Privacy and Accuracy-Aware Model Deduplication for Emerging Model Infrastructures

Hong Guan
Arizona State University
Tempe, AZ
hguan6@asu.edu

Lixi Zhou
Arizona State University
Tempe, AZ
Lixi.Zhou@asu.edu

Lei Yu
Rensselaer Polytechnic Institute
Troy, NY
yul9@rpi.edu

Xusheng Xiao
Arizona State University
Tempe, AZ
xusheng.xiao@asu.edu

Xi He
University of Waterloo
Waterloo, Ontario
xihe@uwaterloo.ca

Li Xiong
Emory University
Atlanta, GA
lxiong@emory.edu

Jia Zou
Arizona State University
Tempe, AZ
Jia.Zou@asu.edu

ABSTRACT

Integrating databases and deep learning (DL) training and inference processes in the AI era is vital, as seen in the emerging infrastructures, such as model marketplaces, model stores, and model serving platforms, which require managing many models trained/finetuned on private relational datasets, for a broad class of AI/ML applications that rely on relational data, such as fraud detection of credit card transactions, recommendation systems based on product lists and customer transaction logs, and chatbots that probe customer profiles and recent transaction records. Recently, large-scale models, such as large language models (LLM) and foundation models, and fine-tuning these models on private datasets, are increasingly popular. Such trends pose significant storage and memory challenges that prevent model management systems from being deployed or integrated with existing database systems. Another important trend is to enforce fine-grained privacy control by allocating a differential privacy budget to each user for accessing a model or the model's training dataset or pricing differently for models that have been trained with different privacy budgets on the same training datasets. This trend not only aggravates the storage and memory challenges by training many versions of models on the same dataset with different privacy budgets, but also significantly complicates the storage optimization process, such as deduplication. To effectively alleviate the storage overheads and improve memory locality and bandwidth, we proposed a novel privacy-aware and accuracy-aware deduplication mechanism based on ranking model weight blocks by salient analysis, lightweight iterative accuracy validation using Sparse Vector Technique, and a successive halving search strategy. We also evaluated the effectiveness of the deduplication mechanism on large language models, vision transformer models, ResNet models, and embedding layers.

PVLDB Reference Format:
Hong Guan, Lixi Zhou, Lei Yu, Xusheng Xiao, Xi He, Li Xiong, and Jia Zou. Privacy and Accuracy-Aware Model Deduplication for Emerging Model Infrastructures. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

The integration of databases and deep learning (DL) techniques has become increasingly crucial in today's data-driven applications. Many services, including credit card fraud detection, anti-money laundering systems, recommendation engines, and customer service chatbots, rely on inference queries that combine SQL processing with deep learning model inferences. Efficiently managing these models while ensuring privacy has emerged as a critical research question for the database community, spurring the development of various model infrastructures such as model marketplaces [30, 34], model stores [59], and model deployment and serving platforms [23].

In this paper, we focus on the storage and memory pressures posed by such emerging model infrastructures.

On the one hand, large-scale models, such as large language models (LLMs) and foundation models, are widely trained and used. The recent trend of finetuning these models on different datasets leads to more storage and memory management pressures. For example, in the first week when Llama-3 [51, 52] was publicly released, around 1300 models finetuned from Llama-3 were published in HuggingFace [56].

On the other hand, data privacy concerns arise when handling sensitive information in the training datasets. Differential privacy

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

(DP) [12], a mathematical framework based on elastic sensitivity for ensuring the privacy of individuals in datasets, is widely adopted in many real-world applications in Google, Apple, Uber, Microsoft, the United Nations, and the U.S. Census Bureau. Recently, Leveraging DP, a user can access a model if sufficient privacy budget is available for the training dataset. Recently, there emerged a trend of fine-grained privacy control at individual level. For example, DProvDB [63] proposed a fine-grained privacy provenance framework for the multi-user scenario that tracks the privacy loss to each single user to avoid wasting privacy budgets.

These developments have led to an increase in model versions, exacerbating storage and memory pressures. In model marketplaces, data owners train models with varying privacy budgets using DP-enhanced algorithms like DP-SGD [1]. These models are then priced following arbitrage-free principles. Similarly, in model deployment platforms, multiple models trained on the same dataset with different privacy budgets serve queries with varying privacy privileges.

These DP mechanisms also pose unique and significant problems for applying the existing storage optimization techniques such as model deduplication [65] to address the storage and memory pressures brought by emerging model infrastructures. Due to the composition property of differential privacy, a deduplicated model composed of weights from multiple models has an upper bound on its privacy budget, which is the sum of the privacy budgets of the all models involved in the process (Sec. 4.1). Therefore, unlike ordinary model deduplication, deduplicating privacy-preserving models (e.g., models trained using DP-SGD with different privacy budgets) will change the privacy budget of the resulting models. Therefore, the deduplication algorithm must ensure that the resulting models meet the model infrastructure constraints:

• A model's accuracy must exceed that of any other model with a lower privacy budget to ensure arbitrage-free pricing and individual privacy control.

• The deduplication process should not significantly impact the accuracy and privacy distribution across all models.

To address these challenges, we propose **a novel accuracy and privacy-aware model deduplication process**, as illustrated in Fig. 1. In this approach, data owners can train multiple versions of models with varying privacy budgets because some model buyers may require highly accurate inference for critical decision-making processes and other buyers might only need less accurate inference. The weights of these models are partitioned into uniform-sized blocks. Through evaluation on some validation datasets, we identify similar blocks that can be deduplicated without significantly degrading model utility, resulting in a reduced overall storage and memory footprint. These deduplicated model blocks are then compactly stored adjacent to one another, facilitating rapid retrieval and efficient reconstruction of the deduplicated models. This method strikes a balance between storage optimization and the preservation of model accuracy and privacy requirements. However, implementing this approach presents several **technical challenges**:

1. Privacy Budget Constraints: The deduplication process, which involves using shared tensor blocks from one model to deduplicate similar blocks in other models, alters the privacy budgets of the affected models. These changes must be carefully constrained to maintain privacy guarantees.

2. Extensive Search Space: Beyond the privacy-related challenges, a significant hurdle lies in navigating the vast search space to identify similar blocks suitable for deduplication. This challenge necessitates efficient algorithms and heuristics to make the process computationally feasible while maintaining effectiveness.

3. Potential Privacy Costs of Deduplication: The deduplication process may introduce new privacy costs if some private data is used for validation. For instance, a naive accuracy-aware deduplication approach that iteratively evaluates models on a private validation dataset would incur additional privacy costs, potentially compromising the overall privacy guarantees.

For the first challenge, we derived an upper bound on the privacy budget of a deduplication plan based on privacy composition principles (Sec. 4.1). This derivation serves as a crucial guideline for our deduplication algorithm, ensuring that privacy requirements are consistently met throughout the process.

To tackle the second challenge, we conducted an in-depth analysis of the problem's characteristics, leading us to divide the whole problem into more manageable sub-problems. This is achieved by grouping models and selecting a base model for each group (Sec. 4.2). We then employ two key techniques for each sub-problem (Sec. 4.3): saliency analysis and dynamic search strategies. Our results demonstrate that combining a gradient-based saliency analysis approach with dynamic search strategies, such as binary search and successive halving search, effectively balances the compression ratio, accuracy, and the number of required validation steps, which directly impacts latency.

For the third challenge, we propose a novel validation process based on the Sparse Vector Technique (SVT) [12] (detailed in Sec. 4.4). This approach transforms the validation steps into a series of questions about accuracy drop, maintaining a fixed privacy budget until a predefined threshold is reached. This method allows for privacy-preserving accuracy assessments during the deduplication process.

Our work may start a line of research and products on privacy-centric system design. The key contributions are summarized as follows:

• We are the first to formulate the problem of deduplicating models trained with fine-grained privacy budgets for different individuals. We are also the first to design and develop an end-to-end pipeline to address the problem. We believe our work will dramatically alleviate the storage costs, improve the memory locality, and save memory bandwidth for emerging privacy-preserving model infrastructures, including but not limited to model marketplaces, model stores, and model serving platforms.

• We proposed a new algorithm to automatically deduplicate the storage of models while meeting the privacy and utility requirements, leveraging privacy cost composition, SVT, model tensor block salient analysis, and dynamic search strategies.

• We implemented the proposed system and conducted comprehensive evaluations. The results demonstrate that our system achieved a high compression ratio for models with minimal degradation in prediction accuracy, and enables faster model loading in memory-constrained environments.

