2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC) | 978-1-7281-9829-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/IPCCC50635.2020.9391544

Privacy Preserving Inference with Convolutional Neural Network Ensemble

Alexander Xiong*
Rice University
Houston, TX, USA
awx1@rice.edu

Michael Nguyen*
Independent Researcher
Garden Grove, CA, USA
michaelnguyenvtol@gmail.com

Andrew So
Cal Poly Pomona
Pomona, CA, USA
acso@cpp.edu

Tingting Chen
Cal Poly Pomona
Pomona, CA, USA
tingtingchen@cpp.edu

***Abstract*—Machine Learning as a Service on cloud not only provides a solution to scale demanding workloads, but also allows broader accessibility for the utilization of trained deep neural networks. For example, in the medical field, cloud-based deep-learning assisted diagnoses can be life-saving, especially in developing areas where experienced doctors and domain expertise are lacking. However, preserving end-users' data privacy while using cloud service for deep learning is a challenge. Some recent works based on fully homomorphic encryption have enabled neural-network predictions on encrypted input data. In this paper, we further extend the capability of privacy preserving deep neural network inference, through a joint decision made by multiple deep neural network models on encrypted data, to address bias caused by unbalanced local training datasets. In particular, we design and implement a privacy preserving prediction method through an ensemble of convolutional neural networks. The extensive experiment results show that our method can achieve higher accuracy compared to individual models, and preserve the user data privacy at the same level. We also verify the time efficiency of our implementation.**

***Index Terms*—Deep learning, Privacy, Convolutional Neural Networks Ensemble, Fully Homomorphic Encryption**

I. Introduction

Deep learning provides powerful data analysis that can automatically extract representative features. It has the capacity to transform many aspects of our lives, especially when deep learning services from cloud can be accessible by end-users world-wide. In the medical field, deep learning assisted diagnoses can be life-saving, for example, in developing areas where experienced doctors and domain expertise are lacking. However, popular deep learning algorithms were typically developed without a thorough consideration of data security and privacy issues. For sensitive data such as medical records, privacy issues have become a critical challenge in harvesting the promising benefits of deep learning [1].

For deep learning data privacy issues, due to the complexity of deep learning models and the high volume of training and testing data, traditional data security solutions fall short in efficiency, flexibility, and scalability [2], [3]. Recently, there have been some emerging works in secure deep learning as a service on the cloud, e.g., [4]–[6]. These works enable privacy preserving deep model predictions based on encrypted input data uploaded by user, which returns the same prediction result as with cleartext input. We follow the procedure framework in previous works as shown in Figure 1. Fully homomorphic encryption [7] is employed here to allow deep learning models to perform predictions based on encrypted data, without knowledge of the raw input data and the prediction result.

This work has been supported in part by NSF grant CNS-1758017, and a gift fund from Microsoft Inc. ⋆ denotes equal contribution to this work.

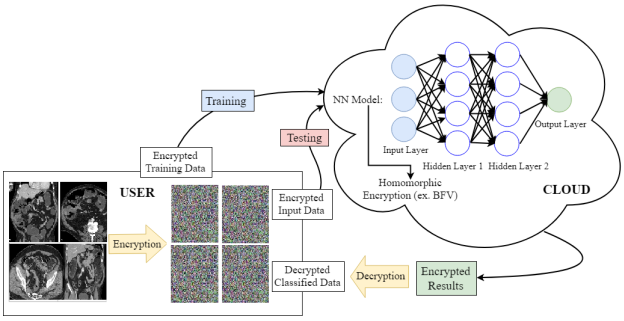


Fig. 1. **Architecture of Privacy Preserving Deep Neural Network as a Service on Cloud.** The end-user encrypts their private data (such as medical images) and uploads it to the cloud. Prediction based on the encrypted data is performed on cloud using the deep learning model. Cloud only holds the end-user's public key, and thus cannot decrypt the input or any computation result. The encrypted prediction result returned is decrypted by the end-user.

In this paper, we advance the encrypted deep learning inference technique one step further, by enabling encrypted deep learning ensemble models. In the real world, when creating artificial neural networks, multiple models are often created for two reasons: 1) data is distributed at different locations or owned by different parties; 2) even when the data is owned by the same party, creating multiple models and combining them often produces a desired output as opposed to creating just one model, to achieve low-bias and low-variance [8]. It is important to allow a deep network ensemble model to perform predictions over encrypted data, which addresses the realistic data distribution issue, enabling better prediction results and preserving end-users' data privacy.

