Interview for a PhD position

Li-Yue Sun
john.chiang.smith@gmail.com

March 5, 2023

- About me
- My 1st Paper
  - *quadratic gradient*
  - work to be done
- My 2nd Work
  - *volley revolver*
  - following work
- About this position

Name : **Li-Yue Sun**
Birthday : 19 November 1988,

- Java SSH frameworks (Structs, Spring and Hibernate); MVC pattern (Model-View-Controller)
- My first two research work is to be implemented on the Linux system by Eclipse using the programming language C++.
- I have obtained very decent maths scores on the Chinese Postgraduate Admission Test. (advanced mathematics, linear algebra, and probability and statistics)
- I quit my PhD in May 2021 after completing all the doctoral courses without publishing any papers even though having done enough research work.

Homomorphic Logistic Regression Training

Logistic Regression: a parametric modelthat has a fixed number of parameters

Parametric Model

- the advantage of often being faster to use
- the disadvantage of making stronger assumptions about the nature of the data distributions.

"We make some assumptions about the nature of the data distribution for a supervised problem. These assumptions are often embodied in the form of a parametric model, which is a statistical model with a fixed number of parameters." [1]

[1] Kevin P Murphy. *Machine learning: a probabilistic perspective*. Cambridge, MA: The MIT Press, 2012. URL: `https://books.google.co.jp/books?id=NZP6AQAAQBAJ`

Logistic Regression

Logistic Regression assumes that the data has some distribution that involves some model with parameters.

Logistic Regression Training: build the model and find its parameters

to determine the parameters of the assumed model
while fitting the training dataset, which is to maximize the loss function (maximum likelihood estimation)
something like finding $x_0, x_1, \cdots, x_d$ to maximize the function
$F = -(x_0 - 0)^2 - (x_1 - 1)^2 - \cdots - (x_d - d)^2$
model parameters: $x_0, x_1, \cdots, x_d$
training dataset: $0, 1, \cdots, d$
Logistic Regression Training: replace F with MLE
For convenience in the calculation: minimize the negative MLE

Logistic Regression

LR does not have a closed form of maximum problem.

First-Order: Gradient Descent Method

Momention: Momentum, Nesterov accelerated gradient (NAG)
Adagrad-like: Adagrad, Adadelta, RMSprop, Adam, AdaMax, Nadam

Second-Order: Newton's Method (Newton–Raphson method)

a root-finding algorithm that successively approximate to the roots (or zeroes) of a real-valued function

The log-likelihood function of LR has at most a unique global maximum where its gradient is zero.

Newton's Method [2]

pros

- quadratic convergence
- ……

cons

- Failure: Bad starting points, Derivative issues.
- **Frequent Invert operation:**
  the inverse of the Hessian matrix in every iteration

[2] https://en.wikipedia.org/wiki/Newton%27s_method

Newton's Method

pros

- quadratic convergence
- $\cdots\cdots$

cons

- Failure: Bad starting points, Derivative issues.
- **Frequent Invert operation**
  BFGS, $\cdots$, Fixed Hessian Method

Fixed Hessian Newton's Method

Dankmar Böhning and Bruce G Lindsay. "Monotonicity of quadratic-approximation algorithms". In: *Annals of the Institute of Statistical Mathematics* 40.4 (1988), pp. 641–663. DOI: https://doi.org/10.1007/BF00049423

- Basic Idea: to replace the varying Hessian with a fixed matrix
- the convergence of this method is guaranteed as long as the fixed Hessian substitute satisfies some conditions.

Fixed Hessian Newton's Method

pros: fast without compromising the efficiency

- a fixed matrix that only needs to be inverted once

cons: difficult to find such a matrix to

- meet the convergence condition
- fixed matrix (constant elements)
- good (lower) bound

They didn't give a systematic way to find or build such a fixed matrix. Personally, there are no such fixed matrices for most optimization problems.

