Meta-Learning with Differential Privacy

ABSTRACT

Wide adoption of federated learning has been hindered by non-iid data statistics at participants and high communication cost during iterative training. Federated meta-learning not only overcomes such drawbacks but also exhibits desired properties in generalizing to unseen learning tasks with fast adaptation on a small dataset. However, federated meta-learning, like deep learning, still suffers from attacks such as membership inference and model inversion. In this paper, we propose DP-MAML, a federated meta-learning algorithm with differential privacy design. We motivate and explore two different levels of differential privacy to accommodate different privacy needs. Extensive experiments on Omniglot, CIFAR-FS, and Mini-ImageNet confirm the feasibility of DP-MAML to train a meta-model with a low privacy budget. We also show that DP-MAML outperforms pre-training as an initialization method.

CCS CONCEPTS

• **Security and privacy**;

KEYWORDS

differential privacy, meta-learning, federated learning

ACM Reference Format:

. 2018. Meta-Learning with Differential Privacy. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Catalyzed by the wide availability of massive amounts of data, deep learning has achieved great success in many sectors of society. However, increasing aggregation of data has raised significant concerns over the privacy in both industry and government, leading to regulation such as GDPR [1] and California Privacy Act [2]. In many cases, it is often impossible to aggregate the data at one central location due to laws and regulations such as Health Insurance Portability and Accountability Act (HIPPA) [3].

Building on the increasingly sophisticated computation power and storage capability at the end points, federated learning was proposed by [15, 19, 29] to enable collaboratively training over distributed data. In federated learning, the central server maintains a global model while local participants upload only gradient update based on their data to jointly train the global model. Federated learning comes with the intrinsic drawback of non-iid data statistics and tremendous communication cost between the central server and participants in iterative training. To tackle these challenges, Chen et al. [6] proposed federated meta-learning, which combines federated learning and meta-learning together to overcome non-iid statistics and accelerate model training.

Unlike the basic deep learning method, meta-learning learns common/meta-knowledge across a large amount of different machine learning tasks[4, 9]. When we apply meta-learning to federated learning, the central server of federated meta-learning maintains a global meta-model while participants perform meta-learning locally and upload gradients from their local data to jointly update the meta-model. Compared with federated learning, federated meta-learning has several fascinating advantages. First, its meta-model can be used for initialization of learning tasks of which the categories have not been seen in the training process due to its ability for generalization. Conversely, federated learning models can only classify known categories. Second, its meta-model can enable a model consumer to obtain a good local model using only a small dataset due to its ability of fast adaptation. Finally, its communication cost usually is smaller due to the smaller size of a meta-model and fewer communication rounds. Despite of the recent success in performance, deep learning, including federated meta-learning, has been demonstrated to suffer from a variety of attacks including membership inference [24, 31, 34] and model inversion [10, 21, 33]. To address these security problems, differential privacy (DP) [8] has been one of the most effective tools [1, 27, 33, 41].

In this paper, we present our exploration of incorporating differential privacy into federated meta-learning. Using Model Agnostic Meta-Learning in [9] (MAML) algorithm as a concrete meta-learning model, we present DP-MAML, our multi-level approach towards protection of user privacy in meta-learning, and highlight the pros and cons of each level with extensive experiments.

The concept of task is unique to meta-learning, which is often used to referred to as the meta-training process of a participant. Each task has two datasets: support set used for local model and query set used for meta-model. For the i-th task $\mathcal{T}_i$, individual participant uses the meta-model Θ to initialize its own neural network, then trains on its support set to obtain a local model θ_i. The gradients g_i for meta-model update is then computed using its query set. then uploaded to the central server which is used to jointly update Θ. Comparing federated meta-learning to federated learning, the differences of their training process root in meta-learning. First, gradients are computed on support set and query set separately for federated meta-learning while on the whole dataset for federated learning. Second, loss of local models on query set are used to update the meta-model, which is a completely different model. Although the unique learning structure in meta-learning allows meta-knowledge to be extracted from all tasks (i.e., local models), the meta-model does become highly sensitive to noise added to

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Woodstock '18, June 03–05, 2018, Woodstock, NY

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

guarantee differential privacy. This has lead us to reconsider the privacy definition in this renewed federated meta-learning setting.

We observe that data of different tasks are more likely to exhibit various characteristics, enhancing uniqueness of each task. This motivates us to explore task-level differential privacy, which is to bound the probability that an attacker is able to differentiate whether or not a specific task participates the meta-training process or not. In task-level DP, each update of meta-model is regarded as a DP mechanism and the final meta-model is the result of multiple DP mechanisms. Based on composition theorems of DP, we can determine privacy level of the final meta-model. To mitigate the noise from DP, we also proposed adaptive clipping method to preserve accuracy without harming privacy performance (i.e., maintaining a fixed noise multiplier). For completeness, we also investigated the impact of sample-level differential privacy in federated meta-learning, which aims to bound the probability that an attacker is able to differentiate whether or not a specific sample participates the meta-training process or not. Although sample-level DP has been explored in previous work [1, 27, 33, 41], federated meta-learning uses gradients from local models to update meta-model, increasing the difficulty of achieving sample-level DP significantly.

We have conducted extensive experiments to assess the impacts of the two different levels of differential privacy on the result of the learning. Base on our experience, we provide key factors crucial to characterizing two levels of privacy. We summarize the contributions of the paper as follows:

- We proposed Differential Private Model Agnostic meta-Learning (DP-MAML), a federated meta-learning algorithm which guarantees the trained meta-model is differentially private. Going forward, We explore task-level DP, which is unique in federated meta-learning, and sample-level DP, which is different from that in federated/machine learning due to the training process of meta-learning. We designed algorithms to achieve these two DPs.
- We investigated the different characteristics of privacy level of DP-MAML and stand-alone deep learning system with DP and provided formal explanation.
- We thoroughly evaluated the performance of DP-MAML under two DP levels on three well-known datasets, i.e., Omniglot, CIFAR-FS, and mini-ImageNet. We evaluated the impact of various factors including gradient clipping threshold, noise multiplier, and *lot* size on model accuracy and privacy and summarized guidelines for adopting DP-MAML. On Omniglot, DP-MAML was able to achieve 93.9% accuracy for 5-way 1-shot learning under $(1.5, 10^{-6})$-DP budget while MAML without DP 99.5%, demonstrating its effectiveness. We also compared the performance of DP-MAML and pre-training (i.e., transfer learning) as initialization method showing that DP-MAML outperformed pre-training and DP-pre-training.

2 RELATED WORK

Three lines of work are closely related to DP-MAML: 1) DP in deep learning 2) meta-learning, and 3) federated meta-learning.

It has been demonstrated in the past several years that deep learning is vulnerable to model inversion attacks [10, 21, 33] and membership inference attacks [24, 31, 34]. Differential privacy is an effective mechanism to defend against such attacks. Song et al. [36] were the first to study DL with DP. They proposed differentially private SGD (DP-SGD), which incorporated DP mechanism in gradient computation step. Specifically, DP-SGD clips the gradients to bound the sensitivity and adds random noise to the gradients thus achieving DP. After the first work, researchers focused on various settings of this subject. In [1], Abadi et al. used moments accountant mechanism to tight privacy analysis for DP-SGD. In [13], Jayaraman et al. evaluated the privacy loss of three popular DP variations (i.e., DP with advanced composition [8], zero-concentrated DP [5], and Rényi DP [22]) against membership inference attacks [34, 40] to logistic regression and neural network models. In [29], Pathak et al. first studied deep learning with DP in multiparty setting while in [33], Shokri et al. investigated DP-SGD in distributed dataset setting. In [20], McMahan et al. further extended DP-SGD for federated learning. Compared to the above work, we are the first to adapt DP-SGD to (federated) meta-earning and explore two levels of DP considering unique training process of meta-learning.

Meta-learning is one of the most heavily researched areas in machine learning community. DP-MAML is most related to MAML, proposed by Finn et al. in [9]. MAML is the first meta-learning algorithm that it is compatible with any model trained with gradient descent and applicable to a variety of different learning problems. Other meta-learning algorithms [2, 25, 30] continued to work on even better initialization or faster domain adaptation. Compared with the related work of meta-learning, we focus on adapting DP mechanism into the training process of meta-learning.

Federated learning emerged as a increasingly prevalent framework since Google researchers [15, 19] proposed to exploit data stored in users' smartphones or tablets for model training. Considering that meta-learning helps to boost performance with limited or defeated training data, meta-learning can be a natural fit for federated learning. In [6], Chen et al. were the first to propose FedMeta, a federated meta-learning algorithm to train a meta-learner with decentralized data. In [18], Lin et al. studied the robustness of federated meta-learning against adversarial samples. Our work shares the same vision with [6, 18] while our focus is to design privacy-preserving mechanism for federated meta-learning.

