CryptoNN: Training Neural Networks over Encrypted Data

Runhua Xu, James Joshi and Chao Li
runhua.xu@pitt.edu
Background

Cloud-based ML Service

Special Scenario, e.g., Small Clinics - Computer Aided Diagnostic Application

Challenges

- Limited IT infrastructure and AI resources/experts
  - v.s.
- Privacy-sensitive data – e.g., patients’ electronic healthcare records

How to train a ML model without leaking privacy-sensitive data using cloud-based ML service?
Background

How existing privacy-preserving ML approaches work in cloud-based service

Privacy-Preserving Approaches
- Noise Addition
  - Differential Privacy, e.g., deep learning with differential privacy.
- Secure Multiparty Computation (SMC)
  - “non-crypto” based approach – garbled circuit (GC) + oblivious transfer (OT), e.g., DeepSecure, etc.
  - “crypto” based approach – homomorphic encryption (HE), e.g., CryptoNets, etc.
Background

Adoption of Privacy-Preserving Approaches in ML Cloud: Trade-off Issue

- Noise Addition
  - Differential Privacy

- Secure Multiparty Computation (SMC)
  - “non-crypto” based approach – GC + OT
  - “crypto” based approach – HE

Trade-off: privacy v.s. utility

Trade-off: computation v.s. transmission

- require large transmission volume
- require higher computation time
  --only support prediction phase
Comparison of Privacy-preserving ML Approaches

<table>
<thead>
<tr>
<th>Proposed Work</th>
<th>Training</th>
<th>Prediction</th>
<th>Privacy</th>
<th>ML Model</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy-Preserving Deep Learning (CCS) [7]</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>Deep Learning</td>
<td>Distributed*</td>
</tr>
<tr>
<td>CryptoNets [3], [9], [10], [11], [12], [13], [14], [15]</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>Covers All</td>
<td>Homomorphic Encryption (HE)</td>
</tr>
<tr>
<td>ML classification over encrypted data (NDSS) [2]</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>Limited ML†</td>
<td>HE + Secure Protocol</td>
</tr>
<tr>
<td>CryptoNN (our work)</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>Neural Networks</td>
<td>Functional Encryption</td>
</tr>
</tbody>
</table>

* This column indicates the privacy strength/guarantee such as mild approach ○ (e.g. differential privacy) and strong guarantee ● (e.g. crypto system).
† It only supports Hyperplane Decision, Nave Bayes, and Decision Trees models.
‡ The data owner trains the model by itself and outsources partial computation in a privacy-preserving setting.
* The model is trained in a distributed manner where each data owner trains a partial model on their private data.
○ It applies differential privacy method on the training data.
CryptoNN in Cloud-based ML Service

**How CryptoNN works in cloud-based ML service**

*Cloud/Server based ML (as a Service) -- Clients*

- **Choose ML Alg** → **ML Alg**
- **Collect data** → **Dataset** → **Training Phase** → **Model** → **Prediction Phase**
- **Cloud**
- **Garbled Label** → **Data** → **Predicted Label**
- **FE based approach**
- **HE based approach**
- **Predicted Label** → **Client**
Functional Encryption

In traditional encryptions scheme, *decryption algorithm reveals all or nothing*

In FE, for a function $f(\cdot)$, the decryption key $sk_f$ only *reveals partial information*, i.e., $f(x)$ instead of $x$.

- **S1**: Setting up, $pk + msk$, deliver $pk$ and hold $msk$
- **S2**: Encryption $x \rightarrow enc(pk, x)$
- **S3**: Request private key $sk_f$ from TPA
- **S4**: Generate $sk_f$ using $msk$
- **S5**: Decryption: $dec(enc(pk, x), sk_f) \rightarrow f(x)$
**Functional Encryption -- Inner-Product**

\[
f(x, y) = \langle x, y \rangle = \sum_{i=1}^{n} (x_i \cdot y_i)
\]

Alice has \( x \), and Box has \( y \).

