Squeezing More Utility via Adaptive Clipping on Differentially Private Gradients in Federated Meta-Learning

Ning Wang
Virginia Tech
Blacksburg, VA, USA
ning18@vt.edu

Yang Xiao
University of Kentucky
Lexington, KY, USA
xiaoy@uky.edu

Yimin Chen
University of Massachusetts Lowell
Lowell, MA, USA
ian_chen@uml.edu

Ning Zhang
Washington University in St. Louis
St. Louis, MO, USA
zhang.ning@wustl.edu

Wenjing Lou
Virginia Tech
Blacksburg, VA, USA
wjlou@vt.edu

Y. Thomas Hou
Virginia Tech
Blacksburg, VA, USA
thou@vt.edu

ABSTRACT

Federated meta-learning has emerged as a promising AI framework for today's mobile computing scenes involving distributed clients. It enables collaborative model training using the data located at distributed mobile clients and accommodates clients that need fast model customization with limited new data. However, federated meta-learning solutions are susceptible to inference-based privacy attacks since the global model encoded with clients' training data is open to all clients and the central server. Meanwhile, differential privacy (DP) has been widely used as a countermeasure against privacy inference attacks in federated learning. The adoption of DP in federated meta-learning is complicated by the model accuracy-privacy trade-off and the model hierarchy attributed to the meta-learning component. In this paper, we introduce DP-FedMeta, a new differentially private federated meta-learning architecture that addresses such data privacy challenges. DP-FedMeta features an adaptive gradient clipping method and a one-pass meta-training process to improve the model utility-privacy trade-off. At the core of DP-FedMeta are two DP mechanisms, namely DP-AGR and DP-AGRLR, to provide two notions of privacy protection for the hierarchical models. Extensive experiments in an emulated federated meta-learning scenario on well-known datasets (Omniglot, CIFAR-FS, and Mini-ImageNet) demonstrate that DP-FedMeta accomplishes better privacy protection while maintaining comparable model accuracy compared to the state-of-the-art solution that directly applies DP-based meta-learning to the federated setting.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Distributed artificial intelligence**; **Machine learning**;

ACSAC '22, December 5–9, 2022, Austin, TX, USA

ACM ISBN 978-1-4503-9759-9/22/12.
https://doi.org/10.1145/3564625.3564652

KEYWORDS

differential privacy, federated meta-learning, adaptive clipping, privacy utility trade-off

ACM Reference Format:
Ning Wang, Yang Xiao, Yimin Chen, Ning Zhang, Wenjing Lou, and Y. Thomas Hou. 2022. Squeezing More Utility via Adaptive Clipping on Differentially Private Gradients in Federated Meta-Learning. In *Annual Computer Security Applications Conference (ACSAC '22), December 5–9, 2022, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3564625.3564652

1 INTRODUCTION

Deep learning (DL) has enjoyed great success in many sectors of society, mainly due to the big leap in computing hardware capability and availability of massive amount of data. However, the increased concentration of data needed by DL has raised wide-spread concerns over data privacy, leading to data privacy regulations such as General Data Protection Regulation (GDPR[1]) in European Union and California Consumer Privacy Act (CCPA[2]) in the US. In many cases, it is often impossible to move the data to a central location due to legal restrictions, such as those imposed by Health Insurance Portability and Accountability Act (HIPAA[3]). Therefore, more and more data are now stored distributively at edge nodes or end devices close to their sources rather than at a central location.

Federated learning (FL) [12, 15, 17] has emerged as a new paradigm to enable collaborative training over distributed private data. In FL, participants jointly train a global model without sharing their private data. FL outputs a common model for all the users and does not customize the model to each user. This is an important missing feature for practical deployment in distribution learning scenarios, especially given the heterogeneity of the underlying data distribution and learning task for various users. In light of this gap, federated meta-learning [5, 8, 16, 31] has emerged as one powerful AI framework for enabling fast model adaptation amid collaborative training, with prime use cases in IoT and mobile computing scenarios. The meta-learning component [9, 20] enables a cloud server to extract common knowledge from distributed data owners with

[1] https://gdpr.eu/
[2] https://oag.ca.gov/privacy/ccpa
[3] https://www.hhs.gov/hipaa/index.html

different training data and local tasks. The common knowledge, in the form of global meta model, can be quickly customized to a new client (called "model consumer") with a few new data samples.

Despite its prospect of making the best of two worlds, the federated meta-learning framework is still prone to inference-based privacy attacks on individual clients' data, including the membership inference attack [19, 32] and model inversion [28]. An adversary can recover the private training data [27, 28] or sensitive partial information [19] by leveraging a specific inference model. Meanwhile, prior wisdom alludes that *Differential privacy (DP)* [7] can be used to provide rigorous privacy guarantee to FL algorithms by adding noise to gradients update in a controlled manner [10, 18, 29]. Privacy protection faces complications in the federated meta-learning framework compared to a traditional ML. On the one hand, the adoption of DP into FL may lead to a significant decrease in model utility. Taking the popular FedAvg [17] algorithm as an example, naively adopting DP into FedAvg may cause 2-10 times training loss than the original model [30]. On the other hand, the privacy notion is further complicated due to the hierarchical nature of federated meta-learning frameworks. There are two types of models across system participants: base models at clients and the meta-model at the central server. It is unclear which gradients contain what level of privacy and are exposed to whom. [14] directly applies DP to the federated meta-learning framework and thus results in a large accuracy sacrifice. We aim to improve the model accuracy while satisfying a minimum privacy protection requirement.

In this paper, we address the above privacy leakage problem by substantiating a new differentially private federated meta-learning architecture, namely DP-FedMeta. Catering to the hierarchical learning framework of federated meta-learning, DP-FedMeta features two DP mechanisms, namely DP-AGR (**AG**regation **R**ule) and DP-AGRLR (**AG**regation **R**ule with **L**ocal **R**andomness), for achieving two practical privacy protection notions for clients' training data against the curious central server and curious local clients, respectively. DP mechanisms in ML generally entail bounding each user's model update contribution by clipping its gradient to some constant value. However, in meta-learning tasks, it can be hard to obtain a priori knowledge of the clipping threshold across tasks and learning settings. To achieve a better trade-off between privacy and accuracy, building on the intuition that the current gradient norm is predictive with the knowledge of data geometry in earlier iterations, we propose an adaptive clipping mechanism to dynamically achieve a minimized clipping threshold while preserving most of the gradient information. We emphasize that naive adaption of adaptive clipping for federated meta-learning, such as determining the clipping threshold based on the current batch of gradients, would leak clients' private information since such clipping threshold is computed with the true gradients. To avoid further privacy leakage in adjusting the clipping threshold, we utilize the historical differentially private aggregated gradients (i.e., product of differentially private gradient aggregation of previous training set) instead of the true online gradients. As a result, our adaptive clipping mechanism retains the same level of privacy protection while boosting the system's overall accuracy. Putting the above designs together, DP-FedMeta attains a significantly lower privacy budget and higher model accuracy simultaneously compared to the state-of-the-art DP solution for federated meta-learning [14] which is a direct application of DP-based meta-learning to the federated setting.

