Dense Vector Retrieval in Data Federation

Amorntip Prayoonwong*, Yang-Chun Hsu†, Xin-Jie Ye‡, Po-Kai Lu‡, Chih-Hang Wang‡, and Chih-Yi Chiu‡
* Suratthani Rajabhat University, Thailand
E-mail: amorntip.pra@sru.ac.th
† National Chiayi University, Taiwan, R.O.C.
E-mail: s1110323@mail.ncyu.edu.tw
‡ National Chung Cheng University, Taiwan, R.O.C.
E-mail: {g13410124, g13410096, chwang, cychiu}@ccu.edu.tw

***Abstract*—Searching in dense vector datasets has been extensively applied in numerous fields. When submitting a query over a data federation of different organizations, how to protect the sensitive and private data becomes a critical concern. Most methods mainly focus on imposing security constraints in federated computation; only a few methods address the computation bottleneck on local data silos, especially for the intensive query processing. In this paper, we propose an efficient dense vector estimation for each query. That is, the server can predict the appropriate number of vectors to be returned from each silo. As the prediction is precise enough, the silo does not need to retrieve redundant vectors and thus can save the computation time during the retrieval. Experimental results demonstrate the effectiveness and efficiency of the proposed method in dense vector data federation.**

I. Introduction

The approximate nearest neighbor search (ANNS) is essential for efficient similarity retrieval in high-dimensional vector spaces and is widely used in applications such as recommendation systems [1][2], natural language processing [3][4], and image retrieval [5][6]. The growing importance of ANNS search in large-scale machine learning applications has driven the adoption of vector databases as a powerful solution for high-dimensional data retrieval. Vector databases enable efficient storage and fast querying of embedding vectors, allowing for rapid similarity search with trade-offs in accuracy, latency, and memory efficiency. As data volumes continue to grow, achieving a low-latency and scalable ANNS method has become increasingly important.

In many real-world scenarios, data is distributed across multiple organizations or devices and often contains sensitive information. Domains such as healthcare and finance face strict legal and ethical constraints that prevent centralized data sharing. This presents significant challenges for traditional centralized vector search systems. Federated computation offers a promising approach for enabling collaborative data analysis and query processing, such as similarity search across distributed clients without transmitting sensitive raw data, thus ensuring data privacy [7][8]. However, most federated computation systems emphasize security through cryptographic techniques such as homomorphic encryption (HE) [9] and secure multiparty computation (SMC) [10], which incur high computational and communication overhead [11]. As a result, balancing privacy with efficiency remains a key challenge. Moreover, in the context of high-dimensional or large-scale datasets, the efficiency bottleneck shifts from security computation to local search computation when using ANNS [7]. The local silos often retrieve excessive data than necessary to meet the query demand. Techniques such as learned adaptive early termination (LAET) have proven to be effective in reducing unnecessary computation in centralized ANNS by applying learned early termination strategies [12] [13]. However, their adaptation to federated settings has not yet been explored.

To address the challenges of efficient vector retrieval in federated environments while maintaining privacy guarantees, we propose a novel adaptive approach that leverages server-side predictive modeling to optimize query processing across distributed clients. Our method dynamically estimates the optimal number of clusters to search and the number of candidate neighbors for each client, thereby minimizing redundant local computations while preserving the privacy of individual datasets. Through extensive experiments on a large-scale vector dataset, we demonstrate that our approach significantly improves both search efficiency and retrieval accuracy compared to existing federated ANNS methods. By jointly addressing privacy preservation and search performance, our method establishes a new paradigm for scalable, privacy-aware information retrieval in federated environments [14].

II. Related Work

Recent advances in federated and privacy-preserving computation have stimulated substantial research on secure and efficient *k*-nearest neighbor and similarity search methods. This section reviews research that contributes novel methodologies to address challenges in ANNS while preserving privacy in federated environments.

