PRACTICAL AND PRIVATE HETEROGENEOUS FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Heterogeneous federated learning (HFL) enables clients with different computation/communication capabilities to collaboratively train their own customized models, in which the knowledge of models is shared via clients' predictions on a public dataset. However, there are two major limitations: 1) The assumption of public datasets may be unrealistic for data-critical scenarios such as Healthcare and Finance. 2) HFL is vulnerable to various privacy violations since the samples and predictions are completely exposed to adversaries. In this work, we develop PrivHFL, a general and practical framework for privacy-preserving HFL. We bypass the limitations of public datasets by designing a simple yet effective dataset expansion method. The main insight is that expanded data could provide good coverage of natural distributions, which is conducive to the sharing of model knowledge. To further tackle the privacy issue, we exploit the lightweight additive secret sharing technique to construct a series of tailored cryptographic protocols for key building blocks such as secure prediction. Our protocols implement ciphertext operations through simple vectorized computations, which are friendly with GPUs and can be processed by highly-optimized CUDA kernels. Extensive evaluations demonstrate that PrivHFL outperforms prior art up to two orders of magnitude in efficiency and realizes significant accuracy gains on top of the stand-alone method.

1 INTRODUCTION

Heterogeneous federated learning (HFL) Li & Wang (2019) Chang et al. (2019), as a promising variant of federated learning (FL), enables clients equipped with different computation and communication capabilities to collaboratively train their own customized models that may differ in size, numerical precision or structure Lin et al. (2020). In particular, clients share the knowledge of models via their predictions on an unlabeled public dataset collected from the Internet or generated by pre-trained generators, rather than model parameters in FL. This flexible approach facilitates customized services in areas like Healthcare and Finance Kairouz et al. (2019), while solving the intellectual property concerns of the FL models Atli et al. (2020). However, HFL suffers from two major limitations. (1) The assumption of public datasets may be unrealistic for many data-critical scenarios Zhu et al. (2021). For example, in Healthcare applications, task-related public datasets that contain patients' sensitive information are usually difficult to obtain because of current strict regulations, such as General Data Protection Regulation. (2) Sharing predictions may still leak the privacy of local data Papernot et al. (2017). Several works currently have demonstrated that given the black-box access to the trained model, adversaries can infer membership information Shokri et al. (2017) Salem et al. (2019) and attribute information Ganju et al. (2018) of the target sample, and even can reconstruct the original training data Yang et al. (2019). Therefore, to promote the deployment of HFL in real-world applications, it is crucial to solve the above two problems.

To the best of our knowledge, in HFL the relaxation of the public dataset assumption has not been explored before. Because it is challenging to achieve collaborative training under heterogeneous models when there is no a public dataset as a medium for the model knowledge transfer Li & Wang (2019). On the other hand, to mitigate the above privacy risks, a natural solution is to integrate advanced secure prediction protocols, such as CrypTFlow2 Rathee et al. (2020), CryptGPU Tan et al. (2021), and HE-transformer Boemer et al. (2019b), which can protect the private information during the model knowledge transfer by utilizing homomorphic encryption (HE) Gentry (2009), garbled

circuit (GC) Yao (1986) or oblivious transfer (OT) Naor & Pinkas (2001) techniques (Details can be referred to Table 5 and Appendix D). Unfortunately, such methods add a huge computation and communication overhead due to the use of heavy cryptographic primitives. For instance, Choquette-Choo et al. (2021) recently proposed CaPC, the first private collaborative learning scheme based on HE-transformer Boemer et al. (2019b) supporting heterogeneous models, which can be directly extended to HFL. As mentioned above, this work still suffers from efficiency issues, inherited from prior secure prediction protocols. Moreover, their scheme is implemented by the interaction between clients, however in real-world applications, clients (e.g., mobile devices) generally cannot establish direct communications channels with others Bonawitz et al. (2017). Therefore, the challenge here is how to efficiently implement secure prediction protocols in the realistic HFL setting.

In this work, to approach the above challenges of practicality and efficiency, we develop PrivHFL, a general and practical framework for privacy-preserving HFL. First, PrivHFL relaxes the assumption of dependence on public datasets, and realizes the entire training process only with clients' private datasets. To this end, we propose a simple but effective dataset expansion method based on Mixup Zhang et al. (2018), which is originally a regularization technique to improve generalization. The key idea is that the expanded data could provide good coverage of natural dataset distributions and hence could be used as an effective medium for transferring model knowledge. Second, to securely and efficiently instantiate the above HFL protocol, we leverage lightweight cryptography called additive secret sharing and construct customized protocols *from scratch* for secure prediction. Our gains mainly come from the improvement in communication and computation through the elimination of costly HE, GC and OT protocols. Moreover, in contrast to the prior works that evaluate cryptographic protocols in CPUs, PrivHFL converts complex cryptographic operations to simple computations on large blocks of data, which are friendly with GPUs and can be processed by highly-optimized CUDA kernels Tan et al. (2021). We evaluate the designed protocol on the GPU and CPU, and the results show that our GPU-based protocol is up to $10\times$ faster than its CPU analog. Our key contributions can be summarized as follows:

- We introduce a practical HFL framework, which is independent on public datasets while provably providing comprehensive privacy protection.
- We design a simple yet effective dataset expansion method to promote the sharing of model knowledge, and construct customized cryptographic protocols for secure prediction.
- Extensive experiments on SVHN, CIFAR10, Tiny ImageNet (including IID and Non-IID settings) and various heterogeneous models demonstrate that PrivHFL outperforms prior art up to two orders of magnitude in efficiency and realizes roughly 10% accuracy gains.

2 BACKGROUND

Before introducing PrivHFL, we first describe the heterogeneous federated learning and the threat model, and then review the cryptographic primitives that are required to understand our work.

2.1 HETEROGENEOUS FEDERATED LEARNING

In heterogeneous federated learning (HFL), the clients independently design their own unique models, but due to the model heterogeneity, they cannot directly share model parameters with each other. Instead, they learn the knowledge of other models via the predictions on a task-related public dataset, where a HFL service provider, i.e., the server, coordinates the whole process and routes messages between the clients since they generally cannot establish direct communication channels with others. Specifically, clients first train local models with individual private dataset. Then, each client performs prediction on an unlabeled public dataset based on the local model and sends the prediction result to the server to aggregate. Later, the server broadcasts aggregated results to clients, which will retrain local models based on the public dataset and received predictions. The whole process is iterative until the local models meet the pre-defined accuracy requirements. Details can be referred to Figure 9 and Appendix A.

2.2 THREAT MODEL

We work in an honest-but-curious adversary setting Goldreich (2009), where each entity (including the clients and the server) strictly follows the specification of the designed protocol but attempts to infer more knowledge about other clients' private information such as model parameters and private datasets. Moreover, to maintain the reputation and provide more services, the server does not collude with any clients. This setting is reasonable and has been widely instantiated in previous works Phong et al. (2018) Sun & Lyu (2021) Choquette-Choo et al. (2021).

2.3 CRYPTOGRAPHIC PRIMITIVES

Additive secret sharing. We adopt lightweight 2-out-of-2 additive secret sharing over the ring $\mathbb{Z}_L$ Demmler et al. (2015) as the cryptographic building block. We let $\mathsf{Share}(x)$ denote the sharing algorithm that takes as input x in $\mathbb{Z}_L$ and outputs random sampled shares $[x]_0$, $[x]_1$ with the constraint that $x = [x]_0 + [x]_1$ in $\mathbb{Z}_L$. The reconstruction algorithm $\mathsf{Recon}([x]_0, [x]_1)$ takes as input the two shares and outputs $x = [x]_0 + [x]_1$. The security of the additive secret sharing protocol guarantees that given a share$[x]_0$ or $[x]_1$, the value x is perfectly hidden.

Pseudorandom generator. A Pseudorandom Generator (PRG) takes as input a uniformly random seed and generates a long pseudorandom string. The security of PRG ensures that the output is indistinguishable from the uniform distribution in polynomial-time. In PrivHFL, we instantiate PRG with the technique from Matyas (1985) and the seed can be generated by the Diffie-Hellman Key Agreement protocol Diffie & Hellman (1976). Details can be referred to Appendix C.3.

3 THE PRIVHFL PROTOCOL

In this section, we introduce the high-level view of PrivHFL, followed by describing in detail our mixup-based dataset expansion and secure prediction schemes.

3.1 HIGH-LEVEL VIEW OF PRIVHFL

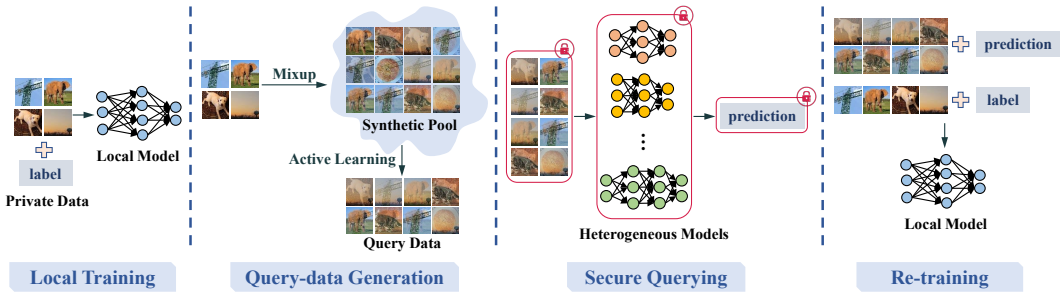


Figure 1: High-level view of PrivHFL

PrivHFL follows the prior HFL works Li & Wang (2019) and iteratively optimizes the clients' models. Each client in PrivHFL can play the role of querying party and responding party at the same time, and without loss of generality, we denote them as P_Q and P_A. As shown in Figure 1, in each communication round, each querying party P_Q performs four-phase operations with other responding parties P_A, i.e., *local training*, *query-data generation*, *secure querying*, and *re-training*. In detail, clients first train the local model on their own private datasets, which is the baseline any future improvements will be compared with. After that, by utilizing our dataset expansion method, each querying party P_Q can generate query data to query other responding parties P_A (C fraction of all clients) for prediction results. To protect the privacy of query samples, predictions and model parameters, clients adopt our secure prediction protocol and conduct collaborative querying in the ciphertext form. To the end, each client can retrain the local model based on both the private data and query results received. Algorithm 1 in Appendix C.1 gives the detailed description of the PrivHFL protocol.

