Privacy Preserving Deep Learning for Image Classification – a Case Study with Microsoft Research Celebrity Data

Meidan Bu
Stanford University
meidanb@stanford.edu

Abstract

While data provides tremendous insights, users' personal information is often exposed with limited protection. This project aims to build a privacy preserving deep learning framework that trains and updates models without directly using raw data. Using an image classification task as a case study, the results show that similar accuracy can be achieved with only sharing a small fraction of model parameters, not data.

1. Introduction

In today's world, the amount of data that we generate every day is mind-boggling. Tech companies take advantage of the massive amount of data collected from their users to deploy deep learning algorithms and to accomplish a broad range of AI tasks. While these data enable computer algorithms to uncover valuable insights, they often contain sensitive information. Many countries have set up regulations and data privacy laws to protect citizen's personal identifiable information (PII). Violations of these laws may put customers under the risk of PII leakage. In January 2019, Google is fined $57 million under Europe's Data Privacy law, after it was hit by a record-breaking 5-billion-dollar fine by EU in 2018.

In a traditional privacy violating case, users' data are collected by companies. Users have very limited control on how data are used, nor to delete them. The idea behind privacy preserving model training is that a central learning system enables users train independently on their own datasets and selectively share small subsets of their model's key parameters during training.

In this study, I implemented a privacy preserving image classification task, and compared results with two baselines: a "privacy violating" CNN that trains entire data in a central machine, and a completely isolated case when there is no update on model parameters. Furthermore, this study aims to show the tradeoff between training accuracy and privacy protection through simulation. The result shows that similar accuracy can be achieved by only sharing a small fraction of gradients.

2. Related Work

Shokri and Shmatikov [1] designed and implemented a system that multiple parties jointly train a deep neural network without sharing their input datasets. Using MNIST and SVHN datasets, they showed that by sharing only a small fraction of gradients at each gradient descent step, they can achieve almost the same accuracy as training with entire data. The work in my project is largely an implementation and modification of [1]. By implementing their idea and building a similar system with Microsoft Research (MSR) celebrity data, I reached similar conclusion.

Google announced TensorFlow Privacy Library on March 6th while I was working on this project (https://venturebeat.com/2019/03/06/google-releases-tensorflow-privacy-a-library-for-training-ai-models-with-strong-privacy-guarantees/). An earlier work by Google [2] built a deep learning framework with differential privacy, and evaluated their approach using MNIST and CIFAR-10. They achieved 97% training accuracy for MNIST, and 73% for CIFAR-10.

3. Data

People's cellphones often store a lot of their personal sensitive data, especially photos. To demonstrate the idea of this project, I designed a hypothetical case where a photo app can collect photos from people's cellphones. Photos consists of four types: men, women, dogs and cats. Data used in this project come from two sources:

- Men and women data are from Microsoft Research Celebrity Faces Open Data: https://www.microsoft.com/en-us/research/project/msra-cfw-data-set-of-celebrity-faces-on-the-web/.
- Kaggle dogs vs cats data (https://www.kaggle.com/c/dogs-vs-cats/data), consists 25,000 images of dogs and cats.

The original data from MSR Celebrity database has 202,792 images. They are stored in different folders by people's name. Not all pictures are of the same quality. Some of them are tricky to classify. Figure 1 provides examples of images that are in the final dataset. To avoid class imbalance issue between human and pets photos, I randomly selected 26,142 pictures from differently people. There are a total of 51,242 pictures from 4 classes. All pictures are resized to 150 × 150 pixels.

Data is then split into two parts: first is for initial model training and validation, second is for privacy preserving model updating process. Table 1 summaries sizes of the train, validation and test datasets.

Table 1. Summary of the initial model training dataset, and the dataset for Privacy Preserving Deep Learning (PPDL) parameter updating process.

	Training	Validation	Total
Men	6,426	714	7,140
Women	6,621	736	7,357
Dogs	5,850	650	6,500
Cats	5,850	650	6,500
Total	**24,747**	**2,750**	**27,497**

	Data for PPDL parameter updating
Men	5,567
Women	6,078
Dogs	6,000
Cats	6,000
Total	**23,645**

Figure 1. Examples of images in the dataset. Not all pictures are of the same quality. Some are tricky to classify.

4. Case Study

Assume Company A developed a photo app. It trains an image classification model using an initial dataset. Then the app is officially published. The classification model gets updated with new data from retail users. At the same time, users' privacy needs to be protected as much as possible. The model follows Privacy Preserving Deep Learning (PPDL) to update model parameters. The updating duration is set to be 28 days. The accuracy of classifying new photos on each day is calculated using updated model parameters on the previous day. This is essentially a "moving window" evaluation framework that simulates real world scenarios.

