Privacy Preserving Training and Evaluation with Homomorphic Encryption: A Case Study on Machine Learning Models

Jiazhi Chen, Yunxiao Song

December 2021

Abstract

This project thoroughly studied homomorphic encryption and attempted to apply it in training machine learning models. We trained some models on plain data and evaluated them on encrypted data using encrypted parameters. We eventually created a logistic regression model that trains on encrypted data and maintains high accuracy.

1 Introduction

Today, machine learning algorithms are widely applied in the real world, which could cause the leakage of sensitive data. Therefore, the concern of data privacy is raised. In this paper, we applied CKKs vectors onto a pre-trained model, enabling them to calculate over the client's encrypted data. In other words, our method provides machine learning computations over the client's encrypted data without actually letting the service provider know it. We implemented and tested some of the most common machine learning algorithms, including Logistic Regression(LR), Fully Connected Neural Network(FCNN), and Convolutional Neural Network(CNN). Furthermore, we explored the outsourced model training of encrypted models on encrypted data by implementing a logistic regression. The model itself trains on encrypted data, providing even stronger securities.

2 Related Works

2.1 Homomorphic Encryption

Privacy-Preserving machine learning allows processing private information like hospital data without disclosing it to persons engaged in processing [12][15], while homomorphic encryption exactly defines a way to delegate the processing of the data, without giving away access to it[11]. Therefore, homomorphic encryption is a perfect fit for cloud computation and "Machine Learning as a Service" (MLaaS)[10]. Based on related work, training machine learning models on encrypted data is evaluated to be extremely slow and almost has no real-world application.

2.2 TenSEAL

Throughout the project, we utilized the package TenSEAL[1] to carry out the homomorphic encryption operations. TenSEAL is a library for doing homomorphic encryption operations on tensors, built on top of Microsoft SEAL[13]. It provides ease of use through a Python API while preserving efficiency by implementing most of its operations using C++. TenSEAL carries out encryption/decryption of vectors of real numbers using the CKKS scheme[3]. The CKKS scheme allows additions and multiplications on encrypted real or complex numbers but yields only approximate results. For integer values, TenSEAL also provides the BFV scheme[6] that yields the exact values.

2.3 Machine Learning Baselines

There are several machine learning models that are possible to apply homomorphic encryption on it. The Logistic regression [14] measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative distribution of logistic distribution. The sigmoid function in logistic regression cannot be directly incorperated into the homomorphic encryption because of the expotentional operater. However previous research on training logsitic regression over encrypted data mentions the polynomial approximation to sigmoid function[2].

Fully Connected Neural Networks[7] is the most essential and almost the first mentioned neural networks in deep learning and it can be used for classification tasks. We incorperated it into our research while comparing it with the Convolutional Neural Network[8], which is also a popular topic in deep learning with homomorphic encryption.

3 Methodology

3.1 Data Evaluation on Encrypted Model

Homomorphic encryption demonstrate ways to do operations on encrypted data without letting the operator known specific information. In the Context of machine learning, Linear unit is the essential part of the logistic regression, convolutional neural network and fully connected neural network, and it outputs the result as the matrix multiplication between the input and the weight and plus the bias. Therefore, the simplicity and repetitive nature of linear unit enables such structure to work with homomorphic encrypted calculation by encrypting machine learning weights, bias and directly apply them on to the encrypted data. The service provider thus calculates the encrypted results and return them to users with full privacy. Three types of machine learning models are compared within this section. We implemented a logistic regression, a simplified Convolutional Neural Network Model, and a fully connected neural network model within pytorch to compare the performance between encrypted model and non-encrypted models.