3 BACKGROUND

3.1 Model Agnostic Meta-Learning

First we introduce few-shot learning and meta-learning in general. A few-shot learning task can also be referred to as an "N-way-K-shot" problem, in which N is the number of classes and K is the number of training samples per class. Usually, K is a small number, e.g., 5, which is how "few-shot learning" comes to existence. Similar to deep learning, a meta-learning algorithm includes two stages: meta-training and meta-testing. Their corresponding datasets are meta-training set denoted by $\mathcal{D}_{train}$ and meta-testing set $\mathcal{D}_{test}$, respectively. Both of $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ consist of N-way-K-shot tasks. Assume that the i-th task of $\mathcal{D}_{train}$ is p_i while the j-th task of $\mathcal{D}_{test}$ is q_j. Each task in meta-learning has two datasets: support set and query set. For p_i, we denote its support set and query set

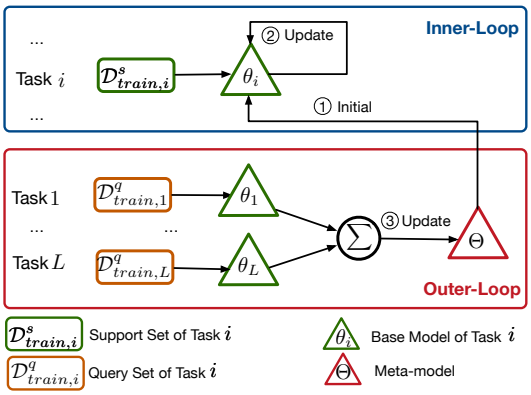


Figure 1: Overview of the training process in meta-learning

by $\mathcal{D}^s_{train,i}$ and $\mathcal{D}^q_{train,i}$, correspondingly. Similarly, for q_j, we can denote its support set and query set as $\mathcal{D}^s_{test,i}$ and $\mathcal{D}^q_{test,i}$.

The global model MAML tries to learn from all meta-training tasks is the meta-model, while the auxiliary model of each task is the base-model. As shown in Figure 1, meta-training on p_i starts by initializing p_i's model parameters as the current meta-model Θ, proceeds to train on the support set $\mathcal{D}^s_{train,i}$, and obtains a trained base-model θ_i. Next, it computes the gradients on the query set $\mathcal{D}^q_{train,i}$ using θ_i and updates Θ by applying the gradients to Θ. The final meta-model Θ^* is produced at the end of meta-training stage. For q_j at meta-testing stage, model parameters are initialized with Θ^*, then the model is trained on its support set $\mathcal{D}^s_{test,j}$ to obtain classification accuracy on its query set $\mathcal{D}^q_{test,j}$. The average accuracy over the whole $\mathcal{D}_{test}$ is used as the performance metric.

Formally, a base-model $f_\Theta(\boldsymbol{x})$ is defined as a neural network with model parameter Θ. For task p_i, $f_\Theta(\boldsymbol{x})$ is initialized as current meta-model Θ and then trained to θ_i with its support set $\mathcal{D}^s_{train,i}$ as follows:

$$\theta_i \leftarrow \Theta - \eta_1 \nabla_\Theta \frac{1}{|\mathcal{D}^s_{train,i}|} \sum_{(x,y)\in\mathcal{D}^s_{train,i}} \mathcal{L}_{p_i}(f_\Theta(x), y), \quad (1)$$

where η_1 is the learning rate, $\mathcal{L}_{p_i}(f_\Theta(x), y)$ is the loss of base-model, and θ_i is the model weights after training. For meta-model Θ, it is updated based on the gradients from all query sets of N_{train} tasks as follows:

$$\Theta \leftarrow \Theta - \eta_2 \nabla_\Theta \frac{1}{N_{train}} \sum_i \frac{1}{|\mathcal{D}^q_{train,i}|} \sum_{(x,y)\in\mathcal{D}^q_{train,i}} \mathcal{L}_{p_i}(f_{\theta_i}(x), y), \quad (2)$$

where η_2 is the learning rate for meta-model Θ, $f_{\theta_i}(x)$ is the trained base-model of task p_i, $\mathcal{L}_{p_i}(f_{\theta_i}(x), y)$ is the loss of $f_{\theta_i}(x)$ on a sample (x, y) of p_i's query set $\mathcal{D}^q_{train,i}$. Therefore, we can see that MAML uses loss of base-models of all tasks to update the meta-model, thus achieving learning across-task knowledge. In implementation, MAML has two loops. Equation (1) occurs in inner loop while Equation (2) in outer loop.

3.2 Differential Privacy

In this section, we will introduce the essential of differential privacy (DP): privacy, sensitivity, and privacy loss. Since inception[8], DP quickly becomes a popular privacy protection tool in different fields such as database, and machine learning. Its core idea is that under certain constraints, the probability of whether a specific data record exists in a dataset can be strictly bounded. The purpose of applying DP varies with the application scenarios. Here our purpose of bringing in DP is to guarantee that even though an attacker can obtain the plaintext meta-model, she[4] cannot decide whether a specific task or sample participates meta-training or not.

The definition of (ϵ, δ)-differential privacy can be expressed as follows.

Definition 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:*

$$\Pr[\mathcal{M}(d) \in S] \leq e^\epsilon \Pr[\mathcal{M}(d') \in S] + \delta. \quad (3)$$

d and d' are two adjacent databases meaning that these two differ in at most a single entry, i.e., $\|d - d'\|_1 \leq 1$.

We further elaborate the above definition here. Considering $\delta = 0$ and $\epsilon \rightarrow 0$, $\frac{\Pr[\mathcal{M}(d)\in S]}{\Pr[\mathcal{M}(d')\in S]} \leq e^\epsilon \approx 1$. Therefore, one cannot distinguish between d and d' by observing $\mathcal{M}(d)$ and $\mathcal{M}(d')$, which is why we call the method "differential" privacy. ϵ and δ have different implications. ϵ is the exponential growth rate of probability ratio $\frac{\Pr[\mathcal{M}(d)\in S]}{\Pr[\mathcal{M}(d')\in S]}$ while δ is the bias even when $\epsilon = 0$.

Second, sensitivity of a mechanism characterizes how the outputs change with different inputs. The formal definition goes as follows.

Definition 2 (Sensitivity). *The sensitivity of a mechanism $\mathcal{M}$, denoted by $\triangle s$, is the largest gap between the observed outputs of $\mathcal{M}$ on all adjacent databases:*

$$\triangle s = max_{d,d'} \|\mathcal{M}(d) - \mathcal{M}(d')\|_1. \quad (4)$$

Therefore, a higher sensitivity means that we need to spend more privacy budget (equivalently, add more noise) to hide the difference between two adjacent databases.

Finally, privacy loss provides a way to characterize privacy level of a mechanism. The definition is as follows.

Definition 3 (Privacy Loss). *For a pair of neighboring databases d and d', the privacy loss of a random mechanism $\mathcal{M}$ incurred by observing the output ξ is*

$$L_{\mathcal{M}}(d, d', \xi) \triangleq \frac{Pr[\mathcal{M}(d) = \xi]}{Pr[\mathcal{M}(d') = \xi]}. \quad (5)$$

4 SYSTEM AND ADVERSARY MODEL

4.1 System Model

Figure 2 illustrates the overview of DP-MAML, there are three actors.

Participants are users who contribute the meta-training process with their private data. Meta-training at a participant includes two steps. The contributor first initializes her neural network with the initial meta-model and trains her base-model with her support set.

[4]No gender discrimination

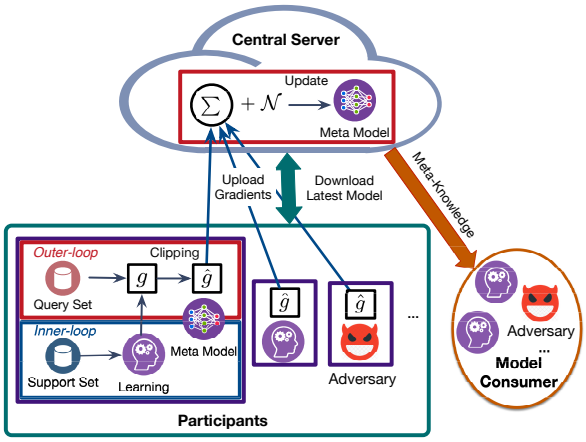


Figure 2: System model of federated meta-learning.

She then computes gradients using her query set and uploads them to the central server.

Central Server is usually at the cloud side. Its primary role in the meta-training process is aggregation. In the beginning, it randomly initializes the meta-model. In each meta-training epoch, it first downloads the current meta-model to a certain group of selected participants, collects gradients from them, and uses the collected gradients to update the meta-model. It outputs the final meta-model in the end of meta-training. It is responsible for distributing the meta-model after the training process.

Model Consumers are usually clients at remote sites. On receiving a meta-model from the central server, a model consumer uses it to initialize her own neural network and then train her own base-model using her dataset. Due to the benefits of federated meta-learning, she can perform training even though her dataset may be small or the categories in her task have not been seen in the meta-training process for the meta-model.

Data Transmission occurs once for each end user participant in the meta-training process. The central server transmits the meta-model to each participant and receives the update gradients from them. Let the number of participants be N_{train} and the central server selects L users to participant in one epoch. The total number of communication rounds is $\frac{N_{train}}{L}$.