Fixed Hessian Newton's Method

pros: fast without compromising the efficiency

- a fixed matrix that only needs to be inverted once

cons: difficult to find such a matrix to

- meet the convergence condition
- fixed matrix (constant elements)
- good (lower) bound

They did give a good lower bound $-\frac{1}{4}X^TX$ for binary LR and later a following work[3] gave a good bound matrix for multiclass LR, by analyzing the Hessian.

[3] Dankmar Böhning. "Multinomial logistic regression algorithm". In: *Annals of the institute of Statistical Mathematics* 44.1 (1992), pp. 197–200.

Simplified Fixed Hessian Method

Bonte and Vercauteren[4] simplify this bound $-\frac{1}{4}X^TX$ further replace the matrix $-\frac{1}{4}X^TX$ by a diagonal matrix B
The entries of the diagonal matrix are simply the sums of the rows of the matrix.

$$B = \begin{bmatrix} \sum_{i=0}^{d} \bar{h}_{0,i} & 0 & \ldots & 0 \\ 0 & \sum_{i=0}^{d} \bar{h}_{1,i} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sum_{i=0}^{d} \bar{h}_{d,i} \end{bmatrix},$$

where $\bar{h}_{d,i}$ is the element of $\bar{H} = -\frac{1}{4}X^TX$.
They proved that their SFH meets the convergence condition by Gerschgorin's circle theorem.

[4] Charlotte Bonte and Frederik Vercauteren. "Privacy-preserving logistic regression training". In: *BMC medical genomics* 11.4 (2018), p. 86. DOI: https://doi.org/10.1186/s12920-018-0398-y.

Simplified Fixed Hessian Method

This diagonal matrix B is in a very simple but diagonal matrix is something between the matrix and (column) vector. easy to invert

Privacy-Preserving Logistic Regression Training

Charlotte Bonte[1], Frederik Vercauteren[1]

imec-Cosic, Dept. Electrical Engineering, KU Leuven

Privacy-Preserving Logistic Regression Training with A Faster Gradient Variant

---

$$
\begin{aligned}
\boldsymbol{\beta}_{t+1} &= \boldsymbol{\beta}_t - B^{-1} \cdot \nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta}), \\
&= \boldsymbol{\beta}_t - \begin{bmatrix} b_{00} & 0 & \dots & 0 \\ 0 & b_{11} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & b_{dd} \end{bmatrix} \cdot \begin{bmatrix} \nabla_0 \\ \nabla_1 \\ \vdots \\ \nabla_d \end{bmatrix} = \boldsymbol{\beta}_t - \begin{bmatrix} b_{00} \cdot \nabla_0 \\ b_{11} \cdot \nabla_1 \\ \vdots \\ b_{dd} \cdot \nabla_d \end{bmatrix},
\end{aligned}
$$

where b_{ii} is the reciprocal of $\sum_{i=0}^{d} \bar{h}_{0i}$ and ∇_i is the element of $\nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta})$.

Consider a special situation: if $b_{00}, \dots, b_{dd}$ are all the same value $-\eta$ with $\eta > 0$, the iterative formula of the SFH method can be given as:

$$
\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - (-\eta) \cdot \begin{bmatrix} \nabla_0 \\ \nabla_1 \\ \vdots \\ \nabla_d \end{bmatrix} = \boldsymbol{\beta}_t + \eta \cdot \nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta}),
$$

which is the same as the formula of the naive gradient *ascent* method. Such coincident is just what

Simplified Fixed Hessian Method

pros

- all the advantages that the Fixed Hessian Newton's method has
- much simpler Hessian matrix without compromising performance

cons

- SFH cannot be applied to numerical optimization problems.
- SFH can still be singular. Fixed Hessian Method

give me a hint to find a systematic way to find the "*fixed*" Hessian for any optimization problems. (they don't notice it!)

Simplified Fixed Hessian Method

- a drawback of Newton's method is time-consuming to calculate the Hessian matrix and much more time-consuming to invert the Hessian matrix.
- a probable reason why the Fixed Hessian Method insisted to find a fixed Hessian substitute.
- Since it is easy to get the inverse of a diagonal matrix, we could abandon the idea of finding a fixed matrix and only focus on the two principles: meeting the convergence condition and a good bound diagonal matrix.