3.1.1 Experiments Setting

The target of the experiment is to learn the efficiency and the effectiveness of our models. All experiments are run within the same environment settings. Different data sets are used for comparing different baselines which is suitable for different tasks. We use the Pima India dataset for comparing the speed and accuracy of the logistic regression baseline and encrypted logistic regression. Since the pima-india-diabetes dataset has a single column “Outcome” with values of 0 or 1, we chose to train a logistic regression model on it. For comparing performance over the neural networks, the MNIST [5] image data set is used for comparing the performance. The MNIST is an image data set that includes images of 10 kinds of hand-written digits, which essentially forms a multi-class classification task. Since the MNIST data set is always used as a performance comparison set in deep learning, evaluations and encrypted evaluations on image classification deep learning models are run on this dataset. In our experiment, we will include the data of computational speed and accuracy of CNNs, FCNNs, EncCNN, EncFCNNS. For the neural networks, the time overhead between the propagation of each layers and the total time overhead for each prediction will be measured within the same environment. Furthermore, the accuracy of the encrypted prediction and non-encrypted prediction will also be compared to check the effectiveness of such scheme.

3.1.2 Baselines Introduction

We implemented three machine learning models as baselines for the model encryption, encrypted data evaluation and performance comparison. To represent the single unit of our model, let $x =$

$(x_1, x_2, ..., x_n)$ be our input data with dimension n, the weights are represented as $w = (w_1, w_2, ..., w_n)$, and the bias are represented as a scalar b. The linear regression is represented as $f(x) = xw + b$. Meanwhile, the sigmoid function can be represented as $Sigmoid(x) = \frac{1}{1-e^{-x}}$. The logistic regression uses sigmoid function to distribute the outputs with a probability between $[0, 1]$, therefore the logistic regression is basically $f(x) = Sigmoid(wx + b)$ which gives an value within $[0, 1]$.

For the fully connected neural networks, we combined listed numbers of units that are fully connected to the input as a layer, and we can further implement hidden layers that are connected to it. for each unit of the layer, it truly is the single linear unit that mentioned in logistic regression. It is more complicated to implement the Convolutional Neural Networks. Besides the linear layers in the model, the core part of the CNN is the convolutional layer which enables the network to extract spatial information from the image and do the recognition. The convolution layer works by moving and multiplying a kernel filter that extracts the image information. As for the activation function, a squared activation is used in order to parallel with the EncCNN.

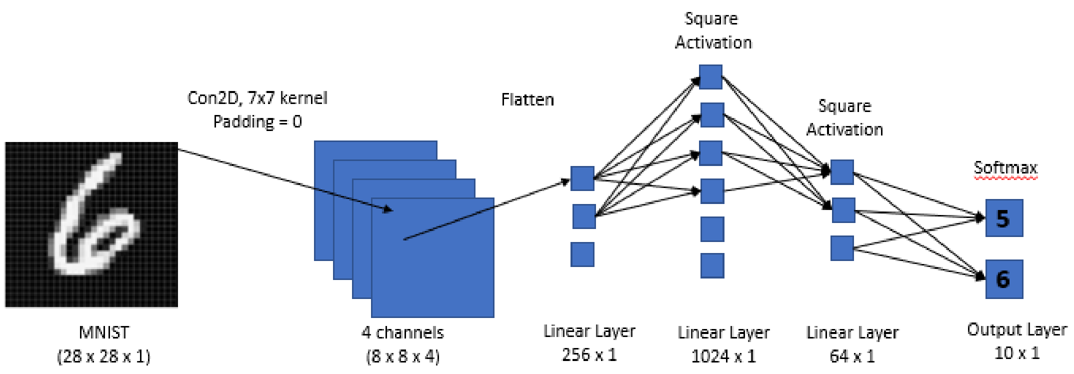

Figure 1: Our Baseline CNN model.

3.1.3 Model Encryption and Evaluation on Encrypted Data

For a encrypted evaluation on encrypted model, we started from encrypt single linear unit which is simple to explain. Given that $f(x) = Sigmoid(wx + b)$ with input size n, the encrypted weights are represented as

$$Enc(w) = (Enc(w_1), Enc(w_2), ..., Enc(w_n))$$

the encrypted bias is represented as $Enc(Bias) = Enc(b)$, and the encrypted input is represented as $Enc(x) = (Enc(x_1), Enc(x_2), ..., Enc(x_n))$ Therefore the encrypted linear model can be represented as $Enc_f(Enc(x)) = Enc(w)Enc(x) + Enc(b)$. For the logistic regression, we enabled the calculation of encrypting the weight and the bias of the single unit carry out encrypted evaluation on encrypted model.

