DPNAS: Neural Architecture Search for Deep Learning with Differential Privacy

Anda Cheng[1,2], **Jiaxing Wang**[3], **Xi Sheryl Zhang**[1,4], **Qiang Chen**[1], **Peisong Wang**[1], **Jian Cheng**[1,2,4]*
[1]Institute of Automation, Chinese Academy of Sciences,
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, [3]JD.com, [4]AIRIA,
chenganda2017@ia.ac.cn, {wjiaxing94, sheryl.zhangxi}@gmail.com,
{qiang.chen, peisong.wang,jcheng}@nlpr.ia.ac.cn

Abstract

Training deep neural networks (DNNs) for meaningful differential privacy (DP) guarantees severely degrades model utility. In this paper, we demonstrate that the architecture of DNNs has a significant impact on model utility in the context of private deep learning, whereas its effect is largely unexplored in previous studies. In light of this missing, we propose the very first framework that employs neural architecture search to automatic model design for private deep learning, dubbed as DPNAS. To integrate private learning with architecture search, we delicately design a novel search space and propose a DP-aware method for training candidate models. We empirically certify the effectiveness of the proposed framework. The searched model DPNASNet achieves state-of-the-art privacy/utility trade-offs, e.g., for the privacy budget of $(\epsilon, \delta) = (3, 1 \times 10^{-5})$, our model obtains test accuracy of 98.57% on MNIST, 88.09% on FashionMNIST, and 68.33% on CIFAR-10. Furthermore, by studying the generated architectures, we provide several intriguing findings of designing private-learning-friendly DNNs, which can shed new light on model design for deep learning with differential privacy.

Introduction

Deep neural networks (DNNs) have achieved massive successes in a variety of tasks, including image understanding, natural language processing, etc (Goodfellow, Bengio, and Courville 2016; LeCun, Bengio, and Hinton 2015). Broadly, on the other hand, DNNs may compromise sensitive information carried in the training data (Shokri et al. 2017; Fredrikson, Jha, and Ristenpart 2015; Zhu, Liu, and Han 2019), thereby raising privacy issues. To counter this, learning algorithms that provide principled privacy guarantees in the line of differential privacy (DP) (Dwork and Roth 2014; Kearns and Roth 2020; Vadhan 2017) have been developed. For instance, differentially private stochastic gradient descent (DPSGD) (Abadi et al. 2016), a generally applicable modification of SGD, is widely adopted in differentially private deep learning (DPDL) applications, ranging from medical image recognition (Wu et al. 2019), image generation (Xie et al. 2018), to federated learning (McMahan et al. 2017), to name a few.

However, training DNNs with strong DP guarantees inevitably degrades model *utility* (Dwork and Roth 2014). To improve the utility for meaningful DP guarantees, prior works have proposed a wealth of strategies, e.g., gradient dimension reduction (Yu et al. 2021; Zhou, Wu, and Banerjee 2021), adaptive clipping (Thakkar, Andrew, and McMahan 2019; Pichapati et al. 2019), adaptive budget allocation (Lee and Kifer 2018), transfer learning with non-sensitive data (Tramèr and Boneh 2021; Wang and Zhou 2020), and so forth. Whilst all these methods focus on improving training algorithms, the impact of model architectures on the utility of DPDL is so far unexplored. Formally, a learning algorithm $\mathcal{M}$ that trains models from the output set $\mathcal{O} = \text{Range}(\mathcal{M})$ is (ϵ, δ)-DP, if $\Pr[\mathcal{M}(D) \in \mathcal{O}] \leq e^{\epsilon}\Pr[\mathcal{M}(D') \in \mathcal{O}] + \delta$ holds for all training sets D and D' that differ by exactly one record. For DPSGD, its output at each step is a high-dimension gradient vector, which is affected by model architecture. As a result, using different model architectures do not change the privacy of DPSGD but could have effect on the output space $\mathcal{O}$, thereby affecting the model utility.

To investigate the effect of model architectures, we delve into the utility comparisons of several existing DNNs by training them with SGD and DPSGD. The comparisons include hand-crafted models (He et al. 2016; Huang, Liu, and Weinberger 2017; Zagoruyko and Komodakis 2016; Papernot et al. 2021) and automatically searched models (Zoph et al. 2018a; Liu, Simonyan, and Yang 2019; Tan and Le 2019; Real et al. 2019; Pham et al. 2018). Without loss of generality, three observations can be summarized from Figure 1. Firstly, for DPDL, the models designed without considering private training, e.g. triangle cores, perform much poorer than the model specially designed, e.g. diamond core. Secondly, there is no clear positive correlation between the performance of a model trained with and without DP. Lastly, the model utility obtained with DPSGD varies greatly even though the model size are comparable, which appears to be a different trend comparing to the model performance in conventional deep learning. At a high level, the above observations suggest that (1) *Model architectures can significantly affect the utility of DPDL;* (2) *Instead of directly using DNNs built for conventional deep learning, it is necessary to redesign models for DPDL to improve the utility.* Whilst model architecture is important for DPDL, there is little experience or common knowledge to draw on to design DP-friendly models.

*Corresponding Author.

