FEDDE: Federated Data Deduplication

Jinming Hu*
jinminghu@sea-land.ai
sea-land.ai
Canada

Armaan Nanji
Zixiu Meng
armaan.nanji@mail.utoronto.ca
zixiu.meng@mail.utoronto.ca
University of Toronto
Canada

Hao Wang
hwang9@stevens.edu
Stevens Institute of Technology
United States

Jingxian Wang
wang@nus.edu.sg
National University of Singapore
Singapore

Wentao Wu
wentao.wu@microsoft.com
Microsoft Research
United States

Qizhen Zhang
qz@cs.toronto.edu
University of Toronto
Canada

Abstract

This paper introduces FEDDE, a general and efficient framework that addresses data redundancy across clients to facilitate effective federated learning (FL). At its core, FEDDE adopts a hierarchical deduplication architecture where clients first perform local, centralized deduplication and then send minimal records that are only meaningful for redundancy detection to the server for global deduplication. To enable flexible trade-offs between FL training efficiency and the accuracy of the training outcomes, FEDDE proposes two-round approximate deduplication protocols. A set of system optimizations is further applied to reduce deduplication overhead. Using FEDDE's general API, we have implemented federated deduplication solutions to common data types. Evaluation results with real-world datasets have confirmed the efficacy and performance of FEDDE: it can significantly improve FL training efficiency without significant model accuracy degradation, it enables flexible trade-offs between the two metrics, and it is efficient and scales to many clients.

1 Introduction

Popularity of FL. The proliferation of mobile services and intelligent edge devices has led to an explosion of user-generated data distributed on the clients connected by the Internet and the Web, creating opportunities for training machine learning models at unprecedented scales. Federated learning (FL) [57] has emerged as a machine learning paradigm that enables collaborative model training on decentralized data without centralizing sensitive user information. The utility of FL has been explored in many domains, including critical Web applications from on-device recommendation systems to personalized content generation [9, 24, 75, 82, 84].

FL training efficiency. Compared to model training in the cloud, where powerful compute servers and fast network fabrics are provided, achieving training efficiency in FL has unique challenges. Specifically, in FL, clients first train the target model locally on their private data and upload model updates, instead of raw training data, to the server, which incorporates, usually via aggregation, the updates in the global model. The updated model is then broadcast to clients for the next round of training. In this process, the constrained resource capacities of the clients and the limited bandwidth and high latency of wide-area networks are all obstacles to achieve high model training efficiency. In response, many solutions have been proposed to improve the efficiency of FL in a variety of aspects, including communication [37, 55, 58, 77], client optimization and sampling [10, 43, 76, 86], and overall architecture [4, 5].

Data duplication on FL clients. Despite the progress made by the aforementioned efforts to improve FL training efficiency, data redundancy, especially across FL clients, has been largely overlooked. Previous work in centralized learning has shown the adverse effects of duplicate data on the performance of model training and the trained models [45, 68, 70]—training time increases proportionally with training dataset, and the final model can be biased towards repeated duplicates. Compared to centralized systems, FL clients generate data independently and are thus prone to data duplication. For example, images or videos of popular objects or events, and texts about trending topics are likely to appear repeatedly on many clients. As FL clients are wimpier than cloud servers, the training overhead incurred by duplicate data is also more pronounced.

Challenges of data deduplication for FL. A variety of domain-specific data deduplication solutions have been proposed in recent years [2, 7, 11, 12, 35, 38, 41, 45, 54, 65, 68, 70], assuming controlled environments and accessibility to the raw data items. In comparison, detecting and removing redundant data between decentralized, federated clients is underexplored and has the following challenges.

- *Resource Constraints.* The computational resources (e.g., CPU and memory) on FL clients (e.g., smart phones and vehicles) are often constrained. The client-side computational workload for deduplication must be minimized.
- *Network Speed.* FL clients are geographically distributed and connected to the server via wide-area networks, which have significantly higher latency and lower bandwidth compared to local networks (e.g., data center networks). Therefore, data transfers over the network must be carefully planned.
- *Data Heterogeneity.* FL clients are often equipped with sensors that can generate data of different types, e.g., texts, images, and videos. Duplicates can appear in any dataset of any type, thus calling for a general solution that can accommodate data heterogeneity. Deduplication approaches for specific data modalities are incomplete solutions [1].
- *Scalability.* FL can scale to a large number of real-world devices [5]. Accordingly, a deduplication solution for FL must also have a scalable design.

*Work done during an internship at the University of Toronto.

- *Data Privacy.* To protect the private information contained in the training data, the raw data items on a client should never be moved outside of the client, which to a great extent hamstrings cross-client duplicate detection.

Our proposal: FEDDE. To address the above challenges, we propose FEDDE, a data deduplication framework purposely designed for FL. It can effectively remove redundant data not only within individual clients but also across clients to improve FL training efficiency. At its core, FEDDE adopts a *hierarchical deduplication* architecture, which first removes duplicates within each client with centralized deduplication and then sends small *records* that are only meaningful for deduplication to the server for global deduplication. This architecture enables several key benefits for federated data deduplication: (1) it has no assumptions on the data type and can thus generalize, (2) it preserves privacy in the raw training dataset, (3) it minimizes communication overhead as the records transferred to the server are significantly smaller than original items, and (4) it scales gracefully with the number of clients as clients communicate only with the server, not broadcasting messages to other clients.

Data redundancy also exists in different but similar items, i.e., near duplicates. Removing near duplicates can speed up the training process but may hazard the accuracy of the trained model, depending on the loss of information in the differences. FEDDE allows flexible trade-offs between FL training efficiency and the accuracy of the training outcomes with *approximate deduplication.* To support approximate deduplication in the hierarchical architecture, clients generate embeddings from their local training datasets as the records, and the server employs a *vector database* to store the embeddings and perform efficient approximate nearest neighbor search (ANNS) to identify near duplicates. Transferring embeddings, however, amplifies network communication overhead. FEDDE introduces *two-round deduplication protocols*, where a round of light-weight exact deduplication is first performed to pre-filter identical items from the embedding generation in the second round. We further apply a set of system optimizations to improve the performance of clients, the server, and the data transfers in between.

On top of the hierarchical, two-round workflow, we propose a general API to instantiate federated deduplication for datasets of any type. To demonstrate its uses and values, we implement deduplication solutions for three common data types with FEDDE: texts, images, and videos. With real datasets and ML models, we extensively evaluate the efficacy and performance of FEDDE. Our experimental results show that it can significantly improve FL training efficiency by removing identical and nearly identical training samples between clients without significant adverse impacts on model accuracy—in some cases, the model performance is improved with FEDDE-deduplicated data. Moreover, FEDDE enables flexibility to make efficient trade-offs between FL training efficiency and model accuracy, minimizes deduplication time (17× faster than the baseline), and scales to many FL clients (saving 99% communication when scaling to hundreds of clients compared to the baseline).

Contributions. This paper makes the following contributions:

- We investigate the general problem of data deduplication for FL and introduce FEDDE, a general federated data deduplication framework that achieves scalability and preserves privacy with hierarchical deduplication (§3.1–§3.2).
- We extend approximate deduplication to the hierarchical architecture to allow flexible trade-offs between FL training efficiency and model accuracy with two-round protocols (§3.3). Atop the deduplication workflow, we propose a general and easy-to-use API (§3.4) and apply a set of system optimizations to improve deduplication efficiency (§3.5).
- We use FEDDE to implement federated data deduplication for three common data types: texts, images, and videos (§4).
- We extensively evaluate FEDDE with realistic datasets, models, and FL tasks, and confirm its effectiveness to improve FL training efficiency and its flexibility, efficiency, and scalability for federated data deduplication (§5).
- We make our source code publicly available athttps://github.com/conanhujinming/fedde. This includes the FEDDE toolkit, data partitioning scripts, model configurations, and training workflows.

