Privacy and Accuracy-Aware Model Deduplication for Emerging Model Infrastructures

Hong Guan[1], Lei Yu[2], Lixi Zhou[1], Xusheng Xiao[1], Li Xiong[3], Jia Zou[1]
[1] Arizona State University, [2] Rensselaer Polytechnic Institute, [3] Emory University

ABSTRACT

Applying data storage techniques to manage multiple AI/ML models trained/finetuned on private datasets is a vital problem in the emerging model infrastructures, such as model marketplaces, model stores, and model serving platforms. Recently, finetuning large language models (LLMs) and foundation models on private datasets has become increasingly popular, requiring the management of many large models. In addition, it becomes essential to train models with different privacy guarantees and utilities to meet diverse user requirements. Such trends not only aggravate memory challenges by having many versions of large models with varying privacy guarantees but also complicate the memory optimization process, such as deduplication, which usually does not consider privacy. In this paper, we study for the first time the problem of deduplicating multiple models with different privacy guarantees to optimize compression while maintaining the privacy and accuracy constraints of the models. We formalize it as an optimization problem. We also propose a novel privacy and accuracy-aware deduplication mechanism based on several techniques, such as model clustering and base model selection, model saliency analysis and dynamic validation, and the sparse vector technique, to reduce the search space and privacy costs. We evaluate and demonstrate the effectiveness of the deduplication mechanism on a variety of models including LLMs, vision transformer models, and ResNet models.

1 INTRODUCTION

Various model infrastructures are being proposed and deployed, such as model marketplaces [43, 49], model stores [2, 76], and model deployment and serving platforms [35], to streamline and enhance the accessibility and management of AI/ML models. Extending data management techniques for efficiently storing and serving multiple AI/ML models in such systems is a critical emerging research area [37, 57, 81]. Existing works [45, 70, 80] applied data deduplication, a popular data management technique, to identify and eliminate similar components across different models to reduce memory footprint and shorten the model inference latency. However, none of these works have considered privacy of the models in such infrastructures. Differential privacy (DP) [18] is a mathematical framework to ensure the privacy of individuals in data sets and is increasingly adopted in real-world applications in Google [30], Apple [61], etc. In the model infrastructures we consider, models can be trained with popular DP-enhanced algorithms (e.g., DP-SGD [3]), corresponding to different privacy guarantees for private training data. In this work, we focus on applying deduplication to compress multiple AI/ML models with a privacy guarantee and optimize trade-offs among accuracy, privacy, and compression ratio.

A Motivating Scenario. In model marketplaces, data brokers train models with varying privacy budgets using the well-known DP-SGD [3], and the trained models can be priced using the arbitrage-free approach for different model buyers or end users [47, 49]. A model with a higher privacy budget corresponds to a lower privacy guarantee but usually better utility (less noise). On one hand, model users can pay more to use models with higher privacy budgets and more accurate inference for critical decision-making or save monetary costs by choosing less accurate models with lower privacy budgets for simpler tasks. On the other hand, maintaining multiple models creates significant memory pressures for querying and serving models, in addition to increased storage costs.

Foundation models, large language models (LLMs), and image/video generation models, are widely deployed. The recent trend of finetuning these models on different datasets, leads to even more memory and storage pressures. For example, in the first week when Llama-3 [67, 68] was publicly released, around 1,300 models finetuned from Llama-3 were published in HuggingFace Hub [73], with model sizes ranging from 16 to 150 gigabytes. LLM models are increasingly transferred to private domains [29] and can be finetuned with DP to protect the privacy of the finetuning datasets.

Existing Deduplication Approaches and Limitations. Since AI/ML models tend to share a significant amount of similar weights, existing works have applied deduplication on the models, which will not significantly impact the accuracy but can dramatically improve memory bandwidth and inference latency for better model services [70, 80]. In a general block-based deduplication approach [80], as illustrated in Fig. 1, the models' weights are partitioned into uniform-sized blocks. It identifies similar blocks that can be deduplicated without significantly degrading models' utility. These remaining blocks are then compactly stored adjacent to one another, facilitating rapid retrieval and efficient reconstruction of the deduplicated models. However, in our target scenario, the privacy guarantee of a model will change if any of its blocks get deduplicated by blocks from other models due to the composition of DP. No existing works have considered how to maintain the privacy constraints of the models during deduplication and optimize trade-offs among privacy, accuracy, and compression ratio.

Target Problem and Challenges. In this paper, we consider the emerging and important problem of deduplicating multiple models with DP guarantees to optimize the compression ratio while maintaining the privacy and accuracy constraints of the models, as illustrated in Fig. 1. Tremendous challenges arise when considering privacy, accuracy, and compression ratio simultaneously for the deduplication process.

Challenge 1. Privacy Cost Composition During Deduplication. Due to the sequential composition property of DP, a deduplicated model composed of weights from multiple models trained on the same datasets will have an accumulated privacy cost equal to the sum of the privacy costs of all models involved in the process (Sec. 4). Therefore, unlike ordinary model deduplication, deduplicating DP models may significantly change the privacy guarantees of the resulting models.

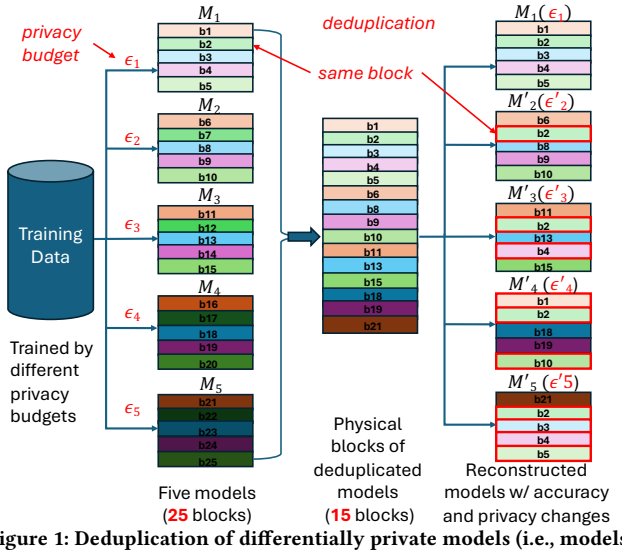

Figure 1: Deduplication of differentially private models (i.e., models trained using DP-SGD with different privacy budgets)

Challenge 2. Extensive Search Space. The search space of optimizing a block-based deduplication plan [80], which uses one block to replace another from the same or a different model, is exponential of the numbers of blocks and models. While existing works [80] only consider simple models such as fully connected neural networks and word2vec, the emerging complicated models, such as LLMs [51] and vision transformer (ViT) models [28], make the balance of the three objectives even more difficult.

Challenge 3. Time Overheads of Accuracy Validation. Throughout the deduplication process, it needs to periodically assess the accuracy of each modified model to ensure that any accuracy decline remains within acceptable limits. This validation process presents a trade-off: frequent accuracy checks are time-consuming, while infrequent validations may cause more inaccurate rollbacks of successful deduplications and hurt the compression ratio.

Challenge 4. Privacy Costs Caused by Private Validation Dataset. When no suitable public dataset is available for validation, using a private dataset incurs additional privacy costs that escalate rapidly, as each validation step introduces a privacy cost.

To address the challenges, we propose **a novel privacy and accuracy-aware model deduplication process** that optimizes the compression ratio of all models while meeting user-specified accuracy and privacy constraints, and a fairness rule, which defines that for all models trained with the same dataset, architecture, and task, the model that has a lower privacy budget should also be lower in utility [49], otherwise some low-utility and high-privacy-budget models will never be used and waste resources.

To address Challenge 1 and 2, models are clustered based on model similarity. For each group, we try to find high-utility models from the group to serve as base models in order to minimize the accuracy change and privacy cost increase of other models after deduplication (See Sec. 5.2. If such models do not exist, it will search similar models in other groups.) Our idea is to reduce the multi-model deduplication problem into a series of *two*-model deduplication sub-problems, each deduplicating a model (i.e., target model) with a base model. This strategy has two benefits. First, the privacy cost increase for each deduplicated model is mainly bounded by the privacy costs of its base models according to the DP composition. Therefore, the increased privacy cost is controlled by carefully selecting the base models. Second, since a base model is similar to its target models, it is easier to balance privacy, accuracy, and compression ratio.

To address Challenge 3, we employ two key techniques for each two-model deduplication sub-problem (Sec. 5.3): saliency analysis and a dynamic validation strategy that groups a dynamic number of blocks for deduplication and accuracy validation each time. Our results demonstrate that combining a gradient-based saliency analysis approach with dynamic validation effectively balances the compression ratio, accuracy, and the number of required validation steps (which determines the deduplication latency).

