FedVS: Towards Federated Vector Similarity Search with Filters

Zeheng Fan
SKLCCSE Lab
Beihang University
Beijing, China
fanzh@buaa.edu.cn

Yuxiang Zeng
SKLCCSE Lab
Beihang University
Beijing, China
yxzeng@buaa.edu.cn

Zhuanglin Zheng
SKLCCSE Lab
Beihang University
Beijing, China
zzlin@buaa.edu.cn

Binhan Yang
SKLCCSE Lab
Beihang University
Beijing, China
yangbh@buaa.edu.cn

Yongxin Tong
SKLCCSE Lab
Beihang University
Beijing, China
yxtong@buaa.edu.cn

Abstract

Vectors are used to represent unstructured data with their embeddings and associated attributes. Similarity search over large-scale vector datasets has gained significant interest from both industry and academia. It aims to identify the k nearest neighbors to a query object from vectors that satisfy a given attribute filter constraint. Despite its popularity, most solutions focus on single-sourced data and overlook the need for vector retrieval across federated datasets. To fill this gap, we introduce a new problem, *federated vector similarity search with filters*, which enables privacy-preserving vector retrieval over multi-sourced data held by mutually untrusted providers. While some solutions can be adapted, they struggle with low recall, excessive search latency, or high communication cost. To address these challenges, we propose FedVS, a privacy-preserving framework enhanced with indexing and pruning based on Trusted Execution Environment (TEE). We also provide a comprehensive theoretical analysis, including complexity, security, and approximation guarantees for recall. Moreover, we deploy our solution over real-world vector databases and conduct extensive experiments. The results demonstrate that our solution outperforms state-of-the-art methods in both effectiveness and efficiency.

CCS Concepts

• **Information systems** → **Nearest-neighbor search**; **Data federation tools**; **Combination, fusion and federated search**.

Keywords

Similarity Search, Vector Retrieval, Nearest Neighbor Search

ACM Reference Format:
Zeheng Fan, Yuxiang Zeng, Zhuanglin Zheng, Binhan Yang, and Yongxin Tong. 2025. FedVS: Towards Federated Vector Similarity Search with Filters. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3736958

KDD Availability Link:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.15504203.

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3736958

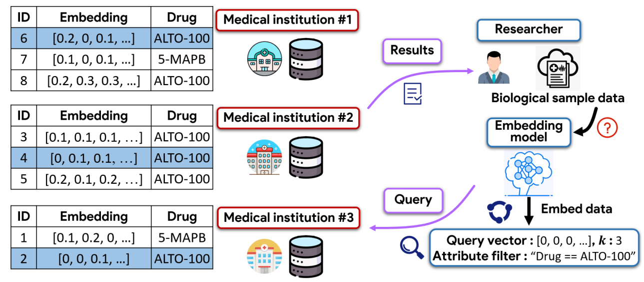


Figure 1: Vector similarity search over federated datasets

1 Introduction

Similarity search has been studied in various areas, such as data mining, databases, and information retrieval [15, 25, 61]. The development of Retrieval-Augmented Generation (RAG) techniques [14, 28] has spurred a new line of research in similarity search, known as **vector similarity search** [43, 74] or nearest neighbor search with filters [26, 34]. This new search paradigm is inspired by a hybrid data type (*i.e.*, vector) that integrates both high-dimensional embeddings and structured attributes. By specifying a query vector and a filter constraint on structured attributes, it identifies k objects from large-scale datasets based on two criteria: (1) their attributes must match the filter and (2) they are the k nearest neighbors (kNNs) to the query vector within the set of filtered data objects.

Vector Similarity Search over Single-Sourced Data. Both industry and academia have developed efficient solutions to vector similarity search. For example, industrial systems for vector databases [45, 61], vector retrieval engines [7, 9, 65], and knowledge graphs [43] have offered robust supports to this query. Recent research [26, 34, 63, 67, 68, 74, 78] has also proposed diverse indexes and optimization methods aimed at balancing search time and answer recall. However, these solutions focus on single-sourced vector data and cannot address the *challenge* involved in searching across multi-sourced datasets (*a.k.a.* **federated datasets**).

Vector Similarity Search over Federated Dataset. With the enactment of data protection regulations (*e.g.*, GDPR [59] and CCPA [3]), vector similarity search over federated datasets needs to simultaneously consider *effectiveness, efficiency*, and *privacy*. As shown in Fig. 1, three medical institutions provide collaborative drug development [41], and a researcher wants to investigate a specific drug test using biological sample data. Using embeddings of this biological sample and the drug type as a filter, vector similarity search over federated datasets can efficiently retrieve relevant drug test results without compromising the data privacy of each medical institution during the search. Other application scenarios include federated RAG [13, 64, 76], joint financial risk assessment [2], cross-platform recommendation system [50], etc.

To perform vector similarity search in these scenarios, each data provider can initially identify k candidates from their local datasets. However, to derive the final answer, they cannot directly share these candidate objects with each other, as this would compromise their data privacy. Instead, additional privacy protection must be adopted, which can inevitably impact the effectiveness or efficiency. Thus, the **main challenge** is how to strike a balance between effectiveness and efficiency while ensuring privacy preservation.

Limitations of Alternative Methods. Despite the absence of dedicated studies on this problem, existing methods for federated kNN search [23, 51, 73, 75, 76] can potentially be extended to address this challenge. These methods adopt either encryption [33] or secure multi-party computation [27] to securely find kNNs to a given query object. However, encryption-based methods [23, 76] are computationally expensive, as they require encrypting the whole dataset and performing searches over encrypted vectors. The other methods [51, 73, 75], which were originally designed for 2D locations or sequence data, exhibit inefficiency or low recall when handling high-dimensional vectors (see our experiments in Sec. 4).

Our Solution. To address these limitations, we propose a new two-phase framework called FedVS. Both phases leverage a dedicated hardware, Trusted Execution Environment (TEE) [39], to protect data privacy during the searches. In Phase I, each provider submits to the TEE the discrete distribution of distances from their initial candidates to the query vector. Then, TEE derives a distance threshold for each provider to effectively remove numerous far-away candidates. In Phase II, the refined candidates from all providers are collected by TEE to securely determines the final answer. Moreover, when multi-sourced data is non-IID, attribute filters can easily make each provider's contribution to the final answer highly unbalanced. Thus, assuming uniformly k initial candidates at each provider may result in redundant computations. To tackle this issue, we also devise a lightweight index to pre-estimate each provider's contribution before performing any search.

Contribution. In summary, our main contributions are as follows:

- To the best of our knowledge, this is the first work to study federated vector similarity search with filters.
- To solve this problem, we propose a privacy-preserving framework and further enhance its efficiency through optimizations based on indexing and pruning.
- We present a comprehensive theoretical study of our solution, covering the approximation guarantee for recall, time and communication complexity, and security analysis.
- We conduct experiments on four benchmark datasets, competing against six baselines extended from state-of-the-art methods [51, 73, 75]. The evaluations are deployed on the industrial vector database Milvus [5]. In the experiments, our solution outperforms all baselines by a large margin.

2 Problem Statement

This section first introduces the key concepts used throughout the paper and then formally defines the studied problem.

2.1 Basic Concepts

Unstructured data objects are often represented as vectors using their embeddings and associated attributes [26, 34, 43, 63, 67, 74].

Definition 1 (Vector Data). *A vector data object v ("vector" as short) usually consists of two main components:*

*(i) **Embedding** is denoted by a point $v.e = (e_1, e_2, \cdots, e_d) \in \mathbb{R}^d$ in a d-dimensional space, where each e_i represents the i-th coordinate.*

*(ii) **Attributes** are represented by a set of c structured attributes $v.a = (a_1, a_2, \cdots, a_c)$ associated with this object, where each attribute a_i can be either numerical or categorical data.*

The dateset $\mathcal{D}$ denotes a collection of such vectors that share the same embedding space and attribute schema.

The *embedding* captures the intrinsic features of the corresponding entity in a continuous space, while the structured *attributes* provide additional context or metadata associated with the entity. These components complement each other, making this data type highly effective for representing unstructured information. As a result, it has been widely adopted in vector databases [45, 61], vector retrieval engines [7, 9, 65], and knowledge graphs [43].

In these systems, a *distance function* $\text{dist}(\cdot, \cdot)$ quantifies the similarity between two embeddings, while an *attribute filter* restricts vectors based on specific search criteria for their attributes.

Definition 2 (Attribute Filter). *An attribute filter ("filter" as short) is represented by a conjunctive boolean predicate $P = p_1 \wedge p_2 \wedge \cdots \wedge p_h$. Each condition p_i is a binary comparison statement in the form $v.a_i \odot const_i$, where $\odot$ is one of the comparison operators from $\{\leq, \geq, <, >, =\}$ and $const_i$ is a constant.*

A vector v satisfies the attribute filter if and only if the predicate $P(v)$ evaluates to true, meaning all conditions p_i are satisfied:

$$P(v) = true \iff \forall i \in [1, h], p_i(v.a_i) = true \tag{1}$$

Example 1. *Each medical institution in Fig. 1 manages a vector dataset with each drug test result containing an embedding of biological data and a drug type. To search for relevant drug test results, a researcher can specify a attribute filter like "Drug == ALTO-100".*

Vector data are ubiquitous and multi-sourced. Inspired by federated learning [18, 69, 70], we focus on large-scale vector data distributed across multiple data providers, *i.e.*, federated dataset [38, 51, 55, 75] defined in Def. 3 as follows.