The major contributions of this paper are summarized as follows:

- We introduce a differentially private federated meta-learning architecture, dubbed DP-FedMeta, to enable collaborative model training by heterogeneous users with fast model adaptation and rigorous privacy guarantee. Catering to two practical trust levels on the central server, DP-FedMeta features two variants of DP mechanisms, DP-AGR and DP-AGRLR, respectively.
- For both DP-AGR and DP-AGRLR, we propose a novel adaptive gradient clipping method that maximally preserves model accuracy while guaranteeing a fixed level of privacy. This method feeds on past differentially private gradients (instead of the sensitive original gradients) which essentially helps achieve a significantly lower privacy budget ($\epsilon = 1.5$ or 2.5) than the state-of-the-art DP federated meta-learning work ($\epsilon = 9.5$) [14] while maintaining reasonable model accuracy.
- We analyze the impact of client/task sampling rate on accumulated privacy loss for the whole training process of DP-FedMeta. We derive an upper bound of the final accumulated privacy loss given sampling rate and noise level (Section 3.5). An important insight is that privacy loss depends on the sampling rate, rather than the number of training tasks sampled.
- We thoroughly evaluate DP-FedMeta's performance on three well-known datasets, i.e., Omniglot, CIFAR-FS, and mini-ImageNet. We evaluate the impact of various factors including gradient clipping, noise multiplier, and user sample size on model accuracy and privacy for both DP-AGR and DP-AGRLR methods. Under a small $(1.5, 10^{-6})$-DP budget, DP-AGR achieves as high as 96.8% accuracy and DP-AGRLR achieves 89.7% accuracy (the accuracy of [14] is 75%) for 5-way 5-shot learning on Omniglot.

2 DP-FEDMETA OVERVIEW

2.1 System Architecture and Vision

The vision of DP-FedMeta is to enable distributed mobile clients to perform federated meta-learning with rigorous privacy guarantee on client's data. Fig. 1 illustrates the basic workflow of the DP-FedMeta architecture. There are three types of participants in our system:

Mobile Clients contribute to the meta-training process with their private data. A mobile client first initializes its neural network θ with the current meta-model Θ. It then trains the base-model θ with its local data. Due to the energy constraint, a mobile client can only perform one to several steps of gradient descent. It then computes model gradients and submit them to the central server. In later sections mobile client is also referred to as client for convenience.

Central Server's primary role is processing and aggregating the model gradients from mobile clients. At the system onset, it randomly initializes the meta-model Θ. In each meta-training round, it first distributes Θ to a group of mobile clients, collects gradients from them, applies gradient clipping with a clipping threshold. The central server then sums over all clipped gradients added with Gaussian noise. The noisy gradient is used to update the meta-model. It outputs the final meta-model in the end of the meta-training round.

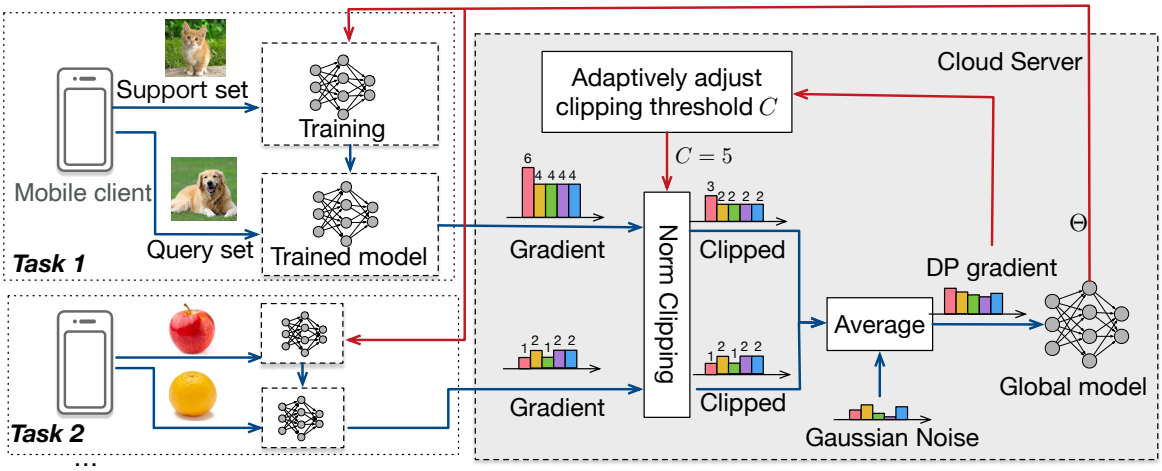


Figure 1: Proposed DP-FedMeta architecture with adaptive gradient clipping. Mobile clients compute gradients with local data and submit them to the cloud server. The multi-dimensional gradient is clipped by an adaptively adjusted clipping threshold C. Gaussian noise is added to the summation of all clipped gradient and the average is used to update the global meta-model Θ.

Model Consumers are some of the mobile clients who only consume meta-model from the central server without reporting local gradients to it.

We assume mobile clients have limited storage space and are energy-constrained. Therefore, there are only a small amount of data can be stored in a mobile device, and the mobile device does not support a computationally-intensive process, e.g., training a deep model from scratch independently. Clients participate in an FL system to cooperatively train a model initialization parameter. Each mobile client (including mobile consumers) has its own training task. We assume that the training tasks for different clients are not necessarily the same. One example of a task is to differentiate dogs from cats, and another task is to differentiate apples from pears. It can be a general case in practice since different clients may have different application scenarios. The overall goal of our system is not to train a model that works for only one task but to train a well-generalized model for various tasks. The traditional FL can not deal with this problem as traditional FL is for a scenario where clients' training tasks are the same. We propose to employ meta-learning to learn a parameter initialization and customize it to different tasks of clients. Meta-learning [4, 9, 26] learns common knowledge across a large number of tasks which is an optimized starting point for various new tasks as it can fast adapt to unseen tasks. We build DP-FedMeta based on a meta-learning algorithm (i.e., MAML[9]), and our design also applies to other meta-learning algorithms. Please refer to Section 3.2 for more detailed discussion on MAML.

2.2 Threat Model and Challenges

In FL, both the central server and clients can be curious about clients' privacy. We assume there are two levels of trust on the central server: *trusted* and *honest-but-curious*. A trusted central server strictly follows the aggregation procedure and is not inquisitive about clients' training data. An honest-but-curious central server also follows the aggregation procedure but may sniff updates from clients to reveal the information of their training data. In practice, some reputable organizations, such as publicly owned and government-backed institutions, can be assumed trusted. For other commercial entities, the central server is assumed honest-but-curious. Furthermore, we also assume clients (including model consumers) are honest-but-curious. Recent literature demonstrates that an adversary [28] who has access to the model and output label can reconstruct the training data. Further, an adversary [19, 32] can differentiate whether one data record is in the training dataset or not by accessing only the model. We apply DP to FL to protect individual clients' training data privacy.

We aim to maximize the model accuracy while satisfying a minimum privacy protection requirement. However, adopting DP into FL may lower model utility significantly, e.g., naively incorporating DP into FedAvg may cause 2-10 times training loss than the original model [30]. The hierarchy gradients of federated meta-learning bring further challenges at applying DP to the learning process. We will detail the challenges and our solutions at what follows.

2.3 Applying DP to Federated Learning Clients

The objective of DP is to enable the utilization of the information on a population while hiding individual's information. Mathematically, DP is defined as follows:

Definition 2.1 (Differential Privacy). A mechanism $\mathcal{M}: D \rightarrow R$ is (ϵ, δ)-differentially private if for any subset of outputs $S \subseteq R$ and for any two adjacent databases $d, d' \in D$, $\mathcal{M}$ satisfies that:

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(d') \in S] + \delta. \quad (1)$$

d and d' differ in at most a single entry, i.e., $\|d - d'\|_1 \leq 1$.

Considering $\delta \rightarrow 0$ and $\epsilon \rightarrow 0$, $\frac{\Pr[\mathcal{M}(d) \in S]}{\Pr[\mathcal{M}(d') \in S]} \leq e^{\epsilon} \approx 1$ indicates that one cannot distinguish between d and d' by observing $\mathcal{M}(d)$ and $\mathcal{M}(d')$.

