Differentially-private Federated Neural Architecture Search

Ishika Singh[*†]
University of California San Diego

Haoyi Zhou[*†]
Rutgers University

Kunlin Yang
University of California San Diego

Meng Ding[†]
University of California San Diego

Bill Lin
University of California San Diego

Pengtao Xie PENGTAOXIE2008@GMAIL.COM
University of California San Diego

Abstract

Neural architecture search, which aims to automatically search for architectures (e.g., convolution, max pooling) of neural networks that maximize validation performance, has achieved remarkable progress recently. In many application scenarios, several parties would like to collaboratively search for a shared neural architecture by leveraging data from all parties. However, due to privacy concerns, no party wants its data to be seen by other parties. To address this problem, we propose federated neural architecture search (FNAS), where different parties collectively search for a differentiable architecture by exchanging gradients of architecture variables without exposing their data to other parties. To further preserve privacy, we study differentially-private FNAS (DP-FNAS), which adds random noise to the gradients of architecture variables. We provide theoretical guarantees of DP-FNAS in achieving differential privacy. Experiments show that DP-FNAS can search highly-performant neural architectures while protecting the privacy of individual parties. The code is available at https://github.com/UCSD-AI4H/DP-FNAS

1. Introduction

In many application scenarios, the data owner would like to train machine learning (ML) models using their data that contains sensitive information, but the size of the data is limited. Many ML methods, especially deep learning methods, are data hungry. Having more data for model training usually improves performance. One way to have more training data is to combine data of the same kind from multiple parties and use the combined data to collectively train a model. However, since each of these datasets contains private information, they are not allowed to share across parties. Federated learning (Konečný

. *Equal contribution

. †The work was done during internship at UCSD.

et al., 2016; McMahan et al., 2016) is developed to address this problem. Multiple parties collectively train a shared model in a decentralized way by exchanging sufficient statistics (e.g., gradients) without exposing the data of one party to another.

While preserving privacy by avoiding sharing data among different parties, federated learning (FL) incurs difficulty for model design. When ML experts design the model architecture, they need to thoroughly analyze the properties of data to obtain insights that are crucial in determining which architecture to use. In an FL setting, an expert from one party can only see the data from this party and is not able to analyze the data from other parties. Without having a global picture of all data from different parties, ML experts are not well-equipped to design a model architecture that is optimal for fulfilling the predictive tasks in all parties. To address this problem, we resort to automated neural architecture search (Zoph and Le, 2016; Liu et al., 2018; Real et al., 2019) by designing search algorithms to automatically find out the optimal architecture that yields the best performance on the validation datasets.

To this end, we study federated neural architecture search (FNAS), where multiple parties collaboratively search for an optimal neural architecture without exchanging sensitive data with each other for the sake of preserving privacy. For computational efficiency, we adopt a differentiable search strategy (Liu et al., 2018). The search space is overparameterized by a large set of candidate operations (e.g., convolution, max pooling) applied to intermediate representations (e.g., feature maps in CNN). Each operation is associated with an architecture variable indicating how important this operation is. The prediction loss is a continuous function w.r.t the architecture variables A as well as the weight parameters W in individual operations. A and W are learned by minimizing the validation loss using a gradient descent algorithm. After learning, operations with top-K largest architecture-variables are retained to form the final architecture.

In FNAS, a server maintains the global state of A and W. Each party has a local copy of A and W. In each iteration of the search algorithm, each party calculates gradient updates of A and W based on its local data and local parameter copy, then sends the gradients to the server. The server aggregates the gradients received from different parties, performs a gradient descent update of the global state of A and W, and sends the updated parameters back to each party, which replaces its local copy with the newly received global parameters. This procedure iterates until convergence.

Avoiding exposing data is not sufficient for privacy preservation. Several studies (Bhowmick et al., 2018; Carlini et al., 2018; Fredrikson et al., 2015) have shown that intermediates results such as gradients can reveal private information. To address this problem, we study differentially-private FNAS (DP-FNAS), which adds random noise to the gradients calculated by each party to retain differential privacy (Dwork et al., 2006). We provide theoretical guarantees of DP-FNAS in privacy preservation. Experiments demonstrate that while protecting the privacy of individual parties, the architectures searched by DP-FNAS can achieve high accuracy that is comparable to those searched by single-party NAS.

The major contributions of this paper are as follows:

- We propose differentially-private federated learning for neural architecture search (DP-FNAS), which enables multiple parties to collaboratively search for a highly-performant neural architecture without sacrificing privacy.

- We propose a DP-FNAS algorithm which uses a parameter server framework and a gradient-based method to perform federated search of neural architectures. The gradient is obfuscated with random noise to achieve differential privacy.
- We provide a theoretical guarantee of our algorithm in terms of privacy preservation.
- We perform experiments which show that DP-FNAS can search highly-performant neural architectures while protecting the privacy of individual parties..

The rest of the papers are organized as follows. Section 2 and 3 present the method and experiments. Section 4 reviews related works and Section 5 concludes the paper.

2. Methods

We assume there are K parties aiming to solve the same predictive task, e.g., predicting whether a patient has pneumonia based on his or her chest X-ray image. Each party k has a labeled dataset D_k containing pairs of input data example and its label. For instance, the data example could be a chest X-ray and the label is about whether the patient has pneumonia. The datasets contain sensitive information where privacy needs to be strongly protected. Therefore, the K parities cannot share their datasets with each other. One naive approach is: each party trains a model using its own data. However, deep learning methods are data hungry: more training data usually leads to better predictive performance. It is preferable to leverage all datasets from different parties to collectively train a model, which presumably has better predictive performance than the individual models belonging-to different parties, each trained on a party-specific dataset. How can one achieve this goal without sharing data between parties?