3.2 QUERY-DATA GENERATION

Prior HFL works rely on public datasets for model knowledge transfer, but data collection could be difficult in real-world applications due to various reasons, such as privacy concerns and rare classes. To relax such assumption, we design a dataset expansion method, inspired by the success of Mixup in improving model generalization Yun et al. (2019) Zhang et al. (2018) and the efficiency of knowledge distillation Wang et al. (2020). Specifically, we repurpose mixup to construct a big synthesized pool on a small private dataset that could provide a good coverage of the manifold of natural samples. Given two private samples x_i and x_j, we generate multiple synthetic query samples by a convex combination with different coefficients λ, as follows:

$$\tilde{x}_{i,j}(\lambda) = \lambda \cdot x_i + (1 - \lambda) \cdot x_j \tag{1}$$

where $\lambda \in [0, 1]$. While this method is simple, it can exponentially expand the size of initial private dataset and hence provide more candidate samples for query. Note that an alternative solution is to directly use private datasets as the query data. As we will show in the experimental section, this method is effective but inferior than the query strategy based on mixup. We also explore to understand the effectiveness of mixup-based datset expansion from the perspective of data distribution in Figure 10 of Appendix B.1.

To further improve query efficiency especially in the cryptographic setting, we use active learning strategies Tong & Koller (2001), Wang et al. (2020) to select informative samples from the generated synthesized pool. Several active learning strategies based on margin, greedy k-center, and entropy have been introduced in Appendix C.2.

3.3 SECURE QUERYING PROTOCOL

Recall that our querying protocol includes three entities, i.e., the server, P_Q and P_A. In order to fully protect the privacy of query samples, predictions and model parameters, we exploit additive secret sharing to construct cryptographic building blocks for the querying protocol. The challenge here is how to efficiently implement such protocol under strict communication constraints, i.e., the communication channels cannot be established among clients. To tackle this challenge, the key idea is to outsource the secure querying task to the server and P_A, and then design a P_Q-assisted customized protocol to accelerate the evaluation. Moreover, an important design principle is to avoid the use of heavy cryptography such as HE and GC, but to conduct GPU-friendly protocol that mainly includes vectorized operations. In the following, we decompose our scheme into three steps: *query-data sharing*, *secure prediction*, and *result aggregation*.

Query-data sharing. We first construct PRG seeds in pairs for P_Q, P_A and the server, denoted as Sk_{QA}, Sk_{SA}, and Sk_{SQ}, which are used to generate random numbers to reduce communication overhead. For more details, please refer to Appendix C.3. Figure 2 shows our query-data sharing protocol, in which P_Q secret-shares the query data x to the server and P_A for secure prediction. In particular, P_Q non-interactively shares $[x]_0 = r$ with P_A using PRGs on the same seed Sk_{QA}. After that, P_Q computes and sends $[x]_1 = x - r$ to the server.

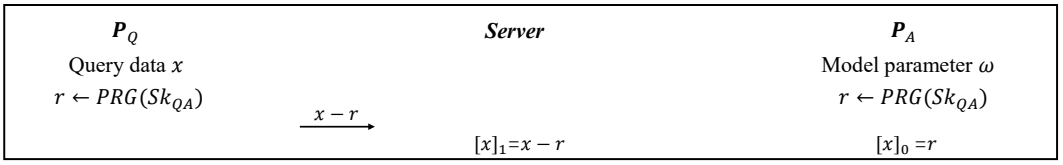


Figure 2: Secure query-data sharing protocol

Secure prediction. Deep learning models consist of a sequence of linear layers (i.e., fully connected layers, convolutional layers and AvgPooling) and non-linear layers (i.e., ReLU, MaxPooling). Recently, many works achieve secure prediction protocols in client-server setting (i.e., 2-party setting), e.g., the state-of-the-art method CrypTFlow2 Rathee et al. (2020) or HE-Transformer Boemer et al. (2019a) that is adopted by CaPC Choquette-Choo et al. (2021). However, such methods add a huge overhead due to the use of heavy cryptographic primitives. In Section 4.3 and Appendix B.2, we compare against such techniques. In addition, the most efficient secure prediction protocol currently

is CryptGPU Tan et al. (2021) under the 3-party setting, but it cannot be directly extended to HFL due to the inability to communicate between clients. To improve the efficiency of secure prediction, we design customized protocols for the linear layer and non-linear layer *from scratch*. Notably, even compared to CryptGPU, PrivHFL also obtains better performance gains (see Section 4.3).

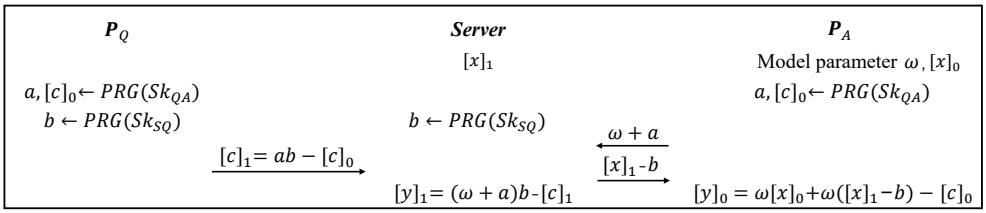


Figure 3: Secure matrix multiplication protocol

(i) Linear layers. P_A and the server compute the matrix multiplication wx, where the parameters w is held by P_A and the input x is secret-shared between P_A and the server. We observe that $wx = w[x]_0 + w[x]_1$, and P_A can compute $w[x]_0$ locally. As shown in Figure 3, we design an efficient matrix multiplication protocol for evaluating $w[x]_1$ with the help of P_Q. In particular, P_Q generates three random matrices a, b and $[c]_0$ using PRGs, and computes $[c]_1$ that satisfies $[c]_1 + [c]_0 = a \cdot b$. Accordingly, the server can generate the same b using PRGs based on the seed Sk_{SQ}, and P_A obtains a and $[c]_0$ using PRGs based on the seed Sk_{QA}. Later, the server and P_A jointly compute $[y]_1$ and $[y]_0$, which are the secret shares of wx. Note that the evaluation of fully-connected, convolutional and AvgPooling layers can be derived directly from the matrix multiplication protocol.

(ii) ReLU. The formula of ReLU is $\mathsf{ReLU}(x) = x \cdot \mathsf{DReLU}(x)$, where $\mathsf{DReLU}(x)$ equals 1 if $x \geq 0$ and 0 otherwise. The multiplication of x and $\mathsf{DReLU}(x)$ can be implemented by using the algorithm in Figure 3. Given that $\mathsf{DReLU}(x) = 1 - \mathsf{sign}(x)$, the main idea is to compute $\mathsf{sign}(r \cdot x)$ given a random value r, where $\mathsf{sign}(x) = \mathsf{sign}(r \cdot x)$ if r is a positive value. As shown in Figure 4, we design a secure DReLU protocol based on the above observation. In detail, P_A and the server first generate a random positive value r using PRGs with the seed Sk_{SA}, and compute $[z] = r \cdot [x]$ locally. After that, P_A and the server disclose the secret-shared value $[z]$ to P_Q, which then computes the sign of z, i.e., the sign of x. $\mathsf{sign}(z)$ is shared to the server and P_A by using the protocol in Figure 2.

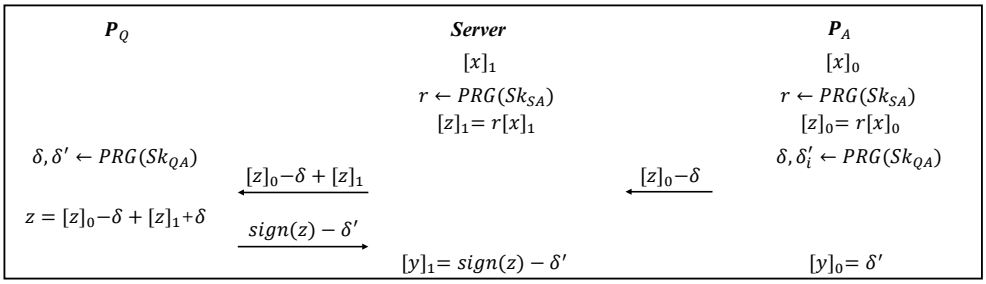


Figure 4: Secure DReLU protocol

(iii) Maxpooling. Maxpooling on n values is computed using a tree-reduction algorithm, which recursively partitions the input into two halves and then compares the elements of each half. Specifically, clients arrange the n values into a 2-ary tree with depth $\log_2^n$, and evaluate the tree in a top-down fashion. In each comparison of two secret-shared element $[x]$ and $[y]$, we reduce it to the evaluation of ReLU. We observe $\mathsf{max}([x], [y]) = \mathsf{ReLU}([x] - [y]) + [y]$, and hence the computational complexity of Maxpooling evaluation mainly comes from the evaluation of $n - 1$ ReLU.