A. Privacy-Preserving Federated k-NN and Similarity Search

Zhang et al. [7] introduced FedKNN, a framework that allows secure search for k-nearest neighbors in federated environments. FedKNN minimizes redundant local processing and safeguards participant data from being exposed to third parties. It leverages differential privacy techniques and hardware enclaves to maintain the privacy of each participant. Zhu et al. [11] proposed FedSQ, a system that performs vector similarity queries within federated learning environments while maintaining user privacy and data confidentiality. FedSQ comprises

979-8-3315-7206-8/25/$31.00

four essential components: local vector index construction, privacy-preserving query execution, top-*k* result selection, and secure aggregation. Addressing *k*-nearest neighbor queries in spatial data federations, Zhang et al. [14] proposed a one-round algorithm that completes queries in a single communication round, with data owners locally selecting candidates and computing similarity scores before a coordinator ranks and selects top-*k* results. This approach significantly reduces communication overhead but may sacrifice accuracy. In contrast, their alternative multi-round Algorithm employs iterative query refinement, achieving higher accuracy with complex spatial relationships at the cost of increased communication and computation time. Although these methods provide privacy guarantees and reduced communication and costs, they face computational and latency challenges, particularly in large-scale federated environments.

B. Efficiency Optimization in ANNS

Recent advances in approximate nearest-neighbor search (ANNS) have increasingly focused on enhancing computational efficiency through early stopping and adaptive termination strategies. Li et al. [12] proposed LAET, a model-agnostic framework that employs lightweight classifiers to predict optimal early stopping points using features such as distance variance and candidate score distributions, achieving up to 40% latency reduction while maintaining high search accuracy. Similarly, Prayoonwong et al. [13] developed a multitask learning model that jointly predicts neighbor distributions and determines minimal codewords for effective re-ranking, thereby supporting informed early termination decisions to reduce unnecessary computations; Busolin et al. [15] introduced an unsupervised, patience-based early exit method for dense retrieval with two-level cluster indexing, achieving up to five times faster retrieval with only a minor accuracy trade-off. In particular, their method incorporates dynamic mechanisms that enhance flexibility in dense retrieval scenarios. These approaches collectively represent a paradigm shift from exhaustive search toward intelligent early termination strategies, offering both learned and heuristic solutions for different ANNS application scenarios.

III. The Proposed Method

Consider a federated data scenario where the federation consists of multiple autonomous databases (also referred to as clients), which collectively enable queries on sensitive data through a unified interface (referred to as the server) without disclosing unauthorized information to any participant involved in the search process. In this federated environment, clients cannot share or communicate data with each other; therefore, ANNS must be executed independently on each local database, with the server aggregating the results. To meet privacy and security requirements, the federation system is built on the semi-honest threat model [14][7]. This model assumes that all participants—including the server and clients—follow the protocol during execution with honesty but curiosity, meaning they may attempt to infer private information about other parties from intermediate data or interaction records observed during the process. Under this model, unauthorized information leakage must be prevented through system design and privacy protection mechanisms.

Fig. 1. illustrates an example of a data federation composed of three clients and one server. The user provides the server with input conditions consisting of a query vector Q and the desired number of nearest neighbors k. The server returns k nearest neighbors collected from all clients. A naive approach requires each client to return k candidate neighbors to the server, which then ranks and returns the top k nearest neighbors to the user. However, this design causes most clients to return excessive candidate neighbors, resulting in redundant ANNS computations and degrading the overall system efficiency. The goal of this paper is to propose a mechanism that effectively reduces the ANNS computation load on clients without compromising privacy, thus improving the efficiency of the federated system while maintaining acceptable accuracy.

