DReS-FL: Dropout-Resilient Secure Federated Learning for Non-IID Clients via Secret Data Sharing

Jiawei Shao[‡], **Yuchang Sun**[‡], **Songze Li**[‡§], **Jun Zhang**[‡]
[‡]Hong Kong University of Science and Technology
[§]Hong Kong University of Science and Technology (Guangzhou)
{jiawei.shao,yuchang.sun}@connect.ust.hk, {songzeli,eejzhang}@ust.hk

Abstract

Federated learning (FL) strives to enable privacy-preserving training of machine learning models without centrally collecting clients' private data. Despite its advantages, the local datasets across clients in FL are non-independent and identically distributed (non-IID), and the data-owning clients may drop out of the training process arbitrarily. These characteristics will significantly degrade the training performance. Therefore, we propose a *Dropout-Resilient Secure Federated Learning* (DReS-FL) framework based on Lagrange coded computing (LCC) to tackle both the non-IID and dropout problems. The key idea is to utilize Lagrange coding to secretly share the private datasets among clients so that each client receives an encoded version of the *global dataset*[1], and the local gradient computation over this dataset is unbiased. To correctly decode the gradient at the server, the gradient function has to be a polynomial in a finite field, and thus we construct *polynomial integer neural networks* (PINNs) to enable our framework. Theoretical analysis shows that DReS-FL is resilient to client dropouts and provides strong privacy guarantees. Furthermore, we experimentally demonstrate that DReS-FL consistently leads to significant performance gains over baseline methods.

1 Introduction

Federated learning (FL) [1] is a machine learning framework in which a central server coordinates a large number of clients to collaboratively train a shared model. The key idea of FL is to train the model locally by individual clients and aggregate updates globally by the server. The main target is to provide privacy protection for clients' local samples and solve the "data islands" problem. However, as local data are typically non-independent and identically distributed (non-IID), the model divergence during the local update may lead to unstable and slow convergence [2, 3, 4]. With many clients involved in the training, some of the clients could drop out of the training process unexpectedly (due to poor connectivity, battery level, etc), and it will cause detrimental model performance [5]. Thus, effective mechanisms are needed to tackle the non-IID data distribution and client dropouts, while preserving the privacy of local datasets, which motivates this work.

To alleviate the non-IID problem, existing methods typically follow *algorithm-based approaches* [2, 6, 7, 8, 9] and add regularization terms to mitigate the model divergence. However, these methods are not dropout-resilient evidenced by the empirical results in [10]. This can be explained by the greatly varying data distributions among different rounds. Another fold of strategy for dealing with the non-IID problem is *data-centric approach* [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21], which generates extra training samples to construct a more balanced data distribution for each client. The common practices are to share the synthesized samples [13, 14, 15, 16] or GAN-based augmented

[1]In the context of this paper, we use the global dataset to denote the concatenation of the clients' datasets.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

data [17, 18, 19, 20, 21]. However, these methods may leak private information about local datasets and violate the privacy criterion in FL.

In this work, we develop a ***D**ropout-**Re**silient **S**ecure **F**ederated **L**earning* (DReS-FL) framework to address the above problems via Lagrange coded computing (LCC) [22]. The key idea of LCC is to encode the datasets using Lagrange polynomials that create computational redundancy across the workers in a privacy-preserving way to tolerate client dropouts. Before the training starts, the clients secretly share their encoded datasets with each other. This allows clients to access an encoded version of the global dataset that solves the data heterogeneous problem. In each communication round of federated training, the clients perform local gradient computations on the mini-batches sampled from the encoded datasets. After collecting the uploaded computation results from surviving clients, the server performs polynomial interpolation to decode the *global gradient*[2] for model training. Therefore, the training process in DReS-FL is made equivalent to centralized training and eliminates the non-IID and dropout problems. With respect to privacy protection, the proposed framework has two salient features:

- It guarantees the privacy of local datasets during data sharing, i.e., no private information can be inferred from the encoded data even if a certain number of clients collude.
- It achieves the same privacy guarantee as the *secure aggregation* protocols, i.e., the server learns no information about the private dataset from a single client's computation result.

Note that to correctly decode the gradient at the server, the gradient has to be a polynomial function in a finite field, which is a main design challenge of DreS-FL. To sum up, our main contributions are summarized as follows:

- The proposed DReS-FL framework provides a unified approach to tackle two critical problems of FL, namely, *non-IID data distribution* and *client dropouts*. Meanwhile, it maintains privacy and security guarantees such that no information about local datasets can be leaked beyond the global model parameters.
- We construct *polynomial integer neural networks* (PINNs) to ensure that the gradient is a polynomial, so that cryptographic primitives can be applied for secure computation. A PINN consists of affine transformation layers with parameters constrained in an integer set, and it adopts the quadratic function as the activation function. The convergence analysis of DReS-FL with PINNs is also provided.
- We conduct extensive experiments on FL benchmark datasets to demonstrate the effectiveness of DReS-FL. It is shown that DReS-FL outperforms baseline methods under the setting where local datasets are heterogeneous and clients may drop out of the training process arbitrarily.

2 Related Works

Non-IID data and client dropouts. Training with heterogeneous data is a unique challenge for FL [1], which significantly affects the convergence performance [5]. The client dropouts exacerbate the non-IID problem as the data distributions among different rounds could vary greatly. Many algorithm-based methods [2, 6, 7, 8, 9] attempt to mitigate the clients' model divergence, but these methods cannot solve the essence of the non-IID problem due to the intrinsic difference between minimizing the local empirical loss and minimizing the global empirical loss. Another line of work adopts data-centric methods [17, 18, 19, 20, 21] to modify the local distributions. Ideally, a perfect data sharing mechanism should achieve that the local datasets have the same distribution as the global dataset while maintaining the privacy guarantee. Common practices include sharing raw datasets [11, 12], synthesized samples, [13, 14, 15, 16] or augmented data [17, 18, 19, 20, 21]. However, these works cannot fully preserve local data privacy in an information-theoretic sense [23]. A special data-centric method is the secret coding scheme, which has been widely utilized

[2]The global gradient corresponds to the stochastic gradient computed from mini-batches that are uniformly sampled from the global dataset. For simplicity, we consider that clients only perform one local stochastic gradient descent (SGD) step in each communication round. Note that the proposed DReS-FL framework, as discussed in Appendix D, can be extended to more general cases in which clients can run multiple local SGD steps.

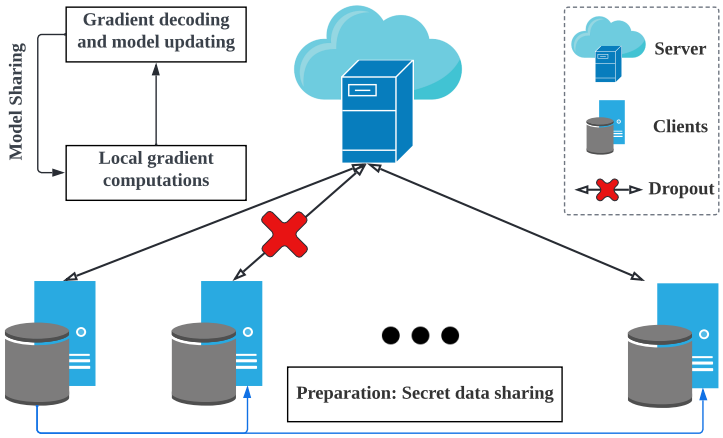


Figure 1: The DReS-FL system model. At the beginning of training, the clients secretly share the local datasets with each other. Then, the model parameters are iteratively trained by (1) local gradient computations and (2) gradient decoding and model updating until convergence.

in homomorphic encryption (HE) [24, 25, 26, 27, 28, 29, 30] and multiparty computation (MPC) techniques [31, 32, 33]. This coding scheme allows computations to be performed on encrypted data and has been used for privacy-preserving machine learning [26, 30, 33]. However, the HE methods often suffer from time-consuming cryptographic tools, and MPC techniques are difficult to generalize such primitives to a large number of clients. Recently, distributed secure machine learning frameworks [34, 35] have been proposed for logistic regression problems. They apply Lagrange coding for secret data sharing and approximate the Sigmoid function by a polynomial function. This paper proposes DReS-FL to further extend these works to train deep neural networks in the FL setting.

Secure aggregation. It has been shown recently that the clients' updates in FL may reveal substantial information about the local datasets, and the private training data can be reconstructed through model inversion attacks [36, 37, 38]. To prevent information leakage from the local models, *secure aggregation* protocols [39, 40, 41, 42] have been developed to allow for global aggregation without revealing the parameters of clients' models. Even if some clients may drop out, these protocols can still recover the aggregated results of the surviving clients. Existing protocols essentially rely on two main principles, including a pairwise random-seed agreement for mask cancellation and secret sharing of the random seeds to construct the dropped masks [39, 43, 40, 41, 44, 42]. However, these approaches may suffer from severe performance degradation in non-IID settings, since the surviving clients in each round vary greatly, and thus the aggregate gradient is biased towards different data distributions. Different from previous works, our proposed DReS-FL framework achieves the same privacy guarantee while solving the data heterogeneity problem.

