Differentially-private Federated Neural Architecture Search

Ishika Singh [*1] **Haoyi Zhou** [*2] **Kunlin Yang** [3] **Meng Ding** [3] **Bill Lin** [3] **Pengtao Xie** [3]

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 differentially-private federated neural architecture search (DP-FNAS), where different parties collectively search for a differentiable architecture by privately exchanging 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.

1. Introduction

In many application scenarios, the data owners prefer 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 and having more training data usually improves performance. One way to have more training data is to combine data of the same kind from multiple parties to collectively train a model. However, since each of these datasets contains private information, they can't be shared across parties. Federated learning (Konečnỳ 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 (NAS) (Zoph & 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 propose federated NAS (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 A indicating how important this operation is. The prediction loss is a continuous function w.r.t A as well as the weight parameters W within individual operations. A and W are learned by minimizing the validation loss using a stochastic gradient descent (SGD) algorithm. After learning, operations with top-K 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 SGD 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.,

[*]Equal contribution [1]Indian Institute of Technology Kanpur, India [2]Beihang University [3]University of California San Diego. Correspondence to: Ishika Singh <ishikas@iitk.ac.in>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020 (FL-ICML'20 Workshop).

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 exhibit 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), enabling 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 paper is organized as follows. Section 2, 3 and 4 present the method, privacy analysis and experiments. Section 5 reviews related works and Section 6 concludes.

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, restricting data sharing across parties. 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).

For ML experts, to design an effective model architecture, the experts need to thoroughly analyze the properties of the data. In an FL setting (facilitated with more data with privacy preservation), 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 & 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:

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

where $D^{(\text{tr})}$ and $D^{(\text{val})}$ 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 $L(D^{(\text{val})}, A, W^*(A))$ 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 & 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:

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

where $D_k^{(\text{tr})}$ and $D_k^{(\text{val})}$ 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 continuous in (1) $A = \{a\}$ representing the architecture and (2) 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 operation weights W:

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

- Update architecture weights A:

$$A \leftarrow A - \eta \sum_{k=1}^{K} \nabla_A L(D_k^{(\text{val})}, A, W') \tag{4}$$

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

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

where $W^+ = W + \epsilon \nabla_{W'} L(D_k^{(\text{val})}, A, W')$, and $W^- = W - \epsilon \nabla_{W'} L(D_k^{(\text{val})}, A, W')$.

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**
    $g_t^{(i)} = \nabla_{W_k} L\left(D_k^{(\text{tr})(i)}, A_k, W_k\right)$
    $\bar{g}_t^{(i)} = g_t^{(i)} / \max\left\{1, \left\|g_t^{(i)}\right\|_2 / R^G\right\}$ *(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**
    $h_t^{(i)} = \nabla_A L\left(D_k^{(\text{val})(i)}, A, W\right)$
    $\bar{h}_t^{(i)} = h_t^{(i)} / \max\left\{1, \left\|h_t^{(i)}\right\|_2 / R^H\right\}$ *(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

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 $D_k^{(\text{tr})}$ to calculate $\nabla_{W_k} L(D_k^{(\text{tr})}, A_k, W_k)$, and sends it to the server; (2) The server aggregates $\{\nabla_{W_k} L(D_k^{(\text{tr})}, A_k, W_k)\}_{k=1}^K$ received from different parties, performs a gradient descent update of the global W: $W \leftarrow W - \xi \sum_{k=1}^K \nabla_{W_k} L(D_k^{(\text{tr})}, A_k, W_k)$, 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 & 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:

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

$$\begin{aligned} H_k =& \nabla_A L(D_k^{(\text{val})}, A, W') - \frac{\xi}{2\epsilon}(\nabla_A L(D_k^{(\text{tr})}, A, W^+) \\ & - \nabla_A L(D_k^{(\text{tr})}, A, W^-)) + V_k \end{aligned} \tag{7}$$

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). 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. Broadly, composition is concerned with a sequence of analysis on the same dataset where each analysis is informed by the exploration of prior analysis in the previous iteration. Our proposed gradient-based FNAS algorithm involves two instances of private gradient sharing or Gaussian mechanism (Dwork & Roth, 2014), for optimizing A and W, 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 (GDP) (the focal point of the f-DP guarantee family due to a central limit theorem). GDP ensure privacy in a very interpretable manner. It states that the privacy guarantee of the composition of private algorithms are approximately equivalent to telling apart two shifted normal distributions.

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.

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. The adversary tests two simple hypotheses: $H0$: the true dataset is S, versus $H1$: the true dataset is S'. Hence, privacy is well guaranteed if this 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 $\mathcal{M}(S)$ 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 formalize some 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:*

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

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 hard as telling apart 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*

$$\frac{B}{N_k^{(tr)}}\sqrt{T\left(\mathrm{e}^{1/\sigma^2-1}\right)}\text{-GDP}$$

for mechanism composition $\mathcal{M}_{t=1-T}^{G_k}(D_k^{(tr)})$ and

$$\frac{B}{N_k^{(val)}}\sqrt{T\left(\mathrm{e}^{1/\tau^2-1}\right)}\text{-GDP}$$

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.
- 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 detailed 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:

- 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.

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

#Parameters and #Operations for each case are 3.36M and 4 respectively.

4.2. Results

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 is able to search highly-performing neural architectures that are as good as those searched by a single machine while preserving individual differential privacy. 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 performs slightly worse than FNAS, due to noise addition to gradients. However, the 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 degrades slightly. The possible reason is: due to decrease in data held per party, gradients calculated get biased towards its party's small dataset, hence degrading the model updates. Fifth, as the number of parties increases, the search cost decreases as more parties can contribute more computing resources. However, the rate of cost reduction is not linear in the number of parties due to communication 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 each setting. This indicates that DP-FNAS and FNAS do not substantially alter the architectures, compared with those searched by a single machine.

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 & 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.

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.

References

Agarwal, N., Suresh, A. T., Yu, F. X. X., Kumar, S., and McMahan, B. cpsgd: Communication-efficient and differentially-private distributed sgd. In *Advances in Neural Information Processing Systems*, pp. 7564–7575, 2018.

Bhowmick, A., Duchi, J., Freudiger, J., Kapoor, G., and Rogers, R. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.

Bu, Z., Dong, J., Long, Q., and Su, W. J. Deep learning with gaussian differential privacy, 2019.

Carlini, N., Liu, C., Kos, J., Erlingsson, Ú., and Song, D. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *arXiv preprint arXiv:1802.08232*, 2018.

Dong, J., Roth, A., and Su, W. J. Gaussian differential privacy, 2019.

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

Dwork, C. and Roth, A. 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.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284. Springer, 2006.

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

He, C., Annavaram, M., and Avestimehr, S. Fednas: Federated deep learning via neural architecture search. *arXiv preprint arXiv:2004.08546*, 2020.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

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

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Song, S., Chaudhuri, K., and Sarwate, A. D. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pp. 245–248. IEEE, 2013.

Xu, M., Zhao, Y., Bian, K., Huang, G., Mei, Q., and Liu, X. Neural architecture search over decentralized data. *arXiv preprint arXiv:2002.06352*, 2020.

Yang, T.-J., Howard, A., Chen, B., Zhang, X., Go, A., Sandler, M., Sze, V., and Adam, H. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 285–300, 2018.

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

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

A. Proof of Theorem 3.1

Algorithm 1 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. 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) 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 3.1.

B. Accuracy-Privacy Tradeoff Analysis

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.

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