

# TransNet: Training Privacy-Preserving Neural Network over Transformed Layer

Qijian He<sup>†</sup>, Wei Yang<sup>‡\*</sup>, Bingren Chen<sup>†</sup>, Yangyang Geng<sup>†</sup>, Liusheng Huang<sup>‡</sup>

School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

<sup>†</sup>{cnstrong, bingren, geng325}@mail.ustc.edu.cn, <sup>‡</sup>{qubit, lshuang}@ustc.edu.cn

## ABSTRACT

The accuracy of neural network can be improved by training over multi-participants' pooled dataset, but privacy problem of sharing sensitive data obstructs this collaborative learning. To solve this contradiction, we propose TransNet, a novel solution for privacy-preserving collaborative neural network, whose main idea is to add a transformed layer to the neural network. It has the advantage of lower computation and communication complexity than previous secure multi-party computation based and homomorphic encryption based schemes, and has the superiority of supporting arbitrarily partitioned dataset compared to previous differential privacy based and stochastic gradient descent based schemes, which support horizontally partitioned dataset only. TransNet is trained by a server which pools the transformed data, but has no special security requirement on the training server. We evaluate TransNet's performance over four datasets using different neural network algorithms. Experimental results demonstrate that TransNet is not affected by the number of participants, and trains as quickly as the original neural network does. With proper variables, TransNet gets close accuracy to the baseline which trains over pooled original dataset.

## PVLDB Reference Format:

Qijian He, Wei Yang, Bingren Chen, Yangyang Geng, Liusheng Huang. TransNet: Training Privacy-Preserving Neural Network over Transformed Layer. *PVLDB*, 13(11): 1849-1862, 2020.  
 DOI: <https://doi.org/10.14778/3407790.3407794>

## 1. INTRODUCTION

Recent developments of deep neural network (also called deep learning [18]) have been an enormous success in many fields. However, there is a serious contradiction between the performance of neural network model and the privacy of training datasets. For example, many applications use

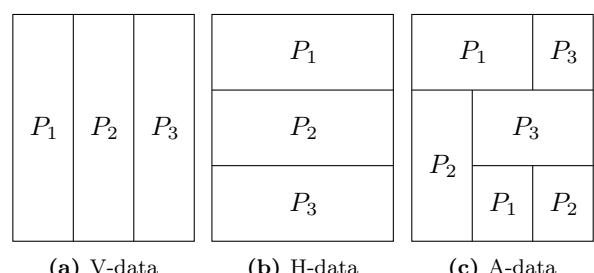
\*Corresponding Author. Supported by the Anhui Initiative in Quantum Information Technologies (No. AHY150300).

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Received at the VLDB Endowment  
*Proceedings of the VLDB Endowment*, Vol. 13, No. 11  
ISSN 2150-8097.  
DOI: <https://doi.org/10.14778/3407790.3407794>

data like personal contacts [21], medical records [22] and even genetic sequences [3], but they are unable to protect the privacy of users. Under such circumstances, the individuals will become less willing to provide their data with the awareness of privacy concerns. On the other hand, the companies are unwilling to share data with others because there are economic values in the data. In addition, some institutions like hospitals, banks and governments are unable to share the valuable datasets by the restriction of law. However, in these cases, people will benefit if they can collaboratively train a machine learning algorithm. Therefore, it is meaningful to design a privacy-preserving collaborative neural network scheme, within which the participants can obtain practical or better applications from partitioned datasets while the privacy is preserved.

In this paper, we focus on privacy-preserving collaborative neural network which is suitable for different *data partition types* (DPTs). In the real world, the participants may have separated data in different ways. For instance, special hospitals differ in the cases of one patient from each other, which is called *vertically partitioned dataset*. And general hospitals of several cities have complete cases of different patients, which is called *horizontally partitioned dataset*. More universally, hospitals have different cases of different patients, which is called *arbitrarily partitioned dataset*. The three DPTs are shown in Fig. 1, where the names are abbreviated to V-data, H-data and A-data respectively.



**Figure 1:** Three DPTs, where  $P_1$ ,  $P_2$  and  $P_3$  represent different participants.

The state-of-the-art methods design such schemes mainly based on homomorphic encryption (HE) [16] (or combined with secure multi-party computation (SMC) [39]), differential privacy (DP) [12] and stochastic gradient descent (SGD). HE based schemes [17, 23, 28, 36, 40] support A-data, but are slow in speed and costly in communication, which are not practical when the datasets become large. Moreover,

DP based scheme [27] and SGD based schemes [1, 25, 31] only support H-data.

To solve this problem, we propose *TransNet*, which trains privacy-preserving neural Network over a **Transformed** layer collaboratively. The basic idea of our design is to add an extra “transformed” layer to the neural network. The original privacy data are computed through an irreversible transformation to be converted into the transformed layer data, and then the deep neural networks can be trained over the transformed layer. Compared with previous SMC, HE and DP based approaches, our solution introduces a new method for this topic, i.e., using the indeterminate system to define the irreversible transformation in our scheme.

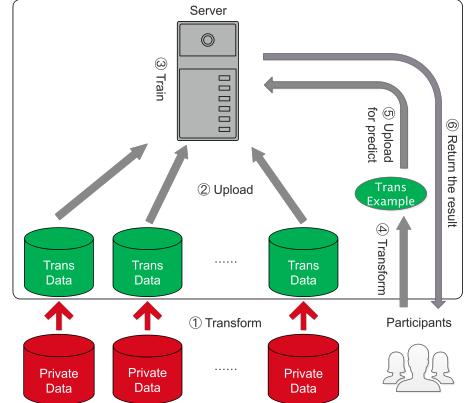
In summary, we make the following contributions:

- We present *TransNet*, which allows multiple participants to train a neural network collaboratively over their pooled dataset with a training server, while keeping their data private. We add a *transformed layer*, which is obtained by a noisy linear transformation, to the neural network to build *TransNet*. And the neural network can train over the transformed dataset. Intuitively, two close examples of the same classification should also be close after transformation, and thus the transformation should be continuous. We analyze why it works through Lipschitz continuity. Besides, *TransNet* is independent of the structure of neural network and has no special security requirement on the training server. These properties make the scheme easy to deploy. Namely, we can turn the current neural networks into *TransNet* method directly.
- We address the challenge of supporting vertically, horizontally and arbitrarily partitioned datasets by developing a method called *Indeterminacy Constitution*, and corresponding schemes are named as *TransNet-V*, *TransNet-H* and *TransNet-A*, respectively. Compared to DP or SGD based schemes, *TransNet* makes up for the incomplete support of DPTs and is not affected by the number of participants; Compared to SMC or HE based schemes, our method has lower computation and communication complexity.
- We perform comprehensive experiments over various datasets by using different kinds of neural networks including MLP, CNN and RNN. Experimental results demonstrate the effectiveness and advantages of *TransNet*. It shows that *TransNet-V* has no accuracy loss when using MLP and RNN. With *TransNet-H*, we find that adding 300 dimensional noise to the MNIST dataset only causes about 3% accuracy loss, and adding 600 dimensional noise causes only about 6% accuracy loss when using MLP. *TransNet-A*’s accuracy is slightly lower than that of *TransNet-H*. As a conclusion, *TransNet* achieves privacy-preservation at the cost of sacrificing only a little accuracy.

## 2. MODELS AND ASSUMPTIONS

### 2.1 System Model

The system contains two parties: a training server, and the participants. Its workflow can be divided into two phases, namely, the training phase and the prediction phase. Each participant first sends the server a 0/1 table to indicate



**Figure 2:** Workflow of *TransNet*, where ①, ②, ③ belong to training phase, and ④, ⑤, ⑥ belong to prediction phase.

the data it owns (including labels). Then the server determines the DPT and instructs the participants to manipulate the disjoint parts of the dataset. As shown in Fig. 2, in the training phase the participants transform their private data and upload them to the server for training (the labels are permuted to avoid the server from knowing the correspondences), while in the prediction phase the participants submit the transformed prediction examples and the server returns the results. The parts inside the outer box are public, while the private data marked with red color and the transformation outside the outer box are secret.

### 2.2 Security Assumptions

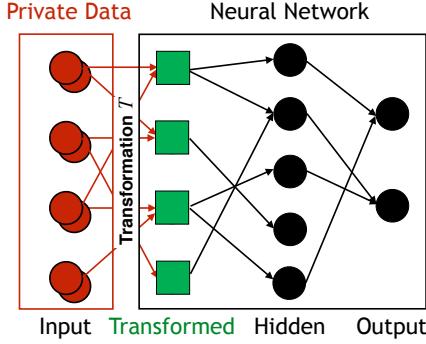
Herein, our target is to protect the data privacy of the participants. Concretely, the numerical values of training and prediction data of participants would not be recovered. The labels of the dataset are permuted as well.

In reality, the risks of disclosure are also in the trained models. *Inversion attacks* can recover input data in “white box” [15] (by analyzing internal model parameters) and in “black box” [14] (by repeatedly querying the models to get the oracles). Considering these attacks, we assume that the adversaries have free access to internal model parameters to meet the “*worst-case*” principle.

Moreover, the system follows the *curious-but-honest* model (also called the *semi-honest* model), which is also adopted by most related work. That is, the server and all the participants will honestly follow the prescriptive steps, but try to learn others’ secret data as much as possible.

## 3. CHALLENGES AND SOLUTIONS

The basic idea of *TransNet* is to add a “transformed” layer to the neural network, as shown in Fig. 3. A transformation  $T(\cdot)$  is performed on dataset  $X$ . Intuitively, the data from different participants should be transformed in a consistent way. Otherwise, the same examples can be transformed into two different examples. There are three main challenges to develop such a basic concept. The first challenge is to support all kinds of partition types. We propose three schemes called *TransNet-V* (§ 3.2), *TransNet-H* (§ 3.3) and *TransNet-A* (§ 3.4) to deal with this challenge, and common operations of them are listed in § 3.1. The second challenge is the privacy protection of the transformed layer. We address



**Figure 3:** The basic idea of TransNet is to add an indirect “transformed” layer (green nodes) to the neural network.

this challenge by developing a method named *Indeterminacy Constitution* to ensure that the transformation equations have infinite solutions. The degree to which privacy is preserved is analyzed in § 3.5. Briefly speaking, it depends on the secret values including private transformation matrix, noise and the secret translation vector. The last challenge is the effectiveness of the training of neural network over the transformed layer. We analyze the Lipschitz continuity of the transformation to explain why the neural network can train over the transformed layer in § 3.6.

