Federated Retrieval over Embedding-Heterogeneous Vector Databases

Yuxiang Wang[1], Yongxin Tong[1], Zimu Zhou[2,3], Ziyuan He[1], Ruixi Hu[1], Ke Xu[1]
[1] State Key Laboratory of Complex and Critical Software Environment,
Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Beihang University, Beijing, China
[2] Department of Data Science, City University of Hong Kong, Hong Kong, China
[3] Shenzhen Research Institute, City University of Hong Kong, Shenzhen, China
{yuxiangwang, yxtong, hzy_he, ruixihu, kexu}@buaa.edu.cn, zimuzhou@cityu.edu.hk

***Abstract*—Vector databases are increasingly used to manage unstructured data by mapping them into high-dimensional embeddings and enabling efficient similarity retrieval. Many real-world applications, such as legal document retrieval and medical question answering, require embedding-based retrieval in federated environments where data are distributed across autonomous silos. We formulate this setting as Federated Approximate Nearest Neighbor Search (FANNS), where a server issues a query along with a query embedding model, aiming to retrieve the top-k nearest objects from datasets across all silos. A key challenge in FANNS is embedding heterogeneity, where silos and the server employ different embedding models, a problem overlooked in prior research. To address this challenge, we exploit the non-IID nature of federated data and propose two novel adaptive algorithms for FANNS queries. The first is a competition-based method that dynamically adjusts retrieval sizes across silos, though it can be sensitive to misleading candidates. The second is a contribution-based method that samples promising silos based on their accumulated contributions, and we provide theoretical guarantees on its latency reduction. Evaluations on four datasets show that our method achieves over 90% retrieval accuracy and 2.3× to 6.2× speedups over existing solutions.**

I. Introduction

Vector databases have become indispensable for managing unstructured data such as images, text, and audio. They store and index high-dimensional *embeddings* of raw objects, enabling effective and efficient semantic search through Approximate Nearest Neighbor Search (ANNS) [1], [2]. Given a query object, the system first maps it into a vector using an *embedding model* and then retrieves its nearest neighbors in the database. This embedding-based retrieval paradigm underpins a wide range of applications including image search [3] and retrieval-augmented generation (RAG) [4].

While traditional deployments assume a *centralized* database, real-world applications are often *federated* [5], [6]. Relevant data are distributed across autonomous silos, *e.g.*, hospitals or law firms, each operating its own task-tailored vector database [7], [8]. To retrieve information scattered across silos, the common workflow sends a user's query to a central server, which dispatches it to participating silos and aggregates the returned results [9]–[12]. However, such *federated retrieval* introduces new challenges in realistic deployments.

Example 1 (Federated Legal Document Retrieval): A law firm preparing a case must consult legal precedents stored across multiple specialized databases (*i.e.*, data silos). Each database may manage its own vector data for semantic retrieval, using an independently chosen embedding model tailored to its content characteristics or resources constraints [8]. Meanwhile, the law firm embeds queries using a custom model trained for its internal case-preparation workflows [13], [14]. The objective is to retrieve documents nearest to the query under the query's embedding space across all silos.

This example highlights a key problem: *embedding heterogeneity*. Retrieval systems increasingly utilize specialized embedding models, such as ResNet [15], ViT [16], BERT [17], Longformer [18], and their fine-tuned variants. As a result, both query users and data silos may independently select distinct models to generate their embeddings. Directly comparing these vectors across different embedding spaces is meaningless. Effective retrieval requires re-embedding database objects under the query model, incurring long latency at query time.

In this paper, we formalize this emerging setting as *Federated Approximate Nearest Neighbor Search (FANNS)*. Given a query object q and its associated query embedding model M_q, a central server aims to retrieve the top-k nearest objects to q from n silos, where each silo S_i stores vectors generated by its own embedding model M_i. The goal is to achieve high retrieval accuracy and low query latency in the presence of embedding heterogeneity. Prior research on ANNS [1], [19]–[21] assumes that query and database vectors reside in same embedding space. Recent studies on federated ANNS [22]–[24] maintain the same assumption, leaving federated retrieval under embedding heterogeneity unaddressed.

Through empirical observations, we identify two challenges posed by embedding heterogeneity. *(i) Re-embedding overhead.* Since database vectors are not directly comparable to the query vector, and the query embedding model is often known only at query time, each candidate must be re-embedded using M_q before distance computation, making re-embedding an efficiency bottleneck. *(ii) Candidate explosion.* Because nearest neighbors vary across the query and silo-specific embedding spaces, uniform silo probing returns numerous irrelevant candidates, increasing re-embedding cost and latency.

Addressing these challenges requires *adaptive* exploration of silos. We observe that, due to the non-IID nature of federated data [25], [26], relevant objects for a query often *concentrate within a few silos*. Leveraging this insight, we design two adaptive query processing algorithms for FANNS.

- *Competition-based FANNS*, which iteratively retrieves candidates from the *current best-performing* silo. However, it may be misled by *bad candidates*, *i.e.*, objects that appear close to the query under local embeddings but are distant under the query embedding.
- *Contribution-based FANNS*, which maintains the *cumulative contributions* of silos for adaptive silo sampling. We further mitigate low-quality candidates via *explicit server feedback* and *adaptive silo exploration*.

Finally, we provide a theoretical analysis on the expected number of re-embedding operations for each strategy and quantify the advantages of contribution-based FANNS.

Our contributions are summarized as follows.

- We introduce FANNS, federated approximate nearest neighbor search under embedding heterogeneity, a new problem motivated by real-world applications such as cross-silo person re-identification and RAG.
- We design two adaptive strategies, competition-based and contribution-based FANNS, and provide theoretical guarantees on their re-embedding costs.
- Extensive experiments show that our solution yields a recall rate of over 90% across four datasets, and improves the query efficiency by 2.3× to 6.2× over the baselines.

II. Problem Statement

We start with preliminaries on vector databases and nearest neighbor search, followed by a formal definition of federated approximate nearest neighbor search (FANNS). We then present a naive baseline solution, empirically analyze its performance, and discuss the challenges and opportunities introduced by embedding heterogeneity.

A. Preliminaries

Vector Database. A vector database manages high-dimensional vector embeddings that capture the semantic features of unstructured data such as images, text, or audio. These embeddings are generated by embedding models, typically deep neural networks like ResNet [15] or BERT [17].

Definition 1 (Vector Database): Given a set of unstructured data objects $D = \{o^1, o^2, \ldots, o^{|D|}\}$, an embedding model M maps each object $o \in D$ to a high-dimensional vector $M(o)$. The original object set D and the resulting embedded vector set $M(D) = \{M(o^1), M(o^2), \ldots, M(o^{|D|})\}$ is stored and managed by the vector database.

Nearest Neighbor Search. Nearest neighbor search is a core operation in vector databases, enabling the retrieval of objects that are most similar to a given query in the embedding space. We first define exact k-nearest neighbor (kNN) search, followed by approximate nearest neighbor search (ANNS), which is widely adopted in modern vector databases [27], [28].

Definition 2 ((Exact) k Nearest Neighbor (kNN)): Given a query object q and a vector database $\{M, M(D)\}$, the (exact) k nearest neighbor retrieves a subset $r^* \subseteq M(D)$ of vectors closest to the embedded query vector $M(q)$. Formally, r^* satisfies $|r^*| = k$ and $\forall u \in r^*, \forall v \in D \setminus r^*$, $dis(M(q), M(u)) \leq dis(M(q), M(v))$, where $dis(\cdot,\cdot)$ denotes the distance function between two vectors, with Cosine distance and Euclidean distance being the common choices.

In large-scale vector databases, ANNS is often used instead of exact kNN to improve efficiency [1]. The accuracy of ANNS is evaluated using the *recall rate*, defined as $\frac{|r \cap r^*|}{k}$, where r is the retrieved result, while r^* is the ground truth for the query. A higher recall indicates higher accuracy.

Vector Indexes. Efficient ANNS relies on indexing methods optimized for high-dimensional data [1]. Commonly used indexes include IVFPQ [19] and HNSW [20]. In practical, these indexes typically achieve high recall rates by searching only a subset of the vectors in the database [27], [28].

The above discussion assumes a *centralized* database. We consider a *federated* setting where unstructured data and their embeddings are distributed across multiple data silos, as defined below.

B. Problem Definition

Federated Approximate Nearest Neighbor Search. We consider a federation F comprising n data silos, denoted as $S_1, S_2, \ldots, S_n$. Each silo autonomously maintains a vector database over the same modality (*e.g.*, images, text, or video), which are generated using silo-specific embedding models.

Definition 3 (Data Silo): Each data silo S_i holds a local dataset $D_i = \{o_i^1, o_i^2, \ldots, o_i^{|D_i|}\}$ and operates a vector database $\{M_i, M_i(D_i)\}$, where M_i is a silo-specific embedding model that maps each object $o \in D_i$ to a high-dimensional vector $M_i(o)$. The embedded vectors are denoted as $M_i(D_i) = \{M_i(o_i^1), M_i(o_i^1), \ldots, M_i(o_i^{|D_i|})\}$. Each silo also builds a local vector index Idx_i over $M_i(D_i)$ to support efficient ANNS within $M_i(D_i)$.

At query time, a centralized server issues a query object q along with a query embedding model M_q. Model M_q is selected by the owner of the application and is optimized for the retrieval or serving tasks to accommodate the personalized requirements of the end user [13], [14], and may differ from *all* silo-specific models M_i.