5. Initial model training with CNN

In the initial model training phase, I used 27,497 pictures and split them by 90% and 10% into training and validation set. Two types of model were implemented in this stage: CNN built from scratch and transfer learning using VGG16 as the base [3]. The architecture is described in Figure 2.

For each framework, accuracy was compared across SGD, RMSProp and Adam optimizers (Table 2). Each scenario runs 100 epochs. Learning rate is set to be 0.0001. The VGG16 model freezes parameters from all layers up to the last layer. I added a fully connected layer where parameters were tuned. In the second phase, VGG16 with Adam optimizer is selected to perform Privacy Preserving Deep Learning (PPDL) parameter updating.

Figure 2. Architecture of the built-from-scratch CNN model. Relu is used as the activation function.

Table 2. Performance summary for different CNN models tried in the initial model training phase. VGG16 with Adam optimizer is selected for the PPDL updating process.

	Optimizer	Training Accuracy	Validation Accuracy
CNN built from scratch model	SGD	75.9%	72.1%
	RMSProp	78.3%	74.8%
	Adam	79.2%	77.7%
CNN VGG16 fix all but last layer	SGD	83.5%	79.6%
	RMSProp	84.1%	81.4%
	Adam	85.4%	83.5%

6. Privacy Preserving Deep Learning (PPDL) Model Updating Process

The main idea behind PPDL is that instead of reading in raw training data which may contain sensitive information, the model takes gradients as the input. Assume there are N participants in the system. Participants asynchronously upload a subset of gradients $\Delta w(i)$, instead of their raw data, to a central server. The central server aggregates all gradients corresponding to each model parameter. Each participant downloads updated parameters from the server and uses them to update his local model.

Privacy is protected through:

1) The central server does not collect raw data, but only collect model updates through gradients,
2) Each participant independently shares a fraction of gradients to avoid information cross shared between participants,
3) The uploaded gradients are further protected through differential privacy with adding random noise and/or value clipping, or through only uploading largest gradient.

   Differential privacy:

   - To prevent these values from leaking too much information about the training data, random noise is added to $\Delta w(i)$. The random noise follows Laplacian distribution [3].
   - *Value clipping*: Before uploading the selected gradients $\Delta w(i)$, their values are truncated into a $[-\gamma, \gamma]$ range.

   In short, participants update $\Delta w(i)$, with values of bound $(\Delta w(i) + random\ noise, \gamma)$ before uploading it.

   Largest Gradient: each participant uploads the gradients with the biggest absolute values from the last local training.

Table 3. Hyperparameters in the PPDL process.

α	Learning rate of stochastic gradient descent
θ_u	Fraction of parameters selected to upload from local training. Participants can selectively upload a fraction of the parameters.
θ_d	Fraction of parameters selected to download from central server
γ	Bound on gradient values shared with other participants
Batch size	Number of data points that will trigger local training process (e.g. number of photos accumulated in our case)

