A Privacy-Preserving Deep Learning Approach for Face Recognition with Edge Computing

Yunlong Mao
Nanjing University

Shanhe Yi
College of William & Mary

Qun Li
College of William & Mary

Jinghao Feng
Nanjing University

Fengyuan Xu
Nanjing University

Sheng Zhong
Nanjing University

Abstract

Deep convolutional neural networks (DNNs) have brought significant performance improvements to face recognition. However the training can hardly be carried out on mobile devices because the training of these models requires much computational power. An individual user with the demand of deriving DNN models from her own datasets usually has to outsource the training procedure onto a cloud or edge server. However this outsourcing method violates privacy because it exposes the users' data to curious service providers. In this paper, we utilize the differentially private mechanism to enable the privacy-preserving edge based training of DNN face recognition models. During the training, DNN is split between the user device and the edge server in a way that both private data and model parameters are protected, with only a small cost of local computations. We show that our mechanism is capable of training models in different scenarios, e.g., from scratch, or through fine-tuning over existed models.

1 Introduction

People with mobile devices such as smartphones, Google glasses or HoloLens can sense the environment and use collected sensitive data (image, sound, and more) to train a deep convolutional neural network (DNN) for various applications, e.g., face recognition of people met before. Usually, there are two ways for these mobile devices to train a DNN. The first one is to send all sensitive data to a central server (or cluster) which has enough computing power. The second one is to perform distributed training with a local DNN being trained on each device. Obviously, the first one is more suitable for mobile devices with limited computing resources. But user's private data will be violated seriously if the central server is untrusted [3]. The second one has higher requirements for devices' computing power. Assuming that it's possible for mobile devices to perform distributed training given powerful equipments like neural engines built in iPhone X and Tensorflow Lite training framework for mobile devices provided by Google, user's private data will still be violated by an active adversary even with efficient privacy-preserving schemes applied [13, 7]. To meet privacy and resource requirements, we offer a more suitable option for privacy-aware DNN training for mobile devices aided by edge computing. In particular, we will show how this can be done through the popular application of DNN based face recognition.

Training a multi-label DNN entirely on a mobile device is daunting due to resource limitations. Meanwhile, users' privacy cannot be guaranteed in client-server model. An untrusted server can peek at users' images containing confidential information. Even if cryptographic tools or obfuscation schemes are applied to protect images, membership inference attacks against a trained model could still be achieved [14, 6] even without training data present. There is still a huge gap between deploying fully functional DNNs on devices and reality. This paper tries to fill the gap by introducing a new client-server model based DNNs training scheme in privacy-preserving manners. Users within the scheme just do limited computation to train very deep neural networks on off-the-shelf devices and aided edge server with users' private data preserved. For example, smartphone users can train their own multi-label face recognition models on a edge server with their private images. Another possible application is to use smart cameras and edge computing server [15] to make real-time neighborhood surveillance with residents' privacy preserved. It will just take seconds to process a batch on mobile client.

Privacy issue in deep learning has been a hot topic recently because of severe privacy leakage results [14, 7, 6]. Several efficient privacy-aware DNN training mechanisms have been proposed. A differentially private (DP) gradients computing mechanism is designed in [1] to protect locally trained parameters. Privacy-preserving parameters aggregation for distributed learning has been studied in [13, 3]. A DP parameters updating mechanism is introduced in [13], while a secure parameters aggregation mechanism based on combing masking technique and threshold secret sharing is proposed in [3]. Targeting at privacy-preserving fine-tuning, [11] migrates the learning process from a client to a server after mixing basic features extracted by clients with noise. However this scheme focuses on fine-tuning only. None of these existed mechanisms can protect mobile user's private data in a client-server training model very well.

The basic idea of our scheme is based on an important observation, that a DNN can be split inside between two successive layers and deployed two partitions on different locations without hazarding the optimization. To minimize the cost of mobile users, we partition DNN after the first convolutional layer. Deploy the first part on user side while the second part on the edge server side. We keep the output of the user part privacy-preserving and feed the output of the user part as the input of the second part on server side. We avoid cryptographic tools so that we can keep user side lightweight. Meanwhile, we use the differential privacy to ensure a strong privacy guarantee for user's confidential datasets. In general, our contribution can be summarized as,

- We have designed a new privacy-preserving algorithm to calculate DP activations for convolutional layers. Based on this algorithm, we have designed a new privacy-preserving DNN training scheme for face recognition.

- We have implemented a privacy-preserving VGG-Face network for face recognition[1]. We evaluate our scheme for training and fine-tuning tasks using public datasets. Evaluation results show that both privacy and accuracy are satisfactory.