To address Challenge 4, we propose a novel validation process based on the Sparse Vector Technique (SVT) [18] (Sec. 5.4). This approach transforms the validation steps into a series of boolean questions about accuracy drops, maintaining a fixed privacy budget until the number of failed validation steps reaches a predetermined cut-off value. This process allows for privacy-preserving accuracy assessments during the deduplication process without incurring a significant privacy cost.

Our work may start a line of research on privacy-centric redesign of data deduplication techniques for managing AI/ML models. The key contributions include:

- We are the first to formulate the problem of deduplicating models trained with different privacy guarantees for different users. We are also the first to design and develop an end-to-end pipeline to address the problem. Our work dramatically alleviates the inference costs, reduced the memory footprint, and saves memory bandwidth for emerging privacy-preserving model infrastructures, including model marketplaces, model stores, and model serving platforms.
- We propose a new algorithm to automatically deduplicate the models while meeting the privacy and utility requirements, by integrating an array of novel techniques such as model clustering, base model selection, model tensor block salient analysis, dynamic validation strategies, and SVT-based validation.
- We implement the proposed system and conduct comprehensive evaluations using a variety of models including fine-tuned foundation models such as Roberta and ViT. The results demonstrate that our system achieves a high compression ratio of models with acceptable accuracy drop and privacy cost increase, and enables faster model serving in memory-constrained environments.

2 PRELIMINARIES

Differential Privacy [18] (DP) is a rigorous mathematical framework for quantifying and managing privacy risks in data analysis. It provides formal guarantees about the privacy of individuals whose information is included in a dataset, even in the face of arbitrary background knowledge and data linkage. DP has several variants, such as pure DP (ϵ-DP), approximate DP ((ϵ, δ)-DP), Local DP, Rényi DP (RDP), Gaussian DP (GDP), etc.

Definition 1. [Differential Privacy (DP)] 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.

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 (Sensitivity)**.** 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.

Properties of DP [18]. DP is characterized by three key properties: sequential composition, parallel composition, and post-processing.

Sequential Composition shows how privacy guarantees add up when multiple analyses are performed on the same data.

Theorem 1. *If $\mathcal{M}_1$ 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.*

Parallel Composition [56] demonstrates that when analyses are performed on disjoint subsets of data, one can take the maximum privacy cost rather than the sum.

Theorem 2. *Let $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$ be disjoint subsets of the database $\mathcal{D}$. If $\mathcal{M}_i$ is (ϵ, δ)-DP for each $i \in [1, k]$, then the parallel composition $M(x) = (\mathcal{M}_1(\mathcal{D}_1), \mathcal{M}_2(\mathcal{D}_2), \ldots, \mathcal{M}_k(\mathcal{D}_k))$ is $(\max_i \epsilon_i, \max_i \delta_i)$-DP.*

Post-processing ensures that any further processing of differentially private outputs remains differentially private, without additional privacy cost.

Theorem 3. *If $\mathcal{M}$ is (ϵ, δ)-DP and f is any function, then $\mathrm{f} \circ \mathcal{M}$ is also (ϵ, δ)-DP.*

Sparse Vector Technique (SVT) [18] 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 $\mathrm{Lap}(\frac{\Delta}{\epsilon_1})$ and added to a predetermined threshold. For each query, noise is sampled from a distribution $\mathrm{Lap}(\frac{2c\Delta}{\epsilon_2})$ 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 $\epsilon_1 : \epsilon_2 = 1 : (2c)^{2/3}$ [54].

Differentially Private Stochastic Gradient Decent (DP-SGD) [3]. 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:

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

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, $\mathrm{clip}(g_i(\theta_t), C)$ denotes the gradient clipping operation, and $\mathcal{N}(0, \sigma^2 I)$ represents the Gaussian noise with variance σ^2.

3 PROBLEM FORMALIZATION

3.1 Model Infrastructure Abstraction

The abstracted model infrastructure that we target in this work involves three major roles: (1) the *data owners* provide datasets used for training ML models and receive compensation for their data usage; (2) the *broker* trains a set of differentially private models, priced following the fairness rule (in addition to the arbitrage-free mechanism, which is orthogonal to our work); (3) and the *model buyers* purchase and run the model within the infrastructure by specifying purchased data or supplying external data. (The buyer can also download the model.) State-of-the-art approaches to deduplicating deep learning models [80] are block-based, storing each tensor as a collection of equal-sized blocks. Leveraging the robustness of deep learning model inference to small weight changes, these methods identify groups of similar blocks that can be replaced by a single physical block, thereby compressing storage. While existing works focus solely on the accuracy impact of deduplication [80], our work demonstrates that deduplicating privacy-preserving models also affects their privacy budgets. It's worth noting that the proposed deduplication algorithm is complementary to other neural network compression techniques, such as quantization [14, 19, 27, 62, 75] and pruning[11, 21, 53, 60, 66] (as we evaluated in Sec. 6.4).

Threat Model. We assume the attackers of the model infrastructure may perform re-identification attacks [5], membership inference attacks [33, 63], and model inversion attacks [24]. In addition, following the prior works [49], we assume that privacy policies and regulations effectively prohibit model buyers from colluding.

3.2 Storage and Serving Extensions

A model includes metadata information and parameter tensors. The metadata information includes pointers to the training and validation datasets, hyper-parameters used in the training process, such as privacy budget, training algorithm, learning rate, batch size, number of epochs, and training accuracy and validation accuracy. Depending on the model architecture, a model may contain many parameter tensors, e.g., embedding vectors, weight matrix at a fully connected layer, or a filter at a convolutional layer. In our implementation, each parameter tensor is flattened into a 1-dimensional array and then partitioned into blocks with fixed size S. The last block could have a smaller size than S, so it is padded with zeros to make the size equal to S to be deduplicable. Some parameter tensors, such as biases and layer norms, are much smaller than the block size, so they are not partitioned and managed separately.

After deduplication, the models are stored in three parts, and cached in memory (or warm storage, e.g., SSD, when memory becomes insufficient [80]). (1) A 2-dimensional array that contains the blocks for all models, where each row represents a block flattened into a 1-dimensional array. (2) A dictionary for each model containing tensor shapes and extra weights, such as biases and norms, which are much smaller than block size and are thus handled separately. (3) A list of model constitutions for each model. A model constitution is a list of block indices in the 2-dimensional array. For example, a Vision Transformer (ViT) model partitioned into 288 blocks would have a constitution comprising 288 integers. Each integer corresponds to a row index in the 2-dimensional array.

Serving a deduplicated model requires reconstructing the model from the blocks and the dictionaries. The process is accelerated by our array-based indexing for accessing blocks.

Whenever a new query requires a model that is not reconstructed, the system first checks if there is enough memory to reconstruct this model. If there is insufficient space, some models will be evicted to disk following the Least Recently Used (LRU) rule to free up memory space for reconstructing the new model. The system will then retrieve the model ID, look up the corresponding model constitution and extra weights, and reconstruct the model.

3.3 Problem Formulation

Assume there are n differentially private models trained on the same dataset or disjoint datasets, denoted as $M_1, \ldots, M_n$, satisfying $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$-DP respectively. (Note that with DP-SGD, the model privacy budget follows (ϵ, δ)-DP rather than ϵ-DP. In this paper, for simplicity, we present our problem and method in the context of ϵ-DP model privacy, which can be easily extended to (ϵ, δ)-DP by applying the same constraints and computations on δ as on ϵ. It is because δ follows the same composition rules as ϵ as shown in our Theorem 4 in the next section.) A model is partitioned into a set of fixed-size blocks. The ith model with r blocks can be represented by $M_i = \{b_{i1}, \cdots, b_{ir}\}$. The set of all blocks before deduplication is denoted as $B = \{b | b \in M_1 \cup ... \cup M_n\}$.

Intuitively, the ***block deduplication problem*** is to select a set of blocks from B to replace the (similar) blocks in each model to minimize the total number of unique blocks needed, while satisfying (1) constraint on M_i's privacy budget increase ($\Delta\epsilon_i$), (2) constraint on M_i's utility drop (Δu_i), and (3) the fairness rule ensuring models trained with lower privacy budgets on the same dataset must maintain lower accuracy than those trained with higher budgets. This complex problem can be characterized as a black-box combinatorial optimization problem [15], formalized as follows.