My 1st Work: quadratic gradient

$$B = \begin{bmatrix} \sum_{i=0}^{d} \bar{h}_{0,i} & 0 & \ldots & 0 \\ 0 & \sum_{i=0}^{d} \bar{h}_{1,i} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sum_{i=0}^{d} \bar{h}_{d,i} \end{bmatrix}$$

$$\tilde{B} = \begin{bmatrix} \epsilon + \sum_{i=0}^{d} |\bar{h}_{0i}| & 0 & \ldots & 0 \\ 0 & \epsilon + \sum_{i=0}^{d} |\bar{h}_{1i}| & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \epsilon + \sum_{i=0}^{d} |\bar{h}_{di}| \end{bmatrix}$$

where ϵ is a small positive constant to avoid division by zero

My 1st Work: quadratic gradient

The diagnoal matrix: easy to invert, mutiplied by a vector

$$H^{-1} \cdot g = \begin{bmatrix} \frac{1}{\epsilon + \sum_{i=0}^{d} |\bar{h}_{0,i}|} & 0 & \ldots & 0 \\ 0 & \frac{1}{\epsilon + \sum_{i=0}^{d} |\bar{h}_{1,i}|} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \frac{1}{\epsilon + \sum_{i=0}^{d} |\bar{h}_{d,i}|} \end{bmatrix} \cdot \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_d \end{bmatrix}$$

$$= \begin{bmatrix} \frac{g_0}{\epsilon + \sum_{i=0}^{d} |\bar{h}_{0,i}|} \\ \frac{g_1}{\epsilon + \sum_{i=0}^{d} |\bar{h}_{1,i}|} \\ \vdots \\ \frac{g_d}{\epsilon + \sum_{i=0}^{d} |\bar{h}_{d,i}|} \end{bmatrix} = G(\textit{quadratic gradient})$$

where g_i is the element of the gradient
$\bar{h}_{1,i}$ could be the elements of the Hessian itself

My 1st Work: quadratic gradient

$$NAG \begin{cases} V_{t+1} &= \boldsymbol{\beta}_t + \alpha_t \cdot g, \\ \boldsymbol{\beta}_{t+1} &= (1-\gamma_t) \cdot V_{t+1} + \gamma_t \cdot V_t, \end{cases}$$

$$Enhanced\ NAG \begin{cases} V_{t+1} &= \boldsymbol{\beta}_t + (1 + \alpha_t) \cdot G, \\ \boldsymbol{\beta}_{t+1} &= (1-\gamma_t) \cdot V_{t+1} + \gamma_t \cdot V_t, \end{cases}$$

$$Adagrad : \boldsymbol{\beta}_{[i]}^{(t+1)} = \boldsymbol{\beta}_{[i]}^{(t)} - \frac{\eta}{\epsilon + \sqrt{\sum_{k=1}^{t} g_{[i]}^{(t)} \cdot g_{[i]}^{(t)}}} \cdot g_{[i]}^{(t)},$$

$$Enhanced\ Adagrad : \boldsymbol{\beta}_{[i]}^{(t+1)} = \boldsymbol{\beta}_{[i]}^{(t)} - \frac{1 + \eta}{\epsilon + \sqrt{\sum_{k=1}^{t} G_{[i]}^{(t)} \cdot G_{[i]}^{(t)}}} \cdot G_{[i]}^{(t)}.$$

My 1st Work: quadratic gradient

- quadratic gradient can be applied to the two basic types of gradient descent methods: Momention and Adagrad-like.
  Momention: Momentum, Nesterov accelerated gradient (NAG)
  Adagrad-like: Adagrad, Adadelta, RMSprop
  Hybrid: Adam, AdaMax, Nadam
- Experiments show that the enhanced methods via quadratic gradient are probably super-quadratic.
- I implemented the enhanced NAG method for logistic regression training in the encrypted domain using C++ and HEAAN library.

