Privacy and Accuracy-Aware AI/ML Model Deduplication

HONG GUAN, Arizona State University, USA
LEI YU, Rensselaer Polytechnic Institute, USA
LIXI ZHOU, Arizona State University, USA
LI XIONG, Emory University, USA
KANCHAN CHOWDHURY, Arizona State University, USA
LULU XIE, Arizona State University, USA
XUSHENG XIAO, Arizona State University, USA
JIA ZOU, Corresponding author, Arizona State University, USA

With the growing adoption of privacy-preserving machine learning algorithms, such as Differentially Private Stochastic Gradient Descent (DP-SGD), training or fine-tuning models on private datasets has become increasingly prevalent. This shift has led to the need for models offering varying privacy guarantees and utility levels to satisfy diverse user requirements. Managing numerous versions of large models introduces significant operational challenges, including increased inference latency, higher resource consumption, and elevated costs. Model deduplication is a technique widely used by many model serving and database systems to support high-performance and low-cost inference queries and model diagnosis queries. However, none of the existing model deduplication works has considered privacy, leading to unbounded aggregation of privacy costs for certain deduplicated models and inefficiencies when applied to deduplicate DP-trained models.

We formalize the problem of deduplicating DP-trained models for the first time and propose a novel privacy- and accuracy-aware deduplication mechanism to address the problem. We developed a greedy strategy to select and assign base models to target models to minimize storage and privacy costs. When deduplicating a target model, we dynamically schedule accuracy validations and apply the Sparse Vector Technique to reduce the privacy costs associated with private validation data. Compared to baselines, our approach improved the compression ratio by up to 35× for individual models (including large language models and vision transformers). We also observed up to 43× inference speedup due to the reduction of I/O operations.

CCS Concepts: • **Security and privacy** → **Privacy protections**; • **Information systems** → *Database management system engines.*

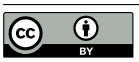
Additional Key Words and Phrases: Model deduplication, Differential privacy, Model management

ACM Reference Format:
Hong Guan, Lei Yu, Lixi Zhou, Li Xiong, Kanchan Chowdhury, Lulu Xie, Xusheng Xiao, and Jia Zou. 2025. Privacy and Accuracy-Aware AI/ML Model Deduplication . *Proc. ACM Manag. Data* 3, 3 (SIGMOD), Article 203 (June 2025), 28 pages. https://doi.org/10.1145/3725340

1 Introduction

AI/ML systems that support model marketplace [15, 54, 57], ML-as-a-Service (MLaaS) [44, 61, 86], and multi-tasking models on edge devices [52, 92] require efficient management of many models, including large language models (LLMs). These models are often trained or fine-tuned on

Authors' Contact Information: Hong Guan, Arizona State University, Tempe, USA; Lei Yu, Rensselaer Polytechnic Institute, Troy, USA; Lixi Zhou, Arizona State University, Tempe, USA; Li Xiong, Emory University, Atlanta, USA; Kanchan Chowdhury, Arizona State University, Tempe, USA; Lulu Xie, Arizona State University, Tempe, USA; Xusheng Xiao, Arizona State University, Tempe, USA; Jia Zou, Corresponding author, Arizona State University, Tempe, USA.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2025 Copyright held by the owner/author(s).
ACM 2836-6573/2025/6-ART203
https://doi.org/10.1145/3725340

private datasets using privacy-preserving algorithms [88, 89], such as DP-SGD [1]. Such algorithms are widely used to protect training data from attacks, such as re-identification [3], membership inference [42, 72], and model inversion [31]. The management of many DP models poses new challenges:

• Model marketplaces [15, 54, 57] train many versions of a model using different privacy budgets (ϵ). Then, these models are sold to different buyers so that models with higher ϵ (less noise) and higher accuracy will require a higher price, providing more options to potential buyers, while ensuring arbitrage freeness that no model buyers can take advantage of the marketplace by combining multiple lower-quality models sold at lower prices into a higher-quality model that has a designated higher price [54, 57]. In addition, each buyer or buyer group should be assigned a maximal accumulative ϵ over a private dataset based on what they pay or their privilege. This is widely adopted in many systems such as Google BigQuery [18], Snowflake [74], and DProvDB [91]. Usually, a model needs to be stored after being sold for customer service and auditing, incurring operational costs.

• MLaaS platforms [44, 61, 86] have similar problems, serving many models trained with various ϵ values [23, 65, 84], while each customer's total privacy cost is limited by her/his privacy budget. In addition, model serving is usually subject to stringent latency requirements defined in service-level objectives (SLOs) [20]. When the models to be served exceed available memory, heavy I/O operations are required to swap models between disk and memory, making it even more challenging to meet the latency SLOs.

• Multi-tasking applications on edge devices [52, 92] deploy multiple DP-protected models (purchased from model marketplaces or trained using ML-as-a-service) in a resource-constrained environment, facing difficulties in balancing response latency and resource (memory/battery) limitations.

Model deduplication is promising to resolve these problems by sharing similar weight blocks across multiple models to dramatically reduce memory footprint and I/O operations, and thus shorten the latency for serving multiple models in resource-constrained environments [52], multi-tenant environments [21], and database systems that support inference queries [93] and model diagnosis queries [80]. In addition, as observed by prior works [52] and illustrated in Fig. 1, the block-wise disparity score [52] is small for models trained with different ϵ using DP-SGD, which also suggests that model deduplication is promising for addressing the target problems. Unlike *single-model* compression techniques such as quantization [34], pruning [7], and knowledge distillation [35], deduplication is a *multi-model* technique and can work together with single-model compression techniques by deduplicating multiple compressed models or compressing deduplicated models [94].

In a block-based model deduplication scenario, as illustrated in Fig. 2, the models' weights are partitioned into uniform-sized blocks, e.g., 20 blocks in Fig. 2(1). Blocks from a base model (i.e., M_1) are identified to replace blocks in other models (i.e., M_2, M_3, and M_4). As a result, only 8 distinct blocks are needed to reconstruct the models (without significantly degrading models' utilities), which alleviated the broker's operational cost challenges (Optimization1(O1)) in Fig. 2(2). However, none of the existing model deduplication works considered the privacy of the models, which poses significant new issues. If any of a model's blocks are replaced (i.e., deduplicated) by blocks from other models trained on the same dataset, the overall privacy budget incurred on the training dataset remains unchanged due to the post-processing rule (Sec. 2), which means the total privacy loss on the training datasets will not change, and thus model deduplication will not affect data owners. However, the privacy loss (ϵ) of a deduplicated model may increase (e.g., ϵ_2 increases to ϵ_2' in Fig. 2(2)) due to the composition of DP (Sec. 2), which should be bounded based on its

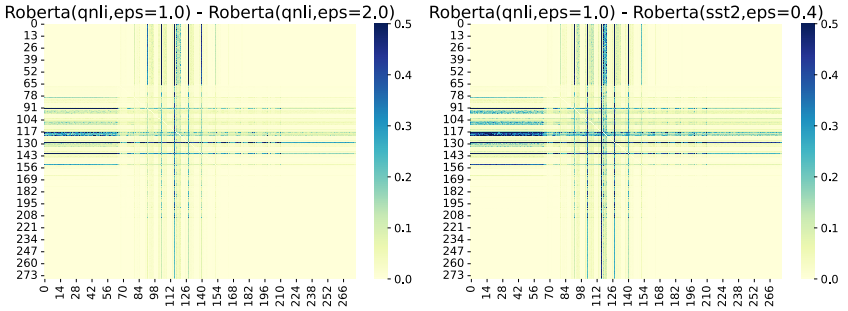


Fig. 1. Weight disparity between DP-SGD-trained models. Left: RoBERTa finetuned on QNLI, with $\epsilon = 1.0$ vs. $\epsilon = 2.0$; right: RoBERTa finetuned on QNLI with $\epsilon = 1.0$ vs fine-tuned on SST2 with $\epsilon = 0.4$). The x- and y-axis represent blocks of two models respectively. Each point represents the disparity score [52] between these blocks. The lower the disparity score, the more similar the two blocks.

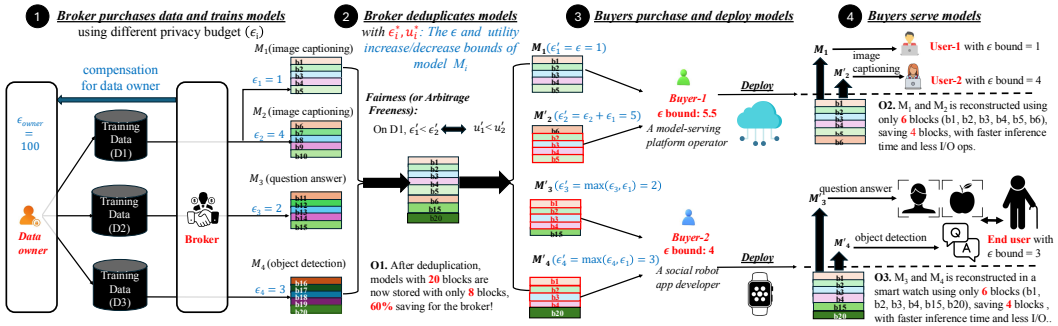


Fig. 2. Model Deduplication Example. Each model M_i is trained with ϵ_i and the corresponding deduplicated model M_i' has an ϵ_i' privacy cost. M_1 (base model) trained on D1 provides blocks for replacing the similar blocks in M_i with privacy and utility bound (e.g., $\epsilon_i' < \epsilon_i + \epsilon_i^*$). ϵ_2' follows sequential composition since M_2 is also trained on D1, while ϵ_3' and ϵ_4' follow parallel composition (see Sec. 2). Each dataset (D1, D2, or D3) is a disjoint partition of a logical data collection, for which privacy loss should be assessed collectively.

target user's privilege and trust-level on the dataset [57, 91], for the following reasons: (1) The private information leakage to a specific user may increase, which should be bounded by the user's authorized overall ϵ. For example, the increased privacy cost of M_2 (ϵ_2') causes more information leakage to Buyer-1 in Fig. 2③, while Buyer-1 has an overall ϵ limit of 5.5. It will further affect User-2 in Fig. 2④, the end user of M_2, with an ϵ limit of 4. (2) The deduplication may affect both the model ϵ and accuracy, which should satisfy *the (arbitrage-free) fairness rule* that a model trained on the same dataset having a higher accuracy must have a higher ϵ [57]. We further consider **minimizing** the ϵ increase of a model caused by deduplication to prevent the broker from selling high ϵ models to buyers.

In light of this scenario, we define a **novel deduplication problem**: to optimize the storage and privacy costs of all DP-trained models while meeting the pre-defined accuracy and privacy constraint of each model, e.g., in Fig. 2, the privacy cost increase $\epsilon_i' - \epsilon_i$ of model M_i is bounded by ϵ_i^*. While the traditional model deduplication problem focuses on addressing the large search space, our identified problem poses new research challenges as follows:

Challenge 1. Privacy Constraint. Existing approaches allow blocks from a target model to be replaced by blocks from any other models called base models, causing the composed privacy loss (Sec. 4.1) of the deduplicated model unbounded. Our target problem requires a careful selection of

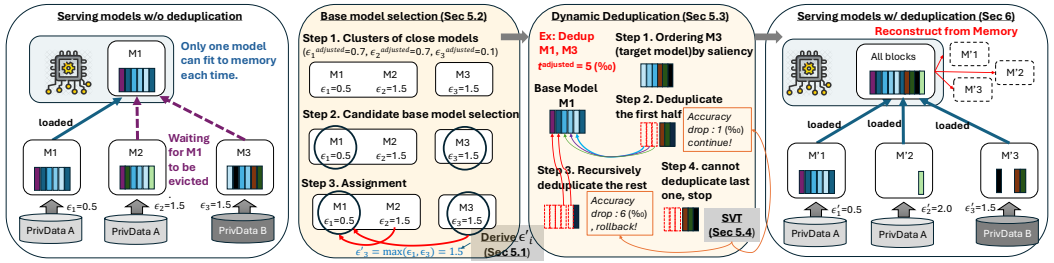


Fig. 3. System Overview. In the second box (B2), M1 and M2 are partitioned to the same group because they are trained on the same dataset, while M3 is in a separate group (B2-Step 1). Then, it selects M1 to be the base model, since there is no qualified base model for $M1$ (B2-Step 2). Then, M2 is assigned to M1, while M3, the unused base model in the other group, is also assigned to M1 (B2-Step 3). The next box (B3) shows how M3 deduplicates with its assigned base model M1. M3's blocks are first ordered by saliency ascendingly (B3-Step 1). Then it first deduplicates the left half of the blocks by replacing each block using the most similar block from M1, followed by an accuracy validation. If the accuracy drop is within 0.005, it recursively deduplicates the right half (B3-Step 2). Otherwise (B3-Step 3), it rolls back the previous step (B3-Step 4), splits the current group into two, and recursively deduplicates the left half. The recursion stops when the number of blocks < 2.

base models to minimize overall storage and privacy costs while meeting privacy and accuracy constraints. Base model selection is a challenging optimization problem, since it is hard to estimate the compression ratio of deduplicating a pair of base and target models under an accuracy constraint.