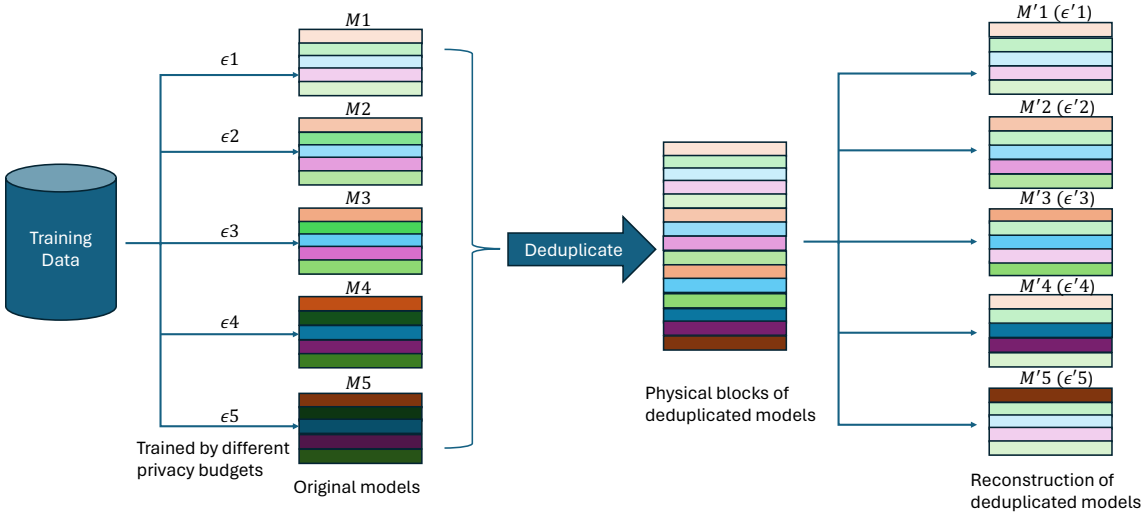

Figure 1: Deduplication of differentially private models

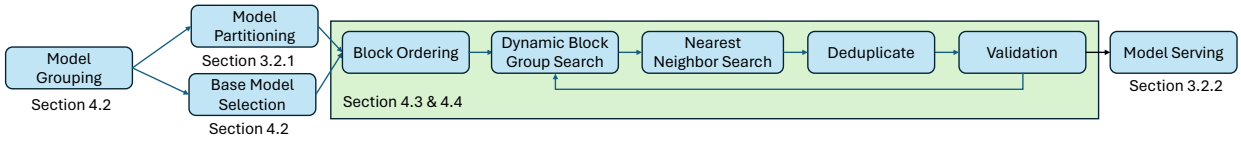

Figure 2: Overview of our privacy-aware and accuracy-aware deduplication process.

2 PRELIMINARIES

Differential Privacy [12] (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 protects against several threat models by ensuring that the inclusion or exclusion of any single individual's data does not significantly affect the outcome of any analysis. It prevents re-identification and membership inference attacks by adding controlled noise to the data, making it difficult for adversaries to determine if an individual's data is in the dataset or to infer sensitive attributes. It also mitigates differencing attacks and correlated information attacks by limiting the amount and accuracy of queries and breaking data correlations. Additionally, differential privacy protects against adversaries exploiting prior knowledge and model inversion attacks by minimizing the impact of any single data point and adding noise during machine learning model training.

Definition 1 (Differential Privacy)**.** A randomized algorithm $\mathcal{M}$ is (ϵ,δ)-differentially private if for all neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$, and for all possible outputs $\mathcal{S}$, $Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{S}] \leq e^{\epsilon} * Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{S}] + \delta$.

In the context of differential privacy, sensitivity measures the maximum change in the output of a function when a single data point in the input dataset is modified. Formally,

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 datasets that differ by exactly one element.

Properties of Differential Privacy [12]. Differential privacy 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)-differentially private and $\mathcal{M}_2$ is (ϵ_2, δ_2)-differentially private, then their sequential composition $\mathcal{M}(x) = (\mathcal{M}_1(x), \mathcal{M}_2(x))$ is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-differentially private.*

Parallel Composition 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)-differentially private 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 $\max_i \epsilon_i, \max_i \delta_i)$-differentially private.*

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)-differentially private and f is any function, then f $\circ$ $\mathcal{M}$ is also (ϵ, δ)-differentially private.*

Sparse Vector Technique (SVT) [12, 37] is a method in differential privacy 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 c denote the maximum number of queries to be answered before halting, Δ represent the sensitivity, and ϵ signify the total privacy budget. The privacy budget is divided into two parts: ϵ_1 for the threshold and ϵ_2 for query results, such that $\epsilon_1 + \epsilon_2 = \epsilon$. The algorithm proceeds as follows: noise is sampled from a distribution $\text{Lap}(\frac{\Delta}{\epsilon_1})$ and added to a predetermined threshold. For each query, noise is sampled from a distribution $\text{Lap}(\frac{2c\Delta}{\epsilon_2})$ and added to the query result. If the noisy query result is less than the noisy threshold, the procedure continues; otherwise, the procedure halts. A recommended approach for allocating the privacy budget is to set $\epsilon_1 : \epsilon_2 = 1 : (2c)^{2/3}$.

Differentially Private Stochastic Gradient Decent (DP-SGD) [1] extends the classical Stochastic Gradient Descent (SGD) by incorporating mechanisms to ensure differential privacy, a rigorous framework that provides strong privacy guarantees for individuals' data within a dataset.

The foundation of DP-SGD lies in the principles of differential privacy (DP), which ensure that the output of a computation does not significantly differ when any single individual's data is included or excluded from the input dataset. This guarantee is quantified by the parameters (ϵ, δ), where ϵ (epsilon) bounds the privacy loss and δ (delta) represents the probability of breaching the bound.

In DP-SGD, privacy is maintained through two primary modifications to the traditional SGD algorithm: gradient clipping and the addition of noise. Mathematically, the update rule for the model parameters θ at step t in DP-SGD can be expressed as:

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

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

3 MOTIVATION, SYSTEM EXTENSIONS, AND PROBLEM FORMALIZATION

3.1 Motivating Scenarios

Model Marketplace. A model marketplace is an online platform designed to facilitate the buying, selling, sharing, and deployment of machine learning models [34]. These marketplaces serve as a central hub for various stakeholders in the AI and machine learning ecosystem, including developers, data scientists, businesses, and researchers. There are three major roles in the model marketplace: the data owners provide datasets used for training ML models and receive compensation for their data usage; the model buyers purchase and integrate machine learning models into their systems or applications; and the broker trains a set of differentially private models, aiming to establish arbitrage-free model pricing. [Review: How can we enforce the arbitrage-free constraint? It is different from the fairness rule.] Specfically, models with the same privacy budget but higher prediction accuracy command higher prices. Similarly, for models with the same prediction accuracy, those with a higher privacy budget are valued more. Conversely, a model with lower prediction accuracy and a higher privacy budget than another model has diminished value in a free market [Review: It is not about the market because a worse model can cost less money. The key is on the privacy side. Specifically, a high privacy budget model is supposed to have a better utility] . This principle establishes crucial constraints in the model deduplication process. Due to the composition property of differential privacy, a deduplicated model composed of weights from multiple models has an upper bound on its privacy budget, which is the sum of the privacy budgets of the individual models. Therefore, unlike ordinary model deduplication where any models can be used, the privacy budget constrains which models can be deduplicated, ensuring that the resulting models remain competitive in the marketplace. Furthermore, the accuracy of a deduplicated model must exceed that of any other model with a lower or equal privacy budget. To maintain data privacy protection, regulations prohibit model buyers from colluding [34].

Model Deployment and Serving Platforms. The scaling law indicates that increasing the size of the model or the training dataset by a certain factor can lead to a corresponding improvement in performance. In recent years, we have witnessed the growing sizes of large models, such as large language models and foundation models trained on public datasets. To balance model utlity and data privacy, these models are further fine-tuned for downstream tasks on private data using differentially private optimizers such as DP-SGD [1] and DP-Adam [50]. In model deployment and serving platforms, it is possible that only a subset of differentially private models can be loaded into memory at the same time due to memory constraints. To answer diverse queries from users with different privacy privileges, the platform frequently transfers DP models between disk and memory, incurring significant I/O overhead. With our proposed accuracy-aware and privacy-aware model deduplication, more DP models can be loaded into memory simultaneously, thereby reducing the memory bandwidth and response time.

3.2 Storage and Serving Extensions for Deduplicated Models

State-of-the-art approaches to deduplicating deep learning models [65] 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 [65], 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 [8, 13, 19, 47, 58] and pruning[5, 15, 36, 43, 49]. Before formalizing the deduplication problem that considers both privacy and accuracy, we first describe the necessary system extensions for storing and serving a model deduplicated from other models, to facilitate understanding.

3.2.1 Storage Extensions. A model includes metadata information and the 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.

One issue is that when similar blocks are being deduplicated, these blocks will be replaced by one physical block with accuracy and privacy preserved. However, some use scenarios may desire keep the original blocks that were replaced during the deduplication process for auditing purposes or for future optimization of the approximation plan. To this end, our work could be easily extended to a tiered storage architecture so that these deduplicated blocks will be archived in cold storage, while the remaining blocks after the deduplication are being cached in memory or warm storage (e.g., SSD), when memory becomes insufficient.

3.2.2 Model Serving Extensions. After model deduplication, we have a combined storage for the models, which is composed of three parts:

(1) A 2-dimensional array that holds the blocks for the models, where each row is a block and each block is flattened into a 1-dimensional array;
(2) A list of model constitutions for each model;
(3) A list of dictionaries containing extra weights, such as biases and norms, which are smaller than block size and are thus handled separately. Each model has a dictionary.

After deduplication, certain blocks are removed from the 2-dimensional array and either discarded or transferred to secondary storage. Reconstructing a model requires careful tracking of these blocks. Fortunately, the process is simplified by the fact that each row in the array represents a single block, enabling straightforward indexing. We introduce the term "model constitution" to describe the representation of an entire model as a list of indices. This concept is best illustrated with an example: a Vision Transformer (ViT) model typically comprises 288 blocks and some additional parameters. Its model constitution would be a list of 288 integers, each corresponding to the row index of a specific block in the 2-dimensional array. This compact representation allows for efficient storage and reconstruction of complex models while maintaining the benefits of deduplication.

At the initial stage of online model serving, the combined storage is loaded into memory. Whenever a new query requires a model that is not in memory, the system first checks if there is enough memory to load this model. If there is insufficient space, the system will attempt to reuse a model of the same architecture as a template to load the new model; otherwise, some models will be evicted to disk to free up memory space for the new model. The system will then retrieve the model ID, look up the corresponding model constitution and extra weights, and reconstruct the model.

