Fast Privacy Preserved Deep Learning

author1[1], author2[2], author3[3]

(Dated: November 26, 2021)

Abstract

1 Introduction

Lung sick pateients in hospital want to take CT image of chest to detect if their lungs are sick and know the type of disease. Hospital doesn't have the ability to identify accurately. So it needs to ask a technology company who has relative deep learning model for help. But patients don't want any party except the hospital to get his chest CT images. At the same time, the technology company also don't want to let its own models leaked to any party. fast privacy preserved deep learning provides a deep learning prediction method to protect both model data of technology company and privacy data of patients.

2 Preliminaries

2.1 Semi-honest participants

Participants joining in the whole process will follow rules of agreements. They won't deliberately send wrong information. However, they are curious about another participants' input data.

2.2 Notations

We denote a shared variable x by $\langle x \rangle$. The secret share of $\langle x \rangle$ held by the $i-th$ party is denoted as $\langle x \rangle^i$, $i \in \{0, 1\}$. We use additive sharing method through whole paper. Specifically, $x = \langle x \rangle^0 + \langle x \rangle^1$.

In our paper, there are two parties to participate in calculation. We denote them by P_0 and P_1. We use P_i to refer to the two parties respectively.

If a variable $x < 0$, we set the sign of $x = -1$, else if $x > 0$, we set the sign of $x = 1$, else if $x = 0$, we set the sign of $x = 0$.

3 Comparison between two secret variables

In this section, we detail the tricks used in the following sections. In general, we introduce comparison between a shared secret variable and zero(**3.2**) and comparison between two shared secret variables(**3.3**).

3.1 multiplication of two shared variables

P_i holds $\langle x\rangle^i$ and $\langle y\rangle^i$. P_i wants to have $\langle z\rangle^i$, s.t. $\langle z\rangle^0 + \langle z\rangle^1 = (\langle x\rangle^0 + \langle x\rangle^1) \cdot (\langle y\rangle^0 + \langle y\rangle^1)$

This multiplication relys on Beaver's multiplication triples (a, b, c), where $c = a \cdot b$. The multiplication triples can be distributed by trsted setup. The trusted setup sends secret shares $\langle a\rangle^i, \langle b\rangle^i, \langle c\rangle i$.

P_i computes $\langle e\rangle^i = \langle x\rangle^i - \langle a\rangle^i$ and $\langle f\rangle^i = \langle y\rangle^i - \langle b\rangle^i$. P_i shares $\langle e\rangle^i$ and $\langle f\rangle^i$ with each other. Then, P_i compute the value of e and f.

P_i sets $\langle z\rangle^i = i \cdot e \cdot f + f \cdot \langle a\rangle^i + e \cdot \langle b\rangle^i + \langle c\rangle^i$.

3.2 Comparison between a shared secret variable and zero

In convolution neural network, ReLU function is used many times as an activation function. ReLU function needs to campare the input variable and 0.

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leqslant 0 \end{cases}$$

Here we design a protocol to compute the sign of shared variable x secretly.

In this paper we introduce the comparison (u, p, q). s.t. $u = \langle u\rangle^0 + \langle u\rangle^1 = p \cdot q$.

The general idea of this section is multiply x by a random number u to blind the value of x. So that $x \cdot u = x \cdot p \cdot q$ Via **3.1**, P_i gets $\langle z\rangle^i$, where $z = u \cdot x$. Then P_0 shares the sign of $1/p$ and P_1 shares the sign of z/q. Since $x = \frac{x \cdot u}{p \cdot q} = \frac{z}{p \cdot q}$, the sign of x can be easily calculated with product of the sign of $1/p$ and z/q.

Concretely, The protocol is run as follow.

Distribution of triples The trusted setup TTP distributes one multiplication triple and one comparison triple to P_0 and P_1. As for comparison triple, TTP randomly generate triple (u, p, q), s.t. $u = \langle u\rangle^0 + \langle u\rangle^1 = p \cdot q$. Then TTP sends $(\langle u\rangle^0, p)$ to P_0, and $(\langle u\rangle^1, q)$ to P_1.

Blinding the additionP_0 and P_1 run the multiplication protocol to get $\langle z\rangle^0 + \langle z\rangle^1 = (\langle u\rangle^0 + \langle u\rangle^1) \cdot (\langle x\rangle^0 + \langle x\rangle^1) = u \cdot x$. P_i shares the value of $\langle z\rangle^i$ and computes $z = \langle z\rangle^0 + \langle z\rangle^1 = u \cdot x$.