My 1st Work: quadratic gradient

- combine the first-order (gradient descent/ascent) algorithms and second-order Newton's method; bridge the gap between ... and ...
- with the help of quadratic gradient, build super-quadratic methods [to be proved]
- supersede the line-search technique used in Newton's method, which uses only one scalar to accelerate $H^{-1} \cdot g$.

work to be done

- a direct following work to my first paper is to study the mini-batch version of the enhanced methods via quadratic gradient.
  Privacy-Preserving Logistic Regression Training on Large Encrypted Data

  Baseline work: Kyoohyung Han et al. "Logistic regression on homomorphic encrypted data at scale". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 9466–9471. DOI: https://doi.org/10.1609/aaai.v33i01.33019466

work to be done

- my third paper extends this work to multiclass LR.
- only need to add some description about the experiments and polish the paper.

Multinomial Logistic Regression Algorithms via Quadratic Gradient

John Chiang

john.chiang.smith@gmail.com

Abstract

Multinomial logistic regression, also known by other names such as multiclass logistic regression and softmax regression, is a fundamental classification method that generalizes binary logistic regression to multiclass problems. A recently work [6] proposed a faster gradient called `quadratic gradient` that can accelerate the binary logistic regression training, and presented an enhanced Nesterov's accelerated gradient (NAG) method for binary logistic regression. In this paper, we extend this work to multiclass logistic regression and propose an enhanced Adaptive Gradient Algorithm (Adagrad) that can accelerate the original Adagrad method. We test the enhanced NAG method and the enhanced Adagrad method on some multiclass-problem datasets. Experimental results show that both enhanced methods converge faster than their original ones respectively.

work to be done

- my fourth paper testified that quadratic gradient can indeed be used for general optimization questions.
- also include some other work and some doubts puzzling me. probably need to redesign the experiments.

[math.OC] 3 Sep 2022

Quadratic Gradient: Uniting Gradient Algorithm and Newton Method as One

John Chiang

john.chiang.smith@gmail.com

Abstract

It might be inadequate for the line search technique for Newton's method to use only one floating point number. A column vector of the same size as the gradient might be better than a mere float number to accelerate each of the gradient elements with different rates. Moreover, a square matrix of the same order as the Hessian matrix might be helpful to correct the Hessian matrix. Chiang [4] applied something between a column vector and a square matrix, namely a diagonal matrix, to accelerate the gradient and further proposed a faster gradient variant called quadratic gradient. In this paper, we present a new way to build a new version of the quadratic gradient. This new quadratic gradient doesn't satisfy the convergence conditions of the fixed Hessian Newton's method. However, experimental results show that it sometimes has a better performance than the original one in convergence rate. Also, Chiang [4] speculates that there might

following work

quadratic gradient: combining the first-order algorithms and the second-order method.

- quasi-Newton's method: a different zone involving convex analysis; extend quadratic gradient to the famous BFGS method
- theoretical prove the convergence of the enhanced methods
- theoretical analysis the complexity of the enhanced methods
  show if they are super-quadratic methods
  try to give a theoretical analysis
- it is difficult to prove it in mathematics for various enhanced methods.
- probably substituting and superseding the line-search method used in Newton's method.

Volley Revolver

Volley Revolver: A Novel Matrix-Encoding Method for Privacy-Preserving Neural Networks (Inference)

John Chiang *
john.chiang.smith@gmail.com

Abstract

In this work, we present a novel matrix-encoding method that is particularly convenient for neural networks to make predictions in a privacy-preserving manner using homomorphic encryption. Based on this encoding method, we implement a convolutional neural network for handwritten image classification over encryption. For two matrices A and B to perform homomorphic multiplication, the main idea behind it, in a simple version, is to encrypt matrix A and the transpose of matrix B into two ciphertexts respectively. With additional operations, the homomorphic matrix multiplication can be calculated over encrypted matrices efficiently. For the convolution operation, we in advance span each convolution kernel to a matrix space of the same size as the input image so as to generate several ciphertexts, each of which is later used together with the ciphertext encrypting input images for calculating some of the final convolution results. We accumulate all these intermediate results and thus complete the convolution operation.