A. Contributions

Our particular contributions in this paper are as listed below.

- We design a privacy preserving inference algorithm with Convolutional Neural Network (CNN) Ensemble. To the best of our knowledge, it is the first work that enables

978-1-7281-9829-3/20/$31.00 ©2020 IEEE

deep neural network ensemble prediction on encrypted data.
- We conduct analysis on the accuracy, security and efficiency of our algorithm.
- We implement and test our encrypted CNN ensemble model, and perform extensive experiments using MNIST dataset [9]. Our evaluation results verify the correctness of our algorithm in maintaining the test accuracy advantage of ensemble compared to individual models. We also demonstrate that our algorithm is time efficient.

The rest of this paper is organized as follows. Section II discusses the related work. Section III covers the preliminaries of this research. Section IV introduces the system model and security model of our privacy preserving inference with CNN ensemble. Section V presents the detailed algorithm built for this system, and the analysis of security, efficiency and correctness of the algorithm. Lastly, in our Sections VI and VII, we discuss the system implementation and experiment results on accuracy and efficiency.

II. Related Works

In the context of sensitive data protection, such as modern medical data, different differential privacy techniques for anonymization, such as k-anonymity, l-diversity, and t-closeness, are employed to protect the socio-demographic and private data supplied by patients. However, studies have shown how machine learning techniques can be used to re-identify a specific person with high accuracy even after de-identification [3]. Thus, enhanced privacy-preserving techniques are necessary.

Within the intersection of privacy and deep learning, the existing work on privacy preserving deep neural network prediction can be classified into two types: fully homomorphic evaluation of deep neural networks [5], [10]–[12], and multi-party computation (MPC)-based approaches (utilizing additively homomorphic encryptions) [13]–[16]. The first type can be applied in the cloud setting while the MPC-based approaches are usually for two-party scenarios. The goal of a privacy-preserving model is to prevent any user from gaining knowledge regarding the model (ie. training data used) and the model from gaining knowledge of the test inputs. This proposed work belongs to the first category, e.g., making predictions on cloud-based encrypted neural network models given an encrypted test data (image).

Other works have built on top of CryptoNets to implement speedup by employing sparse representation and minimizing overhead for encryption [6]. Other researchers have also explored the limits of CryptoNets' architecture, while still enabling inference and improving the latency of the model [17]. Lou et. al. use a Leveled-HE (LHE) approach over Torus in conjunction with ReLU activations to avoid bootstrap overheads while minimizing multiplicative overhead [18]. While previous works, including CryptoNets, use homomorphic encryption to solve the issue of data privacy [5], the difference between our proposed work and the existing work is that we are using the homomorphic crypto-system BGN, which has not been applied before, for its efficiency and capacity to integrate GPU acceleration.

III. Preliminaries

A. Fully Homomorphic Encryption

In cryptosystems, homomorphic schemes allow for arithmetic operations (i.e., addition and/or multiplication) on encrypted data. A cryptosystem is deemed multiplicatively homomorphic: if given two ciphertexts $Encrypt(pk, a)$ and $Encrypt(pk, b)$, the product of the two ciphertexts equals $Encrypt(pk, a * b)$. Similarly, it is additively homomorphic if there is a way to calculate $Encrypt(pk, a+b)$ based on the ciphertexts of a and b, without decrypting any message [7]. A cryptosystem is called Fully Homomorphic if it satisfies both conditions.

B. Encryption Scheme

Since the first solution for Fully Homomorphic Encryption (FHE) was proposed by Gentry [19], there have been many other FHE schemes introduced in the field. In this paper, we use the Brakerski/Fan-Vercauteren (BFV) scheme, a lattice-based cryptographic scheme dependent on the Ring Learning With Errors problem [20].

In the BFV scheme, plaintexts are elements of the ring R_t, and ciphertexts are elements of the ring $R_q \times R_q$, where $R_t = Z_t[x]/(x^n + 1)$, $R_q = Z_q[x]/(x^n + 1)$, with positive integers $q \gg t$ and n a power of 2. Z_q is the set of polynomials with integer coefficient in range $[0, q - 1)$. R_q is the set of all polynomials of degree at most is $n - 1$, with coefficients integers modulo q. Z_t and R_t are defined in the similar way.