We train a prediction model on the server to estimate search parameters of federated clients, so that each client can perform efficient ANNS with reduced vector comparisons. Let $\{C_1, C_2, \ldots, C_M\}$ be the M clients of the data federation, each of which holds a private vector database. The vector data in the m-th client's database is represented as $V_m = \{v_{m,1}, v_{m,2}, \ldots, v_{m,N_m}\}$, where N_m denotes the number of vectors in V_m. Each vector database is indexed using an inverted file structure (IVF), implemented via the Faiss library [16]. The index is built by applying the k-means clustering algorithm over the vector dataset. During the ANNS process, the server inputs the given query vector Q and the desired number of nearest neighbors k into a prediction model, denoted as f, which then outputs the following pairs: $\{nProb_m, kPred_m \mid m = 1, 2, \ldots, M\}$, where $nProb_m$ is the number of clusters to be visited in C_m and $kPred_m$ is the number of candidate neighbors to be returned from C_m to the server. They are used as search parameters to query the IVF index structure in clients; they significantly affect both the search accuracy and computational efficiency in the query process. The training for the prediction model will be elaborated in the next subsection.

The prediction results, together with the query vector, are sent to each client. Each client then executes ANNS based on the estimated search parameters to return the candidate results to the server, including the client ID, local vector IDs $\hat{v}$, and distances $\hat{d}$ with respect to the query vector. After collecting candidates from all clients, the server ranks all samples based on their distances and returns the top k nearest neighbors to the user. This method not only preserves the privacy feature of federated learning by keeping data local but also dynamically adjusts the search scope on clients through the server-side prediction mechanism, reducing redundant computation and achieving accurate and efficient federated ANNS.

The prediction and search algorithm is summarized in Algorithm 1. Server uses the prediction model to estimate the corresponding $nProb_m$ and $kPred_m$ for each client C_m (line

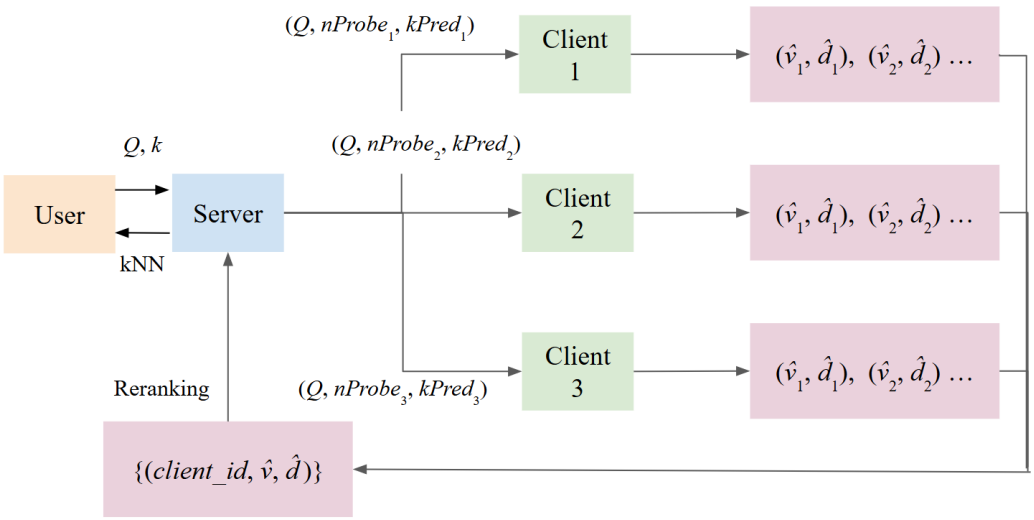

Fig. 1. Overview of the proposed method.

1), and sends these parameters to the respective clients (line 2). Each C_m then searches its local database to retrieve $kPred_m$ nearest neighbors as candidates (line 3–4), including vector identifiers $\hat{v}_{(m,i)}$ and their distances $\hat{d}_{(m,i)}$, where function `IVFSearch` calls the built-in `IndexIVFFlat` method of the Faiss library. Denote the candidate set as $candi_m$. The server aggregates all candidate results returned by the M clients and reranks them based on their distances. Finally the top k vectors with the smallest distances are returned to users as the query response (lines 7–8).