2 Background

2.1 Federated Learning

FL is a decentralized machine learning paradigm [57]. The core principle of FL is to train a global model f_θ where $\theta \in \mathbb{R}^d$ represents the model parameters collaboratively across a multitude of distributed clients C (e.g. mobile devices):

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta, D) := \sum_{i \in C} \frac{|D_i|}{|D|} \mathcal{L}_i(\theta, D_i)$$

without requiring clients to share their raw, potentially sensitive data D_i. A typical FL training round t proceeds as follows: (1) a central server distributes the current global model weights θ^t to a subset of clients $C^t \subseteq C$; (2) each client $i \in C^t$ independently trains the model on its local data D_i for E epochs:

$$\theta_i^{t+1} \leftarrow \theta_i^t - \eta \sum_{j=1}^{E} \nabla \mathcal{L}_i(\theta^t, D_i);$$

(3) each client i sends its model updates θ_i^{t+1}, not their data D_i, to the server; and (4) the server aggregates these updates to produce an improved global model:

$$\theta^{t+1} \leftarrow \sum_{i \in C^t} \frac{|D_i|}{\cup_{j \in C^t} |D|} \theta_i^t.$$

This iterative process allows for the creation of ML models from globally distributed data while preserving client-side data privacy. Key challenges in FL include managing statistical heterogeneity (non-IID data distribution) [43, 58, 78], improving communication efficiency [37, 55, 77], and achieving scalability [4, 5].

2.2 Data Deduplication

Exact Deduplication. Exact deduplication aims to identify identical data items. A straightforward approach is to apply a hash function (e.g., SHA-256) to generate a signature for each item. More sophisticated deduplication approaches for domain-specific data exist. For example, Lee et al. [44] compute suffix arrays of each text document to find identical substrings. Although this is more effective than hashing documents, generating suffix arrays and performing comparisons can be computationally expensive. Content-defined chunking (CDC) serves as an efficient alternative. CDC algorithms, such as those based on Rabin fingerprints [66], use a rolling hash to partition a byte stream into non-uniform and content-aware chunks. This property ensures that local insertions or deletions only affect nearby chunk boundaries, which allows identical content blocks to be identified with the same hash, regardless of their position across different documents. CDC can effectively identify exact duplicates within a large corpus.

Approximate Deduplication. Near-duplicate detection, or approximate deduplication, detects semantically similar but not bit-to-bit identical data items. A popular technique for text documents is locality-sensitive hashing (LSH), which hashes items such that similar items are likely mapped to the same hash value. One of the pioneering LSH schemes is MinHash [6], which estimates the Jaccard similarity between two documents by comparing compact signatures derived from the sets of shingles (n-grams) they contain.

For perceptual data, such as images and videos, perceptual hashing [19] is often adopted, such as the one based on the discrete cosine transform (DCT) [52], which scales the image to a small, fixed size, converts it to grayscale, computes the DCT to capture its frequency components, retains only the low-frequency coefficients, and finally generates a binary hash based on whether each coefficient is above or below the median. The hash values are computed such that perceptually similar images will have hashes with a small Hamming distance. This approach can be extended to videos by computing the perceptual hashes on sampled frames.

3 The FEDDE Framework

FEDDE is a framework for data deduplication that targets decentralized/federated computing and machine learning environments.

3.1 Design Principles

We seek to achieve the following goals in designing FEDDE.

(1) **Generality.** Duplicates can be present in any dataset of any type. FEDDE should generally support deduplication tasks for any dataset or data type.
(2) **Flexibility.** There is an inherent trade-off in deduplication between FL training efficiency and model accuracy, depending on the amount of duplicates removed. FEDDE should allow users to make the trade-off for specific tasks.
(3) **Efficiency.** Many components in a federated environment can incur inefficiencies (e.g., client/server computation and network communication). FEDDE should optimize its deduplication workflow to achieve high efficiency.
(4) **Scalability.** FL can potentially involve many clients. FEDDE should scale gracefully as the number of clients increases.
(5) **Privacy Preservation.** As required in FL, private information contained in the raw data should never leave clients.

We next detail the architecture, deduplication protocols, API, and optimizations of FEDDE and show how these goals are accomplished.

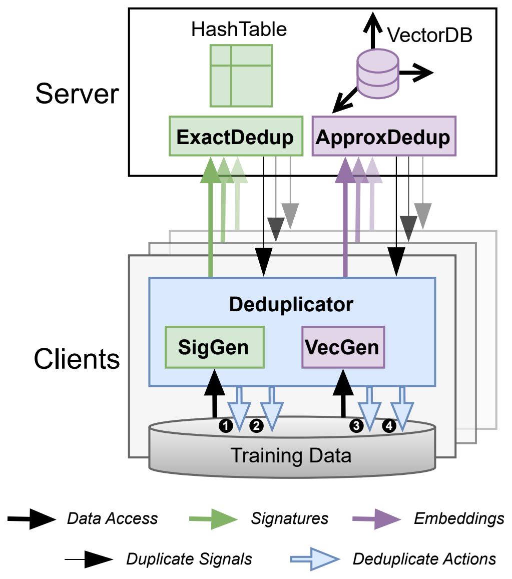

Figure 1: Architecture of FEDDE. Hierarchical deduplication maximizes FL training efficiency and model accuracy.

3.2 FEDDE's Architecture

FEDDE runs across FL clients and the server (coordinator) to deduplicate data at different levels, as shown in Figure 1. It performs data deduplication before iterative FL training begins [58].

A FEDDE `Deduplicator` is deployed on each client i and has access to the local, potentially duplicated training data D_i. The `Deduplicator` can read any training sample $S_i^j \in D_i$ and remove it from the training set if S_i^j is considered a duplicate for the later FL training. It can also generate a minimal record R_i^j based on S_i^j (privacy preservation is provided by `SigGen` or `VecGen` discussed in §3.3) and uploads R_i^j via secure and encrypted network connections to the server for deduplication purposes. The key insight behind FEDDE is that performing deduplication against the client's own data D_i, i.e., *local deduplication*, can only marginally remove redundant information at the federated scale (as shown in Section 5). To maximize the *deduplication ratio* and thus the efficiency of FL training, FEDDE adopts a *hierarchical* approach that also allows the deduplication of the training data of each client against the training data of all clients $D = \bigcup_{i \in \text{AllClients}} D_i$, i.e., *global deduplication*. Deduplication ratio measures how much data is removed from the original dataset and is defined as the ratio between original data size and deduplicated data size.

As the `Deduplicator` can access the raw data, the local deduplication can be performed using any existing, non-federated deduplication approach for specific tasks [45, 80]. FEDDE's global deduplication requires the cooperation with the server. Specifically, the server adopts a proper data structure F that supports two operations for duplicate detection. Upon receiving a record R_i^j, the server first queries F to determine if this is a duplicate:

$$F(R_i^j) \longrightarrow \{0, 1\}.$$

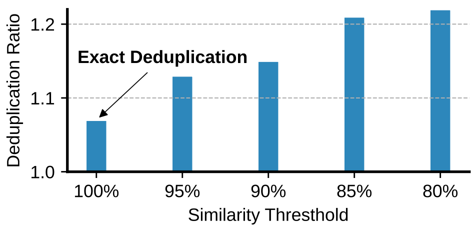

Figure 2: Approximate deduplication enables flexibility.

The duplicate signal is sent back to the client i. The server then performs a merging operation to incorporate the record in F:

$$F \leftarrow F \oplus R_i^j.$$

When the `Deduplicator` receives the duplicate detection result for a record R_i^j from the server, it removes the corresponding training sample from D_i if the server detects the record as a duplicate:

$$D_i \leftarrow D_i - S_i^j \quad \text{if } F(R_i^j) = 1.$$

We note that this overall architecture is deliberately simple for generality and efficiency reasons. It preserves the privacy of raw training data by sending only minimal records to the server, the same guarantee as provided in FL [58]. More technical contributions and how FEDDE achieves other goals are discussed next.

3.3 Deduplication Protocols

A key benefit of FEDDE is to allow flexible trade-offs between the training efficiency of FL and the accuracy of the trained model: users can decide which metric to prioritize based on their task preference by determining to what degree similar samples are considered duplicates in FEDDE. Flexibility is enabled by the detection of near-duplicates [45, 54], or approximate deduplication [22]. Indeed, flexibility is greatly restricted if "duplicates" are defined as identical samples only, i.e., exact deduplication. To show how approximate deduplication enables flexibility, we vary the Jaccard similarity threshold in MinHash to detect duplicates in a real-world text dataset of news articles [30] (with 5-grams and a signature size of 256, details in §4): lower thresholds mean similar documents are more likely to be considered duplicates, and 100% similarity equates to exact deduplication. Figure 2 shows the deduplication ratio resulted from each similarity threshold. As can be observed, exact deduplication only removes a marginal portion of the dataset, leading to limited improvement in training efficiency. In comparison, relaxing the similarity requirement for duplicate detection can significantly increase deduplication ratio. FEDDE extends approximate deduplication in the following aspects.