To generate public-private key pair (pk, sk) for the BFV scheme, sample $s \leftarrow U_3^n$, $a \leftarrow U_q^n$, $e \leftarrow \chi^n$, where U_k denotes a uniform distribution on $Z \cap [-k/2, k/2)$, χ denotes a narrow discrete Gaussian error distribution, and from the given distributions n coefficients are sampled. s, a, e are considered as elements of R_q. We let $pk = ([-(as + e)]_q, a)$, $sk = s$, where $[\cdot]_q$ denotes coefficient-wise reduction modulo q.

To encrypt a plaintext $m \in R_t$ with public key $pk = (p_0, p_1)$, create

$$c = ([\lfloor q/t \rfloor m + p_0 u + e_1]_q, [p_1 u + e_2]_q)$$

where $u \leftarrow U_3^n$, $e_1, e_2 \leftarrow \chi^n$. The corresponding decryption formula for ciphertext $c = (c_0, c_1)$ is

$$m = \lfloor \frac{t}{q}(c_0 + c_1 s) \rceil_t$$

where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

For two ciphertexts $c = (c_0, c_1)$ and $c' = (c'_0, c'_1)$, $c + c' = ([c_0 + c'_0]_q, [c_1 + c'_1]_q) = ([\lfloor q/t \rfloor (m + m') + p_0(u + u') + e_1 + e'_1]_q, [p_1(u + u') + e_2 + e'_2]_q)$, which decrypts to $m + m'$. For multiplicatively homomorphic property, we refer readers to [21] for more details.

IV. System Overview

In this section, we describe the overall structure of our work and the security model that our security analysis is based on.

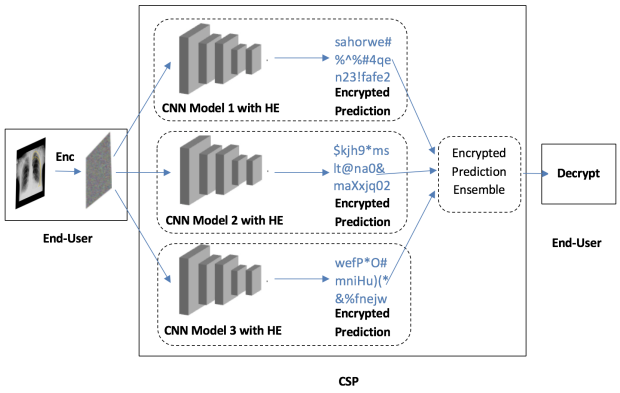


Fig. 2. System Model of Inference based on Encrypted CNN ensemble

A. System Model

In the system, we have three types of entities, i.e., 1) End-users who holds the raw input data and would like to make predictions using the cloud ensemble service; 2) Cloud Service Platform (CSP) who owns multiple trained neural networks built based on different cleartext local datasets; 3) Certificate Authority who manages the key pairs for the fully homomorphic encryption scheme. It is assumed that the cloud platform has the end-user's public key, and only the end-user has their own private key.

Figure 2 shows the overall structure of our privacy preserving inference based on encrypted CNN ensemble. End-users first encrypt their data using the fully homomorphic encryption public key and send the ciphertext to the cloud. Then at the CSP, our privacy preserving CNN ensemble will take the ciphertext as input and return the prediction result in the ciphertext space to the end-user. Finally, end-user decrypts it and obtains the clear-text inference result based on the ensemble network. In Figure 2, we illustrate an example of a CNN ensemble with three models on encrypted input.

B. Security Model

In our system, we assume that all parties, i.e., the cloud service platform, certificate authority, and end-users are all honest but curious; who will honestly follow the protocol but will try to discover others' private data as much as possible. The parties do not fully trust each other. End-users do not disclose private data or private key to the CSP. The data that CSP receives and operates on are all encrypted. We will show that they can not learn the private data from the procedure of executing the privacy preserving neural network ensemble inference and from the encrypted intermediate results. The deep neural network model owned by CSP are not known by end-users.

V. Privacy Preserving Inference using Convolutional Neural Network Ensemble

Among different deep neural network models, we choose to focus on solving the user privacy issues when using the inference service based on Convolutional Neural Networks due to its popularity and wide application in classifying images. In this section, we present the details of our privacy preserving inference algorithm based on CNN ensemble. In addition, we provide some theoretical analysis on security and communication overhead.