Result Aggregation. After executing the secure prediction, the server and P_A hold the shares of predicted logits $[x_j], j \in 1, 2, \cdots, n$, which will be aggregated and then returned to P_Q. As shown in Figure 5, P_A and P_Q first generate a random value r_j based on PRGs. Then each responding party computes $[x_j]_0 - r_j$ and sends it to the server. The server aggregates all received values and sends

$\sum_{j=1}^{n}\left([x_j]_0 - r_j + [x_j]_1\right)$ to P_Q. Thus, P_Q can reconstruct the aggregated logit $y = \sum_{j=1}^{n} x_j$ and obtains the aggregated soft label of the query data.

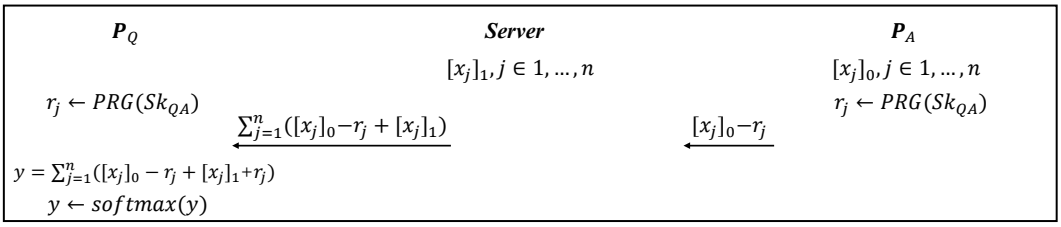


Figure 5: Secure result aggregation protocol

GPU Acceleration. While GPUs have played a critical role in accelerating modern deep learning, cryptographic protocols like GC and HE are less well-suited for taking advantage of GPU parallelism due to extensive conditional statements and modular reductions. On the contrary, our protocols mainly consisting of vectorized secret-sharing are GPU-friendly, and all operations can be processed by highly-optimized CUDA kernels. However, existing general-purpose platforms on GPUs such as NVIDIA's CUDA are designed to operate on floating-point inputs. In PrivHFL, we typically compute over integer values (i.e., 64-bit ring elements). To leverage optimized kernels for protocol acceleration, we integrate CryptGPU Tan et al. (2021) that embeds the integer-valued cryptographic operations into floating-point arithmetic, to design and evaluate PrivHFL.

Security analysis of PrivHFL. We give a formal security proof in Appendix E. Intuitively, PrivHFL reveals zero information to P_A and the server, and only reveals the final aggregated prediction to P_Q, since all intermediate values are secret-shared. Given the above, a corrupted P_A cannot learn anything about the query data of querying parties, while the confidentiality of responding parties' model parameters against corrupted P_Q is also protected.

3.4 Discussions

Discussions on differential privacy extension. Several works Papernot et al. (2017) Sun & Lyu (2021) Choquette-Choo et al. (2021) have developed differential privacy mechanisms to prevent potential leakage from the aggregated values in centralized deep learning, FL and HFL. PrivHFL is compatible with differential privacy techniques such as PATE Papernot et al. (2017). Similar to CaPC Choquette-Choo et al. (2021), we can also entrust the server to add Laplacian or Gaussian noise to the returned aggregation results. A key challenge is that clients' corruption and the increased number of queries may scale the sensitivity parameter and hence increase the amount of noise. We leave further explorations of this aspect for future works.

Discussions on scalability. While PrivHFL's secure prediction protocol has already improved performance over current techniques (e.g., CrypTFlow2 Rathee et al. (2020)) by two orders of magnitude, a main limitation is the scalability of the system, exactly, the number of clients and the scale of trained models that can be supported. Currently, the system can only carry a small number of clients and medium-sized models, since the overall running time mainly depends on these two factors. Hence, we view PrivHFL as a first step in constructing privacy-preserving protocols for HFL, and leave the improvements on scalability for future works.

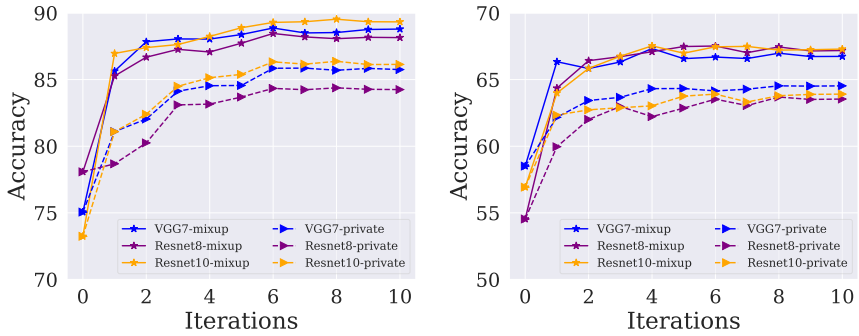
4 Evaluation

4.1 Evaluation setup

Datasets and models. We evaluate PrivHFL on three standard datasets for image classification, SVHN, CIFAR10 and Tiny ImageNet. By default, we assume independent and identically distributed (IID) training data among clients. We also use the Dirichlet distribution $\mathsf{Dir}(\alpha)$ as in Lin et al. (2020) to create disjoint Non-IID training data. The value of α controls the degree of Non-IID-ness, namely that a smaller α indicates higher degree of Non-IID-ness. Figure 12 in Appendix B visualizes how Non-IID samples are distributed. For SVHN and CIFAR10, we set the number of

clients $N = 50$ and use VGG-7, ResNet-8 and ResNet-10 architectures as the clients' heterogeneous local models. For Tiny ImageNet, we use ResNet-14, ResNet-16, ResNet-18 architectures and set $N = 10$. Please refer to Appendix F for more experiment setup.

Cryptographic protocol. We build PrivHFL on top of the CryptGPU library Tan et al. (2021), but reimplement the underlying cryptographic protocols proposed in Section 3. We set the security parameter κ as 128. To match the accuracy in the plaintext form, we set secret-sharing protocols over the 64-bit ring $\mathbb{Z}_{2^{64}}$, and encode inputs using a fixed-point representation with 20-bit precision.



(a) SVHN (b) CIFAR10

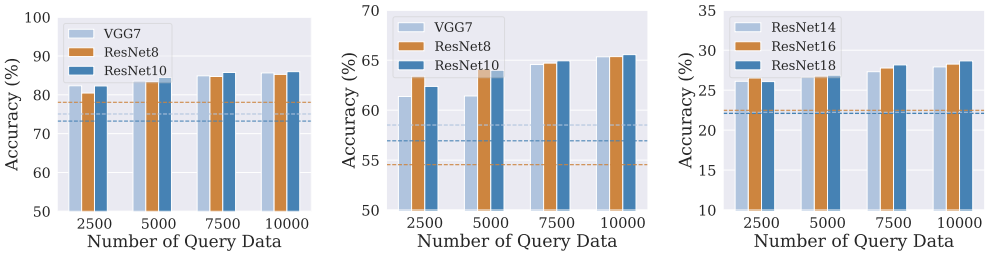
Figure 6: **The performance of PrivHFL on different query datasets: the private dataset and the expanded datset based on mixup.** VGG7, ResNet8 and ResNet10 are used as heterogeneous models, and we report the average accuracy of each model architecture.

4.2 Evaluation on the Heterogeneous Federated Learning

Table 1: **Test accuracy of PrivHFL on different client selection fractions and different numbers of query data.** We also report the accuracy of models trained on the local dataset (i.e., baseline), and the accuracy of models retrained on private data-based queries.

		SVHN			CIFAR10			Tiny ImageNet		
		$C = 0.6$	$C = 0.8$	$C = 1$	$C = 0.6$	$C = 0.8$	$C = 1$	$C = 0.6$	$C = 0.8$	$C = 1$
Baseline			75.46			56.66			22.26	
Private data		79.43	79.56	80.29	60.82	61.01	61.49	24.89	25.11	25.23
Mixup data	2.5K	80.09	80.32	81.69	62.87	63.05	63.23	25.82	26.03	26.23
	5.0K	83.32	83.52	83.82	63.04	63.44	63.69	26.22	26.46	26.75
	7.5K	84.54	84.78	85.12	62.97	63.64	63.88	27.14	27.54	27.75
	10K	84.58	84.97	85.62	63.79	63.82	64.56	27.67	28.19	28.46

Table 1 shows the detailed results on the performance improvement (i.e., accuracy gains on the test datasets) brought by PrivHFL. We randomly select synthetic data for query instead of using active learning strategies. We observe that for SVHN and CIFAR10, the accuracy gain of PrivHFL is about 10% when we use $10K$ mixup samples. However, using private dataset-based query strategy only increases the accuracy by about 4%. For Tiny ImageNet, the accuracy can reach 28.46% when using $10K$ mixup data but only 25.23% on the private dataset-based queries. Table 1 also shows different accuracy gains as the number of mixup query data increases. We observe that compared with $2.5K$ mixup data, we obtain $4.65\%, 1.33\%, 3.23\%$ increased accuracy using $10K$ mixup data on SVHN, CIFAR10 and Tiny ImageNet, respectively. This is because more synthetic data could provide a better coverage of natural dataset distributions. Besides, with the increase of the selected fraction C, the accuracy improves slightly, e.g., 1.6% gains for SVHN from $C = 0.6$ to 1. Figure 10(c) in Appendix B.1 also illustrates the effectiveness of the mixup-based query strategy compared to other data expansion methods.