3 System Model

We consider a federated learning framework as shown in Fig. 1 that consists of one central server and N data-owning clients. Each client $i \in [N]$ holds a local dataset $(\mathbf{X}_i, \mathbf{Y}_i)$ of size m_i, where $\mathbf{X}_i \in \mathbb{R}^{m_i \times d_x}$ represents the set of input features of dimension d_x and $\mathbf{Y}_i \in \mathbb{R}^{m_i \times d_y}$ corresponds to the output vector of dimension d_y. Accordingly, the size of the global dataset $(\mathbf{X}, \mathbf{Y})$ which concatenates all local datasets $(\mathbf{X}_i, \mathbf{Y}_i), \forall i \in [N]$ is denoted as $m \triangleq \sum_{i=1}^{N} m_i$. The clients aim to jointly train a neural network based on their local datasets without sharing private data samples. Particularly, the gradients are computed locally and aggregated globally. However, the local data may be highly heterogeneous, and the clients may drop out at any time unexpectedly, which makes the training process unstable. Our goal is to improve the convergence performance by secret data sharing while preserving the privacy of local datasets.

3.1 Lagrange Coded Computing for Federated Learning

The Lagrange coded computing (LCC) framework enables private computing in distributed settings to provide resiliency and efficiency [22]. The key idea is using Lagrange coding to encode the data for redundant distributed computing, which fits nicely with federated learning due to its dropout-resiliency and privacy requirement.

Specifically, the clients share their encoded datasets with each other and perform gradient computation over the encoded samples. The server decodes the aggregate gradient after receiving the uploaded computation results from clients. To provide a strong privacy guarantee for the datasets and correctly decode the gradient at the server, the gradient function should be a polynomial function in a finite field. However, existing neural networks cannot satisfy this requirement, since the datasets are in the real field and the gradients are non-polynomial.

Polynomial integer neural networks. We define a class of polynomial integer neural networks (PINNs) to ensure that the gradient is a polynomial function in a finite field $\mathbb{F}_p$ with a prime number p. First, we transform the dataset $(\mathbf{X}, \mathbf{Y})$ from the real domain to the finite domain $(\overline{\mathbf{X}}, \overline{\mathbf{Y}})$. Besides, a PINN consists of affine transformation layers (e.g., fully connected layers and convolutional layers) and utilizes the quadratic function as the activation function. The model parameters of PINNs are defined in the integer set $\mathbb{Z}_p \triangleq \{-\lfloor \frac{p+1}{2} \rfloor, \ldots, \lfloor \frac{p-1}{2} \rfloor\}$. Given a feed-forward function $\boldsymbol{f}(\overline{\mathbf{X}}; \mathbf{w})$ and selecting the mean squared error (MSE) as the loss function, the gradient of the input samples is a multivariate polynomial with integer coefficients, i.e., $\boldsymbol{g}(\overline{\mathbf{X}}, \overline{\mathbf{Y}}; \mathbf{w}) \triangleq \nabla_{\mathbf{w}} \|\overline{\mathbf{Y}} - \boldsymbol{f}(\overline{\mathbf{X}}; \mathbf{w})\|_2^2 \in \mathbb{Z}^{d_w}$, where d_w represents the number of model parameters. Denoting the number of quadratic activation layers as L, the degree of the gradient function[3] $\boldsymbol{g}(\overline{\mathbf{X}}, \overline{\mathbf{Y}}; \mathbf{w})$ is $\deg(\boldsymbol{g}) = 2^{L+1}$.

In particular, to avoid wrap-around when computing gradient in the finite field $\mathbb{F}_p$, we assume the prime number p is sufficiently large without leading to overflow errors in the integer set $\mathbb{Z}_p$.

Lagrange coding. The proposed DReS-FL framework uses Lagrange polynomials to achieve a D-resilient, T-private, and K-efficient coding scheme. D-Resiliency means that the global gradient can be decoded by the server in the presence of up to D client dropouts. T-privacy denotes that no information about local datasets can be inferred from the encoded data even if up to T clients collude. K-efficiency corresponds to the complexity of the coding scheme. Specifically, each private dataset is split into K shards in Lagrange coding, and the size of the encoded dataset is proportional to $1/K$. Therefore, increasing the value of K reduces the communication overhead of data sharing. The following theorem characterizes the (D, T, K)-achievable coding scheme, and its proof is available in Section IV of [22].

Theorem 1. *Given the client number N and the degree of the gradient function $\deg(\boldsymbol{g})$, a D-resilient, T-private, and K-efficient Lagrange coding scheme is achievable, as long as*

$$D + \deg(\boldsymbol{g})(K + T - 1) + 1 \leq N. \tag{1}$$

Remark 1. As shown in Theorem 1, there is a tradeoff among resiliency (D), privacy (T), and efficiency (K). As the sum of T and K increases, the proposed framework tolerates fewer client dropouts. Specifically, the maximum value of D is $N - 1 - \deg(\boldsymbol{g})$ by setting $T = K = 1$.

Remark 2. Setting the privacy parameter $T \geq 1$, the gradient computation over the encoded samples leaks no private information according to the data process inequality. This implies that the proposed DReS-FL framework achieves the same privacy guarantee as the secure aggregation protocols. Specifically, the server learns no information about the private dataset from a single client's computation result.

4 DReS-FL Framework

DReS-FL consists of two main phases, as shown in Fig. 1. In the first phase, the private datasets are transformed from the real domain to the finite field, and data-owning clients secretly share datasets by Lagrange coding. Then, the server and the clients train a PINN iteratively via (1) local gradient computations and (2) gradient decoding and model updating.

[3] More details about how to calculate the degree of the gradient are deferred to Appendix E.

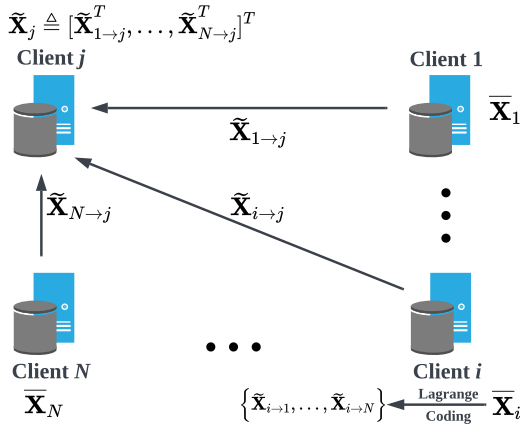


Figure 2: The secret data sharing scheme in the DReS-FL framework. Every client $i \in [N]$ secretly shares the local dataset $\overline{\mathbf{X}}_i$ to other clients $j \in [N]\backslash\{i\}$ by sending the encoded samples $\widetilde{\mathbf{X}}_{i\to j}$. The client j will receive $\widetilde{\mathbf{X}}_j \triangleq [\widetilde{\mathbf{X}}_{1\to j}^T, \ldots, \widetilde{\mathbf{X}}_{N\to j}^T]^T$ as an encoded version of the global dataset.

4.1 Data Transformation and Secret Sharing

To guarantee information-theoretic privacy, each client has to mask the datasets in a finite field $\mathbb{F}_p$ using uniformly random matrices. Firstly, the local datasets $(\mathbf{X}_i, \mathbf{Y}_i)$ are converted from the real domain to the finite field $(\overline{\mathbf{X}}_i, \overline{\mathbf{Y}}_i)$. Considering an element-wise function $\phi(z) = z + c$ that transforms a real value to a non-negative number by adding a proper scalar c[4], we define $\overline{\mathbf{X}} \triangleq Round(2^l \cdot \phi(\mathbf{X}))$, where the rounding operation is element-wise that quantizes each entry to its closest integer, and $l \in \mathbb{Z}$ controls the quantization loss. We adopt the notation $\overline{\mathcal{D}}$ to represent the *global dataset*, which is the concatenation of all the local datasets $(\overline{\mathbf{X}}_i, \overline{\mathbf{Y}}_i)$ for $i \in [N]$.