Computing the signP_0 sets $s^0 = 1/p$ and P_1 sets $s^1 = z/q$. It can be found now :

$$s^0 \cdot s^1 = \frac{z}{p \cdot q} = \frac{x \cdot u}{p \cdot q} = x$$

So the sign of x equals to that of $s^0 \cdot s^1$.

P_i shares the sign of s^0 and s^1. Finally, the sign of x is equal to the sign of s^0 and s^1.

Security proof: Intuitively, after running a protocol completely, any participants shouldn't have any information about true value of the shared variable except for information leaked by the protocol result which cannot be avoided. Specifically, If the result of the protpcol is $x > 0$. P_0 and P_1 shouldn't have any information about x except for $x > 0$. If that holds, we think that the protocol is secure.

Since P_0 and P_1 are dual. We just discuss P_0.

Recall what information P_0 can get from the protocol: P_0 holds value of $(\langle x\rangle^0, \langle u\rangle^0, p, z$,the sign of s^1,the sign of $x)$. Moreover, P_0 knows two equations and two signs of variables.

$$\begin{cases} \langle u\rangle^0 + \langle u\rangle^1 = p \cdot q & (1) \\ z = (\langle u\rangle^0 + \langle u\rangle^1) \cdot (\langle x\rangle^0 + \langle x\rangle^1) & (2) \\ \text{sign of } x & (3) \\ \text{sign of } z/q & (4) \end{cases}$$

Obviously, from the former two equations, there are three unknown variables $(\langle u\rangle^1, \langle x\rangle^1, q)$. The true value of $\langle x\rangle^1$ can't be computed. With (3), we discuss in different situations.

If $x = 0$, since $z = x \cdot u = 0$. So $z/q = 0$.

If $x > 0$, since P_0 hold z, P_0 can deduce the sign of u based on (2). Similarly, P_0 can deduce the sign of q based on (1). Then P_0 can easily get the sign of z/q.

P_0 has already known about the sign of q by considering about the equation 1 and the sign of q.

If $x < 0$, the situations is the same as $x > 0$.

In conclusion, With (1)(2)(3), we can deduce sign of z/q.So (4) can be omitted. P_0 can't get information about x more than the sign of x based on (1)(2)(3).

Overall, the protocol leaks no information about x except the sign of x. As for the purpose of this protocol is to calculate the sign of x. We think that this protocol is secure.

3.3 Comparison between two secret variables

In Max Pooling Layer, the biggest element of near input elements is chosen to construct the feature map in next layer. This asks us to design a protocol to compare between two secret variables.

P_i has $\langle x \rangle^i$ and $\langle y \rangle^i$. They wonders x and y which one is bigger.
P_i computes $\Delta^i = \langle x \rangle^i - \langle y \rangle^i$.

4 Privacy preserved deep learning

In this section, we will introduce framework and concret implement of our system of privacy preserved deep learning.

4.1 Overview

Our system involves five characters: Hospital-$Hosp$, AI technology company-$Comp$, cloud sever_0 P_0, cloud sever_1-P_1 and a trusted setup-TTP. Tasks of each characters are as follow.

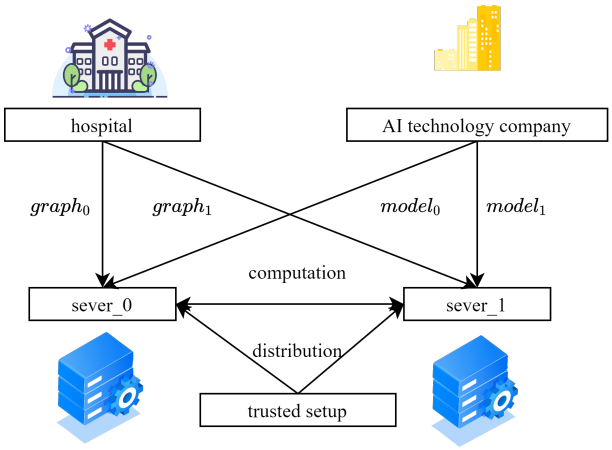

Figure 1: system framework

Hospital-$Hosp$: take CT images of the patient's lungs and distribute two pieces of shared images to two cloud severs.

AI technology company-$Comp$: prepare a trained convolutional neural network model in advance and distribute two pieces of convolution kernel to two cloud severs. upon receiving two shared results of Convolutional Layer and Pooling layer from two cloud severs, add the shared results to get the real result and conduct the rest part of convolutional neural network.

Cloud sever_0-P_0 and P_1: receive images and convolution kernel pieces from $Hosp$ and $Comp$. Compute results of Convlutional Layer, Pooling layer and ReLU function on the basis of **Protocol1-3**.