Challenge 2. Time and Privacy Costs of Accuracy Validation. Existing accuracy-aware model deduplication requires frequent assessment of the accuracy of each modified model in downstream tasks, to ensure that any accuracy decline remains within acceptable limits. This validation process presents a trade-off: frequent accuracy checks will delay the processing and escalate privacy costs for private validation datasets, while infrequent validations lead to more deduplication failures and hurt the compression ratio.

Novel Privacy and Accuracy-aware Model Deduplication (Fig. 3). To address Challenge 1, we quantify different types of privacy cost composition caused by deduplication, and formalize the base model selection problem. We develop a greedy strategy that first clusters models based on the similarity of model metadata, e.g., model architecture and training dataset identifier, and then select base models with high quality (measured by privacy loss and storage benefits) to deduplicate the rest of the models in their cluster. (See Sec. 4.2.)

To address Challenge 2, when deduplicating a pair of base and target models, we propose a dynamic validation strategy coupled with a gradient-based saliency analysis approach to reduce the required number of accuracy validations. At each step, a dynamic number of blocks are deduplicated to gradually separate non-deduplicable blocks from deduplicable blocks. Then, we leverage the Sparse Vector Technique (SVT) [27] to reduce the privacy loss when private validation data is used. This approach transforms the validation steps into a series of boolean questions about whether accuracy drops exceed a threshold, maintaining a fixed privacy budget until the number of negative answers reaches a predetermined cut-off value. (See Sec. 4.3 and 4.4.)

The key contributions of our work include:

(1) We are the first to formalize the problem of model deduplication with privacy constraints, and the privacy loss composition of deduplicated models.
(2) We are also the first to design and develop an end-to-end pipeline for model deduplication to optimize and balance privacy, accuracy, and compression ratio, including base model selection, dynamic block deduplication, and SVT-based accuracy validation.

(3) We implement the proposed system and conduct comprehensive evaluations. The results show that, compared to SOTA model deduplication methods (with our base model selection to ensure privacy), our approach can improve the compression ratio for deduplicating multiple large language models, vision transformers, and ResNet models, by up to 35× for individual models with privacy costs reduced by up to 3× compared to alternative base model selection designs. The multi-model inference speed is also accelerated by 43× by avoiding swapping models between memory and disk.

2 Preliminaries

Differential Privacy [27, 65] (DP) is a mathematical framework for quantifying privacy risks in data analysis, and is increasingly adopted in real-world applications by companies like Google [37] and Apple [69].

Definition 2.1 (Differential Privacy [27])**.** A randomized algorithm $\mathcal{M}$ is (ϵ,δ)-differentially private if for all neighboring datasets (i.e. differ by exactly one element) $\mathcal{D}$ and $\mathcal{D}'$, and for $\forall \mathcal{S} \subseteq \text{Range}(\mathcal{M})$, $Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{S}] \leq e^{\epsilon} * Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{S}] + \delta$. ϵ, also called **privacy budget**, is a metric of privacy loss at a differential change in data. δ denotes the probability of the privacy guarantee being failed.

DP is characterized by three key properties. *Sequential Composition* illustrates how privacy costs accumulate when multiple analyses are performed on the same dataset. When analyses are conducted on disjoint subsets of data, *Parallel Composition* [65] considers the maximum privacy cost on each dataset as the privacy cost. *Post-processing* ensures that any data-independent processing of differentially private outputs will not incur additional privacy costs. The formal statements of these properties are:

Theorem 2.1 (Sequential Composition [27]). *If an algorithm $\mathcal{M}_1$ is (ϵ_1, δ_1)-DP and $\mathcal{M}_2$ is (ϵ_2, δ_2)-DP, then their sequential composition $\mathcal{M}(x) = (\mathcal{M}_1(x), \mathcal{M}_2(x))$ is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP.*

Theorem 2.2 (Parallel Composition[65]). *Let $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$ be disjoint subsets of the database $\mathcal{D}$. If algorithm $\mathcal{M}_i$ is (ϵ, δ)-DP for each $i \in [1, k]$, then the parallel composition $M(x) = (\mathcal{M}_1(\mathcal{D}_1), \ldots, \mathcal{M}_k(\mathcal{D}_k))$ is $(\max_i \epsilon_i, \max_i \delta_i)$-DP.*

Theorem 2.3 (Post-processing [27]). *If $\mathcal{M}$ is (ϵ, δ)-DP and f is any data independent function, then $\mathrm{f} \circ \mathcal{M}$ is also (ϵ, δ)-DP.*

The post-processing property ensures that the privacy budget of a model will not increase with the number of inferences. In addition, given models trained on a dataset D, deduplicating them will not affect the overall privacy budget on D.

In the context of DP, sensitivity measures the maximum change in the output of a function when a single data point in the input dataset is modified. Formally, we have the following definition.

Definition 2.2 (Sensitivity[27])**.** For a given function f, the sensitivity Δf is defined as $\Delta f = \max_{\mathcal{D},\mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|$ where $\mathcal{D}$ and $\mathcal{D}'$ are any two neighboring datasets.

Sparse Vector Technique (SVT) [27] is a method in DP for answering a large number of queries while only incurring the privacy cost for those queries that exceed a certain threshold. This is achieved by adding noise to both the threshold and each query output. Let cut-off c denote the maximum number of queries to be answered with result exceeding threshold before halting, Δ represent the sensitivity, and ϵ signify the total privacy budget. The privacy budget is divided into two parts: ϵ_1 for the threshold and ϵ_2 for query results, such that $\epsilon_1 + \epsilon_2 = \epsilon$. The algorithm proceeds as follows: noise is sampled from a distribution $\text{Lap}(\frac{\Delta}{\epsilon_1})$ and added to a predetermined

threshold. For each query, noise is sampled from a distribution $\text{Lap}(\frac{2c\Delta}{\epsilon_2})$ and added to the query result. If the noisy query result is less than the noisy threshold, it returns a negative answer and the procedure continues; otherwise, it returns a positive answer and increse the counter. The algorithm halts when the counter reaches c. A recommended ratio between the two epsilon values is $\epsilon_1 : \epsilon_2 = 1 : (2c)^{2/3}$ [62].

Differentially Private Stochastic Gradient Decent (DP-SGD) [1]. In DP-SGD, privacy is maintained through two primary modifications to the traditional SGD algorithm: gradient clipping and the addition of noise. Mathematically, the update rule for the model parameters θ at step t in DP-SGD can be expressed as:

$$\theta_{t+1} \leftarrow \theta_t - \eta \left(\frac{1}{B} \sum_{i=1}^{B} \text{clip}(g_i(\theta_t), C) + \mathcal{N}(0, \sigma^2 I) \right)$$

where η is the learning rate, B is the batch size, $g_i(\theta_t)$ is the gradient of the loss function for the i-th data point in the mini-batch, $\text{clip}(g_i(\theta_t), C)$ denotes the gradient clipping operation, and $\mathcal{N}(0, \sigma^2 I)$ represents the Gaussian noise with variance σ^2.

Note in this paper, for simplicity, we focus on ϵ, which dominates the privacy budget. However, all our analysis can be easily extended to δ by applying the same constraints and computations on δ as on ϵ since they follow the same composition rules (Theorem 2.1, 2.2, 2.3).

3 Problem Analysis

Here, we will introduce a motivating example, the problems with the SOTA deduplication procedure, and an overview of our solution.

3.1 Motivating Example

As shown in Fig. 2, a broker of a model marketplace needs to manage models M_1, M_2, and M_3, M_4 for Buyer-1 and Buyer-2. S/he configures ϵ_i^* (privacy loss increase bound) and u_i^* (utility drop bound) for M_i (e.g., leveraging a privacy provenance system [91]), and deduplicates these models to minimize storage costs and privacy costs (within ϵ_i^* and u_i^* bound). As a result, the number of blocks is reduced from 20 to 8 (**O1** in Fig. 2❷) so that more models can be cached in memory, reducing brokers' operational costs.

Buyer-1 represents a startup MLaaS company, which has an overall ϵ bound of 5.5. In Fig. 2❸, s/he purchases deduplicated models M_1' and M_2' with a total ϵ of 5, and deploys them in a public cloud for serving two users (e.g., User1 and User2 with ϵ bounds of 1 and 4 respectively). MLaaS operators may strive to meet the users' model serving latency requirements as specified in SLOs. Purchasing deduplicated models will reduce the memory footprint for serving both models from 10 blocks to 6 blocks, leading to a significant reduction in model serving latency (**O2** in Fig. 2❹).

Buyer-2 represents an edge application developer, who has an overall ϵ bound of 4. In Fig. 2❸, s/he purchases deduplicated models M_3 and M_4 for deployment on a smart watch for serving a social robot [52] that performs both object recognition task and question answer task at the same time. Assuming User B's edge device can only hold nine blocks, deduplicating models will reduce the number of required blocks from 10 to 6, while maintaining compliance with privacy bounds. It means the deduplication at the broker's side (without significantly impacting the utility) enables the serving of both models in memory, eliminates I/O operations fetching blocks from the disk, and significantly reduces the application response latency (**O3** in Fig. 2❹). After this purchase, Buyer-2 still has a remaining ϵ bound of 1 (i.e., $4 - max(\epsilon_2', \epsilon_3') = 1$).

3.2 Existing Model Deduplication Procedure

The state-of-the-art (SOTA) accuracy-aware model deduplication technique [93], termed **Dedup**, assumes that models arrive in order and the first model will not be deduplicated. Starting from the second model, it will repeat the following steps:

Step 1. To deduplicate a target model M_t, use all previously arrived models to serve as potential block providers, called base models.

Step 2. Order all blocks from M_t by some saliency measures (e.g., the third quantile of weight values in the block [93]) ascendingly.

Step 3. For each block $b \in M_t$ in the ordered list, run the following sub-steps: (3-1) Select a block b' that is most similar to b from the collection of blocks in all base models (e.g., using locality sensitive hashing [93]). (3-2) Substitute b with b', resulting in a modified model M'_t. (3-3) Evaluate the accuracy of the modified target model using a validation dataset. (This step could be performed once every N iterations, naively reducing the validation frequency.) (3-4) If the accuracy drop exceeds a threshold, undo the block replacement and stop early, otherwise (3-5) Move to the next block $b \in M_t$.

Limitations of the SOTA method for our targeting problem include:

(L1) No existing works have formalized privacy budget derivation and quantification for the deduplicated models.

(L2) Step 1 naively takes all available models as base models, without checking whether combining the target model with blocks from all base models will cause a privacy budget increase that violates the privacy budget bound of the target model.

(L3) Step 2 uses simple saliency measures, which we found less effective for models with noisy parameters trained with DP.

(L4) Step 3 adopts a static strategy to validate the accuracy of a deduplicated model. (As mentioned, the expensive accuracy validations compose a major bottleneck of the deduplication process.)

(L5) If the validation dataset used in Step 3-3 is private, the accuracy comparison in Step 3-4 will introduce additional privacy costs not considered by existing approaches.

3.3 Methodology Overview

As illustrated in Fig. 3, an overview of our work is as follows.

Private Budget Derivation (Sec. 4.1). To address L1, we are the first to quantify the privacy loss of a target model M_t deduplicated by a base model $\hat{M}$, while M_t and $\hat{M}$ could be trained on the same, disjoint, and even overlapping datasets.

Base Model Selection (Sec. 4.2). To address L2, we first formalize the base model selection as a combined set partitioning and generalized assignment problem, which partitions the models into base models and target models and assigns base models to target models. However, there is no easy way to efficiently estimate the compression ratio for deduplicating each pair of models without incurring additional privacy costs. Therefore, we developed a greedy algorithm to prioritize high-quality models that have lower privacy costs and higher storage benefits to serve as the base models.

Deduplication with Saliency Analysis and Dynamic Validation (Sec. 4.3). L3 and L4 are correlated. We compared several existing weight saliency measurements [30, 53, 58, 77] and found the gradient magnitude [30] well balances saliency profiling latency and effectiveness. We also propose our dynamic deduplication scheme where dynamic ranges of non-salient blocks are grouped and deduplicated together to avoid accuracy validation failures.

Sparse Vector Technique (SVT) for Validation Using Private Data (Sec. 4.4). To address L5, we abstract accuracy validation as a boolean question about whether the accuracy drop is above a

threshold and apply SVT to provide a fixed budget for a pre-specified number of failed validations (i.e., the boolean questions return negative results).

4 Novel Automatic Deduplication

In this section, we will present our solutions in detail.