Definition 4 (Federated Approximate Nearest Neighbor Search (FANNS)): Given a query object q and a query embedding model M_q, FANNS(q, M_q, k) retrieves k objects from $\mathcal{D} = D_1 \cup D_2 \cup \cdots \cup D_n$ that are nearest to q in the embedding space of M_q. Formally, the ground truth of a FANNS query r^* satisfies: $|r^*| = k$, and $\forall u \in r^*, v \in \mathcal{D} \setminus r^*$, $dis(M_q(q), M_q(u)) \leq dis(M_q(q), M_q(v))$, where $M_q(o)$ is the embedding of object o using model M_q.

For simplicity, we define $dis_x(o, o') = dis(M_x(o), M_x(o'))$ as the distance between object o and o' in the embedding space of model M_x. Then the condition of FANNS simplifies to $\forall u \in r^*, v \in \mathcal{D} \setminus r^*$, $dis_q(q, u) \leq dis_q(q, v)$.

Embedding Heterogeneity. As shown in Fig. 1, a key characteristic of FANNS is *embedding heterogeneity*, where each silo uses a unique embedding model M_i, while the query is

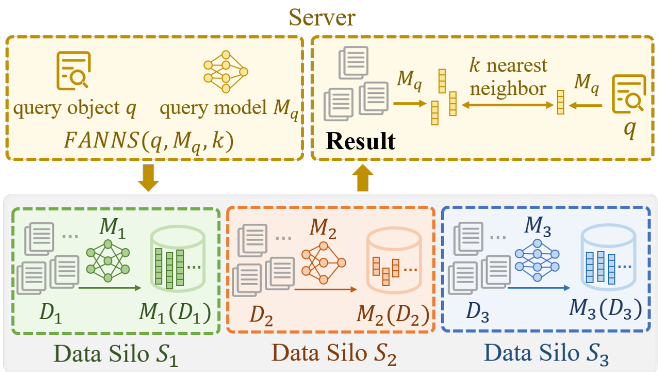


Fig. 1: Federated approximate nearest neighbor search.

evaluated using a different model M_q. Because embedding models are optimized for different downstream tasks, the same object can have different representations across embedding models. Consequently, similarity rankings based on M_i can deviate from those under M_q, degrading the query accuracy if a limited number of objects is retrieved from each silo.

While one might consider sending M_q to each silo for query-time embedding, this is often impractical. Since M_q is unknown in advance, distributing the query model (often hundreds of megabytes or even gigabytes in size) to all silos at runtime would incur substantial communication overhead. Moreover, silos would be required to re-embed their local datasets on the fly, leading to significant computational cost.

TABLE I: Example of FANNS over 3 data silos.

S_1	o_1^1	o_1^2	o_1^3	o_1^4	o_1^5	o_1^6	o_1^7	o_1^8	o_1^9
$dis_1(q,\cdot)$	1.4	1.8	2.1	2.2	3.4	3.8	3.9	4.2	4.3
$dis_q(q,\cdot)$	5.2	2.3	4.7	6.0	5.1	3.4	4.9	6.8	4.1
S_2	o_2^1	o_2^2	o_2^3	o_2^4	o_2^5	o_2^6	o_2^7	o_2^8	o_2^9
$dis_2(q,\cdot)$	1.8	2.4	2.5	2.7	3.3	3.8	3.9	4.1	4.5
$dis_q(q,\cdot)$	2.7	5.8	4.3	4.5	3.3	4.7	4.4	5.9	4.2
S_3	o_3^1	o_3^2	o_3^3	o_3^4	o_3^5	o_3^6	o_3^7	o_3^8	o_3^9
$dis_3(q,\cdot)$	2.8	2.9	3.2	3.7	3.8	4.4	5.7	6.2	7.7
$dis_q(q,\cdot)$	5.0	5.1	3.7	4.4	6.2	4.6	6.7	7.2	6.9

Example 2: Consider a FANNS query over three data silos (S_1, S_2, and S_3) to retrieve the top $k = 3$ objects closest to a query q using model M_q. Table I shows distances from q to objects under both the local embedding models M_i and the query model M_q. Each silo ranks objects using $dis_i(q,\cdot)$, which only involves embedding q by M_i and performing local ANNS on the pre-stored embeddings under M_i. However, computing $dis_q(q,\cdot)$ requires re-embedding all objects with M_q at each silo, which is infeasible during query processing.

Due to embedding heterogeneity, object rankings differ across embedding spaces. For example, S_1 returns o_1^1, o_1^2, o_1^3 as top results to q under M_1, but under M_q, the closest in S_1 are actually o_1^2, o_1^6, o_1^9. Ultimately, the result for query $FANNS(q, M_q, 3)$ are $\{o_1^2, o_2^1, o_2^5\}$.

Objectives. We aim to optimize the performance of FANNS queries using the following metrics:

- **Accuracy:** Maximize the recall rate, defined as $\frac{|r^* \cap r|}{k}$, where r is the retrieved results and r^* is the true top-k nearest neighbors under M_q.
- **Latency:** Minimize query processing time, measured as the average number of queries processed per second.
- **Communication Cost:** Minimize the number of data objects transferred from silos to the server, reducing bandwidth usage and privacy risk.

C. Challenges and Opportunities

Embedding heterogeneity poses new challenges to accurate and efficient FANNS. We first present a baseline, empirically illustrate its limitations, and then highlight opportunities for improvement.

Uniform Selection Strategy for FANNS. A naive approach is to *uniformly* retrieve candidate objects from all silos. Specifically, given a query q, the server retrieves γk objects ($\gamma \geq 1$) evenly from n silos, *i.e.*, each silo returns the top-$\frac{\gamma k}{n}$ objects closest to q based on $dis_i(q,\cdot)$. These candidates are sent to the server, re-embedded under M_q, and re-ranked to select the final top-k results. The expansion factor γ controls the search scope: a larger γ implies a larger search scope and typically results in higher accuracy.

Algorithm 1 shows the detailed procedure. Assume each data silo S_i embeds its local dataset D_i using its own embedding model M_i and builds a vector index (*e.g.*, HNSW [20]) over the resulting embeddings, as described in Definition 3. At query time, the server disseminates the query object q to all silos (line 1). Each silo independently retrieves its local top-$\frac{\gamma k}{n}$ candidates based on $dis_i(q,\cdot)$ using pre-computed embeddings (lines 2-4). The γk objects returned by silos are re-embedded at the server using the query model M_q (lines 5-6). Finally, the server identifies the top-k nearest neighbors based on $dis_q(q,\cdot)$ (lines 7-8).

Algorithm 1: Uniform Selection FANNS

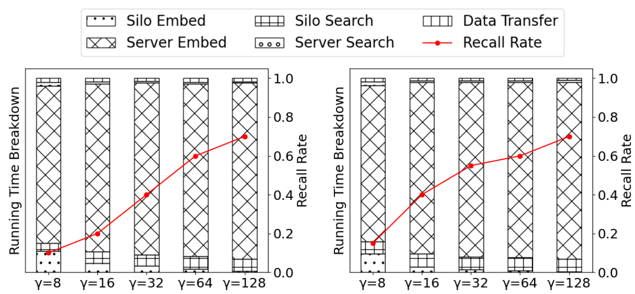
Input: data federation $F = \{S_1, S_2, ..., S_n\}$, query object q, query model M_q, result size k, search scope γ

Output: top-k nearest objects to q under M_q

```
1 Server send object q to all the data silos;
2 foreach i ← 1 to n do
3     Silo S_i embed q with model M_i;        // Silo Embed
4     Silo S_i search the local index Idx_i to select γk/n
        objects nearest to M_i(q);            // Silo Search
5     Silo S_i sends these object to the server;
6 Server embed all the received objects (γk objects in
    total) with model M_q;                    // Server Embed
7 r ← k objects with smallest dis_q(q,·); // Server Search
8 return r
```

Empirical Study. We evaluate the naive solution on the i-Naturalist dataset [29] (see Sec. VI-A for details) with 8 data silos. We report the recall rate under different γ and measure the running time spent across five execution stages: *silo embed, silo search, server embed, server search, and data transfer*. Fig. 2 shows the results.

- **Server embedding dominates latency.** Re-embedding candidate objects at the server using the query model M_q accounts for up to 91.2% of the total query time.
- **High recall demands a large expansion.** Achieving a modest recall rate of 70% necessitates a large expansion factor ($\gamma = 128$), meaning the server must re-embed over $1,000$ candidate objects per query.



(a) $k = 10$ (b) $k = 20$

Fig. 2: Recall rate and breakdown of running time for uniform selection FANNS algorithm. (The experiments are performed on the i-Naturalist dataset [29], with ResNet-152 [15] as the query model and HNSW [20] as the local index)

Implications. The empirical study reveals two issues.

- **Costly re-embedding overhead.** Re-embedding candidates under M_q notably increases latency. This overhead is unavoidable because M_q is unknown before query.
- **Similarity ranking misalignment.** Local ANNS based on M_i does not align with the global similarity ranking under M_q. Consequently, a larger set of candidates from each silo must be returned to ensure acceptable recall rate, resulting in substantial re-embedding and increased query latency.

Therefore, it is crucial to *reduce the number of re-embedded candidates while maintaining high query accuracy*.

Opportunities. Federated data often exhibits non-IID distributions across silos [25]. For instance, in the legal document retrieval application in Example 1, data from different silos often specialize in distinctive legal areas or case types. Such non-IID distributions imply that objects relevant to a query are likely concentrated in a few data silos rather than evenly dispersed. Consequently, uniformly retrieving equal numbers of objects per silo is suboptimal.

To exploit this characteristic, we propose two *adaptive* strategies for FANNS: a competition-based algorithm (Sec. III) and a contribution-based algorithm (Sec. IV). The idea is to **dynamically adjust per-silo retrieval sizes** to reduce server-side re-embedding costs while preserving high recall rate.

