Privacy-Preserving Visual Learning

Bogdan Cebere bcebere@bitdefender.com
Mentors: Tudor Berariu tberariu@bitdefender.com
Elena Burceanu eburceanu@bitdefender.com

Bitdefender

In a nutshell

- The goal is to improve a deep learning model using users' private data, without compromising their privacy, while optimizing the use of cloud storage and network bandwidth.
- Original article: *Visual Learning Using Doubly Permuted Homomorphic Encryption*[1]

Introduction

- Standard machine learning approaches require centralizing the training data on one machine or in a datacenter.
- For models trained from user interaction with mobile devices or IoTs, a new approach was introduced: **Federated Learning**.
- Federated Learning enables devices to:
  - collaboratively learn a shared prediction model.
  - keep all the training data on device.
- This goes beyond the use of local models that make predictions on smart devices by bringing model training to the device as well.
- The **learning flow** is:
  - Your device downloads the current global model.
  - It improves the model by learning from the local data.
  - It summarizes the changes as a small focused update. Only this update to the model is sent to the cloud, where it is averaged with other user updates to improve the shared model.
  - All the training data remains on the devices, and no individual updates are stored in the cloud.
- **Federated learning advantages**:
  - Ability to learn from multiple users.
  - No need to store extra data in cloud.
  - Less network communication.

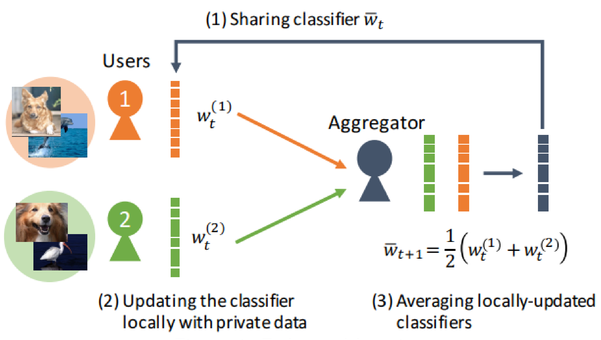

Figure 1: Federated learning

Problem description

Federated learning is vulnerable to potential **privacy leakage**. The aggregator has access to the individual weights, meaning he could reconstruct the private train data.
Several techniques were developed to **reduce the leakage**:

1. **Obfuscation with Differential Privacy**
   - **Key idea:** Adds random noise to some gradients.
   - **Limitations and potential:** The accuracy drops considerably from the added noise. There can still be data leakage. The method is pretty fast, though.
2. **Homomorphic Encryption**
   - **Key idea:** The weights are encrypted using a cryptosystem that permits summation over encrypted data, the aggregator sums the new weights and asks a third-party key-master to decrypt the sum.
   - **Limitations and potential:**. The homomorphic encryption is really slow, requires special hardware for decent performance. There is some accuracy drop from precision loss. The method should be tested on more complex datasets and with fewer variables to encrypt(for speedup).
3. **Multi-Party Computation**
   - **Key idea:** The protection is achieved by sharing secret message with different entities and requiring that these entities cooperate together in order to gain accessibility.
   - **Limitations and potential:** There is not much work done at intersection of MPC with deep learning, should be investigated.

In a privacy-enabled system, we have the following entities: the **aggregator**, the **key-master**(obfuscation/encryption helper) and **the users**.

Article approach

- [1] tries to optimize the **homomorphic encryption solution**, adding some extra-layers of security.
- The data is preprocessed through **ResNet-152** first. They focus on a few datasets, most notably **Caltech101** and **Caltech256** - 2 datasets with classes of images.
- The aggregator uses a **SVM with SGD** model, pretrains it with some public data and shares it with the users. The users train the model with their private data and employ **elastic net regularization** for sparsity.
- The users decompose the new weight matrix in a vector of non-zero values w and a matrix of their positions K. Then they encrypt w and permute the matrix K (the key and the permutation are provided by the key master). They apply a second permutation on the K matrix, negotiated by the user and the aggregator, to prevent the MitM attacks.
- The aggregator receives the encrypted data from all users, sums them and sends the result to the key-master for decryption. The result is averaged to the number of users and becomes the new global model.