2 Threat Model

In a client-server model, the client is supposed to send training data to the server, and the server then performs training for the client. The privacy considered here is about client's confidential data. So the privacy property that we want to guarantee is, no semi-trusted server can tell whether a specific labeled face image is in client's datasets or not if the server has not seen this labeled image in any public dataset.

The semi-untrusted server is considered to be curious. This adversary type is similar to [13, 1]. However ours has a view of the whole learning procedure including input, output and parameters of every network layer which is on the server side, while [13] feeds the adversary with selected gradients and the adversary of [1] only has a view of the model's parameters. This means that our adversary is relatively powerful and hard to defend against.

Generally, the curious server can violate client's privacy by copy-and-embezzling client's input images directly, or infering client's input images from learned model parameters [6]. We will deal with these two attacks in this paper. Other kinds of attacks will be left for further study.

3 Differentially Private Deep Convolutional Neural Network

In neural networks, each hidden layer can be seen as a separate unit taking previous layer's output as its input. Input of the first layer in DNNs is special because it is original image data. It is not recommended to use high noise to perturb images directly because the utility of images will be damaged seriously [10]. However we consider to partition the network inside at some specific convolutional layer instead of the input layer because layers in DNNs are loosely coupled units. Once we select one layer to partition the whole network into two parts for the client and the edge server, the client can hide all intermediate results and input from the server. All information that the server needs to know for the forward passing is the output of the last layer in the client's part. As for backward passing, the client can compute gradients and update parameters by following the chain rule as long as the server provides the partial of loss with respect to client's output. Without the loss of generality, we will introduce our scheme with a simple partitioning strategy where the client holds the first convolutional layer (with ReLU attached) and feeds the output to an untrusted edge server, who will succeed the client to finish following layers in a pre-designed DNN. A partitioning example for the VGG-Face network is illustrated in Figure.1.

In our scheme, client sends output activations of the first convolutional layer instead of raw images to the edge server. Artificial noise will be added to output activations. The addition of artificial noise can prevent adversary edge server from reversing activations in case that the edge server learns parameters of the first convolutional layer somehow (e.g. if client does fine-tuning on any public pre-trained model). We are going to show that if we use DP Gaussian noise then model parameters' updating will also be privacy-preserving.

3.1 Differentially Private Activations

As shown in Figure.1, when we partition VGG-Face network into two parts, except for the first convolutional layer on the client, all other layers being deployed on

[1]We use VGG-Face network as a study example. Our scheme is based on VGG-16 architecture, but not limited to specific image processing techniques.

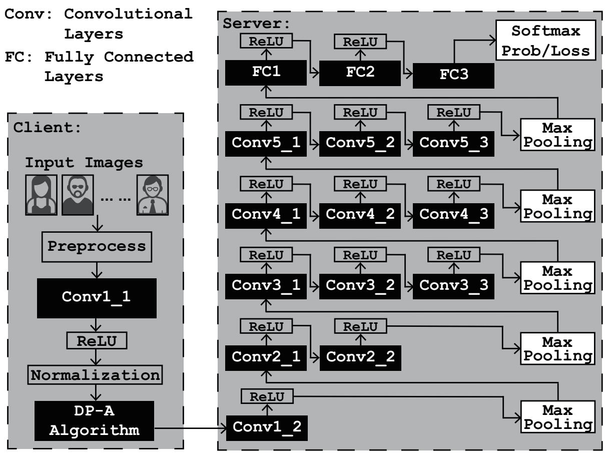


Figure 1: After the partitioning, the first convolutional layer with our DP-algorithm is deployed on the client while the rest part of VGG-Face network will be deployed on an edge server.

the edge server are the same as in original VGG-Face network. When the client communicates with the edge server, in client's view, server's partial network can be seen as a black-box function and vice versa. In forward passing, the first layer on the client can be seen as a composite function, whose input is client's datasets and output is volume of activations. Activation function denoted by f is actually the composition of kernels in convolutional layer, ReLU and local response normalization unit. In the first convolutional layer, we assume that there are m kernels with size $l \times l \times r$, where r is number of color channels. When one training example s in client's private datasets goes into $f^i, i \in [1,m]$, each activation is generated corresponding to one spatial position (e.g. (x,y)) on the surface of face image s. Then function f^i (f will be used equivalently if no ambiguity is caused.) can be defined as,

$$f^i(s(x,y)) = a^i_{(x,y)}/(\gamma + \alpha \sum_{j=\max(0,i-\frac{u}{2})}^{\min(m-1,i+\frac{u}{2})} (a^j_{(x,y)})^2)^\beta, \tag{1}$$

where u, α, β, γ are constants which should be determined using a validation set empirically (Please refer to ImageNet paper [8] for more detailed information if interested). $a^i_{(x,y)}$ is the activation generated by i-th kernel at position (x,y).