(a) SVHN (b) CIFAR10 (c) Tiny ImageNet

Figure 7: **Test accuracy of heterogeneous models after PrivHFL as the number of query-data increases.** Dashed lines represent the baseline, i.e., the test accuracy before executing PrivHFL.

Figure 6 and Figure 7 show the improvement of different heterogeneous models with different iterations and different numbers of query data. We observe that as the number of query data increases, PrivHFL consistently outperforms the baseline for all the datasets and heterogeneous models. Figure 11 and Figure 14 in Appendix B.3 illustrate the impact of active learning strategies and the number of private data on the test accuracy, respectively. Besides, we also show the test accuracy of PrivHFL on CIFAR10 and SVHN for different degrees of Non-IID-ness in Figure 13 in Appendix B.3.

4.3 Evaluation on the Cryptographic Protocols

Table 2: **Runtime (sec) of the three steps in PrivHFL's secure querying protocol.** The runtime of secure prediction represents the query time of each client on different models. CIFAR10 and SVHN have the same time due to the same input size and model architecture.

Dataset	# Queries	1. Query-data sharing	2. Secure prediction			3. Result aggregation
			VGG7	ResNet8	ResNet10	
CIFAR10 (SVHN)	1000	5.08	55.97	87.71	106.46	0.09
	2500	7.16	133.52	206.20	256.39	0.12
	5000	11.32	271.89	428.54	512.88	0.30
			ResNet14	ResNet16	ResNet18	
Tiny ImageNet	1000	9.87	737.43	962.51	1075.13	0.18
	2500	18.78	1778.69	2356.33	2537.49	0.32

In this section, we mainly focus on the extra overhead caused by the secure querying phase. To clearly illustrate the efficiency improvement of PrivHFL, unless otherwise specified, we only show the overhead of one communication round as shown in Section 3. In Table 2, we report the running time for three datasets with different model architectures. Recall that our secure querying protocol consists of three phases, i.e., query-data sharing, secure prediction, and result aggregation. As shown in Table 2, the main cost of our framework comes from the second phase, where the evaluation of batched secure prediction is required. Specifically, it takes 4.5 minutes to evaluate 5000 query samples securely on VGG7 and CIFAR10. Besides, only 11.32 seconds and 0.3 seconds are spent on the query-data sharing and result aggregation phases. More running time is required to evaluate Tiny ImageNet because of increased input size and model architecture.

To demonstrate the effectiveness of PrivHFL's secure prediction protocol, we compare with CrypTFlow2 Rathee et al. (2020) and CryptGPU Tan et al. (2021). In Table 4 of Appendix B.2, we also compare with CaPC Choquette-Choo et al. (2021) that uses HE-Transformer as the building block. Table 3 summarizes this improvement for three models used in CIFAR10. We observe that, PrivHFL achieves significant improvement on all the models in terms of runtime and communication costs. To be specific, PrivHFL requires 105.2-112.0 $\times$ less running time and 233.6-269.3 $\times$ less communication compared with CrypTFlow2-OT. Similarly, compared with CrypTFlow2-HE, we show a greater advantage (i.e., 139.1-183.8 $\times$) in computational overhead, and a relatively small advantage

Table 3: **Runtime (sec) and communication cost (MB) of secure prediction for different methods on CIFAR10.** CrypTFlow2 Rathee et al. (2020) is the SOTA 2-party protocol that contains two variants (OT-based and HE-based) and CryptGPU Tan et al. (2021) is the SOTA 3-party GPU-friendly protocol. We report the cost of a single prediction on three models.

Method	**VGG7**		**ResNet8**		**ResNet10**	
	Time	Comm.	Time	Comm.	Time	Comm.
CrypTFlow2-OT Rathee et al. (2020)	39.22	15562.9	39.35	21261.6	55.76	28517.6
CrypTFlow2-HE Rathee et al. (2020)	48.70	651.51	56.21	1110.39	97.46	1395.18
CryptGPU Tan et al. (2021)	1.61	144.51	2.02	131.39	2.79	221.57
This work (CPU)	1.03	66.61	1.63	73.85	2.00	105.88
This work (GPU)	**0.35**	**66.61**	**0.36**	**73.85**	**0.53**	**105.88**

(i.e., 9.7-15.0 $\times$) in communication overhead. In addition, we also test the performance under the CPU architecture, and the results show that the above three models can still be evaluated within 2 seconds. Notably, even comparing with CryptGPU, our weaker CPU setting obtains an improvement of roughly 1.5 $\times$ and 2 $\times$ in terms of computation and communication overheads, respectively.

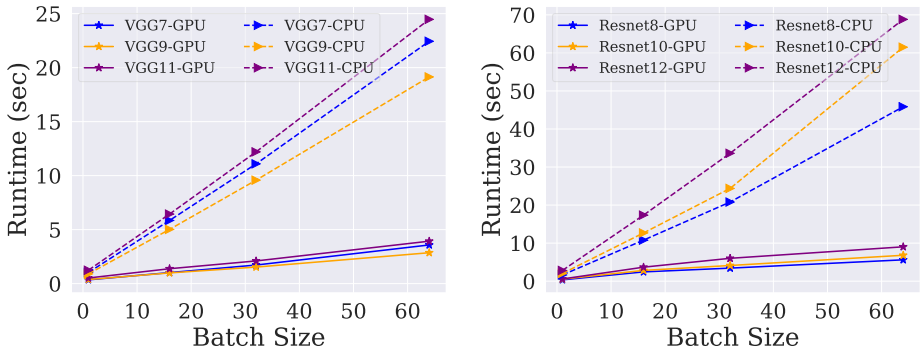


Figure 8: **Runtime (sec) of batch secure prediction on CPU and GPU as the batch-size increases.** We report the result of VGG-style and ResNet-style networks on CIFAR10.

To further understand the impact of GPU-enable acceleration on the total cost of our scheme, we evaluate our secure prediction protocol under both the CPU and GPU setting with different batch size. The numerical results are reported in Figure 8. We observe that on all the models used in evaluation, GPU-based secure prediction is superior to the CPU analogs. As the batch size increases, the advantages of GPU-based protocols will become more pronounced.

5 Conclusion

In this paper, we propose a practical heterogeneous federated learning framework, which is independent on public datasets while provably guaranteeing the privacy of samples, model parameters and predictions. Extensive numerical experiments on SVHN, CIFAR10 and Tiny ImageNet and various heterogeneous models demonstrate that PrivHFL outperforms prior art two orders of magnitude in efficiency and realizes about 10% accuracy gains. In the future, we will further improve the scalability of the system and integrate with advanced differential privacy mechanisms.

References

Buse Gul Atli, Yuxi Xia, Samuel Marchal, and N Asokan. Waffle: Watermarking in federated learning. *arXiv preprint arXiv:2008.07298*, 2020.

James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of ACM CCS*, pp. 1253–1269, 2020.

Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. ngraph-he2: A high-throughput framework for neural network inference on encrypted data. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pp. 45–56, 2019a.

Fabian Boemer, Yixing Lao, Rosario Cammarota, and Casimir Wierzynski. ngraph-he: a graph compiler for deep learning on homomorphically encrypted data. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, pp. 3–13, 2019b.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of ACM CCS*, pp. 1175–1191, 2017.

Alon Brutzkus, Ran Gilad-Bachrach, and Oren Elisha. Low latency privacy preserving inference. In *Proceedings of ICML*, pp. 812–821, 2019.

Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*, 2019.

Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In *Proceedings of ACM CCS*, pp. 395–412, 2019.

Christopher A Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. Capc learning: Confidential and private collaborative learning. *Proceedings of ICLR*, 2021.

Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *Proceedings of NDSS*, 2015.

Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *Proceedings of ICLR*, 2021.

Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

Canh T Dinh, Nguyen H Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. *Proceedings of NeurIPS*, 2020.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *Proceedings of NeurIPS*, 2020.

Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of ACM CCS*, pp. 619–633, 2018.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of ACM STOC*, pp. 169–178, 2009.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of ICML*, pp. 201–210, 2016.

Oded Goldreich. *Foundations of cryptography*. Cambridge university press, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of CVPR*, pp. 770–778, 2016.

Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *Proceedings of USENIX Security*, pp. 1895–1912, 2019.

Bargav Jayaraman and Lingxiao Wang. Distributed learning without distress: Privacy-preserving empirical risk minimization. 2018.

Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. Gazelle: A low latency framework for secure neural network inference. In *Proceedings of USENIX Security*, pp. 1651–1669, 2018.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. *arXiv preprint arXiv:2109.00984*, 2021.

Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. In *Proceedings of NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Proceedings of NeurIPS*, 2020.

Stephen M Matyas. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, 27:5658–5659, 1985.

Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. Delphi: A cryptographic inference service for neural networks. In *Proceedings of USENIX Security*, pp. 2505–2522, 2020.

Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *Proceedings of IEEE S&P*, pp. 19–38, 2017.

Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of SODA*, pp. 448–457, 2001.

Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *Proceedings of ICLR*, 2017.

Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.

Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow2: Practical 2-party secure inference. In *Proceedings of ACM CCS*, pp. 325–342, 2020.

Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *Proceedings of NDSS*, 2019.

