HEPrune: Fast Private Training of Deep Neural Networks With Encrypted Data Pruning

Yancheng Zhang[1], Mengxin Zheng[1], Yuzhang Shang[2], Xun Chen[3], and Qian Lou[1]

[1]University of Central Florida
[2]Illinois Institute of Technology
[3]Samsung Research America
{yancheng.zhang,mengxin.zheng,qian.lou}@ucf.edu;
yshang4@hawk.iit.edu; xunchen@outlook.com

Abstract

Non-interactive cryptographic computing, Fully Homomorphic Encryption (FHE), provides a promising solution for private neural network training on encrypted data. One challenge of FHE-based private training is its large computational overhead, especially the multiple rounds of forward and backward execution on each encrypted data sample. Considering the existence of largely redundant data samples, pruning them will significantly speed up the training, as proven in plain non-FHE training. Executing the data pruning of encrypted data on the server side is not trivial since the knowledge calculation of data pruning needs complex and expensive executions on encrypted data. There is a lack of FHE-based data pruning protocol for efficient, private training. In this paper, we propose, *HEPrune*, to construct a FHE data-pruning protocol and then design an FHE-friendly data-pruning algorithm under client-aided or non-client-aided settings, respectively. We also observed that data sample pruning may not always remove ciphertexts, leaving large empty slots and limiting the effects of data pruning. Thus, in HEPrune, we further propose ciphertext-wise pruning to reduce ciphertext computation numbers without hurting accuracy. Experimental results show that our work can achieve a $16\times$ speedup with only a 0.6% accuracy drop over prior work. The code is publicly available at https://github.com/UCF-Lou-Lab-PET/Private-Data-Prune.

1 Introduction

Machine learning, especially deep neural networks, has been widely applied in various domains, including healthcare [1], finance [2], and so forth. Due to the lack of expertise and computational resources, the average user often outsources the training task to the cloud servers in the machine-learning-as-a-service (MLaaS) setting. However, the training data is often highly private and private, and it should not be directly shared because of business, legal, and ethical constraints. Private training enables the cloud server to train machine learning models on encrypted data, where strong privacy guarantees are offered by cryptographic primitives such as FHE [3].

However, FHE-based private training is significantly hindered by its substantial execution time. For example, the encrypted training can be $1 \sim 3$ orders of magnitudes slower than in the plaintext training [4, 5, 6]. The high execution time primarily stems from the transformation of a plaintext dataset into an encrypted format, along with the substitution of straightforward plaintext computations with more resource-intensive ciphertext operations. Reducing the number of data samples used for training could greatly accelerate the process. To optimize plaintext datasets, methods like dataset pruning are commonly employed, where the model trainer has access to both the dataset and model.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-1616

Dataset pruning involves analyzing data samples, calculating an importance score for each sample during training iterations, and discarding redundant samples to streamline training. Prior research indicates that dataset pruning can enable machine learning model training on a fraction of the data—sometimes as little as 10%—with minimal to no loss in accuracy [7, 8, 9, 10, 11].

Pruning private datasets in encrypted form to accelerate training remains an open problem due to several challenges. First, calculating importance scores and sorting data samples for removal require complex encrypted computations, such as costly non-linear and sorting operations. For instance, calculating the importance score requires tracking changes in training dynamics [11, 12]—such as logits, gradients, and losses—and computing corresponding distances on encrypted data, all of which involve costly FHE operations. These high overheads that may offset—or even surpass—the potential benefits of dataset pruning. Second, plaintext-level data pruning used in the prior plaintext data pruning [7, 8, 9, 10, 11] does not always yield execution and ciphertext reduction benefits, as a single ciphertext often contains multiple plaintext samples within its slots. Therefore, effective ciphertext pruning is critical to enable faster private training.

In this paper, we propose *HEPrune*, the first encrypted data pruning framework for private training. We begin by constructing an encrypted data pruning method using pure FHE. To accelerate FHE-enabled data pruning, we introduce a homomorphic encryption-friendly score (HEFS) to effectively evaluate the importance of data samples. To avoid costly FHE-based sorting, we propose client-aided masking (CAM) to identify less important samples while ensuring data privacy. To bridge the gap between sample-wise importance scores and ciphertext-wise data, we develop a ciphertext-wise pruning (CWP) technique to reduce the number of ciphertexts during private training. We conduct experiments in both training-from-scratch and transfer-learning settings. The proposed encrypted data pruning framework can accelerate private training by $16\times$ with only a 0.6% accuracy drop.

2 Background

2.1 Fully Homomorphic Encryption-Enabled Private Training

Fully Homomorphic Encryption (FHE)[3] is a cryptographic primitive that supports arbitrary computation on encrypted data. An FHE workflow involves four functions: client-side `KeyGen`, `Enc`, `Dec`, and server-side `Eval`. For `Eval`, FHE natively supports homomorphic addition, subtraction, and multiplication between ciphertexts and plaintexts, denoted as $\boxplus$, $\boxminus$, and $\boxtimes$, respectively. Non-linear functions such as the comparison function, max function, SoftMax function, and square root function can be approximated using polynomials. We denote these functions as HE.Cmp[13, 14], HE.Max[14], HE.SoftMax[6, 15], and HE.Sqrt[16], respectively. FHE enables confidential training on encrypted datasets. While early works focused on training simple models on small datasets, such as linear regression and logistic regression models[17, 18, 19, 20], recent research has advanced towards training deep neural networks (DNNs) using HE. Although HE provides theoretical data privacy guarantees, privately training DNNs on large encrypted datasets typically incurs substantial computational overhead. For example, training a dense model for one step on a batch of 60 MNIST samples can take over 1.5 days in FHESGD [4].

A line of research has been proposed to accelerate HE-enabled private training. Glyph [5] incorporates two HE schemes, TFHE and BGV, to enable private training in the transfer learning setting, allowing training of an MNIST mini-batch in only 0.04 hours. The most recent work, HETAL [6], achieves private training within 1 hour on the MNIST and CIFAR-10 datasets by using optimized matrix multiplication and GPU acceleration. While other cryptographic tools, such as Multi-Party Computation (MPC), can also achieve private training, most existing MPC approaches [21, 22, 23, 24] rely on a non-colluding server assumption, where multiple non-colluding servers are required to ensure security. Additionally, MPC-based methods typically incur significant communication overhead; training a LeNet model on the MNIST dataset, for instance, can generate approximately $500\ GB$ of communication overhead [24]. In contrast, FHE-enabled private training does not require the non-collusion assumption and incurs significantly less communication overhead.

2.2 Dataset Redundancy and Pruning

Not all data samples contribute equally to DNN training [25]; some samples are less important and can be pruned during training. Training on only a subset of data can effectively reduce computational

overhead while achieving generalization performance comparable to training on the full dataset [10, 11]. Several methods have been proposed to identify the most informative data samples. Among these, score-based methods [7, 8, 9, 10] are widely used for their simplicity and effectiveness. These methods compute sample-wise importance scores during training and select the most informative samples based on these scores.