To protect confidential datasets of the client, we need to guarantee that output activations of function f for every single image is privacy-preserving. Function f can be regarded as a specific query on client's datasets $d \in \mathscr{D}$. To construct a (ε,δ)-DP mechanism for f with Gaussian noise, sensitivity of f on adjacent datasets d, d' should be clarified. Based on the definition of function sensitivity [4] and ReLU's output activation $a^i_{(x,y)} \geq 0$, we can define f's sensitivity as S_f. Given sensitivity S_f, when f is applied at the same spatial position (x,y) on training face images, we can use $c^i_{(x,y)} + \mathscr{N}(0, S_f^2\sigma^2)$ to replace $c^i_{(x,y)}$ as output of f. Since $0 \leq a^i_{(x,y)}(s) < 1, \forall s \in d$ after we pre-process input images, we can have $0 \leq S_f < 1/\sqrt{2}$ when we select parameter $u = 5, \alpha = 1, \beta = 0.5, \gamma = 2$. Although parameters including stride, padding and kernel size should usually be selected to give perfect alignment, we use rounding operator here just in case. In this way, we will have a (ε,δ)-DP mechanism for activations of convolutional layer.

3.2 Privacy-Preserving Weights Updating

After DP activations are transmitted to the edge server, client's task in forward passing is finished. To continue training, the edge server will run subsequent training process from the second convolutional layer with activations received as its input. There is no additional modification needed for our scheme to be deployed on the edge server. However, it's important to ensure that when all activations generated by DP activation algorithm are compounded through server's convolutional layers, the output will still be privacy-preserving. The output of each layer can be seen as combination of DP activations. The total loss of network prediction can also be seen as a composed mechanism of multiple DP mechanisms. But the real challenge is how to guarantee the privacy loss of our composed mechanism can be tightly bounded instead of a simple composition of multiple DP mechanisms.

When multiple DP mechanisms are composed, the privacy guarantee normally goes down. Basically we encounter an adaptive composition situation suggested in Dwork's boosting theory [5]. If we directly follow the adaptive composition theory, the prediction mechanism of this deep learning network should be $(\varepsilon', pq\delta + \delta')$-DP in adversary's view, where p, q are dimensions of activations for each kernel, $\varepsilon' = \varepsilon\sqrt{2pqln(\frac{1}{\delta'})} + pq\varepsilon(e^\varepsilon - 1), \delta' > 0$. We find that in DNN training, we can actually achieve a tighter bound than simple composition theory.

Assume that we are looking at the first iteration of training session. All weight parameters in convolutional layers are initialized by sampling from normal distribution $N(0,0.01)$. For prediction mechanism $M_p(S)$, the definition of adjacent datasets are still the same as before, but the element of dataset is one integral face image instead of just one group of pixels. This means we group dataset families for activating function into one new dataset family D. If we process two identical batch generated from D with M_p, then output loss for each sample should be the same. So is the gradient estimation. To show that M_p can be privacy-preserving, we need bound probabilistic differential when M_p applies to two adjacent datasets $d, d' \in D$.

Based on this, we can also count privacy loss in backward passing. We use mini-batch stochastic gradient descent (SGD) method to compute gradients. Simply, if we want to update parameters W in t-th iteration, we can use $W_t = W_{t-1} - lr * g_t$, where g_t is an average estimation cross the mini-batch to the gradient, lr indicates learning rate. One trip of backward passing mainly consists of gradient computing and variable updating. When we have gradient computed, variable updating will be trivial. So the part that really matters is gradient computing. To secure user's privacy against revealing from model's parameters, we here prove that our privacy-preserving weights updating satisfies differential privacy. Since we can control privacy loss with ε, the privacy level can be designed as a flexible hyperparamter for user to choose (within a feasible range).

3.3 Fine-tuning

Many fine-grained, well-trained face recognizing models are public for use. Public face recognizing models such as [12, 2] have contributed a lot to accelerate development of both academia and industry. But how to make greater use of these public models is still an open question. One of the major concerns is that public models are pre-trained using fixed datasets. Datasets for training will never be big enough to satisfy all applications, especially when we want to recognize some identities belonging to private datasets. Also, some people may not want to train an entire DNN from scratch in practice, because it is difficult to get a proper dataset of sufficient volume and training procedure usually takes long time. Instead, it is a good idea to do fine-tuning [16] on a pre-trained DNN with one's own target datasets.