Trusted setup-TTP: distribute Beaver's multiplication triples and comparison triples. This task can be conducted at the very beginning, even long before the protocol running.

4.2 System Framework

Step1: Distribution:$\{\} \rightarrow \{P_i : (multiplication - triples, comparison - triples, \langle Imag\rangle^i, \langle Conv\rangle^i)\}$. TTP distributes enough Beaver's multiplication triples and comparison triples to P_0 and P_1. $Hosp$ distributes two pieces of shared part of CT images $\langle Imag\rangle^i$ to P_i. $Comp$ distributes two pieces of shared part of convolution kernel $\langle Conv\rangle^i$ to P^i.

Step2: Computation of Convolutional Layer:$\{P_i : (multiplication - triples, \langle Imag\rangle^i, \langle Conv_c\rangle^i)\} \rightarrow \{P_i : (\langle Feature_{conv}\rangle^i)\}$. Based on Protocol1 P_0 and P_1 compute Convolutional Layer using the shared data $\langle Imag\rangle$ and $\langle Conv\rangle$.

Step3: Computation of Pooling Layer. :$\{P_i : (comparison - triples, \langle Feature_{conv}\rangle^i)\} \rightarrow \{P_i : (\langle Feature_{pool}\rangle^i)\}$. Based on Protocol3 P_0 and P_1 compute max Pooling Layer using the shared data $\langle Imag\rangle$ and $\langle Conv\rangle$.

Step4: Computation of ReLU function: $\{P_i : (comparison - triples, \langle Feature_{pool}\rangle^i)\} \rightarrow \{P_i : (\langle Feature_{ReLU}\rangle^i)\}$. Based on Protocol2 P_0 and P_1 compute ReLU function using the shared data $\langle Imag\rangle$ and $\langle Conv\rangle$. Generally speaking, a CNN model usually has more than 1 Convolutional Layer. If there are more Convolutional Layer to be computed, move to **Step2:** Computation of Convolutional Layer with input:$P_i : (multiplication - triples, \langle Feature_{ReLU}\rangle^i, \langle Conv_c\rangle^i)$

Step54: Computation of rest part of CNN: $\{P_i : (\langle Feature_{ReLU}\rangle^i)\} \rightarrow \{Comp : (answer)\}$. P_i sends $(\langle Feature_{ReLU}\rangle^i)$ to $Comp$ secretly. $Comp$ continues with the remaining steps via the rest of model just like Fully-Connected Layer and Classification Layer.

4.3 Concrete implement

Step1: Distribution.TTP computes enough Beaver's multiplication triples $(a, b, c), s.t. c = a \cdot b,$. TTP distributes additive secret sharing of Beaver's multiplication triples to P_0 and P_1. Specifically, TTP sends $(\langle a\rangle^i, \langle b\rangle^i, \langle c\rangle^i)$ to P_i. Similarly, TTP prepares enough comparison triples $u, p, q, s.t. u = p \cdot q$. Then TTP sends $(\langle u\rangle^0, p)$ to P_0 and $(\langle u\rangle^1, q)$ to P_1.

Next we introduce the method of sharing matrix. Take 2×2 matrix A for example. One who shares the secret of A will randomly chooses 2×2 matrix R. Elements of $R \xleftarrow{\$} Z^*$. Then, set $\langle A\rangle^0 = R$ and $\langle A\rangle^1 = A - R$. The party sharing the secret sends $\langle A\rangle^i$ to P_0.

$$A = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} = \begin{bmatrix} R_{00} & R_{01} \\ R_{10} & R_{11} \end{bmatrix} + \begin{bmatrix} A_{00} - R_{00} & A_{01} - R_{01} \\ A_{10} - R_{10} & A_{11} - R_{11} \end{bmatrix} = \langle A\rangle^0 + \langle A\rangle^1$$

Following rules mentioned above, $Hosp$ distributes two pieces of shared part of CT images $\langle Imag\rangle^i$ to P_i. $Comp$ distributes two shared part of convolution kernel weight matrix $\langle Conv\rangle^i$ and bias b^i to P_i.

Step2: Computation of Convolutional Layer:$\{P_i : (\langle Imag\rangle^i, \langle Conv_c\rangle^i)\} \rightarrow \{P_i : (\langle Feature_{conv}\rangle^i)\}$. Based on Protocol1 P_0 and P_1 using the shared data $\langle Imag\rangle$ and $\langle Conv\rangle$.

