HDNN: Homomorphic Decryption Neural Network

C.H. Chen

This study proposes a homomorphic decryption layer to construct a homomorphic decryption neural network (HDNN) for improving security and reducing computational time. The ciphertexts of inputs can be adopted as the values of neurons in the input layer of HDNN for estimating the plaintexts of outputs. The results showed the proposed method can obtain high performance and save computational resources.

Introduction: In recent years, the applications of Internet of Things have been more and more popular, and a variety of data have been produced. Security issues are important for protecting the privacy of data. However, each encrypted ciphertext should be decrypted for computation with using several computational resources by traditional methods (e.g. RSA or ECC). Therefore, a homomorphic decryption neural network with a homomorphic decryption layer is proposed for improving security and saving computational resources.

Homomorphic Encryption and Decryption: This section designs three cases to explain how to apply homomorphic encryption and decryption functions to inputs: (1) Case 1: the case of integer inputs; (2) Case 2: the case of float inputs; (3) Case 3: the case of weighted inputs.

In Case 1, the function $E(x_i)$ denotes a homomorphic encryption function with a private key p_y (i.e. a prime number), a public key p_c (i.e. a prime number) and a random integer number γ for encrypting the plaintext of the i-th input x_i (i.e. an integer number) to the ciphertext of the i-th input $\tilde{x}_i$ (shown in Equation (1)). Furthermore, the function $D(\tilde{x}_i)$ which denotes a homomorphic decryption function can be used to decrypt the ciphertext of the i-th input $\tilde{x}_i$ to the plaintext of the i-th input x_i (shown in Equation (2)) [1]. The summary of ciphertexts can be decrypted by Equation (3) for obtaining the summary of inputs.

$$\tilde{x}_i := E(x_i) \equiv (x_i + p_y \times \gamma) (\bmod p_y \times p_c) \cdot \quad (1)$$

$$x_i := D(\tilde{x}_i) \equiv \tilde{x}_i (\bmod p_y) \cdot \quad (2)$$

$$\sum_{i=1}^{n} x_i := D\left(\sum_{i=1}^{n} \tilde{x}_i\right) := D\left(\sum_{i=1}^{n} E(x_i)\right) \cdot \quad (3)$$

In Case 2, this study proposes a transformation function $T(x_i, \kappa_I)$ with a parameter ω_I which is ten to the power of κ_I (shown in Equation (4)) for transforming the i-th float input x_i into ς_i. Furthermore, the transformed plaintext ς_i can be encrypted to the ciphertext $\tilde{\varsigma}_i$ by Equation (5), and the ciphertext $\tilde{\varsigma}_i$ can be decrypted to the transformed plaintext ς_i Equation (6). Therefore, the summary of ciphertexts can be decrypted by Equation (7) for obtaining the summary of transformed plaintexts. Furthermore, this study proposes a de-transformation function to de-transform the results of Equation (7) for estimating the summary of float inputs.

$$\varsigma_i = T(x_i, \kappa_I) = \lfloor x_i \times \omega_I \rfloor \text{ and } \omega_I = 10^{\kappa_I} . \quad (4)$$

$$\tilde{\varsigma}_i := E(\varsigma_i) \equiv (\varsigma_i + p_y \times \gamma) (\bmod p_y \times p_c) \cdot \quad (5)$$

$$\varsigma_i := D(\tilde{\varsigma}_i) \equiv \tilde{\varsigma}_i (\bmod p_y) \cdot \quad (6)$$

$$\sum_{i=1}^{n} \varsigma_i := D\left(\sum_{i=1}^{n} \tilde{\varsigma}_i\right) := D\left(\sum_{i=1}^{n} E(\varsigma_i)\right) \cdot \quad (7)$$

$$\sum_{i=1}^{n} x_i \approx \frac{D\left(\sum_{i=1}^{n} \tilde{\varsigma}_i\right)}{\omega_I} . \quad (8)$$