After converting the private datasets to the finite field, the clients adopt a (D, T, K)-achievable Lagrange coding to encode local data for secret sharing. First, each client $i \in [N]$ partitions its local dataset to K shards as $\overline{\mathbf{X}}_i \triangleq [\overline{\mathbf{X}}_i^{(1)T}, \ldots, \overline{\mathbf{X}}_i^{(K)T}]^T$ and $\overline{\mathbf{Y}}_i \triangleq [\overline{\mathbf{Y}}_i^{(1)T}, \ldots, \overline{\mathbf{Y}}_i^{(K)T}]^T$. Assuming that m_i is divisible by K, we have $\overline{\mathbf{X}}_i^{(k)} \in \mathbb{F}_p^{\frac{m_i}{K} \times d_x}$ and $\overline{\mathbf{Y}}_i^{(k)} \in \mathbb{F}_p^{\frac{m_i}{K} \times d_y}$ for $k \in [K]$. A large value of K helps to reduce the complexity in secret data sharing. Then, the clients add padding from T uniform random masks to the data samples for privacy protection. Each client $i \in [N]$ forms the following polynomials $\mathbf{u}_i : \mathbb{F}_p \to \mathbb{F}_p^{\frac{m_i}{K} \times d_x}$ and $\mathbf{v}_i : \mathbb{F}_p \to \mathbb{F}_p^{\frac{m_i}{K} \times d_y}$ of degree $K + T - 1$ to encode the local dataset:

$$\mathbf{u}_i(z) \triangleq \sum_{k\in[K]} \overline{\mathbf{X}}_i^{(k)} \cdot \prod_{j\in[K+T]\backslash\{k\}} \frac{z - \beta_j}{\beta_k - \beta_j} + \sum_{k=K+1}^{K+T} \mathbf{U}_i^{(k)} \cdot \prod_{j\in[K+T]\backslash\{k\}} \frac{z - \beta_j}{\beta_k - \beta_j}, \tag{2}$$

$$\mathbf{v}_i(z) \triangleq \sum_{k\in[K]} \overline{\mathbf{Y}}_i^{(K)} \cdot \prod_{j\in[K+T]\backslash\{k\}} \frac{z - \beta_j}{\beta_k - \beta_j} + \sum_{k=K+1}^{K+T} \mathbf{V}_i^{(k)} \cdot \prod_{j\in[K+T]\backslash\{k\}} \frac{z - \beta_j}{\beta_k - \beta_j}, \tag{3}$$

where $\{\mathbf{U}_i^{(k)}\}$'s and $\{\mathbf{V}_i^{(k)}\}$'s are random noise matrices uniformly sampled from $\mathbb{F}_p^{\frac{m}{K} \times d_x}$ and $\mathbb{F}_p^{\frac{m}{K} \times d_y}$, respectively. These matrices mask the local datasets and provide a privacy guarantee against up to T colluding workers. The clients and the server agree on $K + T$ distinct elements $\{\beta_1, \ldots, \beta_{K+T}\}$ from the finite field $\mathbb{F}_p$ in advance. Particularly, setting $z = \beta_k$ for $k \in [K]$, we reconstruct the data shard $(\mathbf{u}_i(\beta_k), \mathbf{v}_i(\beta_k)) = (\overline{\mathbf{X}}_i^{(k)}, \overline{\mathbf{Y}}_i^{(k)})$. All the clients use the same N distinct elements $\{\alpha_1, \ldots, \alpha_N\}$ selected from $\mathbb{F}_p$ to encode the private datasets, where $\{\alpha_j\}_{j\in[N]} \cap \{\beta_k\}_{k\in[K+T]} = \varnothing$. Each client i obtains N encoded datasets $(\widetilde{\mathbf{X}}_{i\to j}, \widetilde{\mathbf{Y}}_{i\to j}) \triangleq (\mathbf{u}_i(\alpha_j), \mathbf{v}_i(\alpha_j))$ for $j \in [N]$, where each $(\widetilde{\mathbf{X}}_{i\to j}, \widetilde{\mathbf{Y}}_{i\to j})$ is sent to client j from client i. All the received encoded

[4]The scalar c could be the absolute value of the minimum entry in dataset, which is set to 0 in the experiments.

Table 1: Primary notations and descriptions

Notation	Description	Notation	Description
$(\overline{\mathbf{X}}_i, \overline{\mathbf{Y}}_i)$	Transformed dataset at client i in a finite filed $\mathbb{F}_p$	$\overline{\mathcal{D}}$	The global dataset that is a concatenation of clients' datasets $(\overline{\mathbf{X}}_i, \overline{\mathbf{Y}}_i)$ for $i \in [N]$
$(\overline{\mathbf{X}}_i^{(k)}, \overline{\mathbf{Y}}_i^{(k)})$	k-th data shard at client i	$(\widetilde{\mathbf{X}}_{i\to j}, \widetilde{\mathbf{Y}}_{i\to j})$	Encoded dataset sent from client i to client j
$(\widetilde{\mathbf{X}}_j, \widetilde{\mathbf{Y}}_j)$	Concatenation of received encoded datasets $(\widetilde{\mathbf{X}}_{i\to j}, \widetilde{\mathbf{Y}}_{i\to j})$ for $i \in [N]$ at client j	$\mathbf{C}^{(t)}, \mathcal{I}_t$	Row selection matrix $\mathbf{C}^{(t)}$ for data sampling in round t and the corresponding index set $\mathcal{I}_t$
$(\widetilde{\mathbf{X}}_j^{(\mathcal{I}_t)}, \widetilde{\mathbf{Y}}_j^{(\mathcal{I}_t)})$	Local mini-batch sampled from $(\widetilde{\mathbf{X}}_j, \widetilde{\mathbf{Y}}_j)$ at client j in round t based on $\mathcal{I}_t$	$(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)})$	k-th global mini-batch sampled from $\overline{\mathcal{D}}$ in round t based on $\mathcal{I}_t$

Algorithm 1 DReS-FL

Input: Local datasets $(\mathbf{X}_i, \mathbf{Y}_i)$ for $i \in [N]$, batch size b, initialized parameters $\mathbf{w}^{(0)} \in \mathbb{Z}_p^{d_w}$, distinct elements $\{\alpha_j\}_{j\in[N]}$ and $\{\beta_k\}_{k\in[K+T]}$, prime number p, training round τ, learning rate η.
Output: Model parameter $\mathbf{w}^{(\tau)}$.
Clients encode the local datasets according to (2) and (3) and deliver them to other clients.
for $t = 1, 2, \ldots, \tau$ **do**
  Server sends the model parameters $\mathbf{w}^{(t)}$ to the clients.
  for $j = 1, \ldots, N$ **do**
    Client j performs gradient computation on mini-batches $(\widetilde{\mathbf{X}}_j^{(\mathcal{I}_t)}, \widetilde{\mathbf{Y}}_j^{(\mathcal{I}_t)})$.
    Upload local computation results $\widetilde{\boldsymbol{g}}(\widetilde{\mathbf{X}}_j^{(\mathcal{I}_t)}, \widetilde{\mathbf{Y}}_j^{(\mathcal{I}_t)}; \mathbf{w}^{(t)})$ to the server.
  end for
  if Server receives at least $\deg(\boldsymbol{g})(K+T-1)+1$ uploads **then**
    Decode K global gradients $\widetilde{\boldsymbol{g}}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)})$ for $k \in [K]$ by polynomial interpolation.
    Convert gradients from the finite field to the integral domain $\boldsymbol{g}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)})$ by (6).
    Update the model by (7) based on the aggregate gradient $\sum_{k=1}^{K} \boldsymbol{g}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)})$.
  end if
end for

datasets at client j are represented as $(\widetilde{\mathbf{X}}_j, \widetilde{\mathbf{Y}}_j)$, where $\widetilde{\mathbf{X}}_j \triangleq [\widetilde{\mathbf{X}}_{1\to j}^T, \ldots, \widetilde{\mathbf{X}}_{N\to j}^T]^T \in \mathbb{F}_p^{\widetilde{m}\times d_x}$ and $\widetilde{\mathbf{Y}}_j \triangleq [\widetilde{\mathbf{Y}}_{1\to j}^T, \ldots, \widetilde{\mathbf{Y}}_{N\to j}^T]^T \in \mathbb{F}_p^{\widetilde{m}\times d_y}$ for $j \in [N]$. Accordingly, the number of samples in the encoded dataset is $\widetilde{m} \triangleq \frac{1}{K}\sum_{i=1}^{n} m_i$. Fig. 2 demonstrates the secret data sharing scheme.