When applying DP to an ML model, we need to add noise to gradients before updating the model in the training process. Gradient

clipping and noise adding are two critical procedures that impact the privacy-utility trade-off.

2.4 Why Adaptive Clipping?

The trade-off between privacy and utility is a significant challenge to a federated learning system. It is not the actual noise size, but the ratio of noise over the individual gradient impacts the privacy protection level. We call the ratio as noise multiplier denoted by z. A larger z will better conserve privacy but may significantly harm the model accuracy. A more negligible z helps maintain good model utility at each learning step while the privacy budget will be used up quickly. In this way, the utility of the final model may also be low since we may need to terminate the training process early because of depleting privacy budget. We need first select an appropriate z. The real noise level is also related to the gradient clipping threshold besides z. We figure out there is still a space to squeeze the privacy budget after z is selected. We can perform the gradient clipping more smartly. The reason why adaptive clipping is more favorable than constant clipping is explained in the following.

DP is designed to hide the existence of every individual data so that the noise should be z times the most significant gradient. Therefore, the largest gradient determines the final noise size. To maintain model utility, we should avoid large noise size. To this end, we clip the gradients using a pre-defined clipping threshold. A small clipping threshold will result in a small noise size but losing the original gradient's information. We should minimize the clipping threshold while preserving most of the gradient information. It is not trivial to decide the clipping threshold since the gradient norm varies across tasks and training process. In response, we propose an adaptive clipping mechanism to achieve this goal (see Sec. 3.1).

2.5 Two Levels of Privacy Protection

A critical challenge arises from the hierarchical architecture of the federated meta-learning framework, mainly attributed by the meta-learning component. There are two ML models across the participants: base models at clients and the meta-model at the central server, which consume different types of gradient updates. Based on threat model, we need to clearly define which gradients contain what level of privacy and are exposed to whom. Accordingly, we assume two trust levels on central server. For a trusted central server, only the meta-model Θ will be released to adversary. For an honest-but-curious central server, both Θ and base model θ will be exposed to adversary. To incorporate DP mechanisms, we define two levels of privacy protection for the two adversary models accordingly:

1) User-level DP: Releasing Θ will at no point compromise information regarding any specific task client.
2) Two-fold DP: Releasing Θ will at no point compromise either any specific data records nor any task client. Meanwhile, uploading θ_i will at no point reveal the existence of any specific data records used during training to the central server. Here "two-fold" refers to both user-level and record-level DP.

Two-fold DP is more strict than the user-level DP. For DP-FedMeta, both the user-level DP and Two-fold DP can preserve the client-level privacy from honest-but-curious clients or model consumers, meaning that the model consumers who receive the intermediate/final meta-model can not differentiate whether another user participates in the meta-training process or not. When faced with an honest-but-curious central server, two-fold DP additionally incorporating DP at the record-level. Record-level DP means that no one can reveal the information of an individual data record of a client by seeing the update from the client. Compared to previous DP works [10, 14, 18] which either protect record-level privacy or user-level privacy, we are the first to preserve the two different levels of privacy simultaneously to the best of our knowledge.

3 DETAILED SYSTEM DESIGN WITH DIFFERENTIAL PRIVACY

This section presents our methods to achieve DP in DP-FedMeta. We detail our schemes, i.e., DP-AGR (**D**ifferentially **P**rivate **AG**regation **R**ule) and DP-AGRLR (**D**ifferentially **P**rivate **AG**regation **R**ule with **L**ocal **R**andomness), for achieving user-level DP and two-fold DP respectively. We first introduce adaptive clipping since it is a critical component for both algorithms.

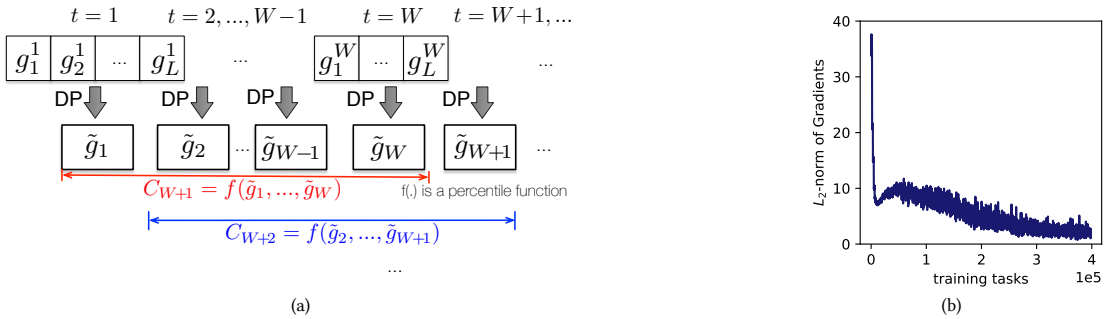
3.1 Adaptive Gradient Clipping Method

The goal of the adaptive gradient clipping method is to minimize the clipping threshold while preserving most of the gradient information. The method is based on the L_2-norm. One observation in our experiments is that gradient norms decrease significantly over the course of the training process as shown in Fig. 2(b). Therefore, an optimized clipping threshold for the early training stage will be too large for the later training stage.

In our adaptive clipping method, we use a sliding window with size W and step size ΔW to obtain a runtime clipping threshold C, illustrated in Fig. 2(a). $\Delta W = 1$ in our implementations. We introduce the concept of time step t, and t increases by 1 once one training iteration is completed. We assume there are L clients sampled to participate in FL in each iteration. Thus, L clients' gradients will be sent to the central server at the end of each learning iteration. Each entry g_i^t in the first row of boxes in Fig. 2(a) is the L_2-norm of gradient from the i-th client of the L sampled clients at the time step t. As shown in the second row of boxes in Fig. 2(a), we will obtain a noisy aggregation $\tilde{g}$ of the current L gradients. Notably, the noisy aggregation step is the key procedure of DP-AGR, not a design for the adaptive clipping method. The adaptive clipping method takes advantage of differentially private version of gradients for calculating a dynamic clipping threshold without incurring additional privacy loss. Specifically, the adaptive clipping threshold at time step $t+1$ is computed with a sequence of differentially private version of gradients before $t+1$ (i.e. , $\tilde{g}_{t-W+1}, \tilde{g}_{t-W+2}, ..., \tilde{g}_t$) by

$$C_{t+1} := f(\{\tilde{g}_{t-W+1}, ..., \tilde{g}_t\}, k) \tag{2}$$

where $f(S, k)$ represents the k-th percentile of a sequence S and time step $t > W$. We use constant clipping method for the first W time steps in the training process since the number of collected differentially private gradients is less than the window size. Another observation in our experiments is that gradient norms in DP-AGR exhibit sharp spikes due to the additive perturbation during the training process. The adaptive clipping threshold C_t (in DP-AGR) fluctuates when we directly apply the aforementioned



(a) (b)

Figure 2: (a) Illustration of our adaptive clipping scheme. $\tilde{g}_t$ is the differentially private version gradient at time step t. The central server computes adaptive clipping threshold C_t using DP gradients in a window of size W. (b) The L_2-norm of gradients during the training process.

window-sliding mechanism. We solve this issue by adding a smoothing scheme, i.e., updating C_t only when it gets no larger than the previous C_{t-1}.