The importance score can be as simple as the entropy loss [8, 10] for each sample. Another commonly used score is the forgetting score [7], which counts the number of "forgetting events" that occur for each sample during training, where a forgetting event is defined as a change in the model's prediction for the same sample between two consecutive epochs. GraNd [9] uses the ℓ_2-norm of the sample-wise gradient to represent sample importance, while ELN2 [9] approximates GraNd by calculating the ℓ_2-norm of the error vector. Additional methods for identifying sample importance include submodularity-based methods like GraphCut [26] and gradient-matching-based methods like GradMatch [12]. These methods typically start with an empty set and gradually add samples, in contrast to score-based methods, where sample importance scores are computed simultaneously and independently.

Data pruning methods are generally divided into static and dynamic pruning methods. Static pruning refers to data pruning performed once before training [7, 8, 9, 26], while dynamic pruning involves pruning data every few epochs during training [12, 11, 10]. Although these methods were designed with different motivations and target settings, they can be adapted for use in either setting. For instance, [8] uses entropy loss in a static setting, while [11, 10] apply it in a dynamic setting. Similarly, dynamic pruning methods like [26, 12] can be adapted to static settings, as demonstrated by [27], without losing effectiveness. In this paper, we focus on dynamic pruning.

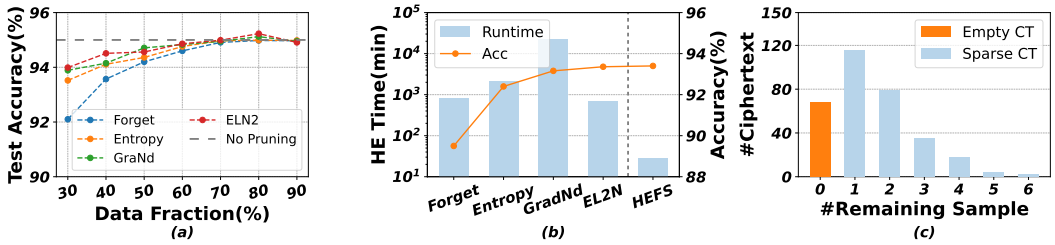

Figure 1: (a). Test accuracy under various pruning fractions in plaintext dataset (ResNet-18 on CIFAR-10). (b) The encrypted training overhead and accuracy of different data pruning methods. (c) The distribution of data sample numbers in a ciphertext after data pruning. Only a small fraction of ciphertexts are empty with a naïve sample-wise pruning.

Motivation. FHE-enabled private training provides a strong privacy guarantee for user data. However, private training remains significantly slower in runtime compared to unencrypted training. One primary reason for this is the large dataset size typically used for training, often comprising tens of thousands of samples. As shown in Figure 1(a), removing a fraction of data during training has minimal impact on model accuracy: even with only 30% of the data, the accuracy drop is within 2%. However, directly applying plaintext data pruning methods to private training might not only fail to accelerate the process but could actually extend the training time. As shown in Figure 1(b), computing importance scores with existing methods—such as the forgetting score [7], entropy score [11, 10], GraNd [9], and EL2N [9]—can introduce prohibitive overhead. This inefficiency arises because FHE natively supports only addition and multiplication, while existing data pruning methods often require a range of non-linear operations. Although these non-linear operations can be implemented in FHE through lookup tables or approximations, these computations are extremely costly in FHE, potentially negating the benefits of data pruning. Furthermore, as illustrated in Figure 1(c), although existing methods can effectively prune samples from datasets, they do not necessarily reduce the number of ciphertexts in private training. This is because, during private training, a single ciphertext can contain multiple samples. Naïvely applying sample-wise data pruning can lead to a large number of sparse ciphertexts that cannot be excluded from training.

Threat Model. We consider a private training scenario where a client outsources model training to a cloud server. Our protocol is designed to allow a client to outsource model training while preserving the privacy of their data and model weights. The client's private dataset and model weights are treated as intellectual property that must not be disclosed to the server at any point. We assume the server is

https://doi.org/10.52202/079017-1616

semi-honest, meaning it will follow the protocol specifications but may attempt to gain unauthorized knowledge, such as the dataset and model weights. Sharing certain meta-information about training, such as the model architecture, dataset size, and early stopping signal, is assumed to be safe [6]. Side-channel attacks are beyond the scope of this paper.

3 HEPrune Design

We begin by outlining the pipeline for private training with encrypted data pruning in Section 3.1. In Section 3.2, we provide details of our FHE-based data pruning algorithm. Specifically, we construct an encrypted data pruning baseline to demonstrate how sample importance evaluation and sample removal can be performed using pure FHE operations. We then enhance the efficiency of the data pruning algorithm through FHE-friendly scoring (HEFS) and client-aided masking (CAM). Finally, in Section 3.3, we introduce ciphertext-wise pruning (CWP), which removes sparse ciphertexts and further accelerates the data pruning process.

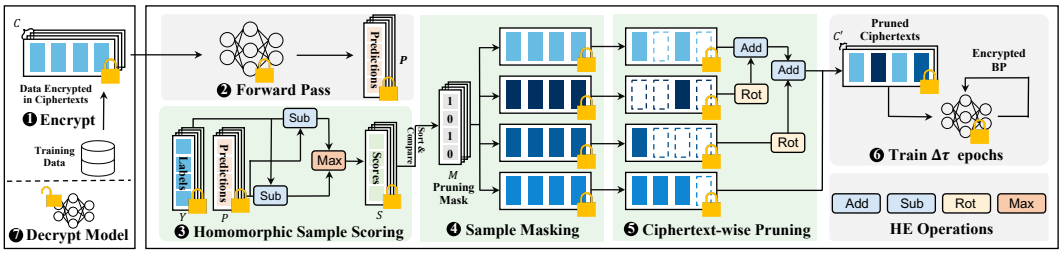

Figure 2: The overall workflow of private training with encrypted data pruning.

3.1 Pipeline of Private Training with Encrypted Data Pruning

It has been widely studied how to perform the gradient descent algorithm in the encrypted state during FHE-enabled private training [4, 5, 6]. However, it remains unclear how data pruning can be incorporated into existing private training frameworks. We illustrate our FHE-based encrypted data pruning framework in Figure 2. During private training, ❶ the client first encrypts the dataset with FHE and sends the encrypted dataset to the server. From this point forward, the dataset will never be decrypted. We refer to the encrypted data samples as ciphertexts. To identify the most informative subset of samples, ❷, the server first runs the forward pass algorithm on all ciphertexts to obtain the encrypted prediction vector P. ❸ With P and the encrypted ground truth label Y, the server computes the sample-wise importance score $S = \mathcal{H}(x)$. We propose an HE-friendly importance score in Section 3.2. ❹ With the importance score, the server can decide which samples to prune. This is achieved by sorting the importance score and finding the threshold importance score $\mathcal{H}_t$ corresponding to the pruning ratio [7]. A pruning mask is generated by comparing the importance score with the threshold as $M = \mathbb{1}\{\mathcal{H}(x) < \mathcal{H}_t\}$. The unimportant samples can be masked by simply multiplying the ciphertexts with the corresponding masks. ❺ To fully reduce the number of ciphertexts, the server performs ciphertext-wise pruning according to the current pruning mask. ❻ The server performs a backpropagation algorithm on the pruned ciphertexts for $\Delta\tau$ epochs, and then ❷ starts a new round of the data pruning algorithm to update the training subset. ❼ At the end of private training, the server sends the encrypted model back to the client, and the client decrypts the model.