Definition 3 (Federated Dataset). *A federated dataset F consists of m data providers, each holding a vector dataset $\mathcal{D}_i$ with the same data schema. These data providers collaboratively provide a vector retrieval service over their union dataset $\mathcal{D} = \bigcup \mathcal{D}_i$.*

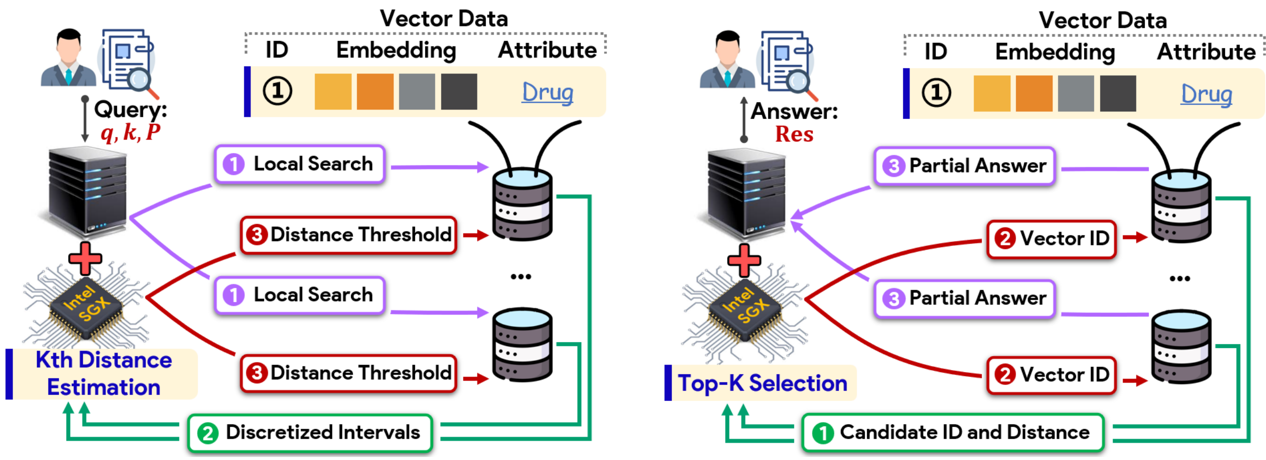
Due to data protection regulations, competitive concerns, or the need to protect business secrets, these data providers are prohibited from directly sharing their dataset without any protection.

2.2 Problem Definition

Based on above concepts, we formally define the Federated Vector Similarity Search with filters (FVSS) problem as follows:

Definition 4 (Federated Vector Similarity Search with Filters). *Given a federated dataset F, a query vector q, a positive integer k, and an attribute filter P, this problem aims to retrieve k data objects, denoted as Res, that are the most similar to q from the vectors in F satisfying the filter P. In other words, the result Res should meet the following two constraints:*

- ***Filter constraint**: For any vector $v \in \mathsf{Res}$, its attributes must satisfy the predicates in the filter P, i.e., $P(v) = true$.*



(a) Phase I: Federated candidate refinement

(b) Phase II: Federated top-k selection

Figure 2: Illustration of our framework FedVS

- ***kNN constraint***: *Let $\mathcal{D}^-$ denotes the set of vectors satisfying the filter constraint. Then,* Res *is a collection of k Nearest Neighbors (kNNs) of q in $\mathcal{D}^-$, i.e.,*

$$\forall v \in \text{Res}, \forall o \in (\mathcal{D}^- \setminus \text{Res}), \text{dist}(v, q) \leq \text{dist}(o, q) \tag{2}$$

Additionally, data privacy must be protected during the search process, and the ***security constraints*** *include:*

(i) The query user can only learn query results Res*, without any knowledge about private data from the data providers.*

(ii) The data providers can only learn information derived from their own vector data such as distance and ids, with no access to other providers' private data.

Attacker Model. Following common assumptions in previous studies [51, 60, 75], we assume the attackers are semi-honest [27]. Under this model, the query user and data providers will faithfully execute the designated search algorithm but may attempt to infer as much private information as possible during the retrieval.

Example 2. *A toy example of the FVSS problem is illustrated in Fig. 1. Suppose we use Euclidean distance to measure the similarity between embeddings. This FVSS query aims to find $k = 3$ nearest neighbors to the query vector $q = [0, 0, \cdots, 0]$ among three medical institutions' records that match the filter "Drug == ALTO-100". Results are highlighted in blue in the left tables of Fig. 1.*

Remark. The high dimensionality of embeddings makes fulfilling the kNN constraint more susceptible to the "curse of dimensionality" [56]. Consequently, recent solutions [14, 16, 19, 45, 47, 48, 73] for vector retrieval have shifted focus towards approximate methods rather than exact solutions. Motivated by this trend, we primarily focus on approximate solutions that maximize the recall of query answers Res relative to exact results Exact, defined as Eq. (3):

$$\text{recall} = \frac{|\text{Exact} \bigcap \text{Res}|}{|\text{Exact}|} = \frac{|\text{Exact} \bigcap \text{Res}|}{k} \tag{3}$$

3 Our Framework FedVS

This section introduces an efficient and secure framework FedVS for the FVSS problem. Specifically, Sec. 3.1 first provides an overview of secure primitives that we use. Then, Sec. 3.2 presents the general framework along with its theoretical analysis. Finally, Sec. 3.3 and 3.4 elaborate on our optimizations (with detailed pseudo-code in Appendix A).

3.1 Preliminary of Security Basics

Privacy-enhancing techniques, homomorphic encryption [12] and secure multi-party computation [27], are widely used in federated learning [18, 70] or federated queries [49, 55]. However, these techniques are computationally intensive, which can significantly reduce search efficiency when utilized to maintain security.

By contrast, our framework leverages a hardware-assisted Trusted Execution Environment (TEE) [39], which has received growing attention for its potential to offer both security and scalability. TEE offers a secure and isolated area within the CPU and memory, where private data can be processed with strong confidentiality guarantees. Intel's SGX [66] is one of the leading industrial products of TEEs. As shown in Fig. 2, the central server of the vector retrieval service is equipped with SGX. This dedicated hardware facilitates efficient processing of private data from providers.

3.2 General Framework: FedVS

Leveraging Intel's SGX [66], we devise a two-phase framework for the FVSS problem. In the following, we first introduce the main idea, then delve into each phase, and finally analyze its recall approximation, complexity, and security guarantee.

3.2.1 Main Idea. Our framework is structured into two phases:

(i) Federated Candidate Refinement. This phase securely estimates the kth nearest distance to q and reduces the number of local candidates at each data provider to fewer than k.

(ii) Federated Top-K Selection. This phase securely picks the top-k nearest vectors to q from the refined candidates.

In both phases, plaintext operations at local vector databases are accelerated using efficient vector indexes like HNSW [5, 34, 63]. Secure operations involving private data from multiple providers are isolated within a hardware-supported TEE like Intel's SGX [66].

3.2.2 Phase I: Federated Candidate Refinement. Fig. 2a illustrates this phase. Specifically, each provider performs local search at their own dataset to obtain k candidates. These candidates' distance distributions are then represented using discretized intervals. Finally, SGX estimates the upper bound of the kth nearest distance to the query vector q through binary-search across all providers' intervals.

Lines 1–17 of Alg. 1 detail this procedure with two key steps:

(i) Partition Initial Candidates' Distances. In line 2, each provider i retrieves initial candidates $cand_i$ from their local dataset $\mathcal{D}_i$ using pre-built vector indexes. These candidates are sorted based on their distances to q in ascending order. Lines 4–7 divide the sorted sequence of k distances into $\sqrt{k}$ intervals T_i, where each interval is denoted by the minimum and maximum distances within it. Then, each provider sends T_i to SGX through a secure channel.

(ii) Estimate kth Nearest Distance. Lines 9–16 estimate the kth nearest distance γ to q among all candidates $\{cand_i\}$ via binary search. Initially, l and u are set as the lower and upper bounds of γ, respectively. For each possible value r to estimate γ, line 12 computes the index z_i of the interval from T_i that covers r. Considering intervals up to z_i as candidates results in $z_i \cdot \sqrt{k}$ candidates for provider i. If the total number of such candidates reaches k, the upper bound u is decreased to r; otherwise, the lower bound l is increased to r. Line 15 derives the *global upper bound* $\widetilde{\gamma}$ of γ by setting it to the maximum right endpoint of intervals in each T_i covering u. To prevent information leakage, line 16 derives the corresponding *local upper bound* $\widetilde{\gamma}_i$ for provider i. Finally, SGX informs each provider with the distance threshold $\widetilde{\gamma}_i$.

3.2.3 Phase II: Federated Top-K Selection. Fig. 2b illustrates the main process of this phase, corresponding to lines 18–28 of Alg. 1.

In lines 18–21, each data provider i removes candidates whose distances to q exceed their received threshold $\widetilde{\gamma}_i$ and submits the remaining distances $\mathcal{L}_i$ back to SGX through a secure channel.

Lines 22–26 use an m-sized min-heap to determine the number of partial answers K_i from provider i that will be included in the final result Res. Initially, this heap Q is populated with the head (shortest) distance from each $\mathcal{L}_i$. Then, in lines 24–26, Q is popped k times, each time extracting the current shortest distance d^* from provider i^*. This indicates that the jth nearest neighbor to q comes from provider i^*, so the next candidate from provider i^* is pushed into Q. Finally, SGX informs each provider i to submit their local K_i nearest neighbors to q and collects these vectors into Res.

3.2.4 Theoretical Analysis. Next, we analyze the recall approximation, complexity, and security guarantee of our framework.

Recall Analysis. To prove the recall guarantee, we first establish Lemma 1 and 2 to demonstrate the correctness of each phase.