In a public cloud with 40 vCPUs, our convolutional neural network implementation on the MNIST testing dataset takes ~ 287 seconds to compute ten likelihoods of 32 encrypted images of size 28×28 simultaneously. The data owner only needs to upload one ciphertext (~ 19.8 MB) encrypting these 32 images to the public cloud.

Homomorphic Encryption

- a type of encryption: message(plaintext) integer or a vector of integer; ciphertext
- can be used to compute operations on encrypted data without decryption
- Important Progress: Bootstrapping[5]; Rescale Operation[6]

[5] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 169–178. DOI: `https://doi.org/10.1145/1536414.1536440`.

[6] Jung Hee Cheon et al. "Homomorphic encryption for arithmetic of approximate numbers". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2017, pp. 409–437. DOI: `https://doi.org/10.1007/978-3-319-70694-8_15`.

Homomorphic Encryption

When applying Homomorphic Encryption to Machine Learning applications
Single Instruction Multiple Data (aka SIMD) manner by Chinese Remainder Theorem (CRT)
message is a vector of some intergers; ciphertext is Rings

$$
\begin{array}{c}
message : \begin{bmatrix} n_0 & n_1 & n_2 & n_3 & n_4 & n_5 & n_6 & n_7 & n_8 & n_9 \end{bmatrix} \\
\Big\downarrow \text{Encrypt} \\
Enc \begin{bmatrix} n_0 & n_1 & n_2 & n_3 & n_4 & n_5 & n_6 & n_7 & n_8 & n_9 \end{bmatrix}
\end{array}
$$

$$
\begin{array}{c}
Enc \begin{bmatrix} n_0 & n_1 & n_2 & n_3 & n_4 & n_5 & n_6 & n_7 & n_8 & n_9 \end{bmatrix} \\
\Big\downarrow \text{Rotate(1)} \\
Enc \begin{bmatrix} n_1 & n_2 & n_3 & n_4 & n_5 & n_6 & n_7 & n_8 & n_9 & n_0 \end{bmatrix}
\end{array}
$$

Homomorphic Encryption

Single Instruction Multiple Data (aka SIMD) manner

$$Enc\left[\begin{array}{cccccc} m_0 & m_1 & m_2 & m_3 & m_4 & m_5 \end{array}\right]$$

$$\bigoplus$$

$$Enc\left[\begin{array}{cccccc} n_0 & n_1 & n_2 & n_3 & n_4 & n_5 \end{array}\right]$$

$$||$$

$$Enc\left[\begin{array}{cccccc} m_0+n_0 & m_1+n_1 & m_2+n_2 & m_3+n_3 & m_4+n_4 & m_5+n_5 \end{array}\right]$$

$$Enc\left[\begin{array}{cccccc} m_0 & m_1 & m_2 & m_3 & m_4 & m_5 \end{array}\right]$$

$$\bigotimes$$

$$Enc\left[\begin{array}{cccccc} n_0 & n_1 & n_2 & n_3 & n_4 & n_5 \end{array}\right]$$

$$||$$

$$Enc\left[\begin{array}{cccccc} m_0\cdot n_0 & m_1\cdot n_1 & m_2\cdot n_2 & m_3\cdot n_3 & m_4\cdot n_4 & m_5\cdot n_5 \end{array}\right]$$

Database Encoding Method

Single Instruction Multiple Data (aka SIMD) manner

Training Dataset: Matrix Z			
$z_{[1][0]}$	$z_{[1][1]}$	$\cdots$	$z_{[1][d]}$
$z_{[2][0]}$	$z_{[2][1]}$	$\cdots$	$z_{[2][d]}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$z_{[n][0]}$	$z_{[n][1]}$	$\cdots$	$z_{[n][d]}$

$$\downarrow \text{Encoding}$$

$$\begin{bmatrix} z_{[1][0]} & \cdots & z_{[1][d]} & z_{[2][0]} & \cdots & z_{[2][d]} & \cdots & z_{[n][0]} & \cdots & z_{[n][d]} \end{bmatrix}$$

$$\downarrow \text{Encrypt}$$

$$Enc\begin{bmatrix} z_{[1][0]} & \cdots & z_{[1][d]} & z_{[2][0]} & \cdots & z_{[2][d]} & \cdots & z_{[n][0]} & \cdots & z_{[n][d]} \end{bmatrix}$$