Sinem Sav, Apostolos Pyrgelis, Juan R Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. Poseidon: Privacy-preserving federated neural network learning. In *Proceedings of NDSS*, 2021.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pp. 309–318. Springer, 2001.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *Proceedings of ICLR*, 2018.

Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

Liyan Shen, Xiaojun Chen, Jinqiao Shi, Ye Dong, and Binxing Fang. An efficient 3-party framework for privacy-preserving neural network inference. In *Proceedings of ESORICS*, pp. 419–439, 2020.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of IEEE S&P*, pp. 3–18. IEEE, 2017.

Lichao Sun and Lingjuan Lyu. Federated model distillation with noise-free differential privacy. *Proceedings of IJCAI*, 2021.

Sijun Tan, Brian Knott, Yuan Tian, and David J Wu. Cryptgpu: Fast privacy-preserving machine learning on the gpu. *Proceedings of IEEE S&P*, 2021.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2(Nov):45–66, 2001.

Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: 3-party secure computation for neural network training. *Proc. Priv. Enhancing Technol.*, 2019(3):26–49, 2019.

Dongdong Wang, Yandong Li, Liqiang Wang, and Boqing Gong. Neural networks are more productive teachers than human raters: Active mixup for data-efficient knowledge distillation from a blackbox model. In *Proceedings of CVPR*, pp. 1498–1507, 2020.

Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. Neural network inversion in adversarial setting via background knowledge alignment. In *Proceedings of ACM CCS*, pp. 225–240, 2019.

Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of IEEE FOCS*, pp. 162–167, 1986.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of ICCV*, pp. 6023–6032, 2019.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *Proceedings of ICML*, pp. 7252–7261, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of ICLR*, 2018.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. *Proceedings of ICML*, 2021.

A HETEROGENEOUS FEDERATED LEARNING

In federated learning (FL), multiple clients collectively learn a single model with a fixed architecture, in which each client maintains a local model for its private data, while the server maintains a global model via aggregating the local model gradients from clients. Although successful, FL is often too restrictive in practice, since it assumes all clients sharing the same model architecture. To tackle this problem, heterogeneous federated learning (HFL) is proposed, which enables clients to collaboratively train their own customized models that may differ in size or structure. Due to the heterogeneity in local models among clients, it's not possible to collaborate via sharing gradients like FL. As shown in Figure 9, HFL uses a task-related public dataset to assist in collaborative learning. To be specific, clients first train local models with individual private dataset. Then each client performs inference on an unlabeled public dataset and obtains the local predictions, which are sent to the server to aggregate, rather than the local model gradients. Then, the server broadcasts the aggregated results to clients, and the clients will retrain local models based on the public dataset and received predictions.

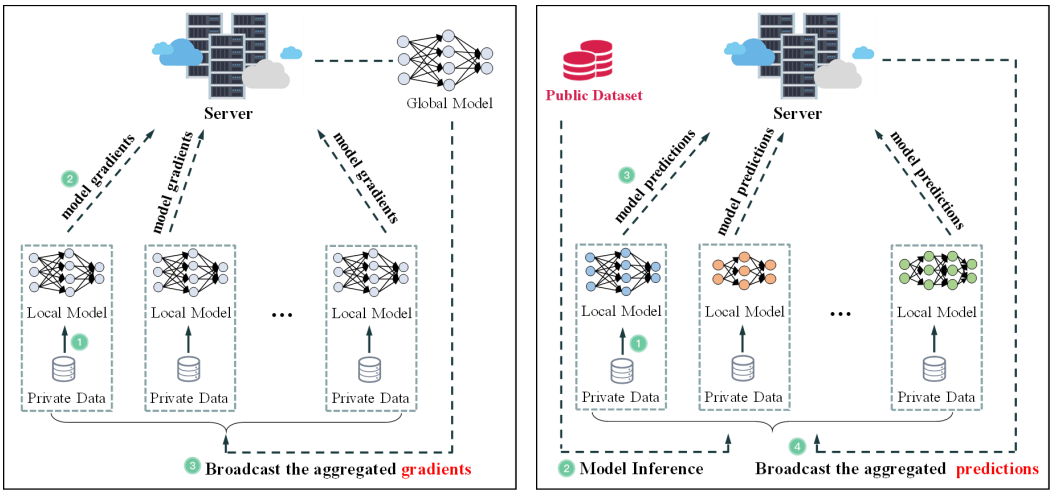
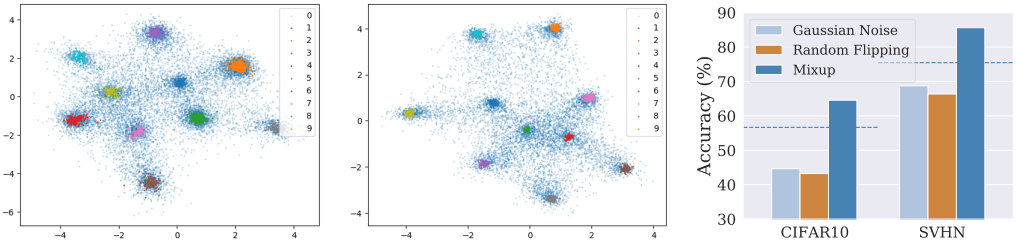


Figure 9: **Comparison with federated learning and heterogeneous federated learning**



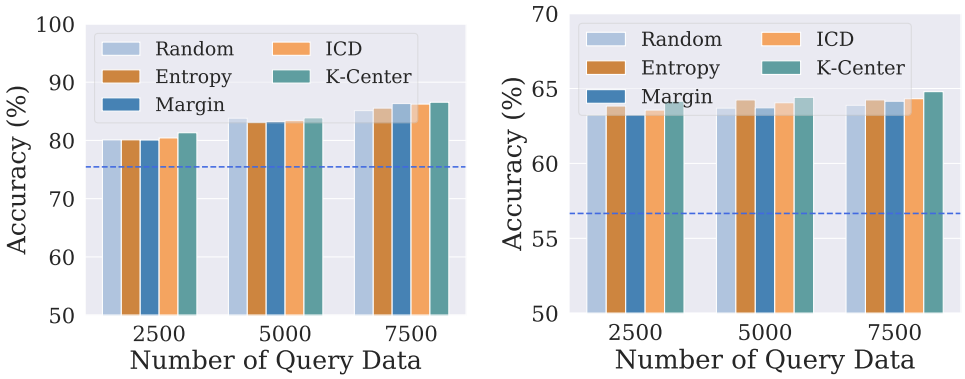
(a) Feature distribution on SVHN. (b) Feature distribution on CIFAR10. (c) Comparison with other dataset expansion methods.

Figure 10: **Understanding the importance of mixup-based datset expansion on improving model performance.** In (a) and (b), the bright colored points indicate the position of the original training data in the feature space, and the light blue points indicate the distribution of the expanded points based on mixup in the feature space. In (c), the left and right dashed lines are the baseline test accuracy on CIFAR10 and SVHN, respectively.

B Additional Experimental Results

B.1 Understanding mixup-based dataset expansion

Figure 10(a) and Figure 10(b) respectively show the feature distribution of the original images and the expanded images of SVHN and CIFAR10. This illustrates the main reason why mixup-based datset expansion works well in PrivHFL. As shown in Figure 10(a) and Figure 10(b), mixup-based expanded images are informative and could provide good coverage of the feature space. Therefore, learning the predictions from other clients on the expanded samples can further improve the accuracy of models. Beside, Figure 10(c) also compares mixup-based dataset expansion with other existing methods, including random uniformly noises sampled from Gaussian distribution and traditional data augmentation strategies, e.g., random flipping. We observe that Gaussian noise-based and random flipping-based dataset expansion methods dramatically reduce the model accuracy. This is because these methods do not necessarily model the problem domain distributions and do not provide informative natural images.



(a) SVHN (b) CIFAR10

Figure 11: **Using PrivHFL with active learning strategies to improve model accracy on SVHN and CIFAR10 datasets.** Dashed line represents the baseline accuracy of models, and the histogram represents the accuracy of the model after PrivHFL based on different active learning strategies.

B.2 Experimental results for secure prediction

In the model querying phase, we leverage GPU parallelism to process private querying on a batch of images, which effectively amortize the cost of private inference. In Table 4, we compare our GPU-friendly method on MNIST using the three model architectures (CryptoNets Gilad-Bachrach et al. (2016), CryptoNets-ReLU Gilad-Bachrach et al. (2016) and MLP Boemer et al. (2019b)) with HE-Transformer[1], which also achieves batch-axis packing for private inference. Note that CaPC Choquette-Choo et al. (2021) also adopt HE-Transformer as the building block. We can observe that PrivHFL is up to three orders of magnitude faster than HE-Transformer on the CryptoNets-ReLU and MLP models. The main reason is the efficiency of the protocol and the parallelism supported by the GPU. However, when evaluating CryptoNets that replaces ReLU activations with squared approximation, the improvement from PrivHFL will gradually decrease as the batch size increases, because the HE-based method does not require communication during the entire evaluation process. Note that approximations result in significant accuracy losses and degrades user experience, especially when evaluating modern large-scale models.