We formalize the above workflow into an encrypted data pruning protocol in Figure 5 in Appendix A. Our protocol is the first of its kind to introduce data pruning to private training and is an enhanced private training framework with encrypted data pruning extension over existing works [4, 5, 6]. Additionally, our protocol can work in both the transfer learning setting [5, 6] and training-from-scratch setting [4]. Our protocol is fully compatible with the early stop techniques adopted in [6]. As in [6], sending meta information during training is allowed in our protocol, such as the logits, early stop signals, and importance scores. The privacy of data and models is strongly guaranteed as they are encrypted and never decrypted by the server. We first instantiate a baseline of encrypted data pruning using only FHE operations in Section 3.2, and then propose HEFS and CAM to optimize the overhead of the encrypted data pruning algorithm. In Section 3.3, we propose CWP to effectively reduce the number of ciphertexts involved in training and thus boost the efficiency of private training.

https://doi.org/10.52202/079017-1616

3.2 HE-enabled Data Pruning

While FHE can support arbitrary computation on encrypted data, computing non-linear functions is typically expensive. Applying existing data pruning methods to private training can lead to additional overhead that negates or even exceeds the benefits of performing data pruning. For the entropy-based methods [8, 11, 10], although the entropy loss is typically considered cost-free in the plaintext, it is not explicitly computed during private training. Instead, we compute the gradient directly without calculating the loss itself, as illustrated in Figure 5, step 3(b). Computing the entropy loss in FHE requires approximating the logarithm function, which is computationally expensive. The forgetting score [7] is simpler to compute, requiring only comparison operations between the current prediction and prediction from the last epoch. However, the forgetting score cannot be easily made dynamic, as it usually requires calculations over the full dataset for multiple epochs. The most viable existing data pruning method is the EL2N [9], which simply utilizes the prediction P and label Y. Although originally proposed as a static pruning method, we find that EL2N also remains effective in a dynamic setting, similar to the entropy score [10]. We first instantiate an FHE-enabled data pruning algorithm, detailed in Algorithm 1, based on the dynamic version of EL2N (HE2LN). Subsequently, we demonstrate that even EL2N remains computationally expensive in the encrypted state and propose the HE-friendly score (HEFS) and client-aided masking (CAM) to accelerate the encrypted data pruning.

Algorithm 1: Homomorphic Data Pruning

Input : The encrypted dataset $\bar{D} = \{\bar{X}_i, \bar{Y}_i\}_{i=0}^{C-1}$ and the model $\mathcal{M}$ with weights $\bar{W}$, the pruning ration p, the sample number N and the target class number n.

Output : The pruned encrypted dataset $\bar{D}' = \{\bar{X'}_i, \bar{Y'}_i\}_{i=0}^{C'-1}$, where $C' \leq C$.

for $i \leftarrow 0$ **to** $C-1$ **do**
- $\bar{P}_i \leftarrow \mathsf{HE.SoftMax}(\mathcal{M}(\bar{X}_i; \bar{W}))$; // Prune.Eval
- $\bar{err}_i \leftarrow \bar{P}_i \boxminus \bar{Y}_i$; // compute the error vector
- $\bar{err}_i \leftarrow \bar{err}_i \boxtimes \bar{err}_i$;
- $\bar{err\_sum}_i = \bar{err}_i$;
- **for** $j \leftarrow 0$ **to** $\log n - 1$ **do**
  - $\bar{err\_t}_i \leftarrow \mathsf{HE.Rotate}(\bar{err\_sum}_i, 2^j)$;
  - $\bar{err\_sum}_i \leftarrow \bar{err\_sum}_i \boxplus \bar{err\_t}_i$;
- $\bar{score}_i \leftarrow \mathsf{HE.Sqrt}(\bar{err\_sum}_i)$; // compute the importance score

$\bar{s\_score} \leftarrow \{\}$; // Prune.Remove

$B \leftarrow N/\ C$;

for $i \leftarrow 0$ **to** $C-1$ **do**
- **for** $j \leftarrow 0$ **to** $B-1$ **do**
  - $\bar{s\_score}_{iC+j} \leftarrow \mathsf{HE.Rotate}(\bar{score}_i, j)$; // extract sample-wise scores

for $i \leftarrow 0$ **to** $N-1$ **do**
- **for** $j \leftarrow 0$ **to** $N-i$ **do**
  - $\bar{less} \leftarrow \mathsf{HE.Cmp}(\bar{s\_score}_i, \bar{s\_score}_j)$; // sort the sample-wise scores
  - $(\bar{s\_score}_i) \leftarrow (\bar{less} \boxtimes s\_score_j) \boxplus ((1 \boxminus \bar{less}) \boxtimes s\_score_i)$;
  - $(\bar{s\_score}_j) \leftarrow (\bar{less} \boxtimes s\_score_i) \boxplus ((1 \boxminus \bar{less}) \boxtimes s\_score_j)$;

$threshold\_index \leftarrow N \times p$;

$\bar{\theta} \leftarrow \bar{s\_score}_{threshold\_index}$;

for $i \leftarrow 0$ **to** $C-1$ **do**
- $\bar{mask} \leftarrow \mathsf{HE.Cmp}(\bar{score}_i, \bar{\theta})$; // remove unimportant samples
- $(\bar{X}_i, \bar{Y}_i) \leftarrow (\bar{mask} \boxtimes 0) \boxplus ((1 \boxminus \bar{mask}) \boxtimes (\bar{X}_i, \bar{Y}_i))$;

return $\bar{D}' \leftarrow \{\bar{X'}_i, \bar{Y'}_i\}_{i=0}^{C-1}$

HEL2N Baseline. The EL2N score is defined as $\mathcal{H}(x) = \mathbb{E}_{w_t} \|p(x; w_t) - y\|_2$, where $p(x; w_t)$ is the prediction vector for sample x and y the corresponding ground truth label. In essence, the EL2N score is the ℓ_2-norm of the error vector, which can be computed via pure FHE operation as shown in Algorithm 1. We first compute the prediction vector P by encrypted forward pass. The $\mathsf{HE.SoftMax}$ in the forward pass is evaluated by polynomial approximation [15] and domain extension technique [28]. Computing the forward pass in the encrypted state is the same as existing private training frameworks [6]. With the encrypted prediction vector P and the ground truth label

https://doi.org/10.52202/079017-1616

Y, we can compute the EL2N score homomorphically. Specifically, we first compute the sum of squares over the error vector, which requires homomorphic subtraction, multiplication, and rotation. Then we take the square root of sum of squares to compute the ℓ_2-norm of the error vector. While HE.Sqrt can be implemented via Newton iterative algorithm [16], the overhead is prohibitive as the Newton iterative algorithm needs ~ 10 iterations to compute an accurate square root. Additionally, the Newton iterative algorithm in FHE involves considerable ciphertext-ciphertext multiplication. This leads to a large number of additional relinearization and bootstrapping operations. A single square root over a ciphertext can take up to 2 minutes.