In Case 3, this study extends the proposed transformation function (i.e. Equation (4)) to propose the another transformation function $R(x_i, \kappa_I, w_i, \kappa_W)$ with a parameter ω_W which is ten to the power of κ_W (shown in Equation (9)) for transforming the i-th float input x_i into ζ_i. Furthermore, the transformed plaintext ζ_i can be encrypted to the ciphertext $\tilde{\zeta}_i$ by Equation (10), and the ciphertext $\tilde{\zeta}_i$ can be decrypted to the transformed plaintext ζ_i Equation (11). Therefore, the summary of ciphertexts can be decrypted by Equation (12) for obtaining the summary of transformed plaintexts. Furthermore, this study proposes a de-transformation function to de-transform the results of Equation (13) for estimating the summary of weighted inputs. In accordance with Equation (14) and Equation (15), the ciphertext $\tilde{\zeta}_i$ can be adopted to estimate the summary of weighted inputs by Equation (16).

$$\zeta_i = R(x_i, \kappa_I, w_i, \kappa_W) = \lfloor w_i \times \omega_W \rfloor \times \lfloor x_i \times \omega_I \rfloor = \lfloor w_i \times \omega_W \rfloor \times \tilde{\zeta}_i , \quad \omega_W = 10^{\kappa_W} \text{ and } \omega_I = 10^{\kappa_I} . \quad (9)$$

$$\tilde{\zeta}_i := E(\zeta_i) \equiv (\zeta_i + p_y \times \gamma) (\bmod p_y \times p_c) \cdot \quad (10)$$

$$\zeta_i := D(\tilde{\zeta}_i) \equiv \tilde{\zeta}_i (\bmod p_y) \cdot \quad (11)$$

$$\sum_{i=1}^{n} \zeta_i := D\left(\sum_{i=1}^{n} \tilde{\zeta}_i\right) := D\left(\sum_{i=1}^{n} E(\zeta_i)\right) \cdot \quad (12)$$

$$\sum_{i=1}^{n} w_i x_i \approx \frac{D\left(\sum_{i=1}^{n} \tilde{\zeta}_i\right)}{\omega_W \omega_I} . \quad (13)$$

$$\tilde{\zeta}_i (\bmod p_y) := (\lfloor w_i \times \omega_W \rfloor \times \tilde{\zeta}_i) (\bmod p_y) \cdot \quad (14)$$

$$\sum_{i=1}^{n} \zeta_i := D\left(\sum_{i=1}^{n} \lfloor w_i \times \omega_W \rfloor \times \tilde{\zeta}_i\right) \cdot \quad (15)$$

$$\sum_{i=1}^{n} w_i x_i \approx \frac{D\left(\sum_{i=1}^{n} \lfloor w_i \times \omega_W \rfloor \times \tilde{\zeta}_i\right)}{\omega_W \omega_I} . \quad (16)$$

Homomorphic Decryption Neural Network: This section proposes a homomorphic decryption layer to construct a homomorphic decryption neural network. The architectures of neural network and homomorphic decryption neural network are shown in Fig. 1 and Fig. 2, respectively. In these neural networks, the number of neurons in the input layer is n (i.e. n inputs), and the number of neurons in the output layer is q (i.e. q outputs). Furthermore, the number of neurons in the first hidden layer is equal to the number of neurons in the homomorphic decryption layer (i.e. m neurons).

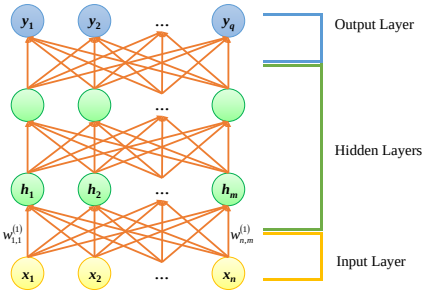

Fig. 1 *The architecture of neural network*

In training stage, the plaintexts of inputs are adopted as the values of neurons (i.e. x_1, x_2, …, and x_n) in the input layer. The values of neurons (i.e. h_1, h_2, …, and h_n) in the first hidden layer can be estimated by Equation (17), where the parameter $w_{i,j}^{(1)}$ denotes the weight of connection from the i-th neuron in the input layer to the j-th neuron in the first hidden layer. The familiar optimizers (e.g. stochastic gradient decent, etc.) and loss functions (e.g. mean squared error, binary cross-entropy, categorical cross-entropy, etc.) can be adopted to train the neural network for estimating the values of these q outputs.