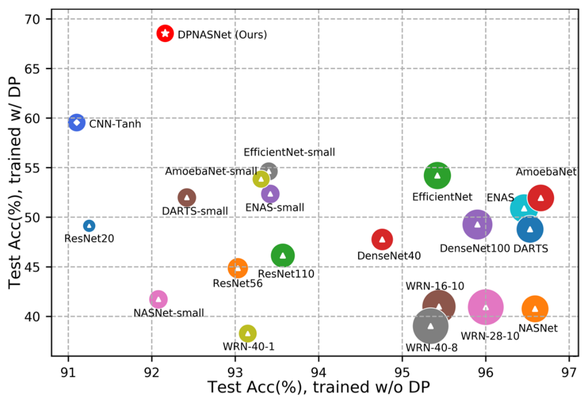


Figure 1: Comparing the utility of different models for DPDL on CIFAR-10. The y-axis and x-axis show the test accuracy of models trained with DPSGD and SGD, respectively. The triangle core denotes the model designed for conventional deep learning. The diamond core denotes the model designed for DPDL. The size of the circumcircle reflects the number of model parameters. Our DPNASNet is superior to the existing DNN models for DPDL.

In this paper, inspired by the above insights and the advances of neural architecture search (NAS) (Zoph and Le 2016), we propose DPNAS, the first effort to automatically search models for DPDL. Our motivation is to boost the privacy/utility trade-offs with least prior knowledge by integrating private learning with architecture search. To this end, we design a novel search space for DPDL and propose a DP-aware method for training candidate models during the search process. The searched model DPNASNet achieves new SOTA results. Especially, for privacy budget $(\epsilon, \delta) = (3, 1 \times 10^{-5})$, we gain the test accuracy of 98.57%, 88.09%, and 68.33% on MNIST, FashionMNIST, and CIFAR-10, respectively. In ablation studies, we verify the effectiveness of our approach and the merits of the resulted models. The advantages of DPNAS enables us to not only automatically design better models with little prior knowledge but also summarise some rules about model design for DPDL. We conduct analysis on the resulted models and provide several new findings for designing DP-friendly DNNs, concluded as (1) SELU (Klambauer et al. 2017) is more suitable for DPDL than Tanh (tempered sigmoid); (2) The activation functions that can retain the negative values could be more effective for DPDL; (3) Max pooling is better than average pooling for DPDL.

Our main contributions are summarized as below:

- We propose DPNAS, the first framework that employs NAS to search models for private deep learning. We introduce a novel search space and propose a DP-aware method to train the candidate models during the search.
- Our searched model DPNASNet substantially advances SOTA privacy/utility trade-offs for private deep learning.
- We conduct analysis on the resulted model architectures and provide several new findings of model design for deep learning with differential privacy.

Related Work

Differentially Private Deep Learning. Differential privacy (DP) (Dwork and Roth 2014) is a measurable definition of privacy that provides provable guarantees against individual identification in a data set. To address the privacy leakage issue in deep learning, Abadi et al.(Abadi et al. 2016) propose differentially private stochastic gradient descent (DPSGD), which is a generally applicable modification of SGD. Despite providing a DP guarantee, DPSGD brings about a significant cost of model utility. To ensure the utility of DPSGD trained models, the line of works (Yu et al. 2021; Zhou, Wu, and Banerjee 2021; Thakkar, Andrew, and McMahan 2019; Pichapati et al. 2019; Lee and Kifer 2018; Wang and Zhou 2020) make efforts to improve the DP training algorithm while neglecting the benefit of enlarging the algorithm space by varying model architectures. Recently, Papernot et al. (Papernot et al. 2021) observes that DPSGD can cause exploding model activations, thereby leading to model utility degradation. To make models more suitable for DPSGD training, they replace the unbounded ReLUs with the bounded Tanhs as the activation function. However, their design is hand-crafted and does not involve other factors of model architectures that may also affect the utility. On the contrary, our work explicitly and systematically studies the effect of model architecture on the utility of DPDL and propose to boost the utility via automatically searching by considering both activation function selection and network topology.

Neural Architecture Search (NAS). NAS is an automatic model architecture designing process to facilitate fewer human efforts and higher model utilities. A typical process of NAS can be described as follows. The *search strategy* first samples a candidate model from the *search space*, where the model is trained and evaluated according to the *performance estimation*. Then, the estimation result is feedback to the search strategy as guidance to select better models. Hence, the optimal model could be obtained by sequential iterations. Following this paradigm, widely used search strategies including reinforcement learning (RL) (Zoph and Le 2016; Zoph et al. 2018a; Baker et al. 2017; Pham et al. 2018), evolution algorithms (EA) (Real et al. 2017, 2019), and gradient-based optimization of architectures (Liu, Simonyan, and Yang 2019; Dong and Yang 2019; Cai, Zhu, and Han 2018) are developed. For providing a subtle search space, previous methods tend to search optimal connections and operations in the cell level (Zoph et al. 2018a; Liu, Simonyan, and Yang 2019; Pham et al. 2018; Real et al. 2019; Cai, Zhu, and Han 2018; Dong and Yang 2019). By doing so, the overall architecture of a model is constructed with cells that share the same architecture. The search process attempts to find optimal internal connections and operations of the cell. The connection topology of the cell can be established based on predefined motifs (Liu et al. 2018) or automatically searched structure (Zoph et al. 2018a). The operations normally are set as non-parametric operations and convolutions with different filter sizes. It deserves to note that the choice of activation functions is usually not taken into account in search spaces dealing with non-privacy-preserving deep learning tasks. In contrast to the existing NAS methods, our DPNAS aims at

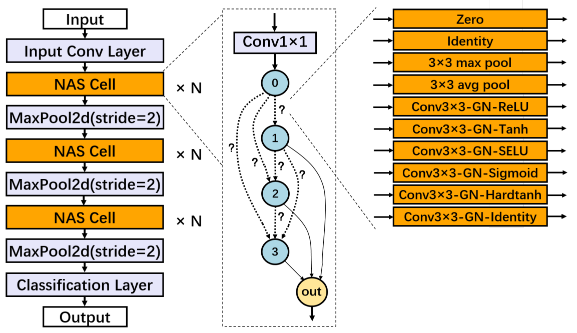


Figure 2: Illustration of search space. The leftmost part shows the overall model architecture. The middle part shows the predefined fully-connected structure of the NAS cell. "?" indicates the operation on each path will be chosen by searching. The rightmost part shows the candidate operations.

searching high-performance models for private learning with a different training method for candidate models on a sophisticated designed search space involving multiple types of activation functions.