4.2 Adversary Model

In this paper, we assume the central server can be trusted. In practice, the central server can reputable companies like Google or Apple We also assume participants are honest but curious, which has two implications. First, a participant follows the protocol strictly and does not launch active attacks, such as data poisoning attack [32, 37]. Second, a participant may try to infer whether a specific training task/sample contributes to her received meta-model. Finally, we assume that some model consumers are benign while some are malicious. Benign model consumers use the meta-model from the central server to improve their local models while malicious ones exploit the meta-model to infer whether a specific training task or sample contributes to it or not. Our adversary model is similar to others in the literature[11, 20, 21].

Symbol	Definition
$\mathcal{N}(0, \sigma^2 I)$	Gaussian noise with variance σ^2 and zero mean
p_i, q_i	training task i, testing task i
$\mathcal{D}^s_{train,i}$	support set of training task i
$\mathcal{D}^q_{train,i}$	query set of training task i
$\mathcal{D}^s_{test,i}$	support set of testing task i
$\mathcal{D}^q_{test,i}$	query set of testing task i
L, z, k	*Lot* size, Noise multiplier, and Clipping percentile
θ_i, Θ	base model and meta-model
$\|A\|$	cardinality of set A: number of elements of the set

Table 1: Symbols and notations

5 DIFFERENTIALLY PRIVATE FEDERATED META-LEARNING

This section presents our DP-MAML algorithm. We first describe the procedure to apply DP to deep learning system and then detail how we adapt it to task-level and sample-level DP-MAML.

5.1 Training with Differential Privacy

There are different ways to provide differential privacy in machine learning. Perturbations can be added to the final parameters of the model [29], objective function[39], or gradients during the training process[1, 20, 33]. We choose to add Gaussian noise to the gradients so that we can control the impact of each individual data on privacy loss during the training process. Our training method, based on MAML [9], is named as Differentially Private MAML (DP-MAML).

Based on the definitions in Section 3.2, we can see that deep learning model does not fit into DP framework naturally. In [1], Adabi et al. proposed a way to incorporate DP into training process of deep learning systems by extending the idea of "batch" to the concept of "*lot*" to represent the group size of data points used for one update. Accordingly, *lot* corresponds to a database $d \in D$, and gradient average of a *lot* corresponds to a mechanism. In general, the whole training process consists of a sequence of mechanisms, each corresponds one model/parameter update. Moments accountant, a composition method, is used to estimate the runtime privacy performance of iterative mechanisms. The end privacy guarantee is in the form of (ϵ, δ)-DP.

As mentioned above, composition theorems of DP are to bound the privacy loss of multiple random algorithms $\{\mathcal{M}_1, \mathcal{M}_2, \ldots\}$. In this paper, we adopt moments accountant in [1] as our composition theorem. Essentially, privacy loss $L_{\mathcal{M}}$ is a random variable depending on $\mathcal{M}$ and the noise added. Specifically for machine learning, privacy analysis are to estimate the (ϵ,δ) corresponding to the accumulated privacy loss during the training process. Training continues until depletion of the pre-defined budget. Existing work directly composed the privacy loss of all $\{\mathcal{M}_1, \mathcal{M}_2, \ldots\}$ together, which results in a loose estimation. Unlike them, moments accountant composes logarithmic moments of privacy loss, which turns out to provide a tighter estimation.

5.2 Task-level DP-MAML

In federated meta-learning, a task refers to the meta-training process with the data of an data contributor. Task-level DP protects

Algorithm 1 Task-Level DP-MAML

Input Tasks $\{\mathcal{T}_1, ..., \mathcal{T}_N\}$;
Parameters: Base Model Learning rate η_1, meta-learning rate η_2, noise scale z, clipping bound C, group size L.
Initialize meta-model φ.
while Not done **do**
  Randomly sample a subset tasks L;
  for $i \in L$ **do**
    $\theta_i \leftarrow \varphi$ # Base model initialization
    $\theta_i \leftarrow \theta_i - \eta_1 \nabla_{\theta_i} \mathcal{L}(\theta_i, x_i^s)$ # **Inner Loop**
    $g_i \leftarrow \nabla_\varphi \mathcal{L}(\theta_i, x_i^q)$ # **Gradient Compute**
    $\hat{g}_i \leftarrow g_i * \min(1, \frac{C}{\|g(x_i)\|})$ # **Gradients Clipping**
  end for
  # **Outer Loop**
  $\tilde{g} \leftarrow \frac{1}{L}\left(\sum_i \hat{g}_i + \mathcal{N}(0, z^2C^2\mathbf{I})\right)$ # **Noise Adding**
  $\varphi \leftarrow \varphi - \eta_2 \tilde{g}$ # **Meta Model Update**
  Update S with the median of gradients norm of L tasks.
end while
Output φ and differential privacy parameters (ϵ, δ)

the privacy of whether a specific task participates meta-training or not. Our motivations for task-level DP are as follows. First, tasks in federated meta-learning tend to be more unique in that they have different categories and statistics, resulting in more prominent privacy impact of a specific task on the meta-model. Second, task-level DP provides one more alternative to deploy deferentially private federated meta-learning systems. We divide the training procedure above (i.e., Section 5.1) into two steps: bounding sensitivity and adding noise. We will discuss them one by one in the followings, which are also summarized in **Algorithm 1**.

5.2.1 Bounding Sensitivity. As discussed in Section 3.2, the most critical step in applying DP is to obtain the sensitivity of a random mechanism. In our case, we need to determine the difference of computed gradients with or without an individual instance during training. In practice, corresponding sensitivity and perturbation can be prohibitively large. Furthermore, there is no prior information of the distribution of gradients either. One effective method to solve the problem is to clip gradients according to the l^2-norms of changing gradients such that the sensitivity is well bounded and the corresponding pertubations are practical. In DP-MAML, we propose a novel *adaptive* per-layer gradient clipping scheme to tackle the challenge.

Gradient clipping is frequently used to avoid gradients exploding in deep learning [14, 28]. In a deep neural network (DNN), we use the derivatives of the loss function on network parameters to update themselves in back propagation. These derivatives can be so excessively large that they lead to numerical issues on back propagation. This issue is also referred to as gradient exploding. Gradient clipping is used to bound these derivatives, thus dramatically decreasing the likelihood of gradient exploding. In this work, we use gradient clipping for different purpose, i.e., bounding the sensitivity of a learning model, apart from avoiding gradient exploding in DP-MAML.

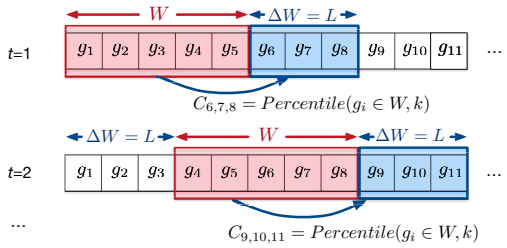


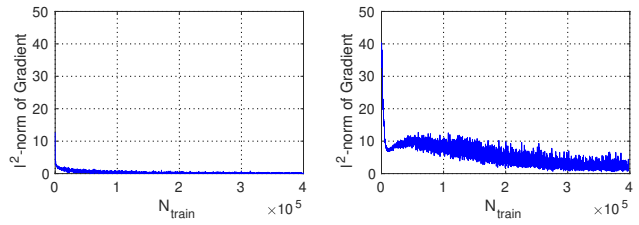
Figure 3: Illustration of adaptive clipping scheme.

Adaptive v.s. Non-Adaptive Clipping. We have two alternatives for applying gradient clipping: adaptive clipping and non-adaptive. For non-adaptive clipping method, we first meta-train the dataset without DP to obtain a distribution of all computed gradients. Next we choose C as the 50th percentile of the gradient distribution. Finally we meta-train the dataset applying DP with the selected C. For adaptive clipping method, we use a sliding window with size W and step size ΔW for help to get clipping threshold C, illustrated in Figure 3. For the current window (each instance is the l^2 norm of the gradient of one task), we can obtain its k-th percentile, i.e., $percentile(g_i \in W, k)$, of all l^2-norms as a clipping threshold C. Recall that DP-MAML uses the gradients of one *lot* for one meta-model update. Therefore, C of current window will be used to clip gradients in the next *lot*. Empirically, we set $W = 2L$ and $\Delta W = L$. We leave the exploration of better W and ΔW for future work.

Adaptive clipping is more preferable than non-adaptive one for two reasons. First, gradient norms tend to become smaller and then converge along training process (shown in Figure 4). C extracted by adaptive clipping method differs from one lot to another, thus offering better clipping than a fixed threshold by non-adaptive clipping method. Second, non-adaptive clipping requires one additional training process to decide a fixed clipping threshold while adaptive method extracts C in the runtime. Therefore, adaptive clipping method is computationally much more efficient than non-adaptive one. Another thing to point out is that unlike previous papers such as [1, 20] which provided only customized clipping threshold for their own, our adaptive clipping method is also applicable to other machine learning systems trained with DP.

One observation in our experiments is that gradient norms generate sharp spikes due to the additive perturbation in the training process. Figure 4 shows how l^2-norm of gradients of MAML and DP-MAML change during one training process. For a window with such a spike, the resulting C turns out to be unreasonably large. We solve this issue by updating C only when it get no larger than the previous update. As a result, the large clipping thresholds due to abnormal spikes can be mitigated. Notice that other strategies such as moving averaging may have different impacts, which can be explored in the future.