Federated learning (FL) (Konečnỳ et al., 2016; McMahan et al., 2016) is a learning paradigm designed to address this challenge. In FL, different parties collectively train a model by exchanging sufficient statistics (e.g., gradient) calculated from their datasets, instead of exchanging the original data directly. There is a server maintaining the weight parameters of the global model to be trained. Each party has a local copy of the model. In each iteration of the training algorithm, each party k uses its data D_k and the local model M_k to calculate a gradient G_k of the predictive loss function $L(M_k, D_k)$ with respect to M_k. Then it sends G_k to the server. The server aggregates the gradients $\{G_k\}_{k=1}^{K}$ received from different workers and performs a gradient descent update of the global model: $M \leftarrow M - \eta\frac{1}{K}\sum_{k=1}^{M} G_k$, where η is the learning rate. Then it sends the updated global model back to each party, which replaces its local model with the global one. This procedure iterates until convergence. In this process, the dataset of each party is not exposed to any other party or the server. Hence its privacy can be protected to some extent (later, we will discuss a stronger way of protecting privacy).

Though FL provides a nice way of effectively using more data for model training while preserving privacy, it poses some difficulties on how to design the model architecture. For ML experts, to design an effective model architecture, the experts need to thoroughly analyze the properties of the data. In an FL setting, the expert from each party can only see the data from this party, not that from others. Without a global picture of all datasets, these experts are not well-equipped to design an architecture that is optimal for the tasks in all parties.

To address this problem, we resort to automatic neural architecture search (NAS) (Zoph and Le, 2016; Liu et al., 2018; Real et al., 2019). Given a predictive task and labeled data, NAS aims to automatically search for the optimal neural architecture that can best fulfill the targeted task. The problem can be formulated in the following way:

$$\begin{array}{ll} \min_A & L(D^{(\mathrm{val})}, A, W^*(A)) \\ s.t. & W^*(A) = \mathrm{argmin}_W \;\; L(D^{(\mathrm{tr})}, A, W) \end{array} \tag{1}$$

where $D^{(\mathrm{tr})}$ and $D^{(\mathrm{val})}$ are the training data and validation data respectively. A denotes the neural architecture and W denotes the weights of the model whose architecture is A. Given a configuration A of the architecture, we train it on the training data and obtain the best weights $W^*(A)$. Then we measure the loss $L(D^{(\mathrm{val})}, A, W^*(A))$ of the trained model on the validation set. The goal of an NAS algorithm is to identify the best A that yields the lowest validation loss. Existing search algorithms are mostly based on reinforcement learning (Zoph and Le, 2016), evolutionary algorithm (Real et al., 2019), and differentiable NAS (Liu et al., 2018). In this work, we focus on differentiable NAS since it is computationally efficient.

To this end, we introduce federated neural architecture search (FNAS), which aims to leverage the datasets from different parties to collectively learn a neural architecture that can optimally perform the predictive task, without sharing privacy-sensitive data between these parties. The FNAS problem can be formulated as:

$$\begin{array}{ll} \min_A & \sum_{k=1}^{K} L(D_k^{(\mathrm{val})}, A, W^*(A)) \\ s.t. & W^*(A) = \mathrm{argmin}_W \;\; \sum_{k=1}^{K} L(D_k^{(\mathrm{tr})}, A, W) \end{array} \tag{2}$$

where $D_k^{(\mathrm{tr})}$ and $D_k^{(\mathrm{val})}$ denote the training and validation dataset belonging to the party k respectively. A naive algorithm for FNAS performs the following steps iteratively: given a configuration A of the architecture, use the gradient-based FL method to learn the optimal weights $W^*(A)$ on the training data; then evaluate $W^*(A)$ on the validation data of each party and aggregate the evaluation results. The validation performance is used to select the best architecture. Certainly, this is not efficient or scalable. We resort to a differentiable search approach (Liu et al., 2018). The basic idea of differentiable NAS is: set up an overparameterized network that combines many different types of operations; each operation is associated with an architecture variable (AV) indicating how important the operation is; optimize these AVs together with the weight parameters in the operations to achieve the best performance on the validation set; operations with top-K largest AVs are selected to form the final architecture. A neural architecture can be represented as a directed acyclic graph (DAG) where the nodes represent intermediate representations (e.g., feature maps in CNN) and edges represent operations (e.g., convolution, pooling) over nodes. Each node x_i is calculated in the following way: $x_i = \sum_{j \in \mathcal{P}_i} e_{ji}(x_j)$, where $\mathcal{P}_i$ is a set containing the ancestor nodes of i. $e_{ji}(\cdot)$ denotes the operation associated with the edge connecting j to i. In differentiable NAS, this DAG is overparameterized: the operation $e_{ji}(\cdot)$ on each edge is a weighted combination of all possible operations. Namely, $e_{ji}(x) = \sum_{m=1}^{M} \frac{\exp(a_{jim})}{\sum_{l=1}^{K} \exp(a_{jil})} o_m(x)$, where $o_m(\cdot)$ is the m-th operation (parameterized by a set of weights) and M is the total number of operations. a_{jim} is an architecture variable representing how important $o_m(\cdot)$ is. In the end, the prediction function of this neural network is a continuous one parameterized by the variables $A = \{a\}$ representing the architecture and the weight parameters

W. The prediction loss function is end-to-end differentiable w.r.t both A and W, which can be learned by gradient descent. After learning, operations with top-K largest architecture variables are retained to form the final architecture. The problem in Eq.(2) can be approximately solved by iteratively performing the following two steps:

- Update weight parameters W:

$$W \leftarrow W - \xi \sum_{k=1}^{K} \nabla_W L(D_k^{(\mathrm{tr})}, A, W) \tag{3}$$

- Update architecture variables A:

$$A \leftarrow A - \eta \sum_{k=1}^{K} \nabla_A L(D_k^{(\mathrm{val})}, A, W - \xi \sum_{j=1}^{K} \nabla_W L(D_j^{(\mathrm{tr})}, A, W)) \tag{4}$$

where $\nabla_A L(D_k^{(\mathrm{val})}, A, W - \xi \sum_{j=1}^{K} \nabla_W L(D_j^{(\mathrm{tr})}, A, W))$ can be approximately computed as

$$H_k = \nabla_A L(D_k^{(\mathrm{val})}, A, W') - \frac{\xi}{2\epsilon}(\nabla_A L(D_k^{(\mathrm{tr})}, A, W^+) - \nabla_A L(D_k^{(\mathrm{tr})}, A, W^-)) \tag{5}$$

where $W' = W - \xi \sum_{j=1}^{K} \nabla_W L(D_j^{(\mathrm{tr})}, A, W)$, $W^+ = W + \epsilon \nabla_{W'} L(D_k^{(\mathrm{val})}, A, W')$, and $W^- = W - \epsilon \nabla_{W'} L(D_k^{(\mathrm{val})}, A, W')$.

The server holds the global version of A and W. Each party k has a local copy: A_k and W_k, and also holds an auxiliary variable W'_k. FNAS iteratively performs the following steps until convergence. (1) Each party uses A_k, W_k, and $D_k^{(\mathrm{tr})}$ to calculate $\nabla_{W_k} L(D_k^{(\mathrm{tr})}, A_k, W_k)$, and sends it to the server; (2) The server aggregates $\{\nabla_{W_k} L(D_k^{(\mathrm{tr})}, A_k, W_k)\}_{k=1}^{K}$ received from different parties, performs a gradient descent update of the global W: $W \leftarrow W - \xi \sum_{k=1}^{K} \nabla_{W_k} L(D_k^{(\mathrm{tr})}, A_k, W_k)$, and sends the updated global W to each party which replaces its W'_k with W; (3) Each party calculates the gradient H_k in Eq.(5) and sends it to the server; (4) The server aggregates $\{H_k\}_{k=1}^{K}$ received from different parties, updates $A \leftarrow A - \eta \sum_{k=1}^{K} H_k$, and sends the updated A to each party; (4) Each party replaces A_k with A and replaces W_k with W'_k.

In federated NAS, while the sensitive data of each party can be protected to some extent by avoiding sharing the data with other parties, there is still a significant risk of leaking privacy due to the sharing of intermediate sufficient statistics (e.g., gradients) among parties. It has been shown in several works that the intermediate sufficient statistics can reveal private information if leveraged cleverly (Bhowmick et al., 2018; Carlini et al., 2018; Fredrikson et al., 2015). To address this problem, we study differentially-private (DP) FNAS, which uses DP techniques (Dwork et al., 2006; Dwork, 2008) to achieve a stronger preservation of privacy. A DP algorithm (with a parameter α measuring the strength of privacy protection) guarantees that the log-likelihood ratio of the outputs of the algorithm under two databases differing in a single individual's data is smaller than α. That means, regardless of whether the individual is present in the data, an adversary's inferences about this individual will be similar if α is small enough. Therefore, the privacy of this individual can be strongly protected. Several works have shown that adding random noise to the gradient can achieve differential privacy (Rajkumar and Agarwal, 2012; Song et al., 2013; Agarwal et al., 2018). In this work, we follow the same strategy. For each worker, the gradient updates of A and W are added with random Gaussian noise before sent to the server:

$$G_k = \nabla_{W_k} L(D_k^{(\mathrm{tr})}, A_k, W_k) + U_k \tag{6}$$

$$H_k = \nabla_{A_k} L(D_k^{(\mathrm{val})}, A_k, W_k') - \frac{\xi}{2\epsilon}(\nabla_{A_k} L(D_k^{(\mathrm{tr})}, A_k, W_k^+) - \nabla_{A_k} L(D_k^{(\mathrm{tr})}, A_k, W_k^-)) + V_k \tag{7}$$

where the elements of U and V are drawn randomly from univariate Gaussian distributions with zero mean and a variance of σ_k^2 and γ_k^2 respectively. Algorithm 1 shows the execution workflow in one iteration of the differentially-private federated NAS (DP-FNAS) algorithm. Per-sample gradient clipping is used with hyperparameters R^G and R^H.

3. Theoretical Analysis

In this section, we provide theoretical analysis on the differential privacy (DP) guarantees of the proposed DP-FNAS algorithm. We consider a recently proposed privacy definition, named f-DP (Dong et al., 2019) owing to its tractable and lossless handling of privacy primitives like composition, subsampling, etc. and superior accuracy results than (ϵ, δ)-DP (Dong et al., 2019; Bu et al., 2019). Broadly, composition is concerned with a sequence of analysis on the same dataset where each analysis is informed by the exploration of prior analysis from the previous iteration. Our proposed gradient-based FNAS algorithm involves two instances of private gradient sharing or Gaussian mechanism (Dwork and Roth, 2014), for optimizing weight parameters and architecture variables, between the parties and the central server. One of the two mechanisms composes over the other in one iteration, hence they keep composing onto each other over further iterations of the algorithm. We provide a decoupling analysis of these two mechanisms over the iterations, by leveraging the fact that the datasets used for the two mechanisms are disjoint (one on training set, the other on validation set). We get the results in terms of Gaussian differential privacy (the focal point of the f-DP guarantee family), which ensure privacy in a very interpretable manner by associating it to the hardness of telling apart two shifted normal distributions.