3.2 Instantiating Meta-learning with MAML

In practice, the data generated by different clients may follow different distributions, e.g., one client has apple and orange images while another client has cat and dog images. In our paper, we propose to employ meta-learning [4, 9, 26] to learn a parameter initialization. Meta-learning learns common knowledge across a large number of tasks which is an optimized starting point for various new tasks as it can fast adapt to unseen tasks.

We leverage a widely accepted meta-learning paradigm, model-agnostic meta-learning (MAML) [9], to learn the common knowledge of multiple clients. In MAML, there are multiple tasks. Each task makes use of two datasets: *support set* for local model training and *query set* for meta-model training. The meta-learner maintains a global meta-model while a task-learner learns a base model for its tasks. Meta-model and base-model are neural networks with the same architecture but different weights, which can be represented by a mapping function f: from input $x \in \mathbb{R}^{w*h*c}$ to output $y \in \mathbb{R}^N$ (N is the number of classes). We use shortcuts f_Θ and f_θ to differentiate the two models.The training phase of MAML starts by initializing the base model as the current meta-model Θ, proceeds to train on the support set, and obtains a trained base model for each task. The training process of a base model for task $\mathcal{T}_i$ is:

$$\theta_i \leftarrow \Theta - \eta_1 \nabla_{\theta_i} \sum_{(x,y)\in \mathcal{D}^s_{train,i}} \mathcal{L}(f_{\theta_i}(x), y), \tag{3}$$

where $\mathcal{D}^s_{train,i}$ is the support set, η_1 is the learning rate, $\mathcal{L}(f_{\theta_i}(x), y)$ is the loss of θ_i, and ∇ is the differential operator for calculating the gradient. After learning L base-models for selected tasks $\mathcal{T}_s = \{\mathcal{T}_1, ..., \mathcal{T}_L\}$), the meta-model Θ is updated using the summation of loss from multiple training tasks' query set as follows:

$$\Theta \leftarrow \Theta - \frac{\eta_2}{L} \nabla_\Theta \sum_{i\in\mathcal{T}_s} \sum_{(x,y)\in \mathcal{D}^q_{train,i}} \mathcal{L}(f_{\theta_i}(x), y), \tag{4}$$

where η_2 is the learning rate for meta-model. MAML uses losses from multiple tasks to update Θ, thus obtains across-task knowledge. In implementation, MAML has two loops. Eq. (3) occurs in inner loop while Eq. (4) in outer loop. We can achieve a trained meta-model Θ which can be used to initialize an ML model for a client with a new task. We can achieve high accuracy with only a few pieces of data since the initialization contains the common knowledge.

Algorithm 1: DP-AGR (Client Side)

Input: Current global model Θ, local data $\mathcal{D}$
Output: gradient g
Function g =**Base-Model-Train**(Θ, $\mathcal{D}^s$, $\mathcal{D}^q$):
Initialize base-model: $\theta \leftarrow \Theta$;
Split local data $\mathcal{D}^s, \mathcal{D}^q \leftarrow \mathcal{D}$;
Update base-model: $\theta \leftarrow \theta - \eta_1 \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^s)$;
Gradient: $g \leftarrow \nabla_\Theta \mathcal{L}(\theta, \mathcal{D}^q)$.

3.3 DP-AGR for User-level DP

Our algorithm is completed by both the clients and the central server as a traditional FL algorithm. At the beginning, L clients are selected to participate the training process by the central server. To take the advantage of meta-learning, the selected clients randomly split their local data into two sets with equal size: a support set $\mathcal{D}^s_i$ and a query set $\mathcal{D}^q_i$. Each client trains its local model using its support set following Eq (3). Then each client further calculates a gradient of the trained model using the query set and sends the gradient to the central server. Instead of aggregating the received gradients as the traditional FL system, the central server first performs gradient clipping and noise adding. The user-level gradient of Client i is:

$$g_i = \frac{1}{|\mathcal{D}^q_i|} \nabla_\Theta \sum_{(x,y)\in \mathcal{D}^q_i} \mathcal{L}(f_{\theta_i}(x), y) \tag{5}$$

The gradient of the i-th client will be clipped as:

$$\bar{g}_i = \text{Clip}(g_i, C) \tag{6}$$

where Clip(a, C) denotes clip a by C. And the clipping threshold C is calculated using Eq. (2) of Sec. 3.1. We use L to denote the group of clients in one learning iteration. The aggregated gradient is:

$$\tilde{g} = \sum_{i=0}^{L} \bar{g}_i + \mathcal{N}(0, z^2 C^2 \mathbf{I}), \tag{7}$$

Algorithm 2: Central Server Aggregation in DP-AGR and DP-AGRLR

Input: Clients set $\mathcal{T}$, noise multiplier z, user sample size L, Learning round T, (ϵ, δ)
Output: meta-model Θ

Initialize t=0;
while (ϵ, δ)-*DP not exhausted and* $t <= T$ **do**
  Randomly Sample L clients $\mathcal{T}_s \leftarrow$ sample$(\mathcal{T}, L)$;
  for $i \in \mathcal{T}_s$ **do**
    $g_i \leftarrow$ **Base-Model-Train**$(\Theta, \mathcal{D}_i^s, \mathcal{D}_i^q)$ *by clients;
    $C \leftarrow$ adaptive-clipping$(\tilde{g}_{t-W}, ..., \tilde{g}_{t-1})$;
    Clip gradient: $\hat{g}_i \leftarrow g_i * \min(1, \frac{C}{\|g_i\|})$;
  $\tilde{g}_t \leftarrow \frac{1}{L}\left(\sum_i \hat{g}_i + \mathcal{N}(0, z^2C^2\mathbf{I})\right)$;
  Update meta-model $\Theta \leftarrow \Theta - \eta_2\tilde{g}$;
  $(\epsilon, \delta) \leftarrow$ Compute-RDP$(z, L, t, *arg)$; t+=1;

where $\mathcal{N}(0, z^2C^2\mathbf{I})$ is the additive Gaussian noise. The perturbed summation $\tilde{g}$ is then used to update the meta-model:

$$\Theta \leftarrow \Theta - \eta_2\tilde{g} \tag{8}$$

In summary, DP-AGR works as follows:

- **Client Side**: A local client splits its own dataset as support set $\mathcal{D}^s$ and query set $\mathcal{D}^q$. The local client initializes its local model as the received global model Θ and trains the model using $\mathcal{D}^s$. Finally, it computes local gradient g with $\mathcal{D}^q$ and submits g to the central server (See **Algorithm1**).
- **Central Server Side**: The central server performs gradient clipping for a group of randomly selected clients using the current clipping threshold in each round. The clipping threshold C is updated automatically at each round using our adaptive clipping method. The central server adds noise to the summation of all clipped gradients $\sum_i \hat{g}_i + \mathcal{N}(0, z^2C^2\mathbf{I})$ (See **Algorithm2**). We calculate the average $\tilde{g}$ of the summation to update the global model Θ (we assume the data amount in every client is the same, otherwise we will compute weighted average taking into account the data amount).
- The **Final Output** our algorithm is a differentially private model Θ. Compared to conventional FL, our global model Θ is not a ready-to-use model but an initialization parameter that has fast adaptation capability. Model consumers can customize Θ to their task with only a few local data points.

To conserve the privacy budget, we enforce that each client is selected up to once. We refer to this enforcement as a one-pass meta-training process. Model accuracy is preserved even with the one-pass learning process thanks to the meta-learning framework and a large number of clients (the typical number of local devices participating in an FL system can be millions as shown in [11]).