Hierarchical Approximate Deduplication. FEDDE first adapts approximate deduplication to its hierarchical architecture. Specifically, the `Deduplicator` on each FL client i adopts `VecGen`, which given a training sample S_i^j generates a vector of features $\mathbf{v}_i^j$, i.e., an embedding, as the record R_i^j. This vector is first used for local deduplication (Step ❸ in Figure 1): the `Deduplicator` runs an approximate nearest neighbor search (ANNS) over $\mathbf{v}_i^j$ against all local vectors. If the similarity between $\mathbf{v}_i^j$ and the vector of the nearest neighbor of S_i^j is above a threshold θ, S_i^j is considered a duplicate.

Algorithm 1 Server-side protocol

```
    // Round 1: Pre-filtering with exact deduplication
1:  for each $h_i^j$ received do
2:      if F_HashTable.exists($h_i^j$) then
3:          $signal \leftarrow$ true
4:      else
5:          $signal \leftarrow$ false
6:          F_HashTable.insert($h_i^j$)
7:      Respond $signal$ to client $i$
    // Round 2: Approximate deduplication
8:  for each $\mathbf{v}_i^j$ received do
9:      if F_VectorDB.anns($\mathbf{v}_i^j$).top(1).sim > $\theta$ then
10:         $signal \leftarrow$ true
11:     else
12:         $signal \leftarrow$ false
13:     Respond $signal$ to client $i$
14:     F_VectorDB.create($\mathbf{v}_i^j$)
```

Algorithm 2 Client(i)-side protocol

```
    // Round 1: Pre-filtering with exact deduplication
1:  for $S_i^j \in D_i$ do
2:      $h_i^j \leftarrow$ SigGen($S_i^j$)
3:      if $S_i^j$ survives local exact deduplication then
4:          $signal \leftarrow$ SendToServer($h_i^j$)
5:          if $singal$ = true then
6:              Remove $S_i^j$ from $D_i$
    // Round 2: Approximate deduplication
7:  for $S_i^j \in D_i$ do
8:      $\mathbf{v}_i^j \leftarrow$ VecGen($S_i^j$)
9:      if $S_i^j$ survives local approximate deduplication then
10:         $signal \leftarrow$ SendToServer($\mathbf{v}_i^j$)
11:         if $singal$ = true then
12:             Remove $S_i^j$ from $D_i$
```

For each locally-survived sample S_i^j, the `Deduplicator` sends its vector $\mathbf{v}_i^j$ to the server for global deduplication. Following FEDDE's architecture, the server determines if S_i^j is a duplicate against the union of all clients' samples. To perform approximate duplicate detection efficiently at FL scale, the server employs a vector database as the data structure F. Due to recent popularity [63], vector databases can now support fast vector insertions, index updates, and ANNS queries. The same threshold θ is used to determine if S_i^j is a duplicate when the nearest neighbor and similarity score for $\mathbf{v}_i^j$ are returned. This duplicate signal is then sent back to the `Deduplicator` on the client for the final deduplication processing (Step ❹): S_i^j is removed if the server signals a duplicate.

Improving Communication Efficiency via Exact Deduplication Pre-Filtering. Embeddings are often large records. For instance, a 256-dimensional integer embedding totals 2 KB, a non-negligible size for an image or a text document. Directly sending embeddings to the server thus compromises the efficiency goal. To

Table 1: FEDDE API.

API	Return	Functionality
`SigGen`(*Sample*)	Signature	Generate signature
`VecGen`(*Sample*)	Embedding	Generate embedding
`Init`(*SigGen*, *VecGen*, *ServerAddr*)	Deduplicator	Initialization
`Run`(*Deduplicator*, *Threshold*, *Path*)	#Duplicates	Execute deduplication

circumvent the overhead of transferring embeddings, FEDDE globally removes identical training samples with exact deduplication before approximate deduplication.

Specifically, the `Deduplicator` on a client adopts `SigGen`, a record generator that computes a hash signature h_i^j from each training sample S_i^j. The signature consists of a few bytes and is much smaller than an embedding, and two identical samples will generate the same signature. The `Deduplicator` first removes local samples that have the same signatures (Step ❶) and then sends the signature of each remaining sample to the server. A concurrent hash table keyed by signatures is adopted as the data structure on the server for exact deduplication. When a signature h_i^j is received, the server queries the hash table to generate a duplicate signal (1 if the signature exists, 0 otherwise) and inserts h_i^j into the table if it is a distinct signature. Finally, the `Deduplicator` keeps or removes S_i^j, depending on the duplicate signal (Step ❷).

Algorithms 1 and 2 summarize the server-side and client-side protocols, respectively. Pre-filtering identical samples can significantly reduce the overhead of approximate deduplication, and this two-round protocol is up to 30% more communication-efficient than directly performing hierarchical approximate deduplication.

3.4 Interface

Based on the two-round hierarchical deduplication protocols, we provide an API to instantiate a deduplication task as shown in Table 1. Specifically, the user first specifies how to generate a signature and an embedding by providing the `SigGen` and `VecGen` implementations. A deduplication job can be initialized with `Init`, which takes `SigGen`, `VecGen`, and the server's address as input and returns a `Deduplicator` object. To execute the deduplication job, `Run` is called to process the training samples in a specified data path (*Path*) using the `Deduplicator`. A similarity threshold (*Threshold*) is also provided to define how similar two embeddings are considered duplicates. The return value of a `Run` invocation indicates the total number of duplicates removed from the local training samples.

This API is general: *Sample* is an abstract type and can be instantiated to any type, *Signature* and *Embedding* can be reconfigured, and exact/approximate deduplication can be selectively enabled or disabled by specifying `SigGen`/`VecGen` or leaving it unspecified.

3.5 System Optimizations

In addition to the communication-efficient deduplication protocols, we further improve FEDDE's efficiency with several system-level optimizations centering around the compute tasks on FL clients and on the server as well as client-server data transfers.

Client Optimizations. The `Deduplicator` on each client primarily performs disk I/O to access and remove training samples, generates records, executes local deduplication, and communicates with the server. It is optimized with the following techniques.

- *Parallelized Data Loading and Processing (PDLP).* As FL clients usually have multi-core processing units [4], FEDDE spawns a deduplication worker thread on each core and partitions training samples across the workers, each loading the respective partition for further processing. When duplicate signals are received from the server, they are dispatched to individual workers for the final data removals. Each worker maintains a thread-local working set to minimize the synchronization overhead between workers.
- *SIMD-based Record Generation (SRG).* SIMD (Single Instruction Multiple Data) is increasingly supported on edge devices, e.g., Arm Neon [3]. FEDDE uses SIMD instructions to accelerate loops on the critical path when generating records (signatures and embeddings) from training samples.

Data Transfer Optimization. In FEDDE three types of messages are transferred between clients and the server: embeddings, signatures, and duplicate signals. The latter two are small. Transferring them individually under-utilizes network bandwidth: e.g., TCP requires a buffer of ~500 KB to maximize transfer performance in a network with 10 Mbps bandwidth and 50 ms latency. FEDDE adopts *Batched Data Transfers (BDT)*, where clients and the server batch sufficient data for each network transfer.

Server Optimizations. The hash table and the vector database the server adopts to support global deduplication are expected scale to a large number of clients and thus have stringent performance requirements. These two data structures are optimized as follows.

- *Sharded Signature Operations (SSO).* The concurrent hash table on the server is first implemented with Cuckoo hashing [62]. Concurrency control is realized by making bucket access a critical section, which becomes a scaling bottleneck when lookup and insertion are frequently invoked. To avoid contention, FEDDE shards the hash table by the key range, and each server thread manages one shard and executes operations accordingly. Network threads that receive signatures from clients dispatch them to the target shards.
- *Tuned Vector Indexing (TVI).* The vector database is implemented with the popular Faiss library [34], which supports high-performance ANNS queries. The library offers a variety of ANN indexing methods, including Inverted File Index (IVF), Hierarchical Navigable Small World (HNSW), and Locality Sensitive Hashing (LSH). FEDDE selects LSH as its indexing approach, which offers the highest performance for building the index, an operation the server executes for each embedding uploaded from clients.

Table 2 reports the performance improvement each of the above techniques enables in our evaluation setup (§5).

4 FEDDE in Practice

To demonstrate the practical uses of FEDDE, we implement federated deduplication for three common data types: text documents, images, and videos. The primary effort is to generate signatures and

Table 2: System optimizations in FEDDE and their effects.