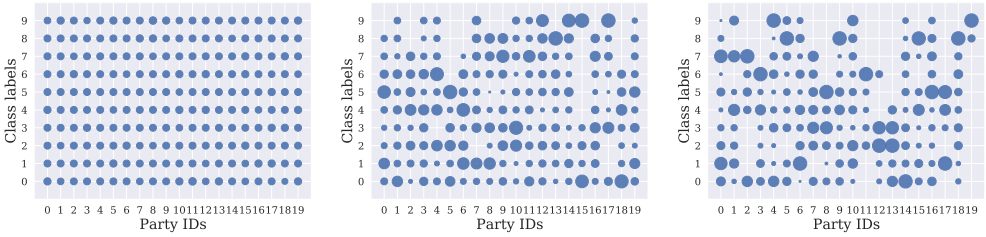
Table 4: **Runtime (sec) of secure prediction for HE-Transformer and our PrivHFL on MNIST as the batch-size increases.** He-transformer is the 2-party protocol used in CaPC (Choquette-Choo et al., 2021).

Model	BS=128		BS=256		BS=512		BS=1024		BS=2048	
	Ours	HE-T	Ours	HE-T	Ours	HE-T	Ours	HE-T	Ours	HE-T
CNet1	**0.07**	17.75	**0.14**	17.56	**0.25**	17.62	**0.49**	17.77	**0.99**	17.67
CNet2	**0.07**	48.83	**0.15**	70.14	**0.26**	112.42	**0.51**	201.42	**1.01**	369.51
MLP	**0.05**	65.01	**0.06**	86.37	**0.10**	129.81	**0.16**	216.61	**0.29**	391.13

B.3 Experimental results for heterogeneous federated learning

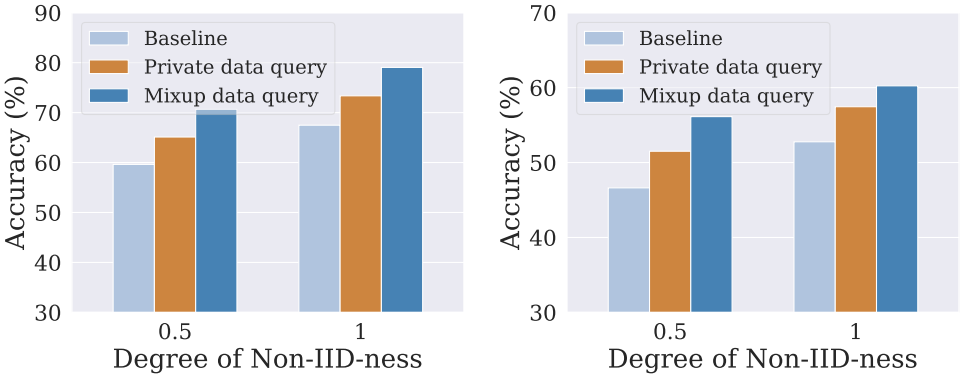
Figure 11 depicts the impact of different active learning stategies, in which even random sampling can drastically improve the model accuracy by 4.46%-9.66% and 6.57%-7.22% on SVHN and CIFAR10 datasets. Nonetheless, we can also achieve additional benefits leveraging active learning, such as 1.44% gains on SVHN with the entropy sampling and 0.8 % gains on CIFAR10 with the margin sampling.

[1] https://github.com/IntelAI/he-transformer



(a) $\alpha = 100$ (b) $\alpha = 1$ (c) $\alpha = 0.5$

Figure 12: **Visualization of Non-IID-ness among clients with different Dirichlet distribution α values on CIFAR10 dataset.** The size of scattered points indicates the number of training samples for a class available to that client.



(a) The model accuracy on SVHN. (b) The model accuracy on CIFAR10.

Figure 13: **The test accuracy of PrivHFL on SVHN and CIFAR10 for different degrees of Non-IID-ness.** The number of clients $N = 20$.

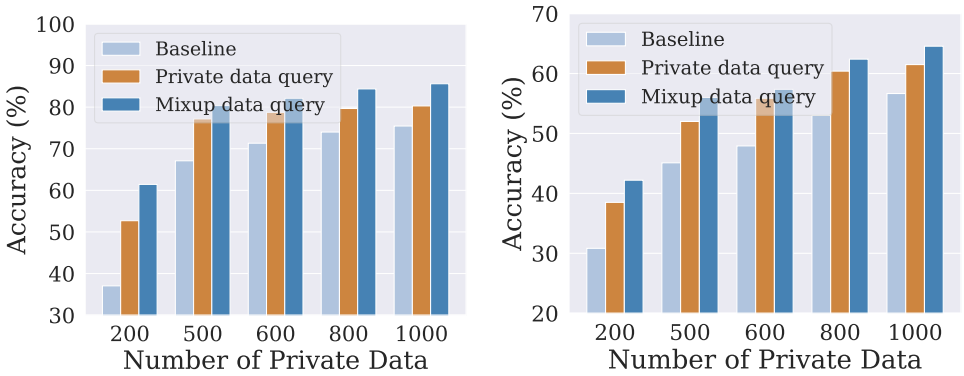
Figure 12 visualizes the distributions of Non-IID samples among clients with different Dirichlet distribution $\mathsf{Dir}(\alpha)$ on CIFAR10 dataset, where the number of clients $N = 20$. The value of α controls the degree of Non-IID-ness, in which a smaller α indicates higher degree of Non-IID-ness. When $\alpha = 100$, the distribution closes to uniform sampling. When $\alpha = 0.5$, the distribution of samples of each class among clients is extremely uneven. Figure 13 shows the test accuracy of PrivHFL on SVHN and CIFAR10 for different degrees of Non-IID-ness. We can observe that the higher the degree of Non-IID, the lower the accuracy of the model. However, fortunately, in this case, mixup-based method can still significantly improve the performance of models.

Figure 14 illustrates the impact of private data volume on the test accuracy. We can see that as the private data volume increases, the performance of models is on the rise. The reason is that the model can learn more knowledge from more data, and it can also generate more mixup synthetic data to query, so as to promote the sharing of model knowledge.

C Missing Details on PrivHFL

C.1 Algorithmic Description

Algorithm 1 gives the detailed description of the PrivHFL framework. In PrivHFL, each client first trains the local model M_j to convergence on the private dataset. Later, clients improve the performance of local models based on the knowledge of others via the predictions on the data they most uncertain. Therefore, they need to first construct a big unlabeled data pool based on the small number of private data, and then actively choosing a subset from the pool as the query data in each iteration. Next, they perform the secure querying protocol based on the query data, and obtains the



(a) SVHN (b) CIFAR10

Figure 14: **Impact of different amounts of private data on the accuracy of models on SVHN and CIFAR10.** The number of clients $N = 50$, and the number of private data for each client ranges from 200 to 1000.

final predictions, which will be used to retrain the local model. Query-data generation and secure querying process (refer to Section 3) will iterate T times until the local model M_j achieves pre-defined performance.

Algorithm 1 The secure PrivHFL framework.

Input: Each client P_j, $j \in 1, \cdots, n$ holding a private dataset $\mathcal{D}_j$ and a customized local model M_j.
Output: Trained models M_j, $j = 1, \cdots, n$

```
for each j ∈ 1, ··· , n do
    P_j locally trains the local model M_j on D_j using stochastic gradient descent optimization.
end for
for each querying party P_Q do
    P_Q trains the local model M_j on D_j using stochastic gradient descent optimization.
    P_Q generates an unlabeled synthetic pool D_pool utilizing mixup-based dataset expansion on
    the private dataset.
    P_Q applies active learning methods to select the query dataset D_Q from D_pool.
    P_Q secret-shares D_Q with P_A and the server.
    for each responding party P_A do
        P_A performs the secure querying protocol with P_Q and the server as shown in section 3.3.
        P_A sends the predictions to P_Q and the server.
    end for
    P_Q performs the secure result aggregation protocol with the server.
    P_Q retrains its local model based on the query dataset and private dataset.
end for
```

C.2 Active Learning Strategies

In order to choose query data that most likely contribute to improve the performance of local models, active learning is adopted. To be specific, active learning allows local models to actively choose the data from which they learn, which contains various pool sampling strategies to estimate the informativeness and diversity of unlabeled samplings. These strategies can be classified into two categories. One important class is that of uncertainty-based approaches, which try to find hard samples using heuristics like highest entropy, such as margin sampling and entropy sampling. There are recent optimization-based methods, such as greedy-k-center sampling to obtain a diverse subset of hard samples.

Definition 1 *(Margin Sampling Scheffer et al. (2001)).* Margin sampling assumes that the most informative samples are those which fall within this margin of decision boundary. Formally, given

an unlabeled dataset D, it can be represented as:

$$\hat{\mathbf{x}} = \arg\min_{x \in D} \left\{ \min_{\omega} |f(x, \omega)| \right\} \tag{2}$$

where $f(x, \omega)$ represents the distance of the data sample x from the hyperplane for class ω.

Definition 2 *(Entropy Sampling Shannon (2001)).* Using entropy as an uncertainty measure as follows:

$$\hat{\mathbf{x}} = \arg\max_{x \in D} -\sum_{i} P_\theta(y_i \mid x) \log P_\theta(y_i \mid x) \tag{3}$$

where $P_\theta(y \mid x)$ is the conditional label distribution of the model, y_i ranges over all possible labels.

Definition 3 *(Greedy-k-center Sampling Sener & Savarese (2018)).* Solving the k-center problem, i.e., choosing b center points such that the largest distance between a data point and its nearest center is minimized. Formally, this goal is defined as:

$$\min_{\mathbf{s}^1 : |\mathbf{s}^1| \leq b} \max_{i} \min_{j \in \mathbf{s}^1 \cup \mathbf{s}^0} \Delta(\mathbf{x}_i, \mathbf{x}_j) \tag{4}$$

where $\mathbf{s}^1$ is new chosen center points, $\mathbf{s}^0$ is the current training set.