Algorithm 1 Prediction and Search

Require: query Q
Ensure: a set of k nearest neighbors
// Server: predict search parameters for each client
1: $\{nProbe_m, kPred_m \mid m = 1, \ldots, M\} \leftarrow f(Q)$
// Client: execute ANNS based on search parameters
2: **for** each client C_m **do**
3: $\{(\hat{v}_{m,i}, \hat{d}_{m,i}) \mid i = 1, \ldots, kPred_m\} \leftarrow C_m.\texttt{IVFSearch}(Q, nProbe_m, kPred_m)$
4: $candi_m \leftarrow \{(C_m, \hat{v}_{m,i}, \hat{d}_{m,i})\}$
5: return $candi_m$ to server
6: **end for**
// Server: rerank all candidates
7: $kNN \leftarrow \texttt{rerank}(\bigcup_m candi_m, k)$
8: **return** kNN to user

A. Model Training

Typically, performing ANNS in an IVF index structure uses fixed search parameters $nProbe$ and $kPred$ for each query without considering the data distribution between the query and its nearest neighbors. It may cause over-searching in dense regions and under-searching in sparse regions, thereby compromising overall search performance. To address this issue, we propose a machine learning-based prediction model that learns data distributions and IVF index structures of local databases. Through the effective prediction mechanism at the server side, each client can dynamically adjust the IVF search range according to the given query and neighbor size to perform efficient ANNS.

We train the prediction model using a three-layer network architecture, as illustrated in Fig. 2. The model takes a query vector Q and the number of nearest neighbors k as input to predict $nProb_m$ and $kPred_m$ for each client C_m, $m = 1, \ldots, M$. Algorithm 2 lists the training procedure. We adopt a self-supervised learning approach, where the labeled data is generated based on the distribution of nearest neighbors for each training query across all client databases. In other words, each training vector is regarded as a query to retrieve its top-k nearest neighbor vectors $\{v_1^*, v_2^*, \ldots, v_k^*\}$ from the data federation, and we count how many of these vectors are located in each client. Let $kPred_m^*$ be the number of nearest neighbors located in client C_m:

$$kPred_m^* = \left| \{v_i^* \in V_m\} \right| \tag{1}$$

For these neighbors $\{\mathbf{v}_i^* \in V_m\}$, we calculate how many distinct clusters these neighbors belong to, denoted $nProbe_m^*$:

$$nProbe_m^* = \left| \texttt{set}\left(\{\texttt{cluster}(v_i^*) \mid v_i^* \in V_m\}\right) \right| \tag{2}$$

where function $\texttt{set}(\cdot)$ returns the unique elements and function $\texttt{cluster}(\cdot)$ returns the cluster id of the vector (in the IVF index structure). The pair $(nProbe_m^*, kPred_m^*)$ thus serves as the ground truth label for client C_m for the training query.

We define the following loss functions: cluster count prediction loss L_{nProbe} and nearest neighbor count prediction loss L_{kPred}, as shown in Eqs. (3) and (4), respectively. The total loss combines the above two losses with a hyperparameter α, as shown in Eq. (5).

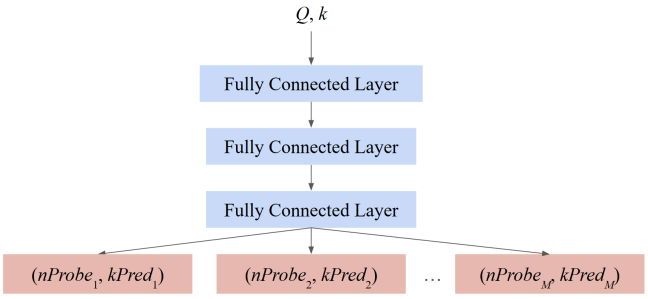

Fig. 2. Neural network architecture of the prediction model.