4.1 Privacy Budget Derivation

The privacy budget of M'_t in the above deduplication process can be determined by Theorem 4.1, of which the intuition is as follows. Consider the case where two similar blocks b_t and $\hat{b}$ are from model M_t with ϵ_t and $\hat{M}$ with $\hat{\epsilon}$ that are trained on dataset D_t and $\hat{D}$ respectively. Using $\hat{b}$ to replace b_t will change ϵ_t into ϵ'_t in three situations: (1) **Intra-Model ($M_t = \hat{M}$)**: If b_t and $\hat{b}$ are from the same model, they can be safely deduplicated with $\epsilon'_t = \epsilon_t$. (2) **Inter-Model Intra-Data ($M_t \neq \hat{M} \wedge D_t = \hat{D}$)**: If b_t and $\hat{b}$ are from different models that were trained with the same dataset, we have $\epsilon'_t = \epsilon_t + \hat{\epsilon}$ by applying DP's sequential composition property [26, 65], detailed in Sec. 2. (3) **Inter-Model Inter-Data ($(M_t \neq \hat{M}) \wedge (D_t \cap \hat{D} = \Phi)$)**: If b_t and $\hat{b}$ are from different models that were trained on disjoint datasets, we have $\epsilon'_t = \max(\epsilon_t, \hat{\epsilon})$ by applying DP's parallel composition [26, 65], detailed in Sec. 2.

Theorem 4.1. *Suppose that d disjoint training datasets are used for training all models. Let $\hat{M}_t$ be a (ϵ_t, δ_t)-DP model to be deduplicated, and let $\hat{\mathcal{M}} = \{\hat{M}_1, ..., \hat{M}_k\}$ be a set of k distinct differentially private models, each satisfying $(\hat{\epsilon}_i, \hat{\delta}_i)$-DP, serving as potential block providers. Any two models in $\{M_t\} \cup \hat{\mathcal{M}}$ are trained on either the same dataset or completely disjoint datasets from each other. Let $\hat{\mathcal{M}}_j = \{\hat{M}_{j_1}, ...\hat{M}_{j_l}\}$ be the group of models trained on the dataset D_j. Then, on D_j, the resulting deduplicated model, denoted as M'_t, satisfies $(\epsilon_t \cdot \mathbb{1}_{M_t, D_j} + \sum_{i=1}^{l} \hat{\epsilon_{j_i}}, \quad \delta_t \cdot \mathbb{1}_{M_t, D_j} + \sum_{i=1}^{l} \hat{\delta_{j_i}})$-DP, where the indicator function $\mathbb{1}_{m, D_j}$ is equal to 1 if model m is trained on D_j, otherwise 0. If $\hat{\mathcal{M}} = \emptyset$, then M'_t satisfies (ϵ_t, δ_t)-DP.*

Given any disjoint D_j $(1 \leq j \leq d)$, if the derived model M'_t satisfies (ϵ^j, δ^j)-DP, the derived model M'_t satisfies $(\max_j \epsilon^j, \max_j \delta^j)$-DP on the union of datasets $\cup_{j=1}^{d} D_j$.

Proof. Because any two models are trained on either the same dataset or the datasets completely disjoint from one another, we can divide the models in $\hat{\mathcal{M}}$ into groups by their training datasets. $\hat{\mathcal{M}}_j = \{\hat{M}_{j_1}, ...\hat{M}_{j_l}\}$ is a group of models using training data D_j. By DP composability, the collection of blocks from this group of models satisfies $(\sum_{i=1}^{l} \hat{\epsilon_{j_i}}, \sum_{i=1}^{l} \hat{\delta_{j_i}})$-DP. Because the deduplication algorithm iterates over all blocks to decide replacements, by DP post-processing property, the derived model M'_t satisfies $(\sum_{i=1}^{l} \hat{\epsilon_{j_i}}, \sum_{i=1}^{l} \hat{\delta_{j_i}})$-DP on D_j when the original model M_t is not trained on D_j. If M_t is trained on D_j, M'_t satisfies $(\epsilon_t + \sum_{i=1}^{l} \hat{\epsilon_{j_i}}, \delta_t + \sum_{i=1}^{l} \hat{\delta_{j_i}})$-DP by DP sequential composition. Given any disjoint dataset D_j $(1 \leq j \leq d)$, if the deduplicated model M'_t satisfies (ϵ^j, δ^j)-DP, it satisfies $(\max_j \epsilon^j, \max_j \delta^j)$-DP on the union of disjoint datasets $\cup_{j=1}^{d} D_j$, by the parallel composition property [65].

Generalization. We can easily remove the disjoint dataset assumption by abstracting the problem as a graph. Assume that for graph G, each node represents a base model, and two nodes $\hat{M}_p$ and $\hat{M}_q$ have an edge if and only if their training datasets overlap. Regarding each connected component C_j of k nodes as a model group $\{\hat{M}_{j_1}, ...\hat{M}_{j_k}\}$, similarly, the collection of blocks from it satisfies $\epsilon^j = \sum_{i=1}^{k} \epsilon_{j_i}$ -DP. Suppose G has d connected components, then the derived model M' satisfies $(\max_{1 \leq j \leq d} \epsilon^j)$-DP.

4.2 Base Model Selection

4.2.1 Problem Analysis and Formalization. A base model is a block provider, e.g., M_1 in Fig. 2.

According to the privacy analysis in Sec. 4.1, the more base models used, the more the privacy loss (ϵ) of the target model increases, which is the **key** to optimize the privacy loss of the target model. The research question is how to select base models and assign target models to base models to minimize storage and privacy costs while meeting the accuracy and privacy constraints. We start from two assumptions: (Assumption-1) each target model can only use one base model for deduplication, which practically reduces the privacy loss of the resulting models and the deduplication overheads. (We allow many target models to share one base model to maximize their utilization) and (Assumption-2) We do not allow a model $\hat{M}$ to be assigned as a base model and a target model at the same time, because deduplicating $\hat{M}$ as a target model may cause parameter and accuracy changes in all models that have been deduplicated using $\hat{M}$ as a base model.

Problem Definition. Given a set of models trained on the same dataset or disjoint datasets, $\mathcal{M} = \{M_1, ..., M_n\}$, satisfying $\epsilon_1, \ldots, \epsilon_n$-DP respectively. These models have utilities (accuracy) $u_1, \ldots, u_n$ respectively. Each model is split into (tensor) blocks of equal size. Each model $M_j (j = 1, \ldots, n)$ has an ϵ increase threshold ϵ^* and a utility drop threshold u_j^*. The deduplication process must satisfy (1) M_j's ϵ increase should be bounded by ϵ^*, (2) M_j's utility drop should be bounded by u_j^*, and (3) the *fairness rule* ensuring models trained with higher ϵ on the same dataset must maintain higher accuracy than those trained with lower ϵ.

We suppose n_{ij} in an $n \times n$ matrix $\mathcal{N} = \{n_{ij}\}$ represents the number of blocks from M_j (as a target model) that can be replaced by blocks from M_i (as a base model) with M_j's utility drop bounded by u_j^*. $\mathcal{A} = \{a_{ij}\}$ represents the partitioning and assignment matrix that partitions models into target models and base models and assigns target models to base models. $a_{ij} = 1$ means M_j is assigned to the base model M_i. The objective function and constraints are formalized below, where $f_\epsilon(M_i, M_j)$ denotes the privacy budget (ϵ_i') of the target model M_j using M_i as a base model based on the derivation in Sec. 4.1. λ is a regularization factor to balance the two objectives (1) To maximize the total number of blocks that can be deduplicated (replaced) ($\sum_{i=1}^{n}\sum_{j=1}^{n}(n_{ij})$) and (2) to minimize the target models' overall ϵ increase caused by deduplication ($\sum_{i=1}^{n}\sum_{j=1}^{n}(f_\epsilon(M_i, M_j))$). The optimization process must satisfy the adjusted privacy constraint ϵ_j^{adjust} (Eq. 2) explained later, the Assumption-1 (Eq. 3), and Assumption-2 (Eq. 4).

$$\max_{a_{ij=0,1}(1\le i,j\le n)} \sum_{i=1}^{n}\sum_{j=1}^{n}(n_{ij} - \lambda \times f_\epsilon(M_i, M_j)) \times a_{ij}, \quad s.t. \tag{1}$$

$$f_\epsilon(M_i, M_j) - \epsilon_j < \epsilon_j^{adjusted} \qquad \text{if} \quad a_{ij} = 1 \tag{2}$$

$$\sum_{i=1}^{n} a_{ij} = 1 \tag{3}$$

$$\forall M_i, \nexists M_k, M_j (i \neq k \neq j) \quad \text{satisfying} \quad a_{ik} = 1 \wedge a_{ji} = 1 \tag{4}$$

Adjusted utility decrease threshold u_i^{adjust} and privacy cost increase threshold ϵ_j^{adjust}. To meet the fairness rule, we enforce that the orderings of these models by the ϵ and by $utility$ do not change after deduplication. Then, if all models satisfy the fairness rule before deduplication, these models still adhere to the fairness rule after deduplication. Suppose M_i and M_{i-1} are two consecutive models in the sequence of k models trained on the same dataset ordered ascendingly by utility ($u_i > u_{i-1}$). If the user-specified utility ranges of the deduplicated models M_i' and M_{i-1}' are $[u_i - u_i^*, u_i]$

and $[u_{i-1} - u^*_{i-1}, u_{i-1}]$ and they overlap, then $u'_i < u'_{i-1}$ is possible, which violates the fairness order by utilities. To eliminate such overlapping ranges, we adjust u^*_i to be $u_i^{adjusted} = min(u^*_i, u_i - u_{i-1})$ for $i = 2, \ldots, k$ so that the deduplication algorithm always ensures $u'_i > u'_{i-1}$. Similarly, we set ϵ^*_i to be $\epsilon_i^{adjusted} = min(\epsilon^*_i, \epsilon_i - \epsilon_{i-1})$ for $i = 2, \ldots, k$.

Analysis. If we fix a subset of models $\mathcal{B} \subset \mathcal{M}$ to be base models, the problem is a generalized assignment problem (GAP) [19], where one or more target models (i.e., tasks in the classical GAP problem) are assigned to each base model (i.e., agents in the classical GAP). The GAP problem was proven to be NP-hard [13, 19]. However, instead of having a fixed $\mathcal{B}$, our problem additionally requires searching for a set-partitioning scheme that divides $\mathcal{M}$ into $\mathcal{B}$ and $\mathcal{M} - \mathcal{B}$.

Moreover, the problem is a black-box optimization problem [24], since it is expensive to estimate n_{ij}. Performing actual deduplication to determine n_{ij} is generally expensive. Using weight-level similarity to estimate n_{ij} is also impractical since the similarity function will compose ϵ of input models. Therefore, we developed a greedy strategy to address the problem.

4.2.2 A Greedy Strategy. The main intuition is that the quality of the base models is more important than the quantity. A high-quality base model should satisfy the following requirements: (**R1**) **Similar to target models.** Using a base model with greater similarity in model architecture, training dataset, utility, and privacy budget to the target model usually leads to a better compression ratio (with the same utility constraint). (**R2**) **Low privacy budget.** Usually, a base model with a lower privacy loss leads to a smaller increase in privacy cost in the deduplicated model. (**R3**) **Low chance to be deduplicated (Low opportunity cost)**. A model having fewer qualified base models than qualified target models is more suitable to serve as a base model than a target model. Based on the intuition, our key idea is to cluster models by similarity of model metadata, and in each cluster, we select the models satisfying R1, R2, and R3, to serve as base models for the rest of the cluster:

(1) **Partitioning Models Based on Similarity.** We first cluster the models from $\mathcal{M}$ based on public metadata, such as model architecture, public training dataset, the anonymous identifier of the private fine-tuning dataset. We used a simple hierarchical clustering strategy and ensured that all models in the same cluster have the same model architecture and similar datasets.
(2) **Assigning Dangling Models as Base Models.** We define dangling models to be models that do not have any qualified base models in their clusters (i.e., M_j cannot find any M_i satisfying Eq. 2). There must be at least one dangling model in each cluster–the model with the smallest ϵ. We only allow these dangling models to serve as the base models to deduplicate other models in the cluster since they are similar to other models (i.e., can deduplicate a good number of blocks from other models) and zero blocks can be deduplicated from them if assigned as target models, thus maximizing the storage benefits in its local cluster. The process is described in Alg. 1.
(3) **Intra-Group and Inter-Group Base Model Selection.** For each non-dangling model, it must be able to find at least one qualified base model in its cluster. If it has two or more qualified base models in its cluster, selecting any of them may lead to similar compression ratios according to our observation. Therefore, we will select the one that brings minimum privacy cost increase as its base model (line 6-9 in Alg. 3). Then, for those dangling models that are never used as base models (i.e., $count == 0$), we search for qualified base models from other clusters to deduplicate them (line 10-15 in Alg. 3).

4.3 Deduplication Algorithms

4.3.1 Deduplication Problem Reduction. Once we select the base model for each target model, the privacy budget increase becomes a constant and the privacy and fairness constraints are ensured. Then, the model deduplication problem is reduced to a set of two-model deduplication problems.