We will provide a formalization of the problem considering both the accuracy and privacy in the next section.

3.3 Problem Formulation

Problem Background and Notations. Assume there are N models trained on the same dataset, denoted as $M_1, \ldots, M_N$. The corresponding privacy budgets are $\epsilon_1, \epsilon_2, \ldots, \epsilon_N$. A model is partitioned into fixed-size blocks. A model can be represented by $M_i = [b_1, \cdots, b_{l_i}]$ for $i \in \{1, \ldots, N\}$, where l_i is the number of blocks in M_i. The set of all blocks before deduplication is denoted as $B = \{b | b \in M_1 \cup \ldots \cup M_N\}$.

Intuition. The ***block deduplication problem*** involves transforming models $M_1, \ldots, M_N$ into a new set of N models $M'_1, \ldots, M'_N$ while preserving their size and architecture. This transformation aims to achieve multiple objectives:

(1) Minimize the total number of unique blocks.
(2) Minimize the privacy budget increase.
(3) Minimize the utility drop.

Simultaneously, the transformation must adhere to three constraints:

(1) The privacy budget increase must not exceed a threshold ϵ_i^*.
(2) The utility drop must remain below a threshold T_i^*.
(3) Models trained with lower privacy budgets on the same dataset must maintain lower accuracy than those trained with higher budgets. We term this the **fairness rule**.

This complex problem can be characterized as a black-box multi-objective combinatorial optimization problem, which we formalize as follows:

Optimization objectives:

$$\text{minimize } B' \triangleq \text{numUniqueBlocks}(M'_1, \ldots, M'_N),$$

$$\text{minimize } \Delta\epsilon_i \triangleq \epsilon'_i - \epsilon_i$$

$$\text{minimize } \Delta U \triangleq U_i - U'_i$$

Subject to constraints:

$$\Delta\epsilon_i \leq \epsilon_i^*$$
$$\Delta U \leq T_i^*$$
$$U'_j \leq U'_k \iff \epsilon'_j \leq \epsilon'_k$$

for any $i, j, k \in \{1, \ldots, N\}$. The privacy budget ϵ' for a transformed model M' is calculated as the sum of privacy budgets from all constituent models. A detailed explanation and proof of this result are provided in Section 4.1. We evaluate both the original and transformed models on a validation dataset to obtain their respective utility values, U_i and U'_i. As the number of models increases, the absolute value of B' (the set of unique blocks after deduplication) also grows. Therefore, to assess the effectiveness of a deduplication scheme, we use the compression ratio $|B'|/|B|$, where B represents the original set of blocks.

This is a general problem definition for different applications. One can change this problem definition by favoring one or two objectives according to their own application. For example, in a scenario where privacy preservation is paramount, one might prioritize minimizing the increase in privacy budget ($\Delta\epsilon$) while still

maintaining a reasonable level of utility. This could be particularly relevant in healthcare applications where patient data confidentiality is crucial. Conversely, in a situation where model performance is the primary concern, such as in a competitive machine learning competition, the focus might shift towards minimizing the decrease in utility (ΔU) while keeping the privacy budget increase within acceptable limits. In yet another case, where storage or transmission efficiency is critical, such as in edge computing or IoT applications, the emphasis might be on maximizing the compression ratio by minimizing the number of unique blocks across models. The flexibility of this problem definition allows practitioners to adapt the optimization strategy to their specific needs and constraints in various real-world applications.

4 AUTOMATIC DEDUPLICATION

4.1 Privacy Budget Derivation

We give Theorem 4 to derive the privacy budget of a model that consists of deduplicated blocks from multiple source models trained on different training datasets with different privacy budgets. [TODO: point out that all models involved in the deduplication process so the sum of privacy budget is upper bound.]

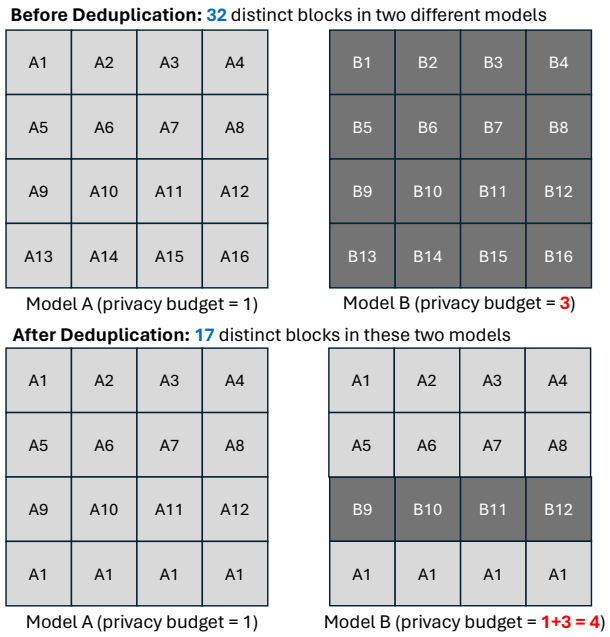

Figure 3: Deduplication influences storage and privacy

Theorem 4. *Given a derived model m' that consists of a list of q (deduplicated) blocks, denoted as $B' = \{b'_1, ..., b'_q\}$, which are from a collection of source models $\hat{M} \subset M$. Then, the upper-bound privacy budget of m' for the training dataset r_j, is defined as the sum of the privacy budget of all distinct source models that are trained on r_j and contain at least one block in B'.*

Proof. Suppose $b'_1 ..., b'_q$ are the blocks from n distinct source models $m_1, ..., m_n$, each trained with $\epsilon_1 ..., \epsilon_n$-DP. For simplicity, assume that any two source models are trained on either the same dataset or the datasets completely disjoint from one another. We can divide these n models into groups by their training datasets. $M_j = \{m_{j_1}, ...m_{j_k}\}$ is a group of models using training data r_j. By DP composability, the collection of blocks from this group of models satisfies ϵ^j-DP where $\epsilon^j = \sum_{i=1}^{k} \epsilon_{j_i}$. Let d be the total number of training datasets involved, that is, d is the number of model groups. Given that any dataset r_j $(1 \leq j \leq d)$ is disjoint from each other, by parallel composition property [41] of DP, the derived model m' is $(\max_j \epsilon^j)$-DP.

Generalization. We can easily remove the above simplicity assumption by formulating the problem into a graph. Assume that for graph G, each node represents a source model, and two nodes m_p and m_q have an edge if and only if their training datasets overlap. Regarding each connected component (including a single node) C_j of k nodes as a model group $\{m_{j_1}, ...m_{j_k}\}$, similarly, the collection of blocks from it satisfies $\epsilon^j = \sum_{i=1}^{k} \epsilon_{j_i}$ -DP. Suppose G has d connected components, then the derived model m' is $(\max_{1 \leq j \leq d} \epsilon^j)$-DP.

Three Rules It is difficult to define whether a block can be deduplicated by another block. For a simple deduplication problem without considering privacy, we may focus on whether deduplicating these two blocks will lead to an accuracy drop smaller than a pre-specified threshold for each model. In this work, we need to consider the changes to the privacy budget of each model caused by the deduplication. The heuristics that we learned from the above privacy budget derivation are (1) **Intra-Model Rule**: If two similar blocks are from the same model, they can be safely deduplicated without affecting the model's privacy budget; (2) **Inter-Model and Intra-Data Rule**: After deduplication, a model contains blocks from k models (the model itself can also be one of these models) that are trained on the same dataset, then the total privacy budget becomes $\epsilon = \sum_{i=1}^{k} \epsilon_i$, where ϵ_i is the privacy budget of model M_i before deduplication. (3) **Inter-Model and Inter-Data Rule**: If two similar blocks b_1 and b_2 are from two different models M_1 with ϵ_1 and M_2 with ϵ_2 that are trained on different datasets R_1 and R_2 respectively. Using b_1 to replace b_2, will not change M_2's privacy budget on R_2, but will increase the M_2's privacy budget on R_1 from 0 to ϵ_1. We apply the first two rules to the deduplication process, see Sec. 4.3. The last rule is mainly used in the online model composition and serving process, see Sec. 3.2.2.

4.2 Model Groups and Base Models

The three optimization objectives—privacy budget, utility, and compression—are inherently conflicting. For instance, minimizing the increase in privacy budget requires choosing models with a small privacy ϵ value for deduplication. However, this results in a significant utility drop, as models with higher privacy budgets typically offer better utility. On the other hand, the compression ratio depends on the models used during deduplication. Models with the same architecture and trained on the same dataset with similar ϵ values are likely to have similar weights, potentially yielding a high compression ratio.

Given these trade-offs, we can now analyze the three objectives more closely. Among utility, privacy budget, and compression, we have direct control over the first two during deduplication. The

utility objective can be managed by setting a threshold: if deduplicating a block causes a utility drop exceeding this threshold, the step is reversed. Regarding privacy, while it's theoretically possible to involve all models in the deduplication process, this approach is impractical for two reasons. First, the resulting models would all have the same privacy budget and, according to the fairness constraint, should have the same utility. However, the system would not need multiple models with identical privacy budgets and utility on the same dataset, rendering most models redundant. Second, according to Theorem 4, the resulting model would have a privacy budget upper bound equal to the sum of all involved models' budgets, potentially many times the original budget. Such a significant increase would severely compromise privacy in practical applications.

