Poster: Privacy-preserving Neural Network with Functional Encryption

Prajwal Panzade
Department of Computer Science
Georgia State University
Atlanta, USA
ppanzade1@student.gsu.edu

Daniel Takabi
Department of Computer Science
Georgia State University
Atlanta, USA
takabi@gsu.edu

***Abstract*—The increasing need to train machine learning models on massive datasets has led to privacy and security problems. The researchers deal with such privacy problems using various privacy-preserving machine learning (PPML) techniques such as differential privacy, secure multi-party computation (SMC), fully homomorphic encryption (FHE), and recently functional encryption (FE). However, very little focus is given to FE-based PPML. This work presents a complete PPML pipeline that successfully performs training and inference on the encrypted data using functional encryption. We use the function-hiding inner product encryption (FHIPE) and inner-product function encryption (IPFE) for designing the secure activation function. Our results show successful training on the MNIST dataset with an overall accuracy of 93% and increased speedup compared to previous work.**

***Index Terms*—Functional Encryption, Function Hiding Inner-product encryption, Privacy-preserving Machine Learning, Secure Activation**

I. Introduction

The advancement of machine learning in computer vision, natural language processing, and speech processing has resulted in an abundance of remarkable applications that have become an integral part of people's lives. Today, many real-world machine learning applications rely on a cloud computing environment, following the concept of machine learning as a service (MLaaS). Cloud-based data and machine learning services are being adopted by many highly regulated firms and organizations, including banks, governments, insurance companies, and healthcare. As a result of this advancement, there is an increased requirement for secure and private computing solutions that protect data and model privacy in machine learning applications. In light of this, researchers are paying close attention to privacy-preserving machine learning (PPML). The PPML aims to address data and model privacy issues in the machine learning training to deployment stages.

In recent years, Secure Multiparty Computation and Fully homomorphic encryption have been prevalent approaches to PPML. Similar to homomorphic encryption, functional encryption has been emerging day by day. Most of the works based on Fully homomorphic encryption and functional encryption support inference on encrypted data. However, very little attention is given to the training on encrypted data [1]–[3]. Xu et al. in [4] proposed a method to train a deep neural network using functional encryption with 95.49% of accuracy but the training time required by their model is 57 hours. In this work, we focus on reducing the training time required by the functional encryption-based neural network.

II. Functional Encryption

Functional encryption is a generalization of public-key encryption that allows a key holder to compute a particular function of encrypted data using constrained secret keys [5]. Here, this function is called functionality. e.g., an FE scheme may be particularly designed to compute inner products; in this case, the functionality becomes an inner product. In the FE scheme, a key management authority with a master secret key generates a secret key sk_{fe}; a decryptor can use that to compute a function on an encrypted message X. An FE scheme consists of four algorithms Setup, Encrypt, KeyDerivation, and Decrypt. Setup generates the initial public and private keys. Encrypt is used for encrypting the message. KeyDerivation provides the key based on functionality, and finally, decrypt is used for decryption. Our work uses two different functional encryption schemes namely inner-product functional encryption (IPFE) [6] and function-hiding inner product encryption (FHIPE) [7] respectively for secure activation computation.

III. The proposed method

Fig. 1 depicts our proposed method. The input layer consists of an encrypted vector of N elements. If the input data fed to the neural network is not vectorized, it must be vectorized. For example, if an input is a grayscale image, it should be flattened to a vector. Then, in the first hidden layer, we use our secure activation function called SecureReLU proposed in our previous work in [8]. We call it secure as it does ReLU operation and produces the results without having access to inputs in plain format. We consider the example of ReLU as it is the widely used activation function in the real world. Our approach can also support other activation functions by changing the activation after the inner-product operation. SecureReLU function takes the input encrypted vector X encrypted using FHIPE or IPFE, weight matrix W, and a bias term. SecureReLU returns the results equivalent to ReLU(W^TX+b), which usually happens in the activation functions in modern neural networks. So, the SecureReLU

activation function looks like:
$A = ReLU(sk_{fe}$(W) * Encrypt(X) + b)
Here, the internals of the ReLU function are secured by either inner-product functional encryption or by function-hiding inner product encryption. IPFE or FHIPE is mainly used for computing products between weight matrix and vector X. Here, X is encrypted with mpk (master public key), and access to the result, i.e., the product of W and X, is given by using the strength of FE. We detail the working of encryption and SecureReLU function in Algorithm 1 and Algorithm 2, respectively. In the process of the first activation function in the forward propagation, SecureReLU is used. In the later stages, the processes may occur similar to regular neural networks. Similarly, the backpropagation is made secure by computing the inner products using FE.