The problem is to transform $M_1, ..., M_n$ into distinct models $M'_1, ..., M'_n$, with $B' = \{b | b \in M'_1 \cup ... \cup M'_n\}$, $B' \subseteq B$, and M'_i and M_i have the same architecture, which means they have the same number of blocks, i.e., $|M'_i| = |M_i|$, for $i \in \{1, \ldots, n\}$, with the following objective and constraints:

$$\begin{aligned} &\text{minimize } |B'| && \text{s.t.} \\ &\Delta\epsilon_i < \epsilon_i^* \\ &\Delta u_i < t_i^* \\ u'_j < u'_k \iff \epsilon'_j < \epsilon'_k \quad &\text{if} \quad M_j \sim M_k \end{aligned}$$

for any $i, j, k \in \{1, \ldots, n\}$ and ϵ_i^* and t_i^* are the privacy cost increase threshold and utility drop threshold for the ith model, respectively. We name the last constraint as *fairness constraint/rule*, the utility u and privacy budget ϵ are evaluated on the same dataset and task. $M_j \sim M_k$ means M_j and M_k are two models trained on the same dataset for the same task using the same model architecture. We evaluate the accuracy of the original and deduplicated models on a validation dataset, called u_i and u'_i, to obtain $\Delta u_i \triangleq u_i - u'_i$. The derivation of the privacy budget ϵ'_i for a deduplicated model M'_i is explained in Sec. 4 . We have $\Delta\epsilon_i \triangleq \epsilon'_i - \epsilon_i$, which are defined on all datasets as explained in Sec. 4. In the case of (ϵ, δ)-DP, we simply add $\Delta\delta_i \triangleq \delta'_i - \delta_i$ and $\Delta\delta_i < \delta_i^*$ as well as $u'_j < u'_k \iff \delta'_j < \delta'_k$.

4 PRIVACY CHALLENGES

In this section, we derive the privacy budget for a deduplicated model M' from a model M, following a general deduplication procedure [80] to illustrate privacy challenges.

4.1 General Model Deduplication Procedure

Following the state-of-the-art accuracy-aware model deduplication technique [80], a general procedure of model deduplication is detailed in the following steps.

Step 1. To deduplicate a target model M_t, identify one or more base models to serve as potential block providers. The target model itself is also a block provider.

Step 2. Order all blocks by some saliency measures (e.g., the third quantile of weight values in the block [80]) ascendingly.

Step 3. For each block $b \in M_t$, run the following sub-steps: (3-1) Select a replacement block b' from the collection of blocks from all block providers. (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, a naive approach to reduce the validation frequency, called Greedy-N in this work.) (3-4) If the accuracy drop exceeds a threshold, undo the block replacement. (3-5) Move to the next block b in M_t.

4.2 Privacy Budget Derivation

The privacy budget of M'_t in the above deduplication process can be determined by Theorem 4. We first introduce the intuition behind Theorem 4. 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 R_t and $\hat{R}$ 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 ($M_t \neq \hat{M}$ & $R_t = \hat{R}$)**: 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 [17, 56]. (3) **Inter-Model&Inter-Data ($M_t \neq \hat{M}$ & $R_t \neq \hat{R}$)**: If b_t and $\hat{b}$ are from different models that were trained with different dataset,we have $\epsilon'_t = \max(\epsilon_1, \epsilon_2)$ by applying DP's parallel composition property [17, 56]. Next, we introduce Theorem 4.

Theorem 4. *Suppose that d disjoint training datasets are used for training all models. Let M_t be a (ϵ_t, δ_t)-DP model to be deduplicated, and let $\mathcal{M} = \{\hat{M}_1, ..., \hat{M}_k\}$ be a set of k distinct differentially private models, each satisfying (ϵ_i, δ_i)-DP, serving as potential block providers. Any two models in $\{M_t\} \cup \mathcal{M}$ are trained on either the same dataset or completely disjoint datasets from each other. Let $\mathcal{M}_j = \{\hat{M}_{j_1}, ...\hat{M}_{j_l}\}$ be the group of models trained on the dataset r_j. Then, on dataset r_j, the resulting deduplicated model, denoted as M'_t, satisfies $(\epsilon_t \cdot \mathbb{1}_{M_t, r_j} + \sum_{i=1}^{l} \epsilon_{j_i}, \quad \delta_t \cdot \mathbb{1}_{M_t, r_j} + \sum_{i=1}^{l} \delta_{j_i})$-DP, where the indicator function $\mathbb{1}_{m, r_j}$ is equal to 1 if model m is trained on r_j otherwise 0. If $\mathcal{M} = \emptyset$, then M'_t satisfies (ϵ_t, δ_t)-DP.*

Given any dataset r_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 $\cup_{j=1}^{d} r_j$.

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 $\mathcal{M}$ into groups by their training datasets. $\mathcal{M}_j = \{M_{j_1}, ...M_{j_l}\}$ is a group of models using training data r_j. By DP composability, the collection of blocks from this group of models satisfies $(\sum_{i=1}^{l} \epsilon_{j_i}, \sum_{i=1}^{l} \delta_{j_i})$-DP. Because the deduplication algorithm iterates over all blocks to decide replacements, by DP post-processing property, the derived model M'_t satisfies $(\sum_{i=1}^{l} \epsilon_{j_i}, \sum_{i=1}^{l} \delta_{j_i})$-DP on r_j when the original model M_t is not trained on r_j. If M_t is trained on r_j, M'_t satisfies $(\epsilon_t + \sum_{i=1}^{l} \epsilon_{j_i}, \delta_t + \sum_{i=1}^{l} \delta_{j_i})$-DP by DP sequential composition. Given any dataset r_j $(1 \le j \le d)$, 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 datasets $\cup_{j=1}^{d} r_j$, by the parallel composition property [56].

Privacy Challenges: According to the above analysis, the more base models that provide the blocks for deduplicating a target model, the higher the privacy budget increase will result for the target model. We assume the validation dataset is public in this and the next section. We will discuss the additional privacy cost caused by the private validation dataset in Sec. 6.2.3.

5 NOVEL AUTOMATIC DEDUPLICATION

5.1 Methodology Overview

Base Model Selection (Sec. 5.2). To satisfy the privacy budget constraint and address the privacy challenges discussed in Sec. 4, our key idea is to ensure that each model (M_t) will be deduplicated with *base model(s)* that satisfy privacy constraints and are similar to M_t, and the selection process should not rely on private information such as model weights to avoid additional aggregation of privacy costs. In addition, multiple models should share the same base model to improve the compression ratio. As such, we propose a base model selection strategy that clusters models based on public metadata information and selects base model(s) for each cluster to meet privacy constraints and similarity requirements. This idea is crucial to reduce the search space by reducing the multi-model deduplication to a series of two-model deduplications.

Deduplication with Saliency Analysis and Dynamic Validation (Sec. 5.3). While applying the state-of-the-art accuracy-aware deduplication approach (See Sec. 4.1) [80] to deduplicate a model with the base model, we identified two drawbacks: (1) Accuracy validation accounts for most of the latency of the deduplication process. In addition, it is hard to statically set a validation frequency (e.g., validating the deduplicated model accuracy after deduplicating every k blocks) to balance the compression ratio and the deduplication latency; (2) The state-of-the-art deduplication approach [80] orders the blocks by simple saliency measures and selects the block with the least saliency to be deduplicated at each iteration to minimize the accuracy impact. However, we find that the simple saliency measures such as weight magnitude [46] are sub-optimal, while many measurements widely utilized by explainable AI/ML research [50, 66] are under-explored for deduplication. We address the first drawback using a dynamic-range deduplication algorithm. To address the second drawback, we propose to leverage the weight gradient [23] as a saliency metric for balancing saliency profiling latency and effectiveness.

Sparse Vector Technique (SVT) for Validation Using Private Data (Sec. 5.4). Additional privacy costs will be incurred if the accuracy validation uses a private dataset. Usually, the more validations are performed, the higher the privacy budget will be required. To make a better trade-off, our main idea is to leverage the fact that accuracy validation is a boolean question about whether the accuracy drop is above a threshold and applies SVT to provide a fixed budget for a pre-specified number of failed validations (i.e., the boolean questions returns negative results). We also investigated the lower bound of the accuracy validation process if the validation dataset and the original model's training dataset are two independent partitions belonging to the same dataset to facilitate the configuration of the SVT parameters (i.e., the number of failed validations or cutoff in our case).

5.2 Base Model Selection

The nature of the optimization problem formalized in Sec. 3.3 is to balance the compression ratio (i.e., the optimization goal) and the constraints on privacy budget increases and utility drops. We are the first to identify the **key factor** that controls the complex trade-offs: **the selection of base models to duplicate a target model**. Here, the base models, as aforementioned, represent the source models that provide blocks for deduplicating the target model, and these models are selected from the models managed by the infrastructure. We first focus on the following questions.