To address these issues, we introduce two concepts:

- `Model Groups:` Models trained on the same dataset using identical architectures are likely to have similar weights and blocks. To optimize deduplication, we group models by architecture and training dataset, then apply the Intra-Model Rule and Inter-Model Intra-Data Rule within each group. Our experiments in Section 5.9 demonstrate that mixing models with different architectures significantly reduces the compression ratio. Similarly, Section 5.10 shows that applying the Inter-Model Inter-Data Rule to deduplicate models trained on different datasets also diminishes the compression ratio. To maintain model quality, we enforce a fairness rule: models with higher privacy budgets should exhibit higher utility. We implement this through a **prefiltering step** where, within each group, we filter out any model that has a higher privacy budget but lower utility compared to another model in the same group.
- `Base Models:` For each model group, we designate one model as the base model, which provides blocks for deduplicating other models. We employ two methods to select base models. The first method simply chooses an existing model from the group to serve as the base model. The second method involves training a new dummy model specifically for this purpose; after deduplication, any unused blocks from this dummy model are discarded. When privacy budgets within a group exhibit significant variation, we extend this approach by subdividing the large group into smaller subgroups, ensuring that models within each subgroup have comparable privacy budgets. Each of these subgroups then has its own designated base model. To evaluate the effectiveness of these different approaches, we conduct experiments in Sec. **??**, which compares various methods for obtaining base model(s) and their impact on the overall deduplication process and resulting model performance.

There are C different way to group N model to s sub-groups. When we study different way to divide models in a subgroups, we mainly think of two aspects. The first aspect is the base model for each subgroup because the choice of base model affects all three objectives. The second aspect is the models in a sub-group, we consider grouping by every N models and every other K models. For example, a group of 10 models are trained with privacy budget from 1 to 20. Every 5 model grouping will make groups: [1,2,3,4,5] and [6,7,8,9,10]. Every other 2 models will group them into [1,3,5,7,9], and [2,4,6,8,10].

Combining these ideas, each model in each (sub)group is deduplicated using blocks from itself and that (sub)group's base model. The resulting privacy budget is the sum of the original budget plus the base model's budget. This approach ensures that the deduplicated models' privacy budgets (ϵ values) remain distinct, provided the original models had different budgets—a condition guaranteed by our prefiltering step.. We can then set the utility threshold T_i accordingly to satisfy the fairness rule.

With model groups and base models, the problem is redefined as follows: without loss of generality, assume that the models $M_1, \ldots, M_N$ are pre-sorted by their privacy budgets in ascending order. The model M_1 with the smallest privacy budget is set as the base model, and the objective is to find

$$\begin{aligned} &\text{minimize numUniqueBlocks}(M_i') \text{ with} \\ &M_i' = \text{Deduplicate}(M_i; M1 \text{ as base}) \ s.t. \\ &U_i' \geq U_i - T_i \text{ for any } i, j, k \in \{2, \ldots, N\} \end{aligned}$$

Interactive Utility Threshold. In the above problem formulation, we used an accuracy drop threshold T_i to enforce the fairness rule. However, if it is set to a constant t_i, the fairness requirement can be violated as long as there is an overlap between two models' accuracy ranges. For example, assume$U_j < U_k$ for two models M_j and M_k. if $U_k - \Delta U_k < U_j$, then there is an overlap in their range, which is $[U_k - \Delta U_k, U_j]$. It is possible that the deduplicated model utility $U_j', U_k' \in [U_k - \Delta U_k, U_j]$ and $U_j' > U_k'$, which violates the fairness rule. To address this issue, we propose an interactive utility threshold. Specifically, we sort the models according to their utility in ascending order. Then we deduplicate them sequentially and set the threshold to the maximum of the previous deduplicated model's utility and a pre-specified utility threshold. Formally, the threshold of the ith model is

$$T_i = \max(U_{i-1}', U_i - t_i) \text{ for } i \in \{2, 3, ..., N\}$$

4.3 Privacy and Accuracy-Aware Deduplication Algorithms with Dynamic Search

Deduplicating a block changes the parameters of the model, which may affect the utility of a model. Not only the amount of changes accumulate during the deduplication process, but also the synergy between blocks affects the holistic behaviour of the model. It implies that, under the fixed threshold, the order of deduplication determines the final model and its utility. Given N blocks, there are $N!$ candidate deduplicate plans. Depending on the block size, deep learning models in our experiments may have hundreds to thousands of blocks. Therefore, it is very computationally expensive to exhaustively search and evaluate the storage benefit and accuracy drop of all deduplication plans, given the exponential search space. We further identify that in existing accuracy-aware model deduplication work [65], the model needs to be tested on a validation dataset after every k deduplication steps to make sure the model utility is within the acceptable threshold during the deduplication process and this validation process comprises most of the deduplication latency. Moreover, as mentioned earlier in our target scenarios, if a private validation dataset is used to evaluate the model after

deduplicating every block, it will increase privacy cost if using a naive privacy accounting method. To alleviate the validation bottleneck and mitigate the privacy accounting problem caused by the validation, we combined multiple key ideas: salient block ranking, block similarity heuristics, novel group-based dynamic search strategy, and the Sparse Vector Technique (SVT) algorithm (introduced earlier in Sec. 4.4). With SVT, the privacy cost is fixed before the accuracy check result reaches a threshold, no matter how many evaluation steps are run.

To tackle the problem that the computation is bottlenecked by the validation process, we first consider the validation data size. Unlike deduplication without privacy constraint [65], keeping a large size of validation dataset is benefitial. A bigger validation dataset not only better represents the actual workload, but also introduces smaller noise level to SVT algorithm, which leads to better utility estimation in every deduplication step.

A naive deduplication algorithm runs validation after deduplicating one block. This process is fine-grained but very expensive. Luckily, with our proposed saliency-based block ordering, blocks that have less effect to the utility are more likely to be in the front of this ordering. With this observation, our main solution is to reduce the validation frequency by deduplicating a group of blocks at each time. However, the design space of grouping the blocks is large. Given n blocks, there are $\sum_{r=1}^{n} \binom{n+r-1}{n}$ ways to group multiple blocks.

To design an effective grouping strategy, our main idea is to avoid the bad blocks that will cause the accuracy drop beyond the pre-specified threshold. Deduplicating some blocks will influence the model utility so that the validation accuracy drop threshold will be reached from time to time, which not only increase the privacy costs under SVT but also increase the computational costs because the current deduplication scheme needs to be rolled back and explore other candidate deduplication schemes. However, it is very difficult to identify these "bad" blocks that will significantly impact the accuracy, because it not only depends on the values of weights in the blocks and the positions of the block in the model, but also depends on the difference between the block and the block that will be used to duplicate it. To address the problems, our novel solution is based on two ideas:

• We can order all the blocks according to a saliency measure, and then group blocks for deduplication in order. The ordering of the blocks should ensure that the less salient blocks (i.e., blocks that are less influential to the model accuracy) are deduplicated as early as possible. Some salient blocks should not be deduplicated because they greatly affects the model utility.

• We can dynamically adjust the size of each group to further reduce the chance that the validation accuracy of the model after deduplicating the group of blocks will drop below the threshold. For example, due to the ranking based on block saliency, at the begining, a group that has a large size consists of many less sailient blocks and the validation usually lead to promising results. However, at the end of the ordered list, the group size has to be controlled as small to avoid validation failures.

Additional optimization of the deduplication process include: [Review: Shall we remove these two points because they are not discussed in detail and not used in the experiments?]

• We can leverage locality sensitive hashing to accelerate the process for identifying the most similar block to duplicate a block in the group to be deduplicated.

• After deduplication, we can finetune the model using public datasets if available, to further improve the model accuracy. Particularly, the finetuning should be designed in a unique way so that it will not affect the number of deduplicated blocks.

We describe these ideas in more detail in the rest of the section.

Saliency measures. [Review: State that it answers the question of which block to be deduplicated or in what order the blocks are deduplicated] Research in model explanability, pruning and quantization found that weights of a deep learning model do not have the same contribution to the model output. Weights that have high impact to the output are called "salient weights" [22, 49]. There are many ways to measure the saliency of weights in the model explanability [], pruning [31], and quantization [16, 17] research literature. The most popular ones include the weight magnitude, activation magnitude, combination of the former two measure, and Fisher information. In particular, Fisher information involves the square of gradients. However, for large models, the gradients are usually very small and the square of them are even smaller, which introduces unstable numerical results. Therefore, we added gradient itself as an additional saliency measure. The above measure are for every weight, not for a block of weights. In this paper, the saliency of a block is defined as the mean of the saliency measure for each weight within a block. We empirically compare these choices in Sec. 5.

Block similarity heuristics. [Review: State that it answers the question of which block to deduplicate] Deduplicating a block by another involves two sub-steps. The first sub-step is choosing a block b to be deduplicated and the second sub-step is choosing a block to deduplicate block b. The ranking according to saliency measure helps the first sub-step. For the second sub-step, if a random block is chosen to deduplicate block b, it is very likely that the model utility is greatly degraded. A common wisdom is to choose a block that is most similar to block b. It is also a natural solution from first principles. The goal of this choice is to maximize the utility and the model is trained to maximize the utility, so the model after deduplication cannot do much better than the origin model. In practice, the utility of the resulting model usually is worse than the original model. From the angle of changing the weights after training yet maintaining the model performance, model pruning and quantization share some similar aspects with model deduplication. However, the latter has more constraints than the former problems: 1) A large block of weights are updated simultaneously; 2) The update of weights cannot be set freely but must equal to another block of weights. These constraints leave model deduplication less freedom to adjust the weights. From the view of minimizing MSE of the layer's output which is a common objective function in model quantization, deduplicating a block by the most similar block is a good heuristic to minimize this objective.