III. Competition-based Adaptive FANNS

As discussed in Sec. II-C, a promising strategy for accurate and efficient FANNS is to dynamically adjust candidate selection across silos, rather than using a fixed retrieval size for all silos. We propose a *competition-based adaptive method*, where silos iteratively compete by submitting their most promising candidates. The novelty lies in a multi-round competition mechanism that **prioritizes silos providing the most relevant objects per round**, thus reducing redundant re-embedding operations at the server.

Key Idea. The method is motivated by the intuition that silos providing relevant objects early are more likely to contain additional relevant objects, due to the non-uniform distribution of semantic content across silos. Accordingly, we adopt an adaptive competition process. Each silo proposes its top candidate based on its local embedding model M_i. These candidates compete by being re-embedded at the server under the query embedding model M_q, and the silo whose candidate is closest (the "winner") is asked to provide its next-best candidate. This iterative selection gradually concentrates on silos that consistently provide highly relevant objects, thereby improving efficiency and recall.

Algorithm Details. Algorithm 2 outlines our competition-based adaptive FANNS algorithm. The server begins by sending the query object q to all silos. Each silo embeds q using its local model M_i and returns the nearest object according to its local index Idx_i (lines 4-7). These initial candidates are re-embedded using M_q and stored in a list $cands$. In each round, the server selects the candidate in $cands$ closest to q under M_q, adds it to the result set res, and requests the corresponding silo s to provide its next-best candidate, which replaces the previously used candidate in $cands$ (lines 10-14). The process continues until γk candidates have been collected, where γ is an expansion factor that controls search scope.

Example 3: Fig. 3 illustrates the operation of the competition-based FANNS with three data silos, result size $k = 3$, and expansion factor $\gamma = 4$. Objects with hatch patterns are not among the k nearest neighbors of q under M_q, while solid-filled objects represent the true top-k results. For simplicity, only the top k objects in the result set res (denoted as res_k) are shown. In round 1, each silo proposes its closest local candidate to query q (o_1^1, o_2^1, o_3^1). After re-embedding under M_q, the server identifies o_2^1 as the closest to q, adds it to res, and requests silo S_2 to return its next candidate (o_2^2). In round 2, the process repeats with o_2^2 replacing o_2^1 in $cands$. Next, o_3^1 is selected, and S_3 provides o_3^2. Eventually, this iterative competition results in the final set $\{o_1^2, o_2^1, o_2^5\}$.

IV. Contribution-based Adaptive FANNS

Although the competition-based adaptive FANNS (Sec. III) improves efficiency by dynamically selecting candidates, it relies solely on each silo's *current top candidate*, potentially leading to suboptimal retrieval. In particular, this method can be misled by *bad candidates*, *i.e.*, objects that appear close to the query in local embeddings (M_i) but are distant under the

Algorithm 2: Competition-based Adaptive FANNS

Input: data federation $F = \{S_1, S_2, ..., S_n\}$, query object q, query model M_q, result size k, expansion factor γ

Output: top-k nearest objects to q under M_q

```
1  res ← min-heap ordered by dis_q(q, ·);
2  cands ← n-length candidate list from n silos;
3  Server sends object q to all silos;
4  foreach i ← 1 to n do
5      Silo S_i embeds q with model M_i;
6      Silo S_i searches Idx_i for the nearest object to M_i(q);
7      Silo S_i sends this object to server;
8  Server keeps received objects in cands;
9  Server embeds objects in cands using model M_q;
10 while |res| ≤ γk do
11     Select object o from cands with smallest dis_q(q, ·) and record its silo s;
12     Add object o to res;
13     Silo s searches Idx_i for the next nearest object o′ to M_s(q);
14     Silo s sends object o′ to server;
15     Server embeds o′ with M_q and replaces o with o′ in cands;
16 return top-k objects in res;
```

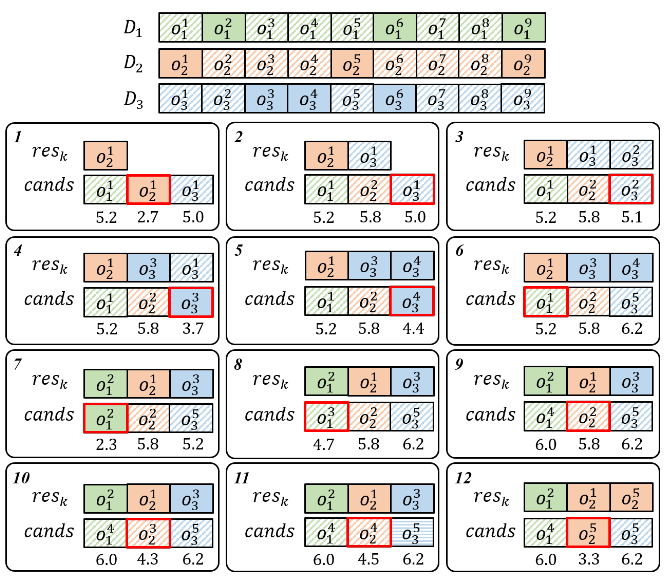


Fig. 3: Example of competition-based adaptive algorithm for FANNS. The number in the upper-left corner represents the round number. Boxes with red borders indicate data objects selected in each round. The digits under boxes indicate the distances from the object to the query q under model M_q's embedding space.

query embedding (M_q), resulting in unnecessary re-embedding (see Lemma 1 in Sec. V).

To overcome this limitation, we propose a novel *contribution-based* FANNS algorithm that adaptively selects silos based on their **cumulative contributions over previous rounds**, rather than solely on their currently proposed objects. We first present the basic version of this method (Sec. IV-A) and then propose multiple optimizations to enhance its efficiency and effectiveness (Sec. IV-B).

A. Basic Contribution-based Algorithm

Key Idea. The contribution-based algorithm is motivated by the hypothesis that silos providing high-quality objects in earlier rounds are likely to have higher data density around the query object. Therefore, we favor silos whose previous objects have frequently appeared among the top-ranked candidates. Concretely, we dynamically maintain and update each silo's sampling probability based on the cumulative contributions to the current result set. Silos with higher contributions are sampled more often, while others are occasionally explored.

Algorithm Details. As in Algorithm 2, the server initially requests each silo to return its nearest object to the query q under the silo's local embedding model M_i (lines 1-8). These candidates are re-embedded using the query model M_q as the initial result set res.

In each subsequent round, the server samples a silo based on its contribution to the current top-k results, denoted as res_k. Specifically, if silo S_i has contributed t_i objects to res_k, its sampling probability is set proportional to $t_i + \theta$, where $\theta > 0$ is a smoothing factor that ensures all silos maintain a non-zero selection probability (lines 10-14). The selected silo s retrieves its next nearest candidate based on $dis_i(q, \cdot)$, which is re-embedded under M_q and added to res (lines 15-16). This process repeats until γk objects have been retrieved, after which the final top-k objects closest to the query under M_q are returned.

Example 4: Fig. 4 illustrates the basic contribution-based algorithm across three silos with $k = 3$, $\gamma = 3$, and smoothing factor $\theta = 1$. Initially, each silo contributes one object, resulting in equal contributions to res_k, and equal sampling probabilities (33% each).

Suppose the server selects silo S_3 and receives its next candidate o_3^2, which is added to res. After this update, silo S_3 contributes two objects to res_k, silo S_2 contributes one, and silo S_1 contributes none. Accordingly, the sampling probabilities change to 17% (for S_1), 33% (for S_2), and 50% (for S_3). This adaptive selection continues until the algorithm terminates at round 9, and the result is $\{o_1^2, o_2^1, o_2^5\}$.

B. Optimized Contribution-based Algorithm

To further enhance the efficiency and effectiveness of the basic contribution-based FANNS algorithm (Sec. IV-A), we propose three optimization strategies.

Algorithm 3: Contribution-based Adaptive FANNS

Input: data federation $F = \{S_1, S_2, ..., S_n\}$, query object q, query model M_q, result size k, expansion factor γ

Output: top-k nearest objects to q under M_q

1 $res \leftarrow$ min-heap ordered by $dis_q(q, \cdot)$;
2 Server sends query q to all silos;
3 **foreach** $i \leftarrow 1$ *to* n **do**
4   Silo S_i embeds q with model M_i;
5   Silo S_i searches Idx_i for the nearest object to $M_i(q)$;
6   Silo S_i sends this object to server;
7 Server stores all received objects in res;
8 Server embeds all objects in res using model M_q;
9 **while** $|res| \leq \gamma k$ **do**
10   **foreach** $i \leftarrow 1$ *to* n **do**
11         $res_k \leftarrow$ top-k objects in res;
12         $t_i \leftarrow$ number of objects in res_k from S_i;
13         $p_i \leftarrow t_i + \theta$;
14   Server samples silo s among the n silos with probability proportional to p_i;
15   Silo s searches its local index for the next nearest object o to $M_s(q)$;
16   Silo s sends object o to server;
17   Server embeds o using M_q and adds o to res;
18 **return** *top-k objects in res*;

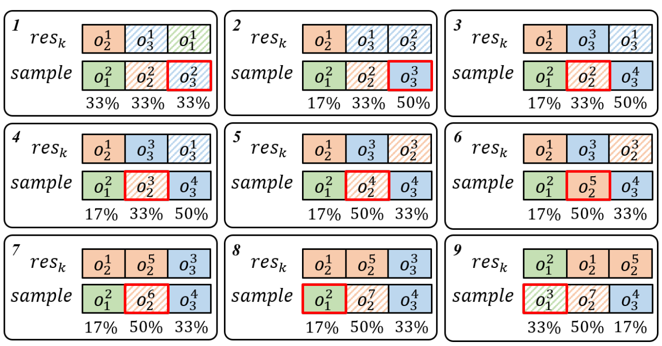


Fig. 4: Example of contribution-based adaptive algorithm for FANNS. The percentages under boxes indicate the probabilities that data objects are sampled.