### 3.1 Common Operations of TransNet

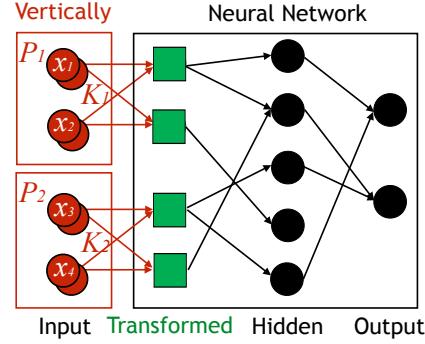
TransNet’s common operations include two aspects: determine the DPT and permute the labels, which are conducted before and during the training phase, respectively.

**Determine the DPT.** Suppose the whole dataset is aggregated from  $p$  participants, and each participant  $P_i$  possesses a partial dataset  $X_i$ . The whole dataset has  $m$  examples and  $n$  attributes, and is denoted as an  $m \times n$  matrix  $X$ . Each participant first sends the server a table to indicate which positions of the dataset it has. Whether it has the labels of an example is also indicated. The IDs of the data examples are hashed. Then the server can determine the DPT and requires the dataset from each participant in disjoint parts. Hence the form is simplified. If the participants have different data examples and the attributes are complete, then  $X$  is horizontally partitioned and can be denoted as  $(X_1^T, X_2^T, \dots, X_p^T)^T$ , where  $X_i$  owned by  $P_i$  has  $m_i$  examples,  $n$  attributes and  $\sum_{i=1}^p m_i = m$ . If the participants have different parts of attributes of the data examples, then  $X$  is vertically partitioned and can be denoted as  $(X_1, X_2, \dots, X_p)$ , where  $X_i$  owned by  $P_i$  has  $m$  examples,  $n_i$  attributes and  $\sum_{i=1}^p n_i = n$ . More universally,  $X$  can be arbitrarily partitioned and denoted as  $\sum_{i=1}^p X_i$ , where the data positions that  $P_i$  does not have are set to 0 in  $X_i$ .

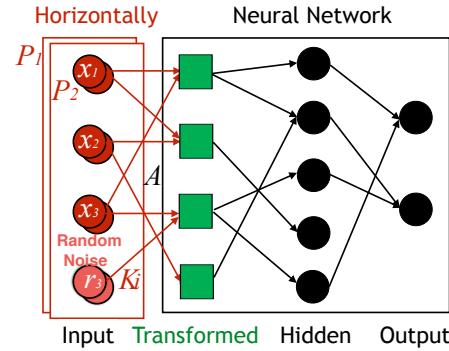
**Permute the labels.** To prevent the server from knowing the dataset’s labels, the participants negotiate a common permutation and perform it to their labels. Then they send the scrambled labels to the server. If some classes of the labels have obvious statistical characteristics, those classes are divided into multiple groups. For example, 26 letters can be divided into 40 kinds of classes and are permuted, and thus the server cannot relate the labels to the data.

### 3.2 Vertically Partitioned Dataset

Suppose a music website wants to precisely recommend genres to a newly registered user, it can aggregate the same



**Figure 4:** Structure of TransNet-V.



**Figure 5:** Structure of TransNet-H.

user’s purchase records of a bookstore website and the viewing records of a video website. The two other websites may also want to do something similar or deliver proper advertisements. This is an application scenario of training over a vertically partitioned dataset.

As mentioned before, the dataset should be transformed in a consistent way. In this situation, each participant, who has different parts of attributes in the dataset, makes  $P_i$  possible to use different private transformation matrices  $K_i$  (size  $n_i \times n_i$ ). To guarantee no information loss, we require  $K_i$  to be full rank. The transformation of  $X_i$  is  $X'_i = X_i K_i$ , and the whole dataset is transformed as

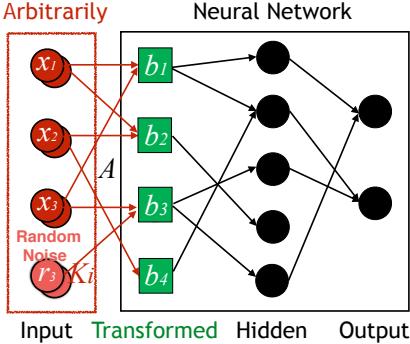
$$X' = XK = (X_1 K_1, X_2 K_2, \dots, X_p K_p), \quad (1)$$

where  $K = \text{diag}(K_1, K_2, \dots, K_p)$ . Fig. 4 shows a simple illustration of two participants’ situation of TransNet-V.

### 3.3 Horizontally Partitioned Dataset

If several hospitals in different cities want to research certain disease based on their patients’ data, it is a typical situation of learning over a horizontally partitioned dataset.

In this case, we use a public transformation matrix  $A$  to ensure the consistency. To do this while preserving privacy, we introduce an approach named *Indeterminacy Constitution*. We can regard the original dataset  $X$  as source and the transformed dataset  $X'$  as destination, and the transformation can be considered as a transmission channel. We have two basic methods to conduct the *Indeterminacy Constitution*. One is to use a privacy-preserving principal component analysis (PCA) scheme to reduce the dimension of attributes of the original dataset, and this is a noiseless lossy channel. The other is to add noise as additional attributes to



**Figure 6:** Structure of TransNet-A.

the original dataset, which is a noisy lossless channel. Algebraically, the former is to decrease the number of equations in the indeterminate system, while the latter is to increase the number of unknowns.

In TransNet-H, we adopt the latter approach, as shown in Fig. 5. The participants first negotiate a public matrix  $A$  (size  $n \times n$ ). Suppose  $d$  dimensional noise  $\mathbf{r}$  is added to the data example  $\mathbf{x}$ . Each  $P_i$  concatenates a noise matrix  $R_i$  (size  $m_i \times d$ ) and its partitioned dataset  $X_i$  (size  $m_i \times n$ ), and then generates a private matrix  $K_i$  (size  $d \times n$ ). Let  $A_i = (A^T, K_i^T)^T$ .  $P_i$ 's dataset  $X_i$  is transformed as

$$X'_i = (X_i, R_i)A_i = X_iA + R_iK_i. \quad (2)$$

### 3.4 Arbitrarily Partitioned Dataset

Sometimes datasets are more fragmented. For example, in a shopping mall, the consumption data held by many merchants make up a complete dataset. These data may contain some relevance between different commodities. In this scenario, datasets are called arbitrarily partitioned.

For an arbitrarily partitioned dataset, if we similarly adopt Eq. (2) to transform it as  $X = \sum_{i=1}^p (X_i, R_i)A_i$ , where  $P_i$  generates  $m \times d$  noise matrix  $R_i$  and  $d \times n$  private matrix  $K_i$ , then the noise errors are accumulated, which results in a decrease in test accuracy after training. To overcome this problem, we can let  $P_i$  generate pairwise disjoint parts of noise matrix  $R_i$ , and the positions of  $R_i$  that are not generated by  $P_i$  are set to 0, i.e., each element of  $R = \sum_{i=1}^p R_i$  is not a compound random variable. However, by doing this, it enforces that  $P_i$  does not generate any noise for some data examples, and the transformation reduces to  $\mathbf{x}' = \mathbf{x}A$ , which is solvable. This situation will happen when the number of participants is greater than the dimension of noise. For example, if  $n+1$  participants own a dataset of  $n$  attributes together, then there must exist one participant that does not generate any noise.

*Indeterminacy Constitution* for TransNet-A introduces a “secret translation vector”  $\mathbf{b} = (b_1, b_2, \dots, b_n)$ . We denote the row extension matrix of  $\mathbf{b}$  as  $B = (\mathbf{b}^T, \mathbf{b}^{T^T}, \dots, \mathbf{b}^T)^T$ . As shown in Fig. 6, the transformation goes as

$$X'_i = (X, R)A_i + B \quad (3)$$

where  $A_i = (A^T, K_i^T)^T$  is similar to TransNet-H.

In TransNet-A, each participant generates a vector  $\mathbf{b}_i = (b_{i1}, b_{i2}, \dots, b_{in})$ , and the server also generates a vector  $\mathbf{b}_0$ , In § 4.3 we will design a method called *Convert Secure Sum* to sum  $\mathbf{b} = \sum_{i=0}^p \mathbf{b}_i$  in Eq. (3), but  $\mathbf{b}$  and all  $\mathbf{b}_i$  are secure.

The secret translation vector  $\mathbf{b}$  can also be generated pairwise disjointly, yet it requires  $d > n/(p-1)$  dimensional noise to meet indeterminate demand. The analysis is as follows: Suppose  $P_i$  has  $n_i$  attributes, it generates  $d_i$  ( $\sum_{i=1}^p d_i = d$ ) dimensional noise and  $t_i$  ( $\sum_{i=1}^p t_i = n$ ) dimensional secrets  $b_{s+1}, b_{s+2}, \dots, b_{s+t_i}$ , labeled in order rather than by position. Then  $P_i$  has  $(n - n_i)$  unknown attributes,  $(d - d_i)$  unknown noise and  $(n - t_i)$  unknown secrets. It should satisfy the indeterminacy that the number of unknowns should be greater than the number of equations, which is:

$$(n - n_i) + (d - d_i) + (n - t_i) > n, \forall i.$$

This means for all  $i$ ,  $P_i$  should satisfy  $t_i + d_i < (n - n_i) + d$ , for all possible  $n_i$ . In the worst case, we let  $n - n_i = 0$ ,  $\forall i$ , and add up the inequalities:  $\sum_{i=1}^p t_i < (p-1)d$ . With  $\sum_{i=1}^p t_i = n$ , we know that it needs  $d > \frac{n}{p-1}$  dimensional noise to meet the requirement of indeterminacy.