$$\mathcal{L}_{\text{nProbe}} = \sum_{m=1}^{M} (nProbe_m^* - nProb_m)^2 \tag{3}$$

$$\mathcal{L}_{\text{kPred}} = \sum_{m=1}^{M} (kPred_m^* - kPred_m)^2 \tag{4}$$

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{nProbe}} + \alpha \times \mathcal{L}_{\text{kPred}} \tag{5}$$

B. Privacy Protection

In the federated computing environment, we adopt the semi-honest threat model [7] [14] and assume that data transmission channels are encrypted and trustworthy. Under this security assumption, the main focus of privacy protection is to ensure that the vector content of each client is not exposed to other clients. Therefore, we focus on the following two types of potential security threat scenario:

- Information leakage based on network transmission: Adversaries attempt to infer the distribution characteristics of private data on different clients by analyzing the amount of data exchanged between the server and the clients.
- Information leakage based on access patterns: Adversaries try to deduce the location or content of data by observing memory access patterns during the query process (i.e., where memory locations are accessed).

To address the first threat, we fix the number of candidate results returned by each client to k, ensuring that the size of the transmitted data packets is approximately the same, making it difficult for adversaries to perform traffic analysis. For the second threat, we adopt the following protection mechanism:

- The server executes both the prediction model and the candidate neighbor reranking process within hardware enclaves. Specifically, we utilize Intel SGX (software guard extensions) to ensure the security of sensitive data during in-memory processing. This provides integrity and confidentiality guarantees for the execution environment, preventing even privileged system administrators or potential malware from observing memory access patterns.
- We adopt oblivious sorting techniques [7] during the candidate re-ranking phase. Oblivious sorting accesses memory based solely on the input length rather than the data values. In other words, regardless of the input data, the sequence of comparisons and swaps is fixed and executed consistently, even if the data is already sorted. As a result, the memory access patterns reveal no information about the input values, preventing adversaries from inferring the true distribution of candidate neighbors by analyzing the memory behavior during sorting.

IV. Experimental Result

In this section, we describe the dataset used, experimental setup, and the results of our experiments.

A. Dataset and Setup

The experimental evaluation was carried out using the SIFT1B dataset [17], which is widely used as a standard benchmark to assess the performance of large-scale ANNS. The dataset comprises one billion SIFT (Scale-Invariant Feature Transform) feature vectors, extracted from images to represent local descriptors invariant to changes in scale, rotation, and illumination. Each vector has a dimensionality of 128, with feature values ranging from 0 to 255. A separate query set, consisting of 10,000 query vectors, is provided to evaluate retrieval performance.

To realistically simulate federated environments, we employed two distinct data partitioning schemes: a uniform distribution and a skewed distribution. In the uniform scheme, the dataset is evenly distributed across the federated clients, and each client contains exactly one million vectors. We considered scenarios involving 5, 10, and 15 clients. To establish ground truth labels, we performed brute-force search on each client's dataset using a test query subset comprising 1,000 queries. The ground truth top-1,000 nearest neighbors for each query were obtained globally, and labels were subsequently determined based on the distribution of these retrieved neighbors across the federated clients. For the skewed scheme, the data distribution among federated clients was generated using a Dirichlet distribution with a concentration parameter α = 0.5. This skewed scenario enabled us to analyze the performance of retrieval methods under more realistic and uneven data distribution conditions typically encountered in federated learning environments.

The evaluation metrics used in our experiments include recall and search time to assess the effectiveness of the proposed prediction model. The definition of recall is shown in Eq. (6):

$$recall\ k@k = \frac{|G_k \cap r_k|}{|G_k|} \tag{6}$$

G_k denotes the top k nearest neighbors to the query point and r_k denotes the top k candidates after retrieval and reranking. We use *recall* 100@100, 500@500, 1000@1000 to report the accuracy of the performance.

All experiments were performed on an Intel Core i7-14700 processor (20 cores, 28 threads) and 64 GB RAM. Each local client process was assigned exactly one thread. The proposed method was evaluated against DANN [7], which serves as the baseline for comparison.