For the evaluation of encrypted data, the parameters of FCNN models are adjusted to be able to work with CKKS vectors, and it is named as EncFCNN model. The prediction overhead was tested by directly applying the real trained parameters on homomorphic encrypted data. The structure of EncFCNN is shown in the figure.

For evaluating encrypted data on EncCNN models, the major obstacles is that the CKKS vector's multiplication can only be carried out within certain times, and there's an bounded degree that the vector won't exceed. Therefore, we managed to avoid the implementation of the traditional kernel filter

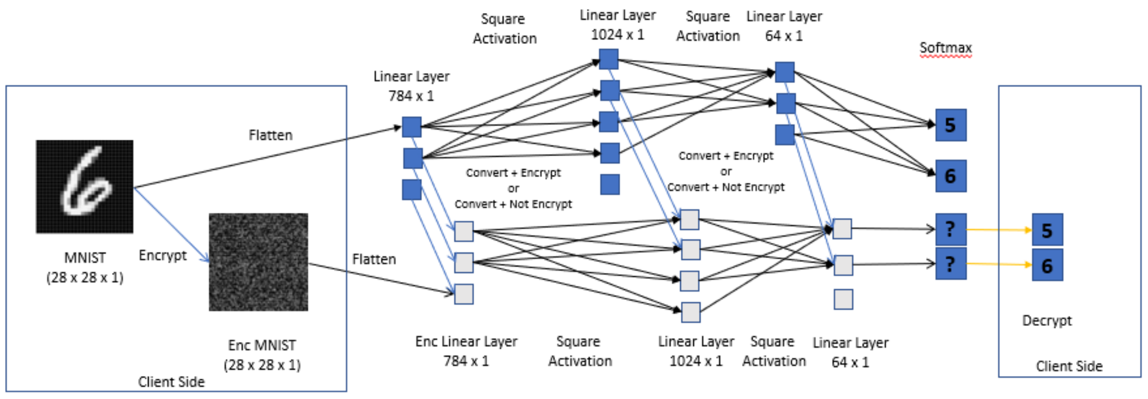

Figure 2: Our Baseline Enc FCNN model and THE Encryption Work Flow.

operation, which requires the CKKS Vector to exceed its maximum degree. In our model, the matrix multiplication insights from TenSEAL is adopted to enable the convolutional layer, where they provided a column convolution that is compatible with CKKS vectors and avoids exceeding degree bound, which is used for our encrypted convolutional neural networks. The overall structure of encrypted evaluation is identical to EncFCNN models except the convolutional layer.

3.2 Training Models on Encrypted Data

3.2.1 Data Encryption

In the process of homomorphically training a logistic regression model, a ciphertext will need a multiplicative depth of 6. We will use 8192 polynomial modulus degree.

3.2.2 Training

In this section, we implemented a logistic regression model on plain data, and a logistic regression model on encrypted data. To evaluate the encrypted model's performance, we made the two's implementations as close as possible, though some modifications must be made on the encrypted version. We split the dataset into a training set and a testing set with a ratio of 7:3. The logistic regression training carries out as follows: we use stochastic gradient descent to optimize our model parameters, and we use binary cross entropy loss to measure the loss. More specifically, for each epoch, the program first calculates the loss of the current model parameters on the training dataset and then updates the parameters. For the plain data training, we have the built-in optimizer and loss criterion of pytorch. For the encrypted version, we cannot use the built-in optimizer and loss criterion, so we explicitly updated the parameters and we would talk about it in the following subsection "parameter update".