Methodology

Our goal is to employ NAS to search networks that are more suitable for private training. As the previous NAS methods are proposed for non-privacy-preserving tasks, their formulations are not ideal for achieving our goal. We introduce our re-formulation in Section . Based on our formulation, the design of our DPNAS framework includes three parts, which are the design of search space, the DP-aware training method for candidate networks, and the search algorithm. For the remainder of this section, we present our novel search space in Section , describe the proposed DP-aware training method in Section , and introduce the search algorithm in Section .

DPNAS Formulation

NAS is formulated as follows. The architecture search space $\mathcal{A}$ is a set of candidate architectures that can be denoted by directed acyclic graphs (DAG). For a specific architecture $a \in \mathcal{A}$, the corresponding network can be denoted by $\mathcal{F}(a, w(a))$, where w represents the network weights of the candidate architecture a. The search of the optimal architecture a^* in NAS can be formulated as a bi-level optimization:

$$\begin{aligned} a^* &= \max_{a\in\mathcal{A}} \mathcal{R}(\mathcal{F}(a, \mathbf{w}_a^*), D_{val}) \\ \text{s.t. } \mathbf{w}_a^* &= \mathcal{OPT}(\mathcal{F}(a, \mathbf{w}_a), D_{train}) \end{aligned} \tag{1}$$

where $\mathcal{R}(\mathcal{F}(a, \mathbf{w}_a^*), D_{val})$ is the reward of network $\mathcal{F}(a, \mathbf{w}_a^*)$ on validation data D_{val}, $\mathcal{OPT}$ is an optimization algorithm to find optimal weight $\mathbf{w}^*$ on training data D_{train} given candidate architecture a. For traditional architecture searching without considering private training, $\mathcal{OPT}$ is typically selected as a non-private learning algorithm, e.g. SGD. But for private learning, the resulting architecture a^* will be trained by a differentially private learning algorithm such as DPSGD. As a result, the architectures resulted by the above formulation is not suitable. Instead, while considering private-training process, we should replace $\mathcal{OPT}$ in equation 1 with a differentially private optimization algorithm $\mathcal{DP}$-$\mathcal{OPT}$, which results to DPNAS formulation as following:

$$\begin{aligned} a^* &= \max_{a\in\mathcal{A}} \mathcal{R}(\mathcal{F}(a, \mathbf{w}_a^*), D_{val}) \\ \text{s.t. } \mathbf{w}_a^* &= \mathcal{DP}\text{-}\mathcal{OPT}(\mathcal{F}(a, \mathbf{w}_a), D_{train}) \end{aligned} \tag{2}$$

Search Space Design

We specially design a novel search space for DPDL. Following (Zoph et al. 2018a; Liu, Simonyan, and Yang 2019; Pham et al. 2018), we search for computation cell as the basic building unit to construct the whole network architecture. As shown in the leftmost part of Figure 2, the overall structure of the network is chained. After an input convolution layer, N cells are stacked as main computation blocks. All the cells share the same architecture. The resolution of input features and output features of a cell is the same. After each cell, a 3×3 max pooling layer with stride 2 is stacked as a down-sampling layer. The stacked cells with a down-sampling layer are referred to as a *block*. We repeat stacking *blocks* for three times and end up with a classification layer.

We aim at searching the internal connection of the cell. As shown in the middle part of Figure 2, a cell is a fully-connected directed acyclic graph $G = (V, E)$ consisting of an ordered sequence of $K + 1$ nodes. Each node V^i (denoted as blue circle) corresponds to an intermediate feature map f^i. The input node V^0 is obtained by applying a convolution with filter size 1×1 to the inputs, which aims at making the dimension of inputs adapt to the channel size of convolution filters in this cell. The nodes $\{V^i|i \in 0...K\}$ are internal nodes, each of which is connected to all the previous nodes in this cell. Each directed edge $E^{(i,j)}$ is associated with some operation $o(i;j)$ chosen from a pre-defined operation pool $\mathcal{O} = \{o_k(\cdot)|k \in 1...n\}$ containing n candidate operations. Each internal node is computed based on all of its predecessors: $f^{(j)} = \sum_{i<j} o^{(i,j)}(f^{(i)})$. The output of the cell is obtained by applying concatenation to all the internal nodes.

As shown in the rightmost part of Figure 2, our operation pool $\mathcal{O}$ is designed with involving different types of activation functions which is much different from the existing NAS methods. As mentioned in (Papernot et al. 2021), the choice of activation functions is crucial for improving the model utility for DPDL. Therefore, in our search space, we include 5 different types of non-linear activation functions, which are ReLU, Tanh, Sigmoid, Hardtanh, and SELU (Klambauer et al. 2017), as well as identity as a linear activation function. The activation functions are integrated into *Conv3×3-Normalization-Activation* blocks. We use Group Normalization (GN) (Wu and He 2018) as normalization method because the widely used Batch Normalization (Ioffe and Szegedy 2015) could cost privacy budget while GN not (Tramèr and Boneh 2021). We also take the topology of cells into considering, which is not considered in (Papernot et al. 2021). We include 4 non-parameter operations into our search space, which are *Identity*, *3×3 max pooling*, *3×3 average pooling*, and *Zero*. The involvement of these operations enables more possible topologies of the cell. For example, DPNAS can use *Zero* operation to indicate a lack of connection between two nodes in the cell.