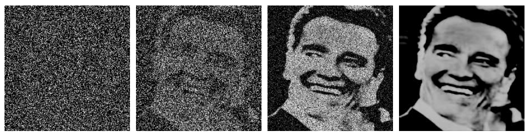

(a) $\varepsilon = 1$ (b) $\varepsilon = 5$ (c) $\varepsilon = 20$ (d) w/o DP-A

Figure 2: Output activations with DP-A and without DP-A when we fine-tune VGG-Face model on LFW dataset for $T = 10000$, $\delta = \delta' = 1e-5$, $u = 5, \alpha = 1, \beta = 0.5, \gamma = 2$.

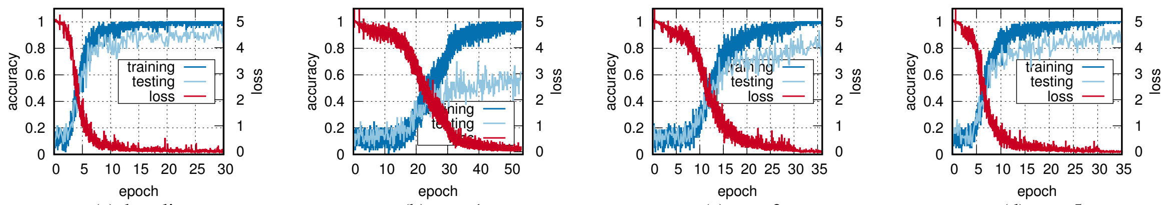
The scheme we proposed is capable of privacy-preserving fine-tuning. Intuitively, fine-tuning with our scheme is a special case of training. Here we will focus on the situation where only output layer parameters will be tuned. Recall that each output activation of DP activation algorithm is (ε, δ)-DP. But in adversary's view, strength of privacy will degrade because images with the same noise distribution may appear in multiple rounds with regarding to one identity. We show some results of output activations of the first convolutional layer with different epsilon values in Figure.2. All activation volumes showed in this figure have been multiplied by 255 with negative values removed to make them more visual-friendly. We can tell from this figure when $\varepsilon = 1$, utility of images is damaged significantly. When ε is around 5, input images can be well preserved. But when ε reaches $14 \sim 20$, privacy is almost gone.

4 Evaluation

We have implemented our privacy-preserving DNN for face recognition with TensorFlow framework. The dataset we use is Labeled Face in the Wild dataset (LFW) [9], which has been regarded as a standard benchmark for unconstrained face recognition. According to [12], training data for published model does not contain LFW dataset. So some subsets of LFW dataset will be used to perform training and fine-tuning. To allow further comparison with other work, we use the same training parameters as described in original VGG-Face training process [12]. All convolutional layers have $stride = 1, pad = 1$. Max pooling size is 2×2. Mini-batch size is 64. Momentum coefficient is 0.9. Learning rate is initialized with 0.01 and exponentially decayed with factor 0.1. Besides, for the local normalization unit that we use in the partitioning layer, $u = 5, \alpha = 1, \beta = 0.5, \gamma = 2$. We first perform experiments with the first convolutional layer as partitioning position. Then we will show how different partitioning positions affect training process exclusively.

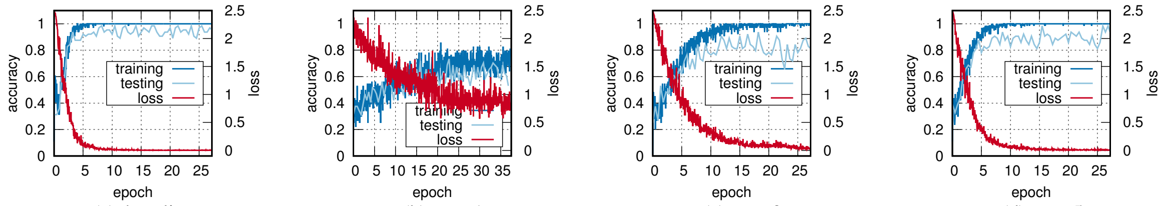
We filter LFW dataset by choosing persons with no less than 10 images. This leads to a subset which contains 158 identities, 4324 labeled face images. We split images of each person in a 9:1 ratio to define training set and testing set. For each iteration, images in training set are randomly sampled to compose a batch. To evaluate training progress, we record output of loss function, training accuracy and testing accuracy. In Figure.3, training results of different epsilon values are shown. Training session with no noise added is recorded as baseline. The smaller the epsilon is, the higher the noise is. It is obvious that small epsilon makes training process more unstable. It will take more epochs to train with higher noise than lower noise to achieve the same training accuracy or the same loss. When $\varepsilon = 2 \sim 5$, our scheme can achieve strong privacy and high accuracy.