Algorithm 1 Identify dangling models in a group $\mathcal{G} = \{M_1, ..., M_k\}$

procedure CandidateBaseModelGen($\mathcal{G}$)
  $\mathcal{B} \leftarrow \{\}$; $\mathcal{T} \leftarrow \mathcal{G}$
  for M_i in $\mathcal{T}$ **do** ▷ Identify dangling models and add them to $\mathcal{B}$
    if $\epsilon_j \geq \epsilon_i^{adjusted}$ for all $j \in \{1, ..., k\}$ and $j \neq i$ **then**
      $\mathcal{B} = \mathcal{B} \cup \{M_i\}$; $\mathcal{T} = \mathcal{T} - \{M_i\}$;
  return

Algorithm 2 Select a base model $\hat{M}$ for a target model M_t in a Group $\mathcal{G}$

procedure BaseModelSelectStep(M_t, $\mathcal{G}$)
  $\mathcal{B} \leftarrow$ The set of candidate base models in $\mathcal{G}$;
  Find base model $\hat{M} \in \mathcal{B}$ with the smallest $\hat{\epsilon}$ to satisfy $f_\epsilon(\hat{M}, M_t) - \epsilon_t < \epsilon_t^{adjusted}$ to serve as the base model for M_t; $\hat{M}.count++$
  return $\hat{M}$

Algorithm 3 Base model selection for all models in $\mathcal{M}$

procedure BaseModelSelectMain($\mathcal{M}$)
  $C = \{\mathcal{G}_1, ..., \mathcal{G}_l\} \leftarrow Cluster(\mathcal{M})$
  for $\mathcal{G}_i \in C$ **do** $CandidateBaseModelGen(\mathcal{G}_i)$
  for $\mathcal{G}_i \in C$ **do**
    sort ($C' = C - \mathcal{G}_i$) based on similarity to $\mathcal{G}_i$
    for $M_j \in \mathcal{T}_i$ **do**
      $ret \leftarrow BaseModelSelectStep(M_j, \mathcal{G}_i)$
      record that M_j's base model is ret
    for $M_k \in \mathcal{B}_i$ **do**
      if $M_k.count == 0$ **then**
        while $ret = NULL \wedge next\_cluster(C') \neq NULL$ **do**
          $\mathcal{G}_k \leftarrow next\_cluster(C')$
          $ret \leftarrow BaseModelSelectStep(M_k, \mathcal{G}_k)$
        record that M_k's base model is ret
  return

Each problem involves a target model M_j and a base model M_i. The optimization objective is to maximize $n_{ij} = |M_i \cap M'_j|$, i.e., the number of unique blocks in the deduplicated model M'_j that are from M_i (i.e., minimize the compression ratio) while meeting the utility drop constraint formalized below.

$$\max n_{ij} \text{ with}$$
$$M'_j = \text{deduplicate}(M_j; M_i \text{ as base model}) \; s.t.$$
$$u_j - u'_j < u_j^{adjusted}$$

4.3.2 Problem Analysis. Given any block from M_j, it can be replaced by any block from M_i (or no replacement), leading to $(|M_i| + 1)^{|M_j|}$ possible deduplication plans. Depending on the block size, deep learning models in our experiments may have hundreds to thousands of blocks. Therefore, it is computationally inhibiting to exhaustively search and evaluate the storage costs and accuracy drop of all deduplication plans. Following the existing deduplication framework presented in Sec. 3.2 to perform two-model deduplication, it performs a validation after deduplicating every N blocks. When N is set to a small number, the frequent utility validation operations will bottleneck the deduplication process. However, if we set N to a large number, it often deduplicates too many blocks to keep the utility drop within the constraint, due to the lack of sophisticated saliency analysis to order the blocks. A poor saliency measurement may mix a few salient blocks, which will significantly impact the model's utility if deduplicated, with many non-salient blocks in one batch. Then, the salient blocks will cause a validation failure, which will further cause *all* deduplications in the batch, including the deduplications of non-salient blocks, to be rolled back. This will lead to

Algorithm 4 Dynamic-Range Deduplication (DRD)

```
1: Input: Target and base models represented by a list of block identifiers B, B̂; utility drop threshold T; minimum #blocks in a batch L.
2: procedure Deduplication(B, B̂, T)
3:     Initialize l = 0; r = |B| − 1;
4:     Recursion(B, B̂, l, r, T);
5:     return ;
6: procedure Recursion(B, B̂, l, r, T)
7:     if r − l < L then                                          ▷ Base case, skip a small batch
8:         return ;
9:     m = (r + l)//2;
10:    for block j in l, l + 1, . . . , m do                      ▷ Try deduplicate the left half
11:        Replace block B[j] by the most similar block in B ∪ B̂;
12:    Evaluate model on validation set, compute utility drop Δu;
13:    if Δu >= T then                                            ▷ If validation fails
14:        Rollback the deduplication of block l, l + 1, . . . , m;
15:        if l < r then
16:            Recursion(B, B̂, l, m, T);                          ▷ Recursively deduplicate the range
17:    Recursion(B, B̂, m + 1, r, T);                              ▷ Recursively deduplicate the right half
18:    return ;
```

missed opportunities and wasting of resources. However, saliency analysis is a challenging task in nature and made even more difficult by the noises introduced by DP [70].

4.3.3 Our Dynamic Deduplication Algorithms. We resolve the problem in two steps: (1) We carefully examine and compare the block saliency measurements for our target problem. (2) We design new algorithms that dynamically adjust the size of each group to balance the frequency of validations and the number of rollbacks (validation failures). For example, at the beginning, a large group of the least salient blocks should be deduplicated together. As we progress through the ordered list, the group size gradually decreases, ensuring that non-salient blocks are checked in a fine-grained manner.

Block Saliency Measures. Weights that significantly impact the prediction are called "salient weights" [43, 77]. There are many ways to measure the saliency of weights for model explanability [73], compression [29, 30, 51, 53]. The most popular ones include the weight magnitude [93], activation magnitude [55], and the Fisher information that involves the square of gradients [58]. We find that the Fisher information metric is the most accurate among them. However, it is also the most computationally intensive. In addition, the squared gradients are usually very small and may introduces unstable numerical results. Therefore, we adopted the absolute value of gradient (called gradient magnitude) as our saliency measure. The saliency of a block is defined as the mean of the gradient magnitude of each weight within a block, which is obtained by one forward and backward pass of a dataset. This dataset could be a public dataset or a disjoint partition from the private training/validation dataset. In the latter case, we added noise to the gradient, and this one-time privacy cost is usually smaller than the model's ϵ and thus gets absorbed due to DP's parallel composition.

Group Deduplication Algorithms. Our idea is to dynamically determine the group size and re-evaluate some blocks in a group that fail to be deduplicated. We designed a recursive dynamic-range deduplication (DRD) algorithm, as formalized in Alg. 4. DRD begins by dividing the input of blocks (i.e., input range) into two halves. It then attempts to deduplicate the left half. If this deduplication fails (i.e., the utility drop exceeds the bound), the algorithm recursively runs on the left half (Line 14) so that at every recursion level, the number of blocks that are deduplicated in a batch will be reduced by half. After deduplicating the left half, the algorithm recursively deduplicates the right half (Line 15). The base case for this recursion occurs when the input range has fewer than L blocks, and it skips such a small range because such remaining blocks are likely to be salient blocks. We also proposed a variant of DRD, called Dynamic-Range-Expansion Deduplication (DRED), slightly

Algorithm 5 ϵ-DP Deduplication with SVT

Input: private validation dataset D; model M to be deduplicated; allowed utility drop threshold T; cut-off c; privacy budget ϵ for validation.
procedure SVT(M, D, T, c, ϵ)
  $\epsilon_1 = \frac{1}{1+(2c)^{2/3}}\epsilon$, $\epsilon_2 = \frac{(2c)^{2/3}}{1+(2c)^{2/3}}\epsilon$;
  $\hat{T} = T + \text{Lap}(\frac{2}{|D|\epsilon_1})$; ▷ Add noise to threshold T
  count = 0;
  while true **do**
    Get next set of blocks to be deduplicated $\mathcal{B} = \{b_1, b_2, ..., b_l\}$ proposed by the greedy or dynamic deduplication algorithms.
    if $\mathcal{B} = \emptyset$ **then** Halt;
    M' = Deduplicate($M, \mathcal{B}$) ;
    $\Delta u = f(D, M) - f(D, M')$; ▷ Compute utility drop
    $v_i = \Delta u + \text{Lap}(\frac{4c}{|D|\epsilon_2})$; ▷ Add noise to utility drop
    if $v_i \geq \hat{T}$ **then** ▷ Deduplication fails
      Output M;
      count = count + 1;
      if count $\geq c$ **then** Halt; ▷ Reach cut-off c
    else ▷ Deduplication succeeds
      Output M';
      $M = M'$;

different from DRD in Line 17. DRED will recursively deduplicate the blocks from $m+1$ to $|B+1|$, rather than $m+1$ to r (i.e., changing line 17 to Recursion($B, \hat{B}, m+1, |B|-1, T$)). It allows for more aggressive (coarser-grained) deduplication attempts.

Given a block b in each deduplication batch, a block from the base model most similar to block b is chosen to replace b. To measure similarity, we compared several distance measures, such as `l1-norm`, `l2-norm`, and `cosine` and chose to directly use `l2-norm` (i.e., Euclidean distance). We use pairwise comparison for best accuracy, since it is not the bottleneck of our target scenario (less than 10% of overheads). If needed, it can be easily replaced by a faster but more error-prone locality-sensitive hashing [22, 64] technique.

4.4 SVT for Validation Using Private Data

Suppose that $f(D, M)$ evaluates the accuracy of a model M on a private validation dataset D, which is disjoint from the training set. Given a classification model, its utility f has a sensitivity of $\frac{1}{|D|}$ because the number of correct predictions changes at most by 1 on two neighbor datasets and the accuracy is calculated as the percentage of the correct predictions. Suppose that M' is the new model after a deduplication step on M. Then, the utility drop, i.e., $f(D, M) - f(D, M')$ has sensitivity $\frac{2}{|D|}$. Each deduplication step is followed by a validation accuracy query on D, and the answer is used to determine whether the deduplication should stop. Accordingly, Sparse Vector Technique (SVT) works by applying Laplace noise and comparing the utility drop with a noisy threshold, as shown in Alg. 5. When the validation dataset is large, the privacy budget required for deduplication can be significantly smaller, as the sensitivity decreases inversely with the size of the dataset.

Alg. 5 employs two distinct privacy parameters: ϵ_1 for noise added to the threshold, ϵ_2 for the validation accuracy comparison. We adopt the recommended ratio of $\epsilon_1 : \epsilon_2 = 1 : (2c)^{2/3}$ as proposed by Lyu et al. [62]. Cut-off c represents the maximum number of times the deduplication process can fail the utility test (i.e., validation), which is pre-specified.

4.5 Implementation

A model may contain many parameter tensors, e.g., embedding vectors, weight matrices, or convolutional filters. In our implementation, each parameter tensor is flattened into a one-dimensional array and then partitioned into blocks with fixed size. The last block will be padded with zeros.

Some parameter tensors, such as biases and layer norms, are much smaller than the block size, so they are not partitioned and are managed separately.

A deduplicated model is stored in three parts : (1) A 2-D array that contains the blocks for all models, where each row represents a block flattened into a 1-D array. (2) A dictionary for each model containing tensor shapes and extra weights, such as biases and norms that are much smaller than the block size and are handled separately. (3) A list of block indices for each model. For example, a model with K blocks would have a K-way array, with the i-th element specifying the i-th block's row index in the 2-D array.

Serving a deduplicated model requires reconstructing the model from the blocks and the dictionaries. Whenever a new query requires a model that is not reconstructed, if there is insufficient memory, a victim model will be evicted following the Least Recently Used (LRU) policy to free up memory space for reconstructing the new model. The system then retrieves the model ID, looks up the corresponding model constitution and extra weights, and reconstructs the queried model. The process is accelerated by reusing a reconstructed model with the same architecture and leveraging our array-based indexing to access blocks for replacement.

5 Experiments

We conducted comprehensive evaluations with an ablation study to demonstrate the effectiveness of our proposed methods.

5.1 Experimental Settings

5.1.1 Workloads. To demonstrate the broad applicability of our proposed approaches, we experimented with a diverse range of model architectures and tasks[1]:

(1) Roberta-Base [59] (called Roberta): A Transformer encoder for natural language inference tasks, composed of embedding layers, attention layers, normalization layers, and linear layers. It contains 277 blocks with a block size of 58, 982 floating points.
(2) Vision Transformer-Large (called ViT) [25]: A Transformer model for image classification tasks, consisting of attention layers, normalization layers, and linear layers. It has 288 blocks with a block size of 1, 048, 576 floating points.
(3) ResNet152 [38] (Called ResNet): A convolutional neural network with residual connections for multi-attribute classification tasks. It has 238 blocks with a size of 262, 144 floating points.

These models encompass all major layer types in modern deep learning architectures, ensuring that our proposed algorithms are applicable to a wide variety of models beyond those explicitly tested. The datasets were pre-partitioned into training, validation, and test sets, which are disjoint with each other.

The *choice of privacy budgets* is contingent upon the data and model architecture. Existing works employ varied ranges. For instance, [1.0, 8.0] and (0, 2] for both natural language and image classification tasks [12, 32, 33, 67, 79, 89], [0.08, 4.6] for image and tabular data classification tasks [4], and [1,16] for image processing tasks [45]. In our work, we adopted ranges (0.0,10.0] for the Roberta-base models and (0, 5.5] for ViT and ResNet, respectively.