f-DP is a relaxation of (ϵ, δ)-DP recently proposed by (Dong et al., 2019). This new privacy definition preserves the hypothesis testing interpretation of differential privacy. Moreover, it can efficiently analyze common primitives associated with differential privacy, including composition, privacy amplification by subsampling, and group privacy. In our proposed FNAS algorithm, mini-batch subsampling is used for improving computational efficiency. A side benefit of subsampling is that it naturally offers tighter privacy bounds since an individual not contained in a subsampled mini-batch enjoys perfect privacy. The f-DP leverages this fact efficiently for amplifying privacy. In addition, f-DP includes a canonical single-parameter family that is referred to as Gaussian differential privacy (GDP). GDP is the focal privacy definition, due to a central limit theorem, stating that the privacy guarantee of the composition of private algorithms are approximately equivalent to telling apart two shifted normal distributions.

Algorithm 1 Execution semantics in each iteration of the DP-FNAS algorithm

```
for each party k do
  Take a Poisson subsample $I_t \subseteq \{1, ..., N_k^{(tr)}\}$ with subsampling probability $p$
  for $i \in I_t$ do
    $g_t^{(i)} = \nabla_{W_k} L\left(D_k^{(\mathrm{tr})(i)}, A_k, W_k\right)$
    $\bar{g}_t^{(i)} = g_t^{(i)} / \max\left\{1, \left\|g_t^{(i)}\right\|_2 / R^G\right\}$ {Gradient clipping}
  end for
  $G_k = \frac{1}{|I_t|}\left(\sum_{i \in I_t} \bar{g}_t^{(i)} + R^G.U_k\right)$ {Gaussian mechanism}
end for

On the server side:
Update $W \leftarrow W - \epsilon \sum_{k=1}^{K} G_k$
Send $W$ to each party

for each party k do
  Update $W_k^{'} \leftarrow W$
  Take a Poisson subsample $I_t \subseteq \{1, ..., N_k^{(val)}\}$ with subsampling probability $p$
  for $i \in I_t$ do
    $h_t^{(i)} = \nabla_A L\left(D_k^{(\mathrm{val})(i)}, A, W\right)$
    $\bar{h}_t^{(i)} = h_t^{(i)} / \max\left\{1, \left\|h_t^{(i)}\right\|_2 / R^H\right\}$ {Gradient clipping}
  end for
  $H_k = \frac{1}{|I_t|}\left(\sum_{i \in I_t} \bar{h}_t^{(i)} + R^H.V_k\right)$ {Gaussian mechanism}
end for

On the server side:
Update $A \leftarrow A - \eta \sum_{k=1}^{K} H_k$
Send $A$ to each party

for each party k do
  Update $A_k \leftarrow A$
  Update $W_k \leftarrow W_k^{'}$
end for
```

3.1. Preliminaries

An algorithm is considered private if the adversary finds it hard to determine the presence or absence of any individual in two neighbouring datasets. Two datasets, say S and S', are said to be neighbors if one can be derived by discarding an individual from the other. The adversary seeks to tell apart the two probability distributions $\mathcal{M}(S)$ and $\mathcal{M}(S')$, where $\mathcal{M}$ is the randomized mechanism, using a single draw. In light of this observation, it is natural to interpret what the adversary does is testing two simple hypotheses: $H0:$ the true dataset is S, versus $H1:$ the true dataset is S'. Intuitively, privacy is well guaranteed if the hypothesis testing problem is hard. Following this intuition, the definition of (ϵ, δ)-DP (Dwork, 2008) essentially uses the worst-case likelihood ratio of the distributions associated with $\mathcal{M}(S)$ and $\mathcal{M}(S')$ to measure the hardness of testing the two simple hypotheses. f-DP utilizes a more informed measure of this hardness by directly operating with the tradeoff function associated with hypothesis testing. Specifically, f-DP uses the trade-off between type I error and type II error in place of a few privacy parameters in (ϵ, δ)-DP or other divergence-based DP definitions. With this context, we move forward to some formal definitions as stated in (Dong et al., 2019) for our proof.

Definition 3.1. *(Trade-off Function) Let P and Q denote the distributions of $\mathcal{M}(S)$ and $\mathcal{M}(S')$, respectively, and let ϕ be any (possibly randomized) rejection rule for testing $H0: P$ against $H1: Q$. With these in place, the trade-off function of P and Q is defined as:*

$$\begin{aligned} T(P,Q) : [0,1] &\mapsto [0,1] \\ \alpha &\mapsto \inf_{\phi} \{1 - \mathbb{E}_Q[\phi] : \mathbb{E}_P[\phi] \leqslant \alpha\} \end{aligned}$$

Definition 3.2. *Let $G_\mu := T(\mathcal{N}(0,1), \mathcal{N}(\mu,1))$ for $\mu \geqslant 0$. A (randomized) algorithm $\mathcal{M}$ is μ-Gaussian differentially private (GDP) if $T\left(M(S), M\left(S'\right)\right) \geqslant G_\mu$, for all neighboring datasets S and S'.*

That is, μ-GDP says that determining whether any individual is in the dataset is at least as difficult as telling apart the two normal distributions $\mathcal{N}(0,1)$ and $\mathcal{N}(\mu,1)$ based on one draw.