1) Reducing Communication Round via Batch Sampling: In the basic contribution-based method (Algorithm 3), one object is sampled per silo in each round, leading to frequent server-silo communication. To reduce this overhead, we adopt a *batch sampling* strategy (Algorithm 4) that retrieves multiple candidate objects from the silos in each round.

Algorithm 4: Batch Sampling

Input: data federation $F = \{S_1, S_2, ..., S_n\}$, query object q, query model M_q, result size k, expansion factor γ, batch size b

Output: top-k nearest objects to q under M_q

// Replace lines 14-17 in Algorithm 3 with lines below
1 Initialize $x_1, x_2, \ldots, x_n \leftarrow 0$;
2 **foreach** $i \leftarrow 1$ *to* b **do**
3   Server samples silo s among the n silos with probability proportional to p_i;
4     $x_s \leftarrow x_s + 1$;
5 **foreach** $i \leftarrow 1$ *to* n **do**
6   **if** $x_i \neq 0$ **then**
7   Silo S_i selects searches Idx_i for the next x_i nearest objects to $M_i(q)$;
8   Silo S_i sends these objects to server;
9 Server embeds all received objects using model M_q and adds them to res;

Specifically, in each round, the server determines the number of objects to sample from each silo (lines 1-4). Each silo S_i is sampled multiple times according to a probability proportional to its cumulative contribution p_i, and its frequency of selection in the current round is denoted as x_i. The server sends a request for x_i candidates to each silo. Each silo then selects and returns x_i objects (lines 5-6). All returned candidates are re-embedded at the server in parallel using the query model M_q, thereby reducing the total number of communication rounds and improving overall query efficiency.

2) Adaptive Exploration via Dynamic θ: In the basic algorithm, each silo's sampling probability is proportional to $t_i + \theta$, where t_i is the silo's current contribution, and θ is a *fixed* smoothing factor ensuring non-zero exploration probabilities for all silos. However, using a fixed θ ignores the *evolving reliability of contribution estimates* over time.

To adaptively balance exploration and exploitation, we propose dynamically adjusting the smoothing factor θ. In the early stages, only a few objects have been retrieved, and the observed contributions t_i may not accurately reflect the true relevance of each silo. Thus, a larger θ encourages exploration by giving all silos a more uniform chance of being sampled. As more objects are retrieved, the contribution estimates become more reliable, and a smaller θ helps concentrate the sampling on the most promising silos.

We implement this optimization using a cooling schedule in simulated annealing [30]. Specifically, the smoothing factor at round r is defined as $\theta(r) = \theta_0 \tau^r$, where θ_0 is the initial smoothing factor, and $\tau \leq 1$ is a decay coefficient controlling the exploration-to-exploitation transition rate.

3) Improving Candidate Quality with Server Feedback: In the basic algorithm, candidate selection relies solely on distances in each silo's local embedding space M_i, which may differ from the global ranking under M_q. This misalignment leads to the selection of lower-quality candidates.

To mitigate this, we introduce a *feedback-guided candidate selection* mechanism. The server provides feedback to each silo regarding whether its previously submitted candidate was included in the current top-k result set (res_k). Silos leverage

Algorithm 5: Feedback-Guided Candidate Selection

Input: silo S_i, query object q, required number x_i
Output: x_i candidate objects

$c \leftarrow$ object from silo S_i nearest to q under M_q;
if $c \in res_k$ **then**
  $cands \leftarrow$ silo S_i searches the local index Idx_i for $2x_i$ objects closest to $M_i(q)$;
  Sort $cands$ in ascending order according to $f(\cdot) = dis_i(q, \cdot) + \lambda dis_i(c, \cdot)$;
  return *top* x_i *objects in* $cands$;
else
  $cands \leftarrow$ silo S_i searches the local index Idx_i for x_i objects closest to $M_i(q)$;
  return $cands$;

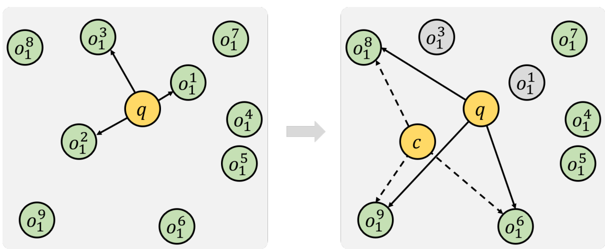


Fig. 5: Example of feedback-guided candidate selection in S_1.

this feedback to refine their candidate selections, biasing future selections toward objects similar to successful past candidates.

Algorithm 5 outlines this process. Let c be the feedback object from silo S_i, *i.e.*, the previously submitted object from S_i with the smallest $dis_q(q, \cdot)$. If c is included in res_k, then S_i evaluates its next $2x_i$ local candidates using a composite scoring function $f(\cdot) = dis_i(q, \cdot) + \lambda \cdot dis_i(c, \cdot)$, where λ is a tunable parameter that balances the impact of the original query and the feedback object. The top-x_i objects with the lowest scores are then returned to the server. Since all embeddings are pre-computed locally under M_i, this procedure incurs negligible additional overhead. The feedback-guided selection strategy is integrated in the candidate selection for the contribution-based adaptive FANNS algorithm (line 15 in Algorithm 3 and line 7 in Algorithm 4).

Example 5: Fig. 5 illustrates the feedback-guided candidate selection in silo S_1. Objects o_1^1 through o_1^9 are ordered by increasing $dis_1(q, \cdot)$ (smaller indices indicate closer distances under M_1). Initially, S_1 selects o_1^1, o_1^2, and o_1^3 purely based on their local distances to q. The server then provides feedback, indicating that o_1^2 is included in the current top-k results, while o_1^1 and o_1^3 are not. In the next round, guided by this feedback, S_1 evaluates objects using the score $dis_1(q, o) + \lambda \cdot dis_1(o_1^2, o)$ and selects those most similar to the successful object o_1^2. As a result, o_1^6, o_1^8, and o_1^9 are chosen.

V. Theoretical Analysis

In this section, we present a statistical analysis of the competition-based adaptive algorithm and the contribution-based adaptive algorithm. As shown in Sec. II-C, the server-side re-embedding of received objects dominates the overall runtime of FANNS queries. Accordingly, our analysis focuses on quantifying the *number of re-embedding operations* required by each method.

A. Assumptions and Notations

Perspectives from Multidimensional Scaling. Representation learning can be viewed as *multidimensional scaling* that transforms raw objects into vector representations while preserving their *relative closeness* [31], [32]. High-quality embedding models, such as ResNet and BERT, are likely to maintain consistent neighborhood structures across embedding spaces. Thus, for high-quality embedding models, objects close to a query in one embedding space are likely to remain close in another. This observation motivates our solutions. Although each data silo uses its own embedding model M_i, and the query embedding model M_q may differ, retrieving a small number of candidate objects, *i.e.*, γk candidates, is often sufficient to achieve high recall for the top-k neighbors under M_q. Empirical results (Sec. VI-B) support that a full scan is rarely needed in various settings.

Measure for Embedding Heterogeneity. While our solutions do not rely on assumptions about the relationship between $\{M_i\}$ and M_q, analyzing their re-embedding cost benefits from a conceptual measure of how similar a local model M_i is to M_q. To this end, we define the notion of *deviation*.

For a given query q, if k nearest neighbors of q under M_q are contained within the top k' nearest neighbors under M_i, we define the deviation of M_i from M_q as $\delta_i = \frac{k'}{k}$. By definition, $\delta_i \geq 1$ for all i. We empirically observe that δ_i typically falls between 10 and 100 across datasets and models. With deviation δ_i, the top $\delta_i k$ objects under M_i always include the true top-k neighbors under M_q. To facilitate analysis, we assume that the exact $\delta_i k$ nearest neighbors are returned by the local vector search. This assumption is reasonable, as modern ANNS algorithms in real-world deployments can achieve recall rates near 100% with proper parameter settings [1]. Furthermore, we assume that the k nearest neighbors under M_q are *uniformly* distributed among the top $\delta_i k$ candidates under M_i. This uniformity assumption enables probabilistic bounds on the number of re-embedding operations.

In the following, we derive the theoretical guarantees on re-embedding efficiency for our proposed algorithms.

B. Analysis and Results

Efficiency of Competition-based Algorithm. Lemma 1 estimates the number of re-embedding operations required by the competition-based adaptive FANNS algorithm.

Lemma 1: Suppose silo S_i contributes k_i objects ($k_i > 0$) to the final FANNS result, and the deviation of its local model is δ_i. To retrieve these k_i objects from silo S_i, the competition-based FANNS algorithm requires $\Omega(\delta_i k_i n)$ re-embedding operations in expectation for objects from n silos.

Proof: By the definition of deviation, retrieving k_i true nearest neighbors from silo S_i requires examining $\delta_i k_i$ candidates under M_i. Among them, k_i are *good* candidates (*i.e.*, true top-k neighbors under M_q), while the remaining $\delta_i k_i - k_i$ are *bad* candidates. Let P_{good} (resp. P_{bad}) denote the probability that a good (resp. bad) candidate from S_i wins the competition (*i.e.*, is selected in line 11 of Algorithm 2). We analyze these probabilities as follows:

- If the current candidate is good, then $P_{\text{good}} \leq 1$, since no probability can exceed 1.
- If the current candidate is bad, then each silo has winning probability of at most $\frac{1}{n}$, *i.e.*, $P_{\text{bad}} \leq \frac{1}{n}$. Specifically, when all silos contribute bad candidates, the winning probability for each silo is $\frac{1}{n}$, and when good candidates are present, this probability becomes smaller.

These upper estimates of selection probabilities lead to a lower bound on re-embedding operations.