Our main experiments involve eight scenarios of models shown as A1-A5 and B1-B3 in Tab. 1, designed to be diverse and representative. Each scenario involves one cluster of models. The clusters

[1]Most of the models in this work are obtained by fine-tuning a pre-trained model on (private) datasets. We focuses on full-finetuning (FFT) that will update all model parameters during the fine-tuning process. The accuracy of FFT outperformed parameter-efficient fine-tuning (PFT) such as Low-Rank Adaptation (LoRA) [41] for scenarios involving large-scale fine-tuning data, low privacy budget, and complex tasks [5, 11, 41]. Our approach can also be applied to deduplicate heterogeneous LLM models or LLMs with LoRA weights merged.

differ in the numbers of $(5-20)$ models, model architectures, training datasets, distribution of epsilons, privacy loss increase threshold ϵ^*, and utility drop threshold u^*. (The latter two constraints are shared by all models in each cluster for simplicity of experiment settings.) Following our base model selection algorithm proposed in Sec. 4.2, the first model is selected as the base model for all scenarios except A2. All models in A2 are dangling models, and they will use the model from A3 with $\epsilon = 0.2$ as their base model. Scenarios C1-C4 in Tab. 1 involve subsets of A1, B2, B3, and A3 respectively, for an ablation study of the base model selection algorithm.

Two Realistic Scenarios. We also introduce two realistic scenarios corresponding to two types of buyers in Fig. 2.

(1) *Multi-User Many-Model Scenario,* where a broker trained a total of 50 distinct models, which is a union of models in Tab. 1. These models are sold to a buyer who serves those models to multiple independent users (with different trust and payment levels) concurrently on a cloud. (A similar scenario is to directly sell each model to an independent buyer.)

(2) *Single-User Heterogeneous-Model Scenario,* where a broker deduplicated seven models to save operational costs, including four ViT models finetuned on CIFAR10 for object detection [49], CelebA (for face attribution classification) [60], GTSRB (for traffic sign recognition) [76], and SVHN (for streetview house number recognition) [68], respectively, all with $\epsilon = 2$, and three RoBERTa-base models finetuned on SST2 (for sentiment analysis) [75], IMDB (for review classification) [63], and QNLI (for question-answering natural language inference) [81], respectively, all with $\epsilon = 5$. It chose to use the first ViT model as the base model to deduplicate the rest six models. Then, a buyer purchased the six deduplicated models and deployed them in a resource-constrained environment to perform various tasks required by a social robot.

Table 1. The Deduplication Scenarios

	Model	Data	Epsilons	ϵ^*	u^*
Diverse scenarios for algorithm evaluation and ablation studies (Sec. 5.2 and Sec. 5.4)					
A1	Roberta	QNLI [81]	[1.0, 2.0, 4.0, 6.0, 7.0]	1.0	0.015
A2	Roberta	SST2 [75]	[0.3, 0.4, 0.6, 0.8, 1.0]	0.05	0.015
A3	Roberta	5 MNLI [82] parts	[0.2, 0.4, 0.8, 1.6, 2.0]	0.2	0.015
A4	ViT	CIFAR100 [49]	[0.5, 0.6, 0.75, 1.0, 2.0]	0.5	0.020
A5	ResNet	CelebA [60]	[0.4, 0.6, 0.8, 1.0, 2.0]	0.5	0.020
B1	Roberta	QNLI	[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0]	1.0	0.015
B2	ViT	CIFAR100	[0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95]	0.5	0.020
B3	ResNet	CelebA	[0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 2.1]	0.2	0.020
Sub-scenarios for base model selection ablation study (Fig. 9)					
C1	Roberta	QNLI	[5.0, 6.0, 7.0, 8.0, 9.0]	5.0	0.015
C2	ViT	CIFAR100	[0.5, 0.6, 0.75, 1.0, 2.0]	0.5	0.020
C3	ResNet	CelebA	[0.7 0.8, 0.9, 1.0, 2.0]	0.7	0.020
C4	Roberta	4 MNLI parts	[0.4, 0.8, 1.6, 2.0]	0.3	0.015

5.1.2 Comparison. None of the existing model deduplication mechanisms has considered privacy. Therefore, **to evaluate the effectiveness of our overall deduplication and accuracy validation approach**, we considered the following algorithms, which used our base model selection mechanism to ensure privacy. Unless explicitly noted, they all use gradient magnitude for saliency measurement and L2 (Euclidean) distance as similarity measurement:

(1) Our **DRD** as formalized in Alg. 4 and its variant, **DRED**.
(2) The SOTA **Dedup** [93] algorithm as described in Sec. 3.2.

(3) **Mistique** [80] uses MinHash [10] to identify and deduplicate similar model weights without considering accuracy. We improve it to use Dedup's saliency measurement and accuracy validation steps described in Sec. 3.2 to provide an accuracy guarantee.
(4) The **Greedy-N** algorithm, which we developed as a static variant of **DRD** and **DRED**. Unlike Dedup, it does not stop after a failure of accuracy validation, instead, it will roll back and then move on to process the next batch.
(5) We also developed a **Monte Carlo Tree Search (MCTS)** [14] algorithm, where each state represents a deduplication scheme, and an action selects a group of N blocks from the target model to be deduplicated using the most similar blocks from the base model and run an accuracy validation. If the validation fails, the reward (i.e., storage costs saving) is computed and back-propagated. It repeats the process until a time budget is achieved.

In addition, we also considered two reference baselines.

(1) **Original**, which does not perform any deduplication.
(2) **Retrain**, which simply finetunes the models using DP-SGD with the new privacy budget achieved by our deduplication approach. This approach provides the upper-bound accuracy with the same privacy budget increase without deduplication.

To evaluate our base model selection method, we extended Alg. 2 to allow multiple base models for one target model by iteratively adding qualified base models until the ϵ increase exceeds the limit, called **Multi**. We further extend **Multi** to allow a deduplicated target model to serve as a base model, called **Cross**.

5.1.3 Measurements. We considered the following measurements.
Compression ratio (C.R.). We measure the compression ratio for both individual models and clusters of models. Let $M_1, ..., M_n$ denote a cluster of models including the base model, and $M'_1, ..., M'_n$ represent these models after deduplication. Each model M_i or M'_i represents a set of unique blocks, and $|M_i|$ or $|M'_i|$ represents the number of blocks in the set. The overall compression ratio is computed as $|M'_1 \cup ... \cup M'_n|/|M_1 \cup ... \cup M_n|$. For a deduplicated model M'_j with its base model M_i , the compression ratio is defined as $|M'_j - M_i|/|M'_j|$. *The smaller the compression ratios, the better.*
Accuracy. We measure the actual accuracy of individual models. In the ablation studies, we use $\max(\Delta u)$ to represent the maximal accuracy drop of individual models in the group.
Model Inference Latency. A great benefit brought by the reduction of the memory footprint of models is the saving of the latency for serving these models in resource-constrained environments where not all models can fit into memory, which is measured in seconds.
Privacy Budget. The privacy budget for fine-tuning a model on a dataset using DP-SGD contains both ϵ and δ. This paper focuses on meeting the constraints on ϵ, which dominates the privacy budget. These delta values are sufficiently small, set to the inverse of the training dataset size, ranging from 6.1×10^{-6} (CelebA) to 2.0×10^{-5} (CIFAR100). For deduplicated models, the epsilon and delta values are computed according to Theorem 4.1.

5.1.4 Experimental Environments. The deduplication experiments are run on a machine with an Intel(R) Xeon(R) Silver 4310 CPU (2.10 Hz) with 128 GB memory and one NVIDIA A10 GPU with 24 GB GPU memory. To simulate a resource-constrained environment, the model serving experiment for the multi-user many-model scenario is run on a c5a.2xlarge instance (8 CPU cores, 16 GB main memory). All other model-serving experiments are run on a c5a.xlarge instance (4 CPU cores, 8 GB main memory).

Table 2. Comparison of Different Deduplication Algorithms. Compression Ratios (C.R.) are in %.

	A1		A2		A3		A4		A5		B1		B2		B3	
	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.
Dedup-20	48.1	42	95.7	18	**56.4**	38	**32.1**	60	**87.4**	12	40.8	98	34.5	116	85.4	57
Dedup-30	48.1	**30**	95.7	**17**	56.9	**28**	33.4	**43**	94.9	**9**	**39.7**	**70**	**32.6**	**85**	**85.2**	57
Mistique-20	84.4	19	100.0	10	**77.3**	26	63.5	56	97.6	**9**	**76.2**	108	90.4	126	91.9	38
Mistique-30	84.4	**15**	100.0	10	78.0	**21**	63.5	**38**	**94.9**	10	88.6	**72**	**88.8**	**81**	**87.6**	38
MCTS-20 (Ours)	**39.9**	257	82.3	**246**	36.9	391	33.4	909	**35.9**	743	23.3	1280	20.4	2266	**39.7**	2628
MCTS-30 (Ours)	41.1	**107**	**58.4**	307	**35.4**	**331**	**32.0**	**571**	42.2	**447**	**20.1**	**1031**	**20.2**	**1417**	40.7	**1031**
Greedy-20 (Ours)	28.5	44	**29.4**	55	**24.4**	44	**30.0**	60	**33.3**	48	**16.1**	99	**18.6**	135	**37.2**	228
Greedy-30 (Ours)	**25.1**	**32**	30.4	**40**	29.9	**32**	32.0	**40**	36.9	**32**	16.2	**72**	20.8	**90**	40.8	**152**
DRD (Ours)	**29.7**	**18**	**27.3**	**30**	24.4	**16**	34.1	**19**	35.5	**21**	**17.0**	**55**	24.7	**50**	**37.6**	**114**
DRED (Ours)	31.6	26	35.6	36	**23.9**	20	**33.4**	26	**34.2**	25	17.7	59	**24.6**	56	37.7	149

5.2 Evaluation on Diverse Scenarios

5.2.1 Overall Results. We first compared the **compression ratio** and the required **number of validations** of various baselines on scenarios A1-A5 and B1-B3. The results, presented in Tab. 2, showed that our DRD and DRED achieved 1.9 to 3.7× better compression ratio than Mistique, and 1.3 to 3.3× better than Dedup. Dedup and Mistique sometimes achieved poor compression ratios (e.g., > 95% for A2 and A3) with very small validation numbers. That is because of their early stop mechanism (i.e., immediately stop in case of a validation failure). We also found that MCTS requires numerous validation steps yet fails to achieve a low compression ratio due to the large search space. For Greedy-N, it is challenging to automatically tune the value of N. When we tune N to achieve a competitive compression ratio, the validation number (a constant $|M_t|/N$) is usually worse than DRD and DRED. In conclusion, our DRD and DRED achieved better trade-offs between the model compression ratio and the validation times.

Deduplication Overhead. Note that most of the deduplication time is spent on accuracy validation. For example, each validation takes 85.6 seconds for Roberta on QNLI, 74.4 seconds for Roberta on SST2, 58.3 seconds for Roberta on one MNLI partition, 84.7 seconds for ViT on CIFAR100, and 84.8 seconds for ResNet on CelebA. It highlights the importance of balancing both objectives.

Impact on Individual Models. Fig. 4 illustrated each model's **compression ratio** and **accuracy** after applying various deduplication approaches to individual models in A1-A5 of Tab. 1. The results confirmed the storage benefits achieved by our proposed DRD and DRED methods. Considering individual models, DRED outperformed Dedup-20 and Mistique-20 by up to 35×. In addition, the Retrain baseline (i.e., full fine-tuning) achieved the best accuracy within the same privacy constraint. However, it does not perform any deduplication. Other baselines achieved similar accuracy to ours since we are using the same accuracy constraints, except that in a few cases where, Dedup and Mistique stopped early due to an accuracy validation failure, leading to high accuracy accompanied by a significantly worse compression ratio (i.e., close to 100%, which means zero compression) than our approaches.