3.2. Privacy analysis

The major results are summarized in the following theorem.

Theorem 3.1. *Consider a gradient-based Federated NAS algorithm (Algorithm 1), which subsamples minibatches (using Poisson subsampling), clips gradients, and perturbs gradients for both weight parameters W and architecture variables A using Gaussian mechanism $\mathcal{M}_t$ at each iteration. Assuming that $D_k^{(tr)}$ and $D_k^{(val)}$ are disjoint for each party k, the algorithm achieves*

$$\frac{B}{N_k^{(tr)}} \sqrt{T\left(\mathrm{e}^{1/\sigma^2 - 1}\right)}\text{-}\mathit{GDP \ for \ mechanism \ composition} \ \mathcal{M}_{t=1:T}^{G_k}(D_k^{(tr)}) \ \mathit{and}$$
$$\frac{B}{N_k^{(val)}} \sqrt{T\left(\mathrm{e}^{1/\tau^2 - 1}\right)}\text{-}\mathit{GDP \ for \ mechanism \ composition} \ \mathcal{M}_{t=1:T}^{H_k}(D_k^{(val)})$$

where GDP refers to Gaussian Differential Privacy, σ^2 and τ^2 represent the variance of the added Gaussian noises U_k and V_k respectively, T is the number of iterations, B is the mini-batch size, $N_k^{(tr)}$ and $N_k^{(val)}$ are the number of training and validation examples owned by party k, respectively.

Remarks

- Intuitively, these privacy bounds reveal that the algorithm gives good privacy guarantees if $B\sqrt{T}/N_k$ is small, and σ or τ are not too small.
- Since GDP is achieved through central limit theorem due to composition of distributions $\mathcal{M}_t(D)$ over T iterations, it is expected that T is large enough. This requirement is usually satisfied with general settings of DP-FNAS training procedure.
- We can also choose different subsampling probability for the two processes, which will reflect accordingly in the privacy bound ($p = B/N_k$). We may also use other subsampling methods like shuffling (randomly permuting and dividing data into folds at each epoch) and uniform sampling (sampling a batch of size L from the whole dataset at each iteration), which will result in slightly varied privacy bounds.
- The utilization of subsampling in the proof adds to the privacy improvement, and is also closer to actual experimental settings. This tighter guarantee allows for some space to reduce the variance of the added Gaussian noise, which decreases privacy (as noted in the first remark), but increases the model convergence accuracy (since the noise' variance is a major factor sacrificing accuracy in private optimization algorithms).

Please refer to the appendix for the proof.

4. Experiments

In this section, we present experimental results on the CIFAR-10 dataset. The task is image classification. Our goal is to search a highly-performing neural architecture for this task. Following (Liu et al., 2018), we first search an architecture cell by maximizing the validation performance. Given the searched cell, we perform augmentation: the cell is used to compose a larger architecture, which is then trained from scratch and measured on the test set.

4.1. Experimental Setup

The search space is the same as that in (Liu et al., 2018). The candidate operations include: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero. The network is a stack of multiple cells, each consisting of 7 nodes. The CIFAR-10 dataset has 60000 images from 10 classes, 50000 for training and 10000 for testing. During architecture search, we used 25000 images of the training set for validation. During augmentation, all 50000 images in the training set were used for training the composed architecture. The variance of noises added to gradient updates of A and W were both set to 1. The hyperparameters R^G and R^H in gradient clipping were set to 0.01 and 0.1 respectively. We experiment with the following settings:

Table 1: Test error under different settings. Note that the search cost is only about architecture search, not including augmentation which trains the composed architecture from scratch.

	#parties	**Test error (%)**	**Params (M)**	**Search cost (GPU days)**	**#ops**
Vanilla NAS	1	2.8 ± 0.10	3.36	1.25	4
FNAS	2	2.9 ± 0.15	3.36	1.21	4
	4	3.2 ± 0.34	3.36	0.67	4
	8	3.3 ± 0.40	3.36	0.55	4
DP-FNAS	1	3.0 ± 0.10	3.36	1.39	4
	2	3.0 ± 0.12	3.36	1.28	4
	4	3.1 ± 0.13	3.36	0.93	4
	8	3.4 ± 0.38	3.36	0.59	4

- NAS with a single party. The vanilla NAS is performed by a single party which has access to all training and validation data.
- Federated NAS with N parties, where $N = 2, 4, 8$. The training data is randomly split into N partitions, each held by one party. So is the validation data. The final architecture is evaluated on the test dataset accessible by the server. The gradients calculated by each party are not obfuscated with random noise.
- Differentially-private FNAS with N parties, where $N = 2, 4, 8$. The gradients calculated by each party are obfuscated with random noise. The rest of settings are the same as those in FNAS.

4.2. Results

Table 2: Validation error achieved by DP-FNAS under different variance of noises. The number of parties is 4.