TABLE I
UNIFORM DATASET RESULT

Method	#clients	k	network latency (μs)	local search (μs)	aggregation (μs)	total time (μs)	*recall k@k*
DANN	5	100	4	2399	1294	3699	0.8557
		500	11	3360	1869	5240	0.8068
		1000	18	3808	2222	6050	0.7797
	10	100	25	6240	2050	8315	0.8737
		500	18	9208	2311	11538	0.8343
		1000	227	7291	4517	12036	0.8124
	15	100	56	7299	3312	10668	0.8817
		500	27	8198	3012	11238	0.8467
		1000	467	8094	6799	15361	0.8267
Ours	5	100	2	490	15	507	0.7958
		500	4	1046	48	1098	0.9005
		1000	6	1418	86	1510	0.9254
	10	100	3	425	18	446	0.7302
		500	5	848	51	904	0.8781
		1000	8	1190	89	1287	0.9147
	15	100	4	426	19	449	0.6703
		500	6	761	55	822	0.8046
		1000	9	1002	90	1101	0.8929

TABLE II
SKEWED DATASET RESULT

Method	#clients	k	network latency (μs)	local search (μs)	aggregation (μs)	total time (μs)	*recall k@k*
DANN	5	100	7	7651	923	8582	0.8407
		500	14	4263	1444	5721	0.7807
		1000	72	7501	2271	9845	0.7467
	10	100	26	5902	2155	8084	0.8573
		500	23	2657	2536	5217	0.8065
		1000	231	6314	4436	10981	0.7774
	15	100	58	6833	4717	11609	0.8704
		500	35	3608	3592	7236	0.8241
		1000	480	7864	9380	17724	0.7985
Ours	5	100	2	703	18	723	0.8935
		500	6	2831	84	2921	0.9544
		1000	12	4028	143	4183	0.9658
	10	100	3	1126	31	1160	0.8630
		500	9	2997	89	3095	0.9374
		1000	16	4327	140	4483	0.9569
	15	100	5	1448	35	1488	0.8451
		500	11	3434	89	3534	0.9264
		1000	19	4972	148	5139	0.9473

B. Experimental Results

Tables I and II summarize the experimental results under the uniform and skewed data partition schemes, respectively. We compare the performance of our proposed method against the baseline method (DANN) in terms of retrieval accuracy and execution latency.

First, to test the uniform dataset, we vary different numbers of clients and k. The results are presented in Table I. Three different time metrics are defined as follows:

- Network latency: It measures the latency time of data transmission between clients and server. We calculate it by dividing the amount of data transmitted by the network bandwidth. The network bandwidth is set following [7] as 20,000 Mbps divided by the number of clients.
- Local search: It records the time that each client spends on retrieval. In our experiments, it specifically refers to the time taken by each client to perform the search using the IVFFlat index.
- Aggregation: It measures the time the server reranks the result after receiving the returned data from all clients.

According to the results in Table I, our method achieves less retrieval time than DANN in all scenarios. This demonstrates that our approach of using predicted values to determine the search scope effectively reduces redundant data searches. Regarding recall, our method also shows higher accuracy in most cases. We mention that DANN performs better while k = 100 is caused by the well-distributed characteristic in a uniform dataset, which induces prediction errors to accumulate as the number of clients increases, ultimately reducing the accuracy. For other values of k, although prediction errors still exist, the predicted values are large enough to ensure that the search can

retrieve a baseline number of nearest neighbor candidates, thus maintaining the final recall rate.

We also performed the same test on the skewed dataset. As results shown in table II, our method also performs better than DANN on search time. On the other hand, unlike the uniform dataset, our method outperforms DANN in the majority of cases. We believe the distribution of the skewed dataset is more concentrated. Therefore, with the more accurately predicted model, a sufficient number of nearest neighbors can be found within fewer clients, reducing the impact of prediction errors and resulting in improved final recall rates.