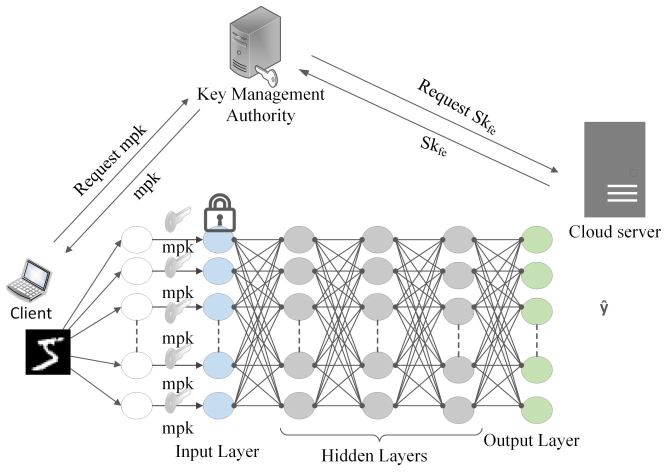

Fig. 1. Overview of Privacy-preserving Neural Network

Algorithm 1: FEncrypt

```
Input: X
Output: Ct
for $i \leftarrow 0$ to $X.size$ do
    Generation of mpk and msk by key management authority
    $Ct_i \leftarrow Encrypt(mpk_i, X)$
return Ct
```

Algorithm 2: SecureReLU

```
Input: Ct, W, b
Output: ReLU(W.X + b)
for $i \leftarrow 0$ to $W.size$ do
    $sk_{fe}.i \leftarrow KeyDerivation(msk_i, W.i)$
    $prod.i \leftarrow Decrypt(Ct_i, sk_{fe}.i, W.i)$
    $result.i \leftarrow ReLU(prod.i + b)$
return result
```

IV. Implementation and Results

We implement our methodology in Python. We take advantage of many predefined libraries in Python and C. We use NumPy for all the scientific calculations. We extensively use CiFEr library [9] proposed by Marc et al. for functional encryption-related computations. Based on the speedup achieved in our previous work in [8] for IPFE and FHIPE, we implement a complete machine learning pipeline for training a machine learning model shown in Fig. 1 on the MNIST dataset. Our implementation based on IPFE achieves 93.14% accuracy, and the training time required is 2 hours. Our other implementation based on FHIPE achieves 92.97% accuracy with a training time of 26 hours. Please note that the cryptographic computations are very complex in terms of time, so the training time required is more than the regular neural networks. However, compared to the previous work in this area, our IPFE-based implementation achieves 28X speedup, and the FHIPE-based implementation achieves 2X speedup, as shown in Table I.

TABLE I
Accuracy and training time

Method	Training time (hrs)	Accuracy	Epoch
Xu et al. [4]	57	95.49 %	2
Ours (FHIPE)	26	92.97 %	2
Ours (IPFE)	2	93.14 %	2

The accuracy achieved by our work is slightly lower than that of earlier work in this field; nevertheless, the training time required by our implementation is significantly shorter. In our future work, we will work on reducing the training time and improving the accuracy of the machine learning model.

References

[1] E. D. Sans, R. Gay, and D. Pointcheval, "Reading in the dark: Classifying encrypted digits with functional encryption." *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 206, 2018.
[2] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.
[3] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv preprint arXiv:1711.05189*, 2017.
[4] R. Xu, J. B. Joshi, and C. Li, "Cryptonn: Training neural networks over encrypted data," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1199–1209.
[5] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography Conference*. Springer, 2011, pp. 253–273.
[6] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," in *IACR International Workshop on Public Key Cryptography*. Springer, 2015, pp. 733–751.
[7] A. Bishop, A. Jain, and L. Kowalczyk, "Function-hiding inner product encryption," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2015, pp. 470–491.
[8] P. Panzade and D. Takabi, "Towards faster functional encryption for privacy-preserving machine learning," in *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 2021, pp. 21–30.
[9] T. Marc, M. Stopar, J. Hartman, M. Bizjak, and J. Modic, "Privacy-enhanced machine learning with functional encryption," in *European Symposium on Research in Computer Security*. Springer, 2019, pp. 3–21.