4.2 Federated Training

Local Gradient Computation. The server randomly initializes a PINN at the beginning of the training process, and the model parameters are constrained to an integer set $\mathbb{Z}_p$ during the training process. In each communication round, the server sends the model parameters to the clients, and they compute the stochastic gradient over the mini-batches with size b. Particularly, we assume that all the clients use the same row selection matrix $\mathbf{C}^{(t)} \in \{0,1\}^{b\times\widetilde{m}}$ for data sampling in each round t[5], and the mini-batch at each client $j \in [N]$ is determined by $[\widetilde{\mathbf{X}}_j^{(\mathcal{I}_t)}, \widetilde{\mathbf{Y}}_j^{(\mathcal{I}_t)}] \triangleq \mathbf{C}^{(t)}[\widetilde{\mathbf{X}}_j, \widetilde{\mathbf{Y}}_j]$. Here, $\mathcal{I}_t = \{l_1^{(t)}, \ldots, l_b^{(t)}\} \subseteq [\widetilde{m}]$ is a randomly selected index set in the t-th round with $l_i \in [\widetilde{m}]$ for $i \in [b]$. The entries of $\mathbf{C}^{(t)}$ satisfy $\mathbf{C}_{i,l_i}^{(t)} = 1$ for $i \in [b]$, and other entries are set to zero. Each client j computes the stochastic gradient $\widetilde{\boldsymbol{g}}(\widetilde{\mathbf{X}}_j^{(\mathcal{I}_t)}, \widetilde{\mathbf{Y}}_j^{(\mathcal{I}_t)}; \mathbf{w}^{(t)}) \triangleq \boldsymbol{g}(\widetilde{\mathbf{X}}_j^{(\mathcal{I}_t)}, \widetilde{\mathbf{Y}}_j^{(\mathcal{I}_t)}; \mathbf{w}^{(t)}) \mod p$ in the finite field, and uploads the result to the server. Particularly, each $\widetilde{\boldsymbol{g}}(\widetilde{\mathbf{X}}_j^{(\mathcal{I}_t)}, \widetilde{\mathbf{Y}}_j^{(\mathcal{I}_t)}; \mathbf{w}^{(t)})$ amounts to an evaluation of the polynomial $\widetilde{\boldsymbol{g}}(\mathbf{u}_{\mathcal{I}_t}(z), \mathbf{v}_{\mathcal{I}_t}(z); \mathbf{w}^{(t)})$ at the point $z = \alpha_j$, where two $(K+T-1)$-degree polynomial functions $\mathbf{u}_{\mathcal{I}_t} : \mathbb{F}_p \to \mathbb{F}_p^{b\times d_x}$ and $\mathbf{v}_{\mathcal{I}_t} : \mathbb{F}_p \to \mathbb{F}_p^{b\times d_y}$ are defined as

[5]This can be achieved by setting the same random seed across all the clients. The weighted sampling method has been adopted in this work, where the number of sampled data from $\widetilde{\mathbf{X}}_{i\to j}$ is proportional to m_i for $i \in [N]$. Note that other sampling schemes can also be applied in DReS-FL.

follows:

$$\mathbf{u}_{\mathcal{I}_t}(z) \triangleq \sum_{k\in[K]} \overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)} \cdot \prod_{j\in[K+T]\setminus\{k\}} \frac{z-\beta_j}{\beta_k-\beta_j} + \sum_{k=K+1}^{K+T} \mathbf{U}_{\mathcal{I}_t}^{(k)} \cdot \prod_{j\in[K+T]\setminus\{k\}} \frac{z-\beta_j}{\beta_k-\beta_j}, \tag{4}$$

$$\mathbf{v}_{\mathcal{I}_t}(z) \triangleq \sum_{k\in[K]} \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)} \cdot \prod_{j\in[K+T]\setminus\{k\}} \frac{z-\beta_j}{\beta_k-\beta_j} + \sum_{k=K+1}^{K+T} \mathbf{V}_{\mathcal{I}_t}^{(k)} \cdot \prod_{j\in[K+T]\setminus\{k\}} \frac{z-\beta_j}{\beta_k-\beta_j}, \tag{5}$$

with

$$\begin{aligned}
\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)} =& \mathbf{C}^{(t)}[\overline{\mathbf{X}}_1^{(k)T}, \ldots, \overline{\mathbf{X}}_N^{(k)T}]^T \in \mathbb{F}_p^{b\times d_x}, \quad \mathbf{U}_{\mathcal{I}_t}^{(k)} = \mathbf{C}^{(t)}[\mathbf{U}_1^{(k)T}, \ldots, \mathbf{U}_N^{(k)T}]^T \in \mathbb{F}_p^{b\times d_x}, \\
\overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)} =& \mathbf{C}^{(t)}[\overline{\mathbf{Y}}_1^{(k)T}, \ldots, \overline{\mathbf{Y}}_N^{(k)T}]^T \in \mathbb{F}_p^{b\times d_y}, \quad \mathbf{V}_{\mathcal{I}_t}^{(k)} = \mathbf{C}^{(t)}[\mathbf{V}_1^{(k)T}, \ldots, \mathbf{V}_N^{(k)T}]^T \in \mathbb{F}_p^{b\times d_y}.
\end{aligned}$$

Every $(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}) = (\mathbf{u}_{\mathcal{I}_t}(\beta_k), \mathbf{v}_{\mathcal{I}_t}(\beta_k))$ for $k \in [K]$ is a *global mini-batch* selected from the global dataset $\overline{\mathcal{D}}$.

Gradient Decoding and Model Updating. According to (4) and (5), the server can obtain the global gradients $\widetilde{\boldsymbol{g}}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)})$ for $k \in [K]$ by evaluating the polynomial $\widetilde{\boldsymbol{g}}(\mathbf{u}_{\mathcal{I}_t}(z), \mathbf{v}_{\mathcal{I}_t}(z); \mathbf{w}^{(t)})$ at the point $z = \beta_k$. But, the server needs to first recover the coefficients of this polynomial, which is a composition of the encoding polynomials $(\mathbf{u}_{\mathcal{I}_t}(z), \mathbf{v}_{\mathcal{I}_t}(z))$ and the gradient function $\widetilde{\boldsymbol{g}}$. As the degree of the composite polynomial is $\deg(\boldsymbol{g})(K+T-1)$, the server requires at least $\deg(\boldsymbol{g})(K+T-1)+1$ local computation results (i.e., evaluation points) to interpolate it[6]. This implies that the proposed DReS-FL framework can tolerate at most $D = N - \deg(\boldsymbol{g})(K+T-1) - 1$ client dropouts.

After decoding the global gradients, the server converts them from the finite field to the integer set $\mathbb{Z}_p$ by $\boldsymbol{g}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)}) = \psi(\widetilde{\boldsymbol{g}}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)}))$, where $\psi(z)$ is an element-wise function defined as follows:

$$\psi(z) = \begin{cases} z & \text{if} \quad 0 \le z < \frac{p-1}{2}, \\ z-p & \text{if} \quad \frac{p-1}{2} \le z < p. \end{cases} \tag{6}$$

As we assume that the prime number p is sufficiently large, the converted gradients do not have overflow errors. Thus, the central sever updates the global model by $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - Q(\frac{\eta}{bK}\sum_{k=1}^{K} \boldsymbol{g}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)})$, where η denotes the learning rate and bK represents the *global batch size*[7]. $Q(z)$ is a stochastic quantization function to ensure the model parameters are in the integer set $\mathbb{Z}_p$ after updating, which is defined as follows:

$$Q(z) = \begin{cases} \lfloor z \rfloor & \text{with probability } 1-(z-\lfloor z \rfloor) \\ \lfloor z \rfloor + 1 & \text{with probability } z - \lfloor z \rfloor. \end{cases} \tag{7}$$

Besides, the probability of rounding z to $\lfloor z \rfloor$ is proportional to the proximity of z to $\lfloor z \rfloor$ so that the stochastic rounding is unbiased. The overall procedure is summarized in Algorithm 1.

5 Convergence Analysis

In this section we characterize the convergence performance of PINNs, which relies on the fact that the global gradients in the training process are unbiased. Define the empirical risk as $\ell(\mathbf{w}) \triangleq \mathbb{E}_{(\overline{\mathbf{X}},\overline{\mathbf{Y}})\sim\overline{\mathcal{D}}}\|\overline{\mathbf{Y}} - \boldsymbol{f}(\overline{\mathbf{X}}; \mathbf{w})\|_2^2$ and the corresponding gradient as $\boldsymbol{g}_e(\mathbf{w}) \triangleq \mathbb{E}_{(\overline{\mathbf{X}},\overline{\mathbf{Y}})\sim\overline{\mathcal{D}}}[\boldsymbol{g}(\overline{\mathbf{X}}, \overline{\mathbf{Y}}; \mathbf{w})]$. The variables $(\overline{\mathbf{X}}, \overline{\mathbf{Y}})$ are drawn from the distribution of the global dataset $\overline{\mathcal{D}}$. To prove that DReS-FL guarantees convergence to the optimal model parameters, we first present the following assumptions to facilitate the analysis.