V. Conclusions

In this paper, we propose an efficient and privacy-preserving method for ANNS in dense vector data federation. Specifically, our method leverages a predictive model deployed on the server side to dynamically estimate the optimal search scope and the number of candidate vectors that each client should retrieve for a given query. This approach effectively reduces redundant local computations, thereby significantly improving query processing efficiency without compromising retrieval accuracy. Comprehensive experimental evaluations on large-scale vector datasets demonstrate that our proposed approach achieves superior performance in terms of both computational efficiency and retrieval accuracy. Future work may involve further refinement of the search algorithm and the predictive model to enhance retrieval accuracy, particularly in scenarios where fewer nearest neighbors (lower k) are returned.

Acknowledgment

This work was supported by the National Science and Technology Council under grants NSTC 112-2221-E-415-008-MY3 and NSTC 113-2634-F-194-001. The authors also thank the National Center for High-performance Computing (NCHC) of National Applied Research Laboratories (NARLabs) in Taiwan for providing computational and storage resources.

References

[1] F. Rios, P. Rizzo, F. Puddu, *et al.*, "Recommending relevant papers to conference participants: A deep learning driven content-based approach," in *Adjunct Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*, 2022, pp. 52–57.

[2] R. Chen, B. Liu, H. Zhu, *et al.*, "Approximate nearest neighbor search under neural similarity metric for large-scale recommendation," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 3013–3022.

[3] L. Xiong, C. Xiong, Y. Li, *et al.*, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," in *International Conference on Learning Representations*, 2021.

[4] Y. Matsui, "Lotusfilter: Fast diverse nearest neighbor search via a learned cutoff table," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 30 430–30 439.

[5] C. H. Song, H. J. Han, and Y. Avrithis, "All the attention you need: Global-local, spatial-channel attention for image retrieval," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2022, pp. 2754–2763.

[6] Z. Wang, Z. Gao, K. Guo, Y. Yang, X. Wang, and H. T. Shen, "Multilateral semantic relations modeling for image text retrieval," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2830–2839.

[7] X. Zhang, Q. Wang, C. Xu, Y. Peng, and J. Xu, "Fedknn: Secure federated k-nearest neighbor search," in *Proceedings of the ACM on Management of Data (SIGMOD)*, vol. 2, New York: Association for Computing Machinery, 2024, pp. 1–18.

[8] S. Vithana, M. Cardone, and F. P. Calman, "Private approximate nearest neighbor search for vector database querying," in *2024 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2024, pp. 3666–3671.

[9] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[10] Y. Lindell, "Secure multiparty computation," *Communications of the ACM*, vol. 64, no. 1, 2021.

[11] Z. Zhu, Z. Fan, Y. Zeng, *et al.*, "Fedsq: A secure system for federated vector similarity queries," *Proc. VLDB Endow.*, vol. 17, no. 12, pp. 4441–4444, 2024.

[12] C. Li, M. Zhang, D. G. Andersen, and Y. He, "Improving approximate nearest neighbor search through learned adaptive early termination," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '20, Portland, OR, USA: Association for Computing Machinery, 2020, pp. 2539–2554.

[13] A. Prayoonwong, K.-L. Zeng, and C.-Y. Chiu, "Improving nearest neighbor indexing by multitask learning," in *Proceedings of the 19th International Conference on Content-Based Multimedia Indexing*, Graz, Austria, 2022, pp. 71–76.

[14] K. Zhang, Y. Tong, Y. Shi, *et al.*, "Approximate k-nearest neighbor query over spatial data federation," in *Database Systems for Advanced Applications*, Springer Nature Switzerland, 2023, pp. 351–368.

[15] F. Busolin, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and S. Trani, "Early exit strategies for approximate k-nn search in dense retrieval," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, Boise, ID, USA, 2024.

[16] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[17] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg, "Searching in one billion vectors: Re-rank with source coding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011.