## 3.5 Privacy Analysis

### 3.5.1 TransNet-V

It is reasonable to regard the transformation  $X' = XK$  as an encryption scheme, where  $K$  is the encryption key,  $X$  is the plaintext, and  $X'$  is the ciphertext.

**THEOREM 1.** *The TransNet-V encryption  $X' = XK$  is of information-theoretic security.*

**PROOF.** For any invertible square matrix  $K_r$ , let  $X_r = X'K_r^{-1}$ , then  $Pr[X_r = X] = Pr[K_r = K]$ , so taking any plaintext amounts to taking any invertible square matrix.

Now for any two different invertible square matrices  $K_0, K_1$ , we let  $X_0 = X'K_0^{-1}, X_1 = X'K_1^{-1}$  as two plaintexts. The two plaintexts have the same probability  $Pr[X_0] = Pr[X_1]$ . Besides,  $Pr[K_0] = Pr[K_1]$ . It follows that

$$Pr[C = X'|M = X_0] = Pr[C = X'|M = X_1],$$

where  $C$  represents the ciphertext and  $M$  represents the plaintext. Thus Theorem 1 holds.  $\square$

There still exist vulnerabilities. For continuous values, the attacker can narrow the range of the possible  $K$  with the background of the values’ ranges. Taking 1 dimensional data for example, suppose the minimum value is  $a$  and the maximum value is  $b$ , then their transformed values will be  $ak$  and  $bk$ , respectively. Therefore, the value of  $k$  can be determined. Similarly, for categorical attributes  $\{1, 2, \dots, n\}$ , the adversary gets  $\{k, 2k, \dots, nk\}$ , which is easy to unravel. For training in these cases, it is essential to have additional patches by adding noise like that in TransNet-H, which will be presented in § 3.5.2. Note that with noise added, the accuracy will also be affected. Besides, for 0/1 encoded attributes,  $|r| \geq \frac{1}{2}|k|$  is required for noise  $r$  to ensure the indistinguishability, which will lead to more accuracy losses in some tasks.

### 3.5.2 TransNet-H

In the transformation  $\mathbf{x}' = (\mathbf{x}, \mathbf{r})A_u = (\mathbf{x}, \mathbf{r}) \begin{pmatrix} A \\ K_u \end{pmatrix}$ , if  $K_u$  is disclosed, the adversary can recover the original data example using the pseudo-inverse  $\text{pinv}(A_u)$ :

$$\begin{aligned} (\mathbf{x}_0, \mathbf{r}_0) &= \mathbf{x}' \cdot \text{pinv}(A_u) \\ &= \mathbf{x}' \cdot (A_u^T A_u)^{-1} A_u^T \\ &= (\mathbf{x}, \mathbf{r})A_u \cdot (A_u^T A_u)^{-1} A_u^T. \end{aligned}$$

Note  $P = A_u(A_u^T A_u)^{-1} A_u^T$  is actually the projection matrix, then  $(\mathbf{x}_0, \mathbf{r}_0)$  is the projection of  $(\mathbf{x}, \mathbf{r})$  in the subspace of the column space of  $A_u$ , and the distance

$$\|(\mathbf{x}, \mathbf{r}) - (\mathbf{x}_0, \mathbf{r}_0)\| = \sum_{i=1}^n (x_i - x_i^{(0)})^2 + \sum_{i=1}^d (r_i - r_i^{(0)})^2$$

gets the minima, where  $\mathbf{x}_0 = (x_1^{(0)}, \dots, x_n^{(0)})$  and  $\mathbf{r}_0 = (r_1^{(0)}, \dots, r_d^{(0)})$ . The recovered example  $(\mathbf{x}_0, \mathbf{r}_0)$  is the best approximation to  $(\mathbf{x}, \mathbf{r})$ .

We denote  $A$  as  $(a_{ij})_{1 \leq i, j \leq n}$  and  $K_u$  as  $(k_{ij}^{(u)})_{1 \leq i \leq d, 1 \leq j \leq n}$ . The transformation of example  $\mathbf{x} = (x_1, \dots, x_n)$  goes as

$$x_1 a_{1i} + \dots + x_n a_{ni} + r_1 k_{1i}^{(u)} + \dots + r_d k_{di}^{(u)} = x'_i, \quad \forall i = 1, \dots, n,$$

and it gets the transformed example  $\mathbf{x}' = (x'_1, \dots, x'_n)$ . Note  $a_{ij}$  is public and  $\mathbf{x}'$  can be observed, while  $r_i$  and  $k_{ij}^{(u)}$  are private, the adversary actually has a set of equations with errors  $\mathbf{e} = (e_1, \dots, e_n)$ , where  $e_i = r_1 k_{1i}^{(u)} + \dots + r_d k_{di}^{(u)}$ :

$$\mathbf{x}A + \mathbf{e} = \mathbf{x}'. \quad (4)$$

The adversary can ignore  $\mathbf{e}$  and compute  $\mathbf{x}_0 = \mathbf{x}'A^{-1}$  to recover the data. It is approximated to the recovery  $(\mathbf{x}_0, \mathbf{r}_0) = \mathbf{x}' \cdot \text{pinv}(A_u)$ , the closest method we can find. It is obvious that the bigger the noise is, the harder the adversary can recover the data. Suppose  $r_i$  and  $k_{ij}^{(u)}$  are normal distributions:

$$r_i \sim \mathcal{N}(0, \sigma_r), k_{ij}^{(u)} \sim \mathcal{N}(0, \sigma_k).$$

Then  $e_i$  is the inner product of vectors  $\mathbf{r}$  and  $\mathbf{k}_i^{(u)}$  (column vector of  $K_u$ ). We have the characteristic function of  $e_i$  (refer to [32]):

$$\Psi_e(\omega) = \begin{cases} (1 + \sigma_r^2 \sigma_k^2 \omega^2)^{-d_1}, & d = 2d_1 \\ (1 + \sigma_r^2 \sigma_k^2 \omega^2)^{-d_1 - 1/2}, & d = 2d_1 + 1. \end{cases}$$

According to the relationship of the derivative of characteristic function and the moment of the random variable, the variance of  $e_i$  can be computed by:

$$D(e_i) = d\sigma_r^2 \sigma_k^2. \quad (5)$$

Therefore, the degree of privacy increases with  $d$ ,  $\sigma_r$  and  $\sigma_k$ .

Next we try to recover the transformed MNIST dataset by  $\mathbf{x}'A^{-1}$  with parameters in Eq. (5). Table 1 lists the results. Row (a) is the original MNIST examples. If the errors get too small, the scheme cannot provide enough privacy for the image dataset (Rows (b), (c) and (d)). We find that the noise's dimension  $d$  brings more stable privacy than  $\sigma_r$  or  $\sigma_k$  with the same  $D(e_i)$ . (Row (d) versus Row (e); Row (c) versus Row (f)). Moreover, we can learn from Row (e) that  $d = 16$  is already hard for the adversary to recover, and in the experiments we find that for TransNet-H,  $d = 300$  only leads to 3% decrease of test accuracy in MNIST dataset. Our subsequent experiments listed in Table 3 (§ 5.6) use the settings of  $d = 100$ ,  $\sigma_r = 1$ ,  $\sigma_k = 1/4$ , and the recovery result is shown in Row (g).

The above analysis is under the real number situation. Now we consider the case that the elements of  $\mathbf{x}$  are integral or rational numbers. Let  $A_q = A \bmod q$  and  $\mathbf{y} = \mathbf{x}' \bmod q$ ,

**Table 1:** The MNIST recovery examples of TransNet-H.

(a)	$\sigma_r$	$\sigma_k$	$d$	
(b)	1	1	1	
(c)	1	2	1	
(d)	2	2	1	
(e)	1	1	16	
(f)	$\frac{1}{2}$	1	16	
(g)	1	$\frac{1}{4}$	100	

we notice the similarity between Eq. (4) and the “Learning with Errors” (LWE) problem, i.e.,

$$\mathbf{x}A_q \approx \mathbf{y} \pmod{q} \quad (6)$$

Denote the question defined by Eq. (4) as the integral-LWE problem, and we have the following theorem:

**THEOREM 2.** *The problems of integral-LWE and LWE can be reduced to each other.*

**PROOF.** If  $\mathbf{x}_0$  is a solution to Eq. (4), then  $\mathbf{x}_0 \bmod q$  is the solution to Eq. (6).

For the other direction, if  $\mathbf{x}_0$  is a solution to Eq. (6), let  $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{t} \cdot q$  and substitute  $\mathbf{x}_1$  into Eq. (4) to compute  $\mathbf{t} = 1/q \cdot (\mathbf{x}' - \mathbf{x}_0 A) \cdot A^{-1}$ . Hence the solution to Eq. (4), i.e.,  $\mathbf{x}_1$ , can be computed.  $\square$

Therefore, the difficulty of solving TransNet-H is equal to solving LWE problem, which is believed to be difficult.

### 3.5.3 TransNet-A

The privacy analysis of the transformation  $\mathbf{x}' = (\mathbf{x}, \mathbf{r})A_i + \mathbf{b}$  is same to that of TransNet-H. The “secret translation vector”  $\mathbf{b}$  can be regarded as one more random variable, and it increases the derivative of the comprehensive error. Thus TransNet-A introduces more randomness than TransNet-H with the same  $\sigma_r$ ,  $\sigma_k$  and  $d$ , and it losses more model accuracy than TransNet-H, which will be shown in experiments.

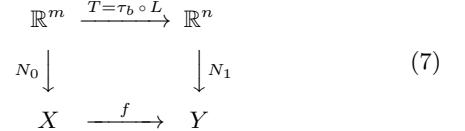
## 3.6 Why TransNet Works

The above transformations can be denoted as a compound function  $T$  of a linear transformation  $L$  and a translation  $\tau_b$ , and the noise-adding function  $\eta : \mathbb{R}^n \rightarrow \mathbb{R}^m$ , i.e.,:

$$T(\mathbf{x}) = \tau_b \circ L(\mathbf{x}) = L(\mathbf{x}) + \mathbf{b}, \quad \eta(\mathbf{x}) = (\mathbf{x}, \mathbf{r}).$$

We know  $T$  is Lipschitz continuity while  $\eta$  is not. The distance between two close vectors  $\mathbf{x}_1, \mathbf{x}_2$  can be greater than a fixed value after noise is added. In § 5 we can also see that the impact of the transformation matrix  $K$  or  $A$  on the performance of the neural network is small, while the impact of the noise  $R$  is notable.