Convolution is a frequently-used operation between an matrix (image or feature map) and a convolution kernel. In this paper, we take one two-dimensional convolution kernel for example. It can be easily generalized to dimensions more than 2 and more convolution kernel.

Let us denote $height \times width$ input image by A , $len \times len$weight matrix of convolution kernel by B and bias by b. $C = conv(A, B)$.

$$C_{k,j} = \sum_{w,l} A_{k+w,j+l} \cdot B_{w,l} + b$$

where $0 \leqslant k < height, 0 \leqslant j < width, 0 \leqslant w < len, 0 \leqslant l < len$.

Take notice of the formula, the right side of the multiplication part of equation consisted by multiplication and addition. Multiplication part can be computed via Protocol1 and the addition part can be operated by adding the secret variable locally.

We denote $mul_{k+w,j+l} = A_{k+w,j+l} \cdot B_{w,l}$.

$$C_{k,j} = \sum_{w,l} mul_{k+w,j+l} + b$$

To get $C_{k,j}$, P_0 and P_1 needs to get $mul^0_{k+w,j+l}+mul^1_{k+w,j+l} = (A^i_{k+w,j+l}+ A^{1-i}_{k+w,j+l}) \cdot (B^i_{w,l} + B^{1-i}_{w,l})$.

Firstly we focus on how to compute $mul_{k+w,j+l}$. When computing multiplication part, there are two situations.

Situation1: under some circumstance (specifically stated in Step4), P_i knows that $A^0_{k+w,j+l} = A^1_{k+w,j+l} = 0$. P_i set $mul^i_{k+w,j+l} = 0$ directly.

Situation2: under general circumstance, P_i has $A^0_{k+w,j+l}$ and $B^i_{w,l}$.

So P_0 and P_1 run **multiplication protocol** to get

$$mul^0_{k+w,j+l} + mul^1_{k+w,j+l} = (A^0_{k+w,j+l} + A^1_{k+w,j+l}) \cdot (B^0_{w,l} + B^1_{w,l})$$

Now, P_i holds $mul^i_{k+w,j+l}$.

After multiplication part, P_i needs to compute $C_{k,j}$ locally.

$$C^i_{k,j} = \sum_{w,l} mul^i_{k+w,j+l} + b^i$$

Step3: Computation of Pooling Layer: $\{P_i : (\langle Feature_{conv}\rangle^i)\} \rightarrow \{P_i : (\langle Feature_{pool}\rangle^i)\}$. Based on Protocol3 P_0 and P_1 compute max Pooling Layer using the shared data $\langle Imag\rangle$ and $\langle Conv\rangle$.

In the max Pooling Layer, P_0 and P_1 are asked to choose the biggest variable of four near elements to contrust the next feature map. P_0 and P_1 run Protocol3 to compare elements.

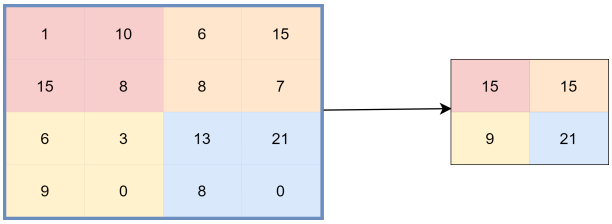

Figure 2: rules of Pooling Layer

By bubble sort, It is easy to see that it takes three times to compute the biggest variable.

Step4: Computation of ReLU function: $\{P_i : (\langle Feature_{pool}\rangle^i)\} \rightarrow \{P_i : (\langle Feature_{ReLU}\rangle^i)\}$. Based on Protocol2 P_0 and P_1 compute ReLU function using the shared data $\langle Imag\rangle$ and $\langle Conv\rangle$.

ReLU function is used in CNN to prevent overfitting. It will set variables which is smaller than 0 to 0 while remain variables bigger than 0.

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leqslant 0 \end{cases}$$

P_0 and P_1 wonder whether elments of feature map are bigger than 0. They run Protocol2 on input $P_0 : \langle Feature_{conv}\rangle^0, P_1 : \langle Feature_{conv}\rangle^1$

Generally speaking, a CNN model usually has more than 1 Convolutional Layer. If there are more Convolutional Layer to be computed, move to $Step2 : Computation of Convolutional Layer$ with input:$P_i : (multiplication-triples, \langle Feature_{ReLU}\rangle^i, \langle Conv_c\rangle^i)$

Step5: Computation of rest part of CNN: $\{P_i : (\langle Feature_{ReLU}\rangle^i)\} \rightarrow \{Comp : (answer)\}.P_i : (\langle Feature_{ReLU}\rangle^i)$. P_i sends $(\langle Feature_{ReLU}\rangle^i)$ to $Comp$ secretly. $Comp$ recovers $Feature_{ReLU}$ by $Feature_{ReLU} = \langle Feature_{ReLU}\rangle^0 + \langle Feature_{ReLU}\rangle^1$. $Comp$ continues with the the rest of model such as Fully-Connected Layer and Classification Layer.