Group deduplication algorithms. After blocks are ordered by their saliency measure, groups of adjacent blocks are deduplicated simultaneously to reduce the number of expensive validation steps. One approach adopted in [65] is to deduplicate a fixed number

Algorithm 1 Deduplicate every N blocks (GreedyN)

Define variables: Total number blocks to be deduplicated l; Deduplicate every N block at each step; Allowed utility drop threshold T.
procedure GreedyN(N, T)
  Sort the blocks by saliency measure;
  for group i in $0, 1, \ldots, \lceil \frac{l}{N} - 1 \rceil$ **do**
    for block j in $0, 1, \ldots, N-1$ **do**
      Deduplicate block $b[i \cdot N + j]$ by the most similar block;
    Evaluate model on validation set, compute utility drop d;
    if $d > T$ **then**
      Revert the change of group i;

Algorithm 2 Binary Search

Inputs: the total number of blocks N, left pointer *left*; right pointer *right*, mininum number of blocks to be deduplicated n; M is a map from blocks that are deduplicated to their corresponding blocks that are used to deduplicate them.
procedure BinarySearch(N, *left*, *right*, n, M)
  $l = 0, r = N - 1$;
  while *right* − *left* >= n **do**
    compute middle point $mid = (right - left)//2$;
    leftDeduplicatedBlocks = BinarySearch($mid - left$, *left*, *mid*, n, M);
    if *leftDeduplicatedBlocks* = $mid - left$ **then**
      Add blocks from *left* to *mid* and their corresponding blocks to M;
    else if *leftDeduplicatedBlocks* > $(mid - left)//2$ **then**
      rightDeduplicatedBlocks = BinarySearch(*right* − *mid*, *mid*, *right*, n, M);
    return *leftDeduplicatedBlocks* + *rightDeduplicatedBlocks*

Algorithm 3 Successive Halving

Inputs: the total number of blocks N, left pointer *left*, right pointer *right*, mininum number of blocks to be deduplicated n, M is a map from blocks that are deduplicated to their corresponding blocks that are used to deduplicate them.
procedure RecursiveHalving(*currSeq*, *rightSeq*, n, M)
  if $len(currSeq) < n$ **then return**
  $mid = len(currSeq)//2$;
  $leftSeq = currSeq[: midPoint]$;
  $rightSeq = currSeq[midPoint :] + rightSeq$;
  if *leftSeq* can all be deduplicated **then**
    $currSeq = rightSeq$;
    $currSeq = [\,]$;
  else
    $currSeq = leftSeq$;
  RecursiveHalving(*currSeq*, *rightSeq*, n, M).

of blocks N without overlap, as shown in Algorithm 1. This algorithm heavily relies on the block ordering according to the saliency measure. It works well if the ordering places salient blocks close to each other so that these blocks are grouped together and not deduplicated. However, the algorithm has a shortcoming: if only a few blocks within a group are salient, the entire group will fail to be deduplicated. Additionally, deciding the number N is challenging. If N is small, the number of evaluation steps is high. Conversely, if N is large, more blocks will fail to be deduplicated. To address this problem, our idea is to group the blocks more aggressively and re-evaluate some blocks in a group that failed to be deduplicated. We proposed two variants of this approach. Algorithm 2 is similar to a binary search to identify the salient blocks within a group. Algorithm 3 changes the halving scheme to ensure that each expansion group is half of the remaining blocks.

4.4 Sparse Vector Technique (SVT) for Validation Using Private Data

To provide a statistical accuracy guarantee for models that have deduplicated a portion of the tensor blocks, we propose to test each model on a validation dataset during the deduplication process and stop deduplicating for a model once the accuracy drops below a threshold.

The selection of the validation datasets could be challenging. If abundant public datasets are available in the domain, we may sample these datasets to form a validation dataset without privacy concerns. However, if we generate the validation datasets based on the private datasets to which the model was once applied, privacy information may leak during the validation phase, which should be accounted for in the end-to-end privacy budget.

To address the problem, we leverage SVT to transform the validation process into a boolean question regarding the accuracy change over the validation dataset.

Suppose that $f(D, M)$ evaluates the validation accuracy of model M on private validation dataset D. Given a classification model, 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 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 indicate whether the deduplication should stop. Accordingly, SVT works by applying Laplace noise and comparing the utility drop with a noisy threshold, as shown in Algorithm 4. It satisfies ϵ-DP [38]. 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.

In Algorithm 4, the Laplace noise applied to the utility is $\text{Lap}(\frac{8c}{|D|\epsilon})$, where c is the maximum number of times the utility drop exceeds the pre-specified threshold, and $|D|$ is the size of the validation dataset D. A high noise level results in inaccurate utility estimation, leading to underperforming deduplicated models. To mitigate this, it is beneficial to have a larger $|D|$ and a smaller c. This rationale justifies our choice of using the full validation set to ensure a large $|D|$. Additionally, since c is pre-determined as an algorithm parameter, it represents the maximum number of times the deduplicated model can fail the utility test. By employing saliency-based ordering and grouping, we aim to cluster salient blocks, thereby reducing the number of deduplication failures.

5 EXPERIMENTS

[Review: How to design experiments to show the effectiveness of our approach? An overview or a flagship experiment table/figure. How to highlight our contributions clearly?] [TODO: Add an

Algorithm 4 ϵ-DP Deduplicate with SVT

Input: private validation dataset D; model M to be deduplicated and its total number of blocks B; allowed utility drop threshold T; cut-off c, the number of times that the utility drop exceeds threshold T; privacy budget ϵ.
procedure SVT(M, D, T, c, ϵ)
    $\epsilon_1 = \frac{1}{1+(2c)^{2/3}}\epsilon$, $\epsilon_2 = \frac{(2c)^{2/3}}{1+(2c)^{2/3}}\epsilon$;
    $\hat{T} = T + \text{Lap}(\frac{2}{|D|\epsilon_1})$; ▷ Add noise to threshold T
    count = 0;
    for step i in 1,...,B **do**
        M' = Deduplicate(M) ;
        $d = f(D, M) - f(D, M')$; ▷ Compute utility drop
        $v_i = \text{Lap}(\frac{4c}{|D|\epsilon_2})$;
        if $d + v_i \geq \hat{T}$ **then** ▷ Deduplication succeeds
            $M = M'$
            count = count − 1;
            if count $\geq c$ **then** ▷ Reach cut-off c
                Output M; Halt;
        else ▷ Deduplication fails
            Output M; Halt;

table/figure to show the utility drop from the deduplicated model and model with model trained with that budget;]

To assess the effectiveness of the deduplication algorithms, we primarily experimented with batch deduplication across four workloads, as detailed in Sec. 5.4. These experiments involve several hyperparameters that affect the algorithm's accuracy and compression ratio. Consequently, in Sec. 5.5, we conducted ablation studies to examine the effects of varying block sizes, saliency measures, aggregation functions, distance measures, accuracy thresholds, and cross-model deduplication. Sec. 5.6 demonstrates that our proposed deduplication algorithm is orthogonal to and can be combined with pruning and quantization techniques. In Sec. 5.7, we establish the algorithm's scalability for deduplicating numerous models, provided one can train models with different privacy budgets and accuracy levels. Sec. 5.9 illustrates that the deduplication algorithm is applicable not only to models of the same architecture but also to those with different architectures. The results of applying the Inter-Data Rule are presented in Sec. 5.10. In Sec. 5.11, we show that using deduplicated model weights allows for simultaneous loading of multiple models into memory, thereby reducing model loading overhead. Finally, Sec. 5.12 examines a special case where the model consists of an embedding layer and a small classification head, discussing specific treatments applicable to such models.

5.1 Experimental Settings

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

(1) RoBERTa [35]: A Transformer encoder for natural language inference tasks, composed of embedding layers, attention layers, layer normalization layers, and linear layers.
(2) Vision Transformer (ViT) [9]: A Transformer model for image classification tasks, consisting of attention layers, layer normalization layers, and linear layers.
(3) ResNet [20]: A convolutional neural network for multi-attribute classification tasks, primarily composed of CNN layers and batch normalization layers.
(4) Multilayer Perceptron (MLP) [45]: A network for product recommendation tasks, consisting of large linear layers.
(5) Word2Vec [42]: An embedding model for sentiment analysis tasks, composed of large embedding layers and linear layers.

These models encompass all major layer types found in modern deep learning architectures, ensuring that our proposed algorithms are applicable to a wide variety of models beyond those explicitly tested.

5.1.2 Comparison. None of existing model deduplication mechanisms have considered privacy, therefore, we used the following baselines for comparison:

(1) The Greedy-N algorithm with our privacy mechanisms;
(2) Our binary search algorithm with our privacy mechanisms;
(3) Our successive halving algorithm with our privacy mechanisms;
(3) We also considered Monte Carlo Tree Search (MCTS) with our privacy mechanisms as a baseline. However, in naive MCTS, the action space is very large. Consider a model with $l1$ blocks and candidate models, including itself, with $l2$ blocks. The action space is on the order of $O(l1 \cdot l2)$. Given that $l1$ and $l2$ are in the hundreds, this results in an action space in the tens of thousands. By applying some heuristics, the action space can be significantly reduced. First, similar to the proposed approaches, in the second sub-action, we always choose the block with the smallest $l1$ distance to the block being deduplicated. Additionally, in the first sub-action, we order the blocks by their saliency measure and group every g blocks in this ordered sequence as one action. This reduces the action space to $O(l1/g)$. In our experiments, g is set to 20 and 30, and the number of episodes is set to 20.