Denote the neural network training over the original dataset as a function  $N_0$ , and the same model training over the transformed layer as function  $N_1$ , then we have the following relation diagram (7).



Suppose there are two points  $p_1, p_2 \in \mathbb{R}^m$ , the neural network function  $N_0$  can effectively classify the original data, whose meaning is that if  $p_1, p_2$  are the same kind of data, then the Euclidean metric  $\|N_0(p_1), N_0(p_2)\|_2$  is small. For example, if we use a 10 dimensional vector to represent the classification of the handwritten numbers, then for the same classification  $p_1, p_2$ ,  $N_0(p_1), N_0(p_2)$  are dropped near the unit vector of the same axis. Here we explain the data relationship of the prediction results of the neural networks trained over the original and the transformed datasets. In the relation diagram (7) we denote function  $f$  as a map from classification space  $X$  to  $Y$ . If we can prove that  $f$  is Lipschitz continuity, then when the prediction results of  $X$  are close, the prediction results of  $Y$  given by  $N_1$  are restricted by the Lipschitz constant, which are also close, i.e., the training results of  $N_1$  are also reasonable.

Define a metric  $d_X$  of space  $X$ , for two arbitrary examples  $x_1 = N_0(p_1) \in X, x_2 = N_0(p_2) \in X$ ,

$$d_X(x_1, x_2) = \|N_0(p_1), N_0(p_2)\|_2.$$

Similarly, define a metric  $d_Y$  of space  $Y$  as

$$d_Y(y_1, y_2) = \|N_1(T(p_1)), N_1(T(p_2))\|_2.$$

In the previous work of Szegedy *et al.* [33] and Ruan *et al.* [29], the authors show the Lipschitz continuity of various deep learning models. And here we have:

**THEOREM 3.** *If  $N_0$  is bi-Lipschitz, and  $N_1$  is Lipschitz, then  $f$  is also Lipschitz.*

**PROOF.** According the Lipschitz continuity of  $N_1$  and  $T$ ,

$$\begin{aligned} d_Y(y_1, y_2) &= d_E(N_1(T(p_1)), N_1(T(p_2))) \\ &\leq C_1 \cdot d_E(T(p_1), T(p_2)) \\ &\leq C_T C_1 \cdot d_E(p_1, p_2). \end{aligned}$$

Because of the bi-Lipschitz continuity of  $N_0$ ,  $\exists C_0 \geq 1$ ,

$$\begin{aligned} \frac{1}{C_0} \cdot d_E(p_1, p_2) &\leq d_X(x_1, x_2) = d_E(N_0(p_1), N_0(p_2)) \\ &\leq C_0 \cdot d_E(p_1, p_2), \end{aligned}$$

hence

$$d_Y(y_1, y_2) \leq C_T C_1 \cdot d_E(p_1, p_2) \leq C_T C_1 C_0 \cdot d_X(x_1, x_2),$$

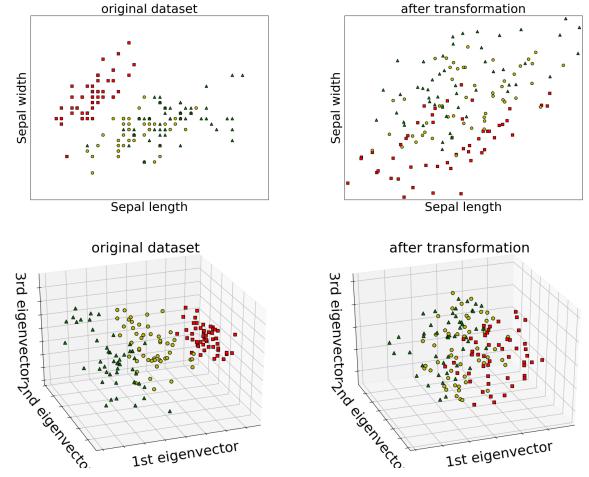
i.e.,  $C_f \leq C_T C_1 C_0$ .  $\square$

To visually present this intuition, we show the iris<sup>1</sup> dataset before and after the transformation  $X' = (X, R)A_i + B$ . Fig. 7 is a visual display of the relation diagram (7). We show the points of first two features and apply three components PCA to the dataset, where three colors of the points denote the three kinds of iris respectively. The data are garbled after the transformation (as shown in the 2 dimensional chart), but they still have obvious boundaries if we apply PCA (as shown in the 3 dimensional chart), which means that PCA over the transformed layer works.

## 4. THE TRANSNET SCHEMES

In this section, we provide the algorithm of TransNet schemes in detail. Before training, each participant first sends the server a 0/1 table to indicate which positions of

<sup>1</sup>Iris dataset contains 150 examples, consisting of 4 attributes (sepal/petal length, sepal/petal width) and label of three kinds.



**Figure 7:** Principal component analysis over the Iris dataset before and after the transformation of TransNet-A.

the dataset it has (including labels). A common hash function is applied to ID attribute. Then the server can determine the type of data partition and sends a 0/1 table back to each participant to instruct it to manipulate the disjoint part of the dataset. Suppose there are  $m$  data examples and  $n$  attributes. The set of the labels  $y \in G$  are scrambled with a common negotiated permutation  $S : G \rightarrow S(G)$  as mentioned in § 3.1. Then the labels  $Y$  of the dataset is in  $(S(G))^m$ .

Suppose  $p$  participants  $P_i$  ( $i = 1, 2, \dots, p$ ) train TransNet. In training phase,  $P_i$  has partitioned dataset  $X_i$ . The private parameters  $R_i$  and  $K_i$  are generated randomly. In prediction phase, an example  $z$  is going to be predicted. If the dataset is vertically or arbitrarily partitioned,  $P_i$  owns part of  $z$ , written as  $z_i$ . Denote the training server as  $S$ , and the trained neural network is represented as a function  $N$ .

### 4.1 TransNet-V

---

#### Algorithm 1: Training of TransNet-V.

---

```

Data: Dataset  $X = (X_1, \dots, X_p)$ ; labels  $Y \in (S(G))^m$ ;
Result:  $S$  trains function  $N$  over  $(X', Y)$ ;
1 for  $i \leftarrow 1$  to  $p$  do
2    $P_i$  generates  $n_i \times n_i$  private matrix  $K_i$ ;
3    $P_i$  computes  $X'_i \leftarrow X_i K_i$ ;
4    $P_i$  sends  $X'_i$  to  $S$ ;
5 end
6  $S$  pools  $X' \leftarrow (X'_1, X'_2, \dots, X'_p)$ ;
7  $S$  trains over  $(X', Y)$  and gets function  $N$ ;
8 Suppose  $N(\mathbf{x}) = f_l(f_{l-1}(\dots f_0(\mathbf{x}W_0 + b_0) \dots) W_l + b_l)$ ;
9 if distribute parameters then
10  |  $W' \leftarrow W_0$ ;
11  |  $S$  divides  $W' = (W_1'^T, W_2'^T, \dots, W_p'^T)^T$  by  $n_i$ ;
12  | for  $i \leftarrow 1$  to  $p$  do
13  |   |  $S$  sends  $W'_i$  to  $P_i$ ;
14  | end
15 end

```

---

The training phase of TransNet-V is described in Algorithm 1. TransNet-V has the option to distribute its model parameters to the participants. If the scheme does not distribute the training parameters, Algorithm 2 is used to predict, otherwise Algorithm 3 is used. Algorithm 3 can avoid the duplicate use of private matrices  $K_i$ .

---

**Algorithm 2:** Prediction of TransNet-V.

---

**Data:** The prediction example  $\mathbf{z}$ ;  
the same  $K_i$  and  $N$  as in Algorithm 1;  
**Result:**  $S$  predicts for  $\mathbf{z}$ ;

- 1 **for**  $i \leftarrow 1$  **to**  $p$  **do**
- 2   |  $P_i$  computes  $\mathbf{z}'_i \leftarrow \mathbf{z}_i K_i$ ;
- 3 **end**
- 4  $S$  pools  $\mathbf{z}' \leftarrow (\mathbf{z}'_1, \mathbf{z}'_2, \dots, \mathbf{z}'_p)$ ;
- 5  $S$  sends  $\mathbf{z}'$  to  $S$ ;
- 6  $S$  computes  $y \leftarrow N(\mathbf{z}')$ ;
- 7  $S$  encrypts  $y$  and sends to all  $P_i$ ;
- 8  $P_i$  decrypts and reverses the permutation  $S^{-1}(y)$ ;

---

**Algorithm 3:** Prediction of TransNet-V.

---

**Data:** The prediction example  $\mathbf{z}$ ;  
the same  $W'_i$  and  $N$  as in Algorithm 1;  
**Result:**  $S$  predicts for  $\mathbf{z}$ ;

- 1 **for**  $i \leftarrow 1$  **to**  $p$  **do**
- 2   |  $P_i$  computes  $W_i \leftarrow K_i W'_i$ ;
- 3   |  $P_i$  computes  $\mathbf{a}_i \leftarrow \mathbf{z}_i W_i$ ;
- 4 **end**
- 5  $\mathbf{a} \leftarrow \sum_{i=1}^p \mathbf{a}_i$  by secure multi-party sum;
- 6 Participants send  $\mathbf{a}$  to  $S$ ;
- 7 Suppose  $N(\mathbf{z}') = f_l(f_{l-1}(\dots f_0(\mathbf{z}' W' + b_0) \dots) W_l + b_l)$ ;
- 8  $S$  computes  $y \leftarrow f_l(f_{l-1}(\dots f_0(\mathbf{a} + b_0) \dots) W_l + b_l)$ ;
- 9  $S$  encrypts  $y$  and sends to all  $P_i$ ;
- 10  $P_i$  decrypts and reverses the permutation  $S^{-1}(y)$ ;

---

**THEOREM 4.** Algorithm 3 and Algorithm 2 have the same prediction result  $y$ .

**PROOF.** In Algorithm 3 we have  $W = K_i W'_i$ , so

$$\begin{aligned} X'_i W'_i &= (X_i K_i) W'_i = X_i (K_i W'_i) = X_i W_i, \\ X' W' &= (X'_1 X'_2 \cdots X'_p) \cdot (W'^T_1 W'^T_2 \cdots W'^T_p)^T \\ &= (X'_1 W'_1 X'_2 W'_2 \cdots X'_p W'_p) \\ &= (X_1 W_1 X_2 W_2 \cdots X_p W_p) = XW \end{aligned}$$

In the same way, we have  $\mathbf{z}' W' = \mathbf{z} W = \mathbf{a}$ , thus in Algorithm 3  $y = N(\mathbf{z}')$ , which is the same result as in Algorithm 2. We call  $W$  the “de-transformed” weights of  $W'$  in Algorithm 1.  $\square$

In Algorithm 2, the participants just submit their data, and asynchronously receive the prediction. While in Algorithm 3, the participants need to work together with others to do the secure multi-party sum.