3.2.3 Sigmoid Approximation

We cannot compute sigmoid on encrypted data. Instead, we could use a low degree polynomial to approximate the sigmoid function. We used this approximation of the sigmoid function in the range [-5, 5] [2].

$$\text{Sigmoid}(x) = 0.5 + 0.197x - 0.0004x^3$$

3.2.4 Parameters Update

For this homomorphic training, we use the same loss function as for the plaintext training, which is the binary cross entropy loss function, but with regularization. For the encrypted version, the way of computing the binary cross entropy error follows:

$$Loss(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(\hat{y}^{(i)}) + (1-y^{(i)})log(1-\hat{y}^{(i)})] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

It turns out that we do not need to explicitly compute the error out. Instead, we can derive the gradients from the formula.

$$\theta_j = \theta_j - [\frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})x^{(i)} + 0.05\theta_j]$$

For each datapoint, we accumulate its effect on the gradients.

4 Empirical Results

4.1 Homomorphic Evaluation - Logistic Regression

We want to make sure when evaluating encrypted data using encrypted parameters, the program maintains the same level accuracy. After training 5 epochs using stochastic gradient descent optimization and binary cross entropy loss, the model achieves an accuracy of 0.68125 on the testing set. When evaluating on encrypted data using encrypted parameters, the model achieves an accuracy of 0.675 on the same data. Evaluating on the encrypted test set doesn't affect the accuracy that much.

4.2 Homomorphic Evalulation - Fully Connected Neural Network

The evaluation is carried out using the MNIST data set, while the baseline gives a pretty high accuracy with a high speed. We tested the baseline with 10,000 MNIST digit images, and the model gives 96.15% accuracy on the testing set, while the computation overhead is small. The Encrypted Fully Connected Neural Network has a huge evaluation overhead, which is approximately 102.8s for a single MNIST image in preset environment. The majority evaluation overhead is on propagating the first 2 layers which are relatively large. Due to the huge overhead, we limited the test set to 50 images, the accuracy is good on EncFCNN even with the noises standing on our side.

Model	Eval Overhead	Accuracy	Test Size
FCNN	23.37 ms	96.15%	10000
CNN	62.09 ms	98.28%	10000
EncFCNN	102.8 s	98%	50
EncCNN	10.74 s	100%	100

Table 1: Performance Comparison between Homomorphic Evaluated Deep Learning Models on MNIST data set. The Evalulation overheads represents the times it takes to homomorphically evaluate a single picture, while the testing size is varied because the overhead is huge on homomorphic evaluations. The polynomial modulus degree is 8192 and the result comes from a Core-i7 8550U CPU@1.80GHZ

layer	(Input, Output)	Eval Overhead
FC1	(784,1024)	54.21 s
FC2	(1024,64)	42.45 s
FC3	(64,10)	0.9442 s

Table 2: Performance Comparison between neural layers within the EncFCNN. The Evalulation overheads represents the average time of first 5 image data to propagate the single fully connected linear layer. The polynomial modulus degree is 8192 and the result comes from a Core-i7 8550U CPU@1.80GHZ

4.3 Homomorphic Evaluation - Convolutional Neural Network

The evaluation still uses the MNIST data set, while the baseline gives a pretty high accuracy with a high speed. We tested the baseline with 10,000 MNIST digit images, and the model gives 98.28% accuracy on the testing set, while the computation overhead is small. The EncCNN has a moderate evaluation overhead, which is approximately 10.74s for a single MNIST image in preset environment. The EncCNN gives an accuracy of 100% within the first 100 MNIST images in test set, which proves the validity of the model on homomorphic evaluation.

layer	(Input, Output)	Eval Overhead
Conv2D	(1,4)	1.099 s
FC1	(256,64)	7.433 s
FC2	(64, 10)	0.6196 s

Table 3: Performance Comparison between neural layers within the EncCNN. The Evaluation overheads represents the average time of first 5 image data to propagate the mentioned neural layers. The polynomial modulus degree is 8192 and the result comes from a Core-i7 8550U CPU@1.80GHZ

4.4 Training Logistic Regression on Encrypted Data

For training a logistic regression model on encrypted data, we not only want to test the model's accuracy, but would also like to measure the program's running time, because we expect the homomorphic training process to consume more time than the plain data version. On average, encrypting the training set takes 8-10 seconds (encryption of 768 rows * 0.7 * 9 columns real numbers). 5 epochs of training takes approximately 200 seconds on average. The model, when evaluated on the test set, yields an accuracy of 0.7375, which is even better than the one trained on plain data.