	Client		Data Transfer	Server	
Optimization	PDLP	SRG	BDT	SSO	TVI
Improvement	4.5×	2.6×	9.3×	3.5×	60.8×

embeddings, i.e., specifying `SigGen` and `VecGen`. The end-to-end deduplication workflow is automatically executed. Table 3 shows duplicates in real-world datasets identified by FEDDE.

4.1 Text Deduplication

For exact deduplication, we treat each text chunk as a sample in FEDDE and generate a hash value as its signature. Specifically, we adopt content-defined chunking (CDC) [61, 80] to segment a document. Let a document T be a sequence of tokens over an alphabet Σ: $T = (t_1, t_2, ..., t_n), t_i \in \Sigma$. We apply Rabin fingerprinting [66] with a polynomial modulus, the average chunk size, p and a base b as the rolling hash over a window of characters $W_i = (t_{i-w+1}, ..., t_i)$:

$$H_i = \left(\sum_{j=0}^{w-1} t_{i-j}\, b^j\right) \bmod p$$

which can be incrementally computed in $O(1)$ time:

$$H_i = \left(\left(H_{i-1} - t_{i-w}\, b^{\,w-1}\right) b + t_i\right) \bmod p$$

Chunk boundaries are determined at each position: $H_i \bmod p = 0$. We compute an 8-byte hash value for each chunk as the signature.

Compared to fixed-size chunking, CDC is resilient to small edits (insertions and deletions) in a document and is more likely to identify duplicate chunks between documents.

For approximate deduplication, we detect near-duplicates at the document level with MinHash [6]. Each document is first converted to its 5-grams: $G = \{g_1, g_2, ..., g_m\}$, which is then used to compute an array of hash values $[h_1(G), h_2(G), ..., h_k(G)]$ as the embedding of the document. To compare two documents G_1 and G_2, their Jaccard similarity index [32] is approximated:

$$J(G_1, G_2) = \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|} \approx \frac{1}{k} \sum_{i=1}^{k} \mathbf{1}\{h_i(G_1) = h_i(G_2)\}$$

MinHash has been widely adopted for text near-duplicate detection [23, 45, 46, 83]. We set $k = 256$ to obtain 256-dimensional embeddings for approximate deduplication.

4.2 Image Deduplication

We generate a perceptual hash using a two-dimensional discrete cosine transformation (DCT) [52] as the signature for each image. Given a gray scaled image $X \in \mathbb{R}^{H\times W}$, the DCT of X is defined as:

$$Y(X)_{k_1,k_2} = \sum_{h=1}^{H} \sum_{w=1}^{W} x_{h,w} \cos\left[\frac{\pi}{H}\left(h + \frac{1}{2}\right) k_1\right] \cos\left[\frac{\pi}{W}\left(w + \frac{1}{2}\right) k_2\right]$$

This transformation decomposes an image and expresses it as a sum of cosine waves. For lower values of k_1 and k_2, Y_{k_1,k_2} is relatively low and can be interpreted as the smoother structures of the image. For example, $Y_{1,1}$ represents the image's overall brightness. As k_1 and k_2 increase, Y_{k_1,k_2} is able to capture finer details. To generate the signature, we flatten the top-left 8×8 segment of X's DCT DCT', compute the median value M of a batch of hash values over DCT', and finally produce a 64-bit value using the following function:

$$H(X) = \mathbb{I}[DCT'(X) > M]$$

Table 3: Near-duplicate examples in real-world datasets

Type	Dataset	Sample	Duplicate Sample
Text	CC News	...Some cities have a shot of setting record highs...The city's high temperature record for Feb. 19, set last year, was 44 at the Grand Forks International Airport...	...Some cities have a shot of setting record highs. The lowest record high for next week was set Feb. 19, 2016, at the Grand Forks International Airport with 44 degrees...
Image	LAION-5B		
Video	Kinetics		

Table 4: Datasets for data deduplication and FL evaluation

Text	**Dataset**: CC News [30], GitHub [28], USPatents [29] **FL Task** : Token Generation **Model**: Qwen3-0.6B [81]
Image	**Dataset**: CIFAR-100 [40], LAION-5B [69], Open Image [42] **FL Task** : Image Classification **Model**: PyramidNet [49]
Video	**Dataset**: Kinetics [36], Something-Something (SS) [21], Jester [56] **FL Task** : Action Recognition **Model**: VideoMAE [74]

To perform approximate deduplication, we use OpenAI's CLIP model [67] as the encoder to generate a 512-dimensional embedding for each image. To measure the similarity between two images, we take their embeddings and compute the cosine similarity.

4.3 Video Deduplication

We sample the key frames of a video and compute their DCTs and then perceptual hashes. To generate the final signature, we employ SimHash [8]. Given a set of b-bit hashes $\mathcal{H} = \{h_1, h_2, \ldots, h_k\}$, a b-bit SimHash SH is computed by signing each bit that is the weighted sum of the respective bit of all hashes in $\mathcal{H}$:

$$SH = \mathbb{I}[\sum_{i=1}^{k}(2\mathcal{H} - 1) > 0]$$

This signature is resilient to minor variations, such as slight shifts in scene boundaries or small edits within the video.

We use a pre-trained Transformer-based VideoMAE model [74] to generate an embedding for each survived video. The model extracts the feature representation from an intermediate layer of its encoder, which is further average-pooled across the spatial-temporal dimension to produce the final fixed-size embedding.

5 Evaluation

We evaluate FEDDE to answer the following questions:

Q1. How well can the deduplication enabled by FEDDE improve the training efficiency of FL, compared to baseline solutions?

Q2. How flexible is FEDDE to make the trade-off between FL training efficiency and FL-trained model accuracy?

Q3. How do the techniques in FEDDE contribute to its efficiency?

Table 5: Data reduced by different deduplication approaches for different types of data and the impacts on training efficiency and the accuracy of the trained models. Overall, FEDDE can effectively remove duplicates without degrading the model accuracy.

Metrics		**Data Reduction (%) ↑**			**Training Time (hours) ↓**				**Test Accuracy (%) ↑**			
Deduplication Solutions		LocalApprox	GlobalExact	FEDDE	Original	LocalApprox	GlobalExact	FEDDE	Original	LocalApprox	GlobalExact	FEDDE
Text	CC News	8.3	21.2	**51.3**	62.5	52.3	47.7	**36.2**	4.9	4.9	4.8	**4.7**
	GitHub	2.1	13.3	**30.5**	85.9	84.8	74.0	**58.3**	31.5	31.1	30.9	**30.3**
	USPatents	9.2	12.3	**24.1**	18.0	17.7	15.3	**14.8**	19.5	16.9	19.6	**17.0**
Image	CIFAR-100	4.5	13.0	**25.6**	2.1	2.0	1.9	**1.7**	22.9	22.0	19.5	**20.3**
	LAION-5B	15.3	28.1	**39.5**	0.5	0.5	0.4	**0.3**	46.6	46.3	46.8	**46.6**
	Open Image	4.3	38.1	**41.2**	0.90	0.88	0.77	**0.75**	39.3	42.2	43.5	**43.5**
Video	Kinetics	21.1	29.2	**91.7**	6.1	4.7	4.4	**3.6**	63.1	61.8	62.4	**63.4**
	SS	7.3	19.1	**33.3**	9.8	8.5	7.9	**6.9**	48.4	45.7	47.1	**47.5**
	Jester	8.4	16.2	**38.3**	22.3	20.3	19.1	**16.2**	90	89.5	89.5	**89.5**

Q4. How scalable is FEDDE with the number of FL clients?

5.1 Experimental Setup

Hardware and OS. All experiments are conducted on a machine equipped with an NVIDIA A800 GPU and a 4-socket AMD EPYC 7513 32-core processor (128 cores in total). The machine installs Ubuntu 24.04 as the operating system. In FEDDE, clients and server communicate via Linux TCP sockets and realistic network specs are emulated when investigating efficiency (§5.4).

Datasets. We use real-world datasets to evaluate data deduplication and the effects on FL for all data types in Section 4 and corresponding FL training tasks. Table 4 specifies the datasets, FL tasks, and the recent ML models to be trained. Due to resource constraints, we randomly sample a subset of classes from the massive datasets, e.g., Kinetics (700 GB) and LAION-5B (300 TB), such that each task can fit into our hardware capacity, and conduct due diligence to verify the existence of duplication in other classes.