## 4.2 TransNet-H

Algorithm 4 describes the training phase of TransNet-H. The server can simply distribute the model to all participants. Then in the prediction phase of TransNet-H (Algorithm 5), the participants can do predictions themselves. Moreover, the training of TransNet-H can be simultaneously carried out with the pooling of the dataset, while TransNet-V and TransNet-A both need to wait until all the datasets are collected.

## 4.3 TransNet-A

Suppose that encryption function is  $c = Enc(m)$  and the corresponding decryption function is  $m = Dec(c)$ . The training phase and the prediction phase of TransNet-A are described in Algorithm 6 and Algorithm 7, respectively.

---

**Algorithm 4:** Training of TransNet-H.

---

**Data:**  $X = (X_1^T, X_2^T, \dots, X_p^T)^T$ ;  $Y \in (S(G))^m$ ;  
**Result:**  $S$  trains function  $N$  over  $(X', Y)$ ;

- 1 The participants consult  $n \times n$  public matrix  $A$ ;
- 2 **for**  $i \leftarrow 1$  **to**  $p$  **do**
- 3   |  $P_i$  generates  $m_i \times d$  noise matrix  $R_i$ ;
- 4   |  $P_i$  generates  $d \times n$  private matrix  $K_i$ ;
- 5   |  $P_i$  computes  $X'_i \leftarrow X_i A + R_i K_i$ ;
- 6   |  $P_i$  sends  $X'_i$  to  $S$ ;
- 7 **end**
- 8  $S$  pools  $X' \leftarrow (X'_1{}^T, X'_2{}^T, \dots, X'_p{}^T)^T$ ;
- 9  $S$  trains over  $(X', Y)$  and gets function  $N$ ;
- 10  $S$  distributes  $N$  to all  $P_i$ ;

---



---

**Algorithm 5:** Prediction of TransNet-H.

---

**Data:** The prediction example  $\mathbf{z}$ ;  
The same  $A$  and  $N$  as in Algorithm 4;  
**Result:**  $P_i$  predicts for  $\mathbf{z}$ ;

- 1  $P_i$  generates  $1 \times d$  noise matrix  $r$ ;
- 2  $P_i$  generates  $d \times n$  private matrix  $K_i$ ;
- 3  $P_i$  computes  $\mathbf{z}' \leftarrow \mathbf{z} A + r K_i$ ;
- 4  $P_i$  computes  $y \leftarrow N(\mathbf{z}')$ ;
- 5  $P_i$  reverses the permutation  $S^{-1}(y)$ ;

---

There is a key point in Algorithm 6 that no one should know the value of  $\mathbf{b} = \sum_{i=0}^p \mathbf{b}_i$ , otherwise the indeterminacy of the Eq. (3) is broken. Herein we develop an original technique called *Convert Secure Sum*. For the server, the terms  $\mathbf{x}' + \mathbf{b}$  and  $\mathbf{b}_0$  are known, the terms  $\mathbf{x}'$  and  $\sum_{i=1}^p \mathbf{b}_i$  are unknown. Thus  $\mathbf{b}$  cannot be computed from  $(\mathbf{x}' + \mathbf{b}) - \mathbf{x}'$ , nor from  $\mathbf{b}_0 + \sum_{i=1}^p \mathbf{b}_i$ . And for participants  $P_i$ , the terms  $\mathbf{b}_i$  and  $\mathbf{x}'$  are known,  $\sum_{k=0}^i \mathbf{b}_k + \mathbf{c}$  can be known in the situation that one data example is completely owned by  $P_i$ . Note that the term  $\mathbf{c}$  is unknown. Therefore,  $P_i$  ( $i = 1, 2, \dots, p$ ) cannot extract  $\mathbf{b}$  from  $(\mathbf{b} + \mathbf{c}) - \mathbf{c}$ .

## 5 EVALUATION

### 5.1 Experimental Overview

We implement the interface between our transformed layer and the neural network based on TensorFlow and PyTorch frameworks. The training uses cross-entropy loss and Adam optimizer, and the learning rate is 0.01. The batch size is 100 when using SGD method. We also use the notation  $[-l_1 - l_2 -]$  to denote the number of nodes of hidden layers when using MLP, the number of filters when using CNN, and the sequence length of hidden layers when using RNN.

For simplicity, we use notation  $ks$  to denote the range  $[-ks, ks]$  of the elements of  $K_i$  and  $A$  when they are randomly generated with uniform distribution, notation  $rs$  to denote that range  $[-rs, rs]$  of  $R_i$ , and notation  $bs$  to denote that range  $[-bs, bs]$  of secret translation vector  $\mathbf{b}$ . We also use notation  $ds$  to denote the dimension of noise. In addition, if these variables are randomly generated with normal distribution, then the same notation  $ks, rs, bs$  denote the standard deviation  $\sigma$ . The default value of  $ks, rs, bs$  is 1 with uniform distribution, the default value of  $ds$  is 100. Table 2 lists the datasets and the neural network we apply.

We denote the *independent* learning as that each participant trains over its split dataset, and denote the *baseline* as

---

**Algorithm 6:** Training of TransNet-A.

---

**Data:**  $X = \sum_{i=1}^p X_i$ ;  $Y \in (S(G))^m$ ;  
**Result:**  $S$  trains function  $N$  over  $(X', Y)$ ;

- 1 The participants consult  $n \times n$  public matrix  $A$ ;
- 2 **for**  $i \leftarrow 1$  to  $p$  **do**
- 3   |  $P_i$  generates  $m \times d$  noise matrix  $R_i$ ;
- 4   |  $P_i$  generates  $d \times n$  private matrix  $K_i$ ;
- 5   |  $P_i$  computes  $X''_i \leftarrow X_i A + R_i K_i$ ;
- 6   |  $P_i$  generates  $1 \times n$  secret translation vector  $b_i$ ;
- 7   |  $B_i \leftarrow (b_i^T, b_i^T, \dots, b_i^T)^T$ ;
- 8 **end**
- 9  $S$  generates  $1 \times n$  secret translation vector  $b_0$ ;
- 10  $S$  generates a random vector  $c$ ,  $C \leftarrow (c^T, c^T, \dots, c^T)^T$ ;
- 11  $S$  encrypts  $e_0 \leftarrow Enc(b_0 + c)$ ;
- 12  $E_0 \leftarrow (E_0^T, E_0^T, \dots, E_0^T)^T$ ;
- 13  $S$  sends  $e_0$  to  $P_1$ ;
- 14 **for**  $i \leftarrow 1$  to  $p$  **do**
- 15   |  $P_i$  decrypts  $M_{i-1} \leftarrow Dec(E_{i-1})$ ;
- 16   |  $P_i$  computes  $M_i \leftarrow M_{i-1} + X''_i + B_i$ ;
- 17   |  $P_i$  encrypts  $E_i \leftarrow Enc(M_i)$ ;
- 18   | **if**  $i \neq p$  **then**
- 19     | |  $P_i$  sends  $E_i$  to  $P_{i+1}$ ;
- 20   | **else**
- 21     | |  $P_i$  sends  $E_i$  to  $S$ ;
- 22 **end**
- 23 **end**
- 24  $S$  decrypts  $M_p \leftarrow Dec(E_p)$ ;
- 25  $S$  computes  $X' = M_p - C$ ;
- 26  $S$  trains over  $(X', Y)$  and gets function  $N$ ;

---

**Algorithm 7:** Prediction of TransNet-A.

---

**Data:** The prediction example  $z$ ;  
the same  $A, b_i$  and  $N$  as in Algorithm 6;  
**Result:**  $S$  predicts for  $z$ ;

- 1 **for**  $i \leftarrow 1$  to  $p$  **do**
- 2   |  $P_i$  generates  $r_i$  and private matrix  $K_i$ ;
- 3   |  $P_i$  computes  $z''_i \leftarrow z_i A + r_i K_i$ ;
- 4 **end**
- 5  $S$  generates a random vector  $c$ ;
- 6  $S$  encrypts  $e_0 \leftarrow Enc(b_0 + c)$ ;
- 7  $S$  sends  $e_0$  to  $P_1$ ;
- 8 **for**  $i \leftarrow 1$  to  $p$  **do**
- 9   |  $P_i$  decrypts  $m_{i-1} \leftarrow Dec(e_0)$ ;
- 10   |  $P_i$  computes  $m_i \leftarrow m_{i-1} + z''_i + b_i$ ;
- 11   |  $P_i$  encrypts  $e_i \leftarrow Enc(m_i)$ ;
- 12   | **if**  $i \neq q$  **then**
- 13     | |  $P_i$  sends  $e_i$  to  $P_{i+1}$ ;
- 14   | **else**
- 15     | |  $P_i$  sends  $e_i$  to  $S$ ;
- 16 **end**
- 17 **end**
- 18  $S$  decrypts  $m_p \leftarrow Dec(e_p)$ ;
- 19  $S$  computes  $z' = m_p - c$ ;
- 20  $S$  computes  $y \leftarrow N(z')$ ;
- 21  $S$  encrypts  $y$  and sends to  $P_i$ ;
- 22  $P_i$  decrypts and reverses the permutation  $S^{-1}(y)$ ;

---

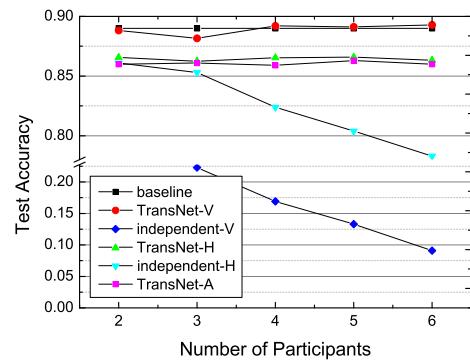
**Table 2:** The datasets and the applied neural network.