HE-friendly Score. Simple as the EL2N score is, using it to evaluate the importance of data samples in the encrypted state can still incur prohibitive overhead due to its complex non-linearity. We propose an HE-friendly importance score (HEFS) to address this issue. We first derive the formulation of HEFS, and then demonstrate how it can be computed via pure FHE operation. To determine how a single data sample affects the training, we can quantify the importance of a sample by the difference of the gradient before and after removing a sample. Denote the gradient of a sample (x, y) over the weights at time t as $g_t(x, y) = \nabla_{w_t}\ell(p(w_{t-1}, x), y)$. Given a minibatch of B samples $S = \{(x_i, y_i)\}_{i=0}^{B-1}$, the importance of a sample can be quantified by the difference of the time derivative of the loss function, Δ_t , before and after removing the sample from the minibatch. The difference of the derivative is bounded by the sample's gradient [9]. Specifically, let $S_{\neg k} = S \setminus \{(x_k, y_k)\}$ be the set after removing a certain sample (x_k, y_k). For $\forall(x_i, y_i) \in S$ and $i \neq k$, it holds that

$$\|\Delta_t((x_i, y_i), S) - \Delta_t((x_i, y_i), S_{\neg k})\|_2 \leq c\|g_t(x_k, y_k)\|_2 \tag{1}$$

EL2N approximates the ℓ_2-norm of the gradient by the ℓ_2-norm of the error vector. We further streamline the EL2N score using the ℓ_1-norm. More formally, we define the HEFS as:

$$\mathcal{H}(x) = \mathbb{E}_{w_t}\|p(x; w_t) - y\|_1 \tag{2}$$

HEFS can be efficiently computed during private training. Specifically, the circuit for HEFS consists of only two FHE subtractions and one max operation, which can be computed as:

$$score = \mathsf{HE.Max}((Y \boxminus P), (P \boxminus Y)) \tag{3}$$

In the above equation, the homomorphic subtraction $\boxminus$ is significantly faster than other FHE operations. HE.Max can be effectively computed via HE.Cmp. While the HE.Max operation has the same time complexity as HE.Cmp, the concrete runtime of HE.Max is even more efficient, typically $1.5 \sim 2\times$ faster under the same parameter setting [14]. The proposed HEFS is a close approximation to the original EL2N score, which guarantees the effectiveness of the HEFS-based data pruning. We show the accuracy of the data pruning using HEFS in Section 4.

Algorithm 2: Client-aided Sample Pruning

Input : The pruning ratio p. The server holds the encrypted dataset $\bar{D} = \{\bar{X}_i, \bar{Y}_i\}_{i=0}^{C-1}$ and encrypted score $\bar{score}$ and the client holds the secret key SK.

Output : The pruned encrypted dataset $\bar{D'} = \{\bar{X'}_i, \bar{Y'}_i\}_{i=0}^{C'-1}$, where $C' \leq C$.

Client:
- $score \leftarrow \mathsf{HE.Dec}(\bar{score}, \mathsf{SK})$;
- $\theta \leftarrow \mathsf{QuickSelect}(score, p)$;
- $mask \leftarrow \mathsf{Compare}(score, \theta)$;
- Sends $mask$ to the server;

Server:
- $\bar{D'} \leftarrow \bar{D}$;
- **for** $i \leftarrow 0$ **to** $C - 1$ **do**
  - **if** $mask_i == 0$ **then**
    - $\bar{D'} \leftarrow \bar{D'} \setminus \{(\bar{X}_i, \bar{Y}_i)\}$; `// remove an empty ciphertext`
  - **else**
    - $(\bar{X}_i, \bar{Y}_i) \leftarrow mask_i \boxtimes (\bar{X}_i, \bar{Y}_i)$; `// remove unimportant samples`

return $\bar{D'}$

Client-aided Masking. After evaluating the importance of each data sample, we need to remove the less informative ones from the training set. This requires the server to sort all importance scores homomorphically to determine the threshold of important scores, $\bar{\mathcal{H}}$. Given a dataset with N data samples, $O(N^2)$ homomorphic comparisons are needed. Since N is typically large for machine learning model training, such sorting incurs prohibitive overhead in the encrypted state, which can offset the benefits of data pruning or even prolong the total training time. To effectively identify and remove the less important data samples, we propose client-aided masking (CAM) in Algorithm 2. We offer an analysis on the computation, communication overhead, and security implications as follows.

Efficiency. In contrast to the heavy server-side homomorphic sorting, finding $\mathcal{H}_t$ is extremely fast on the client's side, with only $O(N)$ runtime via the quick select algorithm. In practice, this process takes only $15\ ms$ on the CIFAR-10 dataset. Additionally, the communication overhead is minimal.

For the CIFAR-10 dataset with $43,750$ training samples, only 2 CKKS ciphertexts are needed to transfer the encrypted importance scores when the slots are fully utilized. This incurs only 4 MB of communication overhead. For comparison, the early stopping signals used to determine early stopping incur 18 MB communication overhead when transferring the encrypted logits [6].

Security. Directly asking the client to decrypt and reveal the model weights and datasets to the server clearly breaches the client's privacy and is therefore prohibited. However, exchanging meta information about training does not directly compromise private information and is typically permitted. For instance, HETAL [6] allows the server to send the logits of the validation set to the client, who then computes the loss and determines whether to stop training early. Such exchanges of meta information are crucial to ensure the effectiveness of private training. Further details can be found in Appendix B.

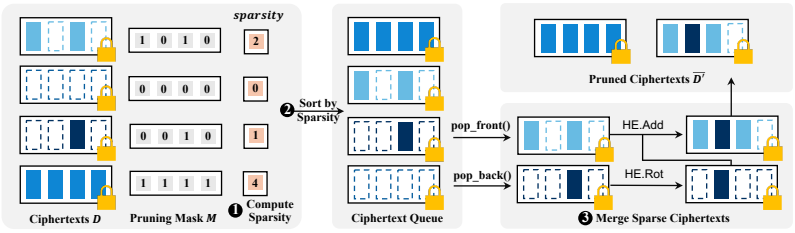

Figure 3: Example of ciphertext-wise pruning.

3.3 Ciphertext-wise Pruning

While the HEFS and CAM make the encrypted data pruning algorithm much more practical, the resulting ciphertexts remain largely sparse, thus limiting the training time acceleration. To this end, we propose ciphertext-wise pruning to effectively reduce the number of ciphertexts involved in training, as illustrated in Figure 3, which further boosts the efficiency of the private training. We detail the steps of ciphertext-wise pruning in Algorithm 3. Without loss of generality, we refer to all the slots occupied by a single sample within a ciphertext as a sample slot. Once the server obtains the pruning mask, ciphertext-wise pruning can be performed without client involvement.