Algorithm 1 Inference with Encrypted CNN Ensemble

INPUT: Client holds cleartext data x, and a pair of keys (pk, sk) for the fully homomorphic encryption scheme. Cloud Service Provider owns multiple pre-trained models $M_1, M_2, ..., M_n$. CSP holds client's public key pk.

OUTPUT: Client obtains clear-text prediction result based on the CNN ensemble on CSP.

1: Client encrypts x with a public key pk to get $Enc(pk, x)$.
2: $Enc(pk, x)$ is uploaded to the cloud service provider.
3: $Enc(pk, x)$ propagrates through the distinct CNN models. Our privacy preserving models produces a set of predictions in ciphertext space $P_i = M_i.predict(Enc(pk, x))$, for $1 \leq i \leq n$.
4: CSP consolidates the predictions in the ciphertext space $Ensemble(P_1, P_2, ..., P_n)$.
5: $Ensemble(P_1, P_2, ..., P_n)$ is returned back to the client.
6: The client decrypts $Ensemble(P)$ with a private key sk to obtain the clear-text ensemble prediction $Dec(sk, Ensemble(P_1, P_2, ..., P_n))$.

A. Algorithm Design

The inference procedure with our CNN ensemble based on encrypted input is summarized in Algorithm 1. The main idea of this algorithm is to utilize the fully homomoprhic encryption scheme, e.g., BFV, to allow users to encrypt their image before sending it to the cloud. On the cloud, there are multiple existing pre-trained models based on different training datasets.

The goal of our algorithm is to design an evaluation function in the ciphertext domain for CNN ensemble, which takes in the encrypted image as input test data for each pretrained model. The ciphertext ensemble output can be decrypted by the user into the correct cleartext inference result. The challenge is to preserve the same properties in the ciphertext domain as in the cleartext domain for both the feed-forward inference process of individual models and the ensemble stage. Individual CNN feed-forward process can be viewed as a sequence of linear and nonlinear operations. Based on CryptoNet [5], we are able to find appropriate linear or low-degree polynomial functions to replace some nonlinear operations, so that it is easy to implement in the ciphertext domain with additive and multiplicative homomorphic properties. Then for each model M_i in our algorithm, we can guarantee that $M_i.predict(Enc(pk, x)) = Enc(M_i.predict(pk, x))$. More details for individual models can be found in Section VI. The ensemble method we introduce in this Section is based on the simple idea of average ensemble in order to demonstrate the viability of the structure of our work. This work can be

extended with other linear or low degree polynomial ensemble models [22].

B. Correctness Analysis

The correctness of our algorithm is guaranteed by homomorphic properties of the encryption scheme. As mentioned above, each model preserves the correctness of prediction results in the ciphertext space, i.e., equal to cleartext prediction result after decryption. Furthermore, since our ensemble method is linear, the correctness of ensemble step can also be guaranteed. The formal proof sketch is shown as below.

$$
\begin{aligned}
&Dec(sk, Ensemble(P_1, P_2, ..., P_n)) \\
=&Dec(sk, \sum_{i=1}^{n} P_i/n) \\
=&Dec\left(sk, \frac{\sum_{i=1}^{n} M_i.predict(Enc(pk, x))}{n}\right) \\
=&Dec\left(sk, \frac{\sum_{i=1}^{n} Enc(pk, M_i.predict(x))}{n}\right) \\
=&Dec\left(sk, Enc\left(pk, \sum_{i=1}^{n} \frac{M_i.predict(x)}{n}\right)\right) \\
=&\sum_{i=1}^{n} \frac{M_i.predict(x)}{n}
\end{aligned}
$$

C. Security Analysis

Here we provide a brief security analysis of our privacy preserving CNN ensemble against semi-honest CSP. In Algorithm 1, the message that CSP receives is the encrypted image data. CSP cannot learn the plaintext x, i.e., the pixel values of original image, due to the security of BFV which is based on the difficulty of solving the Ring Learning with Error (RLWE) problem [23].

D. Efficiency Analysis

The messages transmitted between the user and CSP are $Enc(pk, x)$ and $Ensemble(P_1, P_2, ..., P_n)$. If the dimension of single grayscale image x is $m \times m$, the size of $Enc(pk, x)$ is $m^2 \times n$ bytes, where n is determined by the key setup parameters in BFV, and in our implementation, $n = 65544 \times 8$. The size of $Ensemble(P_1, P_2, ..., P_n)$ is $K \times n$, where K is the number of potential classes in the prediction results, and in our implementation, $K = 10$. We will evaluate the time efficiency of our work in Section VII.