Besides, to better balance uncertainty and diversity, we introduce informative-and-diverse sampling called *ICD*, which chooses query data based on informative and diverse criteria using margin and cluster-based sampling methods. It will return highest uncertainty lowest margin points while maintaining same distribution over clusters as entire dataset.

C.3 More background on Cryptography

Diffie-hellman key agreement protocol. In our secure querying protocol, we align all clients to achieve consistency aided by the Diffie-hellman key change (DH) protocol Diffie & Hellman (1976), which consists of following algorithms.

- $\mathsf{DH.param}(k) \rightarrow (\mathbb{G}, g, q, H)$ generates a group $\mathbb{G}$ of prime order q, along with a generator g, and a hash function H.
- $\mathsf{DH.gen}(\mathbb{G}, g, q, H) \rightarrow (x_i, g^{x_i})$ randomly samples $x_i \in \mathbb{Z}_q$ as the secret key and let g^{x_i} as the public key.
- $\mathsf{DH.agree}(x_i, g^{x_j}, H) \rightarrow s_{i,j}$ outputs the *seed* $s_{i,j} = H((g^{x_j})^{x_i})$.

In DH, correctness requires that for any key pairs (x_i, g^{x_i}) and (x_j, g^{x_j}) generated by two paries P_i and P_j using $\mathsf{DH.gen}(\cdot)$ under the same parameters $(\mathbb{G}, g, q, H)$, $\mathsf{DH.agree}(x_i, g^{x_j}, H) = \mathsf{DH.agree}(x_j, g^{x_i}, H)$. Besides, in the semi-honest adversary setting, security requires that for any adversary who steals g^{x_i} and g^{x_j} (but neither of the corresponding x_i and x_j), the agreed secret $s_{i,j}$ derived from those keys is indistinguishable from a uniformly random value. This exactly mirrors the Discrete Logarithm Problem, which we recall below.

Definition 4 *(Discrete Logarithm Problem).* Given a group $\mathbb{G}$ of prime order q along with a generator g, for any integer $b \in_R \mathbb{Z}_q$, find a unique integer c, satisfying $g^c \equiv b \ (mod\ q)$.

Generally, if choosing q carefully, Discrete Logarithm Problem is considered difficult to solve, and there is no polynomial time algorithm for computing this problem. In order to resist known attacks, q should be at least a 150-bit binary integer, and $q - 1$ has at least one large prime factor.

C.4 GPU Acceleration

Existing general-purpose platforms on the GPU such as NVIDIA's CUDA are designed to operate on floating-point inputs Tan et al. (2021). On the contrary, the shares in our protocol is embedded in the ring $\mathbb{Z}_{2^{64}}$. Thus, we need to convert the ring operations into 64-bit floating point operations to obtain GPU support. Specifically, a 64-bit floating-point number has 1 bit *sign*, 11 bits *exponent*, and 52 bits *precision*, which can exactly represent all integers in the interval $\left[-2^{52}, 2^{52}\right]$. Therefore, the multiplication $E \cdot F$ can be correctly computed and recovered over the integers if and only if $E, F \in \mathbb{Z} \cap \left[-2^{26}, 2^{26}\right]$. Based on this observation, we decompose each input E and F in

$\mathbb{Z}_{2^{64}}$ into 4 blocks, and the values in each block are represented by 16-bit value. Thus, the integer multiplication is converted to 16 (exactly, 10) floating-point operations. Specifically, E is rewritten as $E_0 + 2^{16}E_1 + 2^{32}E_2 + 2^{48}E_3$, and $F = F_0 + 2^{16}F_1 + 2^{32}F_2 + 2^{48}F_3$. The multiplication can be represented as follows:

$$\begin{aligned} E \cdot F &= (E_0 + 2^{16}E_1 + 2^{32}E_2 + 2^{48}E_3) \cdot (F_0 + 2^{16}F_1 + 2^{32}F_2 + 2^{48}F_3) \\ &= E_0F_0 + 2^{16}E_1F_0 + 2^{32}E_2F_0 + 2^{48}E_3F_0 + 2^{16}F_1E_0 + 2^{32}F_1E_1 + 2^{48}F_1E_2 \\ &+ 2^{32}F_2E_0 + 2^{48}F_2E_1 + 2^{48}F_3E_0 \end{aligned} \tag{5}$$

After that, computing $E \cdot F$ from the pairwise products requires element-wise additions and scalar multiplications, which can be executed by optimized CUDA kernels on 64-bit integer values.

D Related Work

Table 5: **Comparison with prior works on properties necessary for federated learning**

Framework	**Privacy**		**Usability**		**Efficiency**	
	Data Privacy	Model Privacy	Model Heterogeneity	w/o Dataset Dependency	GPU Compatibility	Protocol Efficiency
Bonawitz et al. (2017)	✓	✗	✗	✓	✗	✓
Bell et al. (2020)	✓	✗	✗	✓	✗	✓
Sav et al. (2021)	✓	✓	✗	✓	✗	✗
Jayaraman & Wang (2018)	✓	✗	✗	✓	✗	✓
Li & Wang (2019)	✗	✓	✓	✗	✓	-
Choquette-Choo et al. (2021)	✓	✓	✓	✗	✗	✗
Lin et al. (2020)	✗	✗	✓	✗	✓	-
Sun & Lyu (2021)	✗	✓	✓	✗	✓	✓
Diao et al. (2021)	✗	✗	✓	✓	✓	-
This work	✓	✓	✓	✓	✓	✓

D.1 Heterogeneous federated learning

Federated learning achieves collaboration among clients via sharing model gradients. While successful, it still faces many challenges, among which, of particular importance is the heterogeneity that appear in all aspects of the learning process. This consists of system heterogeneity Diao et al. (2021), model heterogeneity Li & Wang (2019) and statistical heterogeneity Zhu et al. (2021). Statistical heterogeneity means that clients' data from real-world comes from distinct distributions (i.e., Non-IID data), which may induce deflected local optimum. Solving the heterogeneous data has been extensively studied, such as Dinh et al. (2020) Zhu et al. (2021) Yurochkin et al. (2019) Fallah et al. (2020), and are out-of-the-scope of this work. However, our PrivHFL may help alleviate data heterogeneity, because of customized model design and model aggregation based on knowledge distillation, and we leave it in future work.

This work mainly focuses on model heterogeneity that has been explored in recent works Li & Wang (2019) Lin et al. (2020) Choquette-Choo et al. (2021), while the issue of system heterogeneous is alleviated through the resource-customized model architecture design. In particular, Li & Wang (2019) proposed the first federated learning framework FedMD supporting heterogeneous models by combining transfer learning and knowledge distillation techniques. They first used a public dataset to pre-train the model and transferred to the task of private dataset. After that, to exchange the knowledge , each client used the public data and the aggregated predictions from others as carrier for knowledge distillation. To further improve test accuracy, Lin et al. Lin et al. (2020) proposed FedDF, similar to FedMD, which also used model distillation technique for knowledge sharing. The difference is that they first performed FedAvg on clients' local models and integrated knowledge distillation on the aggregated model. The dependence on model averaging leads to limited model heterogeneity. Besides, Diao et al. Diao et al. (2021) focused on heterogeneous clients equipped with different computation and communication capabilities. In their framework, each client only

updated a subset of global model parameters through varying the width of hidden channels, which reduces the computation and communication complexity of local models. However, this approach only learns a single global model, rather than uniquely models designed by clients. Moreover, the above three methods rarely consider the issue of privacy leakage from the prediction results.

The privacy protection techniques (i.e., secure aggregation) have been studied in federated learning Bonawitz et al. (2017) Bell et al. (2020) Sav et al. (2021) Jayaraman & Wang (2018). However, these techniques can not be directly extended to privacy-preserving HFL. More recently, Sun & Lyu (2021) proposed a noise-free differential privacy solution for heterogeneous federated learning to guarantee each client's privacy. However, as shown in Jayaraman & Evans (2019), there is a huge gap between the upper bounds on privacy loss analyzed by advanced mechanisms and the effective privacy loss. Thus, differentially private mechanisms offer undesirable utility-privacy trade-offs. To further formally guarantee the privacy, Choquette-Choo et al. (2021) leveraged secure multi-party computation (MPC), homomorphic encryption (HE) and privately aggregated teacher models techniques to realize confidential and private collaborative learning. Specifically, clients learn from each other collaboratively utilizing a secure inference strategy based on MPC and HE protocols and a private aggregation method. As noted above, CaPC's use of heavy cryptography leads to significant efficiency and communication overheads. Besides, to best of our knowledge, existing methods all require a public dataset to implement heterogeneous federated learning. However, it is unrealistic in many data-critical scenarios, such as healthcare and finance. Therefore, our work is dedicated to solving two key challenges. The first is to relax the assumption of relying on public datasets, and the second is to design an efficient cryptographic protocol for knowledge transfer during the model aggregation.

D.2 Private neural network prediction

Neural networks present a challenge to cryptographic protocols due to their unique structure and exploitative combination of linear computations and non-linear activation functions. In real scenarios, model inference can be viewed as a two-party computation case, where one party with private data wants to obtain prediction results from the other party who owns the model. During the whole process, the cryptographic protocols, typically HE and MPC, are applied to ensure the confidentiality of private data and models.