FL Setup. We use the most recent version (v1.22.0) of the Flower framework [4, 20] to emulate a federated environment. By default, there are 10 clients, and the training samples in each dataset are partitioned among clients following a non-IID distribution (Dirichlet distribution with $\alpha = 1.0$). The global model is aggregated with FedAvg. Specifically, we train the following models in FL:

- For next-token generation, we fine-tune Qwen3-0.6 B [81] using Low-Rank Adaptation (LoRA) [26] with a rank of 8.
- For image classification, we train PyramidNet [49].
- For action recognition in videos, we fine-tune a pre-trained VideoMAE model [74].

All models are trained/fine-tuned iteratively with FL until convergence. We use perplexity [45] as the metric to evaluate the language model and show its multiplicative inverse to be consistent ("higher is better") with the classification accuracy in image and video tasks.

5.2 FEDDE Improves FL Training Efficiency

Our first set of experiments evaluates how much FL training efficiency can be improved with federated data deduplication in FEDDE. We fix the similarity threshold to 0.96 in all experiments in this section. We compare FEDDE with two baseline deduplication approaches: (1) *LocalAppox* deduplication where centralized approximate deduplication is performed on each client, using the same embeddings in FEDDE and the same similarity threshold, and (2) *GlobalExact* deduplication, which on top of the previous local deduplication further sends the hash signatures of survived samples to the server for global exact deduplication. For each deduplication solution, we measure (1) how much data in total is removed in terms of the percentage in the original data aggregated across all clients, (1) FL training time, and (3) the test accuracy of the trained model, covering all the data types and datasets in Section 5.1.

Table 5 shows the result. We make the following observations. First, FEDDE effectively eliminates data redundancy: up to 6× and 3× more duplicates can be removed across the datasets compared to *LocalApprox* and *GlobalExact*, respectively. Second, this data reduction directly translates to FL training efficiency. FL can be up to 42% faster when trained on a FEDDE-deduplicated dataset than on the original dataset, and 31% and 24% faster than *LocalApprox* and *GlobalExact*. Finally, as we adopt a conservative similarity threshold, the FL training efficiency is gained with FEDDE while maintaining the model accuracy. In fact, in some cases FEDDE can even improve model accuracy: FL achieves 63.4% accuracy with FEDDE-deduplicated Kinetics vs. 63.1% with the original dataset.

Summary. *Removing data redundancy between FL clients with FEDDE can significantly improve FL training efficiency up to 42% without sacrificing the accuracy of FL-trained models.*

5.3 FEDDE Enables Flexibility

We next investigate the impacts of different similarity thresholds in approximate deduplication on FL training efficiency and model accuracy. For this experiment, we vary the similarity threshold in both FEDDE and *LocalApprox* from 1 (highest) to 0.8 (relatively low) and measure FL training time and the test accuracy of the trained model accordingly for all data types. We randomly sample a subset of each original dataset to keep the evaluation time manageable.

As shown in Figure 3, we first observe the trade-off between FL training efficiency and model accuracy in every task and dataset: by aggressively removing similar training samples with low similarity thresholds, training times (i.e., times to convergence) are significantly shorter, while the final model accuracies drop as the

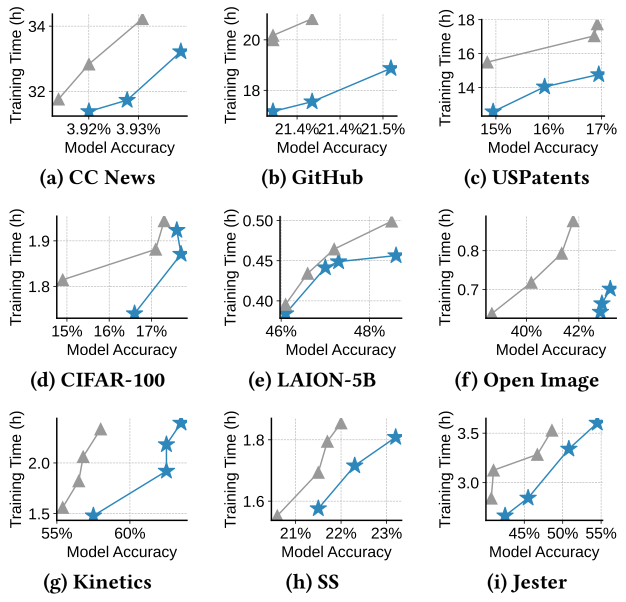

Figure 3: Trade-off between FL training efficiency and model accuracy enabled by approximate deduplication. ★ represents Fedde and ▲ represents *LocalApprox*.

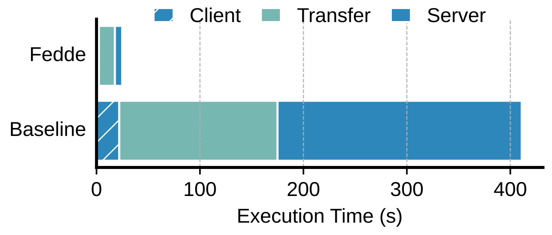

Figure 4: Execution time breakdown in Fedde's workflow.

models are under-trained with less data. Second, benefiting from its two-round hierarchical deduplication protocols, the trade-offs Fedde makes are strictly more efficient than *LocalApprox*: with the same final model accuracy, FL training is much faster with Fedde-deduplicated datasets, and vice versa.

Summary. *Fedde enables flexible trade-offs between FL training efficiency and model accuracy by tuning the similarity threshold. Compared to approximate deduplication on individual clients, Fedde's results form an efficient Pareto frontier.*

5.4 Fedde Minimizes Deduplication Cost

To show the performance of Fedde's deduplication workflow, we compare it with a baseline execution where the optimization techniques in Section 3.5 are disabled. In addition, the baseline does not employ exact duplication pre-filtering and thus incurs higher communication overhead for sending more embeddings.

Figure 4 shows the time in each breakdown component when deduplicating the CC News dataset (similar findings apply to other data types and datasets): processing times on a client and the server, as well as the data transfer time between one client and the server over the network (assuming 10 Mbps network bandwidth). Fedde is 17× faster than the baseline. The speedups on the client and the server (9.6× and 34×, respectively) primarily originate from the execution optimizations discussed in Section 3.5. There are two sources of improvement for communication: (1) pre-filtering exact duplicates in the first round saves 35.5% of embeddings sent from the client the server in the second round for approximate deduplication, and (2) the batched data transfer optimization (§3.5) further brings a 9× improvement.

Summary. *The two-round protocols and system optimizations in Fedde significantly improve deduplication efficiency—federated deduplication with Fedde can be 17× faster than the baseline.*

5.5 Fedde Is Scalable

Finally, we evaluate whether Fedde scales gracefully when the number of clients increases. Specifically, we let each client hold a fraction of, without loss of generality, the CC News dataset and vary the number of clients with the fixed data size. We measure the volume of data transferred in the network for deduplication. In particular, we compare Fedde with an alternative solution where each client broadcasts records to all other clients for duplicate detection, a communication approach adopted in recent work [1]. Figure 5 reports the result. We observe that Fedde scales linearly with the number of clients and saves 99% of network traffic compared to client-to-client broadcast when scaling to 150 clients.

Summary. *Fedde only requires transferring data between a client and the server without interferences between clients and can thus scale to many clients.*

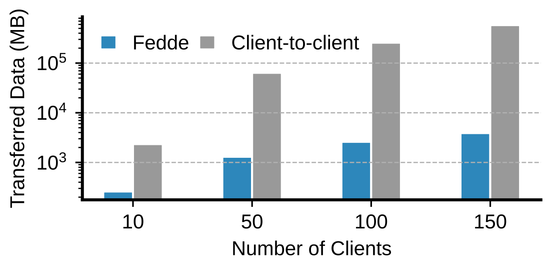

Figure 5: Scalability with the number of clients.