Impact of k. Here we briefly discuss the impact of k on privacy protection and model accuracy. Recall that the clipping threshold C in adaptive clipping method is determined by the kth percentile of l^2 norms of a sequence of gradients. DP-MAML applies Gaussian noise to the summation of gradients after clipping in the same *lot*



(a) MAML (b) DP-MAML

Figure 4: The l^2-norm of gradients of training tasks. Left: the result of non-private MAML, right: the result of DP-MAML.

and uses the perturbed summation to update the meta-model (See Algorithm 1). Therefore, we have

$$\tilde{g} = \sum_{i \in L} \hat{g}_i + \mathcal{N}(0, z^2 C^2 \mathbf{I}), \tag{6}$$

where $\tilde{g}$ is the perturbed graident summation, $\hat{g}_i$ is the i-th graident after clipping, $\mathcal{N}(0, z^2 C^2 \mathbf{I})$ is the additive Gaussian noise.

Note that the larger k, the larger C, which is the exact clipping threshold. Intuitively, a large k means that most of the gradients are preserved and thus is theoretically good for convergence speed and testing accuracy. On the contrary, a large k also results in large added Gaussian noise (considering its standard deviation being zC in Equation (6) and therefore sacrifices model accuracy to meet the requirement of targeted privacy budget. As a result, we expect that k cannot be too large or too small to tradeoff utility and privacy.

5.2.2 Adding Noise. The second step of applying DP is adding appropriate amount of noise such that the existence of one training task is covered in DP-MAML. In this paper, we would like to decide whether applying DP is practical for typical meta-learning scenarios.

We need to cover two parts in applying noise: what kind of noise to add and exactly where to add noise. For the former, we use Gaussian noise with zero mean and $\sigma^2 = z^2 C^2$ variance. Here z is referred to as noise multiplier, which is the ratio between σ and C, i.e., $z = \frac{\sigma}{C}$. σ is the standard deviation of noise, C is clipping threshold described above. For the latter, we apply the generated Gaussion noise to the summation of all clipped gradients of one *lot* of size L (See Algorithm 1).

Impact of z and L. Here we briefly discuss the impact of z and L on privacy protection and model accuracy.

Given a privacy budget (i.e., (ϵ, δ)-DP), z cannot be too small otherwise the training process ends quickly because it soon uses up the privacy budget. In this case, testing accuracy is usually low as the meta-training terminates long before convergence. On the contrary, z cannot be too large otherwise the added noise may completely destroy usefulness of gradients and therefore meta-training may obtain only limited knowledge (See Equation (6)). The minimum z is 1 in our experiments because moments accountant is based on the assumption that $z >= 1$.

Similarly, we can see from Equation (6) that if L is too small, so is $\sum_{i \in Lot} \hat{g}_i$, and therefore $\mathcal{N}(0, z^2 C^2 \mathbf{I})$ may overshadow $\sum_{i \in Lot} \hat{g}_i$ and dominate $\tilde{g}$, making $\tilde{g}$ valueless. Therefore, a larger L is preferred to counter against the added noise part. On the other side, a larger L leads to poorer privacy protection according to Equation (10). To sum up, we need to choose a proper L to achieve good tradeoff between model accuracy and privacy protection.

5.3 Sample-level DP-MAML

Our sample-level privacy shares the same design goal as other work [1, 27, 33, 41], that is, to protect the privacy whether a specific sample participates the training process or not. Compared to the above task-level DP-MAML, sample-level DP-MAML needs to bound the impact of an individual training sample rather than an individual training task. As described below, sample-level DP-MAML targets a (much) higher level of privacy protection, thus inequitably sacrifices model accuracy. In this work, we are exploring applying DP to meta-learning, of which most application scenarios are challenging few-shot learning problems. Intuitively, the cost of applying DP to these scenarios, e.g., 5-way 1-shot learning, will be high.

Here we briefly describe how sample-level DP-MAML works. Recall that for MAML, each task has one support set and one query set. Samples from support set are involved in inner loop while those from query set in the outer loop, depicted in Equation (1, 2). To achieve sample-level privacy, we need to apply perturbations to both support set and query set and therefore in both loops. Algorithm 2 shows how DP-MAML generates Gaussian noise and applies it to gradients to achieve sample-level privacy.

We further elaborate the necessity of adding perturbations in both inner and outer loop of meta-training for one task. As described above, meta-training for one task consists of two steps. The first step is to train a local model based on support set while the second to obtain gradients based on query set. Equivalently, we can treat these two steps as two cascaded mechanism, denoted by $\mathcal{M}_{inner}$ and $\mathcal{M}_{outer}$, correspondingly. To protect sample privacy of support set and query set, we need both $\mathcal{M}_{inner}$ $\mathcal{M}_{outer}$ to be differentially private. Our sample-level DP aims to protect sample privacy of both support set and query set. Thus we add perturbations to the gradients of both loops, illustrated in Algorithm 2.

5.4 Privacy Analysis and Performance Tradeoff

5.4.1 Moments Accountant. DP-MAML uses moments accountant for privacy analysis, which offers the state-of-the-art estimation. As mentioned above (in Section 5.1), privacy analysis is essentially a composition process of privacy loss of a sequence of mechanisms. In DP-MAML, a mechanism corresponds to one model update. Moments accountant method composes the logarithmic moments of privacy loss instead of the privacy loss itself. The outcome of moments accountant module is a (ϵ, δ) pair The estimated (ϵ, δ) by moments accountant is proved to be tighter than those from other methods such as the advanced composition theory [8] as the benefit of transferring to the logarithmic moments space [1]. In practical, we need to fix either a target ϵ or a target δ when mapping the privacy loss to a (ϵ, δ) pair. In general, the lower ϵ, the better privacy. So is δ. Meanwhile, (ϵ, δ)-DP and (ϵ', δ')-DP are comparable only if either $\epsilon = \epsilon'$ or $\delta = \delta'$. We further define *privacy budget* and *privacy result* of the same (ϵ, δ)-DP.

- `Privacy Budget`: Privacy budget corresponds to the maximum privacy loss that the training process can afford. We set privacy budget ahead of experiments.

Algorithm 2 Sample Level DP-MAML

Input $(x^s, y^s) \in \mathcal{D}^s$ $(x^q, y^q) \in \mathcal{D}^s$;
Parameters: Base Model Learning rate η_1, meta-learning rate η_2, noise scale of inner loop z_{inner}, clipping bound C_{inner}, task group size L.
for $i \in L$ **do**
  $\theta \leftarrow \varphi$ # Base model initialization
  for $(x^s_{i,j}, y^s_{i,j}) \in \mathcal{D}^s_i$ **do**
    $g_{i,j} = \nabla_\theta \mathcal{L}(\theta, x^s_{i,j})$ # Compute gradient
    $\hat{g}_{i,j} \leftarrow g_{i,j} * \min(1, \frac{C_{inner}}{\|g_{i,j}\|})$ # Clip gradient
  end for
  $\tilde{g}_i \leftarrow \frac{1}{|\mathcal{D}^s|}\left(\sum_j \hat{g}_{i,j} + \mathcal{N}(0, z^2 C^2_{inner}\mathbf{I})\right)$ # Add noise
  $\theta_i \leftarrow \theta_i - \eta_1 \tilde{g}_i$ # update base model
  for $(x^q_{i,j}, y^q_{i,j}) \in \mathcal{D}^q_i$ **do**
    $g^q_{i,j} = \nabla_\varphi \mathcal{L}(\theta, x^q_{i,j})$ # Compute gradient
    $\hat{g}^q_{i,j} \leftarrow g^q_{i,j} * \min(1, \frac{C_{outer}}{\|g(x^q_{i,j})\|})$ #Clip gradient
  end for
end for
$\tilde{g} \leftarrow \frac{1}{|\mathcal{D}^q|*L}\left(\sum_{ij} \hat{g}^q_{i,j} + \mathcal{N}(0, z^2 C^2_{outer}\mathbf{I})\right)$ # Add noise
$\varphi \leftarrow \varphi - \eta_2 \tilde{g}$ # Update meta-model
Output Meta model parameters φ

- `Privacy Result`: Privacy result corresponds to the true privacy loss, which is solely determined by the added noise. During the experiments, we can track either ϵ or δ while fixing the other.

In practice, we simply terminate the training process if it exhausts the pre-set privacy budget.

5.4.2 Performance Tradeoff. All machine learning systems training with DP need to balance between utility (i.e., model accuracy) and privacy (i.e., privacy loss). Intuitively, the larger perturbations (the larger z), the better privacy protection, however, the worse model accuracy. Deep learning systems are usually trained with iterative process, meaning that the training set is fed into training multiple times. During experiments, there seems to be a difference between applying DP to deep learning and to meta-learning. The overall guideline for applying DP is that each training iteration spends part of a pre-set privacy budget and training stops once it uses up the budget. For deep learning, each training sample contributes to the training process multiple times through epochs. For meta-learning, a training task is fed into the training process only once. Below we also discuss the implication of such a difference formally.