5.1.3 Experimental Environments. Except the online serving experiment that is ran on an AWS c5a.xlarge instance (7.63 GB main memory, 4vCPU), all other experiments are ran on a machine with an Intel(R) Xeon(R) Silver 4310 CPU (2.10Hz) and NVIDIA A10 GPU (24GB memory).

5.2 Base Model Selection

In this section, we studied how different base models affect the compression ratio, model accuracy and privacy budget. First, we trained five ViT models on CIFAR100 using privacy budgets 0.5, 0.55, 0.6, 0.65, 0.7 as a model group. It only makes sense to use a smaller privacy budget model to deduplicate a higher privacy budget model because the latter usually have a higher accuracy, otherwise, the smaller privacy budget model can be complete deduplicated. Therefore, we compared four different choices of privacy budgets to train a base model, which are 0.35, 0.4, 0.45, and 0.5. Among them, model with privacy budget 0.5 is an existing model, the rest are dummy base models.

5.3 Model Sub-Grouping Strategy

5.4 Effectiveness of Model Deduplication

We evaluated our approach on three tasks: natural language inference with language models, mulit-attribute classification with vision models, and movie recommendation with linear recommender models.

5.4.1 Natural Language Inference with Language Models. In this task, we trained and deduplicate Roberta-base models on QNLI (Question-answering Natural Language Inference) dataset with privacy budgets 1, 2, 4, 6, 7. The model with the smallest privacy budget is served as the base model and other models use the blocks from the base model and themselves for deduplication. In this way, the increase in privacy budget is small and thus acceptable. After deduplication, the privacy budgets of these models become 1, 3, 5, 7, 8. The overall compression ratio, number of validation steps, and maximum accuracy drops for five models are in Tab.4. To provide more detailed information for each model, Tab. 1 shows the above three measure as well as the number of deduplicated blocks from the base model and the number of deduplicated blocks from the model itself. This experiment used the successive halving algorithm.

Table 1: Deduplication Details for Roberta-base Language Models Using Recursive Halving Algorithm

Models	M1(eps2)	M2(eps4)	M3(eps6)	M4(eps7)
C.R.(%)	20.6	20.2	20.6	40.1
Num. of Val.	6	8	6	9
Accuracy	0.000	0.008	0.011	0.014
Num. from base	271	272	271	217
Num. from itself	0	0	0	0

5.4.2 Multi-Attribute Classification with Vision Models. In this task, we trained and deduplicate ViT Large models on CIFAR100 dataset with privacy budgets 0.5, 0.6, 0.75, 1.0, 2.0 and ResNet152 models on CelebA datset with privacy budgets 0.4, 0.6, 0.8, 1.0, 2.0.

Table 2: Deduplication Details for ViT-large Models Using Binary Search Algorithm

Models	M1(eps0.6)	M2(eps0.75)	M3(eps1.0)	M4(eps2.0)
C.R.(%)	3.6	22.2	19.1	19.1
Num. of Val.	6	6	6	6
Acc. drop	0.013	0.018	0.019	0.020
Num. from base	279	226	234	234
Num. from itself	0	1	0	0

5.4.3 Movie Recommendation with Recommender Models. In the movie recommendation task, we ran big MLP recommender models on the Movie-Lens dataset. These models are trained using privacy budget 0.5, 1.5, 2.5, 4.5, 8.5. The compression ratios, number of validation steps, maximum accuracy drops using different deduplication algorithms are shown in Tab. 5.

Table 3: Deduplication Details for ResNet Models Using Binary Search Algorithm

Models	M1 (eps0.6)	M2 (eps0.8)	M3 (eps1.0)	M4 (eps2.0)
C.R.(%)	8.6	8.6	8.6	35.6
Num. of Val.	6	6	6	9
Acc. drop	0.008	0.014	0.020	0.018
Num. from base	230	230	230	170
Num. from itself	0	0	0	0

5.5 Ablation Studies

5.5.1 Impact of Block Sizes. In general, block sizes is not a crucial factor that affects the alogrithm. We used the ViT-large to show the impact of block size. The results are shown in Tab. 6.

5.5.2 Choice of Saliency Measure. A saliency measure helps algorithm order the blocks before the actual deduplication. There are many off-the-shelf saliency measures, such as weight magnitude, Wanda, Fisher information, and weight gradient magnitude. In Tab. 7, we compared the effectiveness of the these saliency for deduplication on the Robert-base model.

5.5.3 Choice of Block Aggregation Function. There are different ways to aggregate the saliency measure of the block block, such as l1-norm, l2-norm, third quartile, etc. In general, the l2 distance works well for almost all cases. Here, we compared the effect of l1, l2, l-infinite distance, third quartile on Roberta-base model in Tab. 8.

5.5.4 Choice of Distance Measure. There are different ways to measure the similarity of two blocks, such as l1, l2, l-infinite distance, etc. In general, l2 distance works well for almost all cases. Here, we compared the effect of l1-norm, l2-norm, and cosine distances on Roberta-base in Tab. 9.

5.5.5 Impact of Accuracy Threshold. Generally speaking, better compression ration can be achieved with a lower accuracy threshold (larger accuracy drop threshold). However, with a larger accuracy drop threshold, the actually accuracy drop is also larger, resulting in greater utility compromise. For the text and vision tasks, we deduplicated the same models using different accruacy drop thresholds and show their impact on the compression ratios in %. The results are shown in Tab. 10. A negative accuracy drop threshold requires the resulting model have higher accuracy than the original model. The result shows that a threshold of 1%-2% strikes a good balance between accuracy and compression ratio. The compression ratio for Vision-ResNet with accuracy drop threshold -0.5% is greater than 100% because zero paddings are added to some layers to form fixed size blocks.

5.5.6 Comparsion Between Cross-Model Deduplication and Self Deduplication. In this section, we considered the scenario of training new models given a fixed amount of privacy budget cap ϵ_{cap}. We compared two ways to obtain a new model. The first way is to select a model with a small privacy budget ϵ_{base} as a base model to provide blocks to deduplicate a new model trained with the privacy budget $\epsilon_{cap} - \epsilon_{base}$. This is similar to the other experiments. Another way to train a model with privacy budget ϵ_{cap} and then run

Table 4: Comparison of Different Deduplication Algorithms

	Roberta-base for NLI			ViT for Object Classification)			ResNet for Multi-Attribute Classification)		
	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
Binary Search	26.5	29	0.010	32.8	24	0.019	32.3	27	0.017
Successive Halving	25.9	29	0.014	32.8	31	0.019	32.5	31	0.017

Table 5: Comparison of Different Deduplication Algorithms for Recommender Models

	C.R.(%)	Num. of Val.	Max. acc. drop
MCTS-20blocks	20.0	44	0.020
MCTS-30blocks	20.0	28	0.020
Greedy-1	20.0	804	0.020
Greedy-10	20.0	84	0.020
Greedy-20	20.0	44	0.020
Greedy-30	20.0	28	0.020
Greedy-40	20.0	24	0.020
Binary Search	20.0	21	0.020
Successive Halving	20.0	21	0.020

Table 6: Ablation Study of Different Block Sizes

	C.R.(%)	Num. of Val.	Max. acc. drop
1,048,576	32.8	31	0.018
524,288	42.9	47	0.020
262,144	64.5	58	0.020

Table 7: Ablation Study of Different Saliency Measures

	Analysis Latency(s)	C.R.(%)	Num. of Val.	Max. acc. drop
Weight Magnitude	113	63.8	49	0.015
Wanda	145	47.3	30	0.015
Fisher Information	6576	45.9	24	0.015
Gradient Magnitude	1060	32.8	31	0.019

Table 8: Ablation Study of Block Aggregation Function

	C.R.(%)	Num. of Val.	Max. acc. drop
l1	27.6	26	0.014
l2	26.1	31	0.013
l-infinite	30.0	35	0.015
3rd-quartile	27.1	32	0.015

Table 9: Ablation Study of Different Distance Measures

	C.R.(%)	Num. of Val.	Max. acc. drop
l1-norm	25.9	29	0.014
l2-norm	40.2	31	0.012
cosine	92.1	63	0.014

Table 10: Compression Ratios for Different Accuracy Thresholds

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

deduplication with blocks from itself. We compare the compression ratio and accuracy drop in Table. 11. The results shows that cross-model deduplication is much better than self deduplication in terms of compression ratio.

Table 11: Compression ratio (%) Between Cross-Model Deduplication and Self Deduplication

	Self Deduplication	Cross-Model Deduplication
Text-Roberta	49.4	21.3
Vision-ViT	96.3	29.4
Vision-ResNet	63.1	28.9

5.6 Compatibility With Pruning And Quantization

Pruning and Quantization are two popular technologies to compress deep learning models. They are orthogonal to our proposed deduplication algorithm. We demonstrate the compatibility of these technologies by combining deduplication with each of them as shown in Tab. 12. When combining pruning or quantization with deduplication, a ResNet152 is first pruned or quantized and then deduplicated. We did not deduplicate the model first because of two reasons. First, after deduplication, some weights share the same blocks. These shared weights must change at the same time and these 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 and dedupilcate-then-quantize to future work.