3.4 DP-AGRLR for Two-fold DP

Compared to DP-AGR, DP-AGRLR *additionally* protects the data privacy of clients from the honest-but-curious central server. To this end, a client does not report its true gradients but noisy ones. Note that the process at the central server-side of DP-AGRLR remains the same as DP-AGR. The local client first bounds the impact of an individual record $\sum_{(\boldsymbol{x}_j, y_j) \in \mathcal{D}_i^q}$ by clipping the record-level gradient with threshold C.

Algorithm 3: DP-AGRLR (Client Side)

Input: Current global model Θ, local data $\mathcal{D}$, DP parameter (ϵ_0, δ_0), C_0, z_0
Output: gradient g

Function g = **Base-Model-Train**$(\Theta, \mathcal{D}^s, \mathcal{D}^q)$:
Initialize base-model: $\theta \leftarrow \Theta$;
Split local data $\mathcal{D}^s, \mathcal{D}^q \leftarrow \mathcal{D}$;
$z_0 \leftarrow$ compute_noise$(\epsilon_0, \delta_0, *args)$
for $(x_i, y_i) \in \mathcal{D}^s$ **do**
  record- level gradient: $g_i \leftarrow \nabla_\theta \mathcal{L}(\theta, x_i)$;
  clip gradient: $\hat{g}_i \leftarrow g_i * \min(1, \frac{C_0}{\|g_i\|})$;
$\tilde{g} \leftarrow \frac{1}{|\mathcal{D}^s|}\left(\sum_i \hat{g}_i + \mathcal{N}(0, (z_0C_0)^2\mathbf{I})\right)$;
update base-model: $\theta \leftarrow \theta - \eta_1\tilde{g}$;
for $(x_i, y_i) \in \mathcal{D}^q$ **do**
  record-level gradient: $g_i \leftarrow \nabla_\theta \mathcal{L}(\theta, x_i)$;
  clip gradient: $\hat{g}_i \leftarrow g_i * \min(1, \frac{C_0}{\|g_i\|})$;
$g \leftarrow \frac{1}{|\mathcal{D}^q|}\left(\sum_i \hat{g}_i + \mathcal{N}(0, (z_0C_0)^2\mathbf{I})\right)$.

$$\bar{g_j} = \text{Clip}(\nabla_\Theta \mathcal{L}(f_\theta(\boldsymbol{x}_j), y_j)), C) \tag{9}$$

where C is the clipping threshold. This clipping threshold is determined by the local client using our proposed adaptive clipping method. The local client then computes a Gaussian noise and adds the noise to the true gradients.

$$\tilde{g} = \sum_{j=0}^{|D_i^q|} \bar{g_j} + \mathcal{N}(0, z^2C^2\mathbf{I}), \tag{10}$$

The noisy gradient $\tilde{g}$ is sent to the central server. DP-AGRLR can hide the existence of an individual record against the curious central server. Two-fold DP targets a much higher level of privacy protection, thus inevitably sacrificing more model accuracy. The algorithm is shown in **Algorithm 3**.

3.5 Privacy-Utility Trade-off

In DP-AGR we use one-pass training over clients (i.e., no client is used more than once) to boost privacy. In learning round, we sample the clients with $q = \frac{L}{N_{client}}$. The overall privacy loss of DP-AGR can be represented as:

$$\alpha(\lambda) \leq \frac{q\lambda(\lambda+1)}{(1-q)z^2} + O(q^2\lambda^3/z^3). \tag{11}$$

where z is the noise multiplier and λ is moments number. The privacy loss $\alpha(\lambda)$ is related to q but not the learning rounds. Therefore, we can boost accuracy by adding more clients and keep the same level privacy protection if we keep q as constant. A smaller privacy loss indicates a better privacy protection. The privacy parameter (ϵ, δ) is directly related to the accumulated privacy loss by equation $\delta = \min_\lambda \exp(\alpha(\lambda) - \lambda\epsilon)$. We can then use this equation to convert the moments bound $\alpha(\lambda)$ to the (ϵ, δ) guarantee.

The derivation of Eq. (11) is based on the moments accountant [1] that offers the state-of-the-art estimation of privacy loss of Gaussian

mechanisms. For traditional DL system, each record contributes to the training process multiple times through multiple epochs. Different from traditional DL, a client contributes to the training system only once in meta-learning. We will highlight the impact of such a difference on privacy performance in the following.

For moments accountant [1], the λ-th moments of privacy loss of one SGD step satisfies that,

$$\alpha_{\mathcal{M}}(\lambda) \leq \frac{q^2\lambda(\lambda+1)}{(1-q)z^2} + O(\frac{q^3\lambda^3}{z^3}), \tag{12}$$

where mechanism $\mathcal{M} : \bar{g} = \sum_{i\in L} g_i + \mathcal{N}(0, z^2C^2I)$ represents gradients summation and perturbation, L denotes a batch, and q is the sampling probability. $O(\cdot)$ is the Bachmann–Landau notation that describes how closely a finite series approximates a given function in the case of an asymptotic expansion. The inequality holds if $z \geq 1$ and $q \leq \frac{1}{16z}$. In statistics, 'moment' of a distribution is the quantitative measures related to the shape of the distribution (e.g., the first moment is the expected value and the second central moment is the variance).

By applying composability theory of DP [1], the accumulated privacy loss after T updates are:

$$\alpha(\lambda) \leq \sum_j \alpha_{\mathcal{M}_j}(\lambda) \leq T\left(\frac{q^2\lambda(\lambda+1)}{(1-q)z^2} + O(\frac{q^3\lambda^3}{z^3})\right). \tag{13}$$

For DL, sampling probability is $q = \frac{L}{N_{train}}$, the number of updates is $T = N_{epoch} * \frac{N_{train}}{L}$, where N_{train} is the number of data points in training dataset and L is the batch size. Therefore, the above equation can be rewritten as

$$\alpha(\lambda) \leq \frac{N_{epoch} * q\lambda(\lambda+1)}{(1-q)z^2} + O(N_{epoch} * \frac{q^2\lambda^3}{z^3}). \tag{14}$$

Different from traditional DL, for DP-AGR, $q = \frac{L}{N_{client}}$ and $T = \frac{N_{client}}{L} = \frac{1}{q}$ is the total number of learning rounds where L denotes the number of clients for one learning round and N_{client} the total number of clients. Considering that we use one-pass training over clients in DP-AGR, the inequality in Eq. (13) can be rewritten as Eq. (11).

Eq. (14) and Eq. (11) have different implications. According to Eq. (14), if we increase N_{epoch}, $\alpha_{\mathcal{M}}(\lambda)$ also increases meaning privacy protection is worse. Therefore, more training iterations for DL increase the privacy loss. On the contrary, according to Eq. (11), the privacy loss will not change with adding more clients if we keep q as constant.

4 PRIVACY ANALYSIS

We would like to show that adaptive clipping in DP-AGR and DP-AGRLR does not result in additional privacy loss. Recall that DP-AGR works in the scenario that the central server is trusted while clients are honest-but-curious. DP-AGRLR is designed for a stronger adversary model in which both clients and the central server are honest-but-curious. For convenience, we denote the above-mentioned adversary models as Adversary Model 1 and Adversary Model 2, respectively.