6 Related Work

Domain-specific Data Deduplication. Data deduplication has been an active research topic in multiple domains of computer science, including databases [2, 11, 16, 31, 38, 85], storage [7, 12, 13, 33, 41, 48, 53, 59, 73, 87], distributed systems and networking [15, 27, 47, 64, 72, 79, 80], natural language processing [25, 45, 60], image and video processing [14, 35, 50, 65, 71], and the Web [17, 39, 54]. For instance, Dis-Dedup [11] and Dedoop [38] are two distributed relational database deduplication approaches leveraging MapReduce. DeDe [12] is a protocol for storage-area network (SAN)-connected servers. FastCDC [80] proposes an optimized content-defined chunking (CDC) for fast data deduplication. Lee et al. [45] investigated the effect of text deduplication (with exact and approximate string matches) for training large-language models. ViDeDup [35] and Maze [65] are video deduplication solutions. Koppula et al. [39] proposed to deduplicate web pages by normalizing URLs. Manku et al. [54] presented a fingerprint-based algorithm to detect near-duplicate web pages. Compared to these domain-specific deduplication approaches, FEDDE is distinct in its FL focus and optimization design and extensibility to any data type.

Improving Data Quality for FL. Several recent works proposed solutions to improve various aspects of data quality for FL. Specifically, EP-MPD [1] introduces group private set intersection (G-PSI) to detect and remove identical text training samples for language model training with FL. Compared to FEDDE, this approach is restricted to small scales, as it requires direct message passing between FL clients, deduplicating exact training samples, and language tasks only. AugHFL [18] is a two-stage approach to mitigating the impact of data corruption on FL clients. Finally, FedClean [51] performs data cleaning in FL. with attribute value frequency (AVF). FEDDE is orthogonal to and can benefit from approaches to data quality issues other than duplicates.

7 Conclusion

We have introduced FEDDE, a framework for removing redundant data on FL clients to improve training efficiency. It achieves (1) privacy preservation with a hierarchical architecture, (2) flexibility with approximate deduplication, (3) efficiency with two-round protocols and effective system optimizations, (4) scalability by transferring data only between clients and the server, and (5) generality with a generic API that can accommodate any data type and dataset. We have implemented federated deduplication solutions for several common data types with FEDDE, and its efficacy has been extensively evaluated with real-world datasets and models.

References

[1] Aydin Abadi, Vishnu Asutosh Dasu, and Sumanta Sarkar. 2025. Privacy-Preserving Data Deduplication for Enhancing Federated Learning of Language Models. In *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025.* The Internet Society.

[2] Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. 2002. Eliminating Fuzzy Duplicates in Data Warehouses. In *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002.* Morgan Kaufmann, 586–597. doi:10.1016/B978-155860869-6/50058-5

[3] Arm Developer. 2025. Neon. https://developer.arm.com/Architectures/Neon.

[4] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Hei Li Kwing, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. 2020. Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390* (2020).

[5] Kallista A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. In *Proceedings of the Second Conference on Machine Learning and Systems, SysML 2019, Stanford, CA, USA, March 31 - April 2, 2019*, Ameet Talwalkar, Virginia Smith, and Matei Zaharia (Eds.). mlsys.org. https://proceedings.mlsys.org/paper_files/paper/2019/hash/7b770da633baf74895be22a8807f1a8f-Abstract.html

[6] Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *Compression and Complexity of SEQUENCES 1997, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997, Proceedings*, Bruno Carpentieri, Alfredo De Santis, Ugo Vaccaro, and James A. Storer (Eds.). IEEE, 21–29. doi:10.1109/SEQUEN.1997.666900

[7] Zhichao Cao, Shiyong Liu, Fenggang Wu, Guohua Wang, Bingzhe Li, and David H. C. Du. 2019. Sliding Look-Back Window Assisted Data Chunk Rewriting for Improving Deduplication Restore Performance. In *17th USENIX Conference on File and Storage Technologies, FAST 2019, Boston, MA, February 25-28, 2019*, Arif Merchant and Hakim Weatherspoon (Eds.). USENIX Association, 129–142. https://www.usenix.org/conference/fast19/presentation/cao

[8] Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 380–388.

[9] Qian Chen, Zilong Wang, Jiaqi Hu, Haonan Yan, Jianying Zhou, and Xiaodong Lin. 2024. PAGE: Equilibrate personalization and generalization in federated learning. In *Proceedings of the ACM Web Conference 2024*. 2955–2964.

[10] Wenlin Chen, Samuel Horváth, and Peter Richtárik. 2022. Optimal Client Sampling for Federated Learning. *Trans. Mach. Learn. Res.* 2022 (2022). https://openreview.net/forum?id=8GvRCWKHIL

[11] Xu Chu, Ihab F. Ilyas, and Paraschos Koutris. 2016. Distributed Data Deduplication. *Proc. VLDB Endow.* 9, 11 (2016), 864–875. doi:10.14778/2983200.2983203

[12] Austin T. Clements, Irfan Ahmad, Murali Vilayannur, and Jinyuan Li. 2009. Decentralized Deduplication in SAN Cluster File Systems. In *Proceedings of the 2009 USENIX Annual Technical Conference, USENIX ATC 2009, San Diego, CA, USA, June 14-19, 2009*, Geoffrey M. Voelker and Alec Wolman (Eds.). USENIX Association. https://www.usenix.org/conference/usenix-09/decentralized-deduplication-san-cluster-file-systems

[13] Biplob K. Debnath, Sudipta Sengupta, and Jin Li. 2010. ChunkStash: Speeding Up Inline Storage Deduplication Using Flash Memory. In *Proceedings of the 2010 USENIX Annual Technical Conference, USENIX ATC 2010, Boston, MA, USA, June 23-25, 2010*, Paul Barham and Timothy Roscoe (Eds.). USENIX Association. https://www.usenix.org/conference/usenix-atc-10/chunkstash-speeding-inline-storage-deduplication-using-flash-memory

[14] Rui Deng, Qian Wu, and Yuke Li. 2023. 3D-CSL: Self-Supervised 3D Context Similarity Learning for Near-Duplicate Video Retrieval. In *IEEE International Conference on Image Processing, ICIP 2023, Kuala Lumpur, Malaysia, October 8-11, 2023*. IEEE, 2880–2884. doi:10.1109/ICIP49359.2023.10222915

[15] Ahmed El-Shimi, Ran Kalach, Ankit Kumar, Adi Ottean, Jin Li, and Sudipta Sengupta. 2012. Primary Data Deduplication - Large Scale Study and System Design. In *Proceedings of the 2012 USENIX Annual Technical Conference, USENIX ATC 2012, Boston, MA, USA, June 13-15, 2012*, Gernot Heiser and Wilson C. Hsieh (Eds.). USENIX Association, 285–296. https://www.usenix.org/conference/atc12/technical-sessions/presentation/el-shimi

[16] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.* 19, 1 (2007), 1–16. doi:10.1109/TKDE.2007.250581

[17] Li Fan, Pei Cao, Jussara M. Almeida, and Andrei Z. Broder. 1998. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *Proceedings of the ACM SIGCOMM 1998 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 31 - September 4, 1998, Vancouver, B.C., Canada*, Gerald Neufeld, Gary S. Delp, Jonathan Smith, and Martha Steenstrup (Eds.). ACM, 254–265. doi:10.1145/285237.285287

[18] Xiuwen Fang, Mang Ye, and Xiyuan Yang. 2023. Robust Heterogeneous Federated Learning under Data Corruption. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*. IEEE, 4997–5007. doi:10.1109/ICCV51070.2023.00463

[19] Hany Farid. 2021. An overview of perceptual hashing. *Journal of Online Trust and Safety* 1, 1 (2021).

[20] GitHub. 2025. Flower: A Friendly Federated AI Framework. https://github.com/adap/flower.

[21] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. 2017. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*. 5842–5850.

[22] Peizhen Guo and Wenjun Hu. 2018. Potluck: Cross-Application Approximate Deduplication for Computation-Intensive Mobile Applications. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2018, Williamsburg, VA, USA, March 24-28, 2018*, Xipeng Shen, James Tuck, Ricardo Bianchini, and Vivek Sarkar (Eds.). ACM, 271–284. doi:10.1145/3173162.3173185

[23] Bikash Gyawali, Lucas Anastasiou, and Petr Knoth. 2020. Deduplication of Scholarly Documents using Locality Sensitive Hashing and Word Embeddings. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis (Eds.). European Language Resources Association, 901–910. https://aclanthology.org/2020.lrec-1.113/

[24] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated Learning for Mobile Keyboard Prediction. *CoRR* abs/1811.03604 (2018). arXiv:1811.03604 http://arxiv.org/abs/1811.03604

[25] Nan He, Weichen Xiong, Hanwen Liu, Yi Liao, Lei Ding, Kai Zhang, Guohua Tang, Xiao Han, and Yang Wei. 2024. SoftDedup: an Efficient Data Reweighting Method for Speeding Up Language Model Pre-training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, 4011–4022. doi:10.18653/V1/2024.ACL-LONG.220

[26] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[27] Yu Hua, Xue Liu, and Dan Feng. 2013. Smart in-network deduplication for storage-aware SDN. In *ACM SIGCOMM 2013 Conference, SIGCOMM 2013, Hong Kong, August 12-16, 2013*, Dah Ming Chiu, Jia Wang, Paul Barford, and Srinivasan Seshan (Eds.). ACM, 509–510. doi:10.1145/2486001.2491714

[28] Hugging Face. 2022. Dataset Card for GitHub Code. https://huggingface.co/datasets/codeparrot/github-code.