Assumption 1. *(L-smoothness) There exists a constant $L > 0$ such that for all $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{Z}_p^{d_w}$, we have $\|\boldsymbol{g}_e(\mathbf{w}_1) - \boldsymbol{g}_e(\mathbf{w}_2)\|_2 \le L\|\mathbf{w}_1 - \mathbf{w}_2\|_2$.*

[6]Note that if the server cannot receive enough results due to the client dropouts, the training protocol continues to the next epoch without gradient decoding and model updating.

[7]The global batch size corresponds to the number of samples used for gradient computations in each round.

Table 2: Test accuracy (%) of different methods. Each experiment is repeated five times. Best results are shown in italic and second best results are in bold.

Dataset	MNIST	Fashion-MNIST	EMNIST	CIFAR-10	CIFAR-100	SVHN
FedAvg	96.17 ± 0.05	81.20 ± 0.07	71.50 ± 0.28	89.54 ± 0.09	67.71 ± 0.26	83.82 ± 0.20
FedAvg-IS	97.06 ± 0.10	85.94 ± 0.16	77.09 ± 0.34	89.83 ± 0.07	68.92 ± 0.14	85.27 ± 0.09
SCAFFOLD	71.89 ± 3.92	55.22 ± 1.83	55.15 ± 5.95	54.17 ± 9.13	29.97 ± 1.73	51.27 ± 3.43
DReS-FL (Ours)	**97.38 ± 0.08**	**86.60 ± 0.32**	**78.04 ± 0.29**	**90.31 ± 0.19**	**69.15 ± 0.27**	**86.04 ± 0.15**
Centralized	*97.99 ± 0.04*	*89.02 ± 0.11*	*82.45 ± 0.23*	*90.37 ± 0.12*	*71.12 ± 0.09*	*86.18 ± 0.03*

Assumption 2. *(Unbiased and variance-bounded stochastic gradient) There exists a constant $\sigma > 0$ such that any stochastic gradient $\boldsymbol{g}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)})$ satisfies $\mathbb{E}\left[\frac{1}{b}\boldsymbol{g}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)})\right] = \boldsymbol{g}_e(\mathbf{w}^{(t)})$ and $\mathbb{E}\left[\|\frac{1}{b}\boldsymbol{g}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)}) - \boldsymbol{g}_e(\mathbf{w}^{(t)})\|^2\right] \leq \sigma^2$.*

Assumption 3. *(Unbiased and variance-bounded rounding operation) There exists a constant $\gamma > 0$ such that for any $z \in \mathbb{R}$, the stochastic quantization operation $Q(\cdot)$ satisfies $\mathbb{E}\left[Q(z)\right] = z$ and $\mathbb{E}\left[\|Q(z) - z\|^2\right] \leq \gamma^2 z^2$.*

With the above preparations, we have the following theorem which ensures the convergence. The proof is deferred to Appendix A.

Theorem 2. *(Convergence) Denote $\mathbf{w}^*$ as the first-order optimal solution. With Assumption 1-3, selecting the learning rate as $\eta = \mathcal{O}\left(1/\sqrt{\tau'}\right)$ such that $\Psi \triangleq 1 - \eta L/2 - \eta\gamma^2 L/2 > 0$, after τ' times of model updates, we have:*

$$\frac{1}{\tau'}\sum_{t=1}^{\tau'}\mathbb{E}\left[\|\boldsymbol{g}_e(\mathbf{w}^{(t)})\|^2\right] \leq \frac{\ell(\mathbf{w}^{(0)}) - \ell(\mathbf{w}^*)}{\eta\tau'\Psi} + \frac{\eta^2 L\sigma^2}{2bK\Psi}(\gamma^2 + 1). \tag{8}$$

6 Experiments

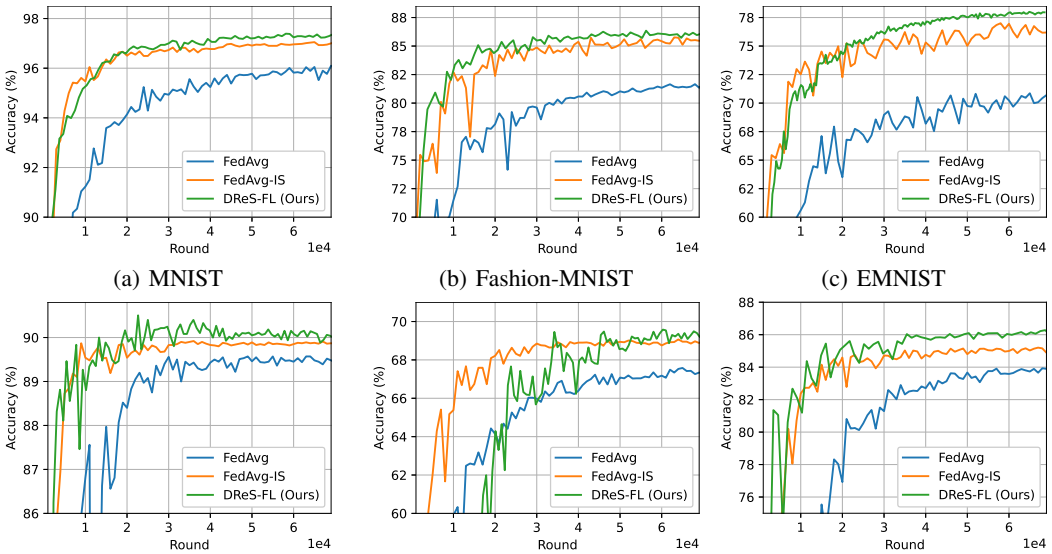
6.1 Experimental Setup

Dataset. We evaluate our proposed algorithm on several benchmark datasets: MNIST [45], Fashion-MNIST [46], EMNIST (Balanced) [47], CIFAR-10 [48], CIFAR-100 [48], and SVHN [49]. Specifically, the extra training samples in the SVHN dataset are not utilized. To simulate the non-IID data distribution, we assume there are $N = 20$ clients in the learning system and adopt the skewed label partition [50] to shuffle the datasets. Specifically, we sort a dataset by the labels, divide it into N shards, and assign one shard to each client. To simulate the client dropouts in the training process, we consider an extreme scenario, where the dropout rate of each client is set to 0.99 with a probability of 0.5 or is uniformly sampled from $[0, 0.1]$ otherwise. More details of the datasets are deferred to Appendix B.

Model structures. We adopt a multi-layer perception (MLP) with two hidden layers for the image classification tasks on MNIST, Fashion-MNIST, and EMNIST datasets. Each hidden layer contains 64 neurons. For CIFAR-10, CIFAR-100, and SVHN datasets, we resize the input images from 32×32 to 224×224 and adopt the convolutional layers of a pretrained VGG model to extract 25088-dimensional features. To classify the extracted features, we select a two-layer MLP model with 4096 hidden units each. The baseline methods train the neural networks on the real field and select the rectified linear unit (ReLU) function as the activation function. In each communication round, clients perform one SGD step for the local model update.

DReS-FL. Our method adopts the same size PINNs to replace MLPs in the federated training, and the degree of gradient is $\deg(\boldsymbol{g}) = 8$. Particularly, the extracted features from the last convolutional layer of VGG19 are secretly shared with other clients. We set the parameters $K = 1$ and $T = 1$ in the Lagrange coding, and the minimum number of clients needed to decode the global gradient is 9.

Baselines. In the experiments, data-centric approaches [11, 15, 17, 33, 34, 35] are not compared since some of them [11, 15, 17] lack strong privacy guarantees while others [33, 34, 35] cannot support federated neural network training with multiple clients. We select algorithm-based methods as baselines, including FedAvg [1], FedAvg with importance sampling (FedAvg-IS) [51, 52], and



(a) MNIST (b) Fashion-MNIST (c) EMNIST

(d) CIFAR-10 (e) CIFAR-100 (f) SVHN

Figure 3: Test accuracies on different datasets.

SCAFFOLD [2], since these methods can be easily combined with secure aggregation methods[8]. Particularly, we assume that the FedAvg-IS method knows the dropout distribution, and the local computation results are weighted by the participation probability (i.e., 1 - dropout probability) to mitigate bias in aggregation. Besides, we also select the centralized training scheme as a performance upper bound, where the server can access all the clients' datasets for model training.

6.2 Performance Evaluation

The experimental results are shown in Table 2 and Fig. 3. The FedAvg method achieves worse performance than the centralized training scheme. This is attributed to the non-IID data and client dropouts. The FedAvg-IS method improves the test accuracy compared with FedAvg, but there is still a noticeable performance gap with the centralized training scheme. It shows that using the knowledge of dropout distribution can partially compensate for the biases in the aggregated models, but the local data distributions are still heterogeneous and degrade the performance. Besides, SCAFFOLD has a low accuracy on all the settings. As the frequency of updating local control variates is low, the estimation of the update direction is highly inaccurate such that the model does not converge as shown in Fig. 4. These results are consistent with the findings in [10]. Our DReS-FL method is superior to all the baseline methods as the server can obtain global gradients after polynomial interpolation. In addition, DReS-FL achieves comparable performance to the centralized training scheme on some datasets, which demonstrates the effectiveness of our proposed framework in solving the non-IID and dropout problems.