In [1], LEMMA 3 says that $\alpha_{\mathcal{M}}(\lambda)$, which is the λ-th moments of privacy loss, satisfies that

$$\alpha_{\mathcal{M}}(\lambda) \leq \frac{q^2\lambda(\lambda+1)}{(1-q)\sigma^2} + O(q^3\lambda^3/\sigma^3). \quad (7)$$

where $\mathcal{M}(d) = \sum_{i\in J} f(d_i) + \mathcal{N}(0, \sigma^2 I)$ is a mechanism, J is a sample from a database with size n, q is the sampling probability an arbitrary sample is chosen. The inequality holds if $\sigma \geq 1$ and $q \leq \frac{1}{16\sigma}$.

Note that the above inequality gives the upper bound of privacy loss of a mechanism. We treat each update of model parameters as one mechanism. After T updates, the resulting composition of

Figure 5: Samples of Ominiglot dataset, CIFAR-FS dataset and Mini-ImageNet dataset.

Dataset	$N_{class} \times N_{sample}$	Image size	Train:Val:Test
Omniglot	1623×20	28×28	71: 3:26
CIFAR-FS	100×600	32×32	64:16:20
Mini-ImageNet	100×600	84×84	64:16:20

Table 2: Datasets used for evaluation of DP-MAML.

privacy loss can be written as:

$$\alpha(\lambda) \leq \sum_i \alpha_{\mathcal{M}_i}(\lambda) \leq T * \left(\frac{q^2\lambda(\lambda+1)}{(1-q)\sigma^2} + O(q^3\lambda^3/\sigma^3)\right). \quad (8)$$

For deep learning, $q = \frac{L}{N_{train-data}}, T = N_{epoch} * \frac{N_{train-data}}{L}$ is the number of updates, where $N_{train-data}$ is the number of data point in training dataset and L is the number of data point in a *lot*. Therefore, the above equation can be rewritten as

$$\alpha(\lambda) \leq \frac{N_{epoch} * q\lambda(\lambda+1)}{(1-q)\sigma^2} + O(N_{epoch} * q^2\lambda^3/\sigma^3). \quad (9)$$

On the contrary, for meta-learning, $q = \frac{L}{N_{train}}, T = \frac{N_{train}}{L} = \frac{1}{q}$ is the number of updates, L denotes the number of tasks in one *lot* (i.e., used for one meta-model update), and N_{train} the total number of training tasks. Assume that no task is used more than once in meta-learning. Therefore, the inequality in Equation (8) can be rewritten as

$$\alpha(\lambda) \leq \frac{q\lambda(\lambda+1)}{(1-q)\sigma^2} + O(q^2\lambda^3/\sigma^3). \quad (10)$$

Equation (9) and Equation (10) have different implications. According to Equation (9), if we increase N_{epoch}, $\alpha_{\mathcal{M}}(\lambda)$ also increases meaning privacy protection is worse. Therefore, more training iterations for deep learning increase the privacy loss and therefore reduce privacy level of the trained model. On the contrary, according to Equation (10), if we generate more training tasks(i.e., increasing N_{train}), q gets smaller, and therefore $\alpha_{\mathcal{M}}(\lambda)$ actually decreases meaning privacy protection is better. The take-away message is that the larger meta-training set (i.e., a larger N_{train}), the higher privacy level of the trained meta-model.

6 PERFORMANCE EVALUATION

In this section, we evaluate the performance of DP-MAML. First we introduce the datasets and implementations for experiments. Then we show the results of our evaluations of DP-MAML including task-level DP-MAML, sample-level DP-MAML, and comparison between DP-MAML (meta-learning) and pre-training (transfer learning).

6.1 Datasets

We used three popular datasets for experiments including Omniglot [17], CIFAR-FS [4], and Mini-ImageNet [38]. Table 2 provides the main information of the three datasets such as number of classes, number of samples per class, image size, splitting ratio among meta-training set, validation set, and meta-testing set. Figure 5 shows examples from these datasets. From these examples we have the idea that samples in Omniglot are much more simple than those in CIFAR-FS and Mini-ImageNet. The testing accuracy in Secion 6.3.1 also confirms such a conjesture. There are 1,623 characters in Omniglot, each with 20 samples, and each sample was written by an individual person and processed as a 28×28 image in black and white. In Mini-ImageNet, there are 100 classes, each with 600 samples, and each sample is a colour image of size 84×84. Similarly, there are 100 classes in CIFAR-FS [4] (adopted from CIFAR-100 [16]), each with 600 samples, and each sample is a colour image of size 32×32. We used these three datasets because Omniglot and Mini-ImageNet are the two most popular ones for few-shot learning while the CIFAR series is a well-known benchmark.

Using Omniglot as an example, the evaluation process of N-way, K-shot learning is as follows. First, the whole dataset is divided into three separate parts: 1,150 classes for meta-training, 50 for validation, and 423 for meta-testing. We denote these three sets by $\mathcal{D}_{train}$, $\mathcal{D}_{valid}$, and $\mathcal{D}_{test}$, respectively. We would like to emphasize that classes for meta-training, validation, and meta-testing are totally separated. Second, we generate a new meta-training task by randomly selecting N classes among 1,150 classes of $\mathcal{D}_{train}$ and further randomly selecting K samples for each selected N classes. Similarly we can generate a new validation or meta-testing task. In our experiments, the number of meta-training tasks on Omniglot is fixed as $N_{train} = 400,000$, the number of validation tasks as $N_{valid} = 600$, and the number of meta-testing tasks as $N_{test} = 600$. Similarly, for Mini-ImageNet and CIFAR-FS, the whole dataset is divided into three separate parts: 64 classes for meta-training, 16 for validation, and 20 for meta-testing. The number of meta-training tasks on CIFAR-FS is fixed as $N_{train} = 400,000$, while that on Mini-ImageNet is $N_{train} = 100,000$ in accordance with [2]. The number of validation tasks is $N_{valid} = 600$ and the number of meta-testing tasks $N_{test} = 600$ for CIFAR-FS and Mini-ImageNet.

6.2 Implementation, Hyper Parameters, and Performance Metric

One advantage of MAML is that it can be readily applied to most current popular neural networks without much alteration. In DP-MAML, we use the same network architecture in [2, 9] to have fair performance comparison. Specifically, we use a VGGNet [35], which comprises four convolutional layers, each followed by a normalization layer and a fully-connected layer. The number of filters on each convolutional layer is slightly different depending on the dataset used, which is 64 for Omniglot and 48 for CIFAR-FS and Mini-ImageNet. The number of output neurons on the fully-connected layer equals to the K value of the few-shot learning under exploring. We use Adam optimizer with a default 0.01 learning rate for outer-loop training. The minimum learning rate of outer-loop for Omniglot is 0.001 with a cosine annealing schedule, while the learning rate for Mini-ImageNet and CIFAR-FS is fixed at 0.01. We use SGD optimizer with 0.1 learning rate for inner-loop training in all the experiments. The difference between inner-loop learning rate and outer-loop learning rate is due to the vastly-different numbers of training steps of the two loops. For adaptive clipping, we set $W = 2L, \Delta W = L$. Other parameters are similar with the training parameters of [2].

We implemented DP-MAML in PyTorch based on the source code of [2, 3]. We ran all our experiments on the same server equipped with a 3.3 GHz Intel Core i9-9820X CPU, three GeForce RTX 2080 Ti GPUs, and Ubuntu 18.04.3 LTS. The default number of epochs is $N_{epoch} = 100$. Each epoch consists of 1,000 training tasks for Mini-ImageNet and 4,000 training tasks for Omniglot and CIFAR-FS. Note that there is a difference between the definitions of "epoch" in meta-learning and deep learning. In deep learning, an epoch refers to the process in which the whole dataset undergoes training for one time. In meta-learning, we use epoch to serve as checkpoint during experiments. The whole meta-training set undergoes training only one time at the end of the 100th epoches. The training process may be terminated ahead of time if it uses up all the given privacy budget. We calculate the accumulated consumed privacy budget in runtime at the end of each epoch using the moments accountant method. Meanwhile, at the end of each epoch, the trained model is saved and validated on 600 fixed validation tasks to see its performance. At the end of meta-traing, we use the ensemble of the top-5 models for meta-testing on 600 fixed testing tasks.

In our experiments, we have the following settings if not specified otherwise. By default, the dataset used is Omniglot, the few-shot learning case is 5-way 1-shot, clipping percentile theshold k-th is 90th, noise multiplier z equals 1, and *lot* size L equals 1,600.

In our experiments, we mainly use the average testing accuracy on N_{test} tasks of $\mathcal{D}_{test}$ as the performance metric. Specifically, for each testing task, a base model initialized with meta-model is trained on its support set $\mathcal{D}^s_{test}$ and then obtains classification accuracy on query set $\mathcal{D}^q_{test}$. The average accuracy over the whole $\mathcal{D}_{test}$ is used as the performance metric.). In some case, we also compare the estimated (ϵ, δ) of trained models. Although we pre-set a target (ϵ, δ)-DP at the beginning, the estimated (ϵ, δ) pair from moments accountant still differs from one experiment to another. For example, we may pre-set $(1.5, 10^{-6})$ as the privacy budget, fix $\epsilon = 1.5$ for moments accountant, and track how δ changes. We terminate the training when either the number of epochs arrives at $N_{epoch} = 100$ or $\delta > 10^{-6}$. In either case, the final δ should be less or equal than 10^{-6} but may differ in different experiments. Our experiment results mostly fall into two cases. (1) Testing accuracy converges before $N_{epoch} = 100$ and the final (ϵ, δ) differs from one to another even under the same pre-set privacy budget. (2) Training process uses up the given privacy budget before $N_{epoch} = 100$ and thus is terminated (i.e., the final (ϵ, δ) equals the privacy budget). However, the training process may not converge yet in this case. The code is available online [5].