Dataset	Train/Test Size	epochs	Neural Network
LETTER	12,000/8,000	5,000	MLP: [-40-]
			MLP: [-400-300-]
MNIST	55,000/10,000	$\approx 7.3$	CNN: [-16-32-] RNN: [-40-]
			MLP: [-400-300-]
FASHION	60,000/10,000	5	CNN: [-16-32-] RNN: [-40-]
			CNN: [-16-32-]
SVHN	73,257/26,032	5	CNN: [-16-32-]

the accuracy obtained by training over the original aggregated dataset. Moreover, all the test accuracy data we show in the figures are the average results of 10 training outcomes.

Our experiments first verify the utility of TransNet over LETTER<sup>2</sup> and MNIST<sup>3</sup> datasets in § 5.2. Second, we adjust  $ks, rs, ds, bs$  to see their impacts on the performance of MLP over MNIST dataset in § 5.3. Third, we apply CNN and RNN over FASHION<sup>4</sup> dataset in § 5.4. § 5.5 checks the impact of operator norm  $\lambda$  of the matrix. Finally, we test our scheme over larger dataset SVHN<sup>5</sup> to further confirm the effectiveness of TransNet. § 5.6 summarizes the performance of TransNet and compares TransNet to other schemes.

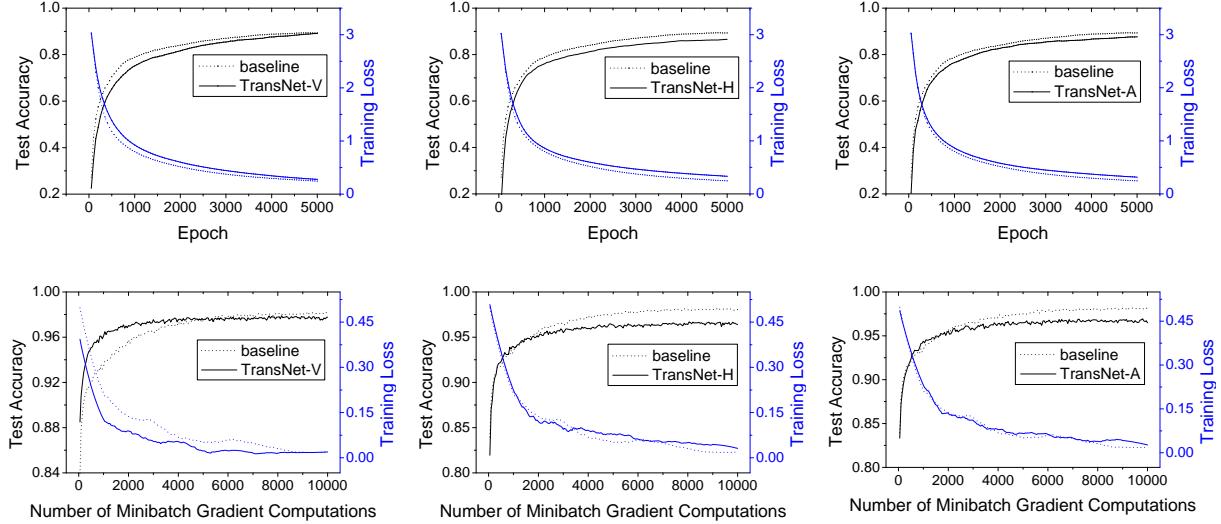
## 5.2 Utility Verification


**Figure 8:** Comparison of test accuracy of TransNet, independent learning and baseline over LETTER dataset using MLP, where  $ds = 2$ .

The accuracy of the privacy-preserving collaborative neural network should be better than the model trained over each participant's own dataset. The experimental results over LETTER dataset using MLP are shown in Fig. 8, and the situations over other datasets are similar. Over horizontally partitioned dataset, independent accuracy declines from 86.1% to 78.3% on average, while TransNet-H gets about 86.7% accuracy. This is because in the experiment, more participants share the whole dataset means each of them will have fewer examples, and this is opposite to the common situation that more participants means a larger dataset. Over vertically partitioned dataset, the independent accuracy declines from 56.4% to 9.1% which is totally useless, while TransNet-V makes the accuracy up to about 89.1%. Over arbitrarily partitioned dataset, TransNet-A gets about 86.5% accuracy, while independent learning cannot be done.

In addition, the accuracy is not affected by the number of participants, as shown in Fig. 8. It keeps basically steady only with little fluctuation because of the randomness. In the experiments over larger datasets, the number of participants is increased from 10 to 100, and it also does not influence the accuracy. This is useful since it makes the schemes easy to extend to participants with a single data example.

<sup>2</sup><http://archive.ics.uci.edu/ml/datasets/Letter+Recognition>
<sup>3</sup><http://yann.lecun.com/exdb/mnist>
<sup>4</sup><https://github.com/zalandoresearch/fashion-mnist>
<sup>5</sup><http://ufldl.stanford.edu/housenumbers/>



**Figure 9:** Training processes of TransNet vs. *baseline*, where black and blue curves represent accuracy and loss, respectively. The top three diagrams are trained over LETTER dataset, and the bottom three diagrams are trained over MNIST dataset.

To intuitively acquaint the training over the transformed layer, we draw charts of test accuracy and training loss with the number of minibatch gradient computations. The diagrams in Fig. 9 show the performances of TransNet over LETTER and MNIST datasets in the top and bottom row respectively. The curves of *baseline* and TransNet are close. TransNet gets slightly lower accuracy than the *baseline* with noise added. Because of SGD, the accuracy presents a fluctuating increase over MNIST dataset. It is noticeable that TransNet-V trains faster than the *baseline* over MNIST dataset. The possible reason is the transformation matrix  $K$  in TransNet-V is a block diagonal matrix  $\text{diag}(K_1, K_2, \dots, K_p)$ .

### 5.3 Impact of Variables Using MLP

To investigate the impact of  $ks$ ,  $rs$ ,  $bs$ , and  $ds$ , the idea of “control variates” is adopted. We first fix other variables and adjust  $ks$  in TransNet-V, for it eliminates the interference of noise  $R$  and secret translation vector  $b$ . Then we choose a relatively good  $ks$  to test the impact of  $rs$  and  $ds$  in TransNet-H. Finally the impact of  $bs$  is evaluated in TransNet-A. In the beginning,  $ks$ ,  $rs$ , and  $bs$  are set to 1 with uniform distribution and  $ds = 100$ .

We perform experiments over MNIST dataset using MLP to explore the impact of variables in TransNet. The values of MNIST dataset are normalized to range  $[0,1]$ . The transformed layer is not normalized and is input into  $[-400-300]$  MLP for 4,000 steps of minibatch gradient computations. The *baseline* gets 98% accuracy. The *independent* learning can only get enough accuracy when the dataset is horizontally partitioned and is influenced by the number of participants, so the corresponding curves are not drawn in the following diagrams. Experimental results are exhibited in charts of Fig. 10.

**Impact of transformation matrix.** Fig. 10a shows the impact of  $ks$ . Let  $ks = 2^{-4}, 2^{-3}, \dots, 2^5$  respectively in TransNet-V, we can learn that too big or too small  $ks$  slightly lowers the performance. It basically declines at a log speed (because x-axis is in log scale) when  $ks \geq 1$ . It gets the highest accuracy when  $\sigma = ks = 1/8$  with normal

distribution, and when  $ks = 1/4$  with uniform distribution. The curve of uniform distribution performs slightly better than that of normal distribution. Thus in the following experiments, we fix  $ks = 1/4$  with uniform distribution.

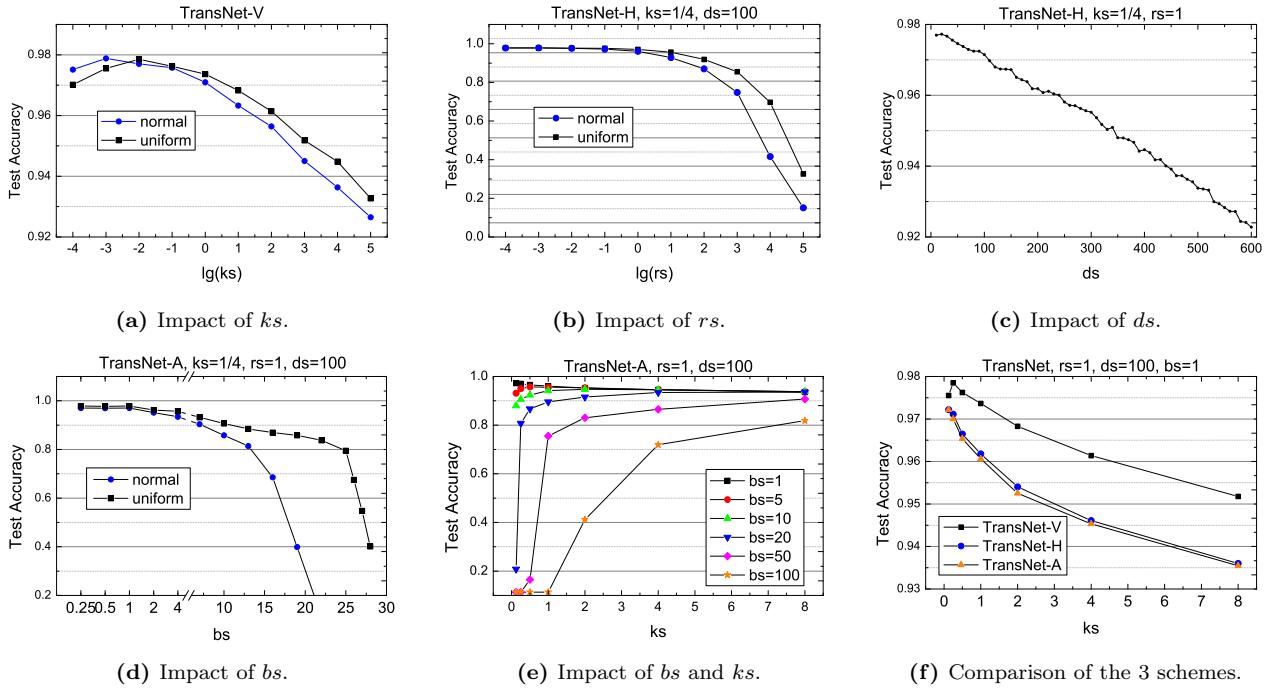
**Impact of noise.** Unlike the linear transformation defined by matrix, the noise-adding function is not a Lipschitz continuity. However, adding more dimensions of noise means higher privacy level, so it is an important factor in the schemes. In TransNet-H we firstly fix  $ds = 100$  and change  $rs$ , then use the appropriate  $rs$  to see how  $ds$  affects the performance.