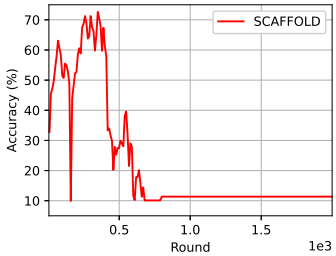


Figure 4: Test accuracy of SCAFFOLD on MNIST.

7 Conclusions

This paper proposed a Dropout-Resilient Secure Federated Learning (DReS-FL) framework via Lagrange coded computing (LCC) to simultaneously solve the data heterogeneity and dropout problems of FL, while providing privacy guarantees for the local datasets. The polynomial integer neural networks (PINNs) have been constructed to ensure that the server can correctly decode the

[8]Note that the secure aggregation mechanism has not been applied in the experiments, since the quantization step in secure aggregation may degrade the performance of the baselines.

global gradient without privacy leakage. Extensive experimental results validated the effectiveness of the proposed method. Potential limitations of our method include that the degree of the gradient in a PINN increases exponentially with the number of layers, which hinders training a deep model for complex tasks. Besides, performing multiple local SGD steps largely increases the finite field size as the range of results grows exponentially with the number of multiplications, and thus it will lead to substantial communication overhead in model transmission. Despite some limitations, we believe DReS-FL is a promising framework for many practical FL application scenarios given its effectiveness in resolving both the non-IID and client dropout problems, while with strong privacy guarantees.

References

[1] McMahan, B., E. Moore, D. Ramage, et al. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Karimireddy, S. P., S. Kale, M. Mohri, et al. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[3] Kairouz, P., H. B. McMahan, B. Avent, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[4] Hsieh, K., A. Phanishayee, O. Mutlu, et al. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[5] Luo, M., F. Chen, D. Hu, et al. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34, 2021.

[6] Sahu, A. K., T. Li, M. Sanjabi, et al. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 3:3, 2018.

[7] Li, Q., B. He, D. Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722. 2021.

[8] Acar, D. A. E., Y. Zhao, R. M. Navarro, et al. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

[9] Hsu, T.-M. H., H. Qi, M. Brown. Federated visual classification with real-world data distribution. In *European Conference on Computer Vision*, pages 76–92. Springer, 2020.

[10] Li, Q., Y. Diao, Q. Chen, et al. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021.

[11] Zhao, Y., M. Li, L. Lai, et al. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[12] Yoshida, N., T. Nishio, M. Morikura, et al. Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2020.

[13] Yoon, T., S. Shin, S. J. Hwang, et al. Fedmix: Approximation of mixup under mean augmented federated learning. In *International Conference on Learning Representations*. 2020.

[14] Sun, Y., J. Shao, S. Li, et al. Stochastic coded federated learning with convergence and privacy guarantees. *arXiv preprint arXiv:2201.10092*, 2022.

[15] Jeong, E., S. Oh, H. Kim, et al. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[16] Hao, W., M. El-Khamy, J. Lee, et al. Towards fair federated learning with zero-shot data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3310–3319. 2021.

[17] Zhang, L., B. Shen, A. Barnawi, et al. Feddpgan: federated differentially private generative adversarial networks framework for the detection of covid-19 pneumonia. *Information Systems Frontiers*, 23(6):1403–1415, 2021.

[18] Nguyen, D. C., M. Ding, P. N. Pathirana, et al. Federated learning for covid-19 detection with generative adversarial networks in edge cloud computing. *IEEE Internet of Things Journal*, 2021.

[19] Jeong, E., S. Oh, H. Kim, et al. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[20] Zhu, Z., J. Hong, J. Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pages 12878–12889. PMLR, 2021.

[21] Li, Z., J. Shao, Y. Mao, et al. Federated learning with GAN-based data synthesis for non-IID clients, 2022.

[22] Yu, Q., S. Li, N. Raviv, et al. Lagrange coded computing: Optimal design for resiliency, security, and privacy. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1215–1225. PMLR, 2019.

[23] Shamir, A. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

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

[25] Gilad-Bachrach, R., N. Dowlin, K. Laine, et al. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.

[26] Hesamifard, E., H. Takabi, M. Ghasemi. Cryptodl: towards deep learning over encrypted data. In *Annual Computer Security Applications Conference (ACSAC 2016), Los Angeles, California, USA*, vol. 11. 2016.

[27] Graepel, T., K. Lauter, M. Naehrig. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pages 1–21. Springer, 2012.

[28] Yuan, J., S. Yu. Privacy preserving back-propagation neural network learning made practical with cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):212–221, 2013.

[29] Han, K., S. Hong, J. H. Cheon, et al. Logistic regression on homomorphic encrypted data at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pages 9466–9471. 2019.

[30] Wang, Q., M. Du, X. Chen, et al. Privacy-preserving collaborative model learning: The case of word vector training. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2381–2393, 2018.

[31] Nikolaenko, V., U. Weinsberg, S. Ioannidis, et al. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE symposium on security and privacy*, pages 334–348. IEEE, 2013.

[32] Gascón, A., P. Schoppmann, B. Balle, et al. Privacy-preserving distributed linear regression on high-dimensional data. *Proc. Priv. Enhancing Technol.*, 2017(4):345–364, 2017.

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

[34] So, J., B. Güler, A. S. Avestimehr. Codedprivateml: A fast and privacy-preserving framework for distributed machine learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):441–451, 2021.

[35] So, J., B. Guler, S. Avestimehr. A scalable approach for privacy-preserving collaborative machine learning. *Advances in Neural Information Processing Systems*, 33:8054–8066, 2020.

[36] Shokri, R., M. Stronati, C. Song, et al. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[37] Nasr, M., R. Shokri, A. Houmansadr. Comprehensive privacy analysis of deep learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–15. 2018.

[38] Geiping, J., H. Bauermeister, H. Dröge, et al. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[39] Bonawitz, K. and Ivanov, Vladimir and Kreuter, Ben and Marcedone, Antonio and McMahan, H Brendan and Patel, Sarvar and Ramage, Daniel and Segal, Aaron and Seth, Karn. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. 2017.

[40] Kadhe, S., N. Rajaraman, O. O. Koyluoglu, et al. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2009.11248*, 2020.

[41] Yang, C.-S., J. So, C. He, et al. Lightsecagg: Rethinking secure aggregation in federated learning. *arXiv preprint arXiv:2109.14236*, 2021.

[42] Jahani-Nezhad, T., M. A. Maddah-Ali, S. Li, et al. Swiftagg+: Achieving asymptotically optimal communication load in secure aggregation for federated learning. *arXiv preprint arXiv:2203.13060*, 2022.

[43] So, J., B. Güler, A. S. Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):479–489, 2021.

[44] Bell, J. H., K. A. Bonawitz, A. Gascón, et al. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269. 2020.

[45] LeCun, Y., L. Bottou, Y. Bengio, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[46] Xiao, H., K. Rasul, R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[47] Cohen, G., S. Afshar, J. Tapson, et al. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[48] Krizhevsky, A., G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[49] Netzer, Y., T. Wang, A. Coates, et al. Reading digits in natural images with unsupervised feature learning. 2011.

[50] Hsieh, K., A. Phanishayee, O. Mutlu, et al. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[51] Ren, J., Y. He, D. Wen, et al. Scheduling for cellular federated edge learning with importance and channel awareness. *IEEE Transactions on Wireless Communications*, 19(11):7690–7703, 2020.

[52] Kairouz, P., H. B. McMahan, B. Avent, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[53] Bonawitz, K., V. Ivanov, B. Kreuter, et al. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] The proofs are provided in Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code is proprietary, but the public datasets used in the experiments and the instructions needed for reproduction are presented in Section 6 and Appendix B.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please refer to Section 6 and Appendix B.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 6.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [Yes] See Appendix B.

(c) Did you include any new assets either in the supplemental material or as a URL? [No] The code is proprietary and the datasets used in our work are public datasets.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We choose the commonly-used datasets in the experiments.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Proof of Theorem 2

For simplicity, we denote $\boldsymbol{g}^{(t)} = \frac{1}{bK}\sum_{k=1}^{K} \boldsymbol{g}(\overline{\mathbf{X}}_{\mathcal{I}_t}^{(k)}, \overline{\mathbf{Y}}_{\mathcal{I}_t}^{(k)}; \mathbf{w}^{(t)})$ in this section. The server updates the global model by $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - Q(\eta \boldsymbol{g}^{(t)})$ in each round t after receiving $\deg(\boldsymbol{g})(K+T-1)+1$ uploads from clients. We first provide an important lemma to show that the model update $Q(\eta \boldsymbol{g}^{(t)})$ on the server is an unbiased estimate of $\eta \boldsymbol{g}_e(\mathbf{w}^{(t)})$.