Q1. How will a base model affect the privacy constraint? Using a base model $\hat{M}$ to deduplicate a target model M_t by transforming it to M'_t, the privacy budget change of M'_t will differ depending on whether the training data of $\hat{M}$ and M_t is the same. Using $R(\cdot)$ to represent the training dataset identifier of the input model, the privacy budget change of M'_t is $\epsilon'_t = \epsilon_t + \hat{\epsilon}$ if $R(M_t) = R(\hat{M})$, or $\epsilon'_t = \max(\epsilon_t, \hat{\epsilon})$, if $R(M_t) \neq R(\hat{M})$. $\Delta\epsilon_t \triangleq \epsilon'_t - \epsilon_t$ must be less than the privacy budget increase constraint ϵ^*_t.

Q2. How will a base model affect the compression ratio? Empirically, a base model containing more similar weights to the target model yields a better compression ratio. However, estimating the similarity between models based on their weights will aggregate their DP privacy costs, leading to a higher privacy budget for the resulting deduplicated models. Therefore, we need to use public model information, such as model architecture type, training dataset, and hyper-parameters, e.g., ϵ, to locate similar models.

Q3. How will a base model affect the utility? To maintain a high utility for a target model after deduplication, it usually desires base models with relatively high privacy budgets on the same training dataset since such models offer better utility [3] and thus provide higher-quality blocks for deduplication.

5.2.1 Local Strategy for Base Model Selection. We first focus on deduplicating one model M_t. The local strategy aims to iteratively select base models for M_t. Each time, it will select one base model $\hat{M}$ for deduplicating M_t with privacy budget ϵ_t and privacy budget increase constraint ϵ^*. We first identify the models that satisfy two criteria: (1) The base model should be the most similar to the target model according to the model's public information (Q2); (2) Its privacy budget $\hat{\epsilon}$ satisfies the following constraints based on Q1:

$$\begin{cases} \hat{\epsilon} < \epsilon^* & \text{if} \quad R(M) = R(\hat{M}_t) \\ \hat{\epsilon} < \epsilon_t + \epsilon^* & \text{if} \quad R(M_t) \neq R(\hat{M}) \end{cases} \tag{1}$$

Then, we query top-k candidates that minimize the distance$(M_t, \hat{M})$ (according to Q2) and select the candidate with the highest utility (according to Q3). The distance function uses Euclidean distance of the embedding vectors of models' public attributes, such as model architecture, training data, task, and privacy budget, without introducing aggregated privacy costs. Assuming that the currently selected base model $\hat{M}_c$ has a privacy budget of $\hat{\epsilon}_c$, the local strategy will update ϵ^* using Eq. 2 after deduplicating $\hat{M}_c$ and M_t. Then it will repeat the process until no qualified base model can be found.

$$\begin{cases} \epsilon^* \leftarrow \epsilon^* - \hat{\epsilon}_c & \text{if} \quad R(\hat{M}_c) = R(M_t) \\ \epsilon^* \leftarrow \epsilon^* - (\hat{\epsilon}_c - \epsilon_t) & \text{if} \quad R(\hat{M}_c) \neq R(M_t) \ \ \& \ \ \hat{\epsilon}_c > \epsilon_t \\ \epsilon^* \leftarrow \epsilon^* & \text{if otherwise} \end{cases} \tag{2}$$

5.2.2 Global Strategy for Base Model Selection. We next generalize the local strategy to deduplicate all models in the infrastructure. An effective heuristic to improve the overall compression ratio of all-model deduplication is to increase the number of target models that use each base model and the similarity between each target model and its base model. In addition, different from local deduplication, global deduplication introduces an *ordering constraint* that a model must be deduplicated (as a target model) before it is used to deduplicate other models (as a base model), since if otherwise, the weights of a deduplicated model may be updated by the deduplication of its base models, leading to non-deterministic utility.

To address the issues, we first leverage public model information, usually available in the model marketplaces or model stores [1], to cluster similar models. For each cluster, it selects base models to be shared by all models in the cluster. We adopt a hierarchical clustering algorithm [59] that groups models using the distance function described in Sec. 5.2.1, which can be easily extended to adopt alternative clustering algorithms and distance functions.

After the clustering, extending the local strategy, given a cluster of target models $\mathcal{G} = \{M_1, ..., M_k\}$, with model M_i's privacy budget being ϵ_i and its privacy budget increase constraint being ϵ_i^*, we first try to search for a base model $\hat{M}$ from $\mathcal{G}$, which has the highest privacy budget $\hat{\epsilon}$ that satisfies the constraints in Eq. 1 for each model $M_i \in \mathcal{G} - \hat{M}$. If no qualified base model exists in the current $\mathcal{G}$, we will search candidate base models in other groups $\mathcal{G}_n$ ordered by their distances to $\mathcal{G}$. If no qualified base models can be found, we move the model(s) with the least ϵ^* and the least $\epsilon^* + \epsilon_i$ from $\mathcal{G}$ into a special group (for models that failed to find qualified base models), to relax the privacy constraint of a qualified base model for $\mathcal{G}$. This process repeats until $\mathcal{G}$ has a qualified base model or becomes unsplittable. The rest of the base model selection strategy for each *group* is similar to the local strategy, except that cyclic dependency, e.g., two groups provide base models for each other, is forbidden to avoid complicating the ordering constraint.

Based on the global strategy, we reduce the optimization problem formalized in Sec. 3.3 into separate two-model deduplication tasks formalized in Sec. 5.2.3 to help meet the ordering constraint and the fairness constraint.

5.2.3 Deduplication Problem Reduction. The two-model deduplication involves a target model $M_i \in \mathcal{G}$ and a base model $\hat{M}$ selected for $\mathcal{G}$ (i.e., the privacy constraint is guaranteed by base model selection). The optimization objective is to minimize the number of unique blocks in M_i' that are not in $\hat{M}$ (i.e., minimize the compression ratio) while meeting the utility drop constraint formalized below.

$$\text{minimize } |\text{UniqueBlocks}(M_i' - \hat{M})| \text{ with}$$
$$M_i' = \text{Deduplicate}(M_i; \hat{M} \text{ as base model}) \ s.t.$$
$$u_i - u_i' < T_i$$

$|\text{UniqueBlocks}(M_i' - \hat{M})|$ represents the set difference of two sets of blocks in M_i' and $\hat{M}$. T_i is derived from the user-specified constraint of utility drop t_i^* (see Sec. 3.3) to enforce $u_i - u_i' < t_i^*$ and the fairness constraint. We sort these models governed by the same fairness constraint (i.e., models trained on the same dataset with the same model architecture for the same task) according to the beginning of their utility ranges $[u_i - t_i^*, u_i]$ in ascending order. Then, we deduplicate the models sequentially, setting the threshold T_i based on the previous deduplicated model's utility (u_{i-1}'), following Eq. 3.

$$T_i = \min(u_i - u_{i-1}', t_i^*) \text{ for } M_i \tag{3}$$

To meet the ordering constraint, we schedule these tasks following a total ordering identified by applying the topology sort algorithm [39] on the nodes of a directed acyclic graph, of which nodes represent models and directed edges represent base-model and target-model relationships assigned by the global strategy.

5.3 Deduplication Algorithms

5.3.1 Problem Analysis. Following the classical deduplication framework presented in Sec. 4.1 to resolve the two-model deduplication problem in Sec. 5.2.3, the order of blocks being deduplicated determines the final model and its utility. Given N blocks, there are $N!$ candidate deduplication plans. Depending on the block size, deep learning models in our experiments may have hundreds to thousands of blocks. Therefore, it is computationally expensive to exhaustively search and evaluate the storage benefit and accuracy drop of all deduplication plans.

The classical deduplication algorithm [80] performs validation after deduplicating every k blocks. When k is set to a small number, the frequent utility validation operations will bottleneck the deduplication process. However, if we set k to a large number, it often deduplicates too many blocks to keep the utility drop within the constraint T_i and triggers an undo of all past k deduplications, which may lead to missed opportunities for deduplication.

To address the validation bottleneck, it is important to (1) validate only after deduplicating multiple blocks in a batch to reduce the validation frequency; and (2) avoid mixing a few salient blocks, which are the blocks that will significantly impact the model's utility, with many non-salient blocks that will not lead to significant model utility drop if deduplicated alone. That's because 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 missed opportunities for deduplicating non-salient blocks, in addition to wasting time in deduplication, validation, and rollback of the batch. However, it is challenging to identify salient blocks because the saliency depends not only on the weight values in the block and the positions of the block in the model, but also on the quality of the block used to duplicate it.