Algorithm 3: Ciphertext-wise Pruning

Input : The encrypted dataset $\bar{D} = \{\bar{X}_i, \bar{Y}_i\}_{i=0}^{C-1}$ and pruning masks $mask = \{M_i\}_{i=0}^{C-1}$, where $M_i \in \{0,1\}^B$ and B is the number of samples in each ciphertexts.

Output : The pruned encrypted dataset $\bar{D}' = \{\bar{X'}_i, \bar{Y'}_i\}_{i=0}^{C'-1}$, where $C' \leq C$.

$\bar{D}' \leftarrow \emptyset$;
for $i \leftarrow 0$ **to** $C-1$ **do**
  $spasity_i \leftarrow B - \sum_{j=0}^{B-1} M_i[j]$; `// compute the ciphertext-wise sparsity`
$ct\_queue \leftarrow Sort(spasity, \bar{D}, mask)$;
$ct\_queue \leftarrow \mathsf{Trim}(ct\_queue)$;
while $ct\_queue.is\_not\_empty()$ **do**
  $ct_{front} \leftarrow ct\_queue.pop\_front()$;
  $ct_{back} \leftarrow ct\_queue.pop\_back()$;
  $slot_{empty} \leftarrow \mathsf{FZero}(ct_{front})$;
  $slot_{used} \leftarrow \mathsf{FOne}(ct_{back})$;
  $k \leftarrow slot_{used} - slot_{empty}$;
  $ct_{align} \leftarrow \mathsf{Mask}(ct_{back})$;
  $ct_{front} \leftarrow ct_{front} \boxplus \mathsf{HE.Rot}(ct_{align}, k)$;
  if $ct\_queue.is\_empty() \vee ct_{front}.sparsity == 0$ **then**
    $\bar{D}' \leftarrow \bar{D}' \cup \{ct_{front}.ct\}$;
  else
    $ct\_queue.push\_front(ct_{front})$;
  $ct\_queue.push\_back(ct_{back})$;
return $\bar{D}'$

We denote the number of samples in each ciphertext as B and M_i is a Boolean array that indicates whether each of the B samples should be pruned. ❶ The server first computes the sparsity of each ciphertext, which is done by simply counting the number of 0-s in each M_i. ❷ After computing the ciphertext-wise sparsity, the server can sort the ciphertexts along with their corresponding masks. We represent the sorted ciphertexts in a dequeue. To perform ciphertext-wise pruning, the server first identifies two ciphertexts ct_{front} and ct_{back} from the queue. ❸ After removing the full ciphertexts and empty ciphertexts from the queue via $\mathsf{Trim}()$, we then leverage the most sparse ciphertext ct_{back} to fill in the empty slots in the least sparse ciphertexts ct_{front}. The key step is to align samples in ct_{back} with the empty slots in ct_{front}. The function $\mathsf{FZero}()$ returns the index of the first empty sample slot $slot_{empty}$ in ct_{front}, sets the corresponding mask to 1, and decreases the sparsity of ct_{front} by 1. Similarly, $\mathsf{FOne}()$ returns the first non-empty sample slot $slot_{used}$ in ct_{back} and sets the corresponding mask and sparsity. By taking the difference of $slot_{used}$ and $slot_{empty}$, the server can

https://doi.org/10.52202/079017-1616

determine how much the ct_{back} should be rotated to align with ct_{front}. We mask out the other slots in ct_{back} and obtain ct_{align} and keep only the first sample slot $slot_{used}$. This ensures only the empty sample slot in ct_{front} is filled and the non-empty slots in ct_{front} will not be corrupted. Then, ct_{align} is rotated and added to ct_{front} to fill in the empty slot in ct_{front}. After merging the two ciphertexts, if ct_{front} is full or there are no remaining ciphertexts to merge, the server adds it to the final set $\bar{D}'$. We defer the details about $\mathsf{Trim}()$, $\mathsf{FZero}()$,$\mathsf{Fone}()$ and $\mathsf{Mask}()$ to the Appendix D. We remark the pruning mask generated by CAM is not encrypted, while enables efficient operations related to the pruning mask without invoking heavy FHE computation.

4 Experiments

4.1 Experimental Setup

Models and Datasets. To demonstrate the generalizability, we evaluate the proposed encrypted data pruning methods in two settings, the transfer learning setting as in HETAL [6] and training-from-scratch setting as in FHESGD [4]. For the transfer learning setting, we adopt the same feature extractors as [6], the pre-trained ViT Base [29] model and the pre-trained MPNet Base [30] model. The client first extracts samples into a 768-dimension feature and then encrypts the features. The server performs private training on the encrypted features. For the training-from-scratch setting, the client encrypts the raw dataset directly, and the server performs private training on the encrypted dataset. We use a 3-layer MLP, with two hidden layers of dimension 128, which is a widely used structure in the private training setting [4, 21]. We perform encrypted data pruning on four widely used image datasets: MNIST [31], CIFAR-10 [32], Face Mask Detection [33], DermaMNIST [34] for image classification and one audio dataset SNIPS [35] for sentiment analysis. We partition the datasets as training, validation, and test set. The size of the training set and validation set is $7:1$. We defer the number of samples in each set in the Appendix C.

System Setup and Implementation. We use the RNS version of the CKKS [36, 37] scheme for homomorphic encryption. We use the bootstrapping method for CKKS in [38]. Our implementation is based on the OpenFHE [39] library. For HE parameters, we set the cyclotomic ring dimension as $N = 2^{16}$ and ciphertext modulus 1555 bits to guarantee a security level of 128-bit under the Homomorphic Encryption Standard [40]. Each ciphertext has $N/2 = 32768$ slots and we set the multiplicative depth as $L = 12$. All experiments were conducted using an AMD Ryzen Threadripper PRO 3955WX processor operating at 2.2GHz, equipped with 125GB of memory. We use the Single-Instruction-MultipleData (SIMD) technique [41] to fully utilize the ciphertext slots and amortize the cost of homomorphic operations. Under SIMD, multiple data samples can be coded into one ciphertext. We adopt the most efficient encoding methods proposed in [6, 42]. For nonlinear operations like $\mathsf{SoftMax}$ and ReLU, we adopt approximation-based methods. We report the CPU version of HETAL, as their GPU implementation is not publicly available.

4.2 Results

End-to-end performance. As shown in Table 1, we compare the end-to-end training time and accuracy on five datasets with HETAL [6] and an unencrypted baseline. HETAL is known to be the most efficient full-data private training framework. For fair comparison, we unify the batch size as 128. For HETAL, we maintain the security parameters used in their original paper. For the proposed encrypted data pruning method, we set the pruning ratio as $p = 0.9$, i.e., only 10% of the data remains for training in each epoch. The total training time is reduced by $\sim 6.6\times$ across datasets. The accuracy drop is as small as 0.25% on the Face Mask Detection dataset. With encrypted data pruning, the accuracy is even 0.14% higher than the unencrypted baseline and HETAL.