$$h_j = \sum_{i=1}^{n} w_{i,j}^{(1)} x_i \cdot \quad (17)$$

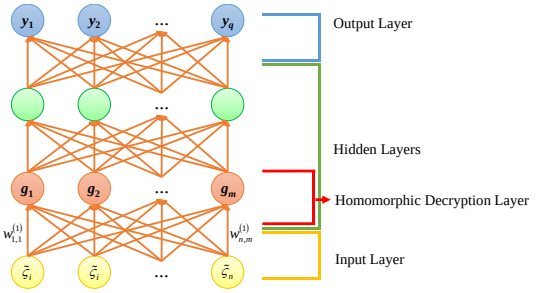

Fig. 2 *The architecture of homomorphic decryption neural network*

In testing and runtime stages, the inputs can be encrypted by Equation (4) and Equation (5) to obtain the ciphertexts of inputs which are stored in a database and adopted as the values of neurons (i.e. $\tilde{\varsigma}_1$, $\tilde{\varsigma}_2$, …, and $\tilde{\varsigma}_n$) in the input layer of homomorphic decryption neural network. The trained weights (e.g. $w_{i,j}^{(1)}$) in the first hidden layer in training stage can be adopted to estimate the value of the *j*-th neuron (e.g. g_j) in the homomorphic decryption layer by Equation (18). Furthermore, the trained weights in other layers can be adopted to estimate the values of outputs for predications and classifications.

$$g_j = \frac{D\left(\sum_{i=1}^{n} \left\lfloor w_{i,j}^{(1)} \times \omega_W \right\rfloor \times \tilde{\varsigma}_i\right)}{\omega_W \omega_I}. \tag{18}$$

Experimental Results and Discussions: Two experiments which were designed for the evaluation of the proposed homomorphic decryption neural network include: (1) a simple case for prediction and (2) an image recognition case for classification. In these experiments, this study adopt some parameters as follows to estimate outputs: p_y = 73939133, p_c = 59393339, γ = 2, κ_I = 3 and κ_W = 3. Furthermore, no biases were used in the first hidden layer, and the constraint of weights in the first hidden layer was non-negative.

In Experiment 1, Table 1 shows the used data of Experiment 1, and the structure of neural network with one hidden layer is 5-1-1 (i.e. n = 5; m = 1; q = 1) [2]. The parameters v_1, v_2, v_3, v_4 and v_5 were adopted as the inputs of neural network, and the parameter v_6 was adopted as the output of neural network. For training the neural network, the Nesterov-accelerated adaptive moment estimation (Nadam) and mean squared error function were adopted as the optimizer and loss function of neural network. In training stage, the trained weights $w_{1,1}^{(1)}$, $w_{2,1}^{(1)}$, $w_{3,1}^{(1)}$, $w_{4,1}^{(1)}$ and $w_{5,1}^{(1)}$ were 1.383, 0.397, 0, 0 and 0, respectively. In testing and runtime stages, the structure of homomorphic decryption neural network is 5-1-1, and the inputs were encrypted by Equation (4) and Equation (5) as the ciphertexts in Experiment 1 (shown in Table 2). The ciphertexts and trained weights can be used to estimate the values of neurons in the homomorphic decryption layer by Equation (18) for obtaining the estimated outputs as 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9 after rounding these output values to the nearest tenth. Therefore, the mean absolute error of homomorphic decryption neural network is zero in Experiment 1.