Database Encoding Method

$$\Big\downarrow \text{Encrypt}$$

$$Enc \begin{bmatrix} z_{[1][0]} & \cdots & z_{[1][d]} & z_{[2][0]} & \cdots & z_{[2][d]} & \cdots & z_{[n][0]} & \cdots & z_{[n][d]} \end{bmatrix}$$

$$\Big\downarrow$$

$$Enc \begin{bmatrix} z_{[1][0]} & z_{[1][1]} & \cdots & z_{[1][d]} \\ z_{[2][0]} & z_{[2][1]} & \cdots & z_{[2][d]} \\ \vdots & \vdots & \ddots & \vdots \\ z_{[n][0]} & z_{[n][1]} & \cdots & z_{[n][d]} \end{bmatrix}$$

Andrey Kim et al. "Logistic regression model training based on the approximate homomorphic encryption". In: *BMC medical genomics* 11.4 (2018), p. 83. DOI: https://doi.org/10.1186/s12920-018-0401-7s

Database Encoding Method

Single Instruction Multiple Data (aka SIMD) manner

$$\texttt{SumRowVec}(Z) = Enc \begin{bmatrix} \sum_{i=1}^{n} z_{[i][1]} & \sum_{i=1}^{n} z_{[i][2]} & \cdots & \sum_{i=1}^{n} z_{[i][f]} \\ \sum_{i=1}^{n} z_{[i][1]} & \sum_{i=1}^{n} z_{[i][2]} & \cdots & \sum_{i=1}^{n} z_{[i][f]} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} z_{[i][1]} & \sum_{i=1}^{n} z_{[i][2]} & \cdots & \sum_{i=1}^{n} z_{[i][f]} \end{bmatrix},$$

$$\texttt{SumColVec}(Z) = Enc \begin{bmatrix} \sum_{j=1}^{f} z_{[1][j]} & \sum_{j=1}^{f} z_{[1][j]} & \cdots & \sum_{j=1}^{f} z_{[1][j]} \\ \sum_{j=1}^{f} z_{[2][j]} & \sum_{j=1}^{f} z_{[2][j]} & \cdots & \sum_{j=1}^{f} z_{[2][j]} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j=1}^{f} z_{[n][j]} & \sum_{j=1}^{f} z_{[n][j]} & \cdots & \sum_{j=1}^{f} z_{[n][j]} \end{bmatrix}.$$

Database Encoding Method

Single Instruction Multiple Data (aka SIMD) manner
I developed my own procedure to facilitate convolution operation in CNN as shown in the following toy example:

$$Z = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \xmapsto{\texttt{SumForConv}(\cdot,2,2)}$$

$$\begin{bmatrix} a+b+e+f & 0 & c+d+g+h & 0 \\ 0 & 0 & 0 & 0 \\ i+j+m+n & 0 & k+l+o+p & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