[29] Hugging Face. 2022. Dataset Card for Pile of Law. https://huggingface.co/datasets/pile-of-law/pile-of-law.

[30] Hugging Face. 2025. Dataset Card for CC-News. https://huggingface.co/datasets/vblagoje/cc_news.

[31] Ihab F. Ilyas and Xu Chu. 2015. Trends in Cleaning Relational Data: Consistency and Deduplication. *Found. Trends Databases* 5, 4 (2015), 281–393. doi:10.1561/1900000045

[32] Paul Jaccard. 1912. The Distribution of the Flora in the Alpine Zone. *The New Phytologist* 11, 2 (1912), 37–50. http://www.jstor.org/stable/2427226

[33] Navendu Jain, Michael Dahlin, and Renu Tewari. 2005. TAPER: Tiered Approach for Eliminating Redundancy in Replica Synchronization. In *Proceedings of the FAST '05 Conference on File and Storage Technologies, December 13-16, 2005, San Francisco, California, USA*, Garth Gibson (Ed.). USENIX. http://www.usenix.org/events/fast05/tech/jain.html

[34] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[35] Atul Katiyar and Jon B. Weissman. 2011. ViDeDup: An Application-Aware Framework for Video De-duplication. In *3rd USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage'11, Portland, OR, USA, June 14, 2011*, Irfan Ahmad (Ed.). USENIX Association. https://www.usenix.org/conference/hotstorage11/videdup-application-aware-framework-video-de-duplication

[36] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950* (2017).

[37] Sarit Khirirat, Sindri Magnússon, Arda Aytekin, and Mikael Johansson. 2021. A Flexible Framework for Communication-Efficient Machine Learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 8101–8109. doi:10.1609/AAAI.V35I9.16987

[38] Lars Kolb, Andreas Thor, and Erhard Rahm. 2012. Dedoop: Efficient Deduplication with Hadoop. *Proc. VLDB Endow.* 5, 12 (2012), 1878–1881. doi:10.14778/2367502.2367527

[39] Hema Swetha Koppula, Krishna P. Leela, Amit Agarwal, Krishna Prasad Chitrapura, Sachin Garg, and Amit Sasturkar. 2010. Learning URL patterns for webpage de-duplication. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu (Eds.). ACM, 381–390. doi:10.1145/1718487.1718535

[40] Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. (2009), 32–33. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[41] Purushottam Kulkarni, Fred Douglis, Jason D. LaVoie, and John M. Tracey. 2004. Redundancy Elimination Within Large Collections of Files. In *Proceedings of the General Track: 2004 USENIX Annual Technical Conference, June 27 - July 2, 2004, Boston Marriott Copley Place, Boston, MA, USA*. USENIX, 59–72. http://www.usenix.org/publications/library/proceedings/usenix04/tech/general/kulkarni.html

[42] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. 2020. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision* 128, 7 (2020), 1956–1981.

[43] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient Federated Learning via Guided Participant Selection. In *15th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2021, July 14-16, 2021*, Angela Demke Brown and Jay R. Lorch (Eds.). USENIX Association, 19–35. https://www.usenix.org/conference/osdi21/presentation/lai

[44] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2021. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499* (2021).

[45] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating Training Data Makes Language Models Better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 8424–8445. doi:10.18653/V1/2022.ACL-LONG.577

[46] Johannes Leveling, Lennard Helmer, Benny Jörg Stein, Dennis Wegener, Zoha Sheikh, Elanton Fernandes, and Hammam Abdelwahab. 2024. Evaluation of Document Deduplication Algorithms for Large Text Corpora. In *Machine Learning, Optimization, and Data Science - 10th International Conference, LOD 2024, Castiglione della Pescaia, Italy, September 22-25, 2024, Revised Selected Papers, Part I (Lecture Notes in Computer Science, Vol. 15508)*, Giuseppe Nicosia, Varun Ojha, Sven Giesselbach, Panos M. Pardalos, and Renato Umeton (Eds.). Springer, 390–404. doi:10.1007/978-3-031-82481-4_27

[47] Shijing Li, Tian Lan, Bharath Balasubramanian, Moo-Ryong Ra, Hee Won Lee, and Rajesh K. Panta. 2019. EF-Dedup: Enabling Collaborative Data Deduplication at the Network Edge. In *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019, Dallas, TX, USA, July 7-10, 2019*. IEEE, 986–996. doi:10.1109/ICDCS.2019.00102

[48] Mark Lillibridge, Kave Eshghi, Deepavali Bhagwat, Vinay Deolalikar, Greg Trezis, and Peter Camble. 2009. Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality. In *7th USENIX Conference on File and Storage Technologies, February 24-27, 2009, San Francisco, CA, USA. Proceedings*, Margo I. Seltzer and Richard Wheeler (Eds.). USENIX, 111–123. http://www.usenix.org/events/fast09/tech/full_papers/lillibridge/lillibridge.pdf

[49] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.

[50] Yao Liu, Sam Blasiak, Weijun Xiao, Zhenhua Li, and Songqing Chen. 2015. A Quantitative Study of Video Duplicate Levels in YouTube. In *Passive and Active Measurement - 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings (Lecture Notes in Computer Science, Vol. 8995)*, Jelena Mirkovic and Yong Liu (Eds.). Springer, 235–248. doi:10.1007/978-3-319-15509-8_18

[51] Lichuan Ma, Qingqi Pei, Lu Zhou, Haojin Zhu, Licheng Wang, and Yusheng Ji. 2021. Federated Data Cleaning: Collaborative and Privacy-Preserving Data Cleaning for Edge Intelligence. *IEEE Internet Things J.* 8, 8 (2021), 6757–6770. doi:10.1109/JIOT.2020.3027980

[52] J. Makhoul. 1980. A fast cosine transform in one and two dimensions. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28, 1 (1980), 27–34. doi:10.1109/TASSP.1980.1163351

[53] Udi Manber. 1994. Finding Similar Files in a Large File System. In *USENIX Winter 1994 Technical Conference, San Francisco, California, USA, January 17-21, 1994, Conference Proceedings*. USENIX Association, 1–10. https://www.usenix.org/conference/usenix-winter-1994-technical-conference/finding-similar-files-large-file-system

[54] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy (Eds.). ACM, 141–150. doi:10.1145/1242572.1242592

[55] Yuzhu Mao, Zihao Zhao, Guangfeng Yan, Yang Liu, Tian Lan, Linqi Song, and Wenbo Ding. 2022. Communication-Efficient Federated Learning with Adaptive Quantization. *ACM Trans. Intell. Syst. Technol.* 13, 4 (2022), 67:1–67:26. doi:10.1145/3510587

[56] Joanna Materzynska, Guillaume Berger, Ingo Bax, and Roland Memisevic. 2019. The jester dataset: A large-scale video dataset of human gestures. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*. 0–0.

[57] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[58] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Xiaojin (Jerry) Zhu (Eds.). PMLR, 1273–1282.