Fig. 10b shows the impact of  $rs$ , where  $rs = 2^{-4}, 2^{-3}, \dots, 2^5$  respectively. We can learn that a larger  $rs$  lowers the accuracy apparently. The curve of uniform gets higher accuracy than that of normal. The performance is steady when  $rs \leq 1$ . Since the values of MNIST dataset are in range  $[0, 1]$ , we suggest  $rs = 1$  with uniform distribution to make the noise similar to the real data examples.

Fig. 10c shows the impact of  $ds$  in the setting of  $rs = 1$ . We add the dimensions of noise from 10 to 600 with step 10. The curve declines in linear speed. Notice it is a very good result that adding 300 dimensional noise loses only about 3% accuracy from about 98% to about 95%, and adding 600 dimensional noise loses only about 6% accuracy to about 92%. Therefore, if  $rs$  is appropriate, the impact of the  $ds$  is limited. From this experimental result we can learn that adding more dimensions of noise can increase the privacy level with low accuracy cost.

**Impact of secret translation vector.** From Fig. 10d we can learn that the increase of  $b$  causes a rapid decline of accuracy. The normal one performs worse than the uniform one. It is even close to random guesses (1/10 accuracy) when the distribution is  $\mathcal{N}(\mu = 0, \sigma = 22)$ . Although translation is Lipschitz, too big  $bs$  eliminates the differences between two different data points. The reason is similar to that normalization can improve the performance of machine learning algorithms.

Since  $b$  seems to have big influence on performance, we draw curves of different  $bs$  with the change of  $ks$  (with uni-



**Figure 10:** Impact of TransNet’s variables on the performance over MNIST dataset using MLP. (*baseline*: 98.0%).

form distribution). As shown in Fig. 10e, if we use bigger  $bs$ , then using bigger  $ks$  will help to increase the accuracy, but the accuracy cannot exceed the curves of smaller  $bs$ .

**Comparison of the three schemes.** We have explored the impact of all the variables in TransNet. Next, we compare the three schemes.  $ks$  is the common variable in the three schemes. Therefore, we set  $ks$  as the x-axis to see how things make different when introducing the noise  $R$  in TransNet-H, and when introducing the secret translation vector  $b$  in TransNet-A.

Fig. 10f shows the comparison result, in which all the variables are randomly generated with uniform distribution. Consistent with the performance in Fig. 10a, the increase of  $ks$  lowers the test accuracy of the three schemes. TransNet-H gets lower accuracy than TransNet-V since there are 100 dimensional noise added, and TransNet-A gets very close but slightly lower accuracy than TransNet-H since it only introduces 1 dimension  $b$  and  $bs = 1$  is small. The curve of TransNet-H will get closer to TransNet-V when  $ds$  declines according to Fig. 10c, and vice versa.

#### 5.4 Impact of Variables Using CNN and RNN

We also perform experiments over FASHION dataset using CNN and RNN, and the corresponding schemes are called TransCNN and TransRNN respectively.

We find that  $ks$  does not obviously influence the accuracy when using CNN. The accuracy stays steady while  $ks$  changes in very large scale intervals. TransCNN-V keeps about 86.2% accuracy while  $ks \in [2^{-21}, 2^{501}]$ , TransCNN-H keeps about 83.3% accuracy while  $ks \in [2^{-20}, 2^{500}]$ , and TransCNN-A keeps about 83.1% while  $ks \in [2^{-7}, 2^{502}]$ . The CNN *baseline*’s accuracy is 90.3%.

Fig. 11 shows the impact of the variables in TransCNN and TransRNN over FASHION dataset, where RNN curves are all lower than the corresponding CNN curves. The impact of  $rs$ ,  $ds$  and  $bs$  in TransCNN and TransRNN are simi-

lar to TransMLP. The impact of  $ks$  in TransRNN is also similar to TransMLP. The RNN *baseline*’s accuracy is 86.0%.

In Fig. 11c, the dimension of noise is up to 1400, which exceeds the number of data attributes (784). It also shows a limited impact when using CNN and RNN, which is similar to the situation that uses MLP.

#### 5.5 Impact of Transformation Matrix Norm

We gave an analysis of why the neural network can train over the transformed layer in § 3.6. The Lipschitz constant of a linear transformation is the norm of the transformation matrix, i.e., the largest singular value  $\lambda$  of the matrix. Thus in this subsection we explore the impact of the matrix norm on the performance of the training model. To make the results more obvious, we generate the matrix that all the singular values are the same.

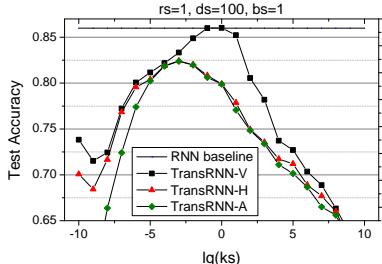
As shown in Fig. 12a, over MNIST dataset, the impact of the matrix norm  $\lambda$  on the test accuracy is similar to  $ks$  but more notable. When using CNN, the impact is the least. When using RNN, the impact is more notable than MLP. As shown in Fig. 12b, over FASHION dataset, the impact of the matrix norm  $\lambda$  is also similar.

According to the analysis of Lipschitz continuity in § 3.6, smaller matrix norm will improve the accuracy, but in the experiment we did not see it. This may be because the numerical calculation of small numbers will bring errors, as the impact of  $ks$  shows. Moreover, the neural network function itself has a big Lipschitz constant. Notice that all Lipschitz functions are uniformly continuous, and thus it works.

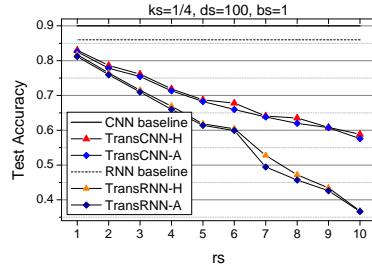
#### 5.6 Experimental Summary and Comparison

##### 5.6.1 Experimental Summary

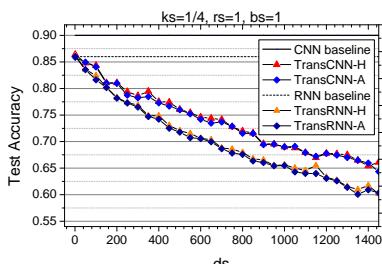
We test TransNet schemes over different datasets using different deep learning models. The results are summarized in Table 3. The experiments demonstrate that TransNet



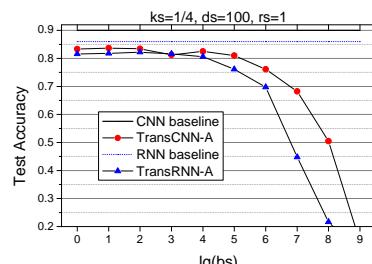
(a) Impact of  $ks$ .



(b) Impact of  $rs$ .



(c) Impact of  $ds$ .



(d) Impact of  $bs$ .

**Figure 11:** Impact of TransNet’s variables on the performance over FASHION dataset using CNN (*baseline*: 90.3%) and RNN (*baseline*: 86.0%).

**Table 3:** Experimental accuracy results with  $ks = 1/4, rs = 1, ds = 100, bs = 1$ . V, H and A are the abbr. of TransNet-V, TransNet-H and TransNet-A respectively.

Dataset	Network	Baseline	V	H	A
LETTER	MLP	0.892	0.891	0.867	0.865
	MLP	0.980	0.978	0.971	0.970
	CNN	0.993	0.971	0.933	0.931
MNIST	RNN	0.974	0.973	0.923	0.921
	MLP	0.753	0.731	0.723	0.717
	CNN	0.903	0.862	0.833	0.831
FASHION	RNN	0.860	0.860	0.822	0.820
	MLP	0.861	0.732	0.706	0.687
	CNN	0.861	0.732	0.706	0.687

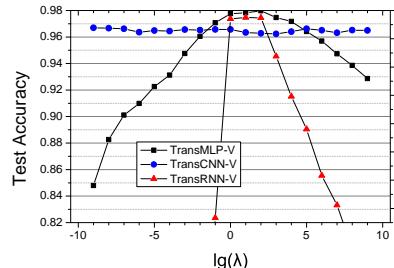
schemes are effective. We also explore the impact of variables in TransNet on the performance of neural network, and find that too big or too small  $ks$  slightly lowers the accuracy; that too big  $rs$  strongly lowers the accuracy; that the impact of  $ds$  on the accuracy is limited; that too big  $bs$  strongly lowers the accuracy; and that too big or too small matrix norm  $\lambda$  slightly lowers the accuracy.

Overall, the transformation matrix has small influence since the linear transformation defined by the matrix is Lipschitz continuity; the noise have big influence since the processing of adding noise is not continuity; however, the secret translation vector  $b$  has big influence.