Model Loading and Inference Speedup. As illustrated in Fig. 5, we compared the latency with and without applying our proposed deduplication approach (based on DRD) for A2, A3, and B1-B3, in AWS c5a.xlarge instance with EBS gp2 SSD and EBS magnetic HDD, as described in Sec. 5.1.4. (A1, A4, and A5 are subsets of B1, B2, and B3, and their performance in this experiment is similar to A2 and A3, which are omitted due to space limits.) For this experiment, we measured and broke down the overall latency into loading (including reconstruction) and inference latency for serving 100 inference queries for each scenario. Each query involves a model randomly sampled from the models in the corresponding scenario. The results showed that our approach achieved significantly better speedups in inference for scenarios that involve more models. It achieved 5.6× and 14.1× speedup of the overall latency in SSD and HDD respectively for serving 10 Roberta models in B1, 4.3× to 31.0× for serving 10 ViT models in B2, and 2.2× to 18.8× for serving 20 ResNet models in B3. Even

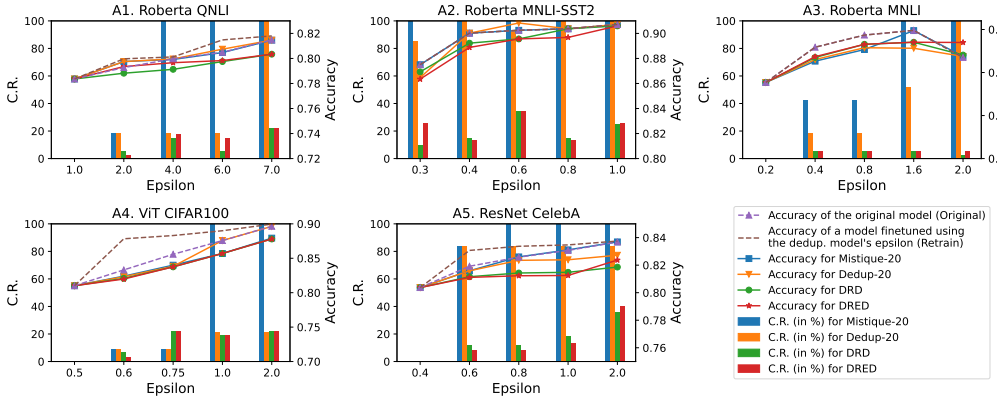


Fig. 4. Compression ratio (C.R.) for individual models in A1 to A5. C.R. of the base model is not affected by deduplication and is thus not shown. A2's base model is the first model in A3.

for smaller scenarios such as A2 and A3, our deduplication approach achieved around 1.3× speedup for both SSD and HDD. The results demonstrated that by deduplicating the models' weights, the overall memory footprint required to serve multiple models can be significantly reduced, bringing lower I/O overheads and lower cache misses.

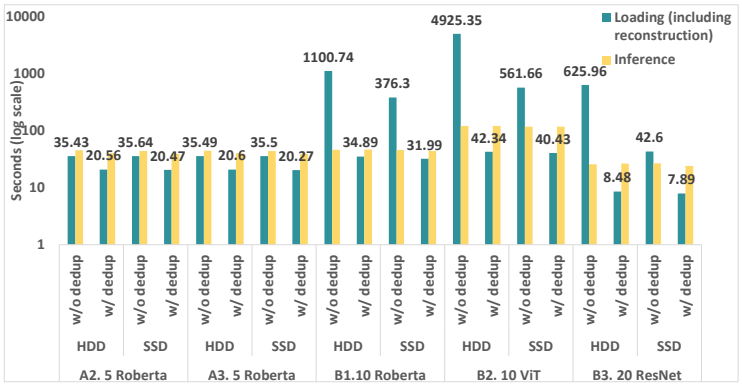


Fig. 5. Latency breakdown of serving 100 inference queries involving multiple models randomly w/o and w/ deduplication (DRD is used, and latency is represented in log scale)

5.2.2 Base Model Selection. We compared the overall compression ratio and increased privacy costs of our base model selection algorithm (Alg. 3) to the Multi and Cross baselines on five distinct scenarios A2, A3, and B1 to B3, as shown in Fig. 6. To allow Multi and Cross, we relax the ϵ^* of each model to be the sum of the first three models in its scenario, while u^* is kept the same. Our Alg. 3 achieved the lowest overall increase in privacy costs, which is 1.7 to 4.3× lower than Cross and 1.0 to 3.5× lower than Multi. Our approach also achieved 1 to 1.7× and 1 to 1.8× better compression ratios than Multi and Cross respectively. Taking A3 as an example, it achieved 1.7× better compression ratio and 1.5× and 1.75× lower privacy costs compared to Multi and Cross.

5.2.3 SVT for Private Data Validation. We next investigated how using the Sparse Vector Technique (SVT) for model evaluation on private validation datasets affects compression ratio and accuracy. We set the SVT cut-off to 3 and allocated the privacy budget for SVT as the sum of the base and target model budgets, ensuring no additional privacy cost over the entire dataset. We present

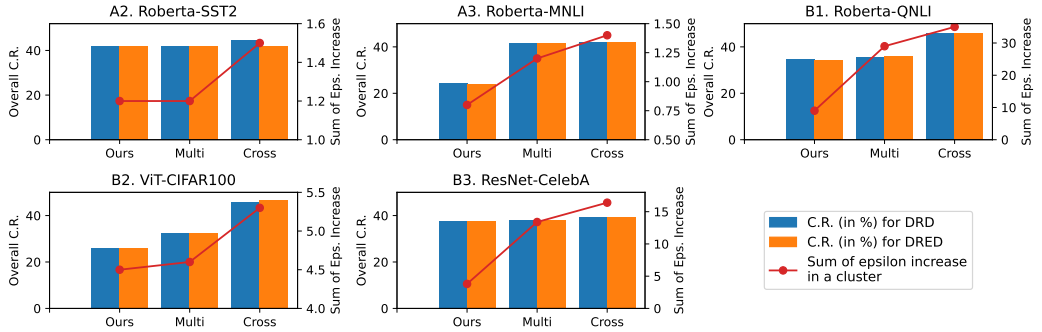


Fig. 6. Comparison of Base Model Selection Strategies.

Table 3. Comparison of Deduplication Algorithms With SVT. Compression ratios (C.R.) are in %.

	A1		A2		A3		A4		A5		B1		B2		B3	
	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.	C.R.	#Val.
Greedy-20	28.9	44	**72.7**	22	**27.3**	44	**35.9**	56	**42.1**	47	18.0	97	26.6	127	**40.3**	206
Greedy-30	28.9	**32**	77.3	**20**	29.9	**32**	42.8	**38**	50.0	**32**	**16.3**	**72**	**24.3**	**90**	42.8	**148**
DRD	33.6	**24**	70.4	21	24.9	**20**	37.8	24	**38.1**	**21**	**18.0**	51	**28.8**	51	**42.3**	113
DRED	**32.1**	28	**69.9**	**17**	**24.2**	26	**35.3**	**21**	40.4	24	19.0	**49**	29.1	**46**	43.5	**96**

the overall results for scenarios A1-A5, and B1-B3 in Tab. 3, using Greedy-N, DRD, and DRED. The comparison between Tab. 2 and Tab. 3 showed that the number of validation steps of the Greedy-N algorithm is more constrained if SVT is used, which worsened the compression ratio, leading to a gap ranging from 0.1% (on A3) to 46.9% (on A5, from 30.4% to 77.3%). However, DRD and DRED are more robust to SVT's cutoff on the validation failures, because of their dynamic range selection without compromising compression ratio, for which the gap between the compression ratios w/o SVT and w/ SVT ranges from −4.0% (on A5) to 4.5% (on B2). We also showed the impacts to individual models on B2 and B3 scenarios in Fig. 7. The results confirmed that SVT usually worsens the compression ratio by up to 16.7% and in certain cases it improves the compression ratio by up to 3.2% (due to randomness introduced by the noisy utility drop comparison). Observations in other scenarios are similar.

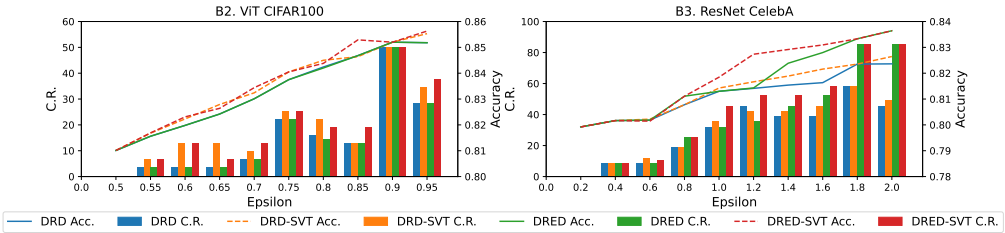


Fig. 7. Deduplication with SVT-based accuracy validation. The compression ratio (C.R.) of the base models, is not not shown.

Table 4. Comparison of different deduplication algorithms for the multi-user many-model scenario including 50 models.

	Dedup-20	Dedup-30	Mistique-20	Mistique-30	MCTS-20 (Ours)	MCTS-30 (Ours)	Greedy-20 (Ours)	Greedy-30 (Ours)	DRD (Ours)	DRED (Ours)
C.R. (%)	57.1	46.7	87.1	88.6	32.2	28.3	**21.2**	23.4	23.7	24.9
#Val	327	257	308	222	6811	4117	561	386	264	320

Table 5. Comparison of different deduplication algorithm for the single-user heterogeneous-model scenario.

Datasets	Roberta-SST2 ($\epsilon = 5$)		Roberta-IMDB ($\epsilon = 5$)		Roberta-QNLI ($\epsilon = 5$)		ViT-GTSRB ($\epsilon = 2$)		ViT-CelebA ($\epsilon = 2$)		ViT-SVHN ($\epsilon = 2$)		Overall	
	C.R.(%)	#Val.	C.R.(%)	#Val.	C.R.(%)	#Val.	C.R.(%)	#Val.	C.R.(%)	#Val.	C.R.(%)	#Val.	C.R.(%)	#Val.
Greedy-1	63.4	210	69.5	211	68.6	210	54.2	578	15.9	578	54.8	578	58.8	2365
Greedy-20	90.3	11	90.8	11	90.3	11	73.7	29	23.1	29	77.6	29	73.8	120
Greedy-30	99.8	7	100.0	8	85.6	7	77.6	20	25.1	20	77.6	20	76.0	82
DRD	87.5	5	88.0	5	84.7	8	77.6	11	21.4	8	79.3	13	73.6	50
DRED	87.5	8	88.0	8	82.8	18	78.2	18	23.5	13	78.2	18	73.8	83

5.3 Two Realistic Scenarios

5.3.1 Multi-User Many-Model Scenario. The overall deduplication effectiveness w/ public validation datasets (w/o SVT-based validation) is shown in Tab. 4. Greedy-20 achieved the best compression ratio of 21.2%. However, it requires 561 validations, representing a long latency of more than 10 hours; Mistique-30 achieved the lowest number of validations, 222, but its compression ratio is not ideal, 88.6%. We found our DRD method achieves the best trade-off between compression ratio and validation overheads, with a near-optimal compression ratio of 24.5% and a near-optimal validation number of 264. The results on private validation dataset (w/ SVT-based validation) is similar, the DRD and DRED achieved similar effective compression ratios (32.0% and 32.4%) with Greedy-20 and Greedy-30 (31.4% for both), while the former's validation numbers (256 and 234) are significantly lower than the latter (496 and 362).

When serving all 50 models concurrently on a c5a.2xlarge instance, the end-to-end latency (including total model loading and inferences) for all 100 inference queries (each query randomly selects one model for inference) are shown in the left part of Fig. 8. Deduplication achieved a 28× speedup when HDD is used, and 11× speedup when SSD is used.

5.3.2 Single-User Heterogeneous-Model Scenario. Tab. 5 demonstrates that our deduplication algorithms (w/o SVT-based validation), such as DRD is able to achieve benefits even for multiple heterogeneous models. DRD outperformed all baselines in minimizing the total required number of validations (50). It also achieved the second-best compression ratio (73.6%). While Greedy-1 achieved the best compression ratio (58.8%), it is at the cost of 2,365 validations (i.e., more than 50 hours). On the contrary, DRD only requires 50 validations, which is a 47× speedup.

When serving all 6 models concurrently on a c5a.1xlarge instance, the end-to-end latency for all 100 random inference queries (each query randomly selects one model for inference) is illustrated in the right part of Fig. 8. Deduplicated models achieved a 43× speedup of the overall inference time on HDD storage and a 14× speedup, compared to non-deduplicated models.

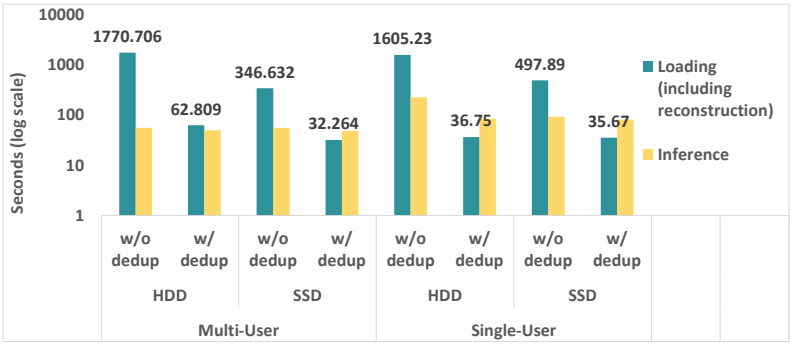


Fig. 8. Latency breakdown of serving 100 random inference queries for the realistic scenarios w/o and w/ deduplication (DRD is used, and latency is represented in log scale)

5.4 Ablation Studies

5.4.1 Impact of Deduplication Hyper-Parameters. We compared the impact of block size, saliency measurement, saliency aggregation method, and distance measure. All experiments in this section used the DRED algorithm. We first measured how varying block sizes affect compression ratio, validation steps, and accuracy drop using scenario A4, which consists of five ViT models. We observed that smaller blocks usually resulted in better compression ratios. However, smaller block sizes also lead to more blocks per model, thereby increasing the number of validation steps. In addition, when the block size decreases to a point, the compression ratio will not improve anymore. This is because salient weights exhibit locality and are distributed in multiple small clusters, as verified by Lee et al. [52]. When the block size is smaller than the size of these clusters, the compression ratio does not improve.