Table 1: End-to-end comparison across different datasets The pruning ratio is set as $p = 0.9$.

Method		MNIST	CIFAR-10	Face Mask Detection	DermaMNIST	SNIPS
Unencrypted	Acc(%)	$95.69_{\pm 0.02}$	$96.62_{\pm 0.02}$	$95.46_{\pm 0.06}$	$75.91_{\pm 0.11}$	$94.43_{\pm 0.05}$
HETAL	Acc(%)	$96.27_{\pm 0.02}$	$96.57_{\pm 0.04}$	$95.46_{\pm 0.05}$	$76.06_{\pm 0.18}$	$95_{\pm 0.08}$
	Runtime(h)	276.75	293.3	32.88	101.55	113.7
Ours	Acc(%)	$95.54_{\pm 0.05}$	$96.31_{\pm 0.06}$	$95.21_{\pm 0.06}$	$75.86_{\pm 0.15}$	$95.14_{\pm 0.08}$
	Runtime(h)	41.89	44.76	5.02	15.5	17.36

Table 2: Effectiveness of the proposed method. We tested different private training methods on the CIFAR-10 dataset, with a pruning ratio $p = 0.9$. Communication is the size of logits and importance score ciphertexts server sends to client for early stopping and data pruning.

Method	Accuracy(%)	Runtime(h)	Speedup	Communication(MB)
Full Data(HETAL)	$96.57_{\pm 0.04}$	293.3	$\times 1$	18.1
Prune Baseline	$95.98_{\pm 0.12}$	488.91	$\times 0.6$	18.1
+Client Aided	$96.16_{\pm 0.07}$	196.91	$\times 1.49$	22
+HEFS	$96.31_{\pm 0.06}$	105.57	$\times 2.78$	22
+Ciphertext-wise Pruning	$96.31_{\pm 0.06}$	44.76	$\times 6.55$	22

Effectiveness of the proposed methods. In Table2, we demonstrate the effectiveness of each proposed technique. We start by training on the full CIFAR-10 dataset using the framework of HETAL. For the encrypted data pruning methods, we fix the pruning ratio as $p = 0.9$. In the prune baseline, we naively apply plaintext data pruning method to HETAL, which prolongs the total training time. Sorting 43750 importance score in CIFAR-10 can take more than 200 hours, which offsets the benefits of data pruning. By incorporating the client-aided masking, the server can remove the unimportant samples more effectively. The client-aided methods speed up the private training by $1.49\times$ with a 0.41% accuracy drop. The communication cost increases slightly by approximately $1.2\times$. Yet, as the EL2N score involves costly square root computation, the speed-up is modest. By incorporating HEFS, we improve the overhead during importance score computation. With HEFS, the private training can be accelerated by $2.78\times$. The HEFS also achieves a $\sim 0.2\%$ higher accuracy. When ciphertext-wise pruning is applied, the runtime speed-up is the most significant, achieving a $6.5\times$ speed-up over HETAL with only 0.2% accuracy drop.

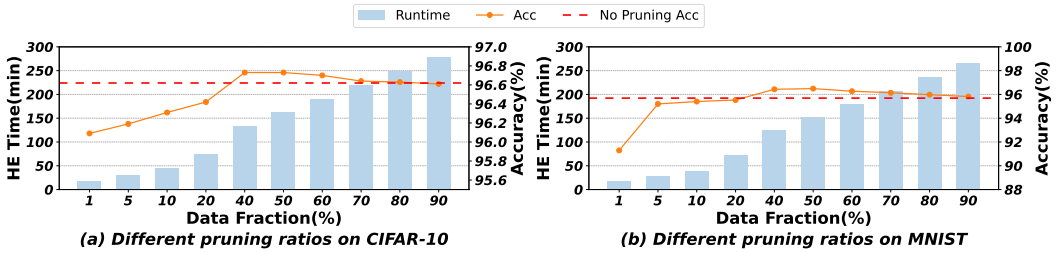

Figure 4: Private training time and accuracy with different fractions of data of (a) CIFAR-10 dataset and (b) MNIST dataset

Ablation on the pruning ratio. In Figure 4, we demonstrate the effectiveness of the proposed encrypted data pruning method under different data pruning ratios. In Figure 4 (a), we demonstrate the training time and accuracy achieved using different fractions of the CIFAR-10 dataset. Surprisingly, we find that even using only 1% of the data, the accuracy drop is only around 0.6%, and the training can be sped up by $\sim 16\times$. Training with the $40\% \sim 70\%$ of the data even leads to higher accuracy than training with the full dataset. The same phenomenon is observed in plaintext data pruning [9]. This is because some noisy or low-quality samples are excluded from training while retaining enough informative samples are maintained. We note that we can achieve $\sim 2.2\times$ speed up when training with 40% of the data without any loss of accuracy on the CIFAR-10 dataset. In Figure 4 (b), we experiment with the MNIST dataset and observe a similar trend. On the MNIST dataset, using only 1% of the data achieves 91.29% accuracy with $\sim 15\times$ speed-up. Yet, using 5% of the data achieves a significantly higher accuracy of 95.2%, which is only 0.5% lower than training with the full dataset.

Table 3: Privately Training an MLP from scratch under different data pruning ratios.

Method		1%	5%	10%	20%	40%	50%	60%	70%	80%	90%
Acc.	Acc(%)	93.23	97.12	97.39	98.38	98.52	98.55	98.5	98.48	98.45	98.45
	$\Delta Acc.$	-5.26	-1.37	-1.1	-0.11	+0.03	+0.06	+0.01	-0.01	-0.04	-0.04
Runtime(h)	Time(h)	32.25	110.61	208.56	404.46	796.26	992.16	1188.06	1383.94	1579.88	1775.72
	speed up	60.8×	17.2×	9.4×	4.8×	2.5×	1.9×	1.7×	1.4×	1.2×	1.1×

Training from scratch. We show the performance of encrypted data pruning in the training from scratch setting in Table 3. We train a 3-layer MLP on the MNIST dataset. We set the pruning

https://doi.org/10.52202/079017-1616

frequency as 10 epoch. When training with only 1% data, the end-to-end training time could be $60.8\times$ faster with an accuracy drop of 5.26%. Increasing the training data fraction can effectively improve the test accuracy. When using 40% and 50% fraction of data, the training accuracy is $0.03\% \sim 0.06\%$ higher than training with the full dataset. We remark that warm-start strategy can also improve the test accuracy.

5 Discussion

Broader Impact. In this paper, we propose a framework that enables encrypted data pruning during confidential training. The proposed techniques can accelerate the private training without sacrificing the model accuracy. By incorporating data homomorphic friendly data pruning techniques, our framework makes the HE-enabled private training more practical while ensuring the data privacy.