To perform fine-tuning on pre-trained VGG-Face model, weights in pre-trained VGG-Face model are loaded before training. The network is still partitioned at the first convolutional layer. Fine-tuning result in the same setting with no noise added is seen as our baseline



(a) baseline (b) $\varepsilon = 1$ (c) $\varepsilon = 2$ (d) $\varepsilon = 5$

Figure 3: Training results of accuracy and loss with regard to different epsilons.



(a) baseline (b) $\varepsilon = 1$ (c) $\varepsilon = 3$ (d) $\varepsilon = 5$

Figure 4: Fine-tuning results of accuracy and loss with regard to different epsilons.

of tuning. Results of tuning with different epsilons are shown in Figure.4. High accuracy can be achieved in early learning stage. But adding noises to activations can affect learning speed. Especially when $\varepsilon \leq 1$, noise is too large for the network to minimize its loss because trainable parameters are limited in tuning cases. However, when $epsilon = 2 \sim 5$, we can still get high accuracy and strong privacy after slightly more epochs. Specifically, it takes no more than 5 epochs for tuning with $\varepsilon = 3$ to achieve similar accuracy as the baseline.

We have deployed our implementation on a Huawei Nexus 6P phone (2GHz Qualcomm Snapdragon 810 processor, with a non-removable Li-Po 3450 mAh battery) and AWS based edge server. Evaluation result shows that a batch of training samples from LFW can be loaded on the phone under 0.4s when batch size is 8. In each iteration, forward pass for the batch will be done under 0.6s. Backward pass will cost less than 0.2s per sample. The allocated mobile memory usage will be under 500MB for processing a batch of samples. When the mobile phone is processing the first convolutional layer, battery will be consumed under 3.5mAh per minute for the batch.

5 Conclusion

We have proposed a new edge computing based DNN training architecture with DP mechanism to protect private data. Evaluation results presented in this paper ensure that applying DP mechanism on activations is a feasible solution for outsourcing training tasks to untrusted edge servers. Taking mobile user's resource cost and training accuracy into consideration, we recommend to keep just the first convolutional layer on user side. Since there are many other DNNs except for VGG networks, verifying similar corollaries and observation in other DNNs may be our future work.

References

[1] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), CCS '16, ACM, pp. 308–318.

[2] Amos, B., Ludwiczuk, B., and Satyanarayanan, M. Openface: A general-purpose face recognition library with mobile applications. Tech. rep., CMU-CS-16-118, CMU School of Computer Science, 2016.

[3] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), CCS '17, ACM, pp. 1175–1191.

[4] Dwork, C., and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci. 9* (2014), 211–407.

[5] Dwork, C., Rothblum, G. N., and Vadhan, S. Boosting and differential privacy. In *IEEE FOCS* (2010), pp. 51–60.

[6] 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* (2015), CCS '15, ACM, pp. 1322–1333.

[7] Hitaj, B., Ateniese, G., and Perez-Cruz, F. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2017), CCS '17, ACM, pp. 603–618.

[8] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS* (2012), pp. 1097–1105.

[9] LEARNED-MILLER, G. B. H. E. Labeled faces in the wild: Updates and new reporting procedures. Tech. Rep. UM-CS-2014-003, University of Massachusetts, Amherst, 2014.

[10] MOHAMMED, N., CHEN, R., FUNG, B. C., AND YU, P. S. Differentially private data release for data mining. In *ACM KDD* (2011), pp. 493–501.

[11] OSIA, S. A., SHAHIN SHAMSABADI, A., TAHERI, A., RABIEE, H. R., LANE, N. D., AND HADDADI, H. A hybrid deep learning architecture for privacy-preserving mobile analytics. *ArXiv e-prints* (2017).

[12] PARKHI, O. M., VEDALDI, A., AND ZISSERMAN, A. Deep face recognition. In *Proceedings of the British Machine Vision Conference* (2015), pp. 41.1–41.12.

[13] SHOKRI, R., AND SHMATIKOV, V. Privacy-preserving deep learning. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security* (2015), CCS '15, ACM, pp. 1310–1321.

[14] SHOKRI, R., STRONATI, M., SONG, C., AND SHMATIKOV, V. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)* (May 2017), pp. 3–18.

[15] YI, S., LI, C., AND LI, Q. A survey of fog computing: Concepts, applications and issues. In *Workshop on Mobile Big Data* (2015), pp. 37–42.

[16] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks? In *NIPS* (2014), pp. 3320–3328.