6.3 Results of Task-Level DP-MAML

The impact of DP in MAML is first shown, followed by our investigation on the three hyper parameters for the system: clipping threshold k, noise multiplier z, and *lot* size L.

[5] https://github.com/git-ccs20/DP-MAML

Dataset	N-way K-shot	MAML	DP-MAML	
		Acc	Acc	(ϵ, δ)-DP
Omniglot	5-way 1-shot	99.5%	93.9%	$(1.5, 10^{-6})$
	20-way 1-shot	96.4%	88.0%	$(1.5, 10^{-6})$
CIFAR-FS	5-way 1-shot	61.0%	48.1%	$(1.5, 10^{-6})$
	5-way 5-shot	78.6%	58.2%	$(1.5, 10^{-6})$
Mini-ImageNet	5-way 1-shot	51.9%	32.7%	$(1.5, 10^{-6})$
	5-way 5-shot	69.8%	45.8%	$(1.5, 10^{-6})$

Table 3: Performance of task-level DP-MAML

6.3.1 With v.s. Without DP. We first compare the testing accuracy of MAML trained with and without DP. For this set of experiments, we set the privacy budget as $(1.5, 10^{-6})$-DP. Compared to other machine learning systems trained with DP [1, 27, 33], $(1.5, 10^{-6})$-DP can be regarded as a relatively small and thus practical privacy budget, which in return casts greater challenge to achieve good accuracy. We ran experiments on the following N-way K-shot learning problems: 5-way 1-shot and 20-way 1-shot learning on Omniglot, 5-way 1-shot and 5-way 5-shot learning on both CIFAR-FS and Mini-ImageNet.

Table 3 summarizes the results of the above experiments. As we can see, applying DP leads to different accuracy tradeoff on the three datasets. For 5-way 1-shot learning, testing accuracy drops from 99.5% without DP to 93.9% with DP on Omniglot, from 61.0% to 48.1% on CIFAR-FS, and from 51.9% to 45.8% on Mini-ImageNet, respectively. We also observed similar trend of testing accuracy on other few-shot learning problems.

We have two conclusions from this set of experiments. For one thing, MAML achieves vastly-different testing accuracy on the three datasets, which in return leaves different amounts of space for adding privacy protection. Learning on Omniglot is relatively easy and MAML achieves nearly perfect classification performance. This gives a lot of space for performance tradeoff when applying DP and DP-MAML on Omniglot still offers nice classification accuracy. On the contrary, learning on sophisticated datasets like CIFAR-FS and Mini-ImageNet is always challenging. Applying DP to MAML on these datasets makes the tasks even more demanding, resulting in severe accuracy drop. DP-MAML is the first to explore the combination of meta-learning and DP. State-of-the-art meta-learning algorithms [23, 26] are also boosting performance on datasets like Mini-ImageNet. We believe that DP-MAML provides useful guidelines for future work on incorporating DP to advanced meta-learning systems. For the other thing, as mentioned above, current privacy budget used here, $(1.5, 10^{-6})$-DP, is relatively small and thus imposes great challenge to achieving high testing accuracy. Compared with related work [1, 27, 33], we believe DP-MAML achieves better tradeoff between utility/accuracy and privacy and has the potential for further improvement with more advanced meta-learning algorithms.

6.3.2 Clipping Threshold. Recall that we have two clipping methods: adaptive clipping and non-adaptive one. The main difference is that C changes from one *lot* to another with adaptive clipping while the same C is used for the whole training process with non-adaptive clipping. In Section 5.2.1 we conclude that k, i.e., the percentile to extract C, cannot be too large or too small to tradeoff utility and privacy.

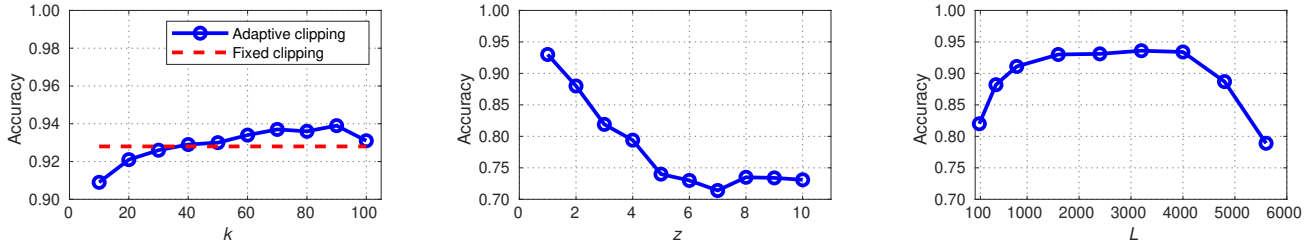
Figure 6(a) shows testing accuracy of DP-MAML for 5-way 1-shot learning on Omniglot with adpative and non-adaptive clipping. We evaluated different k ranging from 10 to 90 corresponding to 10-th percentile to 90-th percentile. Obviously, adaptive clipping method outperforms non-adaptive one when k is chosen properly. Our adaptive clipping method has the potential to choose a more proper C for each *lot* and therefore achieves better accuracy than non-adaptive one. Another observation is that k cannot be too small nor too large to have optimal classification accuracy. For example, we can see from the figure that when k is set as 90-th percentile, DP-MAML achieves the highest testing accuracy for 5-way 1-shot on Omniglot.

6.3.3 Noise Multiplier. In Section 5.2.2, we conclude that z cannot be either too small due to pre-set privacy budget or too large due to model accuracy concern. Figure 6(b) shows testing accuracy of DP-MAML for 5-way 1-shot learning on Omniglot when z increases from 1 to 10. We can see that testing accuracy drops from 93.0% when $z = 1$ with the increase of z and tends to be stable at around 73%. We would like to emphasize that each result in Figure 6(b) probably corresponds to a different (ϵ, δ) pair as explained in Section 5.4. The larger z, the smaller ϵ or δ, meaning better privacy protection. We can also confirm this finding from Figure 7 which shows how testing accuracy changes with the number of epochs when $z = 1, 2, 3$. We can see that the achieved (ϵ, δ) pairs are different. To estimate change of δ with epochs, we need to fix ϵ for each experiment. For $z = 1$, we fix $\epsilon = 1.5$, for $z = 2$, $\epsilon = 0.34$ and for $z = 3$, $\epsilon = 0.23$. Figure 7 shows that the final δ values are around 10^{-6}. The corresponding testing accuracy when $\delta = 10^{-6}$ is around 91% for $z = 1$, 85% for $z = 2$, and 75% for $z = 3$. Therefore, the larger z, the smaller ϵ (i.e., the better privacy protection), which is as expected.

Our another finding from Figure 9 is that a larger z leads to slower convergence speed. Figure 9 depicts how the training loss on query set changes with the number of epochs. From the figure, we can see that DP-MAML takes round 39 iterations to converge when $z = 1$, 58 when $z = 2$, and 83 when $z = 3$. This can be explained by that larger added noise limits the amount of useful knowledge learnt in each epoch (See Equation 6), making the training slower to converge.

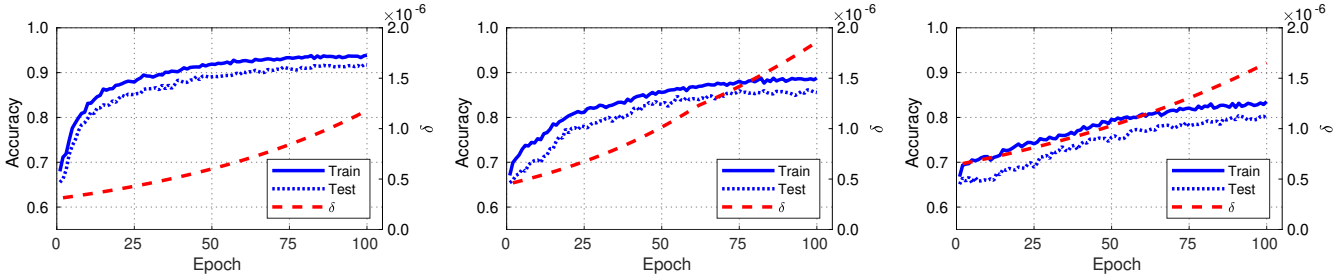
6.3.4 Lot Size. In Section 5.2.2, we conclude that L cannot be either too small or too large to achieve better tradeoff between privacy and accuracy. Figure 6(c) shows testing accuracy of DP-MAML for 5-way 1-shot learning on Omniglot when L ranges from 100 to 5,600. We can see that L has a great influence on testing accuracy. For this set of experiments, DP-MAML achieves the best accuracy when L is about 3,000. Testing accuracy drops quickly when L is smaller than 1,000 or larger than 4,000. In [1], the authors concluded that L should be around the square root of the number of training samples. However, we find that the optimal L also strongly depends on z, i.e., noise multiplier.