For Adversary Model 1, privacy protection is applied by the central server for the individual clients. The central server applies DP at the cloud to train a meta-model. To improve performance while preserving the data privacy of individual clients, our proposed adaptive clipping method (see Sec. 3.1) operates on the deferentially private version of gradients, and thus is a post processing step from the privacy perspective. More concisely, the adaptive clipping threshold C_t at time step t is computed from the differentially private version of gradients $\tilde{g}_{t-W}, ...\tilde{g}_{t-1}$ as:

$$C_t = \mathrm{P}(\{\tilde{g}_{t-W}, \tilde{g}_{t-W+1}, ..., \tilde{g}_{t-1}\}, k), \tag{15}$$

where $\tilde{g}_t$ denotes the differentially private version of gradient at time step t, W represents the size of the sliding window, and $\mathrm{P}(S, k)$ represents the k-th percentile of a sequence S. All the gradients $\tilde{g}_{t-W}, \tilde{g}_{t-W+1}, ..., \tilde{g}_{t-1}$ are differentially private. According to the *post-processing* rule [7], further computation on differentially private data will not incur additional privacy loss. Therefore, the calculation of adaptive clipping threshold C_t does not lead to additional information leakage.

Similarly, adaptive clipping does not compromise data privacy in Adversary Model 2. In summary, the adaptive clipping method in DP-AGR and DP-AGRLR does not harm the target DP level. The privacy loss of DP-AGR and DP-AGRLR comes solely from the accumulated privacy loss of updating the meta-model.

5 IMPLEMENTATION AND EXPERIMENTAL SETTINGS

We implemented DP-FedMeta in PyTorch. We ran all our experiments on a server equipped with a 3.3 GHz Intel Core i9-9820X CPU, three GeForce RTX 2080 Ti GPUs, and Ubuntu 18.04.3 LTS. In DP-FedMeta, we trained a VGG-Net [23] using Adam optimizer with a learning rate 0.01 for outer-loop. For inner-loop training, we used the SGD optimizer with a learning rate 0.1. The optimizer setting is adopted from the benchmark meta-learning algorithm [3]. Our code is available at https://github.com/ning-wang1/DPFedMeta.

For the experiment, we simulated federated meta-learning using data from three popular datasets, including Omniglot [13], CIFAR-FS [4], and Mini-ImageNet [25]. Table 1 provides the detail of the datasets including the number of classes, number of samples per class, image size, the splitting ratio among the meta-training set, validation set, and meta-testing set. N_c represents the number of classes and N_s denotes the number of samples per class in the corresponding dataset.

Table 1: Datasets used for evaluation of DP-FedMeta.

Dataset	$N_c \times N_s$	Image size	Train:Val:Test
Omniglot	1623×20	28 ×28 × 1	71: 3:26
CIFAR-FS	100×600	32×32 × 3	64:16:20
Mini-ImageNet	100×600	84×84 × 3	64:16:20

We simulated 400,000 mobile clients and 600 model consumers, assuming each client is with one learning task. The number of selected mobile clients for each learning round was set to 1,600. We followed the well-received practice in [25] to distribute the whole dataset to mobile clients. Firstly, we separated the total classes into two non-overlapping sets: one set for mobile clients and the other for model consumers. Second, each mobile client/model consumer was assigned with a set of 30 labelled examples from 5 classes

in the corresponding set. Different mobile clients were allowed to have overlapping data. N-way K-shot means we test a model initialization with $K*N$ data points evenly extracted from N classes. Model consumer will first initialize its model with the trained meta-model from the central server. Then it continues training the model with $K * N$ data points and then evaluate model accuracy. The average testing accuracy over all model consumers will be the final reported accuracy.

We estimate the privacy loss in the form of (ϵ, δ) at each learning round using the RDP moments accountant (https://github.com/tensorflow/privacy), and terminate the training process if the privacy budget is used up. Unless otherwise mentioned, the default settings are: the DP scenario is user-level, the few-shot learning case is 5-way 1-shot on Omniglot, clip percentile $k=90$, noise multiplier $z=1$, and *user sample* size $L=1600$, clip window $W=10, \Delta W=1$. We run each experiment 3 times and obtain the average meta-testing accuracy.

6 EVALUATIONS

we implement two algorithms, DP-AGR and DP-AGRLR, to accommodate different trust levels of the central server. Both DP-AGR and DP-AGRLR output a model initialization parameter Θ at the end of meta-training. In this section, we mainly answer three questions as follows:

- **Q1:** *'How is the performance of our adaptive clipping method compared to constant clipping and other adaptive clipping baselines?'*
- **Q2:** *'How to achieve a good trade-off between accuracy and privacy? Specifically, how to improve model accuracy under a fixed ϵ? '*
- **Q3:** *'Will differentially private meta-learning maintain reasonable accuracy? Specifically, will it achieve higher accuracy than other differentially private initialization methods (e.g., DP transfer learning) or random initialization?'*

6.1 Adaptive Clipping

In this part, we will keep all other factors the same and only evaluate the performance of different clipping methods. We compare DP-AGR to the constant clipping method wildly used in [1, 2, 10, 29] which is to use the median norm of unclipped gradients as the clipping threshold. DP-AGR uses an adaptive clipping method with one configurable parameter k. As shown in Fig. 3, the adaptive clipping method outperforms the constant clipping method. We also present an experimental comparison between our adaptive clipping method and one other adaptive clipping method (i.e., the AQC method [2]). Fig. 3 shows that our adaptive clipping method outperforms the AQC method by accuracy, indicating that DP-AGR achieves a better trade-off between accuracy and privacy.

Our adaptive clipping method has one configurable parameter k. As shown in Fig. 4(a), the adaptive clipping method changes when k varies. DP-AGR achieves the highest testing accuracy 93.9% at $k = 90$. With a proper value of $k \geq 20$ (not too small), we can expect a better model accuracy with adaptive clipping than with constant clipping. Our adaptive clipping method achieves a better trade-off between accuracy and privacy.

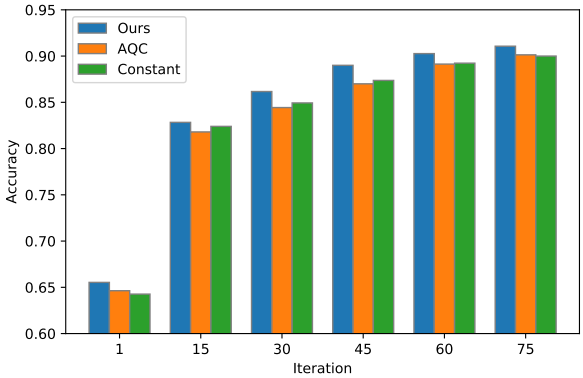


Figure 3: Testing Accuracy. Our Adaptive Clipping Method vs. AQC [2] and constant clipping with 5-way 1-shot learning on Ominiglot dataset.

6.2 Trade-off between Accuracy and Privacy

To achieve a better trade-off between accuracy and privacy of DP-AGR, we tune other parameters, e.g., noise level and user sampling size. Specifically, we **fix** the privacy budget to see how to improve the model accuracy.

Noise Multiplier Fig. 4(b) shows that DP-AGR achieves 93.9% accuracy at $z=1$, and the accuracy drops with the increase of z and tends to be stable at around 73%. We should start from a small z and increase z only when you use up privacy budget before training converges.

***User Sample* Size** We fixed the privacy budget to evaluate the accuracy with various *user sample* size L ranging from 100 to 5,600. As shown in Fig. 4(c), DP-AGR achieves the best accuracy when $1600 \leq L \leq 4000$. Testing accuracy drops fast when L is larger than 4,000. We looked into the experiment and found that the training process was early terminated because of depleting DP budget when $L > 4000$.

Combination of noise multiplier and *user sample* size We find that the optimal L strongly depends on the noise multiplier z. We demonstrate the above finding by Fig. 4(d), in which we show the testing accuracy of different combinations of L and z. For $z = 1$, accuracy reaches the peak at $L = 1,600$ and drops quickly as $L \geq 4,800$ because of draining the privacy budget. For $z = 2$, accuracy peaks at $L=3,200$ and stays stable with larger L. For $z=3$, accuracy peaks at $L=4,800$ and stays stable with larger L. Based on such observations, we conclude that the best L is proportional to z and should be tuned accordingly if z changes.