DP-aware Training for Candidate Architectures

A differently private learning algorithm execute different process compared with its corresponding non-private algorithm. With considering this difference, we need to choose a private training algorithm as $\mathcal{DP}$-$\mathcal{OPT}$ in equation 2 for training the sampled candidate networks to guide the resulted networks more adaptive to $\mathcal{DP}$-$\mathcal{OPT}$. In our implementation, we use DPSGD as $\mathcal{DP}$-$\mathcal{OPT}$. DPSGD makes two changes to every iteration of SGD before updating model weights with computed gradients. It firstly bounds the sensitivity of the learning process to each individual training example by computing per-example gradients $\{g_i|i \in \{0...m-1\}$ with respect to the training loss for m model parameters $\{w_i|i \in 0...m-1\}$, and clipping each per-example gradients to a maximum fixed l_2 norm C. DPSGD then adds Gaussian noise $N(0, \sigma^2 C^2 \mathrm{I})$ to the average of these per-example gradients, where σ is noise intensity selected according to the privacy budget (ϵ, δ) (Abadi et al. 2016). Previous works (Papernot et al. 2021; Yu et al. 2021) indicate that both of these two steps could make negative impacts on the learning process, thereby degrading the utility of the resulted models.

To make the search process aware of the impact of those two steps, we include those two steps into the training processes of sampled architectures and carefully select the value of hyper-parameters C and σ for the search. In practice, we tend to set C to be small as the intensity of the added noise scales linearly with C. To make the resulted architectures more adaptive to the gradient clipping, in the search process, we set C small, e.g. $C = 0.1$. As for σ, we can choose σ without considering privacy cost during the search processes because the resulted architectures will be trained on private datasets from scratch. In practice, we empirically find that setting σ to large values could severely slow down the convergence rate of sampled architectures, thereby making the search process unreliable. Therefore, in our implementation, we set σ to be relatively small.

Search Strategy

We use RL-based search strategy (Zoph and Le 2016; Zoph et al. 2018a; Pham et al. 2018) with parameter sharing (Pham et al. 2018). Given the number of internal nodes K, at each step of the search, the RNN controller samples an architecture denoted as a operation sequence $\{o^{(i,j)}|i \in 0...K, j \in 0...K, i < j\}$. This architecture is trained on a batch of training data using the training method described in Section . After repeating the above sampling and training step for an epoch on the training set, the RNN controller is then trained with the policy gradient on the validation set aiming at selecting architectures that maximizes the expected reward. The expected reward is defined as the validation accuracy of a network adding the weighted controller entropy. The search process executes the above two processes alternately until the RNN controller achieves convergence. The overall search process is depicted in Alg. 1.

Algorithm 1: Search Process of DPNAS

```
Input: Training set $D_{train}$, validation set $D_{val}$, batch size $B_n$ for training candidate networks, batch size $B_c$ for training controller, learning rate $\eta_n$ for networks, learning rate $\eta_c$ for controller, network training iterations $T_n$, controller training iterations $T_c$, total epochs $E$.
Output: Trained controller $S_\theta$
 1: Initialize controller $S_\theta$;
 2: Initialize weights of modules in search space $\mathcal{A}$;
 3: for $e = 1$ to $E$ do
 4:     for $i = 1$ to $T_n$ do
 5:         Sample a batch $\{d_i\}_{i=1}^{B_n} \subseteq D_{train}$;
 6:         Sample a model architecture $a \in \mathcal{A}$ using $S_\theta$;
 7:         Train model: $\mathbf{w}_a \leftarrow \mathcal{DP}\text{-}\mathcal{OPT}(a, \mathbf{w}_a, \{d_i\}_{i=1}^{B_n})$;
 8:     end for
 9:     for $i = 1$ to $T_c$ do
10:         Sample a batch $\{d_i\}_{i=1}^{B_c} \subseteq D_{val}$;
11:         Sample an architecture $a \in \mathcal{A}$ using $S_\theta$;
12:         Update controller: $\theta \leftarrow \theta + \eta_c \cdot \frac{1}{B_c} \sum_i \nabla_\theta \mathcal{R}(a, d_i)$;
13:     end for
14: end for
15: return $S_\theta$
```

Experiments

Experimental Settings

We run DPNAS on MNIST (LeCun and Cortes 2005), FashionMNIST (Xiao, Rasul, and Vollgraf 2017), and CIFAR-10 (Krizhevsky 2009). We split the training data of each dataset into the training set and validation set with the ratio of $0.6 : 0.4$. For the overall architecture, the number of stacked NAS cells in each stage is $N = 1$, the number of internal nodes is $K = 5$, and the internal channels of the three stages are 48, 96, 192 for CIFAR-10 and 32, 32, 64 for MNIST and FashionMNIST. The sampled architectures are trained with DPSGD with weight decay 2×10^{-4}, and moment 0.9. The batch size is set to 300 and the learning rate is set to 0.02. The RNN controller used in our search process is the same as the RNN controller used in (Pham et al. 2018). It is trained with Adam optimizer (Kingma and Ba 2014). The batch size is set to 64 and the learning rate is set to 3×10^{-4}. The trade-off weight for controller entropy in the reward is set to 0.05. Our search process runs for 100 epochs. The first 25 epochs are warm-up epochs that only training sampled architectures without updating the RNN controller.