5.3.2 Our Solution. Our novel solution addresses the problems based on two ideas: (1) Some salient blocks should not be deduplicated because they significantly affect the model's utility. The ordering of the blocks should ensure that the less salient blocks (i.e., blocks that are less influential to the model utility) are deduplicated as early as possible, which will help reduce the number of validation steps in Algorithm 1 and the number of validation failures in Algorithm 2. (2) Instead of deduplicating a constant of N blocks at a time, we dynamically adjust the size of each group to reduce the chance that the utility validation of the model after deduplicating the group of blocks will drop below the threshold. For example, at the beginning, a large group of the least salient blocks should be deduplicated together, which usually leads to successful utility validation with reduced frequency. As we progress through the ordered list, the group size gradually decreases, ensuring that non-salient blocks are checked in fine-grained manner.

Block saliency and similarity measures. Weights significantly impacting the prediction are called "salient weights" [34, 66]. There are many ways to measure the saliency of weights in the research literature on model explanability [64], pruning [44, 46], and quantization [22, 23]. The most popular ones include the weight magnitude [80], activation magnitude [48], and the Fisher information that involves the square of gradients [50]. We find that the Fisher information metric is the most accurate among them. However, it is also the most computationally intensive. In addition, the gradients are usually very small for large models, and their squares are even smaller, which sometimes introduce unstable numerical results. Therefore, we introduced and 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 an independent partition of the same dataset to which the training and the validation datasets belong. In the latter case, this one-time privacy cost is usually smaller than other privacy costs (e.g., models' privacy budget or the privacy costs of accuracy validations) and thus gets absorbed due to DP's parallel composition rule.
Given a block b chosen by the saliency measure, a block from the base model most similar to block b should be chosen to replace b to maximize the utility. To measure similarity, we compared several distance measures, such as `l1-norm`, `l2-norm`, and `cosine` and chose to adopt `l2-norm`. While we can leverage techniques such as locality-sensitive hashing [13, 55] and product quantization [38] to further reduce the nearest neighbor search overhead, it is not the bottleneck of the end-to-end deduplication scenario.

Group deduplication algorithms. As discussed earlier, instead of deduplicating a constant of k blocks at a time followed by utility validation, we propose dynamically adjusting the group size. To achieve that, 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 dynamic-range deduplication (`DRD`) algorithm , as formalized in Algorithm 1, recursively deduplicating the left and right halves of the input range. `DRD` begins by dividing the input range of blocks into two halves. It then attempts to deduplicate the left half. If this deduplication fails (i.e., the utility drop exceeds the predetermined threshold), the algorithm recursively calls itself to