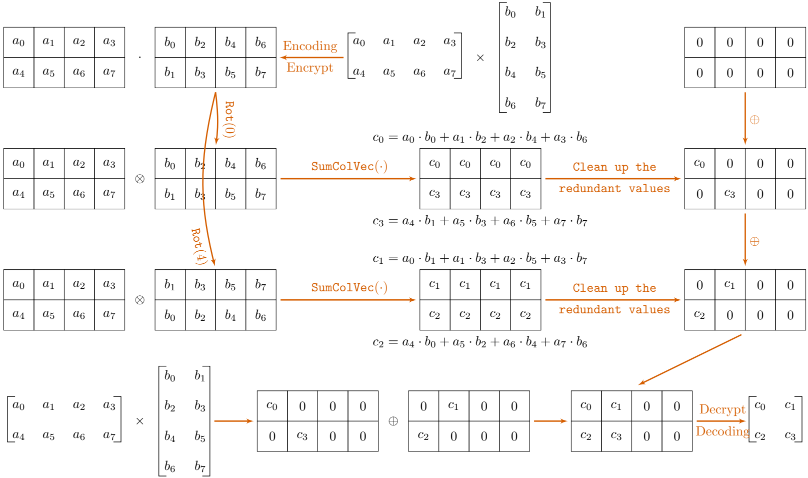
Database Encoding Method

$$
Enc\begin{bmatrix}
I^{(1)}_{[1][1]} & I^{(1)}_{[1][2]} & \cdots & I^{(1)}_{[h][w]} \\
I^{(2)}_{[1][1]} & I^{(2)}_{[1][2]} & \cdots & I^{(2)}_{[h][w]} \\
\vdots & \vdots & \ddots & \vdots \\
I^{(m)}_{[1][1]} & I^{(m)}_{[1][2]} & \cdots & I^{(m)}_{[h][w]}
\end{bmatrix}
\longrightarrow
\begin{bmatrix}
\mathtt{vEnc}\begin{bmatrix} I^{(1)}_{[1][1]} & I^{(1)}_{[1][2]} & \cdots & I^{(1)}_{[h][w]} \end{bmatrix} \\
\mathtt{vEnc}\begin{bmatrix} I^{(2)}_{[1][1]} & I^{(2)}_{[1][2]} & \cdots & I^{(2)}_{[h][w]} \end{bmatrix} \\
\vdots \\
\mathtt{vEnc}\begin{bmatrix} I^{(m)}_{[1][1]} & I^{(m)}_{[1][2]} & \cdots & I^{(m)}_{[h][w]} \end{bmatrix}
\end{bmatrix},
$$

or

$$
Enc\begin{bmatrix}
I^{(1)}_{[1][1]} & I^{(1)}_{[1][2]} & \cdots & I^{(1)}_{[h][w]} \\
I^{(2)}_{[1][1]} & I^{(2)}_{[1][2]} & \cdots & I^{(2)}_{[h][w]} \\
\vdots & \vdots & \ddots & \vdots \\
I^{(m)}_{[1][1]} & I^{(m)}_{[1][2]} & \cdots & I^{(m)}_{[h][w]}
\end{bmatrix}
\longrightarrow Enc
\begin{bmatrix}
\mathtt{vEnc}\begin{bmatrix} I^{(1)}_{[1][1]} & \cdots & I^{(1)}_{[1][w]} \\ \vdots & \ddots & \vdots \\ I^{(1)}_{[h][1]} & \cdots & I^{(1)}_{[h][w]} \end{bmatrix} \\
\vdots \\
\mathtt{vEnc}\begin{bmatrix} I^{(m)}_{[1][1]} & \cdots & I^{(m)}_{[1][w]} \\ \vdots & \ddots & \vdots \\ I^{(m)} & & I^{(m)} \end{bmatrix}
\end{bmatrix}.
$$

Homomorphic matrix multiplication



homomorphic neural networks training

Privacy-preserving CNN Inference

- I adopt the highly customizable library Keras with Tensorflow to define our own model layers such as the activation layer to enact the polynomial activation function.
- I use C++ to implement the homomorphic CNN inference. The HE programming work is challenging.

following work

- add some comparison with the baseline work
- a following work to my second paper is to build deep CNN inference based on Volley Revolver.

following work

- Encrypt the transpose of the second matrix for two matrices to perform multiplication
- There is some symmetry between the first matrix and the transpose of the second matrix
- My encoding method can be used to implement homomorphic neural networks training. (*Important Work*)

For a PhD Position

- happened to know that I could find a PhD position without an IELTS score
- I wish to restart my research as soon as possible so that I don't need to waste another year.
- My already done work can help to initiate, attain and coordinate research project ideas.

Thank You!