We evaluate the utility of searched models for DPDL on three common benchmarks: MNIST (LeCun and Cortes 2005), FashionMNIST (Xiao, Rasul, and Vollgraf 2017), and CIFAR-10 (Krizhevsky 2009). On each dataset, the evaluated models are constructed by the resulted cells searched on this dataset. The evaluated models are trained on the training set *from scratch* using DPSGD and tested on the testing set. The privacy cost of training is computed by using the Rényi DP analysis of Gaussian mechanism (Mironov, Talwar, and Zhang 2019; Wang, Balle, and Kasiviswanathan 2019). We implement the search process and private training by PyTorch (Paszke et al. 2019) with *opacus* package. As for model ar-

Datasets		Models		
		CNN-Tanh(Papernot et al. 2021)	DPNASNet-S	DPNASNet
MNIST	Acc(%), $\epsilon = 1$	89.08 ± 0.07	92.87 ± 0.07	$\mathbf{97.22} \pm 0.08$
	Acc(%), $\epsilon = 2$	97.24 ± 0.10	97.74 ± 0.08	$\mathbf{98.18} \pm 0.11$
	Acc(%), $\epsilon = 3$	98.10 ± 0.07	98.35 ± 0.09	$\mathbf{98.57} \pm 0.10$
	#Params	0.03M	0.03M	0.21M
FashionMNIST	Acc(%), $\epsilon = 1$	73.86 ± 0.12	78.66 ± 0.14	$\mathbf{82.08} \pm 0.28$
	Acc(%), $\epsilon = 2$	82.03 ± 0.17	84.83 ± 0.16	$\mathbf{86.17} \pm 0.28$
	Acc(%), $\epsilon = 3$	86.25 ± 0.20	86.73 ± 0.19	$\mathbf{88.09} \pm 0.29$
	#Params	0.03M	0.03M	0.21M
CIFAR-10	Acc(%), $\epsilon = 1$	38.74 ± 0.27	–	$\mathbf{52.95} \pm 0.35$
	Acc(%), $\epsilon = 2$	52.74 ± 0.25	–	$\mathbf{65.03} \pm 0.42$
	Acc(%), $\epsilon = 3$	59.07 ± 0.28	–	$\mathbf{68.33} \pm 0.45$
	#Params	0.55M	–	0.53M

Table 1: Comparison with the SOTA model for DPDL on MNIST, FashionMNIST, and CIFAR-10. DPNASNet-S is a smaller version of DPNASNet, it has comparable parameters with CNN-Tanh. The models are trained with DPSGD for the privacy budgets of $\epsilon = 1, 2, 3$ and $\delta = 1 \times 10^{-5}$.

chitectures, the number of stacked NAS cells in each stage is $N = 1$, the number of internal nodes is $K = 5$, and the internal channels of three stages are set to 48, 96, 192 for CIFAR-10 and 32, 32, 64 for MNIST and FashionMNIST. All experiments are conducted on a NVIDIA Titan RTX GPU with 24GB of RAM.

Main Results

In Table 1, we compare the searched model DPNASNet with the recent SOTA model CNN-Tanh from (Papernot et al. 2021), which is a CNN model with Tanh as the activation function. Comparisons are conducted on MNIST, FashionMNIST, and CIFAR-10 for different privacy budgets of $\epsilon = 1, 2, 3$ with fixed privacy parameter $\delta = 1 \times 10^{-5}$. On each dataset, DPNASNet is built by stacking the searched cell on this dataset. For fair comparison, we also construct a smaller version of DPNASNet by reducing the channel numbers of convolution layers, denoted as DPNASNet-S, which has a comparable number of parameters with CNN-Tanh. We train each model with DPSGD from scratch for 10 times and report the mean and standard deviation of test accuracy of 10 models. Our DPNASNet achieves new SOTA. On MNIST, DPNASNet achieves 98.18% for the privacy budget of $\epsilon = 2.0$, whereas the previous SOTA reported in (Papernot et al. 2021) is 98.1% for the privacy budget of $\epsilon = 2.93$. On CIFAR-10, we match the best accuracy in (Papernot et al. 2021), namely 66.2% for the privacy budget of $\epsilon = 7.53$, with a much smaller budget of 2.20, which is an improvement in the DP-guarantee of $e^{5.33} \approx \mathbf{206}$. We also consider training DPNASNet for this larger DP budget and we get $\mathbf{72.57}\%$ for the privacy budget of $(\epsilon, \delta) = (7.53, 1 \times 10^{-5})$.

In Table 2, we compare DPNASNet with the models searched by the previous NAS methods that do not consider private learning, e.g. DARTS (Liu, Simonyan, and Yang 2019), NASNet (Zoph et al. 2018a), AmoebaNet (Real et al. 2019), and EfficientNet (Tan and Le 2019). We also compare DPNASNet with the small version of these models that have a comparable number of parameters with DPNASNet. The small models are constructed by reducing the channel numbers of convolution layers in the original models. We train these models on CIFAR-10 with DPSGD for different privacy budgets of $\epsilon = 1, 2, 3$ with $\delta = 1 \times 10^{-5}$. From Table 2, we observe that DPNASNet dramatically superior to the existing models obtained by NAS methods.