Lemma 1. *In Alg. 1, Phase I ensures that the k nearest neighbors to q among all providers' initial candidates will not be removed.*

Proof. Suppose Cand $= \bigcup cand_i$. We can proof that if each data provider only discards vector data that $\mathrm{dist}(q, v) > \widetilde{\gamma}$, then the k nearest neighbors to q among Cand will not be removed. We prove this by *contradiction*. Suppose that one of the k nearest neighbors to q, denoted as v^*, is removed during phase II which initially comes from $cand_i$. Then we have $\mathrm{dist}(v^*, q) > \widetilde{\gamma}$. Additionally, denote right endpoint of the interval covering r from T_i as γ_i, we have $\gamma_i \le \widetilde{\gamma}$. it's calculated in *Alg.* 1 that there exists z_i intervals with the right endpoint lower than γ_i which indicates that at least $z_i \cdot \sqrt{k}$ vectors' distance to q is lower than γ_i among $cand_i$. Thus there exists at least $\sum_{i=1}^{m} (z_i \cdot \sqrt{k}) \ge k$ vector data with distance to q lower than

Algorithm 1: Our framework FedVS

Input: federated dataset F and vector search (q, k, P)
Output: search result Res
`// Phase I: Federated Candidate Refinement`
1 **foreach** *data provider* $i \leftarrow 1$ **to** m **do** // Perform in parallel
2 $\quad cand_i \leftarrow$ vector similarity search (q, k, P) locally in $\mathcal{D}_i$;
3 $\quad$ Sort candidates $cand_i$ based on their distances to q;
4 $\quad$ **foreach** *distance interval* $j \leftarrow 1$ **to** $\sqrt{k}$ **do**
5 $\quad\quad v_j \leftarrow (\sqrt{k}(j-1)+1)$th vector in $cand_i$;
6 $\quad\quad u_j \leftarrow (\sqrt{k}(j-1)+\sqrt{k})$th vector in $cand_i$;
7 $\quad\quad$ Append interval $[\mathrm{dist}(v_j, q), \mathrm{dist}(u_j, q)]$ to set T_i;
8 SGX receives intervals T_i from provider i;
9 $l \leftarrow 0, u \leftarrow$ longest distance among intervals in $\{T_i\}$;
10 **while** $u > l$ **do** // Binary-search in SGX
11 $\quad r \leftarrow (l+u)/2$;
12 $\quad z_i \leftarrow$ binary-search interval in each T_i that covers r;
13 $\quad$ **if** $\sum_{i=1}^{m} (z_i \cdot \sqrt{k}) \ge k$ **then** Upper bound $u \leftarrow r$;
14 $\quad$ **else** Lower bound $l \leftarrow r$;
15 $\widetilde{\gamma} \leftarrow \max\{$right endpoint of interval in T_i covering $u\}$;
16 $\widetilde{\gamma}_i \leftarrow$ right endpoint of interval in T_i covering $\widetilde{\gamma}$;
17 SGX sends distance threshold $\widetilde{\gamma}_i$ to ith data provider;
`// Phase II: Federated Top-K Selection`
18 **foreach** *data provider* $i \leftarrow 1$ **to** m **do** // Perform in parallel
19 $\quad$ Remove any vector $v \in cand_i$ such that $\mathrm{dist}(v, q) > \widetilde{\gamma}_i$;
20 $\quad \mathcal{L}_i \leftarrow$ sort candidates' distances $\{\mathrm{dist}(v, q) \mid v \in cand_i\}$;
21 SGX receives sorted distances $\mathcal{L}_i$ from provider i;
22 Heap $Q \leftarrow$ pop the head distance from each $\mathcal{L}_i$;
23 K_i maintains #(partial answers) in Res from provider i;
24 **foreach** $j \leftarrow 1$ **to** k **do** // Top-K in SGX
25 $\quad$ Pop shortest distance d^* from provider i^* out of Q;
26 $\quad K_{i^*} \leftarrow K_{i^*} + 1$, push next distance from $\mathcal{L}_{i^*}$ into Q;
27 SGX sends non-negative integer K_i to data provider i;
28 **return** Res $\leftarrow$ collect K_i nearest vectors from provider i;

$\widetilde{\gamma}$. As $\mathrm{dist}(v^*, q) > \widetilde{\gamma}$, we derive that there exists at least k vector data closer to q than v^*. This corollary leads to a contradiction. Therefore, Lemma 1 holds. □

Lemma 2. *In Alg. 1, Phase II ensures that the k nearest neighbors to q among all providers' remaining candidates will be selected.*

Proof. We also prove the Lemma 2 by *contradiction*. Suppose there exists $v^- \in$ Res and $v^* \notin$ Res with $\mathrm{dist}(v^-, q) > \mathrm{dist}(v^*, q)$. Suppose v^* is in Q when v^- is popped. Then, we know $\mathrm{dist}(v^*, q) \ge \mathrm{dist}(v^-, q)$, as Q is a min-heap. Suppose v^* is in $\mathcal{L}_j$ as v^- is popped. There must exist a vector v_0^* that satisfies $\mathrm{dist}(v_0^*, q) \le \mathrm{dist}(v^*, q)$ and $\mathrm{dist}(v_0^*, q) \in Q$ according to line 26 in Alg. 1. As v^- is the closest vector to q in Q, thus $\mathrm{dist}(v^*, q) \ge \mathrm{dist}(v^-, q)$ which leads to a contradiction. Therefore, phase II will exactly choose the k nearest neighbors to q from all providers' remaining candidates. □

Based on these lemmas, Theorem 1 establishes the approximation guarantees for the answer recall of our framework.

Theorem 1. *If the initial candidates $cand_i$ are obtained by vector search at data provider i with recall rate δ_i ($\delta_i \in [0,1]$), the overall recall rate of Alg. 1 is at least* $\min_i \delta_i$.

Proof. To derive a meaningful worst-case recall, our proof relies on a mild assumption: for each provider i, their respective k nearest vectors kNN_i to q appear in the initial candidates $cand_i$ with uniform probability ρ_i. This assumption broadly aligns with the randomness inherent in existing approximate solutions for vector retrieval [16]. Consequently, the number $X_i = |kNN_i \cap cand_i|$ follows a binomial distribution with success probability ρ_i, and its expectation is $\mathbb{E}[X_i] = k \cdot \rho_i$. The prerequisite of this theorem ensures that $X_i \geq k \cdot \delta_i$, indicating $\rho_i \geq \delta_i$ (for each provider i).

In the worst case, all vectors in the exact answer Exact come from the provider with the lowest ρ_i. Then, Lemma 1 and Lemma 2 ensure that $(k \cdot \min_i \rho_i)$ vectors in the search result Res are also contained in Exact. Thus, the overall recall is at least $\min_i \delta_i$. □

Practical Implication. Theorem 1 aligns with the bucket effect [46]: a bucket's total capacity is mainly determined by its shortest board. To achieve high recall, each provider should therefore adopt an effective solution (*e.g.*, Milvus [5]) for local vector search.

Complexity Analysis. Complexity is analyzed from two aspects:

(i) Computational Time: Let T denote the time cost for local vector retrieval. In Phase I, lines 1–7 take $O(T + k\log k)$ time, and lines 8–17 take $O(m \log d_{max} \log k)$ time, where d_{max} is the maximum candidate distance. Phase II takes $O(m + k\log m)$ time. Thus, we derive that the overall time complexity is $O(T + k\log k + k\log m + m\log d_{max} \log k)$.

(ii) Communication Overhead: In Phase I, SGX receives $2m\sqrt{k}$ interval endpoints. In Phase II, it receives $\sum_i |\mathcal{L}_i|$ distances and k vectors, each with d dimensions and c attributes. Thus, the overall communication overhead is $O(m\sqrt{k} + \sum_i |\mathcal{L}_i| + (c+d)k)$.

Security Guarantee. Alg. 1 satisfies the security constraints:

(i) The query user receives only the search result containing exactly k vectors and learns no additional information.

(ii) All private data is processed within a hardware-enabled TEE, Intel SGX [66], ensuring robust security. Each data provider knows only the distance threshold defined by their own candidates, and there is no communication between providers. This ensures that no information leakage occurs between data providers.

3.3 Reducing Communication Overhead

In the communication overhead, the median term $\sum_i |\mathcal{L}_i|$ can be approaching mk in the worst case. To mitigate this, we propose an optimized method for the SGX procedure in Phase I of Alg. 1.

3.3.1 Main Idea. We optimize lines 8–17 of Alg. 1 from two aspects:

(i) Simplified Representation for Intervals. Since distance thresholds $\widetilde{\gamma_i}$ are determined solely by right endpoints of intervals, providers no longer send the left endpoints. This simplification reduces the communication cost of interval transmission by 50%.

(ii) Tighter Distance Threshold. To achieve a tighter distance threshold, we replace the binary search in lines 9–15 with a min-heap based search. This m-sized heap Q^* maintains the minimum right endpoint across all intervals from m providers. With this change, $\sum_i |\mathcal{L}_i|$ is reduced from mk to $k + m\sqrt{k}$ in the worst case.

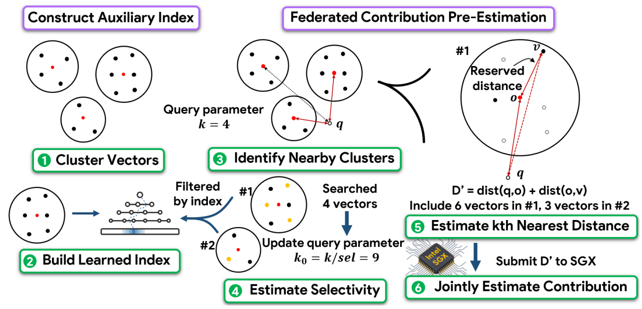


Figure 3: Illustration of contribution pre-estimation

3.3.2 Algorithm Details. In Phase I of Alg. 1, lines 1–7 and 16–17 remain unchanged under this optimization.

The major changes occur in lines 8–15. Specifically, in line 8, each provider i sends only the right endpoints of intervals T_i to SGX. Lines 9–15 are replaced with the following heap based search. An m-sized min-heap Q^* is initialized with the first right endpoint from each provider. The heap is then processed by performing $\sqrt{k}$ pops. Whenever a right endpoint is popped from Q^*, we immediately refill Q^* with the next right endpoint from the same provider, except for the last pop. Finally, the last right endpoint popped from Q^*, which belongs to provider i^*, determines the distance upper bound $\widetilde{\gamma}$.

Remark. This heap-based search takes $O((m + \sqrt{k})\log m)$ time. Consequently, it *reduces the time complexity* of Phase I into $O(T + k\log k + (m+\sqrt{k})\log m)$. Moreover, the communication cost $\sum_i |\mathcal{L}_i|$ depends on the number of right endpoints that have been inserted into Q^*. This number comprises the popped endpoints (*i.e.*, $\sqrt{k}$) and remaining endpoints in Q^* (*i.e.*, $m - 1$). Thus, this optimization also *reduces the communication cost* into $O(k + m\sqrt{k})$. Additionally, optimized framework can achieve the same recall as Alg. 1, which will be explained in Appendix A.1.

3.4 Pruning via Contribution Pre-Estimation

Both existing solutions [51, 73, 75] and our Alg. 1 select k nearest vectors to q as initial candidates at each provider. However, due to the non-IID property of federated data [37], only a few providers make meaningful contributions to the final answer. Leveraging the non-IIDness, we pre-estimate each provider's contribution to the final answer and eliminate redundant candidates.

3.4.1 Main Idea. Our estimation considers two primary factors:

(i) Distance to Query Vector. Providers with shorter nearest distance to query vectors tend to contribute more in final answers.