Variance of Noise	**Validation error (%)**
0.5	14.0 ± 0.32
1.0	14.0 ± 0.32
2.0	14.4 ± 0.43
5.0	15.1 ± 0.85
8.0	16.4 ± 1.01
10.0	19.2 ± 3.27

Table 1 shows the test error and search cost (measured by GPU days) under different settings. From this table, we make the following observations. First, the performance of DP-FNAS with different numbers of parties is on par with that of single-party vanilla NAS. This demonstrates that DP-FNAS are able to search highly-performing neural architectures that are as good as those searched by a single machine while preserving differential privacy of individual parties. Second, in DP-FNAS, as the number of parties increases, the performance drops slightly. This is probably because: Gaussian noise is added to the gradient of each party; more parties result in more added noise, which hurts the convergence of the algorithm. Third, under the same number of parties, DP-FNAS works slightly worse than FNAS. This is because FNAS is noise-free while the gradients in DP-FNAS are obfuscated with noise. However, the performance difference is very small. This shows that DP-FNAS is able to provide stronger privacy protection without substantially degrading performance. Fourth, in FNAS, as the number of parties increases, the performance becomes slightly worse. The possible reason is: as the number of parties increases, the size of data held by each party decreases. Accordingly, the gradient calculated by each party using its hosted data is biased to the data of this party. Such bias degrades the quality of model updates. Fifth, as the number of parties increases, the search cost decreases. This is not surprising since more parties can contribute more computing resources. However, the rate of cost reduction is not linear in the number of parties. This is because communication between parties incurs latency. Sixth, under the same number of parties, DP-FNAS has slightly larger search cost than FNAS. This is because adding noise renders the gradient updates less accurate, which slows down convergence. Seventh, the number of parameters and operations remain the same under different parties, with or without noise. This indicates that DP-FNAS and FNAS do not substantially alter the architectures, compared with those searched by a single machine.

Table 2 shows how the validation error of DP-FNAS with 4 parties varies with the variance of noise. As can be seen, large variance results in larger validation error. This is because noises with larger variance tend to have larger magnitude, which makes the gradient updates less accurate. However, a larger variance implies a stronger level of differential privacy. By tuning the variance of noise, we can explore a spectrum of tradeoffs between strength of privacy protection and classification accuracy.

5. Related Works

Federated NAS There are several works independently conducted in parallel to ours on the topic of federated NAS. In (He et al., 2020), each client locally performs neural architecture search. The architecture variables of different clients are synchronized to their average periodically. This approach has no convergence guarantees. In our work, different parties collaboratively search for a global architecture by exchanging gradients in each iteration, where the convergence is naturally guaranteed. In (Zhu and Jin, 2020), a federated algorithm is proposed to search neural architectures based on the evolutionary algorithm (EA), which is computationally heavy. In our work, a gradient-based search algorithm is used, which has lower computational cost. In (Xu et al., 2020), the search algorithm is based on NetAdapt (Yang et al., 2018), which adapts a pretrained model to a new hardware platform, where the performance of the searched architecture is limited to that of the pretrained model. In our work, the search is performed in a large search space rather than constrained by a human-designed architecture.

Federated Learning Federated learning (FL) is a decentralized learning paradigm which enables multiple parties to collaboratively train a shared model by leveraging data from different parties while preserving privacy. Please refer to (Li et al., 2019) for an extensive review. One key issue in FL is how to synchronize the different parameter copies among parties. One common approach is periodically setting different copies to their average (McMahan et al., 2016), which however has no convergence guarantees. Client-server-based architectures guarantee convergence by exchanging gradients and models between servers and clients, but incur high communication overhead. Konečnỳ et al. (Konečnỳ et al., 2016) proposed two ways to reduce communication costs: learning updates from a restricted space parametrized using a smaller number of variables and compressing updates using quantization, random rotations, and subsampling.

Neural Architecture Search Neural architecture search (NAS) has achieved remarkable progress recently, which aims at searching for the optimal architecture of neural networks to achieve the best predictive performance. In general, there are three paradigms of methods in NAS: reinforcement learning (RL) approaches (Zoph and Le, 2016; Pham et al., 2018; Zoph et al., 2018), evolutionary learning approaches (Liu et al., 2017; Real et al., 2019), and gradient-based approaches (Cai et al., 2018; Liu et al., 2018; Xie et al., 2018). In RL-based approaches, a policy is learned to iteratively generate new architectures by maximizing a reward which is the accuracy on the validation set. Evolutionary learning approaches represent the architectures as individuals in a population. Individuals with high fitness scores (validation accuracy) have the privilege to generate offspring, which replaces individuals with low fitness scores. Gradient-based approaches adopt a network pruning strategy. On top of an over-parameterized network, the weights of connections between nodes are learned using gradient descent. Then weights close to zero are later pruned.

Differential Privacy Rajkumar and Agarwal (Rajkumar and Agarwal, 2012) developed differentially-private machine learning algorithms in a distributed multi-party setting. A client-server architecture is used to aggregate gradients computed by individual parties and synchronize different parameter copies. The gradient calculated in each iteration by each party is added with two sources of random noise: (1) party-dependent and iteration-independent random noise; (2) party-independent and iteration-dependent random noise. Agarwal et al. (Agarwal et al., 2018) studied distributed stochastic gradient descent algorithms that are both computationally efficient and differentially private. In their algorithm, clients add their share of the noise to their gradients before transmission. Aggregation of gradients at the server results in an estimate with noise equal to the sum of the noise added at each client. Geyer et al. (Geyer et al., 2017) proposed an algorithm for preserving differential privacy on clients' side in federated optimization, by concealing clients' contributions during training and balancing the trade-off between privacy loss and model performance.

6. Conclusions and Future Works

In this paper, we study differentially private federated neural architecture search (DP-FNAS), where multiple parties collaboratively search for a highly-performing neural architecture by leveraging the data from different parties, with strong privacy guarantees. DP-FNAS performs distributed gradient-based optimization of architecture variables and weight parameters using a parameter server architecture. Gradient updates are obfuscated with random Gaussian noise to achieve differential privacy. We provide theoretical guarantees of DP-FNAS on privacy preservation. Experiments on varying numbers of parties demonstrate that our algorithm can search neural architectures which are as good as those searched on a single machine while preserving privacy of individual parties. For future works, we aim to reduce the communication cost in DP-FNAS, by developing methods such as gradient compression, periodic updates, diverse example selection, etc.