We demonstrate the above finding by Figure 10, in which we shows the testing accuracy of different L and z. For $z = 1$, accuracy peaks at $L = 1,600$ and drops quickly as $L \geq 4,800$ because of draining the privacy budget. For $z = 2$, accuracy peaks at $L = 3,200$



(a) Different clipping percentile (b) Different noise multiplier (c) Different lot size

Figure 6: Test accuracy of 5-way 1-shot learning on Ominiglot. The default setting: clipping percentile $k = 90$, *lot* size $l = 1600$, and noise multiplier $z = 1$. (a) varies k; (b): varies the L. (c) varies z. The privacy budget for all combination is $(2, 10^{-6})$-DP.



(a) z=1 (b) z=2 (c) z=3

Figure 7: Left axis: Accuracy of 5-way 1-shot learning on Ominiglot with different noise multiplier $z = 1, 2, 3$. Right-axis: privacy performance are shown as varying δ with fixed $\epsilon = 1.5, 0.34$, or 0.23 respectively.

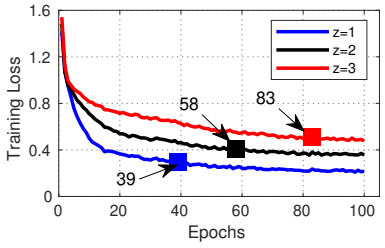


Figure 9: Training loss with different noise multiplier $z = 1, 2, 3$.

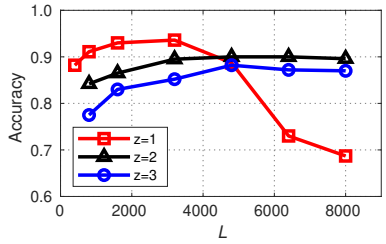


Figure 10: Accuracy with different lot size and different noise multiplier.

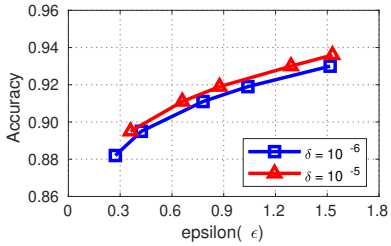


Figure 11: Accuracy with different privacy budget.

and stay stable with larger L. For $z = 3$, accuracy peaks at $L = 4,800$ and stay 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.

6.3.5 Guideline for k, z, and L. From above results, we can see that k, z, and L cannot be too large nor too small to achieve the best testing accuracy under the given privacy budget. Here we share our general guideline to work with DP-MAML. First, we recommend to select an appropriate noise multiplier z given a desired privacy budget since it is the most critical factor to impact privacy. For example, the change of ϵ is usually one order from $z = 1$ to $z = 2$. Second, the *lot* size need to satisfy $q = \frac{L}{N_{train}} < \frac{1}{16*z}$ due to the requirement of moments accountant. Third, we recommend to start with a relatively large L (e.g. close to $\frac{N_{train}}{16*z}$ especially when z is large. To get better privacy, decrease L slowly before accuracy drops significantly. One thing to note is that the learning rate should also increase with L for computation time concern. Privacy loss is determined when z and L are fixed. Finally, adjust k, W, and ΔW to boost the model accuracy.

6.3.6 Privacy-Utility Tradeoff. Here we present the results of trade-off between privacy protection and model accuracy. Figure 7 shows training accuracy, testing accuracy, and δ of DP-MAML for 5-way 1-shot learning on Omniglot change with the number of epochs when $z = 1, 2, 3$. We can see that all of them increase when the number of epochs increases with different noise multipliers. Recall that we need to fix ϵ (δ) in order to obtain the corresponding δ (ϵ) using moments accountant. For this set of experiments, we fixed $\epsilon = 1.5$, ran the experiments, and recorded the results. δ can be

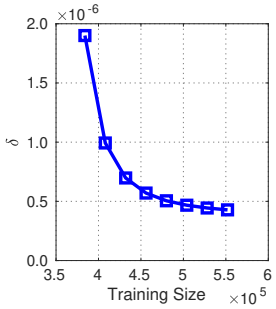


Figure 12: Privacy performance of DP-MAML with different training size.

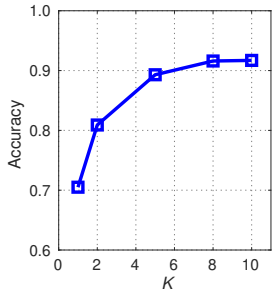


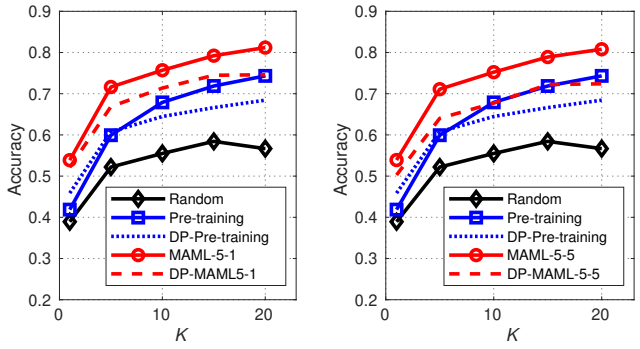
Figure 13: Accuracy of sample-level DP-MAML for 5-way K-shot learning.

regarded as the probability that ϵ-DP is broken, i.e., the larger δ, the worse privacy protection. Therefore, more training iterations improve model utility while reducing privacy protection. The decrease of privacy level is consistent with composition theorem of DP, which states that additional differentially private mechanism always lowers the privacy level.

Figure 12 shows that δ decreases with the number of total training tasks (i.e., N_{train}) for 5-way 1-shot learning on Omniglot, meaning the final privacy level improves with N_{train}. In order to differentiate it from Figure 7, the δ value in Figure 7 shows privacy performance changes during the training process. However, each point in Figure 12 represents the final privacy (ϵ, δ) of independent training process with different training size (N_{train} tasks ranges from 380,000 to 560,000.). For the experiments in this figure, all settings remain the same except that different N_{train}s are used. Note that a larger N_{train} corresponds to a larger meta-training set. The intuition behind the results of Figure 12 is that it is easier to protect the individual privacy when the database contains a larger number of instances. Thus, the results are consistent with what we mention in Section 5.4 that a larger N_{train} thus a smaller sampling rate ($q = \frac{L}{N_{train}}$) improves privacy protection. Therefore, DP-MAML has the benefit of enhancing privacy by increasing the number of meta-training tasks.

6.4 Results of Sample-Level DP-MAML

Here we present the testing accuracy of DP-MAML with sample-level DP. For this set of experiments, $k = 90, z = 1, L = 10$. L is smaller compared to 1,600 for task-level DP-MAML for two reasons. First, we find that increasing L does not benefit testing accuracy much. This may be due to that a larger L is only helpful for outer loop while the main limitation comes from the inner loop graident update. Second, computation cost is too huge with large L due to large scale of per-sample gradient calculation. Figure 13 shows the results for 5-way K-shot ($K \in \{1, 2, 5, 8, 10\}$) on Omniglot with $(1.18, 10^{-5})$-DP. We can see that testing accuracy increases with the number of shots and it reaches 92% when $K = 8$. Therefore, sample-level DP-MAML achieves good model performance only when the number of training sample per class is slightly larger. We do not show the results of sample-level DP-MAML on CIFAR-FS or Mini-ImageNet because we cannot find a good tradeoff between accuracy and privacy on these dataset. We believe that more advanced meta-learning and neuron networks give better performance on such dataset, therefore offering more room for privacy budget.



(a) 5-way 1-shot Initial (b) 5-way 5-shot Initial

Figure 13: The accuracy of five initialization methods on 5-way 1/5/10/15/20-shot learning on CIFAR-FS.

Here we further explain when it is much harder to achieve good results with sample-level DP-MAML. Recall that we add Gaussian noise to the summation of gradients after clipping in one *lot* to achieve task-level DP. In those experiments, the number of tasks in one *lot* is usually large, e.g., $L = 1,600$. To achieve sample-level DP, we always need to add Gaussian noise to the summation of gradients in the inner loop for a single task (including $N*K$ samples for a N-way K-shot learning), therefore easily resulting in much less meaningful gradients for base-model update. Consequently, meta-learner cannot extract useful knowledge from the heavily perturbed base-models. The performance of sample-level DP-MAML may be improved if the number of samples within one task is sufficiently large. Since we focus on meta-learning and few-shot learning in this paper, we leave it to future research to explore this direction.

6.5 DP-MAML v.s. Pre-Training

In this part, we mainly explore whether applying DP to meta-learning disqualifies its suitability for few-shot learning. Specifically, we would like to see whether DP-MAML can outperform other few-shot learning alternatives such as pre-training. We use the following setting for this set of experiments.