Table 12: Compatiblity With Pruning And Quantization

	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

5.7 Deduplicating a Bigger Batch of Models

In the previous sections, the number of models in a batch was five, which is relatively small. However, these models are representative of all cases in our experiments because the difference in accuracy for different training privacy budgets is minimal due to the nature of DP finetuning. Tab. 13 shows the results of extending the natural language inference task to 20 ViT-large models trained on the CIFAR100 dataset with privacy budgets of 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 1.00, 1.50, 2.00, 2.50, 3.00, 3.50, 4.00, 4.50, 5.50, 6.00.

5.8 Comparison of SVT and Naive Privacy Accounting Mechanism for Private Data Validation

In this experiment, we showed that with SVT, the utility of the deduplicated models are much better a naive privacy account mechanism when using private dataset as the validation set during deduplication. The naive privacy account mechanism accumulates privacy cost each time the utility drop exceeds the threshold ([TODO: Hong: we may need a privacy accounting baseline]) . To show its impact in a wide spectrum, we experimented it with base settings in Sec. 5.4, but no longer consider the dataset as public.

5.9 Heterogeneous Model Deduplication

Our deduplication algorithm supports deduplication not only between models of the same architecture (homogeneous) but also between models of different architectures (heterogeneous). In this experiment, we deduplicate ViT-large models and ResNet152 models trained on the same CelebA dataset. Since the CNN layer size of the ResNet is smaller than the linear layer size of the ViT, a block size of 262,144, typical for a CNN layer in the ResNet152 model, is a good choice. We compared the heterogeneous and homogeneous model deduplication, as shown in Tab. 17. The experiments are named using the format "base model-deduplicated model". For example, the experiment "ViT-ResNet" indicates that ViT is the base model and ResNet is the model being deduplicated. These experiments except ResNet-ViT uses Greedy-1 algorithm because Greedy-N algorithms with a small N can achieve better compression ratio when the compression ratio are high. The ResNet-ViT experiment uses Greedy-4 algorithm because with the block size is small compared to a ViT model, and 4 times smaller than the blocks size for the ViT-ViT experiment.

5.10 Inter-Data Model Deduplication

Similar to heterogeneous model deduplication, one can deduplicate models trained on different datasets. For example, model M_1 is trained on dataset D_1 with privacy budget ϵ_1, and is used as the base model to deduplicate model M_2 that is trained on dataset D_2 with privacy budget ϵ_2. As a result, the deduplicated model M_2' has privacy budget ϵ_1 on dataset D_1 and privacy budget ϵ_2 on dataset D_2.

For this experiment, we trained ViT-large models using two different datasets, CIFAR100 and CelebA, with different privacy budgets. We designed two groups of experiments. In the first group, a model trained on the CIFAR100 dataset with privacy budget of 0.5 is set as the based model to deduplicate four models trained on CelebA with privacy budget 0.6, 0.8, 1.0, 2.0, respectively. In the second group, a model trained on the CelebA dataset with privacy budget 0.4 is set as the base model to deduplicate four models trained on CIFAR100 with privacy budget 0.6, 0.75, 1.0, 2.0, respectively. The Binary Search algorithm with minimum deduplication length 10 are applied.

5.11 How Deduplication Affect Online Model Serving Performance?

By deduplicating the weights of the models, the overall model storage footprint can be significantly reduced, minimizing I/O overhead when serving multiple models. In this experiment, we compared the model loading latency between serving the original models and serving the deduplicated models. On an AWS C5a.xlarge instance (7.63 GB memory, 4 vCPUs), at most two ViT large models can be served simultaneously. When serving the original models, each time a query requires a different model, a model currently in memory must be offloaded to disk to make room for the new model. However, with deduplicated models, the entire storage of five models can fit into memory. Thus, when a new query requires a different model, it can be quickly composed from the weights already in memory.

We designed the following workloads: there are 10 models, whose combined storaged before deduplication is 12.0 GB which cannot fit to the memory, and after deduplication is 3.1 GB which can fit to memory; a total of 100 queries generated by two ways, Round Robin and Random. In Round Robin, the queries use model from the first to last and repeat for 10 times. In Random, 100 random integers from one to ten are generated as the model id. The results, shown in Table 19, indicate that for both Round Robin and Random workload, the model loading overhead is significant for model storage without deduplication, whose latency is even longer than inference latency. The model loading latency is greatly reduced with deduplicated model storage.

5.12 A Special Case: Deduplicating DP-finetuned Embedding Vectors

We need to refine and cut this part.

Here, we discuss deduplicating word2vec embeddings layers, which are fine-tuned using DP-SGD on private datasets. We notice that such scenario has very different behaviors in the deduplication process that requires a special treatment.

Table 13: Deduplication of 20 ViT Models Trained on CIFAR100 With Different Privacy Budgets

Models	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	Overall
C.R.(%)	3.6	3.6	3.6	6.7	22.2	16.0	12.9	50.2	28.5	19.1	47.1	22.2	22.2	16.0	28.5	19.1	28.5	40.9	25.4	25.8
Num. of Val.	5	5	5	5	5	7	10	5	8	5	5	5	5	7	8	5	8	7	5	110
Acc. drop(%)	0.76	1.33	1.99	1.98	1.83	1.80	1.84	1.54	1.90	1.87	1.89	1.84	1.90	1.98	1.86	1.91	1.80	1.79	1.93	1.99

Table 14: Comparison Between SVT and Naive Privacy Accounting Mechanism Deduplication of 20 ViT Models Trained on CIFAR100 With Successive Halving

Models	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	Overall
C.R.(%)	3.2	6.7	6.0	9.8	12.9	5.3	12.9	56.5	37.8	17.1	44.0	22.2	14.6	17.1	28.5	17.1	25.3	47.1	28.5	25.6
Num. of Val.	10	6	11	8	6	8	6	7	7	9	7	8	9	9	8	9	6	7	9	150
Acc. drop(%)	0.75	1.23	1.89	1.75	2.39	2.66	1.84	0.92	1.45	2.01	2.34	1.84	2.39	1.80	1.86	1.94	2.04	1.40	1.50	2.66

Table 15: Comparison Between SVT and Naive Privacy Accounting Mechanism Deduplication of 20 ViT Models Trained on CIFAR100 With Binary Search

Models	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	Overall
C.R.(%)	6.7	16.0	16.0	6.7	22.2	19.1	9.8	53.3	53.3	31.6	47.1	28.5	25.3	22.2	31.6	16.0	53.3	68.9	59.6	34.4
Num. of Val.	4	5	5	4	6	5	5	5	5	8	5	5	7	5	8	5	5	8	6	106
Acc. drop(%)	0.63	0.88	1.39	1.98	1.78	1.66	2.15	1.27	0.62	1.41	1.89	1.51	1.67	1.42	1.56	2.05	0.64	0.59	0.46	2.15

Table 16: Comparison of Different Deduplication Algorithms With SVT

	Roberta-base for NLI			ViT for Object Classification)			ResNet for Multi-Attribute Classification)		
	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	35.9	110	0.020	36.1	98	0.021
Greedy-20	28.9	44	0.005	36.1	52	0.026	48.5	52	0.014
Greedy-30	28.9	32	0.003	34.1	44	0.022	42.6	36	0.012
Greedy-40	34.7	24	0.003	32.0	36	0.022	52.3	28	0.010
Binary Search	33.6	24	0.012	34.7	19	0.024	39.7	22	0.018
Successive Halving	32.1	28	0.012	28.9	34	0.024	34.4	30	0.013

Table 17: Heterogeneous and Homogeneous Model Deduplication. The experiments are named using the format "base model-deduplicated model". ResNet-ViT uses Greedy-4 deduplication algorithm, other experiments uses Greedy-1 algorithm.

	C.R.(%)	Num. of Val.	Max. acc. drop
ViT-ResNet	70.6	958	0.019
ResNet-ViT	98.1	1160	0.020
ViT-ViT	29.4	1156	0.019
ResNet-ResNet	28.9	956	0.020

Table 18: Inter-data model deduplication

	C.R.(%)	Num. of Val.	Max. acc. drop
CIFAR100-CelebA	71.38	38	0.0197
CelebA-CIFAR100	92.53	24	0.0197

We first present the key challenges associated with deduplicating the DP-finetuned embedding layers.

Firstly, without DP finetuning, some embedding vectors are not updated if their words/tokens are not present in the finetuning dataset because the gradients of those embedding vectors are all zeros in the training process. In the DP-SGD fine-tuning process, Gaussian noise is added to the gradients at each step, making the zero gradients nonzero. As a result, all embedding vectors are updated after finetuning. This fact makes deduplication of the same pretrained model on different tasks or privacy settings more challenging. For example, a pretrained model can be used for many tasks by being finetuned with domain-specific datasets and we want to deduplicate two such models. Assume that the test dataset has similar distribution as the training set, then deduplicating the embeddings vectors that are not updated or rarely updated have less impact to the overall utility compared to the frequently used embedding vectors. In the case of regular finetuning when many of the embedding vectors are not updated, the blocks in the same position from two models can be deduplicated without sacrificing utility in many cases.

Secondly, because of the noises are culmulated during the training process, the more steps a model is finetuned, the bigger the variance is. For a reasonable big dataset, DP finetuned models have much bigger variance in their weights compared to their non-DP counterpart. In the prior work, the blocks are ordered by their 3rd

Table 19: Online model serving experiment: comparison of model loading and inference latency for 100 queries. Instance type: C5a.xlarge (7.63 GB memory, 4vCPUs), physical batch size: 1. Latency unit: second.

Workload type	Model storage	Model loading latency	Inference latency	Total latency
Round Robin	w/o deduplication	542.08	116.29	658.37
	w deduplication	40.50	116.14	156.64
Random	w/o deduplication	561.66	116.80	678.46
	w deduplication	40.43	116.20	156.64

percentile magnitude and then send to the deduplication process. With DP, the magnitude of the blocks are very similar to each other because they are dominated by the noise. Thus the trick to order blocks according to their magnitude does not work in the DP case.