The number of retrieved objects (and also the number of re-embedding operations) required for a candidate to be selected follows a geometric distribution with success probability p, which has an expected value of $1/p$ [33]. Hence, the expected number of re-embedding operations needed to retrieve k_i good candidates from S_i is:

$$N_{embed} = k_i \cdot \frac{1}{P_{\text{good}}} + (\delta_i k_i - k_i) \cdot \frac{1}{P_{\text{bad}}} \tag{1}$$

$$\geq k_i + (\delta_i k_i - k_i) \cdot n \tag{2}$$

$$= \Omega(\delta_i k_i n) \tag{3}$$

which completes the proof. ■

Efficiency of Contribution-based Algorithm. We now analyze the number of re-embedding operations required by the basic contribution-based FANNS algorithm under the same assumptions as Lemma 1. The following theorem considers a specified case of Algorithm 3, where the smoothing factor is selected as $\theta = \frac{k}{n}$ according to given silo number n and result size k.

Theorem 1: Suppose silo S_i contributes k_i objects ($k_i > 0$) to the FANNS result, and the deviation of its local model is δ_i. To retrieve these k_i objects from S_i, the basic contribution-based FANNS algorithm requires $O\left(\delta_i k \ln\left(\frac{nk_i}{k} + 1\right)\right)$ re-embedding operations in expectation for objects from n silos, if $\theta = \frac{k}{n}$ in Algorithm 3.

Proof: We first compute the probability of sampling silo S_i in a given round (line 14 of Algorithm 3). Since $\theta = \frac{k}{n}$ is for all silos, the total sampling weight is:

$$\sum_{i=1}^{n}\left(t_i + \frac{k}{n}\right) = \left(\sum_{i=1}^{n} t_i\right) + n \cdot \frac{k}{n} = k + k = 2k. \tag{4}$$

Thus, if silo S_i currently contributes t_i objects to res_k, the probability of selecting S_i is:

$$P_{t_i} = \frac{t_i + \theta}{2k} = \frac{nt_i + k}{2kn}. \tag{5}$$

Since the number of rounds required to sample a given silo follows a geometric distribution, the expected number of rounds retrieve one object from S_i is $\frac{1}{P_{t_i}}$. Given the deviation δ_i, each good object from S_i appears among δ_i candidates, so the total expected number of embeddings to retrieve one such object is $\delta_i \cdot \frac{1}{P_{t_i}}$.

Summing over all k_i good objects from S_i, we obtain the total expected number of retrieved objects (and also the number of re-embeddings):

$$N_{embed} = \delta_i \sum_{t_i=0}^{k_i} \frac{1}{P_{t_i}} = \delta_i \sum_{t_i=0}^{k_i} \frac{2kn}{nt_i + k} = 2\delta_i k \sum_{t_i=0}^{k_i} \frac{n}{nt_i + k}. \tag{6}$$

We now analyze this summation asymptotically:

$$2\delta_i k \sum_{t_i=0}^{k_i} \frac{n}{nt_i + k} = O\left(\delta_i k \sum_{t_i=0}^{k_i} \frac{1}{t_i + \frac{k}{n}}\right) \tag{7}$$

$$= O\left(\delta_i k \sum_{t_i=\lfloor\frac{k}{n}\rfloor}^{k_i+\lfloor\frac{k}{n}\rfloor} \frac{1}{t_i}\right) \tag{8}$$

$$= O\left(\delta_i k \left[\sum_{t_i=1}^{k_i+\lfloor\frac{k}{n}\rfloor} \frac{1}{t_i} - \sum_{t_i=1}^{\lfloor\frac{k}{n}\rfloor} \frac{1}{t_i}\right]\right) \tag{9}$$

$$= O\left(\delta_i k \left[\ln\left(k_i + \frac{k}{n}\right) - \ln\left(\frac{k}{n}\right)\right]\right) \tag{10}$$

$$= O\left(\delta_i k \ln\left(\frac{nk_i}{k} + 1\right)\right), \tag{11}$$

which completes the proof. ■

Comparisons. We now compare the re-embedding costs of the competition-based and contribution-based FANNS algorithms based on the theoretical results in Lemma 1 and Theorem 1.

For convenience, we define $\alpha_i = \frac{k}{k_i}$ as the inverse proportion of result objects contributed by silo S_i, where k_i is the number of final result objects from S_i. Substituting α_i into the re-embedding number from Theorem 1, we have: $O\left(\alpha_i \delta_i k_i \ln\left(\frac{n}{\alpha_i} + 1\right)\right)$, which can be directly compared with the re-embedding number from Lemma 1, given by $\Omega(\delta_i k_i n)$.

To compare the two, consider the inequality $x \geq \ln(x+1)$ for all $x \geq 0$. Substituting $x = \frac{n}{\alpha_i}$ yields $n \geq \alpha_i \ln\left(\frac{n}{\alpha_i} + 1\right)$. Multiplying both sides by $\delta_i k_i$, we get:

$$\delta_i k_i n \geq \alpha_i \delta_i k_i \ln\left(\frac{n}{\alpha_i} + 1\right), \tag{12}$$

which shows that the contribution-based algorithm always requires *fewer* re-embedding operations than the competition-based algorithm to retrieve k_i objects from S_i, regardless of the value of α_i.

Theorem 1 also suggests that the efficiency of the contribution-based algorithm improves when the result distribution is *skewed*, which is common in federated environments [25]. When the result objects mainly come from one silo, α_i approaches 1. In the extreme case where all k result objects originate from a single silo (*i.e.*, $\alpha_i = 1$), the re-embedding cost becomes $O(\delta_i k_i \ln n)$ for the contribution-based algorithm, which is a remarkable reduction over the $\Omega(\delta_i k_i n)$ cost by the competition-based algorithm.

Efficiency of Optimized Contribution-based Algorithm. We analyze the roles of the three optimizations in the contribution-based algorithm.

- The *batch sampling* strategy mainly aims to improve performance by reducing communication rounds while keeping the re-embedding number small.
- The *adaptive exploration* strategy decreases the number of re-embeddings for given δ_i and k_i by accounting for the changing reliability of contributions during execution.
- The *server feedback* optimization aligns distances computed under the local model with those under the query model, thereby reducing δ_i and improving efficiency.

VI. Evaluation

A. Experiment Setup

Datasets. We use the following four datasets for evaluation.

- **i-Naturalist [29]**: A image dataset covering a diverse range of plant and animal species, compiled from biodiversity observations contributed through mobile apps.
- **MS MARCO [34]**: A dataset for question answering, containing real-world questions and answers sourced from Bing search queries.
- **Sentiment [35]**: A real-world federated dataset of user-generated text collected via the public Twitter API.
- **Reddit [35]**: A real-world federated dataset including user comments posted on the social network.

The object cardinalities, raw data sizes and distance functions used in our experiments are summarized in Table II. Among these datasets, Sentiment and Reddit is already partitioned based on the text senders [35]. For the other two datasets, we adopt the partitioning approach from [25], where datasets are partitioned according to Dirichlet distribution with parameter β, with smaller β values indicating higher skewness.

TABLE II: Statistics of datasets.

Dataset	Type	Cardinality	Size	Distance
i-Naturalist	Image	100,000	25.46GB	Euclidean
MS MARCO	Text	837,727	6.59GB	Euclidean
Sentiment	Text	1,599,490	12.48GB	Euclidean
Reddit	Text	3,848,330	65.10GB	Cosine

TABLE III: Parameter settings.

Parameter	Setting
Data Silo Number n	4, **8**, 12, 16
Result Number k	5, **10**, 20, 40
Skewness β	0.1, 0.3, **0.5**, 0.7, 0.9
Vector Index	IVFFlat, IVFPQ, **HNSW**

Parameter Setting. Table III presents the parameter settings used in the experiments, with default values of parameters highlighted in bold. IVFFlat, IVFPQ, and HNSW are three commonly used indexes in vector databases [28]. For IVFFlat and IVFPQ [19], the parameters are set as $n_{\text{list}} = 4\sqrt{|D_i|}$ and $n_{\text{probe}} = 256$. For HNSW [20], the parameters are set as $M = 32$ and $ef_{\text{search}} = 64$.

Embedding Models. We use common embedding models for image and text data in our experiments. For image data, we use ResNet-101, ResNet-152, SwinV2-B, ViT-B, and ViT-H as embedding models [15], [16], [36]. For text data, we use SBert-DistilBERT, SBert-MiniLM, SBert-MPNet, Longformer-Base, Longformer-Large, DeBERTa-Base and DeBERTa-Large as the embedding models [18], [37], [38].

Metrics. We use the following metrics to evaluate the performance of methods for FANNS:

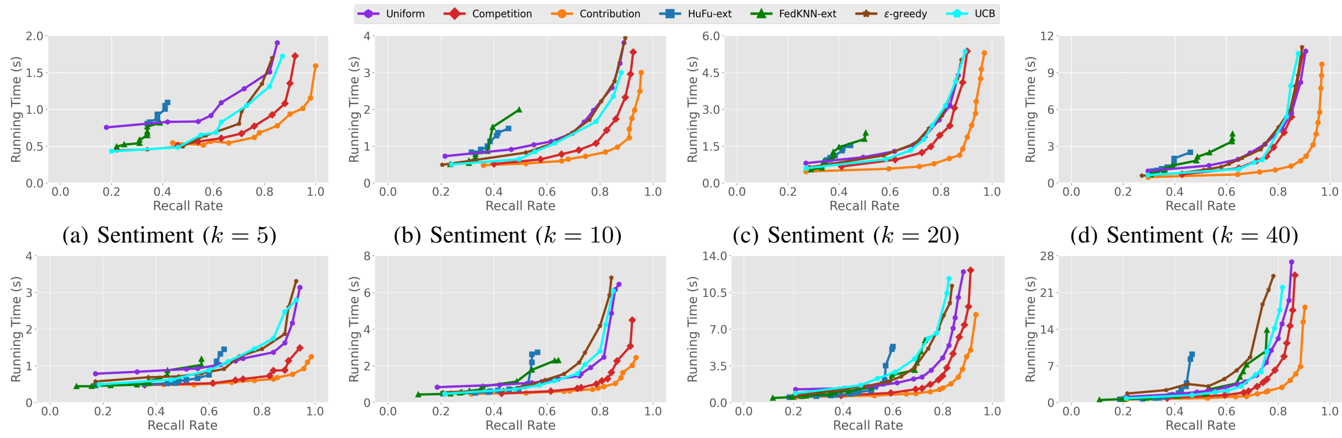
- **Recall rate**: It is defined as $\frac{|r^* \cap r|}{k}$, where r is the returned result of FANNS and r^* is the ground truth.
- **Running time**: It measures the time from query submission to obtaining the final results.
- **Communication cost**: It measures the total amount of data transmitted between data silos and the server during query processing.