Guidelines for k, z, and L From the above results, we share our general guidelines to work with DP. First, we recommend to start from a small noise multiplier z (e.g., 1) and increase z only when you can not guarantee convergence before using up the privacy budget. Second, we recommend starting with a relatively large L especially when z is large. A good start is $L = \frac{N_{task}}{16*z}$ as $L < \frac{N_{task}}{16*z}$ by moments accountant. We can decrease L only when you can not guarantee convergence before using up the privacy budget. Compared with the non-private training, we need apply a larger learning rate since the training rounds are limited because of privacy concerns. Finally, as privacy parameter ϵ is only determined by z and L, we can adjust other parameters, such as k, to boost the model accuracy.

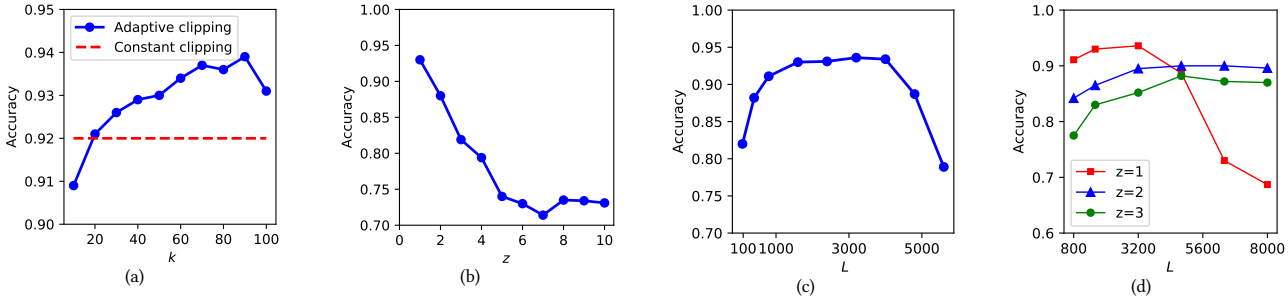


Figure 4: The impact of parameters on model accuracy when the privacy budget is fixed. (a) Vary k (i.e., the percentile number of our Adaptive Clipping Method) (b) Vary z. (c) Vary L. (d) The impact of different combinations of noise z and *user sample* size L on model accuracy.

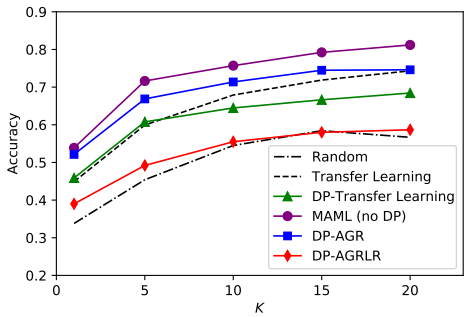


Figure 5: Model accuracy with different initialization on CIFAR-FS dataset. The x-axis K denotes the number of shots in the *Initialization testing* phase.

We explore various k values in multiple datasets and find that the accuracy peaks at different k values in different datasets. However, the adaptive clipping method outperforms the fixed clipping when $k \geq 50$ across all three datasets, implying that the median is a good initial choice.

6.3 Comparison with Other Model Initialization Methods

Meta-learning is a model initialization method that outputs an initialized meta model for clients. This partly resembles transfer learning in that one model is used for initializing the training process for another model. In this experiment, we evaluate the model utility (i.e., accuracy) of DP-AGR and DP-AGRLR compared with other model initialization methods, including `Random`, `MAML`, `transfer learning` [6], and `DP-transfer learning`. The first three methods are with no privacy protection. `DP-transfer learning` is the transfer learning method incorporated DP.

We use CIFAR-FS dataset (contains 100 classes) as an example to show the comparisons. We first divide the dataset into two separate sets: *Initialization training set* (i.e., $\mathcal{D}_{train}$) consists of 80 classes and *Initialization testing set* ($\mathcal{D}_{test}$) consists of 20 classes. We perform both meta-training and DP meta-training to respectively obtain a meta-model Θ_{maml}, Θ_{dp-agr} and $\Theta_{dp-agrlr}$. We also conduct a transfer learning process and a DP transfer learning process [6] on the same set to achieve a trained model Θ_{trans} and $\Theta_{trans-dp}$ respectively. In the *Initialization testing* phase, the test set $\mathcal{D}_{test}$ is organized as tasks and each task contains a support set and a query set. We generate 600 tasks from $\mathcal{D}_{test}$. For each task, we use the trained initialization parameter to initialize its neuron network, train 30 epochs on its support set and obtain testing accuracy on its corresponding query set. The average testing accuracy on all 600 tasks is used as the ultimate evaluation metric.

Fig. 5 show the results of different initialization methods on CIFAR-FS. We can see that `MAML` achieves the best testing accuracy among all initialization methods as expected. The proposed `DP-AGR` outperforms all other initialization methods including `transfer learning` and `DP-transfer learning` by achieving higher accuracy. The `Random initialization` is without doubt the worst among the five methods. We also see that when K is large enough, the accuracy of `transfer learning` approaches that of `DP-AGR`.

Besides the transfer learning method, we also compare DP-AGR and DP-AGRLR with state-of-the-art differentially private meta-learning solution, GBML [14]). In Table 2, *GBML* is the state-of-the-art differentially private meta-learning algorithm proposed in [14], *Random initial* denotes that the learning begins with random initialization, and *Non-private* denotes the MAML algorithm without privacy consideration. We can see that DP-AGR outperforms GBML by achieving higher accuracy in all six learning tasks. DP-AGRLR achieves higher accuracy than GBML on both Omniglot dataset and CIFAR-FS dataset while showing lower accuracy on Mini-ImageNet dataset.

In Table 2, both DP-AGR and DP-AGRLR are with $\epsilon = 1.5$ in task level DP. The record-level DP parameter of DP-AGRLR is $\epsilon = 2.5$. The baseline GBML is with $\epsilon = 9.5$. A smaller ϵ is better for privacy protection. We can see that both DP-AGR and DP-AGRLR achieves higher accuracy than GBML on both the Omniglot dataset and the CIFAR-FS dataset. Further, DP-AGR is better in model accuracy than DP-AGRLR while DP-AGRLR provides much more strict privacy protection. Noted that GBML has a different DP notion from DP-AGR and DP-AGRLR, and its DP notion is more strict than DP-AGR but less strict than DP-AGRLR. The practical privacy leakage relies on both the value of DP parameters and the DP notion itself. For example, DP-AGRLR provides better privacy protection since it has a more strict DP notion and a smaller ϵ than GBML. On the other hand, it is not clear whether DP-AGR provides better privacy protection than GBML since its DP notion is less strict but has a smaller ϵ than GBML. We will consider evaluating the resilience

Table 2: Meta-testing accuracy (%) with DP-AGR, DP-AGRLR and other baselines.

Dataset	N-way K-shot	Random initial	Non-private	Private Algorithm		
				DP-AGR	DP-AGRLR	GBML [14]
Omniglot	5-way 1-shot	49.2	99.4	93.9	72.4	44.6
	5-way 5-shot	61.0	99.8	96.8	89.7	75.0
CIFAR-FS	5-way 1-shot	33.8	61.0	47.1	39.0	32.2
	5-way 5-shot	45.4	78.6	58.2	49.2	48.6
Mini-ImageNet	5-way 1-shot	23.3	51.7	37.3	27.7	26.1
	5-way 5-shot	24.2	65.3	48.8	33.2	38.0

of different DP algorithms against inference attacks to test their practical privacy protection in future work.