Algorithm 1 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 to be deduplicated 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 ;
```

deduplicate the left half further (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, reducing the deduplication batch size by half on each recursive call (Line 15). The base case for this recursion occurs when the input range has fewer than L blocks, and it skips such a small remaining range and immediately returns without deduplicating any blocks, because such remaining blocks are likely to be salient blocks.

We also proposed a variant of `DRD`, called Dynamic-Range-Expansion Deduplication (`DRED`), slightly different from `DRD` in Line 17. `DRED` will recursively deduplicate not only the blocks from $m+1$ to r, but also $r+1$ to $|B|-1$, so that line 17 is changed to Recursion($B, \hat{B}, m+1, |B|-1, T$). It allows for more aggressive deduplication attempts.

5.4 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 Algorithm 2. It satisfies ϵ-DP [54]. 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.

Algorithm 2 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 $\epsilon_1 : \epsilon_2 = 1 : (2c)^{2/3}$

Algorithm 2 ϵ-DP Deduplicate with SVT

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

as proposed by Lyu et al. [54]. Cut-off c represents the maximum number of times the deduplication process can fail the utility test (i.e., validation), which is pre-specified. Let $\hat{\epsilon}$ and ϵ_t denote the privacy budgets of the base model and target model, respectively. The total privacy cost for the deduplicated model is then calculated as $\max(\hat{\epsilon} + \epsilon_t, \epsilon_1 + \epsilon_2)$ if $R(\hat{M}) = R(M_t)$ or $\max(\hat{\epsilon}, \epsilon_t, \epsilon_1 + \epsilon_2)$ if $R(\hat{M}) \neq R(M_t)$.

A high noise level can result in inaccurate validation, leading to underperforming of the deduplicated models. To mitigate this issue, we propose to increase $|D|$. This rationale justifies our adoption of a full validation set to ensure a large $|D|$, as opposed to the classical model deduplication framework [80] that uses a sampled validation set to reduce validation latency. In addition, the deduplication algorithm proposed in Sec. 5 will effectively deduplicate most of the non-salient blocks with few validation failures, which allows for setting a small c value to reduce the noise level (see Algorithm 2 Line 3 and 12). These remedies effectively alleviate the problem of noisy comparisons in the SVT, according to our empirical observation.

6 EXPERIMENTS

6.1 Experimental Settings

6.1.1 Workloads. To demonstrate the broad applicability of our proposed approaches, we experimented with a diverse range of model architectures and tasks:
1. Roberta-Base [51] (called Roberta): A Transformer encoder for natural language inference tasks, composed of embedding layers, attention layers, layer normalization layers, and linear layers.
2. Vision Transformer-Large (ViT) [16]: A Transformer model for image classification tasks, consisting of attention layers, layer normalization layers, and linear layers.
3. ResNet152 [31]: A convolutional neural network with residual connections for multi-attribute classification tasks, primarily composed of CNN layers and batch normalization layers.

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 *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 [9, 25, 26, 58, 69, 78], [0.08,4.6] for image and tabular data classification tasks [6], and [1,16] for image processing tasks [36]. 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 experiments involved 15 clusters of models that have different numbers of models (i.e., ranging from four to 20 models), model architectures, training datasets, distribution of epsilons, privacy budget increase threshold ϵ^*, and user-specified utility drop threshold t^*. (The latter two constraints are shared by all models in each cluster in our experiments for simplicity.) These model groups are shown in Tab. 1, designed to be diverse and representative.

Table 1: The Deduplication Scenarios

	Model	Data	Epsilons	ϵ^*	t^*
Five diverse clusters of models for Fig. 2, Tab. 2 to Tab. 7					
A1	Roberta	QNLI [71]	[1.0, 2.0, 4.0, 6.0, 7.0]	1.0	0.015
A2	Roberta	SST2 [65]	[0.3, 0.4, 0.6, 0.8, 1.0]	0.05	0.015
A3	Roberta	5 MNLI [72] parts	[0.2, 0.4, 0.8, 1.6, 2.0]	0.2	0.015
A4	ViT	CIFAR100 [42]	[0.5, 0.6, 0.75, 1.0, 2.0]	0.5	0.020
A5	ResNet	CelebA [52]	[0.4, 0.6, 0.8, 1.0, 2.0]	0.5	0.020
Three large clusters of models for latency comparison (Fig. 3)					
B1	Roberta	QNLI	[0.8, 0.9, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0]	0.8	0.015
B2	ViT	CIFAR100	[0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 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, 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
Three clusters of models for base model selection comparison (Fig. 4)					
C1	Roberta	QNLI	[6.0, 7.0, 8.0, 9.0]	5.0	0.015
C2	ViT	CIFAR100	[0.6, 0.75, 1.0, 2.0]	0.5	0.020
C3	ResNet	CelebA	[0.8, 0.9, 1.0, 2.0]	0.7	0.020
C4	Roberta	QNLI	[5.0, 6.0, 7.0, 8.0]	10.0	0.015
C5	Roberta	4 MNLI parts	[0.4, 0.8, 1.6, 2.0]	0.3	0.015
Two large clusters of models for SVT-based validation(Fig. 5)					
D1	Roberta	QNLI	[0.8, 0.9, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0]	0.8	0.015
D2	ResNet	CelebA	[0.2, 0.5, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0]	0.2	0.020

6.1.2 Comparison. None of the existing model deduplication mechanisms has considered privacy. Therefore, we considered the following algorithms, which used our base model selection mechanism to ensure privacy. They also used consistent saliency and similarity measurements, i.e., gradient magnitude and L2 similarity:
1. Our Dynamic-Range Deduplication (**DRD**) as formalized in Alg. 1 and its variant, Dynamic-Range-Expansion Deduplication (**DRED**).
2. The **Greedy-N** algorithm as described in Sec. 4.1;
3. The **Monte Carlo Tree Search (MCTS)** [10] algorithm, where each state represents the unique block mapping for each model, and an action selects a group of N target blocks to be deduplicated and then use the most similar block from the base model to replace each target block. However, we find this naive MCTS performed poorly due to the explosive search space. Therefore, we improved the block selection leveraging the gradient magnitude as a heuristic, so that a less salient block has higher probability to be selected.

In addition, we also considered two reference baselines.
1. **Original**, which does not perform any deduplication.
2. **Retrain**, which does not perform any deduplication, but 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.

6.1.3 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 the group of original models and $M'_1, ..., M'_n$ represent their deduplicated counterparts. The group compression ratio is computed as $|B'|/|B|$, where $B = \{b|b \in M_1 \cup ... \cup M_n\}$ and $B' = \{b|b \in M'_1 \cup ... \cup M'_n\}$. For an individual model M_t and its deduplicated version M'_t, we calculate the compression ratio as $|\text{UniqueBlocks}(M'_t - \hat{M})|/|B_t|$. Here, $\hat{B}$ represents the blocks of the base model, B_t denotes the blocks of the original target model, and B'_t signifies the blocks of the deduplicated target model.

Accuracy. We measure the accuracy of individual models and the accuracy drop for groups of models. The latter is defined as 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 epsilon and delta. This paper focuses on meeting the constraints on epsilon, 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.

6.1.4 Experimental Environments. Most of the experiments are ran on a machine with an Intel(R) Xeon(R) Silver 4310 CPU (2.10Hz) with 128 GB memory and one NVIDIA A10 GPU with 24GB GPU memory, except that the model loading and inference latency comparison is conducted in an AWS c5a.xlarge instance (7.63 GB main memory, 4vCPU) to simulate a resource-constrained environment.

6.2 Evaluation Results

6.2.1 Overall Results. Fig. 2 illustrated each model's **compression ratio** and **accuracy** after we applied our proposed approaches (i.e., DRD and DRED coupled with base model selection) to each cluster of models shown in Tab. 1. The first model is selected as the base model for A1, A3, A4, and A5. For A2, where no models qualified as the base, we chose an external model (the $\epsilon = 0.2$ model from A3) as the base. As illustrated in Fig. 2, our DRD methodology achieved overall compression ratios of 29.8%, 33.3%, 23.9%, 34.1%, and 38.0% for clusters A1 to A5, respectively. DRED achieved compression ratios of 31.7%, 35.6%, 24.4%, 33.4%, and 36.6% for the same clusters. Using DRD, the accuracy drops of deduplicated models compared to the original models (with a smaller ϵ) ranges from 0.45% to 1.14%, 0.0% to 0.69%, 0.0% to 1.38%, 1.23% to 1.87%, and 0.76% to 1.82% for models in A1 to A5 respectively. For DRED, the accuracy drops were 0.31% to 1.14%, 0.11% to 1.15%, 0.0% to 1.38%, 1.33% to 1.87%, and 0.81% to 1.82% correspondingly. In addition, from A1, A4, and A5, we observed that the models whose ϵ values are closer to the ϵ of the base model tend to achieve a better compression ratio and accuracy balance. Retrain achieved the best accuracy since its privacy budget equals DRD/DRED, but it does not perform any deduplication.

As aforementioned, most of the **deduplication overhead** 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, 84.7seconds for ViT on CIFAR100, and 84.8 seconds for ResNet on CelebA. We thus compared the efficiency and the effectiveness of various deduplication mechanisms, such as MCTS, Greedy-N, DRD, and DRED, for deduplicating A1, A2, and A3 (Tab. 1). The results, presented in Tab. 2, reveal that MCTS requires numerous validation steps yet fails to achieve a low compression ratio. For Greedy-N algorithms, we observed that when N is small, the number of validation steps is exceptionally high, while larger N values result in relatively high compression ratios. In contrast, DRD and DRED balance both objectives effectively. These findings highlight the advantages of our proposed dynamic approaches in achieving good trade-offs between computational efficiency and model compression.