**Goal:**
Let Bob know the result of \( f(x, y) \), but is not able to learn \( x \).

---

Secure Matrix Computation

Two Parties: $X_{l \times n}, Y_{n \times m}$, s.t. $n > m$

Alice

$$X_{l \times n}$$

$$x_{11} \cdots x_{1n}$$

$$\vdots \ \vdots \ \vdots$$

$$x_{l1} \cdots x_{ln}$$

$$ct_1 \leftarrow enc(x_1)$$

$$ct_i \leftarrow enc(x_i)$$

$$enc(X) = (ct_1, \ldots, ct_l)$$

Bob

$$Y_{n \times m}$$

$$y_{11} \cdots y_{1m}$$

$$\vdots \ \vdots \ \vdots$$

$$y_{n1} \cdots y_{nm}$$

$$sk_{f_1} \cdots sk_{f_m}$$

$$sk_{f_1} \cdots sk_{f_m}$$

Decryption Computation

$$xy_{l \times m}$$

$$\text{dec}(ct_1, sk_{f_1})$$

$$\text{dec}(ct_1, sk_{f_m})$$

$$\text{dec}(ct_l, sk_{f_m})$$
Neural Networks - Gradient Descent

\[ A^{[1]} = g(Z^{[1]}), \quad Z^{[1]} = W^{[1]}X + b^{[1]} \]
\[ A^{[2]} = g(Z^{[2]}), \quad Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]} \]
... ...
\[ A^{[l-1]} = g(Z^{[l-1]}), \quad Z^{[l-1]} = W^{[l-1]}A^{[l-2]} + b^{[l-1]} \]
\[ A^{[l]} = g(Z^{[l]}), \quad Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]} \]
\[ \hat{Y} = A^{[l]} \]
\[ E = \frac{1}{n} \sum_{i} (\hat{y}^{(i)} - y^{(i)})^2 \quad g(z) = \frac{1}{1 + e^{-z}} \]
\[ W^{[l]} = W^{[l]} - \alpha \frac{\partial E}{\partial W^{[l]}} \]
\[ \frac{\partial E}{\partial W^{[l]}} = \frac{\partial E}{\partial A^{[l]}} \frac{\partial A^{[l]}}{\partial Z^{[l]}} \frac{\partial Z^{[l]}}{\partial W^{[l]}} \]
\[ \frac{\partial Z^{[l]}}{\partial W^{[l]}} = A^{[l-1]}, \quad \frac{\partial Z^{[l]}}{\partial A^{[l]}} = A^{[l]}(1 - A^{[l]}), \quad \frac{\partial Z^{[l]}}{\partial Y} = A^{[l]} - Y \]
Neural Networks meet Functional Encryption

Input Layer
Hidden Layer
Output Layer

feed-forward
back-propagation
cost evaluation

\[ A^{[1]} = g(Z^{[1]}) , \quad Z^{[1]} = W^{[1]}X + b^{[1]} \]

\[ \begin{bmatrix} W^{[1]} \end{bmatrix} \cdot \begin{bmatrix} Enc(X) \end{bmatrix} \rightarrow \begin{bmatrix} Dec \end{bmatrix} \]

\[ sk_{f,W^{[1]}} \quad \rightarrow \quad W^{[1]}X \]

Secure Matrix Computation

Secure Matrix Computation

FE for Basic Operations or Garbled Labels

\[ W^{[l]} = W^{[l]} - \alpha \frac{\partial E}{\partial W^{[l]}} \]

\[ \frac{\partial E}{\partial W^{[l]}} = \frac{\partial E}{\partial A^{[l]}} \cdot \frac{\partial A^{[l]}}{\partial Z^{[l]}} \cdot \frac{\partial Z^{[l]}}{\partial W^{[l]}} \]

\[ \frac{\partial Z^{[l]}}{\partial W^{[l]}} = A^{[l-1]} , \quad \frac{\partial Z^{[l]}}{\partial A^{[l]}} = A^{[l]}(1 - A^{[l]}) , \quad \frac{\partial Z^{[l]}}{\partial Z^{[l]}} = A^{[l]} - Y \]
CryptoNN – Framework Overview

- Input Layer
- Hidden Layer
- Output Layer

KeyDerive Service

TPA

Public-key Distribution

mpk

msk

Client

Cloud

Secure Feed-forward

Normal Feed-forward

Secure Backpropagation/Cost Evaluation

Normal Backpropagation

One-hot Encoding

Randomly Mapping

Randomly Mapping

Option

Enc

Dec

y = 3

label

sample x
Experimental Evaluation

• Prototype Implementation
  • A scratch implementation of LeNet-5 in Python
  • *FE scheme implementation*
    • Charm-crypto (Python) – underlying numerical calculations rely on GMP library (C)

• Test platform
  • *Intel Core i7/16GB/macOS*
Experimental Evaluation

Time cost of dot-product in secure matrix computation

(a) pre-processing for encryption
(b) pre-processing for function key
(c) secure dot-product computation
(d) secure dot-product (parallelized)

x-axis: the element size
y-axis: the computing time

\[ x = (x_1, x_2, ..., x_{10}) \]
\[ y = (y_1, y_2, ..., y_{10}) \]
\[ x, y : 100 \]
\[ \rightarrow 7-8 \text{ seconds (parallelized)} \]

\[ x_i, y_i \in [1,100] \]
\[ X^{100 \times 10} \cdot Y^{10 \times 100} = XY^{100 \times 100} \]
Experimental Evaluation

LetNet-5 Neural Networks

MNIST dataset

60000 training / 10000 test

Hyper Parameters Setting

- Float Point Precision Setting: 2
  - the # of bits used after the decimal point of a floating point number
  - encoding floating point number → integer number
- Bath Size: 64
- Learning Rate: 5e-4

Comparing to baseline:

- achieving similar average batch accuracy
- costing about 14 times training time

Note this is result of submitted version.

In our follow-up work, we have an efficient implementation of decryption: $X^{1 \times 25} \cdot Y^{25 \times 1}$ from 40s → 0.2ms
Summary

• CryptoNN framework
  • Secure multiparty computation based on FE
  • CryptoNN framework
  • Concrete instance, CryptoCNN
  • Evaluation Results

• Future work
  • More efficient approaches
  • Prevent intermediate model inference attack
  • Other NN architecture
Thanks