[59] Dutch T. Meyer and William J. Bolosky. 2011. A Study of Practical Deduplication. In *9th USENIX Conference on File and Storage Technologies, San Jose, CA, USA, February 15-17, 2011*, Gregory R. Ganger and John Wilkes (Eds.). USENIX, 1–13. http://www.usenix.org/events/fast11/tech/techAbstracts.html#Meyer

[60] Yida Mu, Mali Jin, Xingyi Song, and Nikolaos Aletras. 2024. Enhancing Data Quality through Simple De-duplication: Navigating Responsible Computational Social Science Research. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 12477–12492. doi:10.18653/V1/2024.EMNLP-MAIN.694

[61] Athicha Muthitacharoen, Benjie Chen, and David Mazières. 2001. A Low-Bandwidth Network File System. In *Proceedings of the 18th ACM Symposium on Operating System Principles, SOSP 2001, Chateau Lake Louise, Banff, Alberta, Canada, October 21-24, 2001*, Keith Marzullo and Mahadev Satyanarayanan (Eds.). ACM, 174–187. doi:10.1145/502034.502052

[62] Rasmus Pagh and Flemming Friche Rodler. 2001. Cuckoo Hashing. In *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2161)*, Friedhelm Meyer auf der Heide (Ed.). Springer, 121–133. doi:10.1007/3-540-44676-1_10

[63] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Survey of vector database management systems. *VLDB J.* 33, 5 (2024), 1591–1615. doi:10.1007/S00778-024-

00864-X

[64] Himabindu Pucha, David G. Andersen, and Michael Kaminsky. 2007. Exploiting Similarity for Multi-Source Downloads Using File Handprints. In *4th Symposium on Networked Systems Design and Implementation (NSDI 2007), April 11-13, 2007, Cambridge, Massachusetts, USA, Proceedings*, Hari Balakrishnan and Peter Druschel (Eds.). USENIX. http://www.usenix.org/events/nsdi07/tech/pucha.html

[65] An Qin, Mengbai Xiao, Ben Huang, and Xiaodong Zhang. 2022. Maze: A Cost-Efficient Video Deduplication System at Web-scale. In *MM '22: The 30th ACM International Conference on Multimedia, Lisboa, Portugal, October 10 - 14, 2022*, João Magalhães, Alberto Del Bimbo, Shin'ichi Satoh, Nicu Sebe, Xavier Alameda-Pineda, Qin Jin, Vincent Oria, and Laura Toni (Eds.). ACM, 3163–3172. doi:10.1145/3503161.3548145

[66] Michael O Rabin. 1981. *Fingerprinting by random polynomials*. Ph. D. Dissertation. Cambridge, MA, USA.

[67] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. *CoRR* abs/2103.00020 (2021). arXiv:2103.00020 https://arxiv.org/abs/2103.00020

[68] Alexandra Schofield, Laure Thompson, and David M. Mimno. 2017. Quantifying the Effects of Text Duplication on Semantic Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, Martha Palmer, Rebecca Hwa, and Sebastian Riedel (Eds.). Association for Computational Linguistics, 2737–2747. doi:10.18653/V1/D17-1290

[69] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems* 35 (2022), 25278–25294.

[70] Emily Silcock, Luca D'Amico-Wong, Jinglin Yang, and Melissa Dell. 2023. Noise-Robust De-Duplication at Scale. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/forum?id=bAz2DBS35i

[71] Eric Slyman, Stefan Lee, Scott Cohen, and Kushal Kafle. 2024. FairDeDup: Detecting and Mitigating Vision-Language Fairness Disparities in Semantic Dataset Deduplication. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*. IEEE, 13905–13916. doi:10.1109/CVPR52733.2024.01319

[72] Neil T. Spring and David Wetherall. 2000. A protocol-independent technique for eliminating redundant network traffic. In *Proceedings of the ACM SIGCOMM 2000 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 28 - September 1, 2000, Stockholm, Sweden*, Craig Partridge (Ed.). ACM, 87–95. doi:10.1145/347059.347408

[73] Kiran Srinivasan, Timothy Bisson, Garth R. Goodson, and Kaladhar Voruganti. 2012. iDedup: latency-aware, inline data deduplication for primary storage. In *Proceedings of the 10th USENIX conference on File and Storage Technologies, FAST 2012, San Jose, CA, USA, February 14-17, 2012*, William J. Bolosky and Jason Flinn (Eds.). USENIX Association, 24. https://www.usenix.org/conference/fast12/idedup-latency-aware-inline-data-deduplication-primary-storage

[74] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. 2022. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems* 35 (2022), 10078–10093.

[75] Runze Wang, Jiahao Liu, Miao Hu, Yipeng Zhou, and Di Wu. 2025. Local Differentially Private Release of Infinite Streams With Temporal Relevance. In *Proceedings of the ACM on Web Conference 2025*. 921–930.

[76] Dingzhu Wen, Ki Jun Jeon, and Kaibin Huang. 2022. Federated Dropout - A Simple Approach for Enabling Federated Learning on Resource Constrained Devices. *IEEE Wirel. Commun. Lett.* 11, 5 (2022), 923–927. doi:10.1109/LWC.2022.3149783

[77] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. 2022. Communication-efficient federated learning via knowledge distillation. *Nature communications* 13, 1 (2022), 2032.

[78] Xueyang Wu, Hengguan Huang, Youlong Ding, Hao Wang, Ye Wang, and Qian Xu. 2023. FedNP: Towards Non-IID Federated Learning via Federated Neural Propagation. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, Brian Williams, Yiling Chen, and Jennifer Neville (Eds.). AAAI Press, 10399–10407. doi:10.1609/AAAI.V37I9.26237

[79] Wen Xia, Hong Jiang, Dan Feng, and Yu Hua. 2011. SiLo: A Similarity-Locality based Near-Exact Deduplication Scheme with Low RAM Overhead and High Throughput. In *Proceedings of the 2011 USENIX Annual Technical Conference, USENIX ATC 2011, Portland, OR, USA, June 15-17, 2011*, Jason Nieh and Carl A. Waldspurger (Eds.). USENIX Association. https://www.usenix.org/conference/usenixatc11/silo-similarity-locality-based-near-exact-deduplication-scheme-low-ram

[80] Wen Xia, Yukun Zhou, Hong Jiang, Dan Feng, Yu Hua, Yuchong Hu, Qing Liu, and Yucheng Zhang. 2016. FastCDC: a Fast and Efficient Content-Defined Chunking Approach for Data Deduplication. In *Proceedings of the 2016 USENIX Annual Technical Conference, USENIX ATC 2016, Denver, CO, USA, June 22-24, 2016*, Ajay Gulati and Hakim Weatherspoon (Eds.). USENIX Association, 101–114. https://www.usenix.org/conference/atc16/technical-sessions/presentation/xia

[81] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388* (2025).

[82] Ming Yin, Yichang Xu, Minghong Fang, and Neil Zhenqiang Gong. 2024. Poisoning federated recommender systems with fake users. In *Proceedings of the ACM Web Conference 2024*. 3555–3565.

[83] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, Chen Li, Ziyan Gong, Yifan Yao, Xinjing Huang, Jun Wang, Jianfeng Yu, Qi Guo, Yue Yu, Yan Zhang, Jin Wang, Hengtao Tao, Dasen Yan, Zexuan Yi, Fang Peng, Fangqing Jiang, Han Zhang, Lingfeng Deng, Yehong Zhang, Zhe Lin, Chao Zhang, Shaojie Zhang, Mingyue Guo, Shanzhi Gu, Gaojun Fan, Yaowei Wang, Xuefeng Jin, Qun Liu, and Yonghong Tian. 2021. PanGu-α: Large-scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation. *CoRR* abs/2104.12369 (2021). arXiv:2104.12369 https://arxiv.org/abs/2104.12369

[84] Chunxu Zhang, Guodong Long, Tianyi Zhou, Zijian Zhang, Peng Yan, and Bo Yang. 2024. When federated recommendation meets cold-start problem: Separating item attributes and user interactions. In *Proceedings of the ACM Web Conference 2024*. 3632–3642.

[85] Zihao Zhang, Huiqi Hu, Zhihui Xue, Changcheng Chen, Yang Yu, Cuiyun Fu, Xuan Zhou, and Feifei Li. 2021. SLIMSTORE: A Cloud-based Deduplication System for Multi-version Backups. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 1841–1846. doi:10.1109/ICDE51399.2021.00164

[86] Boxin Zhao, Ziqi Liu, Chaochao Chen, Mladen Kolar, Zhiqiang Zhang, and Jun Zhou. 2021. Adaptive Client Sampling in Federated Learning via Online Learning with Bandit Feedback. *CoRR* abs/2112.14332 (2021). arXiv:2112.14332 https://arxiv.org/abs/2112.14332

[87] Benjamin Zhu, Kai Li, and R. Hugo Patterson. 2008. Avoiding the Disk Bottleneck in the Data Domain Deduplication File System. In *6th USENIX Conference on File and Storage Technologies, FAST 2008, February 26-29, 2008, San Jose, CA, USA*, Mary Baker and Erik Riedel (Eds.). USENIX, 269–282. http://www.usenix.org/events/fast08/tech/zhu.html