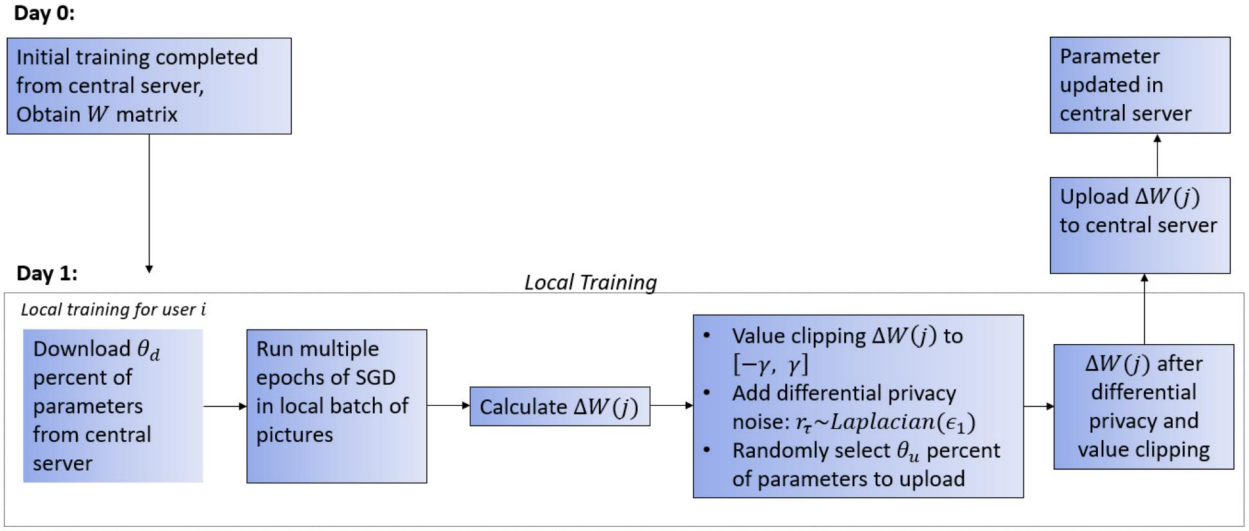

Figure 3. Diagram of local training process for a single user on a day. There are 10 users in our hypothetical scenario. Final updated model parameters on central server goes to the next day to start a new round of updating process.

Two simulations are conducted by sharing 10% and 20% of gradients each time from a single user to examine the tradeoff between accuracy and privacy protection. We specifically selected low percentages of gradients to share. This takes into consideration network and other hardware burdens for downing and uploading large number of gradients in real engineering scenario. Number of participants is 10. Batch size is set to be 32 pictures. This means that local training starts for every 32 pictures in a local machine. The clipping boundary γ is set to be 0.1. $\in$ is the privacy budget, and set to be 1.

7. Results

Results confirmed findings from [1]. By sharing only a small fraction of gradients (10%, and 20% in our case) at each gradient descent step, we can achieve similar accuracy as the privacy violating case of training in a centralized machine with 100% data. Figure 4 summarizes simulation results for two scenarios: 1) largest gradients and 2) differential privacy. There are two benchmarks in the figures below (Figure 4). The red line is centralized modeling updating using the entire 23,645 pictures. This is a privacy-violating scenario when all raw pictures are exposed to central server. The blue line is the scenario where model does not update as new pictures come in. As expected, the "no update" training has lowest accuracy, and the centralized "all data exposing" training achieves the highest accuracy.

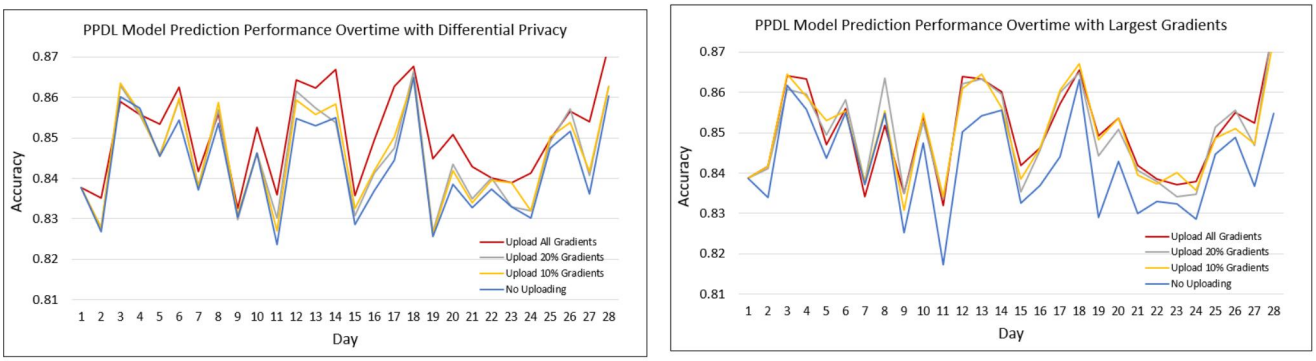

Figure 4. DDPL model prediction performance over 28 days of updating period.

8. Acknowledgement

This project is completed by Meidan Bu alone. GitHub repository with scripts is here: **https://github.com/aruba29/PrivacyPreservingDeepLearnring.** The author thanks Shawn Chai from Microsoft Windows Core Data Science Team for introducing and explaining their work in privacy preserving deep learning. This work is largely an implementation and modification of [1]. To the best of the author's knowledge, the original paper did not open source the code. The author appreciates the idea and thorough explanation of the mechanism design in the paper. The author wants to extend her gratitude to the CS230 teaching team and TA Kaidi Cao at Stanford University.

References

[1] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security,* pp. 1310-1321, 2015.

[2] M. Abai, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar and L. Zhang, "Deep learning with differential privacy," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security,* pp. 308-318, 2016.

[3] N. Ketkar, Deep Learning with Python, Apress, 2017.

[4] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," in *Foundations and Trends® in Theoretical Computer Science*, pp. 9(3–4), 211-407..