Baselines. We extend the existing work to FANNS problem.

- **HuFu-ext [11]**: HuFu addresses federated kNN queries over spatial data silos. To adapt HuFu for FANNS queries, we extend it by first sampling 1,000 points from each dataset and applying the least squares method to derive a linear function that maps distances in local embedding space to the query model's embedding space.
- **FedKNN-ext [22]**: FedKNN is the state-of-the-art method for kNN search in data federation. Similar to HuFu-ext, FedKNN-ext employs the least squares method on the sampled dataset to learn a linear function. However, instead of directly estimating the distance under the query model, it estimates a lower bound for $dis_q(\cdot, \cdot)$.
- **ε-Greedy [39]**: It is designed to balance exploration and exploitation in multi-arm bandit problem. In our setting, each silo is treated as an arm, and the reward is defined as whether an object from the silo appears in the top-k results. For each experimental configuration, we tested different values of ε and report the best-performing result.
- **UCB (Upper Confidence Bound) [39]**: It is also designed for multi-arm bandit problem, and we adapt it to our problem in the same way as ε-greedy.

For all the baselines above, we use HNSW index as the local vector index, following the same setting as in our methods. Similar to our method, these baselines also retrieve γk objects in total for verification, where γ is the expansion factor.

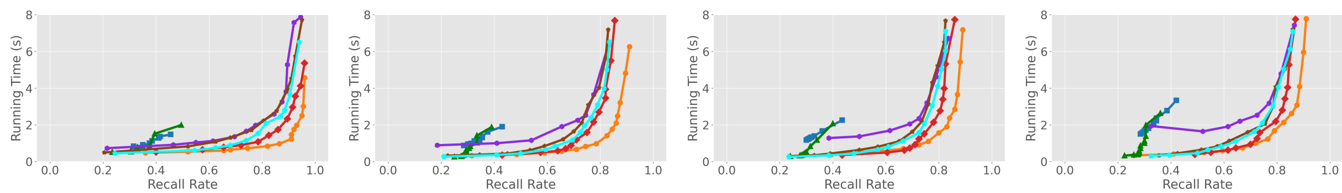
Implementation. All algorithms were implemented in Python 3.12. We used FAISS [40] for vector indexing and searching, PyTorch [41] for embedding data objects, and gRPC for communication. Our codebase includes pluggable interfaces to support flexible deployment of different embedding models, dataset formats, and vector indexes. For the contribution-based adaptive algorithm, we set the batch size to $b = 8$ and the feedback weighting factor to $\lambda = 0.05$. The smoothing factor was initialized as $\theta_0 = \frac{2k}{n}$ and decayed with $\tau = 0.85$.



(a) Sentiment ($k = 5$) (b) Sentiment ($k = 10$) (c) Sentiment ($k = 20$) (d) Sentiment ($k = 40$)

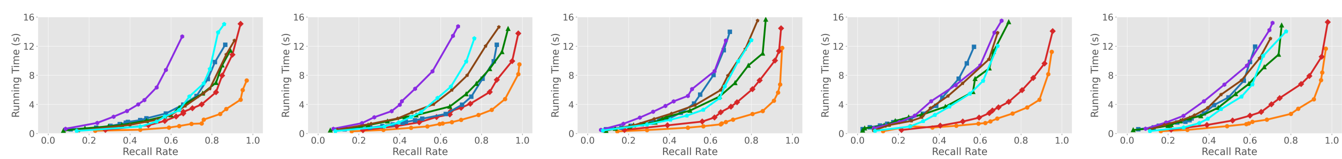
(e) Reddit ($k = 5$) (f) Reddit ($k = 10$) (g) Reddit ($k = 20$) (h) Reddit ($k = 40$)

Fig. 6: The searching performance of FANNS under different datasets and result sizes.



(a) Sentiment ($n = 4$) (b) Sentiment ($n = 8$) (c) Sentiment ($n = 12$) (d) Sentiment ($n = 16$)

Fig. 7: The searching performance of FANNS under different data silo numbers n.



(a) i-Naturalist ($\beta = 0.1$) (b) i-Naturalist ($\beta = 0.3$) (c) i-Naturalist ($\beta = 0.5$) (d) i-Naturalist ($\beta = 0.7$) (e) i-Naturalist ($\beta = 0.9$)

Fig. 8: The searching performance of FANNS in i-Naturalist dataset under different degrees of partition skewness.

Environment. Experiments are conducted on three servers connected by network with bandwidth of 2 Gbps. One server acts as the central coordinator and is equipped with an NVIDIA Tesla V100 32GB GPU, Intel Xeon Gold 6230R 2.10GHz CPUs, and 256GB of RAM. The other two servers act as data silos, each configured with Intel Xeon Gold 6240 2.60GHz CPUs and 768GB of RAM. To simulate multiple data silos, we adopt container-based virtualization, following the common practice in prior work [42], [43], with each silo running in an isolated container.

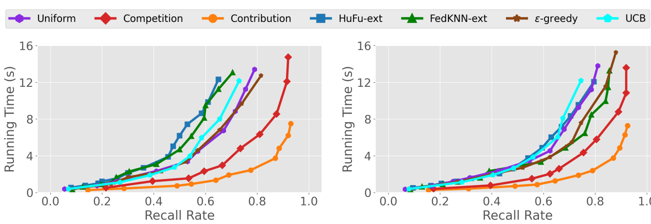
B. Experiment Results

1) End-to-end Performance: We perform experiments to evaluate the performance of three proposed methods–Uniform Selection FANNS, Competition-based Adaptive FANNS, Contribution-based Adaptive FANNS–and four baselines. Following established benchmarks for ANNS [1], [44], we use running time vs. recall rate plots to show the tradeoff between search speed and accuracy. Different data points in these plots are generated by varying the expansion rate γ.

Varying the Result Size k. Fig. 6 presents the recall rate and running time of all methods in Sentiment and Reddit datasets for different result size k. The results show that the contribution-based algorithm can achieve a recall rate of over 90% across all datasets and outperforms other methods in terms of query efficiency at the same recall level. For example, on the i-Naturalist dataset, our adaptive methods improve query efficiency by 2.3× to 6.2× compared to other methods at the same recall rate of 80%. This efficiency gain stems from its adaptive retrieval strategy, which prioritizes promising silos and leverages server feedback to guide local search, enabling it to achieve the same accuracy with a smaller expansion rate. As k increases, the running time of all methods also grows, since more re-embedding operations are required.

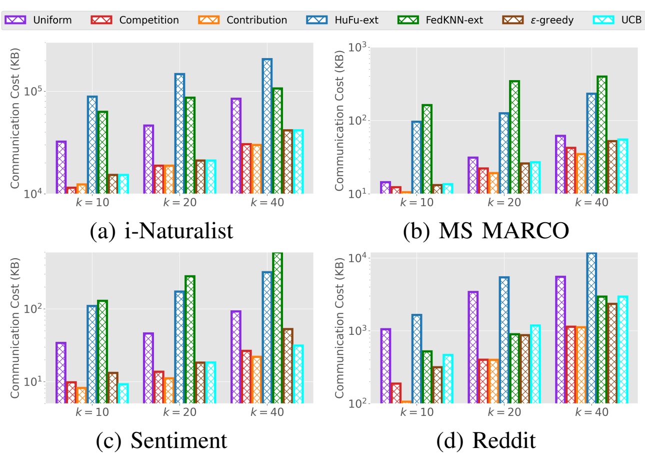
Varying the Data Silo Number n. We evaluate the performance of FANNS under varying numbers of data silos using the Sentiment dataset, with results shown in Fig. 7. The contribution-based adaptive method consistently outperforms the baselines, achieving over 80% recall within 2 seconds across all silo numbers. Besides, the running time of competition-based and contribution-based algorithm remains stable as the number of silos increases, while the other three methods exhibit slight growth in latency.

Varying the Partition Skewness. As shown in Fig. 8, we evaluated all methods under varying partition skewness, where smaller β indicating higher skewness. The results show that both the competition-based and contribution-based methods maintain robust recall rates under different skewnesses. For instance, our adaptive methods consistently achieve recall rates above 90%, while the maximum recall rate of HuFu-ext drops from 86.5% to 62.5% as skewness increases. Moreover, the advantage of the contribution-based method is more pronounced as skewness grows, aligning our analysis in Sec. V.



(a) IVFPQ (b) IVFFlat

Fig. 9: The searching performance of FANNS in i-Naturalist dataset under different local vector indexes.