(ii) Selectivity of Attribute Filter. Providers whose local datasets exhibit higher selectivity for attribute filters are more likely to make major contributions to the final answer.

To derive these information without conducting local vector searches, we devise a lightweight and learning-enhanced index in Sec. 3.4.2 and propose an effective estimation method in Sec. 3.4.3.

3.4.2 Construct Auxiliary Index. We propose a Cluster-based Learned Index, called CLI, to hold distance information and structured attributes in each provider's local dataset with two key steps:

(i) Cluster Vectors. First, we adopt a balanced clustering algorithm over embeddings to partition vectors $\mathcal{D}_i$ into multiple clusters $\{C_j\}$. For each cluster C_j, we sort its vectors based on their distances to the centroid o_j in ascending order.

(ii) Build Learned Index. For vectors within each cluster C_j, we construct a multi-dimensional learned index [30] (*e.g.*, PGM-index [8]) over their structured attributes. Besides, we perform systematic sampling on computed distances to the centroid and store these distances at intervals of $\sqrt{|C_j|}$ (*i.e.*, $\sqrt{|C_j|}$, $2\sqrt{|C_j|}$, $\cdots$, up to $|C_j|$) within our index. This index CLI facilitates the estimation of the *selectivity* and the kth nearest *distance* to q, as will be detailed later.

3.4.3 Federated Contribution Pre-Estimation. To collaboratively pre-estimate each provider's contribution to the final answer (denoted by number k_i), our solution involves four essential steps:

(i) Identify Nearby Clusters. Given a local dataset partitioned into Φ clusters $\{C_1, C_2, \cdots, C_\Phi\}$, each provider identifies the cluster whose centroid o_i is the closest to q, and selects additional nearby clusters for estimation. The selected clusters C^* are defined as:

$$C^* = \left\{ C_j \mid \mathrm{dist}(o_j, q) \le (1+\alpha) \cdot \min_{i=1}^{\Phi} \mathrm{dist}(o_i, q) \right\} \tag{4}$$

where the parameter $\alpha \in [0, 1]$ tunes the threshold for determining whether a cluster is considered sufficiently nearby.

(ii) Estimate Selectivity. In this step, we map the conjunctive predicate $p_1 \wedge p_2 \wedge \cdots \wedge p_h$ of the filter P into a multi-dimensional search window. We then execute range counting searches within each cluster $C_j \in C^*$ using the pre-built multi-dimensional learned index [30]. For cluster C_j, cnt_j denotes the exact count, and $\widetilde{cnt}_j$ is the range count through learned index. The selectivity denoted as sel can be estimated as follows:

$$\begin{aligned} sel &= \frac{\left| \left\{ v \mid v \in C_j \bigwedge C_j \in C^* \bigwedge P(v) = \mathrm{true} \right\} \right|}{\sum_{C_j \in C^*} |C_j|} \\ &= \frac{\sum_{C_j \in C^*} cnt_j}{\sum_{C_j \in C^*} |C_j|} \approx \frac{\sum_{C_j \in C^*} \widetilde{cnt}_j}{\sum_{C_j \in C^*} |C_j|} \end{aligned} \tag{5}$$

The approximation in Eq. (5) holds due to the bounded worse-case error of PGM-index [31], indicating the accuracy of our estimation.

(iii) Estimate kth Nearest Distance. Based on the selectivity in Eq. (5), the top $\frac{k}{sel}$ nearest vectors in C^* to the query vector q are expected to contain enough vectors satisfying the filter constraints. For a vector v in the cluster $C_j \in C^*$, the upper bound on the distance between v and q is derived by the triangle inequality:

$$\mathrm{dist}(q, v) \le \mathrm{dist}(q, o_j) + \mathrm{dist}(o_j, v) \tag{6}$$

Our index CLI has stored distances between $\sqrt{k}$ sampled vectors v_i and centroid o_j. Thus, we only need to find the smallest distance γ^* such that the conditions in Eq. (7) and Eq. (8) are met:

$$\forall C_j \in C^*, z_j = \arg\min_{v_i \in C_j} \left\{ i \mid \gamma^* \le \mathrm{dist}(q, o_j) + \mathrm{dist}(o_j, v_i) \right\} \tag{7}$$

$$\frac{k}{sel} \le \sum_{C_j \in C^*} \left(z_j \cdot \sqrt{|C_j|} \right) \tag{8}$$

To efficiently compute γ^*, we adopt the aforementioned method in Sec. 3.3 with two key modifications: the distance upper bound as defined in Eq. (6) and diversified interval sizes $\sqrt{|C_j|}$.

(iv) Jointly Estimate Contribution. Each provider i submits their estimated kth nearest distance γ_i^* to SGX. Intuitively, providers with smaller γ_i^* are likely to contribute more significantly in the final answer. To retain high recall, the provider with the minimum γ_i^* remains with k initial candidates. Accordingly, SGX estimates the other providers' contributions as in Eq. (9), and sends integer $k_i \le k$ to each provider for subsequent local vector retrieval.

Table 1: Statistics of datasets (distance function: L_2)

Dataset	Card.	Dim.	Embedding	Attribute	Partition
WIT	5×10^4	2048	Image	Image Size	IID
YT-Audio	10^6	128	Audio	Category	Dirichlet
YT-Rgb	10^6	1024	Video	Category	Dirichlet
DEEP	10^7	96	Image	Synthetic	Quantity

$$k_i = k \cdot \frac{\min_i \gamma_i^*}{\gamma_i^*} \tag{9}$$

4 Experimental Study

We deploy our experimental study on six cloud servers over industrial vector databases, Milvus v2.5.2 [5]. The main hardware includes Intel Xeon(R) Platinum 8361HC CPUs and 32GB of RAM. One server is equipped with Intel's SGX SDK. They are interconnected with a public network bandwidth of up to 10Mbit/s.

4.1 Experimental Setup

Dataset. We adopt four real-world datasets from prior studies [26, 67, 68, 78]: WIT [10], YT-Audio [11], YT-Rgb [11], and DEEP [1]. These datasets feature cardinalities of up to 10 million vectors and dimensionalities of up to 2048. Each vector in the first three datasets has a single attribute, while each vector in DEEP includes two attributes. To test both IID and non-IID scenarios, we allocate datasets into providers using various partition methods in federated learning [17, 24, 29, 42]: (1) WIT is uniformly divided; (2) YT-Audio and YT-Rgb are partitioned based on Dirichlet distributions with parameter $\beta = 0.5$; (3) DEEP employs a classic quantity-based partition [37]. We generate the query workloads by following previous researches [26, 68, 78] for vector similarity search with filters.
Parameter Setting. We evaluate the impacts of query parameter k ranging from 32 to 256 and the number of providers m ranging from 5 to 20. The default values of k and m are 128 and 5, respectively.
Compared Solution. We make two types of extensions to three state-of-the-art methods, HuFu [51], Mr [73], and DANN* [75], to solve our FVSS problem. These baselines mainly rely on indexes which are inefficient for high-dimensional vector data and attribute filter. In our first extension, we primarily substitute their local multi-dimensional indexes with the same dedicated vector indexes as in our solution. In the other extension, We use a "post-filter" strategy to extend these methods as HuFu-Post, Mr-Post, and DANN*-Post. Explicitly, we first identify k nearest vectors without considering filter constraint through Milvus in each data provider, then verify filter constraint with one round of scanning, thus yielding potentially fewer than k initial candidates. Additionally, we implement a plaintext baseline that selects the top-k nearest neighbors to q in plaintext from the initial mk candidates. To ensure a fair comparison, we implement all the solutions in C++ and utilize gRPC v1.62.0 [4] for network communications.
Metric. The above methods are compared from three metrics: (1) *Answer recall* represents accuracy of search results relative to ground truths; (2) *Search time* quantifies average time for performing a federated vector similarity search with filters; (3) *Communication cost* is network traffic generated during the search procedure. We also report our index *construction time* and *size* in Sec. 4.4.2.

Table 2: Query performance of our solution (FedVS) compared to one plaintext baseline (Plaintext) and six secure baselines, with answer recall (%), communication cost (KB), and search time (ms) as metrics (↑: higher is better, ↓: lower is better). Among secure solutions, the best result is marked in blue and the runner-up performance is underlined.

Compared Algorithms	WIT Dataset [10]			YT-Audio Dataset [11]			YT-Rgb Dataset [11]			DEEP Dataset [1]		
	Answer Recall ↑	Comm. Cost ↓	Search Time ↓	Answer Recall ↑	Comm. Cost ↓	Search Time ↓	Answer Recall ↑	Comm. Cost ↓	Search Time ↓	Answer Recall ↑	Comm. Cost ↓	Search Time ↓
Plaintext	99.66	990	186.05	98.59	72	48.84	96.17	538	95.91	99.00	54	88.51
HuFu	93.42	15200	9871.37	97.62	1064	3991.67	96.15	8062	7602.56	97.00	804	4187.00
Mr	83.98	15281	5579.43	66.35	1006	652.78	81.87	8050	4064.27	95.16	791	768.44
DANN*	96.48	5703	4491.06	92.76	407	801.50	90.34	3173	2910.62	92.71	165	905.46
HuFu-Post	24.60	11024	10825.16	16.57	592	5369.70	18.32	4535	6679.56	24.03	467	5175.60
Mr-Post	8.35	10833	4870.95	18.06	566	549.09	20.94	4449	1957.18	4.88	370	578.88
DANN*-Post	8.00	5643	4342.08	22.94	371	757.08	23.95	2749	2505.94	4.62	278	811.33
FedVS	**99.63**	**997**	**362.25**	**98.38**	**72**	**305.29**	**96.16**	**541**	**304.70**	**99.00**	**55**	**348.19**

Table 3: Ranking of secure solutions based on their overall query performance

Metric	Average rank across four datasets
Search time (from shortest to longest)	FedVS < Mr-Post < DANN*-Post < Mr < DANN* < HuFu < HuFu-Post
Answer Recall (from highest to lowest)	FedVS > HuFu > DANN* > Mr > HuFu-Post > DANN*-Post > Mr-Post

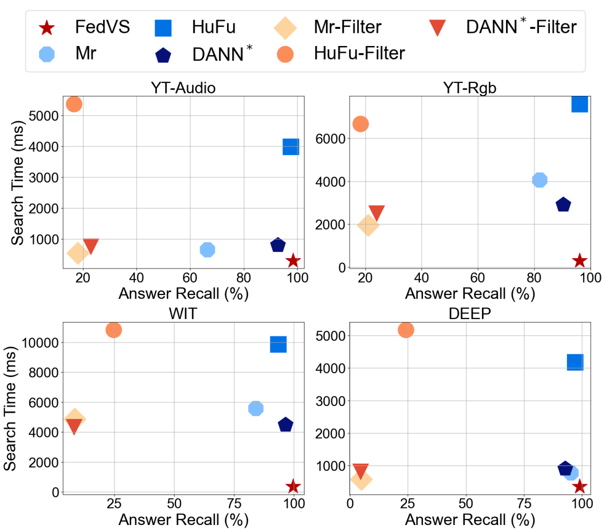


Figure 4: Answer recall vs. search time on four datasets

4.2 Overall Query Performance

Table 2 presents the overall query performance. From these results, we have made the following observations.