Our approach

- The data is preprocessed through **ResNet-152** here as well.
- We used a **linear layer** instead of SVM. This leads to bad results when we try to enforce zero weights using elastic net regularization, and another approach is needed.
- The aggregator has the global model, pretrains it with some public data and sends it to the users.
- The users use the global weights for training only some deltas to the original model. We can apply **elastic net regularization** to the loss, i.e., the combination of L1 and L2 regularizations, to enforce sparsity on the deltas.
- After the local training is done, the user keeps only the top 1% of deltas and zeroes the rest. Non-zero deltas are encrypted and they are sent to the aggregator.
- The aggregator sums the deltas, decrypts the sum and adds the average to the global model for the updated version.
- Our approach mostly ignored the double permutation for various reasons:
  1. The first permutation is a non-standard approach to a **symmetric encryption**, trying to hide the original order of the elements.
  2. The second permutation is a non-standard approach to **SSL/TLS** protocol, trying to hide the user-aggregator communication from eavesdroppers. Neither of the permutations affect the accuracy.

Experimental results

Ref	Cal101	Cal256
Article	**89.3%**	**74.7%**
Linear layer	**89.0%**	**76.7%**

Table 1: Reference accuracy

Users	Cal101	Cal256
5	**89.6%**	**77.1%**
10	**88.9%**	**77.2%**
50	**87.7%**	**74.4%**
100	**84.0%**	**66.7%**

Table 2: Unencrypted distributed learning accuracy

Users	Cal101	Cal256	TP
5	**83%**	**72%**	**16x**
10	**81%**	**68.3%**	**30x**

Table 3: Fully encrypted learning accuracy

Users	Cal101	Cal256	TP
5	**80.7%**	**64,7%**	**2.1x**
10	**77.6%**	**63.9%**	**1.8x**
50	**74%**	**55.0%**	**8x**
100	**71%**	**43.2%**	**10x**

Table 4: Learn over weights and encrypt top 1% deltas

User count	Optimizer	Cal101	Cal256	TP
5	SGD	**83.7%**	**68.1%**	**2.1x**
5	Adam	**86.2%**	**74.1%**	**2x**

Table 5: Learn over deltas, use elastic net and encrypt top 1%

TP - Time penalty compared to unencrypted distr. network - less is better

Result interpretation

- **Homomorphic encryption** over the entire model **reduces the accuracy** with **5-6%** due to precision loss, while the training takes **16x more time**.
- If we train the model, extract the differences and encrypt top 1%, we **reduce the duration** overhead **8 times**, but the **accuracy drops** even more, with **3-8%**.
- If we train directly over deltas, using elastic net regularization, and encrypt top 1%, we **improve the accuracy** with **6-10%**. This has **better accuracy than the fully encrypted scenario**.
- In distributed scenarios, SGD finds a good accuracy after a few epochs, but drops ~5% after a few more, remaining there. Adam finds a lower accuracy, but keeps improving it while training the model.

Cybersecurity potential benefits

- IoTs network behavior profiling without uploading network traces to our cloud.
- Profiling user behavior when using a device/a product.
- Predictions over HTTP parameters/mails/private documents.

Issues and future work

- The article contains some serious **security gaps**:
  1. There is no proof of work provided to the key-master, for decrypting the input.
  2. There is no protection from spoofing fake users, in order to decrypt the real data from a single user.
  3. The permutations are a useless touch, without any impact.
- **Future work**:
  1. The architecture should be reworked security-wise.
  2. The model should be tested on more complex datasets.
  3. The model should be extended to work over cybersecurity primitives like logs, network traces(pcaps), HTTP parameters etc.
  4. Another direction for offloading the encryption part is to anonymize some data locally, before training the model.

References

[1] Yonetani, Ryo et al., *Visual Learning Using Doubly Permuted Homomorphic Encryption*. 2017.
[2] P Vepakomma et al., *No Peek: A Survey of private distributed deep learning*. 2018
[3] Z Wu, *Towards Privacy-Preserving Visual Recognition via Adversarial Training: A Pilot Study*. 2018
[4] *Our implementation- contact bcebere for access.*
[5] Slack: #private_ai