Lemma 1. *(Unbiased and variance-bounded model update) In the t-th round, the model update $Q(\eta \boldsymbol{g}^{(t)})$ has the following properties:*

$$\mathbb{E}\left[Q(\eta \boldsymbol{g}^{(t)})\right] = \eta \boldsymbol{g}_e(\mathbf{w}), \tag{9}$$

$$\mathbb{E}\left[\|Q(\eta \boldsymbol{g}^{(t)}) - \eta \boldsymbol{g}_e(\mathbf{w})\|^2\right] \leq (\gamma^2+1)\eta^2 \frac{\sigma^2}{bK} + \gamma^2\eta^2 \left\|\boldsymbol{g}_e(\mathbf{w}^{(t)})\right\|^2. \tag{10}$$

Proof. According to Assumption 2-3, we directly obtain that $\mathbb{E}\left[Q(\eta \boldsymbol{g}^{(t)})\right] = \mathbb{E}\left[\eta \boldsymbol{g}^{(t)}\right] = \boldsymbol{g}_e(\mathbf{w})$. Since the batch sampling and rounding operation cause independent errors, the variance is upper bounded as follows:

$$\begin{aligned}
&\mathbb{E}\left[\left\|Q(\eta \boldsymbol{g}^{(t)}) - \eta \boldsymbol{g}_e(\mathbf{w}^{(t)})\right\|^2\right] \\
&= \mathbb{E}\left[\left\|Q(\eta \boldsymbol{g}^{(t)}) - \eta \boldsymbol{g}^{(t)}\right\|^2\right] + \mathbb{E}\left[\left\|\eta \boldsymbol{g}^{(t)} - \eta \boldsymbol{g}_e(\mathbf{w}^{(t)})\right\|^2\right] \\
&\overset{(a)}{\leq} \gamma^2 \mathbb{E}\left[\left\|\eta \boldsymbol{g}^{(t)}\right\|^2\right] + \mathbb{E}\left[\left\|\eta \boldsymbol{g}^{(t)} - \eta \boldsymbol{g}_e(\mathbf{w}^{(t)})\right\|^2\right] \\
&= \gamma^2\eta^2\left[\mathbb{E}\left[\left\|\boldsymbol{g}^{(t)} - \boldsymbol{g}_e(\mathbf{w}^{(t)})\right\|^2\right] + \left\|\boldsymbol{g}_e(\mathbf{w}^{(t)})\right\|^2\right] + \eta^2 \mathbb{E}\left[\left\|\boldsymbol{g}^{(t)} - \boldsymbol{g}_e(\mathbf{w}^{(t)})\right\|^2\right] \\
&\overset{(b)}{\leq} (\gamma^2+1)\eta^2 \frac{\sigma^2}{bK} + \gamma^2\eta^2 \left\|\boldsymbol{g}_e(\mathbf{w}^{(t)})\right\|^2,
\end{aligned}$$

where (a) follows Assumption 3. (b) is due to Assumption 2 and the independence of mini-batch sampling noises among clients. □

With Lemma 1, we prove Theorem 2 as follows:

Proof. The model update in the t-th iteration can be expressed as $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - Q(\eta \boldsymbol{g}^{(t)})$. According to the Taylor's expansion, we have:

$$\begin{aligned}
&\mathbb{E}\left[\ell(\mathbf{w}^{(t+1)})\right] - \mathbb{E}\left[\ell(\mathbf{w}^{(t)})\right] \\
&\leq -\mathbb{E}\left\langle \boldsymbol{g}_e(\mathbf{w}^{(t)}), Q(\eta \boldsymbol{g}^{(t)})\right\rangle + \frac{L}{2}\mathbb{E}\left[\left\|Q(\eta \boldsymbol{g}^{(t)})\right\|^2\right] \\
&\overset{(c)}{=} -\mathbb{E}\left\langle \boldsymbol{g}_e(\mathbf{w}^{(t)}), \eta \boldsymbol{g}_e(\mathbf{w}^{(t)})\right\rangle + \frac{L}{2}\mathbb{E}\left[\left\|Q(\eta \boldsymbol{g}^{(t)})\right\|^2\right] \\
&\overset{(d)}{=} -\eta\mathbb{E}\left[\|\boldsymbol{g}_e(\mathbf{w}^{(t)})\|^2\right] + \frac{L}{2}\mathbb{E}\left[\left\|Q(\eta \boldsymbol{g}^{(t)}) - \eta \boldsymbol{g}_e(\mathbf{w}^{(t)})\right\|^2\right] + \frac{L}{2}\mathbb{E}\left[\left\|\eta \boldsymbol{g}_e(\mathbf{w}^{(t)})\right\|^2\right] \\
&\overset{(e)}{\leq} -\left(\eta - \frac{\eta^2 L}{2}\right)\mathbb{E}\left[\|\boldsymbol{g}_e(\mathbf{w}^{(t)})\|^2\right] + \frac{L}{2}\left((\gamma^2+1)\eta^2\frac{\sigma^2}{bK} + \gamma^2\eta^2\mathbb{E}\left[\|\boldsymbol{g}_e(\mathbf{w}^{(t)})\|^2\right]\right) \\
&= -\left(\eta - \frac{\eta^2 L}{2} - \frac{\eta^2\gamma^2 L}{2}\right)\mathbb{E}\left[\|\boldsymbol{g}_e(\mathbf{w}^{(t)})\|^2\right] + \frac{\eta^2 L\sigma^2}{2bK}(\gamma^2+1),
\end{aligned}$$

where (c) follows Lemma 1, (d) holds according to the fact that $\mathbb{E}\left\langle \nabla Q(\eta \boldsymbol{g}^{(t)}) - \eta \boldsymbol{g}_e(\mathbf{w}^{(t)}), \boldsymbol{g}_e(\mathbf{w}^{(t)})\right\rangle = 0$, and (e) is due to Assumption 2-3. If it holds

Table 3: Details of the datasets

	MNIST	Fashon-MNIST	EMNIST	CIFAR-10	CIFAR-100	SVHN
No. of classes	10	10	47	10	100	10
No. of training samples	60,000	60,000	112,800	50,000	50,000	73,257
No. of test samples	10,000	10,000	18,800	10,000	10,000	26,032
Image size	28×28	28×28	28×28	32×32	32×32	32×32
License	Creative Commons Attribution-Share Alike 3.0 License	MIT License	Apache License 2.0	MIT License	MIT License	CC0:Public Domain License

Table 4: Hyperparameters for our DReS-FL method

Parameters	MNIST	Fashion-MNIST	EMNIST	CIFAR-10	CIFAR-100	SVHN
Maximum L2-norm for gradient clipping	2×10^4	2×10^4	5×10^6	2×10^4	1×10^9	2×10^4
Prime number p	$2^{200} - 75$	$2^{200} - 75$	$2^{440} - 33$	$2^{440} - 33$	$2^{440} - 33$	$2^{440} - 33$
Parameter l in data transformation	4	4	4	2	2	2

that $\eta - \frac{\eta^2 L}{2} - \frac{\eta^2 \gamma^2 L}{2} > 0$, we summarize the above inequality over $t = 1, 2, \ldots, \tau'$ to conclude the proof. □

B Additional Experimental Details

All experiments are performed by Pytorch on an Intel Xeon Gold 6246R CPU @ 3.40 GHz and a Geforce RTX 3090. Some details of the datasets are summarized in Table 3. We adopt mini-batch SGD with a batch size of 64 to optimize the models in federated training. The communication round is set to be 7×10^4, and the clients perform one local SGD step in each round. The learning rate is initialized as 0.1, and it will decay with a factor of 0.65 after every 1500 rounds. Other parameters in our DRes-FL framework are summarized in Table 4.

C Complexity Analysis and Comparison

In this part, we analyze the communication and computational complexities of the proposed DReS-FL framework with respect to the parameters $(N, T, K, \tau, d_w, b_g)$. Parameter N is the number of clients, and T denotes the privacy threshold in Lagrange coding [22]. Parameter K denotes the number of shards in the local datasets. A large value of K reduces the communication and computation overheads in the proposed DReS-FL framework. In the federated training, the parameter τ corresponds to the number of communication rounds. Parameters d_w and b_g denote the model size and the global batch size, respectively. Before training starts, each client's computation cost for Lagrange coding and communication complexity for data sharing are $\mathcal{O}(N \log^2(K+T) \log\log(K+T))$ and $\mathcal{O}(N/K)$, respectively. In each round of federated training, the local computation complexity is $\mathcal{O}(d_w b_g / K)$, and the model uploading cost is $\mathcal{O}(d_w)$. Besides, the communication overhead of the server for model distributing is $\mathcal{O}(N d_w)$, and the model decoding complexity by polynomial interpolation is $\mathcal{O}(R \log^2 R \log\log R d_w)$, where R denotes the minimum uploads needed for gradient decoding.