As illustrated in Fig. 3, we compared the **model loading and inference latency** with and without applying our DRD approach for three scenarios, which served 10 Roberta, 10 ViT, and 20 ResNet models, detailed in Tab. 1, in a resource-constrained environment, AWS c5a.xlarge instance, as described in Sec. 6.1.4. These three scenarios achieved a compression ratio of 26.7%, 26.4%, and 47.2%, respectively. For this experiment, we measured and broken down the overall latency into loading (including reconstruction) and inference latency for serving 100 inference queries for each cluster. In the **round-robin** serving approach, each query runs an inference using the next model in the cluster in a round-robin style. In the **random** serving approach, each query runs an inference using a model randomly sampled from the cluster. The results showed that our approach achieved 4.2× to 4.33× speedup of the overall latency for serving 10 Roberta models, 5.6× to 6.1× for serving 10 ViT models, and 1.8× to 2.2× for serving 20 Resnet models. Fig. 3 demonstrated that by deduplicating the models' weights, the overall memory footprint required to serve multiple models can be significantly reduced, which brings lower I/O overheads. Taking ViT models as an example, our targeting environment can only cache two ViT models in memory to be served simultaneously without deduplication. Each time a query requires a different model, a model currently in memory must be evicted to make room for the new model. However, all ten models can fit into the memory by applying our DRD approach. When a different model is requested, it can be quickly reconstructed using the weight blocks cached in memory.

6.2.2 Base Model Selection. We investigated how the selection of different base models will affect the compression ratio and number of validations (i.e., deduplication efficiency) in three different situations. **(1) Base model selection for clusters of models trained on the same dataset with different privacy budgets.** We used the three clusters C1, C2, and C3, described in Tab. 1 for evaluating this situation. For each cluster, we first compared the overall compression ratio achieved and the number of validations required in deduplicating all models from this cluster using qualified base models with different ϵ specified in the x-axis. As presented in the top three plots in Fig. 4, when the base model's epsilon increased, both the compression ratio and the number of validations required improved (i.e., decreased), which is expected since the utility of the base model and the similarity to the target models (because of closer privacy budget) improved. **(2) Using multiple base models for a group.** We used one to four external models to deduplicate the models in Cluster C4. As shown in the bottom left plot, using one model with $\epsilon = 4$ as the base model achieves a better compression

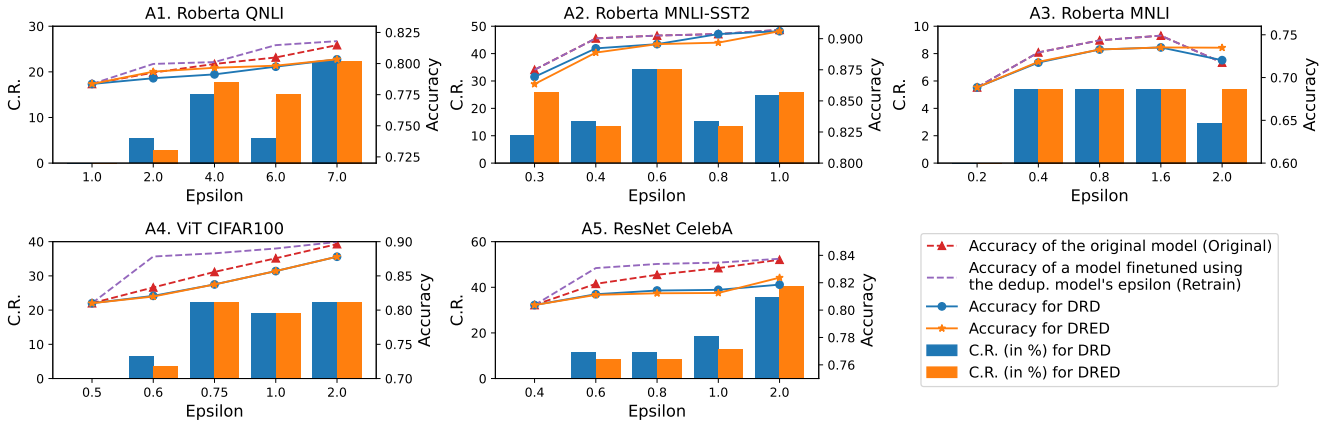

Figure 2: Overall Effectiveness for individual models in A1 to A5. The compression ratio (C.R.) of the base models, which are used as a block provider, is not affected during the deduplication and thus not shown in the figure. A2's base model is the first model in A3.

Table 2: Comparison of Different Deduplication Algorithms

	A1. Roberta-base for QNLI			A4. ViT-large for CIFAR 100			A5. ResNet152 for CelebA		
	C.R.(%)	Num. of Val.	Max. acc. drop	C.R.(%)	Num. of Val.	Max. acc. drop	C.R.(%)	Num. of Val.	Max. acc. drop
MCTS-20blocks	39.9	257	0.002	33.4	909	0.019	35.9	743	0.020
MCTS-30blocks	41.4	107	0.002	32.0	571	0.020	42.2	447	0.018
Greedy-1	**21.3**	860	0.015	**29.4**	1156	0.019	**28.9**	956	0.020
Greedy-10	24.6	92	0.008	29.9	120	0.019	29.9	100	0.018
Greedy-20	27.5	48	0.007	30.0	64	0.019	30.8	52	0.018
Greedy-30	23.1	36	0.008	32.0	44	0.019	34.4	36	0.015
Greedy-40	28.9	**28**	0.007	34.2	**36**	0.017	34.4	**28**	0.015
DRD	26.5	**29**	0.010	**32.8**	**24**	0.019	**32.3**	**27**	0.017
DRED	**25.9**	**29**	0.014	**32.8**	31	0.019	**32.5**	31	0.017

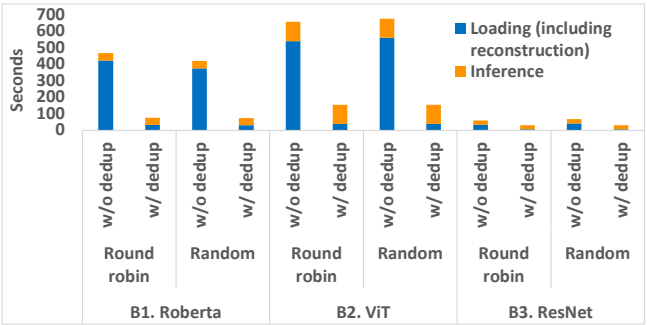

Figure 3: Latency breakdown of serving 100 inference queries involving multiple models w/o and w/ deduplication (DRD is used)

ratio for C4 than using three or four base models, which showed that the quality of base models (e.g., utility and similarity to the target models) is more important than quantity. **(3) Base model selection for clusters of models trained on different datasets.** For Cluster C5 that has multiple models trained on independent MNLI partitions, we found using a base model trained on a similar dataset (e.g., MNLI) outperformed other candidate base models.

6.2.3 SVT for Private Data Validation. We next investigated how using the Sparse Vector Technique (SVT) for model evaluation on private 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 evaluated the impacts using two 10-model clusters: D1 and D2, (See Tab. 1). The results illustrated in Fig. 5 demonstrate that SVT usually worsens the compression ratio by up to 9.9% and in certain cases it improves the compression ratio by up to 3.2% (due to randomness introduced by the noisy utility drop comparison). We extended our analysis by applying the SVT to smaller clusters A1, A4, and A5 described in Tab. 1, with results presented 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 worsens the compression ratio by 18.3% to 35.6%. However, DRD and DRED are more robust to SVT's cutoff on the validation failures, because of their dynamic range selection, with the compression ratio worsened by 7.6% to 23.9%.

6.3 Ablation Studies

6.3.1 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 the deduplication of cluster A4, which consists of five ViT models, as an example, the results showed that smaller block sizes lead to more blocks per model, thereby increasing the number of validation steps. These smaller blocks resulted in lower compression ratios and higher accuracy drops. However, when the block size increases

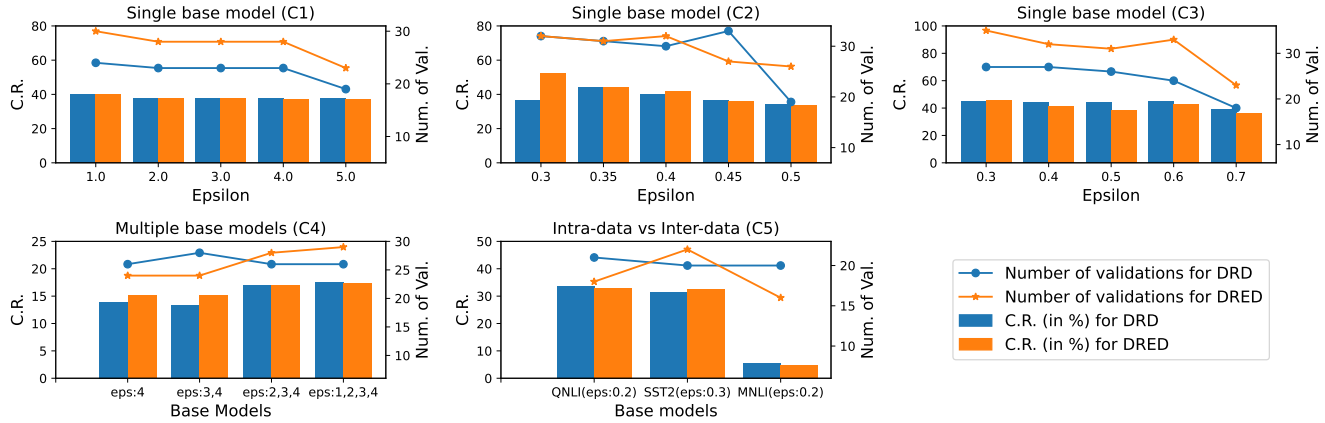

Figure 4: Comparison of Base Model Selection in Different Scenarios.

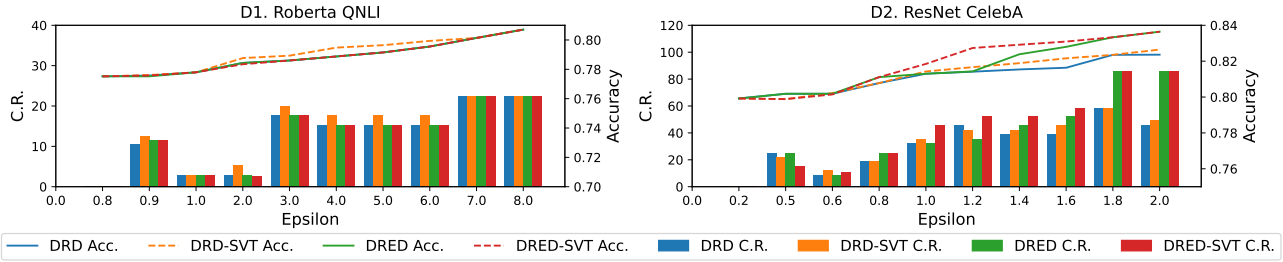

Figure 5: Deduplication using privacy data with SVT for 10 Roberta and 10 ResNet models. The compression ratio (C.R.) of the base models, which are used as a block provider, is not affected during the deduplication and thus not shown in the figure.

Table 3: Comparison of Different Deduplication Algorithms With SVT.

	A1. Roberta-base for QNLI			A4. ViT-large for CIFAR100			A5. ResNet152 for CelebA		
	C.R.(%)	Num. of Val.	Max. acc. drop	C.R.(%)	Num. of Val.	Max. acc. drop	C.R.(%)	Num. of Val.	Max. acc. drop
Greedy-10	29.6	85	0.008	37.2	100	0.016	43.7	98	0.014
Greedy-20	**28.9**	44	0.005	35.9	56	0.015	42.1	47	0.014
Greedy-30	**28.9**	32	0.003	42.8	38	0.014	50.0	32	0.010
Greedy-40	34.7	**24**	0.003	**34.8**	**24**	0.016	**40.3**	**24**	0.014
DRD	33.6	**24**	0.012	37.8	24	0.016	**38.1**	**21**	0.017
DRED	**32.1**	28	0.012	**35.3**	**21**	0.018	40.4	24	0.016

to a point, the compression ratio worsens because some layers become smaller than the block size. These smaller layers are then stored separately and excluded from the deduplication process.

We further compared the effectiveness of different saliency measures, the saliency aggregation methods, and block similarity measurements, using the deduplication of the cluster A1, which involves five Roberta-base models (see Tab. 1), as an example. The results in Tab. 4 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, 6576 seconds, and 1060 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. 4. For measuring the similarity of two blocks, as shown in Tab. 4, l2 distance outperformed other metrics.

Table 4: Ablation Study of Deduplication Hyper-Parameters

	Values	C.R.(%)	#Val.	Max. acc. drop
A4. block size	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	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 aggregate	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. distance	l1-distance	25.9	29	0.014
	l2-distance	40.2	31	0.012
	cosine	92.1	63	0.014

6.3.2 Impact of Accuracy Threshold. We used various accuracy drop thresholds to deduplicate A1, A4, and A5 in Tab. 1, and recorded the compression ratios shown as percentages in Tab. 5. 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 higher accuracy than the original. When the threshold was set to -0.5%, no blocks could be deduplicated from the clusters of A4 and A5. However, for A1, a few blocks could be deduplicated to improve the accuracy by 0.5%.

Table 5: 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

6.3.3 Impacts of SVT hyper-parameter tuning. We also investigated the impacts of SVT parameters, cut-off c and the validation privacy budgets ϵ. We set $\epsilon = \epsilon_b + \epsilon_t$, where ϵ_b and ϵ_t represent the privacy costs of the base and target models, respectively. This way, due to the DP parallel composition rule, the aggregated privacy budget will be $\max(\epsilon, \epsilon_b + \epsilon_t) = \epsilon_b + \epsilon_t$. Taking the deduplication of cluster A1 (with the first model as the base model) as an example, we varied the value of c from 2 to 7, and recorded the compression ratio, the maximum accuracy drop, and the number of validations in Tab. 6. 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. We also examined the effect of the validation privacy budget. Keeping the cut-off $c = 3$ constant, we varied the extra privacy budget $\epsilon_{extra} \triangleq \epsilon - \epsilon_b - \epsilon_t$ from 0 to 6. Interestingly, we observed no changes in the compression ratio (31.7%), maximum accuracy drop (1.14%), or number of validations (23). This suggests that $\epsilon_b - \epsilon_t$ is sufficiently large to reduce the noise level, preventing the noisy utility-threshold comparison from causing catastrophic failures. These findings indicate that when privacy is the primary concern, allocating additional privacy budget for private data validation using SVT may not be necessary.

Table 6: Impact 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. acc. drop(%)	0.66	1.14	1.14	1.14	1.14	1.14
Num. of val.	18	23	26	28	28	28

6.4 Compatibility With Pruning/Quantization

Pruning and quantization are two popular technologies to compress deep learning models. They are orthogonal to our proposed deduplication algorithm. As shown in Tab. 7, taking the A5 cluster (involving five ResNet152 models) as an example, combining pruning or quantization with deduplication will achieve significantly better compression ratio. In this experiment, we used the L1Unstructured method from the Pytorch's prune library to implement the pruning, and use linear quantization to quantize every floating point to an 8-bit integer.