Appendix A. Proof of Theorem 3.1

A.1. Proof sketch

Here we first present a proof sketch. For the detailed proof, please refer to Section A.2. Algorithm 1 in the main paper has two instances of gradient sharing steps, one for optimizing the weight parameters W, and the other for the architecture parameters A. The gradient for W is calculated using training data, while that for A is calculated using validation data. These two steps in each iteration include two randomized mechanisms, namely $\mathcal{M}^G(D^{(tr)})$ and $\mathcal{M}^H(D^{(val)})$ which are perturbed gradients w.r.t. to W and A respectively. We leverage the fact that the two mechanisms have query functions which are querying on two different datasets with disjoint data points, i.e., the training set will not contain information about individuals which are part of the validation set and vice versa. This limits the association of privacy risk for any individual with only one of the two datasets. Also we know that composition is concerned with a sequence of analysis on the same dataset where each analysis is informed by the exploration of prior analysis. Hence, composition of these two mechanisms over each iteration will not affect the privacy bounds of each other. In that sense, the compositions 8 and 9 decouple as 10 and 11 respectively for any party k as shown:

$$G_k : \mathcal{M}_t^{G_k}(D_k^{(tr)}, W[\mathcal{M}_{t-1}^{G_k}(D_k^{(tr)})], A[\mathcal{M}_{t-1}^{H_k}(D_k^{(val)})]) \tag{8}$$

$$H_k : \mathcal{M}_t^{H_k}(D_k^{(val)}, W[\mathcal{M}_t^{G_k}(D_k^{(tr)})], A[\mathcal{M}_{t-1}^{H_k}(D_k^{(val)})]) \tag{9}$$

$$G_k : \mathcal{M}_t^{G_k}(D_k^{(tr)}, W[\mathcal{M}_{t-1}^{G_k}(D_k^{(tr)})]) \tag{10}$$

$$H_k : \mathcal{M}_t^{H_k}(D_k^{(val)}, A[\mathcal{M}_{t-1}^{H_k}(D_k^{(val)})]) \tag{11}$$

where $\mathcal{M}_t^{G_k}$ represents a randomized mechanism for gradient w.r.t. W at the t^{th} iteration for a party k. It takes previous mechanisms ($\mathcal{M}_{t-1}^{G_k}$ via W and $\mathcal{M}_{t-1}^{H_k}$ via A) as inputs. Similarly, $\mathcal{M}_t^{H_k}$ represents a randomized mechanism for gradient w.r.t. A at the t^{th} iteration for a party k. The above expression is to suggest the recursive phenomena as also evident from Algorithm 1 in the main paper. With these in place, we can argue that the two mechanisms are composing independently along the direction of the iterations for each party. (Note that we ignored the presence of validation set ($D_k^{(val)}$) in the same way we ignore that of datasets from other parties ($D_{l \neq k}^{(tr)}$) since in both scenarios the datasets are presumably disjoint to $D_k^{(tr)}$.)

Note that adding or removing one individual would change the value of $\sum_{i \in I_t} \bar{g}_t^{(i)}$ or $\sum_{i \in I_t} \bar{h}_t^{(i)}$ (from Algorithm 1 in the main paper) by at most R^G or R^H (clipping constants) in the l_2 norm due to the clipping operation. Hence the query function for mechanisms $\mathcal{M}^G(D^{(tr)})$ and $\mathcal{M}^H(D^{(val)})$ has sensitivity R^G and R^H respectively. The major role played by clipping constants reflects in the accuracy achieved by the algorithm. We also subsample the dataset for computing gradients at both instances. We perform Poisson subsampling by choosing a data point with probability p for making a place in the mini-batch used for gradient computation. This gives us the subsampled randomized mechanisms $\mathcal{M}_t^{G_k}(D_k^{(tr)}) \circ Sample_p(D_k^{(tr)})$ and $\mathcal{M}_t^{H^k}(D_k^{(val)}) \circ Sample_p(D_k^{(val)})$ similar to the one in (Bu et al., 2019). The above analysis has translated our problem into two instances of the problem in (Bu et al., 2019). This allows us to leverage the results from (Bu et al., 2019) for each of these compositions independently, which completes the proof of Theorem 1 in the main paper.

A.2. Detailed proof

Writing $f = T(P, Q)$, the definition of tradeoff function says that $f(\alpha)$ is the minimum type II error among all tests at significance level α. Self-evidently, the larger the trade-off function is, the more difficult the hypothesis testing problem is (hence more privacy). With this intuition we have the following privacy definition,

Definition A.1. *A (randomized) algorithm M is f-differentially private if:*

$$T\left(M(S), M\left(S'\right)\right) \geqslant f$$

for all neighboring datasets S and S'.