VI. Implementation

In our implementation, we use C#, Python, and Keras to create our neural network which similarly follows the design set forth in CryptoNet. With Python and Keras, we reconstruct the Keras equivalent of the CryptoNet neural network. In doing so, we are able to train our model in Keras and transfer our weights and biases into the CryptoNet. We also use C# to integrate the logic to produce the ensemble prediction in CryptoNet and use its integration of Microsoft SEAL for homomorphic encryption functionalities.

More specifically, the CNN Ensemble consists of 3 models each of which has the architecture defined in Table I and is trained with 3 distinct distributions of Keras's MNIST dataset and validated with a testing MNIST dataset as described in Section VII. However, before we can train the dataset with the architecture, the input shape must be padded with zeros at the bottom row and right column to transform the original 28 x 28 shape to 29 x 29.

TABLE I
KERAS MODEL ARCHITECTURE

#	Layer	Parameters	Output Shape
1	Convolution	(5x5) size, (2,2) strides, 5 maps	13 x 13 x 5
2	Square Activation	-	13 x 13 x 5
3	Flatten	-	845
4	Dense	100 outputs	100
5	Square Activation	-	100
6	Dense	10 outputs	10
7	Softmax Activation	10 outputs	10

After training, the weights and biases are broken down into the 3 corresponding blocks defined in the CryptoNet design as shown in Table II. Additionally, we can add the activation function between each block without affecting the weights and biases.

TABLE II
FEEDFORWARD BLOCKS

#	Block
1	Convolution Block
-	Square Activation
2	Dense Block
-	Square Activation
3	Dense Block

After the segmentation, we append the weights and biases back together and repeat this process 3 times, one for each of the 3 models in our CNN Ensemble. We integrate these 3 sets of weight and bias .csv files into CryptoNet's input weights and biases such that the input dataset can propagate through 3 different models and produce an average of the 3 resulting predictions.

VII. Experiments

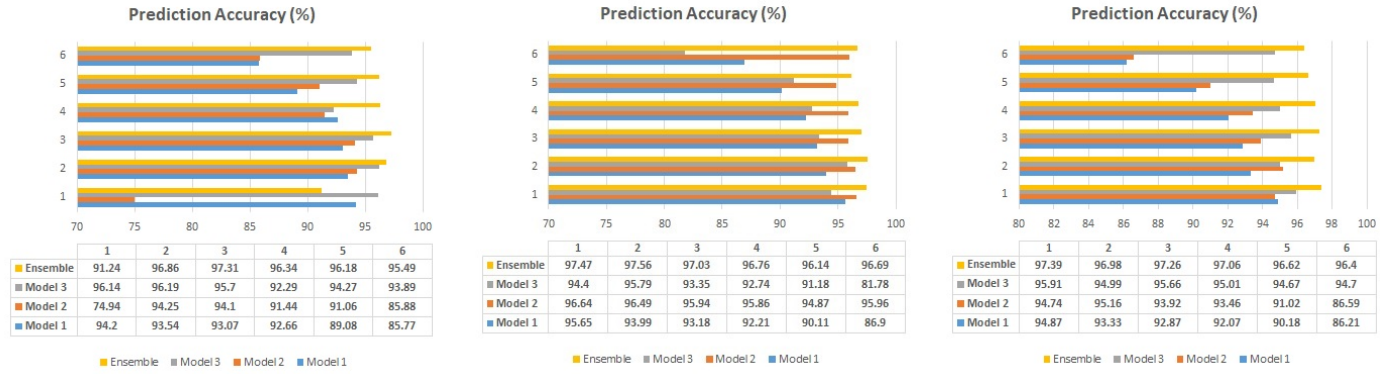
In this section, our experiment demonstrates the accuracy and efficiency of our encrypted CNN ensemble algorithm. We use an 64 bit Intel Core i9-9900k @ 3.6 GHz with 32 GB of RAM and Nividia GeForce RTx 2080 Ti as the base conditions to run our experiments.

A. Experiment Setup

To cover different data distribution scenarios, we use 3 different training data split methods to build individual CNN models in Keras, i.e., regular data split, likewise data split and random data split. Using the regular data split method, majority of digit "0","1","2" images are grouped to train individual Model 1, most "3","4","5" images for Model 2,

(a) Regular Data Split (b) Likewise Data Split (c) Random Data Split

Fig. 3. **Training Data Distributions based on 3 Different Data Split Methods.** For each method, we further break down into 6 split settings, where 3 individual CNN models are trained using subsets of data with different distributions as shown in this figure.