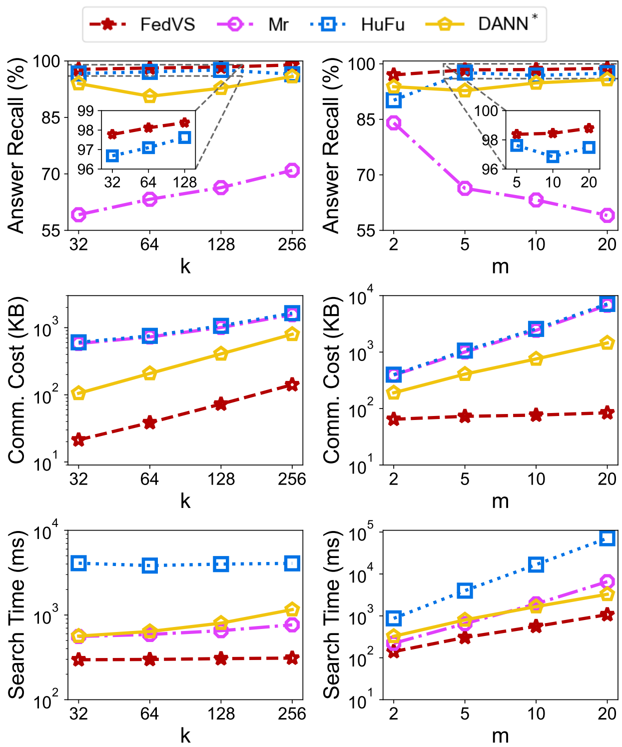
Result of Recall. We first observe that our solution FedVS consistently achieves the highest recall among secure solutions. Across four datasets, the recall of FedVS is up to 6.21%, 32.03%, and 6.29% higher than HuFu, Mr, and DANN*, respectively. Even compared with insecure baseline Plaintext, FedVS decreases the answer recall by up to 0.21% in YT-Audio, 0.03% in WIT, and only 0.01% in YT-Rgb while maintaining the same recall in DEEP. Another observation is that the baselines implemented with "post-filter" strategy generally have lower recall. This is because the post-filtering may reduce the answer size to less than k. These results also indicate that HuFu, Mr, and DANN* are strong competitors in terms of recall.

Result of Efficiency. In terms of efficiency, our solution takes the shortest search time and lowest communication cost across all secure baselines. Specifically, the communication cost of our FedVS is 5.65–15.32×, 5.15–14.77×, 5.08–14.90×, and 3.05–14.61× lower than the six secure baselines on WIT, YT-Audio, YT-Rgb, and DEEP datasets, respectively. Additionally, FedVS is up to 27.25×, 15.40×, and 12.39× faster than HuFu, Mr, and DANN*, respectively. Compared to the insecure baseline, existing secure baselines are at least 6.54× slower, while FedVS is at most 6.25× slower. Moreover, the communication cost of FedVS is close to that of Plaintext, whereas other secure methods require at least 3.05× more communications.

Besides, Fig. 4 is the scatter diagram of each secure method based on its average results of 100 queries on four datasets. The closer a method is to the bottom-right corner, the better its performance. Based on this diagram, we can easily obtain the ranks of these secure solutions in terms of search time and recall on each dataset, which is shown in Table 3. According to this overall rank, our solution FedVS outperforms all existing baselines in terms of both effectiveness and efficiency across the datasets.

4.3 Impact of Query Parameters

The following experiment evaluates the performance of *each secure solution* under varying query parameter settings for integer k and the number of data providers m. We exclude HuFu-Post, Mr-Post, and DANN*-Post from these comparisons, because their recall is significantly lower than that of HuFu, Mr, and DANN*. Due to page limitations, we only present the results on the YT-Audio dataset here and please refer to the other results in Appendix B.



(a) Impact of k (b) Impact of m

Figure 5: Impact of query parameters on YT-Audio dataset

Impact of Query Parameter k. When varying k in Fig. 5a, the *recall* of FedVS and HuFu almost remains unchanged while the recall of DANN* shows minor fluctuations. By contrast, the recall of Mr increases as k grows. Regarding the changes in the value of k, our solution always achieves the highest recall. For instance, the recall of FedVS is up to 2.46%, 38.63%, and 7.50% higher than that of HuFu, Mr, and DANN*, respectively. This improvement demonstrates the robustness of our solution in the effectiveness.

In terms of *query efficiency*, the communication overhead and search time of any method generally increase as k increases. This is reasonable, since a larger k implies more nearest neighbors in the result set, thereby requiring higher computational and communication cost. The baselines, Mr and HuFu, often have higher communication overhead and longer search latency than DANN* and FedVS. Overall, our solution still requires the shortest search time and lowest communication cost. It is up to 25.18×, 15.72×, and 14.60× faster than HuFu, Mr, and DANN*, respectively.

Impact of #(Data Providers) m. When involving more data providers in Fig. 5b, FedVS maintains a relatively stable *recall* between 97.00% and 98.79% while the recall of HuFu and DANN* exhibit certain fluctuations in 90.16%–97.62% and 92.76%–95.77%, respectively. By comparison, the recall of Mr drops dramatically as m increases. This may be because its contribution evaluation algorithm may introduce larger errors with more data providers. Under different settings of m, HuFu always ranks first in terms of recall, and the improvement over the runner-up method is 0.76%–3.18%.

The *communication cost* and *search time* increase with a growing number of data providers m. This trend complies with the computational and communication complexity of these secure methods,

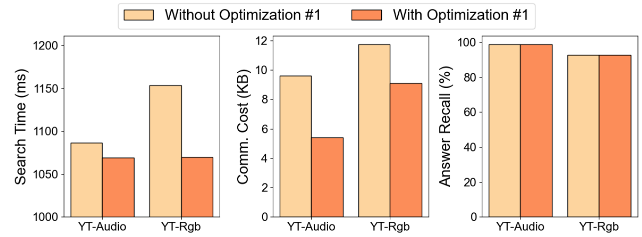


Figure 6: Results of ablation study on optimization #1

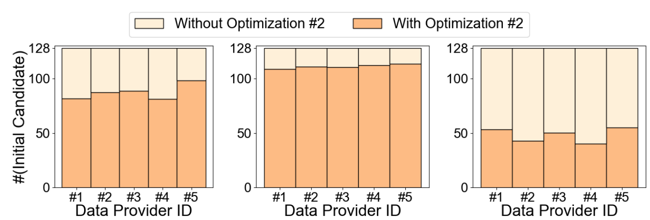
where a larger m requires more secure computations across data providers. Despite these changes, our FedVS is always the most efficient solution. The baselines, HuFu, Mr, and DANN*, are up to 66.59×, 6.12×, and 3.06× slower than FedVS, respectively. Moreover, when m grows from 2 to 20, the communication cost of FedVS only increases by 29.96%, while that of others increases by up to 17.88×.

4.4 Ablation Study

The following ablation studies assess the effectiveness of our optimization methods introduced in Sec. 3.3 and Sec. 3.4.

4.4.1 Optimization #1: Reducing Communication Overhead. This ablation experiment assesses the effectiveness of the optimization method described in Sec. 3.3 ("optimization #1" as short). Fig. 6 shows the query performance of our framework with and without optimization #1, using the YT-Audio and YT-Rgb datasets partitioned into twenty data providers. The communication cost reported here excludes the input query and output answer, as they remain constant regardless of whether this optimization is applied.

Using optimization #1, our framework effectively reduces both the search time and communication overhead. For example, 7.32% search time and 22.33% communication cost are saved in the YT-Rgb dataset. Meanwhile, the recall remains unchanged. These results clearly validate the functionality of this optimization.



(a) YT-Audio (b) YT-Rgb (c) DEEP

Figure 7: Results of ablation study on optimization #2

4.4.2 Optimization #2: Pruning via Contribution Pre-Estimation. To evaluate our pruning strategy proposed in Sec. 3.4, we conduct another ablation experiment on YT-Audio, YT-Rgb, and DEEP datasets with five data providers. Since optimization #2 aims to reduce the initial candidate size, we directly report the sizes with and without this pruning strategy in our framework FedVS.

Impact on Initial Candidate Size. As shown in Fig. 7, optimization #2 can reduce the initial candidate size at each provider by up to 15.19%–68.56% in these datasets with minimal impact on recall. Then, each local vector database now needs to search fewer local candidates and hence has lower computational cost.

Table 4: Construct our auxiliary index CLI

Dataset	YT-Audio	YT-Rgb	DEEP
Clustering Time	28s	150s	6778s
Index Build Time	31ms	18ms	1789ms
Index Space Cost	17KB	51KB	346KB

Additional Cost for Auxiliary Index. Table 4 presents the average cost for our auxiliary index CLI. The DEEP dataset contains 10 million vectors, making it significantly larger than the other datasets. Thus, we generate 100 clusters for DEEP and 10 clusters for the other datasets using the method in [40]. Other more efficient high-dimensional data clustering algorithms [21, 32, 35] are orthogonal to our index. After clustering, it takes less than 2 seconds and 1MB space to build an auxiliary index in each provider. This demonstrates space and time efficiency of building the index.