Limitations. (1) More Model Support. Currently, the proposed methods have only been applied to simple models like MLP. Extending the encrypted data pruning to private training on larger models like CNNs and Transformers can enhance its utility. (2) More Dataset Support. In this paper, we have experimented on relatively small datasets. Extending to larger datasets will enhance its utility.

6 Conclusion

The paper presents a novel framework for encrypted data pruning aimed at enhancing private training of deep neural networks. Plaintext data pruning methods offer limited benefits in encrypted settings due to their lack of optimization for cryptographic processes. Our approach introduces crypto-oriented optimizations, including HE-friendly score, client-aided masking, and ciphertext-wise pruning, which effectively harness the potential of data pruning, achieving up to a 16-fold acceleration in training times without compromising accuracy.

References

[1] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

[2] Matthew F Dixon, Igor Halperin, and Paul Bilokon. *Machine learning in finance*, volume 1170. Springer, 2020.

[3] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[4] Karthik Nandakumar, Nalini Ratha, Sharath Pankanti, and Shai Halevi. Towards deep neural network training on encrypted data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[5] Qian Lou, Bo Feng, Geoffrey Charles Fox, and Lei Jiang. Glyph: Fast and accurately training deep neural networks on encrypted data. *Advances in neural information processing systems*, 33:9193–9202, 2020.

[6] Seewoo Lee, Garam Lee, Jung Woo Kim, Junbum Shin, and Mun-Kyu Lee. Hetal: Efficient privacy-preserving transfer learning with homomorphic encryption. In *International Conference on Machine Learning*, pages 19010–19035. PMLR, 2023.

[7] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

[8] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019.

[9] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607, 2021.

[10] Truong Thao Nguyen, Balazs Gerofi, Edgar Josafat Martinez-Noriega, François Trahay, and Mohamed Wahib. Kakurenbo: Adaptively hiding samples in deep neural network training. *Advances in Neural Information Processing Systems*, 36, 2024.

[11] Ziheng Qin, Kai Wang, Zangwei Zheng, Jianyang Gu, Xiangyu Peng, Zhaopan Xu, Daquan Zhou, Lei Shang, Baigui Sun, Xuansong Xie, et al. Infobatch: Lossless training speed up by unbiased dynamic data pruning. *arXiv preprint arXiv:2303.04947*, 2023.

[12] Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR, 2021.

[13] Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*, pages 221–256. Springer, 2020.

[14] Eunsang Lee, Joon-Woo Lee, Jong-Seon No, and Young-Sik Kim. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Transactions on Dependable and Secure Computing*, 19(6):3711–3727, 2021.

[15] Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *iEEE Access*, 10:30039–30054, 2022.

[16] Hongyuan Qu and Guangwu Xu. Improvements of homomorphic evaluation of inverse square root. *Available at SSRN 4258571*.

[17] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE symposium on security and privacy*, pages 334–348. IEEE, 2013.

[18] Hao Chen, Ran Gilad-Bachrach, Kyoohyung Han, Zhicong Huang, Amir Jalali, Kim Laine, and Kristin Lauter. Logistic regression over encrypted data from fully homomorphic encryption. *BMC medical genomics*, 11:3–12, 2018.

[19] Andrey Kim, Yongsoo Song, Miran Kim, Keewoo Lee, and Jung Hee Cheon. Logistic regression model training based on the approximate homomorphic encryption. *BMC medical genomics*, 11:23–31, 2018.

[20] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, Xiaoqian Jiang, et al. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics*, 6(2):e8805, 2018.

[21] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*, pages 19–38. IEEE, 2017.

[22] Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 35–52, 2018.

[23] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *arXiv preprint arXiv:2004.02229*, 2020.

[24] Jean-Luc Watson, Sameer Wagh, and Raluca Ada Popa. Piranha: A {GPU} platform for secure computation. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 827–844, 2022.

 https://doi.org/10.52202/079017-1616

[25] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[26] Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. In *Algorithmic Learning Theory*, pages 722–754. PMLR, 2021.

[27] Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*, pages 181–195. Springer, 2022.

[28] Jung Hee Cheon, Wootae Kim, and Jai Hyun Park. Efficient homomorphic evaluation on large intervals. *IEEE Transactions on Information Forensics and Security*, 17:2553–2568, 2022.

[29] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[30] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867, 2020.

[31] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

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

[33] Larxel. Face mask detection, 2020. Accessed: May 20, 2024.

[34] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.

[35] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, 2018.

[36] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017.

[37] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full rns variant of approximate homomorphic encryption. In *Selected Areas in Cryptography–SAC 2018: 25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers 25*, pages 347–368. Springer, 2019.

[38] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part I 37*, pages 360–384. Springer, 2018.

[39] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. Openfhe: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915, 2022. `https://eprint.iacr.org/2022/915`.

[40] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption standard. *Protecting privacy through homomorphic encryption*, pages 31–62, 2021.

[41] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, codes and cryptography*, 71:57–81, 2014.

[42] Eric Crockett. A low-depth homomorphic circuit for logistic regression model training. *Cryptology ePrint Archive*, 2020.

[43] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 648–677. Springer, 2021.

 https://doi.org/10.52202/079017-1616

Appendix

A Encrypted data pruning protocol

In this section, we detail the proposed encrypted data pruning protocol. The main overhead associated with performing performing encrypted data pruning is determined by Step 3.(d) in Figure 5.

Parties: Client C, Server S.
Input: C holds a private dataset D, with N samples and labels $\{(X_i, Y_i)\}_{i=0}^{N-1}$.
Output: C learns a trained model $\mathcal{M}$ and S learns nothing.

Protocol:

1: C encrypts the dataset with $\mathsf{HE.Enc}$, and sends the encrypted dataset $\bar{D} = \{\bar{X}_i, \bar{Y}_i\}_{i=0}^{C-1}$ to the server.

2: S initializes $\mathcal{M}$ with random parameters W_0 and chooses the hyperparamerters for training: the learning rate η, pruning ratio p and the pruning frequency $\Delta\tau$.

3: For each epoch τ, repeat the following:

(a) S performs forward pass homomorphically: $P = \mathsf{HE.SoftMax}(\mathcal{M}(X, W))$

(b) S computes gradients and updates model:
$W_{\tau+1} \leftarrow W_\tau + \eta \nabla_W$, where $\nabla_W \mathcal{L}_{CE} = \frac{1}{N}(P - Y)^T X$

(c) S sends P to C, C determines early stop.

(d) If $(\tau \ mod \ \Delta\tau == 0)$, S performs data pruning as follows:

(i) S computes the importance score: $\bar{score} = \mathsf{Prune.Eval}(\bar{P}, \bar{Y})$

(ii) S sends $\bar{score}$ to C, can C computes a pruning mask $mask$.

