we provide supplementary plots corresponding to the results in Figures 6–9 [1]. Each figure in the paper text contains four plots (best, worst, 25%, and 75%), but the 50% plots were excluded for space limitations, as they give almost the same conclusions as the 75% percentile plots. To confirm this, we include both the 50% and 75% plots in Figures 2–5.



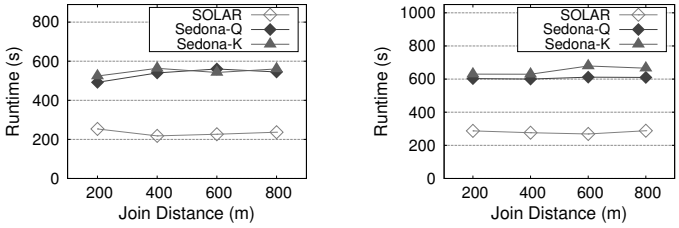
(a) 50 Percentile (b) 75 Percentile

Fig. 2: Runtime under different percentages of training data for *training joins* (50% and 75% plots)



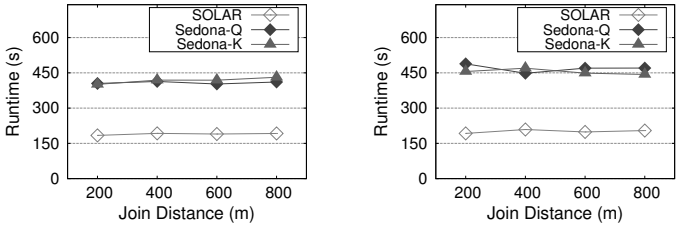
(a) 50 Percentile (b) 75 Percentile

Fig. 3: Runtime under Different Percentage of Training Data for *test joins* (50% and 75% plots)



(a) 50 Percentile (b) 75 Percentile

Fig. 4: Runtime under Different Join Distances for *training joins* (50% and 75% plots)



(a) 50 Percentile (b) 75 Percentile

Fig. 5: Runtime under Different Join Distances for *test joins* (50% and 75% plots)

REFERENCES

[1] Y. Liu, A. Abdelmaguid, A. R. Mahmood, A. Magdy, and M. Zhu, "Solar: Scalable distributed spatial joins through learning-based optimization," in *Proceedings of the IEEE 42nd IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2026.