4.5 Summary of Major Findings

The key findings from the experiments are summarized as follows:

- Among the secure solutions, our solution FedVS always achieves both the highest recall and the best efficiency across the datasets. Specifically, the recall of FedVS is up to 6.21%–32.03% higher than that of HuFu, Mr, and DANN*, respectively. Meanwhile, using our method, the communication cost can be reduced by up to 15.32× and search time can be saved by up to 27.25×.
- When using different query parameters, such as the size of result set and the number of data providers, the query performance of FedVS demonstrates greater robustness than baselines. It consistently ranks first in recall, communication cost, and search time. This highlights its superior performance under various conditions.
- Our candidates refinement and contribution pre-estimation algorithms effectively reduce the initial candidates, thus taking less communication cost than other secure solutions. Moreover, HuFu and Mr require multi-round secure multi-party operations, eg. comparisons and summations. DANN* performs secure top-k over more candidates, resulting in higher time and communication complexity than ours.
- Among the state-of-the-art baselines adapted for our problem, HuFu is often more accurate than Mr and DANN*, while Mr and DANN* exhibit better efficiency than HuFu.

5 Related Work

We review related work from the following two categories.

Vector Similarity Search with Filters. Prior studies on similarity search have predominantly focused on computing exact or approximate kNN search [19, 36, 44, 47, 48, 72]. Recently, vector data is commonly used to represent unstructured data objects with their embeddings and associated attributes [16, 61]. This hybrid data type has spurred several studies [26, 34, 43, 63, 67, 74] into a new form of similarity search: *vector similarity search with filters.*

Existing studies can be classified into three kinds: *pre-filter*, *post-filter*, and *hybrid index*. *Pre-filter* solutions [62, 65] refine the vector data by attribute filters before selecting kNNs among refined vectors. Conversely, *post-filter* solutions [26, 71, 74] first identify kNNs from the entire dataset and then verify them using filters. *Hybrid index* based methods either fuse both embeddings and attributes into a single distance function before indexing [63, 67], or they design indexes with hybrid structures to store embeddings and attributes [34, 43, 68, 78]. However, they all focus on singled-sourced data.

Federated kNN Search. Inspired by federated learning [18, 69, 70] and privacy-preserving data mining [20, 57, 58], federated kNN search has been studied in various applications, such as enhancing RAG with multi-sourced data [23, 64, 76, 77], collaborative gene searching [75], and cross-platform spatial crowdsourcing [52, 53].

To prevent privacy leakage between providers, most studies employ either encryption [23, 76] or secure multi-party computation [51, 54, 73] during the kNN search. When handling high-dimensional vectors, these solutions can be computationally expensive. Among these studies, DANN* [75] leverages distance lower bounds [39] to accelerate secure computations. However, their original solutions do not support vector similarity search with filters.

Summary. Although vector similarity search with filters is widely supported in industrial vector databases (*e.g.*, Pinecone [6], Milvus [5], and Qdrant [7]), existing work still offers limited supports for this emerging type of vector retrieval over federated datasets. This gap motivates us to propose a dedicated solution FedVS.

Furthermore, our *optimization via contribution pre-estimation* (Sec. 3.4) is different from *contribution estimation in federated learning (FL)* [22]. In the latter, contribution estimation in FL quantifies each participant's impact during collaborative training [22]. By contrast, our method aims to estimate each provider's contribution in the final result prior to the local search process.

6 Conclusion

Motivated by real-world application needs, this work introduces a new problem called federated vector similarity search with filters. This problem aims to identify kNNs to a query vector under an attribute filter constraint from multi-source vector datasets. Existing solutions are either inefficient or inaccurate to address this problem. To overcome these limitations, we propose a two-phase framework FedVS and devise two optimizations via indexing and pruning. We also analyze the recall guarantee, computational and communication complexity, and security. Extensive experiments demonstrate that our solution achieves consistently better query performance than state-of-the-art methods. Overall, FedVS accelerates search time by up to 27.25× and reduces communication overhead by up to 15.32×, while maintaining the highest recall.

Acknowledgments

We are sincerely grateful to anonymous reviewers for their constructive comments. This work was partially supported by National Key Research and Development Program of China under Grant No. 2023YFF0725103, National Science Foundation of China (NSFC) (Grant Nos. 62425202, U21A20516, 62336003), the Beijing Natural Science Foundation (Z230001), the Fundamental Research Funds for the Central Universities No. JK2024-03, the Didi Collaborative Research Program and the State Key Laboratory of Complex & Critical Software Environment (SKLCCSE). Yuxiang Zeng and Yongxin Tong are the corresponding authors.

References

[1] 2023. The DEEP Dataset. https://big-ann-benchmarks.com/neurips23.html

[2] 2024. Applying Vector Databases in Finance for Risk and Fraud Analysis. https://zilliz.com/learn/applying-vector-databases-in-finance-for-risk-and-fraud-analysis

[3] 2024. California Consumer Privacy Act (CCPA). https://oag.ca.gov/privacy/ccpa

[4] 2024. gRPC. https://grpc.io/

[5] 2024. Milvus. https://milvus.io

[6] 2024. Pinecone. https://www.pinecone.io/

[7] 2024. Qdrant. https://qdrant.tech/

[8] 2024. The PGM-Index. https://pgm.di.unipi.it/

[9] 2024. Weaviate. https://weaviate.io/

[10] 2024. WIT dataset. https://github.com/google-research-datasets/wit

[11] 2024. YouTube Dataset. https://research.google.com/youtube8m/download.html

[12] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2018. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Comput. Surv.* 51, 4 (2018), 79:1–79:35.

[13] Parker Addison, Minh-Tuan H. Nguyen, Tomislav Medan, Jinali Shah, Mohammad T. Manzari, Brendan McElrone, Laksh Lalwani, Aboli More, Smita Sharma, Holger R. Roth, Isaac Yang, Chester Chen, Daguang Xu, Yan Cheng, Andrew Feng, and Ziyue Xu. 2024. C-FedRAG: A Confidential Federated Retrieval-Augmented Generation System. *CoRR* abs/2412.13163 (2024).

[14] Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. Retrieval-based Language Models and Applications. In *ACL*. 41–46.

[15] Gérard Biau and Luc Devroye. 2015. *Lectures on the Nearest Neighbor Method.* Springer.

[16] Sebastian Bruch. 2024. *Foundations of Vector Retrieval.* Springer.

[17] Daoyuan Chen, Dawei Gao, Weirui Kuang, Yaliang Li, and Bolin Ding. 2022. pFL-Bench: A Comprehensive Benchmark for Personalized Federated Learning. In *NeurIPS*.

[18] Jingxue Chen, Hang Yan, Zhiyuan Liu, Min Zhang, Hu Xiong, and Shui Yu. 2024. When Federated Learning Meets Privacy-Preserving Computation. *ACM Comput. Surv.* 56, 12 (2024), 319:1–319:36.

[19] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. 2021. SPANN: Highly-efficient Billion-scale Approximate Nearest Neighborhood Search. In *NeurIPS*. 5199–5212.

[20] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. 2002. Tools for Privacy Preserving Data Mining. *SIGKDD* 4, 2 (2002), 28–34.

[21] Vincent Cohen-Addad, Alessandro Epasto, Silvio Lattanzi, Vahab Mirrokni, Andres Muñoz Medina, David Saulpic, Chris Schwiegelshohn, and Sergei Vassilvitskii. 2022. Scalable Differentially Private Clustering via Hierarchically Separated Trees. In *KDD*. 221–230.

[22] Yue Cui, Chung-ju Huang, Yuzhu Zhang, Leye Wang, Lixin Fan, Xiaofang Zhou, and Qiang Yang. 2024. A Survey on Contribution Evaluation in Vertical Federated Learning. *CoRR* abs/2405.02364 (2024).

[23] Yichao Du, Zhirui Zhang, Bingzhe Wu, Lemao Liu, Tong Xu, and Enhong Chen. 2023. Federated Nearest Neighbor Machine Translation. In *ICLR*.

[24] Jean Ogier du Terrail, Samy-Safwan Ayed, Edwige Cyffers, Felix Grimberg, Chaoyang He, Regis Loeb, Paul Mangold, Tanguy Marchand, Othmane Marfoq, Erum Mushtaq, Boris Muzellec, Constantin Philippenko, Santiago Silva, Maria Telenczuk, Shadi Albarqouni, Salman Avestimehr, Aurélien Bellet, Aymeric Dieuleveut, Martin Jaggi, Sai Praneeth Karimireddy, Marco Lorenzi, Giovanni Neglia, Marc Tommasi, and Mathieu Andreux. 2022. FLamby: Datasets and Benchmarks for Cross-Silo Federated Learning in Realistic Healthcare Settings. In *NeurIPS*.

[25] Karima Echihabi, Themis Palpanas, and Kostas Zoumpatianos. 2021. New Trends in High-D Vector Similarity Search: AI-driven, Progressive, and Distributed. *PVLDB* 14, 12 (2021), 3198–3201.

[26] Joshua Engels, Benjamin Landrum, Shangdi Yu, Laxman Dhulipala, and Julian Shun. 2024. Approximate Nearest Neighbor Search with Window Filters. In *ICML*.

[27] David Evans, Vladimir Kolesnikov, and Mike Rosulek. 2018. A Pragmatic Introduction to Secure Multi-Party Computation. *Foundations and Trends in Privacy and Security* 2, 2-3 (2018), 70–246.

[28] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *SIGKDD*. 6491–6501.

[29] Tiantian Feng, Digbalay Bose, Tuo Zhang, Rajat Hebbar, Anil Ramakrishna, Rahul Gupta, Mi Zhang, Salman Avestimehr, and Shrikanth Narayanan. 2023. FedMultimodal: A Benchmark for Multimodal Federated Learning. In *SIGKDD*. 4035–4045.

[30] Paolo Ferragina, Fabrizio Lillo, and Giorgio Vinciguerra. 2020. Why Are Learned Indexes So Effective?. In *ICML*, Vol. 119. 3123–3132.

[31] Paolo Ferragina and Giorgio Vinciguerra. 2020. The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds. *PVLDB* 13, 8 (2020), 1162–1175.

[32] Yujian Fu, Cheng Chen, Xiaohui Chen, Weng-Fai Wong, and Bingsheng He. 2024. Optimizing the Number of Clusters for Billion-Scale Quantization-Based Nearest Neighbor Search. *IEEE Trans. Knowl. Data Eng.* 36, 11 (2024), 6786–6800.

[33] Oded Goldreich. 2001. *The Foundations of Cryptography - Volume 1: Basic Techniques.* Cambridge University Press.