(iii) S removes unimportant samples: $\bar{D}' = \mathsf{Prune.Remove}(\bar{D}, mask)$

(iv) Continue step 3(a) to train the model on the pruned dataset $\bar{D}' = \{\bar{X'}_i, \bar{Y'}_i\}_{i=0}^{C'-1}$

4: S sends the trained model $\mathcal{M}$ with encrypted optimal weights $\bar{W}^*$ to C. C decrypts the model with $\mathsf{HE.Dec}$ and obtain the final model $\mathcal{M}$ with optimal plaintext weights W^*.

Figure 5: Private Data Pruning Protocol.

B Analysis of Client-aided Masking

In this section, we offer more details about the proposed Client-aided Masking (CAM).

The main privacy issue is that (1) the server may learn private information based on the meta information, (2) the server may crack the secret key. Exposing the pruning mask $mask$ to the server only reveals the location of unimportant data samples, while all samples are still encrypted by HE, hence the server still learns nothing about the data sample itself, which avoids the security issue (1). One of the most recent and well-known attacks is the IND-CPA+ [43]. However, it operates under the assumption that the server can ask the client to decrypt specific ciphertexts. The assumption is not upheld in Algorithm 2. Since the server can not specify the ciphertext to be decrypted and that the client only sends the server the resulting mask, rather than the decrypted scores.

The communication depends on the level of ciphertexts. As shown in Table 4, the lower level the ciphertext is at, the smaller the ciphertext is. Therefore, the server can set the ciphertext to $L = 0$ to reduce the communication overhead.

Table 4: Ciphertext sizes under different levels.

$N = 2^{16}$	$L = 0$	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$
Ciphertext Size(MB)	1.01	2.03	3.02	4	5.02	6

Algorithm 4: Ciphertext-wise Pruning

Input : The encrypted dataset $\bar{D} = \{\bar{X}_i, \bar{Y}_i\}_{i=0}^{C-1}$ and pruning masks $mask = \{M_i\}_{i=0}^{C-1}$, where $M_i \in \{0,1\}^B$ and B is the number of samples in each ciphertexts.

Output : The pruned encrypted dataset $\bar{D}' = \{\bar{X'}_i, \bar{Y'}_i\}_{i=0}^{C'-1}$, where $C' \leq C$.

$\bar{D}' \leftarrow \emptyset$;
for $i \leftarrow 0$ **to** $C-1$ **do**
  $sparsity_i \leftarrow B - \sum_{j=0}^{B-1} M_i[j]$; // compute the ciphertext-wise sparsity
$ct\_queue \leftarrow Sort(sparsity, \bar{D}, mask)$;
while $ct\_queue.is\_not\_empty()$ **do**
  $ct\_front \leftarrow ct\_queue.pop\_front()$; // The Trim() removes full and empty ciphertexts from the ct_queue
  if $ct\_front.sparsity == 0$ **then**
    $\bar{D}' \leftarrow \bar{D}' \cup \{ct\_front.ct\}$;
    continue;
  else
    $ct\_queue.push\_front(ct\_front)$;
    break;
while $ct\_queue.is\_not\_empty()$ **do**
  $ct\_back \leftarrow ct\_queue.pop\_back()$;
  if $ct\_back.sparsity == B$ **then**
    continue;
  else
    $ct\_queue.push\_back(ct\_back)$;
    break;
while $ct\_queue.is\_not\_empty()$ **do**
  $ct_{front} \leftarrow ct\_queue.pop\_front()$;
  $ct_{back} \leftarrow ct\_queue.pop\_back()$;
  $slot_{empty} \leftarrow 0$; // The FZero() returns index of the first 0 in ct_{front}
  for $slot \leftarrow 0$ **to** $B-1$ **do**
    if $ct\_front.mask[slot] == 0$ **then**
      $slot_{empty} \leftarrow slot$;
      $ct\_front.mask[slot] \leftarrow 1$;
      $ct\_front.sparsity \mathrel{-}= 1$;
  $slot_{used} \leftarrow 0$; // The FOne() returns index of the first 1 in ct_{back}
  for $slot \leftarrow 0$ **to** $B-1$ **do**
    if $ct\_back.mask[slot] == 1$ **then**
      $slot_{used} \leftarrow slot$;
      $ct\_back.mask[slot] \leftarrow 0$;
      $ct\_back.sparsity \mathrel{+}= 1$;
  $slot_{used} \leftarrow \mathsf{FOne}(ct_{back})$;
  $k \leftarrow slot_{used} - slot_{empty}$;
  $M_t \leftarrow \{0\}^B$; // The Mask() keep only the sample at $slot_{used}$ in ct_{back}
  $M_t[slot_{used}] \leftarrow 1$;
  $ct_{align} \leftarrow ct_{back} \boxtimes M_t$;
  $ct_{front} \leftarrow ct_{front} \boxplus \mathsf{HE.Rot}(ct_{align}, k)$;
  if $ct\_queue.is\_empty() \vee ct_{front}.sparsity == 0$ **then**
    $\bar{D}' \leftarrow \bar{D}' \cup \{ct_{front}.ct\}$;
  else
    $ct\_queue.push\_front(ct_{front})$;
  $ct\_queue.push\_back(ct_{back})$;
return $\bar{D}'$

C Datasets and Hyperparameters

We detail how we divide the training set, validation set and test set in Table 5. For all tasks, we use a batch size of 128. For the transfer learning setting, we use a learning rate of 0.5 and set the pruning frequency as every $\Delta_\tau = 5$ epochs. For the training from scratch setting, we use a leaning rate of

https://doi.org/10.52202/079017-1616

0.1 and set the pruning frequency as every $\Delta_\tau = 10$ epochs. Prior works [9, 12] adopt a warm start strategy, which trains the model for $10 \sim 20$ epochs on the full dataset. The warm start period is computation intensive as it uses the whole dataset. Therefore, we use a random start strategy and randomly choose a fraction of data samples according to the pruning ratio p.

Table 5: Dataset Distribution for Various Machine Learning Challenges

Dataset	Total	Train	Validation	Test
MNIST	70000	52500	7500	10000
CIFAR-10	60000	43750	6250	10000
Face Mask Detection	4072	2849	408	815
DermaMNIST	10015	7007	1003	2005
SNIPS	14484	13084	700	700

D Ciphertext-wise pruning

In Algorithm 4, we detail the $\mathsf{Trim}()$, $\mathsf{FZero}()$, $\mathsf{Fone}()$ and $\mathsf{Mask}()$ functions, which are used during ciphertext-wise pruning. When $k > 0$, the server perform left rotation on ct_{back} by $|k|$ sample slot. If $K < 0$, right rotation is performed.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We accurately describe the Abstract and Introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section 5.

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Section 3.1

https://doi.org/10.52202/079017-1616

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer:[Yes]

Justification: Code is released with anonymity.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In Section 4, we report the mean and variance of the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

https://doi.org/10.52202/079017-1616

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4 System Setup.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We report Broader Impact in Section 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite datasets and models in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

https://doi.org/10.52202/079017-1616

Answer: [Yes]

Justification: We provide an anonymous URL to release our code in the abstract.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This work does not involve crowdsourcing or human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification:

Guidelines:This work does not involve crowdsourcing or human subjects.

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.