Ablation Studies

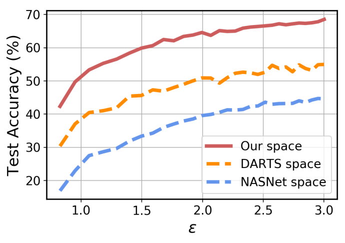
We present the ablation study results to verify the effectiveness of our search space, training method, and search strategy. More experimental results are provided in Appendix.

Effect of search space. To verify the effectiveness of the search space, we replace our search space with two search spaces widely used in previous NAS works, which are NASNet search space (Zoph et al. 2018a) and DARTS search space (Liu, Simonyan, and Yang 2019). In NASNet, the predefined cell structure is different from ours. The cells are divide into two groups, e.g. Normal Cell and Reduction Cell. In a cell, each internal node has two inputs selected from the outputs of previous nodes or previous two layers. In DARTS, the cells are also divided into two groups as NASNet but its predefined cell structure is the same as ours. The candidate operations in both of these two search spaces are much different from ours as they do not involve different types of activation functions. More details about these two search spaces could be found in (Zoph et al. 2018a) and (Liu, Simonyan, and Yang 2019). We apply our search process on these two spaces and compare the resulted models with DPNASNet. As shown in Figure 3(a), it is obvious that DPNASNet is superior to the models searched on these two spaces, which indicates that our search space is more effective.

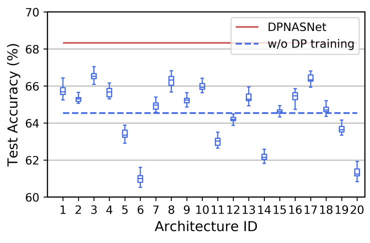
Effect of training method. To verify the effectiveness of our training method for sampled architectures during the search process, we run our search process by replacing our training method with the ordinary SGD. Then we use the resulted controller to generate 20 architectures and train all these architectures with DPSGD 10 times on CIFAR-10 for

Models	#Params	Acc(%) under budget ϵ		
		$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$
NASNet (Zoph et al. 2018a)	3.3M	26.29 ± 0.45	33.44 ± 0.45	40.77 ± 0.51
NASNet-S† (Zoph et al. 2018a)	0.6M	24.51 ± 0.32	36.56 ± 0.40	41.72 ± 0.48
AmoebaNet (Real et al. 2019)	3.2M	39.21 ± 0.43	46.36 ± 0.47	51.46 ± 0.49
AmoebaNet-S † (Real et al. 2019)	0.5M	38.47 ± 0.32	49.01 ± 0.46	53.84 ± 0.46
DARTS (Liu, Simonyan, and Yang 2019)	3.3M	36.08 ± 0.46	43.50 ± 0.52	48.79 ± 0.50
DARTS-S† (Liu, Simonyan, and Yang 2019)	0.6M	37.48 ± 0.29	47.91 ± 0.42	51.99 ± 0.42
EfficientNet (Tan and Le 2019)	3.6M	40.17 ± 0.42	48.93 ± 0.45	54.20 ± 0.48
EfficientNet-S† (Tan and Le 2019)	0.6M	41.99 ± 0.32	49.76 ± 0.36	54.63 ± 0.43
DPNASNet	0.5M	$\mathbf{52.95} \pm 0.35$	$\mathbf{65.03} \pm 0.42$	$\mathbf{68.33} \pm 0.45$

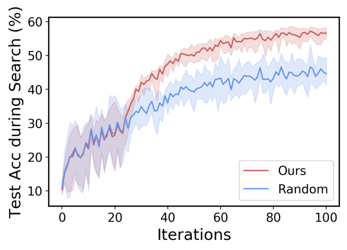
Table 2: Comparison with existing NAS searched models on CIFAR-10. † indicates the models that have comparable number of parameters with DPNASNet. The models are trained with DPSGD for the privacy budgets of $\epsilon = 1, 2, 3$ and $\delta = 1 \times 10^{-5}$.



(a) Effect of search space.



(b) Effect of training method.



(c) Effect of search strategy.

Figure 3: Ablation studies on CIFAR-10. (a) Comparing the utility of models searched on our search space, the search space of DARTS, and the search space of NASNet, using our search process. (b) Comparing the utility between DPNASNet and the models searched without using our training method. The models are trained for the privacy budget of $(\epsilon, \delta) = (3, 1 \times 10^{-5})$. (c) Comparison of the test accuracy during the search process of RL-based search and random search.

the privacy budget of $(\epsilon, \delta) = (3.0, 1 \times 10^{-5})$. The results are presented in Figure 3(b). Each blue box represents the 10 times results of each architecture and the blue dotted line is the average accuracy of those 20 architectures. We can see that without using our training method, the resulted architectures still perform better than handcrafted models, e.g. 59.07% test accuracy of CNN-Tanh (Papernot et al. 2021), but all the sampled models are less effective than DPNASNet searched by using our training method.

Effect of search strategy. To show the effectiveness of the search algorithm, we replace the RL-based search method in our search process with random search and draw the test accuracy of 10 sampled architectures after each epoch in Figure 3(c). We find that, after warm-up (the first 25 epochs), the test accuracy of models sampled by RL-controller keeps increasingly higher than that from the random sample, which means the search algorithm of our DPNAS is effective.