Different from our method, secure aggregation approaches [39, 53, 40, 42, 44, 41] generate random masks to protect the local model parameters. In each round, clients first share coded masks with each other, which allows for aggregating the masked models at the server. As some clients may drop out of the training process unexpectedly, the surviving clients upload the shared information belonging to the dropped clients to reconstruct the aggregated model. The main drawback of such approaches is that the clients need to generate new masks in each round, and their computational and communication complexities increase linearly with the number of training rounds. In comparison, the data sharing phase of our method only introduces extra costs for one time, which is independent of the training rounds. In the scenario that the number of training rounds is very large, the proposed DReS-FL method achieves lower computational and communication costs than the secure aggregation protocols.

Table 5: Computational complexity comparison

	Preparation	Iterative training (τ rounds)		
	Lagrangian coding	Generating coded random masks	Local model update	Global model aggregation
FedAvg	—	—	$\mathcal{O}\left(\tau d_w b_g / N\right)$	$\mathcal{O}(\tau N d_w)$
FedAvg with LightSecAgg	—	$\mathcal{O}\left(\frac{\tau d_w N^2 \log N}{R-T}\right)$	$\mathcal{O}\left(\tau d_w b_g / N\right)$	$\mathcal{O}\left(\frac{\tau d_w R \log R}{R-T}\right)$
DReS-FL	$\mathcal{O}(N^2 \log^2(K+T) \log\log(K+T))$	—	$\mathcal{O}\left(\tau d_w b_g / K\right)$	$\mathcal{O}(\tau d_w R \log^2 R \log\log R)$

Table 6: Communication complexity comparison

	Preparation	Iterative training (τ rounds)			
	Data sharing	Coded masks sharing among clients	Local model uploading	Coded masks uploading	Global model downloading
FedAvg	—	—	$\mathcal{O}(\tau d_w)$	—	$\mathcal{O}(\tau N d_w)$
FedAvg with LightSecAgg	—	$\mathcal{O}\left(\frac{\tau N^2 d_w}{R-T}\right)$	$\mathcal{O}(\tau d_w)$	$\mathcal{O}\left(\frac{\tau d_w R}{R-T}\right)$	$\mathcal{O}(\tau N d_w)$
DReS-FL	$\mathcal{O}(N^2/K)$	—	$\mathcal{O}(\tau d_w)$	—	$\mathcal{O}(\tau N d_w)$

The detailed comparisons among FedAvg, FedAvg with LighSecAgg [41], and our DReS-FL method are summarized in Table 5 and 6.

D Model Extension

Our DReS-FL framework can be extended to more general cases in which clients can run s ($s \geq 1$) local SGD steps each round. Denote the computation results after s local SGD steps in round t as $\Delta\widetilde{\mathbf{w}}_j(s;\mathbf{w}^{(t)})$ for $j \in [N]$. Specifically, $\Delta\widetilde{\mathbf{w}}_j(s=1;\mathbf{w}^{(t)}) \triangleq \widetilde{\boldsymbol{g}}(\widetilde{\mathbf{X}}_j^{(\mathcal{I}_t)}, \widetilde{\mathbf{Y}}_j^{(\mathcal{I}_t)}; \mathbf{w}^{(t)})$ and $\Delta\widetilde{\mathbf{w}}_j(s=2;\mathbf{w}^{(t)}) \triangleq \widetilde{\boldsymbol{g}}(\widetilde{\mathbf{X}}_j^{(\mathcal{I}_{t+1})}, \widetilde{\mathbf{Y}}_j^{(\mathcal{I}_{t+1})}; \mathbf{w}^{(t)} - \frac{\eta}{bK}\widetilde{\boldsymbol{g}}(\widetilde{\mathbf{X}}_j^{(\mathcal{I}_t)}, \widetilde{\mathbf{Y}}_j^{(\mathcal{I}_t)}; \mathbf{w}^{(t)}))$. By carefully selecting the learning rate η such that $\frac{\eta}{bK} \in \mathbb{F}_p$, the function $\Delta\widetilde{\mathbf{w}}_j(s;\mathbf{w}^{(t)})$ is still a polynomial in the finite field $\mathbb{F}_p$. Therefore, the central server can recover the desired model update by polynomial interpolation at the cost of low dropout-resiliency caused by the high degree of $\Delta\widetilde{\mathbf{w}}_j(s;\mathbf{w}^{(t)})$.

E Degree of Gradient in PINN

Given a data sample as $(\boldsymbol{x}, \boldsymbol{y})$, the feedforward process of a PINN with L quadratic activation layers $\boldsymbol{h}$ is as follows:

$$\boldsymbol{z}_0' \to \boldsymbol{z}_1 \to \boldsymbol{z}_1' \to \boldsymbol{z}_2 \to \boldsymbol{z}_2' \to \cdots \to \boldsymbol{z}_L \to \boldsymbol{z}_L' \to \boldsymbol{z}_{L+1},$$

where $\boldsymbol{z}_0' \triangleq \boldsymbol{x}$, $\boldsymbol{z}_l' = \boldsymbol{h}(\boldsymbol{z}_l)$, and $\boldsymbol{z}_l = \mathbf{W}_l \boldsymbol{z}_{l-1}' + \boldsymbol{b}_l$ for $l \in 1:L+1$. $\boldsymbol{z}_{L+1}$ is the output of PINN, and the loss function is the squared error between $\boldsymbol{y}$ and $\boldsymbol{z}_{L+1}$, i.e., $\ell = \|\boldsymbol{y} - \boldsymbol{z}_{L+1}\|_2^2$. The notations $\mathbf{W}_l$ and $\boldsymbol{b}_l$ correspond to the weight matrix and bias vector in PINN. With the above preparation, we derive the gradients as follows:

$$\frac{\partial \ell}{\partial \boldsymbol{z}_{L+1}} = 2(\boldsymbol{z}_{L+1} - \boldsymbol{y})^T, \ \frac{\partial \boldsymbol{z}_l'}{\partial \boldsymbol{z}_l} = 2\mathrm{diag}(\boldsymbol{z}_l), \ \frac{\partial \boldsymbol{z}_l}{\partial \mathbf{W}_l[j]} = \mathbf{I}\boldsymbol{z}_{l-1}'[j], \ \frac{\partial \boldsymbol{z}_l}{\partial \boldsymbol{z}_{l-1}} = \mathbf{W}_l,$$

where $\mathbf{W}_l[j]$ is the j-th column in $\mathbf{W}_l$, and $\boldsymbol{z}_{l-1}'[j]$ is the j-th element in the vector $\boldsymbol{z}_{l-1}'$. According to the chain rule of gradient, the gradient of the loss function with respect to the weight $\mathbf{W}_l[j]$ is

$$\begin{aligned}\frac{\partial \ell}{\partial \mathbf{W}_l[j]} =& \frac{\partial \ell}{\partial \boldsymbol{z}_{L+1}} \frac{\partial \boldsymbol{z}_{L+1}}{\partial \boldsymbol{z}_L'} \frac{\partial \boldsymbol{z}_L'}{\partial \boldsymbol{z}_L} \cdots \frac{\partial \boldsymbol{z}_{l+1}}{\partial \boldsymbol{z}_l'} \frac{\partial \boldsymbol{z}_l'}{\partial \boldsymbol{z}_l} \frac{\partial \boldsymbol{z}_l}{\partial \mathbf{W}_l[j]}, \\ =& 2(\boldsymbol{z}_{L+1} - \boldsymbol{y})^T \mathbf{W}_{L+1} 2\mathrm{diag}(\boldsymbol{z}_L) \cdots \mathbf{W}_{l+1} 2\mathrm{diag}(\boldsymbol{z}_l) \boldsymbol{z}_{l-1}'[j].\end{aligned}$$

Note that the mappings from input $\boldsymbol{z}_0'$ to $\boldsymbol{z}_l'$ and $\boldsymbol{z}_{l+1}$ are polynomials with degree 2^l. Therefore, the degree of the gradient $\frac{\partial \ell}{\partial \mathbf{W}_l[j]}$ is

$$\begin{aligned}\deg\left(\frac{\partial \ell}{\partial \mathbf{W}_l[j]}\right) &= 2^L + 2^{L-1} + \cdots + 2^{l-1} + 2^{l-1}, \\ &= 2^{L+1}.\end{aligned}$$