	1	2	3	4	5	6
Ensemble	91.24	96.86	97.31	96.34	96.18	95.49
Model 3	96.14	96.19	95.7	92.29	94.27	93.89
Model 2	74.94	94.25	94.1	91.44	91.06	85.88
Model 1	94.2	93.54	93.07	92.66	89.08	85.77

	1	2	3	4	5	6
Ensemble	97.47	97.56	97.03	96.76	96.14	96.69
Model 3	94.4	95.79	93.35	92.74	91.18	81.78
Model 2	96.64	96.49	95.94	95.86	94.87	95.96
Model 1	95.65	93.99	93.18	92.21	90.11	86.9

	1	2	3	4	5	6
Ensemble	97.39	96.98	97.26	97.06	96.62	96.4
Model 3	95.91	94.99	95.66	95.01	94.67	94.7
Model 2	94.74	95.16	93.92	93.46	91.02	86.59
Model 1	94.87	93.33	92.87	92.07	90.18	86.21

(a) Accuracy results with regular data split (b) Accuracy results with likewise data split (c) Accuracy results with random data split

Fig. 4. Test Accuracy of Encrypted CNN Ensemble Compared with Individual Models Trained Based on 3 Data Split Methods.

and majority of "6","7","8","9" images for Model 3. In the second method, we group most of images for digits which look similar together to train one individual model. The particular partition is ({"0","6","8"},{"2","3","5","7"},{"1","4","9"}). In random data split method, we randomly group images with different labels to train individual CNN models. For each data split method, we have 6 different settings which vary in the range of percentage for each label. The detailed compositions of 6 different settings for each training data split method are illustrated in Figure 3. To measure the accuracy of our trained models, we use 10k images.

B. Evaluation of Accuracy

We evaluate the test accuracy of our encrypted CNN ensemble using different pre-trained models as described in the experiment setup. In this subsection, we focus on comparing the accuracy of encrypted ensemble to those of the individual models in order to show the advantage our work over existing secure CNN models. See Figure 4(a), 4(b), and 4(c) for a summary of results.

In Figure 4, we observe that the test accuracy levels of our encrypted CNN ensemble models across different settings are consistently high, in the range of 96%-97%. In addition, except one setting using regular data split method, the ensemble produces a higher test accuracy than individual CNN models. This implies that building ensembles based on multiple networks has its strengths in achieving lower bias and thus higher test accuracy. Our algorithm enables end-users to utilize more powerful deep learning models in a privacy preserving manner.

C. Evaluation of Efficiency

As the computing time and resources are an important factor in deep learning, we evaluate the efficiency of our algorithm. First we breakdown the run-time for the individual CNN model and measure the time it takes to complete each layer for each batch in the model. The result is shown in Table III. It can be noted that we took out the StartTimingLayer and StopTimingLayer as both times are essentially zero and have minimal impact to our evaluation of time. Based on our different test settings as described in Section VII A, we average our overall runtime per layer per batch and conclude that the most time consuming layers in the encrypted CNN model are the Activation Layer, the Dense Layer and Convolutional Layer, which contain a significant number of ciphertext addition and multiplication operations.

In addition, we investigate how increasing the input size would affect the total run-time of predicting the labels for the data in the experiment. By controlling the input size on the data that we use for the experiment, our aim is to see if there is a linear relationship between elapsed time and input size.

From Figure 5 we can see that the total elapsed time increases roughly as expected, i.e., linearly to input size.

TABLE III
AVERAGE TIME ELAPSED IN SECONDS PER LAYER

Layer	Seconds
EncryptedLayer	1.126
ConverLayer1	2.930
ActivationLayer2	6.308
DenseLayer3	6.853
ActivationLayer4	0.741
DenseLayer5	0.117

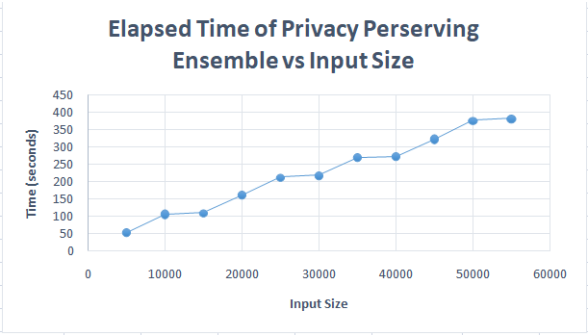


Fig. 5. Elapsed Time with Different Input Sizes

Overall, our algorithm can efficiently make inferences for 50,000 images in less than 7 minutes.