Transferability of searched architectures. The main results for MNIST, FashionMNIST, and CIFAR-10 are obtained by networks searched on these three datasets, respectively. To show the transferability of searched architectures, for each dataset, we construct a model with the cell searched on this dataset and evaluate it on the other two datasets. As showen in Table 3, the cell searched on one dataset also perform well when evaluated on other datasets. They can achieve compara-

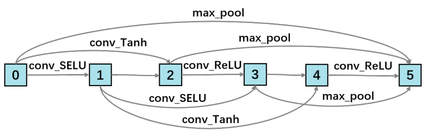
Search Dataset	Evaluation Dataset (Acc, %)		
	MNIST	FashionMNIST	CIFAR-10
MNIST	98.57	87.92	67.95
FashionMNIST	98.58	88.09	68.06
CIFAR-10	98.66	87.94	68.33

Table 3: Transferability of resulted architectures. Each model is constructed by using the cells searched on one dataset (search dataset) and evaluated on other two datasets (evaluation datasets). The models are trained with DPSGD for a privacy budget of $(\epsilon, \delta) = (3, 1 \times 10^{-5})$.

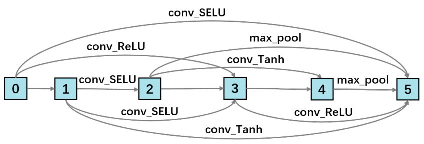
ble accuracy with the cells searched on the evaluation dataset. The transferability of NAS searched architectures have been verified in (Zoph et al. 2018b). Our experiments validate that this transferability still holds when private training is taken into considering in the search process.

Architecture Analysis and Discussions

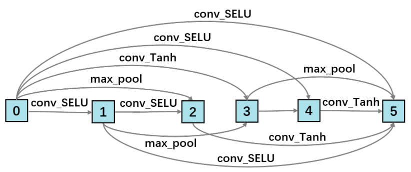
Our searched networks for DPDL show meaningful patterns that are distinct from the networks searched for non-private image classification. Figure 4 shows the cell architecture of our searched DPNASNet. We conduct qualitative and quantitative analysis on the searched architectures by DPNAS and



(a) Searched on MNIST



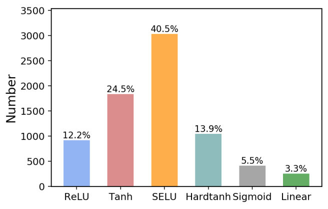
(b) Searched on FashionMNIST



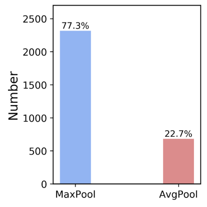
(c) Searched on CIFAR-10

Figure 4: Cell architectures of DPNASNet.

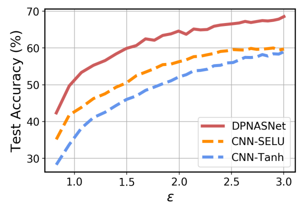
sum up several observations for designing private-learning-friendly networks as following.



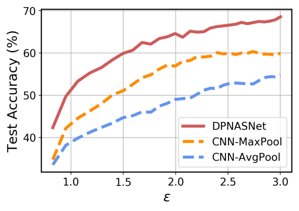
(a) Proportion of activation functions.



(b) Proportion of pooling functions.



(c) Comparison between SELU and Tanh.



(d) Comparison between AvgPool and MaxPool.

Figure 5: Analysis of the generated cells on CIFAR-10.

SELU out-performs Tanh for DPDL. From Figure 4, we observe that SELU is the most frequently used activation function in the resulted architectures. To statistically analyze the occurrence frequency of each activation function, we use the trained controller to sample 1000 architectures and count the number of each activation function in these architectures. As shown in Figure 5(a), the occurrence frequency of SELU is much higher than others. Tanh also frequently appears and its effectiveness for DPDL has been verified in previous work (Papernot et al. 2021). As SELU appears more frequently than Tanh in our results, we wonder *whether SELU is better than Tanh to construct networks for DPDL*. To answer this question, we employ the simple CNN model from (Papernot et al. 2021) with SELU and Tanh as activation function respectively to build two models, CNN-SELU and CNN-Tanh. From Figure 5(c), we find that CNN-SELU consistently out-performs CNN-Tanh. Papernot et al. (Papernot et al. 2021) argue that the reason for Tanh out-performing ReLU for DPDL is that Tanh is bounded while ReLU is unbounded. However, we find that SELU is yet efficient for DPDL, although it is also unbounded. Our intuitive explanation for this is that the activation functions that can retain negative values of their inputs could be more suitable for DPDL. Here, we try to experimentally verify this intuition. From Table 4, we observe that the activation functions that can retain negative values out-perform those only having non-negative values in their outputs. This result is consistent with our intuition. We leave the theoretical explanation for future studies.

Function	Has Negative	Bounded	Test Acc
ReLU	×	×	51.02%
ReLU6	×	✓	53.03%
ELU	✓	×	57.35%
SELU	✓	×	60.16%
Tanh	✓	✓	59.21%
HardTanh	✓	✓	58.46%
LeakyReLU	✓	×	59.32%

Table 4: Comparing the utility of CNN models with different activation layers trained with DPSGD on CIFAR-10.

MaxPool performs better than AvgPool. From Figure 4, we also observe that MaxPool appears more frequently than AvgPool. We also do a statistic on the occurrence frequency of each pooling function by using a similar statistical method for activation functions. From Figure 5(b), we find that the portion of MaxPool used in the resulted architectures is much higher than that of AvgPool. Based on this observation, we are curious about *whether MaxPool is better than AvgPool for DPSGD trained models' utility.* To figure it out, we conduct a comparison on two simple CNN models. One model employs MaxPool for all its pooling operations, and the other uses AvgPool for all pooling layers. Both of these two models are trained with DPSGD on CIFAR-10 with the same settings. Figure 5(d) shows that MaxPool is a better selection than AvgPool for the DPSGD trained models.