To tackle the first challenge, we propose a novel deduplication method that minimize the model size after deduplication without sacrifice utility and privacy budget. In a high level, we only run the deduplication process on the frequently used embedding vectors and recover the whole embedding layer by simulating the DP finetuning process. In particular, we first measure and save the variance of the layer. Then we find all the words/tokens frequently used in the training set and group their corresponding embedding vectors (we call them active embeddings) and normalize them to have the same variance as the pretrained weights. When we need to reconstruct the embeddings layer, we first reconstruct the active embeddings and scale them to have the saved variance in the previous step. Then we create a Gaussian noise embedding layer with the saved variance, then replace the active embeddings with the reconstructed active embeddings.

To tackle the second challenge, we propose order the blocks not according to magnitude but according to their frequence that is used in the training data. This is a heuristic that can be used to guide the search algorithm.

In our experiments, we identified that different models trained using similar model architectures and datasets, but with different privacy budgets share a significant portion of similar model parameters that can be deduplicated. For example, we trained three simple language models on the IMDB movie rating dataset using the DP-SGD algorithm. Each model has a word embedding layer, three linear layers, and a regression head. These models differ in their embedding layers. We used wiki500 (1009375×500 dimensional), wiki250 (1009375×250 dimensional), and nnlm128 (973771×128 dimensional), which were all downloaded from the TensorFlow Hub. Then, each of these models is finetuned over the IMDB dataset using DP-SGD (implemented by TensorFlow-privacy) with different noise multipliers (denoted as *nm*), leading to different privacy budgets. Then, we partitioned these models into parameter blocks with equivalent sizes and ordered the blocks based on the L2 norm of the parameters in each block. Then, the algorithm went through the blocks in the ascending order, so that each block was replaced by a previously seen similar block (i.e., the L2 distance between these two blocks is smaller than a threshold 0.1) until the accuracy drop exceeds a threshold of 3.5%. The results are illustrated in Tab. 20, demonstrating impressive compression ratios as low as 2%.

6 RELATED WORKS

Privacy-Preserving Mechanisms in Databases: Several research efforts have focused on privacy-preserving mechanisms within traditional relational databases. Differential privacy [11] has been extensively studied to provide privacy guarantees for statistical query release. Wang et al.[6] proposed a locally differentially private mechanism for releasing synthetic data to protect privacy. In the context of access control, Poddar et al.[44] introduced fine-grained access control mechanisms for data stored in databases, allowing fine-tuning of access rights based on user attributes. Additionally, research has explored secure multiparty computation [61] and secure query processing [64] to enable private computation on encrypted data in the database context.

Privacy in Deep Learning: Deep learning techniques have also seen significant research efforts to address privacy concerns. Differential privacy has been applied to training deep learning models [1, 39, 48, 60] to protect individual training data. Federated learning [29, 40] has emerged as an approach to train models across multiple decentralized devices while preserving data privacy. Privacy-preserving machine learning algorithms such as Secure Multi-Party Computation (SMPC)[10, 24, 33, 53, 54] and Homomorphic Encryption[18, 26] have also been investigated for protecting sensitive data during model training and inference.

Privacy-Preserving Techniques for Integrated Database and Deep Learning: While the integration of databases and deep learning brings opportunities, ensuring data and model privacy in a consistent manner has been relatively unexplored. A few recent works [25, 46, 48] have started investigating privacy-preserving techniques for deep learning models over relational data. Shokri et al.[48] proposed PrivML, a system for privacy-preserving deep learning training using differential privacy in a distributed setting. Rakin et al.[46] introduced a privacy-preserving protocol for deep learning training using Secure Multi-Party Computation (SMPC). Jiang et al. [25] explored differential privacy in federated learning scenarios with privacy constraints for local training data. However, these studies often focus on specific aspects of data or model privacy and do not provide a unified mechanism for integrating database and deep learning privacy.

Model Deduplication Mistique [55] 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 [32] 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 [14], using various blocking techniques to avoid the pair-wise comparison for

Table 20: Comparison of storage (number of blocks) and accuracy drop for different DP-model deduplication

	wiki500 (nm=0.4)	wiki250 (nm=0.41)	wiki250 (nm=0.4)	nnlm (nm=0.41)	nnlm (nm=0.4)
NumBlocks w/o dedup	1014	1014	1014	978	978
NumBlocks w/ dedup	218	214	150	18	16
Accuracy drop	0.6%	3.3%	3.4%	1.8%	1.6%

dissimilar items [2–4, 7, 21, 27, 28]. In addition, various techniques leveraged similarity functions to filter out pairs that have similarity scores below a threshold [57] or used LSH to convert similarity join to an equi-join problem [62]. However, these works are not applicable to numerical tensor data, and they never considered how the deduplication of tensor data will affect the accuracy and privacy of ML applications.

7 CONCLUSIONS

ACKNOWLEDGMENTS

This work was supported by the [...] Research Fund of [...] (Number [...]). Additional funding was provided by [...] and [...]. We also thank [...] for contributing [...].

REFERENCES

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

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

[3] 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.

[4] 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.

[5] 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

[6] Xue Chen, Cheng Wang, Qing Yang, Teng Hu, and Changjun Jiang. 2023. Locally differentially private high-dimensional data synthesis. *Science China Information Sciences* 66, 1 (2023), 112101.

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

[8] 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.

[9] 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

[10] Wenliang Du, Yunghsiang S Han, and Shigang Chen. 2004. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 2004 SIAM international conference on data mining*. SIAM, 222–233.

[11] Cynthia Dwork. 2006. Calibrating noise to sensitivity in private data analysis. *TCC* 5 (2006), 265–284.

[12] 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

[13] 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).

[14] 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.

[15] 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

[16] 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.

[17] 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).

[18] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, and Michael Naehrig. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. *ICML* 1, 2 (2016), 2016.

[19] 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).

[20] 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.

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

[22] 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.

[23] 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.

[24] Geetha Jagannathan and Rebecca N Wright. 2005. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 593–599.

[25] Haoyi Jiang, Chang Liu, Tongxin Su, Xin Chen, and Jiawei Han. 2019. Differentially private federated learning: A client level perspective. *ICDM* (2019), 61–70.

[26] Chinmay Juvekar, Anand Raghunathan, Dan Boneh, Julian McAuley, Andrew Y Ng, and Ion Stoica. 2017. Gazelle: A low-latency framework for secure neural network inference. *SOSP* (2017), 117–132.

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

[28] 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.

[29] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. In *NeurIPS*. 2810–2819.

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

[31] 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

[32] 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.

[33] Yehuda Lindell and Benny Pinkas. 2000. Privacy preserving data mining. In *Annual international cryptology conference*. Springer, 36–54.

[34] 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).

[35] 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).

[36] 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__
[37] Min Lyu, Dong Su, and Ninghui Li. 2016. Understanding the sparse vector technique for differential privacy. *arXiv preprint arXiv:1603.01699* (2016).
[38] Min Lyu, Dong Su, and Ninghui Li. 2017. Understanding the Sparse Vector Technique for Differential Privacy. *Proceedings of the VLDB Endowment* 10, 6 (2017).
[39] H Brendan McMahan, Galen Andrew, Joseph Austerweil, Alex Dawid, Smith Emma, and Himabindu Thakkar. 2018. Generalization in private deep learning. In *ICLR*.
[40] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. 1273–1282.
[41] 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
[42] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013).
[43] 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
[44] Shalmali Poddar, Sruthi Ganta, Prateek Mittal, Ninghui Li, and Alok Desai. 2018. Enabling fine-grained access control on encrypted databases. *VLDB* 11, 2 (2018), 142–155.
[45] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems* 8, 7 (2009), 579–588.
[46] Abu Zaher Md Rakin, Atef Al-Khouri, Tao Li, Chu-Hsing Huang, and Carson K Leung. 2019. Privacy preserving protocol for deep learning. *IEEE Access* 7 (2019), 41108–41117.
[47] 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).
[48] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1310–1321.
[49] 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
[50] Qiaoyue Tang, Frederick Shpilevskiy, and Mathias Lécuyer. 2024. DP-AdamBC: Your DP-Adam Is Actually DP-SGD (Unless You Apply Bias Correction). In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 15276–15283.
[51] 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).
[52] 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).
[53] Jaideep Vaidya and Chris Clifton. 2002. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 639–644.
[54] Jaideep Vaidya, Murat Kantarcıoğlu, and Chris Clifton. 2008. Privacy-preserving naive bayes classification. *The VLDB Journal* 17, 4 (2008), 879–898.
[55] 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.
[56] 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).
[57] 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.
[58] 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.
[59] Minke Xiu, Zhen Ming, Bram Adams, et al. 2019. An exploratory study on machine learning model stores. *arXiv preprint arXiv:1905.10677* (2019).
[60] Hanxiao Xu, Jing Zhu, and Animashree Anandkumar. 2020. GanoObfuscator: Scalable and expressible privacy-preserving GAN learning with obfuscated gradients. In *NeurIPS*. 14281–14293.
[61] Andrew Chi-Chih Yao. 1982. Protocols for secure computations. In *FOCS*. 160–164.
[62] 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.
[63] 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.
[64] Bo Zhao, Hui Su, Junfei Hong, Debin Gao, Ronald Lee, and R Sekar. 2015. SecureDB: A secure database system. *NSDI* (2015).
[65] 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