We further compared the effectiveness of different saliency measures, the saliency aggregation methods, and block similarity measurements, using the deduplication of the A1 scenario as an example. The results in Tab. 6 showed that our proposed gradient magnitude measurement outperformed other measurements regarding compression ratio. We also find that profiling of the weight magnitude is the most efficient, taking 113 seconds, while calculating the Wanda score, Fisher information, and gradients using the entire validation dataset takes 145 seconds, 6,576 seconds, and 1,060 seconds, respectively. For saliency aggregation, we found using the l2-norm of weight gradients as the block saliency outperformed l1-norm, l-infinite, and third quartile of weight gradients, as illustrated in Tab. 6. For measuring the similarity of two blocks, as shown in Tab. 6, l2 (Eclidean) distance outperformed other metrics.

Table 6. Ablation Study of Deduplication Hyper-Parameters

	Values	C.R.(%)	#Val.	max (Δu)
A4. block size (#floating points)	4,194,304	53.6	9	0.016
	2,097,152	41.1	16	0.017
	1,048,576	33.4	26	0.018
	524,288	32.5	29	0.020
	262,144	32.12	29	0.020
A1. saliency measurement	Weight Magnitude	63.8	49	0.015
	Wanda	47.3	30	0.015
	Fisher Information	45.9	24	0.015
	Gradient Magnitude	32.8	31	0.019
A1. saliency aggregation method	l1-norm	27.6	26	0.014
	l2-norm	26.1	31	0.013
	l-infinite	30.0	35	0.015
	3rd-quartile	27.1	32	0.015
A1. pairwise distance measurement	l1-distance	25.9	29	0.014
	l2-distance	40.2	31	0.012
	cosine	92.1	63	0.014

5.4.2 Impact of Accuracy Threshold. We used various accuracy drop thresholds to deduplicate models in A1, A4, and A5 scenarios, and recorded the compression ratios shown as percentages in Tab. 7. We found that generally, larger accuracy drop thresholds yield better compression ratios. In addition, a negative accuracy drop threshold requires the resulting model to have a higher accuracy than the original one. When the threshold was set to -0.5%, no blocks could be deduplicated in A4 and A5. However, for A1, a few blocks could be deduplicated to improve the accuracy by 0.5%, which indicates that although we focus on constraining the utility drop, deduplication may improve utility in certain cases.

5.4.3 Impacts of SVT hyper-parameter tuning. We also investigated the impacts of cut-off c in SVT-based validation. (Our configuration of privacy budget for SVT-based validation is consistent with Sec. 5.2.3.) Taking the deduplication in the A1 scenario as an example, we varied the value of

Table 7. Compression Ratios for Different Accuracy Thresholds

Thresholds(%)	-0.5	0.0	0.5	1.0	1.5	2.0
A1. Roberta	83.7	58.0	23.4	21.9	21.1	20.8
A4. ViT	N/A	84.4	54.1	41.5	35.9	31.0
A5. ResNet	N/A	80.2	44.8	37.8	35.1	31.2

c from 2 to 7, and recorded the compression ratio, the maximum accuracy drop, and the number of validations in Tab. 8. The results showed that when c reaches a point (e.g., 3 in the example), increasing it will not improve the compression ratio, although a larger cut-off allows for more validation failures before it terminates the deduplication process. That's because increasing c will raise the noise level added to the query result, leading to less accurate comparisons.

Table 8. Impacts of SVT's Cut-off for the A4 cluster

Cut-off c	2	3	4	5	6	7
C.R.(%)	33.6	31.7	31.7	31.7	31.7	31.7
max (Δu)(%)	0.66	1.14	1.14	1.14	1.14	1.14
Num. of val.	18	23	26	28	28	28

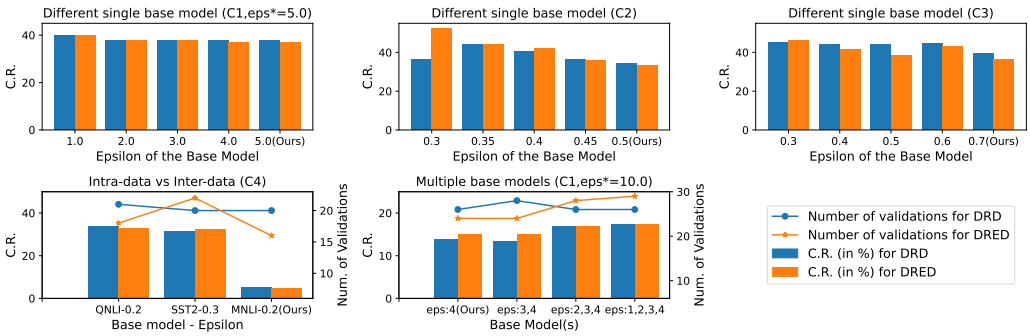


Fig. 9. Ablation study for base model selections.

5.4.4 Impacts of base model selection. We further investigated how the selection of different base models in each group will affect the compression ratio and number of validations (i.e., deduplication efficiency) in three different situations. *(1) Base model selection for models trained on the same dataset with different privacy budgets* evaluated on scenarios C1-C3. For each scenario, we compared the overall compression ratio achieved in deduplicating all models from this scenario using different base models with their ϵ specified in the x-axis. Among these base models, only the model marked with "Ours" was selected by our algorithm. Other base models were trained for comparison. As presented in Fig. 6, when the base model's epsilon increased, the compression ratio improved (i.e., decreased), which is expected since the similarity to the target models (i.e., similarity of privacy budgets) improved. *(2) Base model selection for models trained on different datasets.* For scenario C4 with multiple models trained on disjoint MNLI partitions, we found using a base model trained on a similar dataset (e.g., MNLI) outperformed other candidate base models. *(3) Using multiple base models for a group.* In C1, if we change the privacy constraint into 10.0, as shown in the last plot, using a model with $\epsilon = 4$ (selected by our algorithm) as the base model achieves a better compression than using three or four base models, which showed that the quality of base models is more important than quantity.

5.5 Summary of Findings

Overall, our proposed work brings 17% to 38% C.R. (compression ratio) for eight representative scenarios, leading to a 31× model serving speedup. In addition, in a realistic scenario that involves 50 models, our work achieved 23.7% C.R., resulting in 28× serving speedup. Even in a challenging scenario that runs multiple heterogeneous models required by a social robot in a resource-constrained environment, our approach achieved an impressive 43× inference speedup brought by a C.R. of 74%.

In addition, our dynamic deduplication algorithms DRD and DRED have improved the compression ratio by up to 3.3× compared to the best of existing deduplication algorithms (with our base model selection to ensure privacy). Moreover, our base model selection strategy has reduced privacy costs by up to 3× compared to alternative base model selection designs. Our SVT-based accuracy validation strategy is also demonstrated as effective in minimizing the privacy loss on private validation datasets, with negligible impacts on compression ratio if coupled with DRD and DRED.

Our other observations include:

Deduplication Algorithm. Our proposed DRD and DRED algorithms provide the best trade-off between compression ratio and deduplication overhead, compared to all other baselines. Greedy-1 can achieve the optimal compression ratio at the cost of inhibiting deduplication latency.

Block Size Selection. A smaller block size usually leads to a better compression ratio with higher deduplication overhead. However, a very small block size not only slows down the deduplication, but also overlooks the synergy among adjacent blocks, thus not benefiting the compression ratio.

Saliency Measure. Weight magnitude is easy to measure but its accuracy is sub-optimal. Wanda [77] based on weights and activation is only applicable to linear layers. Fisher information [58] suffers from the value vanishing problem caused by the square of small gradient values. Our proposed gradient magnitude measure is more effective in our target problems than the above approaches.

Base Model Selection. Compared to our proposed base model selection strategy, baselines such as Multi and Cross do not improve the compression ratios. However, these baselines significantly increase the composed model-specific privacy costs because they introduce multiple high-privacy-cost base models.

SVT Cut-off. A smaller cutoff incurs smaller noise. However, it limits the maximum number of failure during deduplication. A default choice of 3.0 works well in most of our experiments.

6 Further Discussion

In this section, we discuss the scalability of our proposed approach and the dynamic addition/removal of models.

Scalability of the Deduplication Process. First, each pair of base and target models can be deduplicated in parallel with other pairs. Second, the deduplication of a target model (M_j) using a base model (M_i) can be split into two phases: (1) running the deduplication algorithm, where the peak memory is $mem\_dedup = size(M_i) + block\_size$, if the base model M_i is cached in memory; and (2) validating the accuracy, with peak memory being $mem\_valid = size(M_j) + size(feature\_maps) + size(input)$, which mainly sums up the size of the target model, the intermediate feature maps created for inference, and the validation data. Therefore, we have $peak\_mem = \max(mem\ \_dedup, mem\_valid)$.

Model Addition/Removal. Our algorithm can be extended to handle dynamic model addition and removal. A batch of newly arrived models will be dispatched to existing or new clusters based on metadata similarity. Then, it will apply Alg. 2 to each new model. For model removal, the process depends on whether the model is a base model. Non-base models can be safely removed without affecting the system. However, if a base model is deemed no longer useful, two strategies can be employed. The first strategy is to retain its blocks that are in use by other models, minimizing

disruption. The second strategy involves archiving the original models in cold storage and re-run the deduplication process for affected models, using an alternative base model.

7 Other Related Works

Both SmartLite [56] and Nexus [71] propose to reuse the shared layers of multiple models fine-tuned from the same pre-trained model. However, they did not consider deduplicating similar but non-identical tensor blocks, which will significantly improve the compression ratio in a broad class of scenarios, e.g., full-parameter fine-tuning, heterogeneous models, models with multiple DP versions, etc. Weight virtualization [52] merges pages across multiple models into a single page. Model merging [85] builds a universal model that excels at multiple tasks by merging the weights of multiple models. However, none of these works have considered privacy. Deduplication of relational data in RDBMS, also known as record linkage, identifies duplicate items through entity matching [28, 66, 78], using various blocking techniques to avoid the pair-wise comparison for dissimilar items [2, 6, 9, 17, 40, 47, 48]. In addition, various techniques leveraged similarity functions to filter out pairs that have similarity scores below a threshold [83] or used LSH to convert similarity join to an equi-join problem [87]. However, these works did not consider how the deduplication will affect the accuracy and privacy of downstream ML applications.

8 Conclusions

This work proposed a privacy-centric redesign of model deduplication techniques to compress multiple models and alleviate the inference and storage costs of multi-tenant and multi-tasking model serving platforms in resource-constrained environments. We identified and formalized a novel accuracy- and privacy-aware model deduplication problem. To address the problem, we proposed novel techniques including a greedy base model selection strategy to optimize the privacy costs and novel deduplication strategies (e.g., DRD/DRED) that deduplicate a dynamic number of blocks each time based on an ordering of blocks by saliency. We also leverage SVT to reduce privacy costs of validating accuracy using private datasets. We conducted detailed evaluations on diverse and representative clusters of models and obtained promising results.

In the future, we will use the techniques introduced in this work to enhance AI/ML in database systems [8, 16, 36, 39, 46, 50, 90, 94], leveraging our prior works on managing AI/ML models using database buffer pool management [93, 95, 96]. It will enable database systems to better support emerging applications such as model marketplace, ML-as-a-Service, and multi-tasking model serving. We will also extend this work to deduplicate model checkpoints and the intermediate data of the model training/inference processes to accelerate privacy-preserving model explanation and debugging queries and integrate the solution with a database system to provide end-to-end support for such queries.

Acknowledgements

This material is based upon work supported by the U.S. Department of Homeland Security under grant award number 17STQAC00001-07-00, National Science Foundation under grant award number 2144923, 2431532, 2302968, 2124104, and 2125530, and National Institute of Health under R01ES033241 and R01LM013712.

Disclaimer

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *CCS*. 308–318.

[2] Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. 2002. Eliminating fuzzy duplicates in data warehouses. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 586–597.

[3] Kathleen Benitez and Bradley Malin. 2010. Evaluating re-identification risks with respect to the HIPAA privacy rule. *Journal of the American Medical Informatics Association* 17, 2 (2010), 169–177.

[4] Daniel Bernau, Günther Eibl, Philip W Grassal, Hannah Keller, and Florian Kerschbaum. 2021. Quantifying Identifiability to Choose and Audit ϱ in Differentially Private Deep Learning. In *Proceedings of the Conference on Very Large Databases*.

[5] Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, et al. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673* (2024).

[6] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. 2006. Adaptive blocking: Learning to scale up record linkage. In *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 87–96.

[7] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? *Proceedings of machine learning and systems* 2 (2020), 129–146.