[34] Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. 2023. Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. In *WWW*. 3406–3416.

[35] Chaoyu Gong, Yongbin Liu, Di Fu, Yong Liu, Pei-hong Wang, and Yang You. 2022. Self-reconstructive evidential clustering for high-dimensional data. In *ICDE*. 2099–2112.

[36] Shunsuke Kanda and Yasuo Tabei. 2020. Dynamic Similarity Search on Integer Sketches. In *ICDM*. 242–251.

[37] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated Learning on Non-IID Data Silos: An Experimental Study. In *ICDE*. 965–978.

[38] Shuyuan Li, Yuxiang Zeng, Yuxiang Wang, Yiman Zhong, Zimu Zhou, and Yongxin Tong. 2024. An Experimental Study on Federated Equi-Joins. *IEEE Trans. Knowl. Data Eng.* 36, 9 (2024), 4443–4457.

[39] Xiaoguo Li, Bowen Zhao, Guomin Yang, Tao Xiang, Jian Weng, and Robert H. Deng. 2023. A Survey of Secure Computation Using Trusted Execution Environments. *CoRR* abs/2302.12150 (2023).

[40] Hongfu Liu, Ziming Huang, Qi Chen, Mingqin Li, Yun Fu, and Lintao Zhang. 2018. Fast Clustering with Flexible Balance Constraints. In *BigData*. 743–750.

[41] Dandan Lu, Ming Li, Yi Liao, Guihua Tao, and Hongmin Cai. 2022. Verifiable privacy-preserving queries on multi-source dynamic DNA datasets. *IEEE Transactions on Cloud Computing* 11, 2 (2022), 1927–1939.

[42] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*, Vol. 54. 1273–1282.

[43] Jason Mohoney, Anil Pacaci, Shihabur Rahman Chowdhury, Ali Mousavi, Ihab F. Ilyas, Umar Farooq Minhas, Jeffrey Pound, and Theodoros Rekatsinas. 2023. High-Throughput Vector Similarity Search in Knowledge Graphs. *SIGMOD* 1, 2 (2023), 197:1–197:25.

[44] Marius Muja and David G. Lowe. 2009. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *VISAPP*. 331–340.

[45] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Survey of vector database management systems. *VLDB J.* 33, 5 (2024), 1591–1615.

[46] Laurence J. Peter and Raymond Hull (Eds.). 2011. *The Peter Principle: Why Things Always Go Wrong.* Harper Business.

[47] Ninh Pham and Tao Liu. 2022. Falconn++: A Locality-sensitive Filtering Approach for Approximate Nearest Neighbor Search. In *NeurIPS*.

[48] Parikshit Ram and Kaushik Sinha. 2019. Revisiting kd-tree for Nearest Neighbor Search. In *SIGKDD*. 1378–1388.

[49] Donghyun Sohn, Xiling Li, and Jennie Rogers. 2024. Everything You Always Wanted to Know About Secure and Private Database Systems (but were Afraid to Ask). *IEEE Data Eng. Bull.* 47, 2 (2024), 3–20.

[50] Zehua Sun, Yonghui Xu, Yong Liu, Wei He, Lanju Kong, Fangzhao Wu, Yali Jiang, and Lizhen Cui. 2025. A Survey on Federated Recommendation Systems. *IEEE Trans. Neural Networks Learn. Syst.* 36, 1 (2025), 6–20.

[51] Yongxin Tong, Xuchen Pan, Yuxiang Zeng, Yexuan Shi, Chunbo Xue, Zimu Zhou, Xiaofei Zhang, Lei Chen, Yi Xu, Ke Xu, and Weifeng Lv. 2022. Hu-Fu: Efficient and Secure Spatial Queries over Data Federation. *PVLDB* 15, 6 (2022), 1159–1172.

[52] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. 2016. Online mobile Micro-Task Allocation in spatial crowdsourcing. In *ICDE*. 49–60.

[53] Yongxin Tong, Yuxiang Zeng, Bolin Ding, Libin Wang, and Lei Chen. 2021. Two-Sided Online Micro-Task Assignment in Spatial Crowdsourcing. *IEEE Trans. Knowl. Data Eng.* 33, 5 (2021), 2295–2309.

[54] Yongxin Tong, Yuxiang Zeng, Yang Song, Xuchen Pan, Zeheng Fan, Chunbo Xue, Zimu Zhou, Xiaofei Zhang, Lei Chen, Yi Xu, Ke Xu, and Weifeng Lv. 2025. Hu-Fu: efficient and secure spatial queries over data federation. *VLDB J.* 34, 2 (2025), 19.

[55] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Boyi Liu, Yexuan Shi, Shuyuan Li, Ke Xu, and Weifeng Lv. 2023. Federated Computing: Query, Learning, and Beyond. *IEEE Data Eng. Bull.* 46, 1 (2023), 9–26.

[56] Csaba D. Toth, Joseph O'Rourke, and Jacob E. Goodman (Eds.). 2017. *Handbook of Discrete and Computational Geometry, Third Edition.* Chapman and Hall/CRC.

[57] Jaideep Vaidya and Chris Clifton. 2004. Privacy-Preserving Data Mining: Why, How, and When. *IEEE Secur. Priv.* 2, 6 (2004), 19–27.

[58] Jaideep Vaidya, Yu Zhu, and Christopher W. Clifton. 2006. *Privacy Preserving Data Mining.* Advances in Information Security, Vol. 19. Springer.

[59] Paul Voigt and Axel Von dem Bussche. 2017. *The EU General Data Protection Regulation (GDPR): A Practical Guide.* Vol. 10. Springer International Publishing.

[60] Nikolaj Volgushev, Malte Schwarzkopf, Ben Getchell, Mayank Varia, Andrei Lapets, and Azer Bestavros. 2019. Conclave: secure multi-party computation on big data. In *EuroSys*. 3:1–3:18.

[61] Jianguo Wang, Eric Hanson, Guoliang Li, Yannis Papakonstantinou, Harsha Simhadri, and Charles Xie. 2024. Vector Databases: What's Really New and

What's Next? *PVLDB* 17, 12 (2024), 4505–4506.

[62] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *SIGMOD*. 2614–2627.

[63] Mengzhao Wang, Lingwei Lv, Xiaoliang Xu, Yuxiang Wang, Qiang Yue, and Jiongkang Ni. 2023. An Efficient and Robust Framework for Approximate Nearest Neighbor Search with Attribute Constraint. In *NeurIPS*.

[64] Shuai Wang, Ekaterina Khramtsova, Shengyao Zhuang, and Guido Zuccon. 2024. FeB4RAG: Evaluating Federated Search in the Context of Retrieval Augmented Generation. In *SIGIR*. 763–773.

[65] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. 2020. AnalyticDB-V: A Hybrid Analytical Engine Towards Query Fusion for Structured and Unstructured Data. *PVLDB* 13, 12 (2020), 3152–3165.

[66] Newton Carlos Will and Carlos Alberto Maziero. 2023. Intel Software Guard Extensions Applications: A Survey. *ACM Comput. Surv.* 55, 14s (2023), 322:1–322:38.

[67] Wei Wu, Junlin He, Yu Qiao, Guoheng Fu, Li Liu, and Jin Yu. 2022. HQANN: Efficient and Robust Similarity Search for Hybrid Queries with Structured and Unstructured Constraints. In *CIKM*. 4580–4584.

[68] Yuexuan Xu, Jianyang Gao, Yutong Gou, Cheng Long, and Christian S. Jensen. 2024. iRangeGraph: Improvising Range-dedicated Graphs for Range-filtering Nearest Neighbor Search. *CoRR* abs/2409.02571 (2024).

[69] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* 10, 2 (2019), 12:1–12:19.

[70] Xuefei Yin, Yanming Zhu, and Jiankun Hu. 2022. A Comprehensive Survey of Privacy-preserving Federated Learning: A Taxonomy, Review, and Future Directions. *ACM Comput. Surv.* 54, 6 (2022), 131:1–131:36.

[71] Shangdi Yu, Joshua Engels, Yihao Huang, and Julian Shun. 2023. PECANN: Parallel Efficient Clustering with Graph-Based Approximate Nearest Neighbor Search. *CoRR* abs/2312.03940 (2023).

[72] Haoyu Zhang and Qin Zhang. 2020. MinSearch: An Efficient Algorithm for Similarity Search under Edit Distance. In *SIGKDD*. 566–576.

[73] Kaining Zhang, Yongxin Tong, Yexuan Shi, Yuxiang Zeng, Yi Xu, Lei Chen, Zimu Zhou, Ke Xu, Weifeng Lv, and Zhiming Zheng. 2023. Approximate k-Nearest Neighbor Query over Spatial Data Federation. In *DASFAA*. 351–368.

[74] Qianxi Zhang, Shuotao Xu, Qi Chen, Guoxin Sui, Jiadong Xie, Zhizhen Cai, Yaoqi Chen, Yinxuan He, Yuqing Yang, Fan Yang, Mao Yang, and Lidong Zhou. 2023. VBASE: Unifying Online Vector Similarity Search and Relational Queries via Relaxed Monotonicity. In *OSDI*. 377–395.

[75] Xinyi Zhang, Qichen Wang, Cheng Xu, Yun Peng, and Jianliang Xu. 2024. FedKNN: Secure Federated k-Nearest Neighbor Search. *SIGMOD* 2, 1 (2024), V2mod011:1–V2mod011:26.

[76] Dongfang Zhao. 2024. FRAG: Toward Federated Vector Database Management for Collaborative and Secure Retrieval-Augmented Generation. *CoRR* abs/2410.13272 (2024).

[77] Zeqi Zhu, Zeheng Fan, Yuxiang Zeng, Yexuan Shi, Yi Xu, Mengmeng Zhou, and Jin Dong. 2024. FedSQ: A Secure System for Federated Vector Similarity Queries. *PVLDB* 17, 12 (2024), 4441–4444.

[78] Chaoji Zuo, Miao Qiao, Wenchao Zhou, Feifei Li, and Dong Deng. 2024. SeRF: Segment Graph for Range-Filtering Approximate Nearest Neighbor Search. *SIGMOD* 2, 1 (2024), 69:1–69:26.