5 Security Analysis

TenSEAL uses the CKKS scheme to encrypt and decrypt vectors of numbers. In the CKKS scheme, when encrypting a message, a small error will be inserted into the message to guarantee the security of hardness of the ring-LWE (RLWE) problem which is assumed to be hard. The CKKS scheme is believed to satisfy the IND-CPA security based on the hardness assumption of ring-LWE [4].

Baiyu Li and Daniele Micciancio proposed that the IND-CPA security achieved by CKKS might not strong in enough in practice and presented passive attacks to the scheme, primarily by inferring the error used in encryption from the decrypted results [9]. They were able to recover the key with high probability and modest run time. SEAL has not made a modification for this attach but it noted the users that the decryption results should be treated as private information only available to the secret owner, therefore preserving some securities. Other HE libraries based on CKKS presented mitigation strategies against this attack, and shared the same idea to attach a proper error at the end of the decryption process to avoid the linearity on the secret key [3].

Based on the security of CKKS Vector and Microsoft SEAL, The data owner keeps the secret key and encrypts the data with it. The homomorpic evaluation is purely based on encrypted data and no detail is revealed to the model owner. Therefore, our scheme guarantees the same level of encryption security as CKKs Vectors and SEAL.

6 Conclusions

In this paper, we applied CKKs vectors onto pretrained model, enabling them to calculate on client's encrypted data. We enables EncCNN and EncFCNN to homomorphically evaluate data from the data owner, while providing data owner with the same level of accuracy from the baselines. Although the evaluation overhead is significantly high, the time is still manageable in real life scenarios. Furthermore, we explored the outsourced model training of encrypted model on encrypted data by implementing a logistic regression. The model itself trains on encrypted data, providing even stronger securities.

References

[1] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption. *CoRR*, abs/2104.03152, 2021.

[2] Hao Chen, Ran Gilad-Bachrach, Kyoohyung Han, Zhicong Huang, Amir Jalali, Kim Laine, and Kristin Lauter. Logistic regression over encrypted data from fully homomorphic encryption. *BMC medical genomics*, 11(4):3–12, 2018.

[3] Jung Hee Cheon, Seungwan Hong, and Duhyeong Kim. Remark on the security of ckks scheme in practice. *IACR Cryptol. ePrint Arch.*, 2020:1581, 2020.

[4] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[5] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[6] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An improved rns variant of the bfv homomorphic encryption scheme. In *Cryptographers' Track at the RSA Conference*, pages 83–105. Springer, 2019.

[7] K-Y Hsu, H-Y Li, and Demetri Psaltis. Holographic implementation of a fully connected neural network. *Proceedings of the IEEE*, 78(10):1637–1645, 1990.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[9] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. Cryptology ePrint Archive, Report 2020/1533, 2020. https://ia.cr/2020/1533.

[10] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 896–902. IEEE, 2015.

[11] Ronald L. Rivest and Michael L. Dertouzos. On data banks and privacy homomorphisms. 1978.

[12] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning, 2018.

[13] Microsoft SEAL (release 3.2). https://github.com/Microsoft/SEAL, February 2019. Microsoft Research, Redmond, WA.

[14] Raymond E Wright. Logistic regression. 1995.

[15] Sergey Zapechnikov. Privacy-preserving machine learning as a tool for secure personalized information services. *Procedia Computer Science*, 169:393–399, 2020. Postproceedings of the 10th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2019 (Tenth Annual Meeting of the BICA Society), held August 15-19, 2019 in Seattle, Washington, USA.