Existing works Boemer et al. (2019b), Gilad-Bachrach et al. (2016), Brutzkus et al. (2019) support pure HE protocols for secure predictions. Typically, nGraph-HE Boemer et al. (2019b), Boemer et al. (2019a) allows linear computations using CKKS homomorphic encryption scheme. However, since a solution that builds upon HE protocols should be restricted to compute low degree polynomials, the non-polynomial activation functions, such as MaxPooling and ReLu, are forced to be evaluated in the clear by the party who owns private query data. This leaks the feature maps, from which adversaries may deduce the model weights. To fix it, Gilad-Bachrach et al. (2016), Chen et al. (2019) use low-degree polynomial approximation to estimate non-linear functions. Unfortunately, it will affect the accuracy of predictions. On the other hand, server libraries Mohassel & Zhang (2017), Knott et al. (2021), Wagh et al. (2019), Shen et al. (2020) employ primarily MPC technology in secure predictions, which provides support for linear and non-linear activations through the use of oblivious transfer (OT), garbled circuit (GC) and secret sharing. For example, CryptTen Knott et al. (2021) performs linear operations based on n-out-of-n additive secret sharing over a large ring, $\mathbb{Z}_{2^{64}}$. However, it uses boolean secret sharing for the non-linear operations, which will result in a higher communication round. CrpytGPU Tan et al. (2021) builds on CrypTen, working in a 3-party setting using *replicated secret shares*. Although the scalability is poor, it introduces an interface to losslessly embed cryptographic operations over secret-shared values in a discrete somain into floating-point operations, which can implement the whole inference process on the GPU.

Many other works focus on hybrid protocols, in which they combines the advantages of HE and MPC to improve prediction efficiency Juvekar et al. (2018), Mishra et al. (2020), Rathee et al. (2020). CrypTFlow2 Rathee et al. (2020) points out that it was not clear whether HE-based linear operations would provide the best latency. Therefore, the authors implement two class of protocols, HE-based and OT-based, for linear operations. For non-linear layers, they also design efficient protocols based on OT. It turns out that in a WAN setting, HE-based inference is always faster and in a LAN setting OT and HE are incomparable. HE-transformer employs nGraph-HE for evaluation of linear operations, and ABY framework Demmler et al. (2015) for GC to evaluate non-linear

functions. Since non-linear operations cannot be parallelized between query data, GC is inefficient, especially for large networks with thousands of parameters. In contrast, our PrivHFL avoids the use of heavy cryptographic tools, and only employs additive secret sharing to achieve high efficiency, confidentiality and practicability.

E Security Analysis

Our security proof follows the standard ideal-world/real-world paradigm: in real-world, three parties interact according to the protocol specification, and in ideal-world, they have access to a ideal functionality. When a protocol invokes another sub-protocol, we use $\mathcal{F}$-hybrid model for security proof by replacing the sub-protocol with the corresponding functionality. Note that our proof works in the $\mathcal{F}_{PRG}$-hybrid model where $\mathcal{F}_{PRG}$ represents the ideal functionality corresponding to the protocol PRG. The executions in both worlds are coordinated by the environment Env, who chooses the inputs to parties and plays the role of a distinguisher between the real and ideal executions. We will show that the real-world distribution is computationally indistinguishable to the ideal-world distribution.

Theorem 1. Π_{Share} securely realizes the functionality $\mathcal{F}_{\text{Share}}$ in the $\mathcal{F}_{\text{PRG}}$-hybrid model.

Proof. Note that P_Q and P_A receive no messages in the protocol, and hence the sharing protocol is trivially secure against corruption of P_Q and P_A. Next, the only message that P_S receives is the value $[x]_1$. However, $[x]_1 = x - r$, where given the security of PRG, r is a random value unknown to P_S. Thus, the distribution of $[x]_1$ is uniformly random from P_S's view and the information learned by P_S can be perfectly simulated.

□

Theorem 2. Π_{Matmul} securely realizes the functionality $\mathcal{F}_{\text{Matmul}}$ in the $\mathcal{F}_{\text{PRG}}$-hybrid model.

Proof. Note that P_Q receives no messages in the protocol, and hence the sharing protocol is trivially secure against corruption of P_Q. The only message that P_A receives is the value $[x]_1 - b$. However, given the security of PRG, b is a random value unknown to P_A. Thus, the distribution of $[x]_1 - b$ is uniformly random from P_A's view and the information learned by P_A can be perfectly simulated. Next, during the protocol, P_S learns $[c]_1$ and $w + a$. However, the distribution of $[c]_1$ and $w + a$ is uniformly random from P_S's view, since given the security of PRG, a and $[c]_1$ are random values unknown to P_S. Thus, the information learned by P_S can be perfectly simulated.

□

Theorem 3. Π_{DReLU} securely realizes the functionality $\mathcal{F}_{\text{DReLU}}$ in the $\mathcal{F}_{\text{PRG}}$-hybrid model.

Proof. Note that P_A receives no messages in the protocol, and hence the sharing protocol is trivially secure against corruption of P_A. Next, the messages that P_S receives are $[z]_0 - \delta$ and $\text{sign}(z) - \delta'$. However, given the security of PRG, δ and δ' are random values unknown to P_S. Thus, the distribution of $[z]_0 - \delta$ and $\text{sign}(z) - \delta'$ is uniformly random from P_S's view and the information learned by P_S can be perfectly simulated. Then, the message that P_S learns is $z = r'y$, where r' is a random positive number in the ring. Thus, the distribution of z is uniformly random from P_S's view. Note that the sign of y will be learned by P_S, but this is a allowed leakage as illustrated in our privacy definition.

□

Theorem 4. Π_{ReLU} securely realizes the functionality $\mathcal{F}_{\text{ReLU}}$ in the ($\mathcal{F}_{\text{Matmul}}$, $\mathcal{F}_{\text{DReLU}}$)-hybrid model.

Proof. Note that as shown in Figure 4, Π_{ReLU} consists of Π_{DReLU} and Π_{Matmul} Therefore, the ReLU protocol is trivially secure in the ($\mathcal{F}_{\text{Matmul}}$, $\mathcal{F}_{\text{DReLU}}$)-hybrid model.

□

Theorem 5. Π_{MaxPool} securely realizes the functionality $\mathcal{F}_{\text{MaxPool}}$ in the $\mathcal{F}_{\text{ReLU}}$-hybrid model.

Proof. Note that as shown in Figure 4, Π_{MaxPool} consists of Π_{ReLU}. Therefore, the MaxPool protocol is trivially secure in the $\mathcal{F}_{\text{ReLU}}$-hybrid model.

□

Theorem 6. Π_{Agg} securely realizes the functionality $\mathcal{F}_{\text{Agg}}$ in the $\mathcal{F}_{\text{PRG}}$-hybrid model.

Table 6: Experiment setting of different datasets during local model training.

	Loss Function	Learning Rate	Batch Size	Epoch	Number of Private Data
SVHN	cross-entropy	0.5	256	250	1465
CIFAR10	cross-entropy	0.1	64	500	1000
Tiny ImageNet	cross-entropy	0.01	64	500	10000

Proof. Note that P_A receives no messages in the protocol, and hence the aggregation protocol is trivially secure against corruption of P_A. Next, the only message that P_S receives is the value $[x_i]_1 - r_i$. However, given the security of PRG, r_i is a random value unknown to P_S. Thus, the distribution of $[x_i]_1 - r_i$ is uniformly random from P_S's view and the information learned by P_S can be perfectly simulated. After the aggregation, P_Q only learns the aggregated result $\sum_{i\in[n]} x_i$, but is unknown to a single x_i. Therefore, the aggregation protocol is secure assuming the aggregation result will not reveal privacy.

□

F Additional Experimental Setup

Datasets. We evaluate PrivHFL on the following standard datasets for image classification:

- **SVHN.** SVHN is a real-world image dataset obtained from house numbers in Google Street View images. Each sample is 32×32 RGB image. We use 73257 digits for training and 26032 digits for testing.
- **CIFAR10.** CIFAR-10 consists of 60,000 32×32 RGB images in 10 classes. There are 50,000 training images and 10,000 test images.
- **Tiny ImageNet.** Tiny ImageNet contains 100,000 images of 200 classes (500 for each class) downsized to 64×64 colored images. Each class has 500 training images, 50 validation images and 50 test images.

Model architecture. We use VGG-7, ResNet-8 and ResNet-10 for SVHN and CIFAR10 datasets. VGG-7 contains 6 convolutional layers, followed by a fully-connected layer, where ReLU is applied as the activation function and MaxPooling is used for downsampling, giving 7 layers in total. ResNet-8 we use follows the same architecture in CaPC, which consists of a convolutional layer, 2 residual blocks with 4 convolutional layers, followed by a fully-connected layer. Compared with the original architecture, the last block is excluded and neurons of the last layer is increased. ResNet-10 begins with a convolutional layer, followed by 4 residual blocks with 2 convolutional layers in each block, and one final fully-connected layer, thus there are 10 functional layers overall. Besides, we use ResNet-14, ResNet-16 and ResNet-18 He et al. (2016) for Tiny ImageNet.

Training procedure. Utilizing the private data, each client first trains the local model from scratch using SGD optimizer, where the detail training setting is shown in Table 6. Next, the clients run the PrivHFL protocol to generate query-response pairs, which will be used to retrain the local model. When retraining the models, they using Adam optimizer for 50 epochs with learning rate of 2e-3 decayed by a factor of 0.1 on 25 epochs, where the batch size is 256 on SVHN, and 64 on CIFAR10 and Tiny ImageNet.