In Experiment 2, three images with 2×2 pixels (i.e. {{r_1, r_2}, {r_3, r_4}}) were collected in Table 3, and the structure of neural network with one hidden layer is 4-1-1 (i.e. n = 4; m = 1; q = 1) [2]. The first and second images showed a horizontal line, so the parameter r_5 was labelled as 0; the third image showed a slash line, so the parameter r_5 was labelled as 1. The parameters r_1, r_2, r_3 and r_4 were adopted as the inputs of neural network, and the parameter r_5 was adopted as the output of neural network. For training the neural network, the Nadam and binary cross-entropy function were adopted as the optimizer and loss function of neural network. In training stage, the trained weights $w_{1,1}^{(1)}$, $w_{2,1}^{(1)}$, $w_{3,1}^{(1)}$ and $w_{4,1}^{(1)}$ were 0, 4.913, 4.676, 0 and 0, respectively. In testing and runtime stages, the structure of homomorphic decryption neural network is 4-1-1, and the inputs were encrypted by Equation (4) and Equation (5) as the ciphertexts in Experiment 2 (shown in Table 4). The ciphertexts and trained weights can be used to estimate the values of neurons in the homomorphic decryption layer by Equation (18) for obtaining the estimated outputs as 0, 0 and 1 after rounding these output values to the nearest whole number. Therefore, the accuracy of homomorphic decryption neural network is 100% in Experiment 2.

Table 1: The data in Experiment 1.

v_1	v_2	v_3	v_4	v_5	v_6
0.1	0.1	0.3	0.5	0.7	0.3
0.2	0.1	0.3	0.5	0.7	0.4
0.3	0.1	0.3	0.5	0.7	0.5
0.4	0.1	0.3	0.5	0.7	0.6
0.5	0.1	0.3	0.5	0.7	0.7
0.6	0.1	0.3	0.5	0.7	0.8
0.7	0.1	0.3	0.5	0.7	0.9

Table 2: The ciphertexts in Experiment 1.

$\tilde{\varsigma}_1$	$\tilde{\varsigma}_2$	$\tilde{\varsigma}_3$	$\tilde{\varsigma}_4$	$\tilde{\varsigma}_5$
147878366	147878366	147878566	147878766	147878966
147878466	147878366	147878566	147878766	147878966
147878566	147878366	147878566	147878766	147878966
147878666	147878366	147878566	147878766	147878966
147878766	147878366	147878566	147878766	147878966
147878866	147878366	147878566	147878766	147878966
147878966	147878366	147878566	147878766	147878966

Table 3: The data in Experiment 2.

r_1	r_2	r_3	r_4	r_5
1	1	0	0	0
0	0	1	1	0
1	0	0	1	1

Table 4: The data in Experiment 2.

$\tilde{\varsigma}_1$	$\tilde{\varsigma}_2$	$\tilde{\varsigma}_3$	$\tilde{\varsigma}_4$
147879266	147879266	147878266	147878266
147878266	147878266	147879266	147879266
147879266	147878266	147878266	147879266

Conclusions: The proposed homomorphic decryption neural network can analyse the ciphertexts of inputs to estimate outputs with low error and high accuracy for predictions and classifications. Furthermore, the time complexity of decryption by the proposed method is $O(m)$, and the time complexities of decryption by RSA and ECC are $O(nm)$.

C.H. Chen
(*Telecommunication Laboratories, Chunghwa Telecom Co., Ltd.*)

E-mail: chchen.scholar@gmail.com

ORCID: 0000-0003-3628-3033

References

1. Tran, J., Farokhi, F., Cantoni, M., Shames, I.: 'Implementing homomorphic encryption based secure feedback control', *Control Engineering Practice*, 2020, **97**, pp. 104350, doi: 10.1016/j.conengprac.2020.104350
2. Chen, C.H.: Homomorphic decryption neural network. *GitHub*, https://github.com/ChiHua0826/HDNN, accessed October 2022.