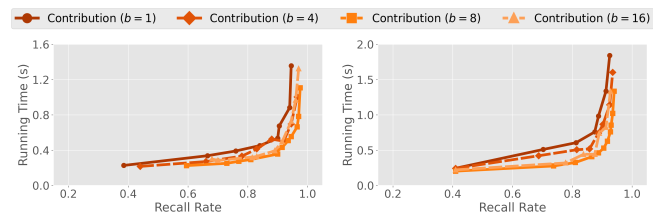
(a) i-Naturalist (b) MS MARCO

(c) Sentiment (d) Reddit

Fig. 10: The communication cost of FANNS under four datasets at a recall rate of 70%.

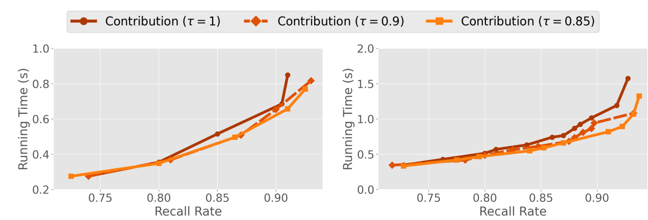
Varying the Local Vector Indexes. We evaluated the algorithms on the i-Naturalist dataset with different local vector indexes, as shown in Fig. 9. In addition to HNSW, we used IVFPQ and IVFFlat as local indexes. Our adaptive algorithm consistently outperforms the baselines across all index settings. Moreover, the running time of each method remains relatively stable across different index types, indicating that server-side re-embedding operations dominate the overall runtime.

Communication Costs. We measured the communication cost of the methods at the same recall rate of 70% in i-Naturalist and Sentiment dataset. As shown in Fig. 10, the proposed adaptive methods achieve lower communication costs at same recall rate compared to other methods. The competition-based algorithm can reduce the communication cost by 31.4% to 96.3% across all the datasets relative to the baselines. In general, the communication cost grows linearly with the result size k. The experiments also indicate that our adaptive methods can achieve high recall by re-embedding only a small fraction of objects from all data silos. For instance, in Sentiment dataset, the contribution-based method reaches a recall rate of 90% with an expansion rate of 5, requiring transfer of only 0.13‰ of the raw dataset.



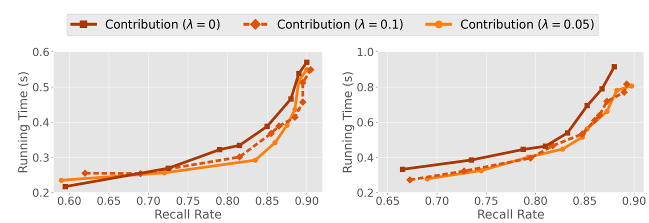
(a) MS MARCO ($k = 10$) (b) MS MARCO ($k = 20$)

Fig. 11: The search performance of contribution-based FANNS under different batch size b.



(a) MS MARCO ($k = 10$) (b) MS MARCO ($k = 20$)

Fig. 12: The search performance of contribution-based FANNS under different selection of smoothing factor.



(a) MS MARCO ($k = 10$) (b) MS MARCO ($k = 20$)

Fig. 13: The search performance of contribution-based FANNS under different feedback weighting factor λ.

2) Ablation Study: We conducted ablation studies on MS MARCO dataset to assess how the three optimization strategies affect the performance of the contribution-based method.

Impact of Batch Sampling. We compare the search performance of contribution-based method under four different batch sizes b on the Sentiment dataset, as shown in Fig. 11. The results show that all batch size settings lead to performance improvements over the basic contribution-based algorithm (*i.e.*, the method with $b = 1$). As the batch size increases, the number of communication rounds decreases, thereby enhancing search efficiency. However, larger batch sizes also reduce the timeliness of contribution updates, resulting in more re-embeddings and slightly impacting overall performance. The experiments indicate that contribution-based FANNS achieves optimal performance at a batch size of 8, yielding an up to $2.8\times$ improvement of query efficiency at the same recall rate.

Impact of Adaptive Smoothing Factor. We compare different values of decay coefficient τ to study the effectiveness of adaptive smoothing factor. As shown in Fig. 12, adjustment of smoothing factor θ can effectively improve the performance of the contribution-based algorithm, particularly in the later stages of query processing. Specifically, with result size $k = 20$, setting $\tau = 0.85$ improves query efficiency by $1.3\times$ at 80% recall and by $1.5\times$ at 90% recall. This gain arises because the adaptive θ enables the algorithm to increasingly focus on promising silos as more feedback accumulates.

Impact of Feedback-Guided Candidate Selection. We evaluated the impact of feedback-guided candidate selection and different values of the feedback weighting factor λ on the MS MARCO dataset, as shown in Fig. 13. The results show that incorporating server feedback with $\lambda = 0.05$ effectively improves candidate quality, thereby enhancing the accuracy of the contribution-based algorithm. For example, using the score function with $\lambda = 0.05$ improved the recall rate by up to 4.5% when $k = 10$, and by up to 8% when $k = 20$.

VII. Related Work

Approximate Nearest Neighbor Search. ANNS is a core operation in vector databases [2], [27], [28], with methods broadly categorized into partition-based [45]–[49], quantization-based [19], [50]–[52], and graph-based techniques [20], [53]–[57]. They assume that both the query and data vectors reside in the same *embedding space* and are drawn from the same *data distribution*.

Recent work starts to explore relaxed settings. OOD-DiskANN [58], RoarGraph [21], and LeanVec [59] address out-of-distribution (OOD) queries, where the query vector follows a different distribution than indexed vectors, which is common in cross-modal retrieval (*e.g.*, text-to-image). However, they still operate within a single embedding space. In contrast, FANNS involves multiple embedding spaces. Each silo uses a different model, and the query's model may differ from all of them. Other studies support customized distance metrics in ANNS, such as OASIS [60] and ONIAK [61], which allow query-specific Mahalanobis metrics by applying linear transformations. While flexible, these methods are insufficient for FANNS, where the distance mapping between embedding models can be non-linear.

Federated Data Management. Federated database systems were originally proposed to support querying over autonomous and heterogeneous data sources [9], [62]. This concept has been extended to various data types, including relational [12], [63]–[65], spatial [10], [11], [66], and graph data [67].

Recent efforts have explored federated vector retrieval [22]–[24], [68], which propose methods for federated approximate nearest neighbor search across vector databases. However, they assume that all data silos use a shared embedding model, enabling direct comparison of vectors across silos. This assumption does not hold in our work, where each silo independently selects its embedding model, resulting in heterogeneous embedding spaces.

Model Heterogeneity in Federated Learning. FL enables collaborative model training across decentralized data silos without sharing raw data [25], [69], [70]. While early FL research assumes homogeneous model architectures across clients, recent studies have explored model heterogeneity, where each silo may adopt a different neural network architecture [71]. However, this line of research differs fundamentally from our work in its objective. Federated learning focuses on model training whereas we aim to retrieve similar objects.

VIII. Conclusion

This paper defines the problem of Federated Approximate Nearest Neighbor Search (FANNS) over embedding-heterogeneous vector databases and proposes both accurate and time-efficient solutions for FANNS queries. Leveraging the non-IID nature of federated data, we design two adaptive algorithms—competition-based and contribution-based—for FANNS query processing. We provide theoretical analysis of the expected number of re-embedding operations for both methods. Extensive evaluations show that our solution yields over 90% recall rate across four datasets, while achieving up to 6.2× query efficiency compared to existing solutions.

AI-Generated Content Acknowledgement

The authors used deepseek to improve the language clarity and readability of this paper. All generated content was carefully reviewed and edited by the authors to ensure accuracy and alignment with the paper's intended meaning.

Acknowledgement

This work was partially supported by National Science Foundation of China (NSFC) (Grant Nos. 62425202, 62336003), National Key Research and Development Program of China under Grant No. 2023YFF0725103, 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). Zimu Zhou's research is supported by National Science Foundation of China (NSFC) (No. 62572412), the RGC of Hong Kong SAR, China (Project No. CityU 11206425). Yongxin Tong is the corresponding author.

REFERENCES

[1] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, "Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1475–1488, 2019.

[2] K. Echihabi, K. Zoumpatianos, and T. Palpanas, "New trends in high-d vector similarity search: al-driven, progressive, and distributed," in *Proceedings of the VLDB Endowment*, 2021, pp. 3198–3201.

[3] O. Moll, M. Favela, S. Madden, V. Gadepally, and M. Cafarella, "Seesaw: interactive ad-hoc search over image databases," *Proceedings of the ACM on Management of Data*, pp. 1–26, 2023.

[4] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, and Q. Li, "A survey on rag meeting llms: Towards retrieval-augmented large language models," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 6491–6501.

[5] Z. Zhang, X. Hu, J. Zhang, Y. Zhang, H. Wang, L. Qu, and Z. Xu, "Fedlegal: The first real-world federated learning benchmark for legal nlp," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2023, pp. 3492–3507.

[6] B. Pfitzner, N. Steckhan, and B. Arnrich, "Federated learning in a medical context: a systematic literature review," *ACM Transactions on Internet Technology (TOIT)*, pp. 1–31, 2021.

[7] P. Xia, K. Zhu, H. Li, H. Zhu, Y. Li, G. Li, L. Zhang, and H. Yao, "Rule: Reliable multimodal rag for factuality in medical vision language models," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 1081–1093.

[8] N. Wiratunga, R. Abeyratne, L. Jayawardena, K. Martin, S. Massie, I. Nkisi-Orji, R. Weerasinghe, A. Liret, and B. Fleisch, "Cbr-rag: case-based reasoning for retrieval augmented generation in llms for legal question answering," in *International Conference on Case-Based Reasoning*, 2024, pp. 445–460.

[9] A. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Computing Surveys*, pp. 183–236, 1990.