Conclusion

We demonstrate that the model architecture has a significant impact on the utility of DPDL. We then present DPNAS, the first framework of automatically searching models for DPDL. We specially design a novel search space and propose a DP-aware training method in DPNAS to guide the searched models to be adaptive to DPSGD training. The searched model DPNASNet consistently advances the SOTA accuracy on different benchmarks. Finally, we analyze the generated architectures and provide several new findings of operation selection for designing private-learning-friendly DNNs.

Acknowledgment

This work was supported in part by National Key Research and Development Program of China (No.2020AAA0103402), National Natural Science Foundation of China (No.62106267), Strategic Priority Research Program of Chinese Academy of Sciences (No.XDA27040300), and Jiangsu Key Research and Development Plan (No.BE2021012-2).

References

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep Learning with Differential Privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.

Baker, B.; Gupta, O.; Naik, N.; and Raskar, R. 2017. Designing Neural Network Architectures using Reinforcement Learning. *International Conference on Learning Representations*.

Cai, H.; Zhu, L.; and Han, S. 2018. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. *International Conference on Learning Representations*.

Dong, X.; and Yang, Y. 2019. Searching for a Robust Neural Architecture in Four GPU Hours. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1761–1770.

Dwork, C.; and Roth, A. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4): 211–407.

Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Huang, G.; Liu, Z.; and Weinberger, K. Q. 2017. Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. PMLR.

Kearns, M.; and Roth, A. 2020. Ethical algorithm design. *ACM SIGecom Exchanges*, 18(1): 31–36.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Klambauer, G.; Unterthiner, T.; Mayr, A.; and Hochreiter, S. 2017. Self-Normalizing Neural Networks. In *NIPS*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images.

LeCun, Y.; Bengio, Y.; and Hinton, G. E. 2015. Deep learning. *Nat.*, 521(7553): 436–444.

LeCun, Y.; and Cortes, C. 2005. The mnist database of handwritten digits.

Lee, J.; and Kifer, D. 2018. Concentrated Differentially Private Gradient Descent with Adaptive per-Iteration Privacy Budget. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; and Kavukcuoglu, K. 2018. Hierarchical Representations for Efficient Architecture Search. In *International Conference on Learning Representations*.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. *International Conference on Learning Representations*.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; and Arcas, B. A. Y. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

Mironov, I.; Talwar, K.; and Zhang, L. 2019. Rényi Differential Privacy of the Sampled Gaussian Mechanism. *ArXiv*, abs/1908.10530.

Papernot, N.; Thakurta, A.; Song, S.; Chien, S.; and Erlingsson, Ú. 2021. Tempered Sigmoid Activations for Deep Learning with Differential Privacy. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32: 8026–8037.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient Neural Architecture Search via Parameter Sharing. In *ICML*.

Pichapati, V.; Suresh, A. T.; Yu, F.; Reddi, S. J.; and Kumar, S. 2019. AdaCliP: Adaptive Clipping for Private SGD. *ArXiv*, abs/1908.07643.

Real, E.; Aggarwal, A.; Huang, Y.; ; and Le, Q. V. 2019. Regularized Evolution for Image Classifier Architecture Search. In *AAAI*.

Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y.; Tan, J.; ; Le, Q. V.; and Kurakin, A. 2017. Large-Scale Evolution of Image Classifiers. *International Conference on Machine Learning*.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership Inference Attacks Against Machine Learning Models. 3–18.

Tan, M.; and Le, Q. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, 6105–6114. PMLR.

Thakkar, O.; Andrew, G.; and McMahan, H. B. 2019. Differentially Private Learning with Adaptive Clipping. *ArXiv*, abs/1905.03871.

Tramèr, F.; and Boneh, D. 2021. Differentially Private Learning Needs Better Features (or Much More Data). *International Conference on Learning Representations*.

Vadhan, S. 2017. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, 347–450.

Wang, J.; and Zhou, Z. 2020. Differentially Private Learning with Small Public Data. In *AAAI*.

Wang, Y.; Balle, B.; and Kasiviswanathan, S. 2019. Subsampled Rényi Differential Privacy and Analytical Moments Accountant. In *AISTATS*.

Wu, B.; Zhao, S.; Sun, G.; Zhang, X.; Su, Z.; Zeng, C.; and Liu, Z. 2019. P3SGD: Patient Privacy Preserving SGD for Regularizing Deep CNNs in Pathological Image Classification. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2094–2103.

Wu, Y.; and He, K. 2018. Group Normalization. In *ECCV*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *ArXiv*, abs/1708.07747.

Xie, L.; Lin, K.; Wang, S.; Wang, F.; and Zhou, J. 2018. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*.

Yu, D.; Zhang, H.; Chen, W.; and Liu, T. 2021. Do Not Let Privacy Overbill Utility: Gradient Embedding Perturbation for Private Learning. *International Conference on Learning Representations (ICLR)*.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. *British Machine Vision Conference*.

Zhou, Y.; Wu, Z.; and Banerjee, A. 2021. Bypassing the Ambient Dimension: Private SGD with Gradient Subspace Identification. *International Conference on Learning Representations (ICLR)*.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep Leakage from Gradients. *Neural Information Processing Systems (NeurIPS)*.

Zoph, B.; and Le, Q. V. 2016. Neural Architecture Search with Reinforcement Learning. *ArXiv*, abs/1611.01578.

Zoph, B.; Vasudevan, V.; Shlens, J.; ; and Le, Q. V. 2018a. Learning Transferable Architectures for Scalable Image Recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8697–8710.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018b. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710.