[8] Matthias Boehm, Michael W Dusenberry, Deron Eriksson, Alexandre V Evfimievski, Faraz Makari Manshadi, Niketan Pansare, Berthold Reinwald, Frederick R Reiss, Prithviraj Sen, Arvind C Surve, et al. 2016. Systemml: Declarative machine learning on spark. *Proceedings of the VLDB Endowment* 9, 13 (2016), 1425–1436.

[9] Andrew Borthwick, Stephen Ash, Bin Pang, Shehzad Qureshi, and Timothy Jones. 2020. Scalable Blocking for Very Large Databases. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 303–319.

[10] Andrei Z Broder. 1997. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE, 21–29.

[11] Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. [n. d.]. Differentially Private Bias-Term Fine-tuning of Foundation Models. In *Forty-first International Conference on Machine Learning*.

[12] Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. 2022. Differentially Private Bias-Term Fine-tuning of Foundation Models. In *Workshop on Trustworthy and Socially Responsible Machine Learning, NeurIPS 2022*.

[13] Dirk G Cattrysse, Marc Salomon, and Luk N Van Wassenhove. 1994. A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research* 72, 1 (1994), 167–174.

[14] Guillaume Maurice Jean-Bernard Chaslot Chaslot. 2010. Monte-carlo tree search. (2010).

[15] Lingjiao Chen, Paraschos Koutris, and Arun Kumar. 2019. Towards model-based pricing for machine learning in a data marketplace. In *Proceedings of the 2019 international conference on management of data*. 1535–1552.

[16] Lingjiao Chen, Arun Kumar, Jeffrey Naughton, and Jignesh M Patel. 2016. Towards linear algebra over normalized data. *arXiv preprint arXiv:1612.07448* (2016).

[17] Xu Chu, Ihab F Ilyas, and Paraschos Koutris. 2016. Distributed data deduplication. *Proceedings of the VLDB Endowment* 9, 11 (2016), 864–875.

[18] Google Cloud. [n. d.]. Restrict data access using analysis rules. Retrieved Jan 17, 2024 from https://cloud.google.com/bigquery/docs/analysis-rules

[19] Reuven Cohen, Liran Katzir, and Danny Raz. 2006. An efficient approximation for the generalized assignment problem. *Inform. Process. Lett.* 100, 4 (2006), 162–166.

[20] Daniel Crankshaw. 2019. *The Design and Implementation of Low-Latency Prediction Serving Systems*. Ph. D. Dissertation. UC Berkeley.

[21] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J Franklin, Joseph E Gonzalez, and Ion Stoica. 2017. Clipper: A low-latency online prediction serving system. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 613–627.

[22] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*. 253–262.

[23] Damien Desfontaines. 2021. A list of real-world uses of differential privacy. *Ted is writing things* (2021).

[24] Benjamin Doerr, Johannes Lengler, Timo Kötzing, and Carola Winzen. 2011. Black-box complexities of combinatorial problems. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 981–988.

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. https://openreview.net/forum?id=YicbFdNTTy

[26] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*. Springer, 1–19.

[27] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3–4 (aug 2014), 211–407. doi:10.1561/0400000042

[28] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2006. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering* 19, 1 (2006), 1–16.

[29] Elias Frantar and Dan Alistarh. 2022. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems* 35 (2022), 4475–4488.

[30] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).

[31] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333.

[32] Jie Fu, Qingqing Ye, Haibo Hu, Zhili Chen, Lulu Wang, Kuncan Wang, and Ran Xun. 2023. Dpsur: Accelerating differentially private stochastic gradient descent using selective update and release. *arXiv preprint arXiv:2311.14056* (2023).

[33] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. 2021. Deep learning with label differential privacy. *Advances in neural information processing systems* 34 (2021), 27131–27145.

[34] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2022. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*. Chapman and Hall/CRC, 291–326.

[35] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision* 129, 6 (2021), 1789–1819.

[36] Hong Guan, Saif Masood, Mahidhar Dwarampudi, Venkatesh Gunda, Hong Min, Lei Yu, Soham Nag, and Jia Zou. 2023. A Comparison of End-to-End Decision Forest Inference Pipelines. In *Proceedings of the 2023 ACM Symposium on Cloud Computing*. 200–215.

[37] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[39] Joe Hellerstein, Christopher Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleksander Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, et al. 2012. The MADlib analytics library or MAD skills, the SQL. *arXiv preprint arXiv:1208.4165* (2012).

[40] Mauricio A Hernández and Salvatore J Stolfo. 1995. The merge/purge problem for large databases. *ACM Sigmod Record* 24, 2 (1995), 127–138.

[41] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[42] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)* 54, 11s (2022), 1–37.

[43] Kai Huang, Boyuan Yang, and Wei Gao. 2023. Elastictrainer: Speeding up on-device training with runtime elastic tensor selection. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*. 56–69.

[44] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. 2018. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961* (2018).

[45] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. 2020. Auditing differentially private machine learning: How private is private SGD? *Advances in Neural Information Processing Systems* 33 (2020), 22205–22216.

[46] Dimitrije Jankov, Shangyu Luo, Binhang Yuan, Zhuhua Cai, Jia Zou, Chris Jermaine, and Zekai J Gao. [n. d.]. Declarative Recursive Computation on an RDBMS. *Proceedings of the VLDB Endowment* 12, 7 ([n. d.]).

[47] Lars Kolb, Andreas Thor, and Erhard Rahm. 2012. Dedoop: Efficient deduplication with hadoop. *Proceedings of the VLDB Endowment* 5, 12 (2012), 1878–1881.

[48] Lars Kolb, Andreas Thor, and Erhard Rahm. 2012. Load balancing for mapreduce-based entity resolution. In *2012 IEEE 28th international conference on data engineering*. IEEE, 618–629.

[49] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[50] Arun Kumar, Jeffrey Naughton, and Jignesh M Patel. 2015. Learning generalized linear models over normalized data. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 1969–1984.

[51] Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip H. S. Torr. 2020. A Signal Propagation Perspective for Pruning Neural Networks at Initialization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HJeTo2VFwH

[52] Seulki Lee and Shahriar Nirjon. 2020. Fast and scalable in-memory deep multitask learning via neural weight virtualization. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 175–190.
[53] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning Filters for Efficient ConvNets. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJqFGTslg
[54] Bing-Rong Lin and Daniel Kifer. 2014. On arbitrage-free pricing for general data queries. *Proceedings of the VLDB Endowment* 7, 9 (2014), 757–768.
[55] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. In *MLSys*.
[56] Qiuru Lin, Sai Wu, Junbo Zhao, Jian Dai, Meng Shi, Gang Chen, and Feifei Li. 2023. SmartLite: A DBMS-Based Serving System for DNN Inference in Resource-Constrained Environments. *Proceedings of the VLDB Endowment* 17, 3 (2023), 278–291.
[57] Jinfei Liu, Jian Lou, Junxu Liu, Li Xiong, Jian Pei, and Jimeng Sun. 2021. Dealer: an end-to-end model marketplace with differential privacy. *Proceedings of the VLDB Endowment* 14, 6 (2021).
[58] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. 2021. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*. PMLR, 7021–7032.
[59] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[60] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*. 3730–3738.
[61] Tao Luo, Mingen Pan, Pierre Tholoniat, Asaf Cidon, Roxana Geambasu, and Mathias Lécuyer. 2021. Privacy budget scheduling. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*. 55–74.
[62] Min Lyu, Dong Su, and Ninghui Li. 2017. Understanding the Sparse Vector Technique for Differential Privacy. *Proceedings of the VLDB Endowment* 10, 6 (2017).
[63] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 142–150.
[64] Xian-Ling Mao, Bo-Si Feng, Yi-Jing Hao, Liqiang Nie, Heyan Huang, and Guihua Wen. 2017. S2JSD-LSH: A locality-sensitive hashing schema for probability distributions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.
[65] Frank D. McSherry. 2009. Privacy Integrated Queries: An Extensible Platform for Privacy-Preserving Data Analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data* (Providence, Rhode Island, USA) *(SIGMOD '09)*. Association for Computing Machinery, New York, NY, USA, 19–30. doi:10.1145/1559845.1559850
[66] Joty MESTS and MON Tang. 2018. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment* 11, 11 (2018).
[67] Priyanka Nanayakkara, Mary Anne Smart, Rachel Cummings, Gabriel Kaptchuk, and Elissa M Redmiles. 2023. What are the chances? explaining the epsilon parameter in differential privacy. In *32nd USENIX Security Symposium (USENIX Security 23)*. 1613–1630.
[68] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. Reading digits in natural images with unsupervised feature learning. (2011).
[69] S Ramaswamy, R Mathews, K Rao, and F Beaufays. 2019. Learning for Emoji Prediction in a Mobile Keyboard. *arXiv preprint arXiv:1906.04329* (2019).
[70] Phillip Rust and Anders Søgaard. 2023. Differential privacy, linguistic fairness, and training data influence: Impossibility and possibility theorems for multilingual language models. In *International Conference on Machine Learning*. PMLR, 29354–29387.
[71] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. 2019. Nexus: A GPU cluster engine for accelerating DNN-based video analysis. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 322–337.
[72] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.
[73] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR* abs/1312.6034 (2013). https://api.semanticscholar.org/CorpusID:1450294
[74] snowflake. [n. d.]. Differential Privacy in Snowflake. https://docs.snowflake.com/en/user-guide/diff-privacy/differential-privacy-overview#label-diff-privacy-loss-budgets.

[75] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1631–1642.

[76] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. 2012. Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. *Neural Networks* 32 (Aug. 2012), 323–332. doi:10.1016/j.neunet.2012.02.016

[77] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. A Simple and Effective Pruning Approach for Large Language Models. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=PxoFut3dWW

[78] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep learning for blocking in entity matching: a design space exploration. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2459–2472.

[79] Florian Tramer and Dan Boneh. 2021. Differentially Private Learning Needs Better Features (or Much More Data). In *International Conference on Learning Representations*. https://openreview.net/forum?id=YTWGvpFOQD-

[80] Manasi Vartak, Joana M F. da Trindade, Samuel Madden, and Matei Zaharia. 2018. Mistique: A system to store and query model intermediates for model diagnosis. In *Proceedings of the 2018 International Conference on Management of Data*. 1285–1300.

[81] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* (2018).

[82] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 1112–1122. doi:10.18653/v1/N18-1101

[83] Chuan Xiao, Wei Wang, and Xuemin Lin. 2008. Ed-join: an efficient algorithm for similarity joins with edit distance constraints. *Proceedings of the VLDB Endowment* 1, 1 (2008), 933–944.

[84] Zheng Xu and Yanxiang Zhang. 2017. Advances in private training for production on-device language models.

[85] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666* (2024).

[86] Wangsong Yin, Mengwei Xu, Yuanchun Li, and Xuanzhe Liu. 2024. Llm as a system service on mobile devices. *arXiv preprint arXiv:2403.11805* (2024).

[87] Chenyun Yu, Sarana Nutanong, Hangyu Li, Cong Wang, and Xingliang Yuan. 2016. A generic method for accelerating LSH-based similarity join processing. *IEEE Transactions on Knowledge and Data Engineering* 29, 4 (2016), 712–726.

[88] Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. 2021. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. *arXiv preprint arXiv:2102.12677* (2021).

[89] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. 2021. Large scale private learning via low-rank reparametrization. In *International Conference on Machine Learning*. PMLR, 12208–12218.

[90] Binhang Yuan, Dimitrije Jankov, Jia Zou, Yuxin Tang, Daniel Bourgeois, and Chris Jermaine. 2021. Tensor relational algebra for distributed machine learning system design. *Proceedings of the VLDB Endowment* 14, 8 (2021).

[91] Shufan Zhang and Xi He. 2023. DProvDB: Differentially private query processing with Multi-Analyst provenance. *Proceedings of the ACM on Management of Data* 1, 4 (2023), 1–27.

[92] Lixi Zhou, K Selçuk Candan, and Jia Zou. 2024. DeepMapping: Learned Data Mapping for Lossless Compression and Efficient Lookup. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 1–14.

[93] Lixi Zhou, Jiaqing Chen, Amitabh Das, Hong Min, Lei Yu, Ming Zhao, and Jia Zou. 2022. Serving deep learning models with deduplication from relational databases. *Proc. VLDB Endow.* 15, 10 (jun 2022), 2230–2243. doi:10.14778/3547305.3547325

[94] Lixi Zhou, Qi Lin, Kanchan Chowdhury, Saif Masood, Alexandre Eichenberger, Hong Min, Alexander Sim, Jie Wang, Yida Wang, Kesheng Wu, et al. 2024. Serving Deep Learning Models from Relational Databases. *Advances in Database Technology-EDBT* 27, 3 (2024), 717–724.

[95] Jia Zou, Arun Iyengar, and Chris Jermaine. 2019. Pangea: monolithic distributed storage for data analytics. *Proceedings of the VLDB Endowment* 12, 6 (2019), 681–694.

[96] Jia Zou, Arun Iyengar, and Chris Jermaine. 2020. Architecture of a distributed storage that combines file system, memory and computation in a single layer. *The VLDB Journal* (2020), 1–25.

Received October 2024; revised January 2025; accepted February 2025