We experimented on CIFAR-FS rather than other datasets as CIFAR-FS is frequently used for general machine learning research. The targeted testing scenarios were 5-way K-shot, $K \in \{1, 5, 10, 15, 20\}$. Recall that CIFAR-FS has 100 classes and each class has 600 samples. We first separated CIFAR-FS into two non-overlapping sets: training set Φ consisting of 80 classes and testing set Ψ consisting of the remaining 20 classes. We used VGGNet as the base neural network and Adam optimizer with 0.004 learning rate. The output of learning on Φ is a set of initialization parameters, which are readily used to initialize a brand-new neuron network of an unseen task. Note that each task, no matter for meta-training or meta-testing, has two datasets: support set for training and query set for testing. Given an initialization (i.e., the output of learning on training set Φ), we first generated 600 unseen tasks from Ψ. For each of the 600 tasks, we applied the initialization to its neuron

network, trained 30 epoches on its support set, and used the derived model to obtain testing accuracy on its corresponding query set. The average testing accuracy on all 600 tasks is used as the ultimate evaluation metric. We experimented five initialization methods in total for this set of experiments, described as follows.

- `Random Initialization`: For this method, Φ is not used. Given each one of 600 tasks in Ψ, its neuron network is randomly initialized, trained, and then evaluated.
- `Pre-Training`: We applied a pre-training method that shares the same spirit as [7], which is similar to transfer learning. Specifically, we trained directly on the whole Φ for 80-class learning and derived a model which was originally for 80-class classification. After training, the parameters of the pre-training model except the last layer (i.e., a fully-connected layer with 80 output neurons) are used for initialization.
- `DP-Pre-Training` We applied DP to the above mentioned Pre-Training approach with the method in [1]. Specifically, we trained the model with $z = 6, L = 2000, N_{epoch} = 100$ used by [1] on CIFAR-10 obtaining $(1.73, 10^{-5})$-DP .
- `MAML`: We used MAML to train for 5-way 1-shot or 5-way 5-shot learning on Φ without applying DP. The obtained meta-model is used to initialize testing tasks on Ψ.
- `DP-MAML`: We used task-level DP-MAML to train for 5-way 1-shot or 5-way 5-shot model on Φ. For this experiment, $k = 90, z = 1, L = 1600$, resulting $(1.5, 10^{-6})$-DP. The obtained meta-model is used to initialize testing tasks on Ψ.

Figure 13 shows the results of different initialization methods on CIFAR-FS. The difference between Figure 13(a) and Figure 13(b) is that we trained 5-way 1-shot meta-model in Figure 13(a) while 5-way 5-shot in Figure 13(b). As can be seen, MAML achieves the best testing accuracy among these methods. The performance of 5-way 5-shot MAML is almost the same as that of 5-way 1-shot. On the other hand, the performance of 5-way 5-shot DP-MAML is slightly lower than that of 5-way 1-shot DP-MAML. We point out that training based 5-way 1-shot initialization usually has the benefit of faster convergence than training based on 5-way 5-shot initialization. In most cases, DP-MAML still outperforms pre-training and DP-pre-training. Together with the results of MAML, we can conclude that meta-learning offers better initialization for unseen tasks, especially when K is small. When K is large enough, pre-training achieves similar or even slightly better accuracy than DP-MAML. Random initialization is without doubt the worst among the four methods. Experimental results here suggest that DP-MAML provides good initialization for machine learning systems and also differential privacy protection for training dataset.

6.6 Computation Time

Here we present the results of computation cost in DP-MAML. In [1, 12], the authors mentioned that per-example gradient computation was time-consuming compared to batch computation, slowing down the training process of deep learning with DP including DP-MAML. On the contrary, increasing L in DP-MAML reduces the round of parameter update, resulting in time-saving than MAML. Therefore, it is unclear that which of MAML and DP-MAML is more time-consuming for training. We see from Table 4 that in general the computation time of DP-MAML is only slightly longer than that of MAML. Specifically, per-task running time of task-level DP-MAML is around 54.7 ms for 5-way 1-shot learning on Omniglot dataset, 14.8 ms slower than 39.9 ms of MAML. The training time of a typical meta-learning process (e.g., with 400,000 tasks) is around 6 hours. The same thing happens on CIFAR-FS while per-task running time of DP-MAML on Mini-ImageNet is even smaller than that of MAML. Therefore, we believe DP-MAML is affordable for a wide range of learning applications.

Dataset	MAML		DP-MAML	
	batch size	time ($/task$)	batch size	time ($/task$)
Omniglot	8	39.9ms	1	54.7ms
CIFAR-FS	8	68.7ms	1	81.3ms
Mini-ImageNet	2	112.3ms	1	102.7ms

Table 4: Computation time of task-level DP-MAML

7 LIMITATIONS AND FUTURE WORK

DP-MAML faces the issue of poor accuracy on challenging datasets such as CIFAR-FS and Mini-ImageNet. We mentioned in Section 6.4 that there were two main reasons. The first is CIFAR-FS and Mini-ImageNet are challenging even for non-DP meta-learning (See Table 3) and the second is that sample-level DP adds noise to inner loop gradient update. We foresee that a more advanced learning model or a more powerful composition theory for privacy analysis may help to tackle the issue. Tadam in [26] achieved 58.5% accuracy on 5-way 1-shot learning with Mini-ImageNet, which is potential to applying DP and boost results on challenging datasets.

In [19], the authors put forward several key issues of federated learning such as non-iid data and limited communication connectivity. We conjecture that the two challenges may not be critical for federated meta-learning thanks to the nature of meta-learning. Non-iid data from participants can actually boost the performance of meta-learning since it aims to extract only common knowledge from tasks. Federated learning has a bigger issue on communication cost due to iterative update between the central server and each participant in each training epoch. DP-MAML does not have this issue because all tasks (participants) are required to report their gradients only once in the training process. However, DP-MAML needs to train the tasks sequentially, which still leaves room to improve training efficiency. We leave it to our future work.

In practice, parameter-tuning may bring about privacy leakage due to the interactive training process between the central server and participants if the end-users are not trusted. As such, we have outlined the details of our experience in how the parameter can be tune in few iterations in Sec. 6.3.5.

8 CONCLUSION

In this paper, we proposed DP-MAML as a federated meta-learning algorithm to guarantee a trained meta-model is differentially private. We motivated and explored task-level DP and sample-level DP. We thoroughly evaluated DP-MAML under two DP levels on three well-known datasets, i.e., Omniglot, CIFAR-FS, and Mini-ImageNet, under different factors such as gradient clipping threshold, noise multiplier and lot size. We also showed that DP-MAML outperformed pre-training when used to initialize new tasks.

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] Antreas Antoniou, Harrison Edwards, and Amos Storkey. 2019. How to train your MAML. In *International Conference on Learning Representations (ICLR 19)*.

[3] Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. 2019. Differential privacy has disparate impact on model accuracy. In *Advances in Neural Information Processing Systems (NeurIPS 19)*. 15453–15462.

[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] Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*. Springer, 635–658.

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

[7] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning (ICML 14)*. 647–655.

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

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 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] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS 15)*. ACM, 1322–1333.

[11] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* (2017).

[12] Ian Goodfellow. 2015. Efficient per-example gradient computations. *arXiv preprint arXiv:1510.01799* (2015).

[13] Bargav Jayaraman and David Evans. 2019. Evaluating Differentially Private Machine Learning in Practice. In *28th USENIX Security Symposium (USENIX Security 19). Santa Clara, CA: USENIX Association.*

[14] Sekitoshi Kanai, Yasuhiro Fujiwara, and Sotetsu Iwamura. 2017. Preventing gradient explosions in gated recurrent units. In *Advances in neural information processing systems (NeurIPS 17)*. 435–444.

[15] 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).

[16] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning multiple layers of features from tiny images*. Technical Report. Citeseer.

[17] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science* 350, 6266 (2015), 1332–1338.

[18] Sen Lin, Guang Yang, and Junshan Zhang. 2020. A Collaborative Learning Framework via Federated Meta-Learning. *arXiv preprint arXiv:2001.03229* (2020).

[19] 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.

[20] 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)*.

[21] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 691–706.

[22] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 263–275.

[23] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. 2018. Rapid Adaptation with Conditionally Shifted Neurons. In *International Conference on Machine Learning (ICML 18)*. 3664–3673.

[24] Milad Nasr, Reza Shokri, and Amir Houmansadr. 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)*. IEEE, 739–753.

[25] Alex Nichol and John Schulman. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999* 2 (2018), 2.

[26] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. 2018. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS 18)*. 721–731.

[27] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2017. Semi-supervised knowledge transfer for deep learning from private training data. In *International Conference on Learning Representations (ICLR 17)*.

[28] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning (ICML 13)*. 1310–1318.

[29] Manas Pathak, Shantanu Rane, and Bhiksha Raj. 2010. Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Information Processing Systems(NeurPIS 10)*. 1876–1884.

[30] Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR 17)*.

[31] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2019. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Network and Distributed Systems Security Symposium (NDSS 19)*.

[32] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*. 6103–6113.

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

[34] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[35] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[36] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing (GlobalIPS 13)*. IEEE, 245–248.

[37] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. 2017. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems (NeurIPS)*. 3517–3529.

[38] 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.

[39] Yue Wang, Cheng Si, and Xintao Wu. 2015. Regression model fitting under differential privacy and model inversion attack. In *24th International Joint Conference on Artificial Intelligence (IJCAI 15)*.

[40] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 268–282.

[41] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. 2019. Differentially private model publishing for deep learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 332–349.