VIII. CONCLUSION

This work advances the fully homomorphic encryption based secure deep learning technique by enabling encrypted CNN ensembles. In particular, we present a privacy preserving CNN ensemble algorithm which takes encrypted images as input and produces encrypted prediction results based on average ensemble of multiple pre-trained CNN models. Our encrypted CNN ensemble model preserves inference correctness (i.e., same prediction result as using cleartext input) and user data privacy. Our experiments on MNIST have verified the accuracy and efficiency of our algorithm.

REFERENCES

[1] H. Bae, J. Jang, D. Jung, H. Jang, H. Ha, and S. Yoon, "Security and privacy issues in deep learning," *arXiv preprint arXiv:1807.11655*, 2018.

[2] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 439–450.

[3] L. Rocher, J. M. Hendrickx, and Y.-A. de Montjoye, "Estimating the success of re-identifications in incomplete datasets using generative models," *Nature Communications*, vol. 10, no. 1, p. 3069, 2019. [Online]. Available: https://doi.org/10.1038/s41467-019-10933-3

[4] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig, "Crypto-nets: Neural networks over encrypted data," *arXiv preprint arXiv:1412.6181*, 2014.

[5] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," 2018. [Online]. Available: https://www.microsoft.com/en-us/research/publication/cryptonets-applying-neural-networks-to-encrypted-data-with-high-throughput-and-accuracy/

[6] E. Chou, J. Beal, D. Levy, S. Yeung, A. Haque, and L. Fei-Fei, "Faster cryptonets: Leveraging sparsity for real-world encrypted inference," 2018. [Online]. Available: http://arxiv.org/abs/1811.09953

[7] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[8] D. W. Opitz and J. W. Shavlik, "Generating accurate and diverse members of a neural-network ensemble," in *Advances in neural information processing systems*, 1996, pp. 535–541.

[9] "The mnist database." [Online]. Available: http://yann.lecun.com/exdb/mnist/

[10] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," Cryptology ePrint Archive, Report 2017/1114, 2017. [Online]. Available: https://eprint.iacr.org/2017/1114

[11] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network," Cryptology ePrint Archive, Report 2017/035, 2017. [Online]. Available: https://eprint.iacr.org/2017/035

[12] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," Cryptology ePrint Archive, Report 2018/1041, 2018. [Online]. Available: https://eprint.iacr.org/2018/1041

[13] C. Orlandi, A. Piva, and M. Barni, "Oblivious neural network computing via homomorphic encryption," *EURASIP Journal on Information Security*, vol. 2007, no. 1, p. 037343, 2007. [Online]. Available: https://doi.org/10.1155/2007/37343

[14] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 19–38.

[15] D. Demmler, T. Schneider, and M. Zohner, "Aby - a framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.

[16] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," Cryptology ePrint Archive, Report 2017/452, 2017. [Online]. Available: https://eprint.iacr.org/2017/452

[17] A. Brutzkus, R. Gilad-Bachrach, and O. Elisha, "Low latency privacy preserving inference," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 812–821. [Online]. Available: http://proceedings.mlr.press/v97/brutzkus19a.html

[18] Q. Lou and L. Jiang, "She: A fast and accurate deep neural network for encrypted data," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 10 035–10 043. [Online]. Available: http://papers.nips.cc/paper/9194-she-a-fast-and-accurate-deep-neural-network-for-encrypted-data.pdf

[19] C. Gentry, S. Halevi, and N. P. Smart, "Better bootstrapping in fully homomorphic encryption," in *International Workshop on Public Key Cryptography*. Springer, 2012, pp. 1–16.

[20] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." [Online]. Available: https://eprint.iacr.org/2012/144

[21] J.-C. Bajard, J. Eynard, M. A. Hasan, and V. Zucca, "A full rns variant of fv like somewhat homomorphic encryption schemes," in *International Conference on Selected Areas in Cryptography*. Springer, 2016, pp. 423–442.

[22] U. Naftaly, N. Intrator, and D. Horn, "Optimal ensemble averaging of neural networks," *Network: Computation in Neural Systems*, vol. 8, no. 3, pp. 283–296, 1997.

[23] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.