5 Conclusion

Acknowledgments

These are acknowledgments.

References

[1] This is reference.

[2] This is reference.

6 Appendix

Algorithm 1: Protocol1:multiplication of two shared secret variables

Input: P_0: $\langle x\rangle^0$,$\langle y\rangle^0$
P_1: $\langle x\rangle^1$,$\langle y\rangle^1$
TTP: no
Output: P_0: $\langle z\rangle^0$
P_1: $\langle z\rangle^1$
s.t. $\langle z\rangle^0 + \langle z\rangle^1 = (\langle x\rangle^0 + \langle y\rangle^0) \cdot (\langle x\rangle^1 + \langle y\rangle^1)$

Algorithm 2: Protocol2: Comparison between a shared secret variable and zero

Input: P_0: $\langle x\rangle^0$
P_1: $\langle x\rangle^1$
TTP: no
Output:calculate the sign of x

TTP randomly chooses $\langle u\rangle^0 \xleftarrow{\$} Z^*, p \xleftarrow{\$} Z^*, q \xleftarrow{\$} Z^*$
TTP computes $\langle u\rangle^1 = p \cdot q - \langle u\rangle^0$
s.t. $\langle u\rangle^0 + \langle u\rangle^1 = p \cdot q$
$TTP \rightarrow P_0$: $(\langle u\rangle^0, p)$
$TTP \rightarrow P_1$: $(\langle u\rangle^1, q)$
P_0, P_1 run Protocol1 on input
$\{P_0 : (\langle u\rangle^0, \langle x\rangle^0), P_1 : (\langle u\rangle^1, \langle x\rangle^1), TTP()\}$, P_0 gets $\langle z\rangle^0$ and P_1 gets $\langle z\rangle^1$, s.t. $\langle z\rangle^0 + \langle z\rangle^1 = (\langle u\rangle^0 + \langle u\rangle^1) \cdot (\langle x\rangle^0 + \langle x\rangle^1)$
$P_0 \rightarrow P_1$: $\langle z\rangle^0$
P_1 computes $z = \langle z\rangle^0 + \langle z\rangle^1$
P_0 sets $s^0 = p$
P_1 sets $s^1 = z \cdot q$
So that $s^0 \cdot s^1$ P_i sets $sign^i$ = the sign of s^i
P_i shares $sign^i$ and computes Comparison result $res = sign^0 \cdot sign^1$
The sign of x equals to res

Algorithm 3: Protocol3: Comparison between two shared secret variables

Input: P_0: $\langle x\rangle^0$,$\langle y\rangle^0$
P_1: $\langle x\rangle^1$,$\langle y\rangle^1$
TTP: nothing
Output:figure out x and y which one is bigger.

P_0 computes $\Delta^0 = \langle x\rangle^0 - \langle y\rangle^0$
P_1 computes $\Delta^1 = \langle x\rangle^1 - \langle y\rangle^1$
P_0, P_1 run Protocol2 on input $\{P_0 : (\Delta^0), P_1 : (\Delta^1), TTP()\}$, P_0 and P_1 get information about whether $x \geq 0$.

Algorithm 4: Protocol4: Comparison between two secret variables

Input: P_0: x
P_1: y
TTP: nothing
Output: figure out x and y which one is bigger.

P_0 randomly chooses $\langle x\rangle^1 \xleftarrow{\$} Z^*$
P_0 sets $\langle x\rangle^0 = x - \langle x\rangle^1$
$P_0 \rightarrow P_1$:$\langle x\rangle^1$
P_1 randomly chooses $\langle y\rangle^0 \xleftarrow{\$} Z^*$
P_1 sets $\langle y\rangle^1 = y - \langle y\rangle^0$
$P_1 \rightarrow P_0$:$\langle y\rangle^0$
s.t.$x = \langle x\rangle^0 + \langle x\rangle^1, y = \langle y\rangle^0 + \langle y\rangle^1$
P_0, P_1, TTP run Protocol2 on input
$\{P0 : (\langle x\rangle^0, \langle y\rangle^0), P1 : (\langle x\rangle^1, \langle y\rangle^1), TTP()\}$