6.4 Computation Time

We present the per-task computation overhead of DP-AGR and DP-AGRLR in Table 3. Noted that each task contains $N * K$ data records for N-way K-shot learning. We measure the training time of 5-way 1-shot learning on three different datasets. We can see that the per-task computation time of DP-AGR is only slightly longer than that of MAML. On our server, the training time of a typical DP-AGR algorithm on the Omniglot dataset with 400,000 tasks is around 6 hours. DP-AGR achieves comparable computational performance with the original non-private MAML algorithm. Compared to DP-AGR, DP-AGRLR is more time-consuming due to the need for computing per-record gradients at the local device. Although the DP-AGRLR is more time-consuming, it is scalable since the time-intensive per-record gradient computation is done at the distributed local devices of clients rather than the central server and the number of records within an individual device is small. Therefore, we believe DP-AGR and DP-AGRLR are affordable for a wide range of learning applications.

Table 3: Per-task computation time

Dataset	MAML	DP-AGR	DP-AGRLR
Omniglot	39.9ms	54.7ms	0.52s
CIFAR-F	68.7ms	81.3ms	1.06s
Mini-ImageNet	112.3ms	102.7ms	1.17s

7 RELATED WORK

After Song et al. [24] formulated the first differentially private stochastic gradient descent (DP-SGD) algorithm, Abadi et al. [1] proposed a powerful tool (moments accountant) for tight privacy analysis and control in DP-SGD. Pathak et al. [21] were the first to study the DP framework in a multiparty setting. This work provided a privacy-preserving protocol to compose a differentially private aggregate classifier using classifiers trained locally by separate mutually untrusted parties. Shokri et al. [22] investigated DP-SGD for distributed dataset and proposed a practical system that enables multiple parties to learn an accurate and private neural network model jointly. More recently, federated learning (FL) emerged as an promising framework since its introduction by Google researchers [12, 17] with its initial goal to learn from data stored in users' smartphones or tablets. Though the data is stored locally and never exchanged among clients, it has been shown that clients' data is still susceptible to inference attacks [19, 28]. [10, 18] explored the client-level privacy in FL and focused on balancing the trade-off between privacy and utility, assuming the central server is trusted. In order to save more privacy budget, Andrew et al. [2] proposed adaptive quantile clipping (AQC) to make an estimate of a target quantile of the distribution of unclipped gradient norms.

Building on the recent advances in meta-learning [9, 20, 26] and FL, a significant body of work has been devoted to federated meta-learning [5, 8, 16, 31]. Despite the compelling applications of federated meta-learning in many domains, its privacy problem remains less understood. Li et al. [14] are the first to study the privacy problem in meta-learning. It directly applies a DP-enabled meta-learning algorithm to the federated setting, which is the current state of the art and also the most relevant work to ours. In our paper, we explored two notions of DP, including user-level DP and two-fold DP, which are different from the notions of DP studied in [14]. Compared to [14], we achieve relatively high model accuracy with a much lower DP budget.

8 CONCLUSION

We develop DP-FedMeta, a differentially private federated meta-learning architecture that protects clients from inference-based data privacy attacks. To deal with different requirements on privacy levels pertaining to the trust on the central server, we customize two DP mechanisms for DP-FedMeta, DP-AGR and DP-AGRLR. DP-AGR protects the participating information of an individual client against curious clients and model consumers. DP-AGRLR provides two-fold privacy protection from curious clients (including model consumers) and an honest-but-curious central server. We design an adaptive gradient clipping method and a one-pass training process to conserve the privacy budget. Our adaptive gradient clipping shows superior performance over both constant gradient clipping and another adaptive one. Extensive evaluation on multiple datasets demonstrates that our DP-AGR outperforms the state-of-the-art differentially private federated meta-learning solution by achieving a much lower privacy budget without sacrificing additional accuracy. Further, we provide useful insights and heuristic guidelines on how to set various parameters, such as gradient clipping threshold, noise multiplier, and *lot* size, which are applicable when incorporating DP in different types of deep learning problems.

ACKNOWLEDGMENTS

This work was supported in part by the Office of Naval Research under grant N00014-19-1-2621, the US National Science Foundation under grants CNS-1837519 and CNS-1916902, and the Army Research Office under grant W911NF-20-1-0141.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 16)*. ACM, 308–318.

[2] Galen Andrew, Om Thakkar, H Brendan McMahan, and Swaroop Ramaswamy. 2019. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871* (2019).

[3] Antreas Antoniou et al. 2019. How to train your MAML. In *International Conference on Learning Representations (ICLR 19)*.

[4] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. 2019. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR 19)*.

[5] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. 2018. Federated meta-learning for recommendation. *arXiv preprint arXiv:1802.07876* (2018).

[6] Jeff Donahue, Yangqing Jia, Oriol Vinyals, et al. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning (ICML 14)*. 647–655.

[7] Cynthia Dwork et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[8] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *Advances in Neural Information Processing Systems(NeurPIS 20)*. 3557–3568.

[9] Chelsea Finn et al. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML 17)*. 1126–1135.

[10] Robin C Geyer et al. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* (2017).

[11] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and Open Problems in Federated Learning. *arXiv preprint arXiv:1912.04977* (2019).

[12] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[13] Brenden M Lake et al. 2015. Human-level concept learning through probabilistic program induction. *Science* 350, 6266 (2015), 1332–1338.

[14] Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. 2020. Differentially Private Meta-Learning. In *International Conference on Learning Representations*.

[15] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[16] Sen Lin et al. 2020. A Collaborative Learning Framework via Federated Meta-Learning. In *2020 International Conference on Distributed Computing Systems*.

[17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics (AISTATS 17)*. 1273–1282.

[18] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *International Conference on Learning Representations (ICLR 18)*.

[19] Milad Nasr et al. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP 19)*. IEEE, 739–753.

[20] Alex Nichol et al. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999* 2 (2018), 2.

[21] Manas Pathak et al. 2010. Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Information Processing Systems(NeurPIS 10)*. 1876–1884.

[22] Reza Shokri et al. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (CCS 15)*. ACM, 1310–1321.

[23] Karen Simonyan et al. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[24] Shuang Song et al. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing (GlobalIPS 13)*. IEEE, 245–248.

[25] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems (NeurIPS 16)*. 3630–3638.

[26] Lingxiao Wang, Qi Cai, Zhuoran Yang, and Zhaoran Wang. 2020. On the global optimality of model-agnostic meta-learning. In *International Conference on Machine Learning*. PMLR, 9837–9846.

[27] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. 2009. Learning your identity and disease from research papers: information leaks in genome wide association study. In *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 534–544.

[28] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE Conf. on Computer Communications (INFOCOM)*. IEEE, 2512–2520.

[29] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. on Information Forensics and Security* 15 (2020), 3454–3469.

[30] Nan Wu, Farhad Farokhi, David Smith, and Mohamed Ali Kaafar. 2020. The value of collaboration in convex machine learning with differential privacy. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 304–317.

[31] Sheng Yue, Ju Ren, Jiang Xin, Sen Lin, and Junshan Zhang. 2021. Inexact-ADMM Based Federated Meta-Learning for Fast and Continual Edge Learning. In *Proceedings of the 22ed International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. 91–100.

[32] Jingwen Zhang, Jiale Zhang, Junjun Chen, and Shui Yu. 2020. Gan enhanced membership inference: A passive local attack in federated learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.