A Detailed Pseudo-Code for Our Optimizations

A.1 Optimization #1: Reducing Communication Overhead

Alg. 2 presents all the technical details of the optimization method described in Sec. 3.3. This optimization primarily focuses on Phase I of our framework FedVS.

After local vector similarity search and candidates sorting from lines 2–3, each provider now only needs to send $\sqrt{k}$ distances to SGX which are denoted as the right endpoints of the intervals in Phase I of Alg. 1. Then, SGX calculates the distance threshold with a min-heap Q^*. In line 8, each head (shortest) distance from T_i is pushed into Q^*. Then, lines 10–12 illustrate the process of $\sqrt{k}$ rounds of pop and insert as described in Sec. 3.3.2.

This optimization also satisfies Lemma 1, since the min-heap based selection process is similar to Phase II of our framework. In Alg. 2, we can exactly pop $\sqrt{k}$ shortest distances from $\{T_i\}$ during lines 10–12 according to Lemma 2. Each distance from $\{T_i\}$ represents the right endpoint of a $\sqrt{k}$-sized distance interval. Thus, at least k vectors are remained during the candidate refinement with final threshold $\widetilde{\gamma}$ and denote as candidates of k nearest neighbors to q. In other words, the threshold $\widetilde{\gamma}$ is guaranteed to be the upper bound of the kth nearest distance to the query vector q.

Algorithm 2: Reducing Communication Overhead

Input: federated dataset F and vector search (q, k, P)
Output: distance threshold $\widetilde{\gamma}_i$ for each data provider
// **Phase I: Federated Candidate Refinement**
1 **foreach** *data provider* $i \leftarrow 1$ **to** m **do** // Perform in parallel
2 $\quad cand_i \leftarrow$ vector similarity search (q, k, P) locally in $\mathcal{D}_i$;
3 $\quad$ Sort candidates $cand_i$ based on their distances to q;
4 $\quad$ **foreach** *distance interval* $j \leftarrow 1$ **to** $\sqrt{k}$ **do**
5 $\quad\quad v_j \leftarrow \sqrt{k} \cdot j$th vector in $cand_i$;
6 $\quad\quad$ Append $\mathrm{dist}(v_j, q)$ to set T_i;
7 SGX receives set T_i from provider i;
8 Min-heap $Q^* \leftarrow$ pop the head distance from each T_i;
9 t_i denotes id of the distance to be popped from provider i;
10 **foreach** $j \leftarrow 1$ **to** $\sqrt{k}$ **do**
11 $\quad \widetilde{\gamma} \leftarrow$ pop shortest distance d^* from provider i^* out of Q^*;
12 $\quad t_{i^*} \leftarrow t_{i^*} + 1$, push next distance from T_{i^*} into Q^*;
13 $\widetilde{\gamma}_i \leftarrow$ the shortest distance from T_i not smaller than $\widetilde{\gamma}$;
14 SGX sends distance threshold $\widetilde{\gamma}_i$ to ith data provider;

Discussion. With this optimization, our FedVS can be extended to prevent memory access-pattern disclosure, which serves as one of the most common vulnerabilities for TEE hardware. As both federated top-k selection and optimized candidate refinement relies on a min-heap for search. We can replace the min-heap with an oblivious priority queue, which can protect the interval variables like size of heap during process, thus preventing access-pattern leakage when performing our FedVS.

A.2 Optimization #2: Pruning via Contribution Pre-Estimation

In the following, we present the technical details of the optimization method described in Sec. 3.4 from two aspects: *auxiliary index construction* and *federated contribution pre-estimation.*

Auxiliary Index Construction (Pre-processing). Alg. 3 illustrates the details of constructing our auxiliary index CLI. First, each data provider generates Φ clusters in line 2. Then, they build multi-dimensional learned index for each cluster with its structured attributes in lines 3–4. For enhancing accuracy for the subsequent contribution pre-estimation, $\sqrt{k}$ distances between vectors and the centroid of each cluster are reserved in lines 5–8. The index is pre-processed before performing any vector retrieval.

Federated Contribution Pre-Estimation. Alg. 4 illustrates our federated contribution pre-estimation algorithm from four steps. Firstly, we choose the nearby cluster(s) through Eq. (4) in line 2. Then, by using pre-built CLI, we calculate the number of data objects that satisfy the attribute filter and derive the selectivity in line 3. Lines 5–15 estimate the upper bound of the top $\frac{k}{sel}$ distances

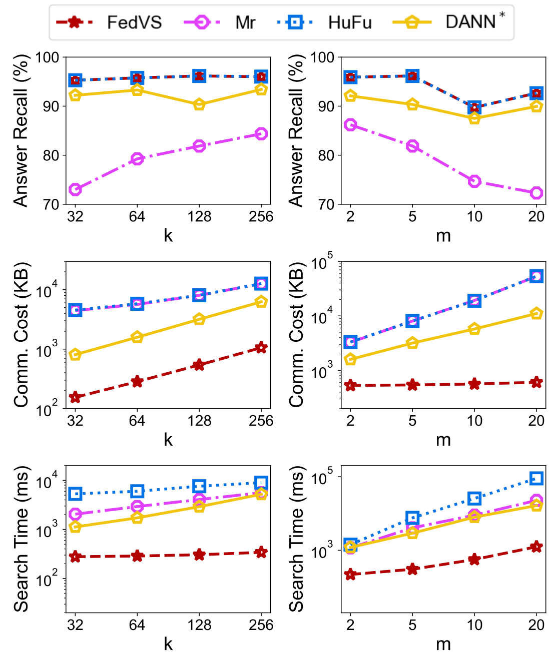
Algorithm 3: Construct Auxiliary Index

Input: data federation F, clusters' number Φ
Output: auxiliary indexes
// Pre-processing
foreach *data provider* $i \leftarrow 1$ **to** m **do** // Perform in parallel
  $\{C_1, C_2, \ldots, C_\Phi\} \leftarrow$ clustering $\mathcal{D}_i$ into Φ clusters;
  foreach $j \in [1, \Phi]$ **do**
    $index_j \leftarrow$ build multi-dimensional learned index on structured attributes from C_j;
    foreach $v \in C_j$ **do** // o_j is the centroid of C_j
      $R_j = R_j \cup \mathrm{dist}(o_j, v)$;
    sort R_j in ascending order;
    store the $\sqrt{|C_j|}th, 2\sqrt{|C_j|}th, \ldots, |C_j|th$ distance and $|C_j|$ locally;

Algorithm 4: Federated Contribution Pre-Estimation

Input: data federation F, a FVSS query (q, k, P)
Output: pruned query parameter k_i for each data provider
1 **foreach** *data provider* $i \leftarrow 1$ **to** m **do** // Perform in parallel
  // Identify Nearby Clusters
2   $C^* \leftarrow$ clusters from $\{C\}$ satisfying Eq. (4);
  // Estimate Selectivity
3   $sel \leftarrow$ calculate Eq. (5) with auxiliary index for each cluster $C \in C^*$;
4   $k_0 \leftarrow k/sel$;
  // Estimate kth Nearest Distance
5   **foreach** *cluster* $C_j \in C^*$ **do**
6     $D_j \leftarrow$ sorted stored distance in C_j;
7     $R_j = \{\mathrm{dist}(q, o_j) + dis \mid dis \in D_j\}$;
8   Min-heap $Q \leftarrow$ pop the shortest distance of each R_j;
9   t_j denotes id of the next distance to be popped from R_j;
10   $cnt \leftarrow 0$;
11   **while** $cnt < k_0$ **do**
12     $\gamma_i^* \leftarrow$ pop shortest distance from cluster j^* out of Q;
13     $t_{j^*} \leftarrow t_{j^*} + 1$, push next distance from R_{j^*} into Q;
14     $cnt \leftarrow cnt + \sqrt{|C_{j^*}|}$
15   submit γ_i^* to SGX;
// Jointly Estimate Contribution
16 **foreach** *data provider* $i \leftarrow 1$ **to** m **do**
17   $k_i \leftarrow$ calculate pruned k according to Eq. (9);
18   send k_i to data provider i for vector similarity search;

from vectors in C^*. In line 5–7, we prepare $\sqrt{|C_j|}$ thresholds with stored distances according to the distance upper bound formulated in Eq. (6). It can be derived that the ith threshold of R_j is the upper bound of $i\sqrt{|C_j|}$th nearest distance to q among vectors in C_j. Thus, we only need to choose a minimum threshold that includes totally larger than $\frac{k}{sel}$ vectors among all clusters from C^* (described by Eq. (7) and Eq. (8)). Lines 8–14 correspond to the detailed process through a min-heap which differs from Alg. 2 solely on the number of rounds. Finally, SGX collects each provider's upper bound as their contribution and utilizes Eq. (9) to calculate pruned results k_i.



(a) Impact of k (b) Impact of m

Figure 8: Impact of query parameters on YT-Rgb dataset

B Additional Experimental Results

Fig. 8 illustrates the results of our experiment on the YT-Rgb dataset concerning impact of query parameters.

Impact of Query Parameter k. As shown in Fig. 8a, when the integer k increases from 32 to 256, both FedVS and HuFu maintain more stable recall than the others, which is up to 22.20% higher than other secure methods. As for the communication overhead and search time, they both increase as k increases, which is similar to the experimental pattern in Fig. 5a. Moreover, regardless of the values of k, FedVS requires lower communication overhead and shorter search time than HuFu, Mr, and DANN*. For example, the search time of FedVS is up to 25.18×, 15.72×, and 14.60× shorter than that of HuFu, Mr and DANN*, respectively.

Impact of #(Data Providers) m. As illustrated in Fig. 8b, when m grows from 2 to 20, the recall of FedVS, HuFu, and DANN* exhibit fluctuations within the ranges of 89.73%–96.16%, 89.75%–96.15%, and 87.50%–92.10%, respectively. Among these solutions, FedVS and HuFu still achieve the highest accuracy, with merely identical answer recall. In terms of communication cost and search time, our FedVS is also the most efficient. For example, FedVS is up to 72.72×, 18.21×, and 13.14× faster than HuFu, Mr, and DANN*. Moreover, we can also observe that the communication overhead of any algorithm gets higher as m increases due to higher network communications for secure computations.

Summary Overall, the experimental results of varying query parameters k and m demonstrate a robust query performance of our solution FedVS. In each query parameter setting, FedVS would lead to better effectiveness and efficiency than existing solutions.