[10] Y. Shi, Y. Tong, Y. Zeng, Z. Zhou, B. Ding, and L. Chen, "Efficient approximate range aggregation over large-scale spatial data federation," *IEEE Transactions on Knowledge and Data Engineering*, pp. 418–430, 2021.

[11] Y. Tong, X. Pan, Y. Zeng, Y. Shi, C. Xue, Z. Zhou, X. Zhang, L. Chen, Y. Xu, K. Xu *et al.*, "Hu-fu: Efficient and secure spatial queries over data federation," in *Proceedings of the VLDB Endowment*, 2022, pp. 1159–1172.

[12] J. Bater, S. Goel, G. Elliott, A. Kho, C. Eggen, and J. Rogers, "Smcql: Secure querying for federated databases," in *Proceedings of the VLDB Endowment*, 2016, pp. 673–684.

[13] Databricks, "Improving retrieval and rag with embedding model finetuning," 2025. [Online]. Available: https://www.databricks.com/blog/improving-retrieval-and-rag-embedding-model-finetuning

[14] LlamaIndex, "Fine-tuning embeddings for rag with synthetic data," 2023. [Online]. Available: https://www.llamaindex.ai/blog/fine-tuning-embeddings-for-rag-with-synthetic-data-e534409a3971

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2020, pp. 1–21.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 4171–4186.

[18] Huggingface, "Longformer," 2025. [Online]. Available: https://huggingface.co/docs/transformers/model_doc/longformer

[19] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 117–128, 2010.

[20] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 824–836, 2018.

[21] M. Chen, K. Zhang, Z. He, Y. Jing, and X. S. Wang, "Roargraph: A projected bipartite graph for efficient cross-modal approximate nearest neighbor search," in *Proceedings of the VLDB Endowment*, 2024, pp. 2735–2749.

[22] 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*, 2024, pp. 1–26.

[23] Z. Zhu, Z. Fan, Y. Zeng, Y. Shi, Y. Xu, M. Zhou, and J. Dong, "Fedsq: A secure system for federated vector similarity queries," in *Proceedings of the VLDB Endowment*, 2024, pp. 4441–4444.

[24] D. Zhao, "Frag: Toward federated vector database management for collaborative and secure retrieval-augmented generation," *arXiv preprint arXiv:2410.13272*, 2024.

[25] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *IEEE International Conference on Data Engineering*, 2022, pp. 965–978.

[26] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, pp. 1–210, 2021.

[27] J. J. Pan, J. Wang, and G. Li, "Survey of vector database management systems," *The VLDB Journal*, pp. 1591–1615, 2024.

[28] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu *et al.*, "Milvus: A purpose-built vector data management system," in *Proceedings of the ACM on Management of Data*, 2021, pp. 2614–2627.

[29] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist species classification and detection dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8769–8778.

[30] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, pp. 671–680, 1983.

[31] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, pp. 1–27, 1964.

[32] J. Xie, R. Gao, E. Nijkamp, S.-C. Zhu, and Y. N. Wu, "Representation learning: A statistical perspective," *Annual Review of Statistics and Its Application*, pp. 303–335, 2020.

[33] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[34] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "Ms marco: A human-generated machine reading comprehension dataset," in *Proceedings of the Workshop on Cognitive Computation*, 2016, pp. 1–10.

[35] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[36] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[37] P. He, J. Gao, and W. Chen, "Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing," in *International Conference on Learning Representations*, 2020, pp. 1–16.

[38] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019, pp. 3982–3992.

[39] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[40] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," *arXiv preprint arXiv:2401.08281*, 2024.

[41] PyTorch, "Tensors and dynamic neural networks in python with strong gpu acceleration," 2025. [Online]. Available: https://pytorch.org

[42] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, 2021, pp. 19–35.

[43] F. Lai, Y. Dai, S. Singapuram, J. Liu, X. Zhu, H. Madhyastha, and M. Chowdhury, "Fedscale: Benchmarking model and system performance of federated learning at scale," in *International conference on machine learning*. PMLR, 2022, pp. 11 814–11 827.

[44] M. Aumüller, E. Bernhardsson, and A. Faithfull, “Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms,” *Information Systems*, pp. 1–13, 2020.
[45] A. Gionis, P. Indyk, R. Motwani *et al.*, “Similarity search in high dimensions via hashing,” in *Proceedings of the VLDB Endowment*, 1999, pp. 518–529.
[46] M. Muja and D. G. Lowe, “Scalable nearest neighbor algorithms for high dimensional data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2227–2240, 2014.
[47] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, “Multi-probe lsh: efficient indexing for high-dimensional similarity search,” in *Proceedings of the VLDB Endowment*, 2007, pp. 950–961.
[48] J. Pound, F. Chabert, A. Bhushan, A. Goswami, A. Pacaci, and S. R. Chowdhury, “Micronn: An on-device disk-resident updatable vector database,” *arXiv preprint arXiv:2504.05573*, 2025.
[49] J. Wei, B. Peng, X. Lee, and T. Palpanas, “Det-lsh: A locality-sensitive hashing scheme with dynamic encoding tree for approximate nearest neighbor search,” in *Proceedings of the VLDB Endowment*, 2024, pp. 2241–2254.
[50] Y. Matsui, Y. Uchida, H. Jegou, and S. Satoh, “A survey of product quantization,” *ITE Transactions on Media Technology and Applications*, pp. 2–10, 2018.
[51] J. Gao and C. Long, “Rabitq: Quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search,” in *Proceedings of the ACM on Management of Data*, 2024, pp. 1–27.
[52] J. Gao, Y. Gou, Y. Xu, Y. Yang, C. Long, and R. C.-W. Wong, “Practical and asymptotically optimal quantization of high-dimensional vectors in euclidean space for approximate nearest neighbor search,” in *Proceedings of the ACM on Management of Data*, 2025, pp. 1–27.
[53] M. Wang, X. Xu, Q. Yue, and Y. Wang, “A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search,” in *Proceedings of the VLDB Endowment*, 2021, pp. 1964–1978.
[54] I. Azizi, K. Echihabi, and T. Palpanas, “Graph-based vector search: An experimental evaluation of the state-of-the-art,” in *Proceedings of the ACM on Management of Data*, 2025, pp. 1–31.
[55] C. Fu, C. Xiang, C. Wang, and D. Cai, “Fast approximate nearest neighbor search with the navigating spreading-out graph,” in *Proceedings of the VLDB Endowment*, 2019, pp. 461–474.
[56] Y. Peng, B. Choi, T. N. Chan, J. Yang, and J. Xu, “Efficient approximate nearest neighbor search in multi-dimensional databases,” in *Proceedings of the ACM on Management of Data*, 2023, pp. 1–27.
[57] I. Azizi, K. Echihabi, and T. Palpanas, “Elpis: Graph-based similarity search for scalable data science,” in *Proceedings of the VLDB Endowment*, 2023, pp. 1548–1559.
[58] S. Jaiswal, R. Krishnaswamy, A. Garg, H. V. Simhadri, and S. Agrawal, “Ood-diskann: Efficient and scalable graph anns for out-of-distribution queries,” *arXiv preprint arXiv:2211.12850*, 2022.
[59] M. Tepper, I. S. Bhati, C. Aguerrebere, M. Hildebrand, and T. L. Willke, “Leanvec: Searching vectors faster by making them fit,” *Transactions on Machine Learning Research*, pp. 1–16, 2024.
[60] H. Zhang, L. Cao, Y. Yan, S. Madden, and E. A. Rundensteiner, “Continuously adaptive similarity search,” in *Proceedings of the ACM on Management of Data*, 2020, pp. 2601–2616.
[61] J. Meng, H. Wang, J. Xu, and M. Ogihara, “One index for all kernels (oniak): A zero re-indexing lsh solution to anns-alt (after linear transformation),” in *Proceedings of the VLDB Endowment*, 2022, pp. 3937–3949.
[62] M. T. Özsu and P. Valduriez, *Principles of Distributed Database Systems*. Springer Nature, 2019.
[63] F. Kiehn, M. Schmidt, D. Glake, F. Panse, W. Wingerath, B. Wollmer, M. Poppinga, and N. Ritter, “Polyglot data management: state of the art & open challenges,” in *Proceedings of the VLDB Endowment*, 2022, pp. 3750–3753.
[64] D. Agrawal, L. Ba, L. Berti-Equille, S. Chawla, A. Elmagarmid, H. Hammady, Y. Idris, Z. Kaoudi, Z. Khayyat, S. Kruse *et al.*, “Rheem: Enabling multi-platform task execution,” in *Proceedings of the ACM on Management of Data*, 2016, pp. 2069–2072.
[65] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. Zdonik, “The bigdawg polystore system,” *ACM SIGMOD Record*, pp. 11–16, 2015.
[66] Y. Chen, X. Pang, X. Li, H. Wang, B. Niu, and S. Hu, “U-dpap: Utility-aware efficient range counting on privacy-preserving spatial data federation,” in *Proceedings of the ACM on Management of Data*, 2025, pp. 1–25.
[67] Y. Yuan, D. Ma, Z. Wen, Z. Zhang, and G. Wang, “Subgraph matching over graph federation,” in *Proceedings of the VLDB Endowment*, 2021, pp. 437–450.
[68] P. Zhang, B. Yao, C. Gao, B. Wu, X. He, F. Li, Y. Lu, C. Zhan, and F. Tang, “Learning-based query optimization for multi-probe approximate nearest neighbor search,” *The VLDB Journal*, pp. 623–645, 2023.
[69] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*, 2017, pp. 1273–1282.
[70] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology*, pp. 1–19, 2019.
[71] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao, “Heterogeneous federated learning: State-of-the-art and research challenges,” *ACM Computing Surveys*, pp. 1–44, 2023.