In this definition, the inequality holds pointwise for all $0 \leq \alpha \leq 1$, and we abuse notation by identifying $\mathcal{M}(S)$ and $\mathcal{M}(S')$ with their associated distributions. We have the following relation of f-DP with (ϵ, δ)-DP from (Wasserman and Zhou, 2008),

Definition A.2. *(Wasserman and Zhou, 2008) (ϵ, δ)-DP is a special instance of f-DP in the sense that an algorithm is (ϵ, δ)-DP iff it is $f_{(\epsilon,\delta)}$-DP with (for all $0 \leq \alpha \leq 1$)*

$$f_{\epsilon,\delta}(\alpha) = \max\left\{0, 1 - \delta - \mathrm{e}^{\epsilon}\alpha, \mathrm{e}^{-\epsilon}(1 - \delta - \alpha)\right\}$$

Definition A.3. *Consider privately releasing a univariate statistic $\theta(S)$. The Gaussian mechanism adds $\mathcal{N}(0, \sigma^2)$ noise to the statistic θ, which gives μ-GDP if $\sigma = \Delta(\theta)/\mu$. Here the sensitivity of θ is defined as $\Delta(\theta) = sup_{S,S_0}|\theta(S) - \theta(S_0)|$, where the supremum is over all neighboring datasets.*

Definition A.4. *(Binary Function) Given trade-off functions $f = T(P, Q)$ and $g = T\left(P', Q'\right)$, the binary function is defined as $f \otimes g = T\left(P \times P', Q \times Q'\right)$.*

A central limit theorem phenomenon arises in the composition of many "very private" f-DP algorithms in the following sense: the trade-off functions of small privacy leakage accumulate to G_μ for some μ under composition. More formally stated as,

Lemma A.1. *(f-DP composition theorem) Assuming each f_t is very close to $Id(\alpha) = 1-\alpha$, which corresponds to perfect privacy, then we have*

$$f_1 \otimes f_2 \otimes \cdots \otimes f_T \text{ is approximately } G_\mu$$

when T is very large and $\otimes$ is a binary function.

As an important fact, the privacy bound $f_1 \otimes f_2 \otimes \cdots \otimes f_T$ cannot be improved in general.
According to Poison subsampling, for each point in the dataset S, any point makes to the subsample independently with probability p. The resulting subsample is denoted by $Sample_p(S)$. Given any algorithm $\mathcal{M}$, denote by $\mathcal{M} \circ Sample_p(S)$ the subsampled algorithm.

Lemma A.2. *(f-DP Subsampling theorem) Let $\mathcal{M}$ be f-DP, write f_p for $pf + (1-p)Id$, and denote by f^{-1} ($f = T(P,Q)$, its inverse $f^{-1} = T(Q,P)$), the subsampled algorithm $M \circ Sample_p$ is $min\{f_p, f_p^{-1}\}^{**}$-DP, where ** represents double conjugate.*

The privacy bound $min\{f_p, f_p^{-1}\}^{**}$ is larger than f and cannot be improved in general.

Lemma A.3. *According to the central limit theorem, when $p\sqrt{T} \to \nu$ for a constant ν, then as $T \to \infty$,*

$$f = \left(pG_{1/\sigma} + (1-p)\mathrm{Id}\right)^{\otimes T} \to G_\mu$$

where $\mu = \nu\sqrt{\mathrm{e}^{1/\sigma^2} - 1}$.

Theorem A.1. *Given an optimization algorithm with a general deep neural network loss function, a Gaussian mechanism with noise variance $R\sigma$, where R is the gradient clipping constant and also the sensitivity of the mechanism's query function, and σ is the variance of the noise random variable. The algorithm along with Possion subsampling (p) for gradient computation at each iteration, composed over T iterations achieves the following privacy guarantee,*

$$p\sqrt{T\mathrm{e}^{1/\sigma^2} - 1} - GDP$$

Proof The query function for the Gaussian mechanism $\mathcal{M}$ is the gradient of a general neural network loss evaluated w.r.t. any model parameters to be optimized. The sensitivity of the query function is given to be R. The standard deviation of the added noises is $R\sigma$. According to definition A.3, it is ensured that $\mathcal{M}$ is $\frac{1}{\sigma}$-GDP. As per the arguments in the Appendix of (Bu et al., 2019) (using composition and subsampling theorem for f-DP), $\mathcal{M}_{t=1-T}$ is $min\{f_p, f_p^{-1}\}^{**}$-DP with $f_p = \left(pG_{1/\sigma} + (1-p)\mathrm{Id}\right)^{\otimes T}$ (composition over T iterations). Using lemma A.3,

$$\min\left\{f, f^{-1}\right\}^{**} \approx \min\left\{G_\mu, G_\mu^{-1}\right\}^{**} = G_\mu^{**} = G_\mu$$

Hence, the algorithm with composition of subsampled algorithm $\mathcal{M}_t \circ Sample_p(D)$ over T iterations is $p\sqrt{T\mathrm{e}^{1/\sigma^2} - 1}$-GDP (from lemma A.3).

References

Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpsgd: Communication-efficient and differentially-private distributed sgd. In *Advances in Neural Information Processing Systems*, pages 7564–7575, 2018.

Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.

Zhiqi Bu, Jinshuo Dong, Qi Long, and Weijie J Su. Deep learning with gaussian differential privacy. *arXiv preprint arXiv:1911.11607*, 2019.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *arXiv preprint arXiv:1802.08232*, 2018.

Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL https://doi.org/10.1561/0400000042.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

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

Chaoyang He, Murali Annavaram, and Salman Avestimehr. Fednas: Federated deep learning via neural architecture search. *arXiv preprint arXiv:2004.08546*, 2020.

Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Arun Rajkumar and Shivani Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In *Artificial Intelligence and Statistics*, pages 933–941, 2012.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

Larry Wasserman and Shuheng Zhou. A statistical framework for differential privacy, 2008.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

Mengwei Xu, Yuxin Zhao, Kaigui Bian, Gang Huang, Qiaozhu Mei, and Xuanzhe Liu. Neural architecture search over decentralized data. *arXiv preprint arXiv:2002.06352*, 2020.

Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 285–300, 2018.

Hangyu Zhu and Yaochu Jin. Real-time federated evolutionary neural architecture search. *arXiv preprint arXiv:2003.02793*, 2020.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.