We first quantize/prune the models and then deduplicate them, for two reasons. First, after deduplication, some weights share the same blocks. These shared weights must change at the same time and such special kind of structure is usually not implemented in off-the-shelf pruning and quantization algorithms. Second, if deduplication is run first, every weight change due to the pruning and quantization affects all other models, which makes the pruning and quantization extremely difficult. Thus we leave the study of deduplicate-then-prune/quantize to future work.

Table 7: Compatiblity w/ Pruning And Quantization for Cluster A5

	prune	quant	dedup	prune+dedup	quant+dedup
acc. drop	0.003	0.015	0.008	0.011	0.020
C.R.(%)	18.2	25.0	8.6	18.0	7.7

7 RELATED WORKS

Mistique [70] proposed a data store for managing the intermediate data generated from ML models. However, their deduplication techniques do not consider the accuracy, privacy, and latency requirements of ML workloads. Weight virtualization [45] proposed to merge pages across multiple models into a single page. However, their work did not consider privacy. Deduplication of relational data in RDBMS, also known as record linkage, identifies duplicate items through entity matching [20], using various blocking techniques to avoid the pair-wise comparison for dissimilar items [4, 7, 8, 12, 32, 40, 41]. In addition, various techniques leveraged similarity functions to filter out pairs that have similarity scores below a threshold [74] or used LSH to convert similarity join to an equi-join problem [77]. However, these works are not applicable to numerical tensor data, and they never considered how the deduplication of tensor data will affect the privacy of ML applications. Recently, a trend of fine-grained DP-based privacy control has emerged at the individual level [79], which justifies having many versions of models trained on the same dataset with different differential privacy budgets.

8 CONCLUSIONS

This work started a line of research on privacy-centric redesign of data management techniques with a focus on novel privacy-preserving data deduplication methods to compress multiple models and alleviate the inference and storage costs of emerging model management infrastructures such as model marketplaces, model hubs, and model serving platforms. The novel techniques include (1) Using model clustering and base model selection strategy to reduce the optimization problem with huge search space into a series of two-model deduplication problems, each with a significantly smaller search space. (2) Designing a new saliency measurement based on gradient magnitude and developing novel deduplication strategies (DRD and DRED) that deduplicate a dynamic number of blocks each time to balance the validation frequency and compression ratio. (3) Leveraging SVT to allow accuracy validation on customers' private datasets with optimized privacy costs. We also conducted a comprehensive evaluation of these techniques using diverse and representative clusters of models.

In the future, this work will be extended to deduplicate model checkpoints and the intermediate data (e.g., feature maps) generated during the model training/inference processes to accelerate privacy-preserving model explanation and debugging queries.

REFERENCES

[1] [n.d.]. Tensorflow Hub. "https://www.tensorflow.org/hub".

[2] [n.d.]. universal-sentence-encoder-large. ([n. d.]). https://tfhub.dev/google/universal-sentence-encoder-large/5.

[3] 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.

[4] 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.

[5] 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.

[6] 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*.

[7] 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.

[8] 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.

[9] 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*.

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

[11] Beidi Chen, Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Re. 2022. Pixelated Butterfly: Simple and Efficient Sparse training for Neural Network Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Nfl-iXa-y7R

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

[13] 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.

[14] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems* 35 (2022), 30318–30332.

[15] 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.

[16] 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

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

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

[19] Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. 2024. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118* (2024).

[20] 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.

[21] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJl-b3RcF7

[22] 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.

[23] 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).

[24] 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.

[25] 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).

[26] 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.

[27] Yi Guo, Fanliu Kong, Xiaoyang Li, Hui Li, Wei Chen, Xiaogang Tian, Jinping Cai, Yang Zhang, and Shouda Liu. 2024. decoupleQ: Towards 2-bit Post-Training Uniform Quantization via decoupling Parameters into Integer and Floating Points. *arXiv preprint arXiv:2404.12759* (2024).

[28] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. 2022. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence* 45, 1 (2022), 87–110.

[29] Vincent Hanke, Tom Blanchard, Franziska Boenisch, Iyiola Emmanuel Olatunji, Michael Backes, and Adam Dziedzic. [n.d.]. Open LLMs are Necessary for Private Adaptations and Outperform their Closed Alternatives. In *ICML 2024 Workshop on Foundation Models in the Wild*.

[30] 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).

[31] 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.

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

[33] 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.

[34] 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.

[35] Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. 2018. Serving deep learning models in a serverless platform. In *2018 IEEE International conference on cloud engineering (IC2E)*. IEEE, 257–262.

[36] 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.

[37] Dimitrije Jankov, Shangyu Luo, Binhang Yuan, Zhuhua Cai, Jia Zou, Chris Jermaine, and Zekai J Gao. 2019. Declarative recursive computation on an RDBMS: or, why you should use a database for distributed machine learning. *Proceedings of the VLDB Endowment* 12, 7 (2019), 822–835.

[38] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[39] Donald Ervin Knuth. 1997. *The art of computer programming*. Vol. 3. Pearson Education.

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

[41] 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.

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

[43] Abhishek Kumar, Benjamin Finley, Tristan Braud, Sasu Tarkoma, and Pan Hui. 2020. Marketplace for AI models. *arXiv preprint arXiv:2003.01593* (2020).

[44] 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

[45] 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.

[46] 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

[47] 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.

[48] 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*.

[49] 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).

[50] 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.

[51] 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).

[52] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[53] Miao Lu, Xiaolong Luo, Tianlong Chen, Wuyang Chen, Dong Liu, and Zhangyang Wang. 2022. Learning Pruning-Friendly Networks via Frank-Wolfe: One-Shot, Any-Sparsity, And No Retraining. In *International Conference on Learning Representations*. https://openreview.net/forum?id=O1DEtITim__

[54] Min Lyu, Dong Su, and Ninghui Li. 2017. Understanding the Sparse Vector Technique for Differential Privacy. *Proceedings of the VLDB Endowment* 10, 6 (2017).

[55] 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.

[56] 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. https://doi.org/10.1145/1559845.1559850

[57] Kabir Nagrecha and Arun Kumar. 2023. Saturn: An optimized data system for multi-large-model deep learning workloads. *Proceedings of the VLDB Endowment* 17, 4 (2023), 712–725.

[58] 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.

[59] Frank Nielsen and Frank Nielsen. 2016. Hierarchical clustering. *Introduction to HPC with MPI for Data Science* (2016), 195–211.

[60] Manuel Nonnenmacher, Thomas Pfeil, Ingo Steinwart, and David Reeb. 2022. SOSP: Efficiently Capturing Global Correlations by Second-Order Structured Pruning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=t5EmXZ3ZLR

[61] S Ramaswamy, R Mathews, K Rao, and F Beaufays. 2019. Learning for Emoji Prediction in a Mobile Keyboard. *arXiv preprint arXiv:1906.04329* (2019).

[62] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2023. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137* (2023).

[63] 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.

[64] 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

[65] 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.

[66] 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

[67] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[68] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[69] 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-

[70] 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.

[71] 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).

[72] 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. https://doi.org/10.18653/v1/N18-1101

[73] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[74] 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.

[75] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*. PMLR, 38087–38099.

[76] Minke Xiu, Zhen Ming, Bram Adams, et al. 2019. An exploratory study on machine learning model stores. *arXiv preprint arXiv:1905.10677* (2019).

[77] 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.

[78] 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.

[79] 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.

[80] 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. https://doi.org/10.14778/3547305.3547325

[81] 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.