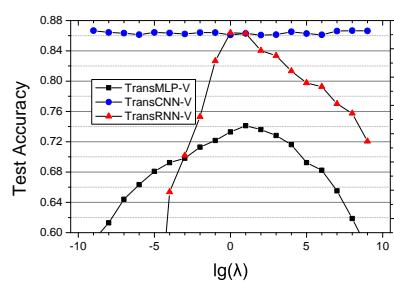
We can see from Table 3 that there are little accuracy losses using MLP, a bit of accuracy losses using RNN, and relatively large accuracy losses using CNN. Over LETTER and MNIST datasets, TransMLP-V almost loses no accuracy, TransRNN-V also loses little accuracy.

### 5.6.2 Comparison to Other Schemes

We compare TransNet to other privacy-preserving neural network schemes in Table 4. Our TransNet has lower computation and communication complexity than homomorphic encryption (HE) based schemes [40], and has the advantage



(a) Experiments over MNIST dataset.



(b) Experiments over FASHION dataset.

**Figure 12:** Impact of the transformation matrix norm on the performance.

**Table 4:** Comparison to state-of-the-art schemes, where V, H and A are the abbr. of vertically, horizontally and arbitrarily partitioned dataset respectively.

Scheme	Data Partition	Affected by No. Parties?	Complexity
Yuan2014 [40]	V H A	No	High
Shokri2015 [31]	H	Yes	Low
McMahan2017 [25]	H	Yes	Low
Papernot2016 [27]	H	Yes	Low
TransNet	V H A	No	Low

of supporting arbitrarily partitioned dataset compared with the stochastic gradient descent (SGD) [25, 31] or differential privacy (DP) based schemes [27]. Besides, the accuracy of TransNet is not influenced by the number of participants.

Below is the complexity comparison. The methods based on HE have the highest computation and communication complexity because the spending of the encryption and decryption is very high, and the calculation results need to be transmitted frequently. Schemes based on SGD have the same computation complexity as the original neural network, but need time to wait for the uploads of SGD parameters for every training step. They also have low communication complexity because they only need to upload the SGD parameters. Papernot *et al.*’s approach [27] is of high efficiency because the models are training independently and finally aggregated using DP with low computation complexity and very low communication complexity. The computation complexity of TransNet is the same as the original neural network since it trains over the transformed layer. And the communication complexity of TransNet is  $O(m)$ , where  $m$  is the number of data examples, which is also small.

Furthermore, we compare our scheme with HE based work over arbitrarily partitioned datasets. Yuan2014’s scheme [40] is a representative of *directly training the neural network* based on HE to date, while recent HE based work have diffe-

**Table 5:** Comparison to Yuan2014’s scheme [40], where  $ks = rs = bs = 1$ .

Dataset	Sample	Architecture	Test Accuracy			Training Time	
			Baseline	TransNet-A <sup>1</sup>	TransNet-A <sup>2</sup>	Yuan2014	TransNet-A
Iris	150	4-5-3	98.17%	93.17%	84.17%	83.79%	4.8s~4.9s
Diabetes	768	8-12-2	69.57%	66.72%	65.97%	63.99%	5.1s~6.8s
kr-vs-kp	3196	36-15-2	93.93%	90.45%	87.17%	86.19%	7.6s~9.8s

<sup>1</sup>ds = 1/2 number of attributes

<sup>2</sup>ds = number of attributes

rent aims that convert pre-trained neural networks to cryptographic versions [7, 17] or provide secure CNN prediction as a service [9, 20]. Therefore, [40] is still the state-of-the-art work in this field, and we adopt it as the comparison scheme. We directly extract the results from [40], and employ the same network architecture to train TransNet-A. We run more epochs than [40] does, because with small epochs we cannot reach adequate accuracy. Yuan2014’s scheme is implemented via C language and executed on Amazon EC2 cloud, including 10 nodes with 8-core 2.93-GHz Intel Xeon CPU and 8-GB memory, while TransNet is implemented using Python TensorFlow and launched with 2-core 2.70-GHz Intel Pentium G630 CPU and 4-GB memory locally to simulate multi-party. Obviously, our running specification is lower than that of [40], and thus theoretically, TransNet would perform better, or at least not worse, should it be conducted using the same running specification of [40].

The comparison is in terms of runtime performance and test accuracy over iris, diabetes<sup>6</sup> and kr-vp-kp<sup>7</sup> datasets of UCI machine learning repository. The numbers of attributes  $n$  of the datasets are 4, 8 and 36, respectively. We use two settings ( $ds = n/2$  and  $ds = n$ ) to train TransNet-A for comprehensive consideration. The results are shown in Table 5. Note that it takes more time with a bigger  $ds$ , and thus the training time of TransNet-A varies. As can be seen from the table, for test accuracy, TransNet-A gets a small lift. While for training time, TransNet-A performs much better than Yuan2014. In general, TransNet-A surpasses Yuan2014’s scheme in all aspects.

## 6. RELATED WORK

The research of the privacy-preserving data mining or machine learning has been started early, and there are surveys [2, 10, 30] of tools and methods to approach such schemes. We use a taxonomy according to the technique that the schemes adopt to classify the state-of-the-art work, including SMC, HE, DP, and SGD based schemes.

SMC computes a boolean function on multiple participants’ inputs securely by converting it into a garbled circuit. The circuit complexity is proportional to the computation complexity of the function. It has been used for learning decision tree [24], linear regression and classification [11], k-means clustering [19], association rule [34], Naive Bayes classifier [35], and  $k$ -Nearest Neighbor [37] etc. However, the extremely high computation and communication complexity of SMC usually make it far from practical when the algorithm or the data size becomes larger.

HE can do computations over ciphertexts. It is firstly applied to linear algorithms, including linear regression [26, 38], linear classifier [6], etc. By converting the activation function into its polynomial approximation, neural network

<sup>6</sup><http://archive.ics.uci.edu/ml/datasets/Diabetes>

<sup>7</sup><http://archive.ics.uci.edu/ml/datasets/Chess+%28King-Rook+vs.+King-Pawn%29>

[40] can use HE to approach its privacy-preserving version, followed by encrypted CNN [17] which is based on Bos *et al.*’s HE [5]. Li *et al.* [23], Phong *et al.* [28], and Wang *et al.* [36] also designed privacy-preserving neural networks based on HE. Similar to SMC based approaches, schemes based on HE also have the limitation of low computational efficiency.

DP has become the most popular technique in the privacy topic in recent years. It has been applied to boosting [13], linear and logistic regression [41], principal component analysis [8], and risk minimization [4]. Papernot *et al.*’s work [27] proposed a *PATE* scheme that a “student” classifier is trained by Private Aggregation of Teacher Ensembles using DP technique. The schemes based on DP possess the advantage of low computation and communication complexity, but have the insufficiency that they are only suited to horizontally partitioned datasets and they have to be applied to database rather than individual data examples.

Furthermore, SGD has also been exploited for approaching privacy-preserving neural networks, for it is an essential component in the algorithms. Such work include distributed selective SGD [31], differential privacy SGD [1], and SGD of the “federated averaging” [25].

Although there exist many privacy-preserving machine learning schemes, the methods toward the neural network are still insufficient. Different from previous work, our TransNet uses indeterminate system to constitute the transformed layer to protect to privacy of the original dataset. TransNet has the advantage of lower computation and communication complexity than the SMC and HE based schemes, and the training speed is close to the DP based scheme [27] but faster than the SGD based schemes [1, 25, 31]. Furthermore, it has the superiority of supporting arbitrarily partitioned dataset and being not influenced by the number of participants.

## 7. CONCLUSIONS

In this paper, we proposed TransNet, a new method to train privacy-preserving collaborative neural network over transformed layer. In TransNet, the participants transform their vertically, horizontally or arbitrarily partitioned datasets using irreversible transformation defined by linear indeterminate equations, and upload the transformed data to the server. The server then can train a practical model without knowing any private information over the transformed layer. We use Lipschitz continuity to analyze why the neural network can train over the transformed layer. TransNet supports all kinds of partitioned datasets compared to most of previous schemes that support single partitioned dataset only. It is independent of the neural network structure and not affected by the number of participants. It takes no extra computational burden and has no special security requirement on the training server. Finally, extensive experiments showed the effectiveness and the above advantages of our TransNet.

## 8. REFERENCES

- [1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.
- [2] C. C. Aggarwal and S. Y. Philip. *Privacy-preserving data mining: models and algorithms*. Springer Science & Business Media, 2008.
- [3] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831, 2015.
- [4] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.
- [5] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *IMA International Conference on Cryptography and Coding*, pages 45–64. Springer, 2013.
- [6] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.
- [7] Bourne, Florian and Minelli, Michele and Minihold, Matthias and Paillier, Pascal. Fast homomorphic evaluation of deep discretized neural networks. In *Annual International Cryptology Conference*, pages 483–512. Springer, 2018.
- [8] K. Chaudhuri, A. D. Sarwate, and K. Sinha. A near-optimal algorithm for differentially-private principal components. *The Journal of Machine Learning Research*, 14(1):2905–2943, 2013.
- [9] Chen, Hao and Dai, Wei and Kim, Miran and Song, Yongsoo. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 395–412, 2019.
- [10] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *ACM Sigkdd Explorations Newsletter*, 4(2):28–34, 2002.
- [11] W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 222–233. SIAM, 2004.
- [12] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.
- [13] C. Dwork, G. N. Rothblum, and S. P. Vadhan. Boosting and differential privacy. pages 51–60, 2010.
- [14] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.
- [15] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 17–32, 2014.
- [16] C. Gentry et al. Fully homomorphic encryption using ideal lattices. In *Stoc*, volume 9, pages 169–178, 2009.
- [17] 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*, pages 201–210, 2016.
- [18] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
- [19] G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 593–599. ACM, 2005.
- [20] Juvekar, Chiraag and Vaikuntanathan, Vinod and Chandrakasan, Anantha. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1651–1669, 2018.
- [21] A. Kannan, K. Kurach, S. Ravi, T. Kaufmann, A. Tomkins, B. Miklos, G. Corrado, L. Lukacs, M. Ganea, P. Young, et al. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 955–964. ACM, 2016.
- [22] I. Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.
- [23] P. Li, J. Li, Z. Huang, T. Li, C. Gao, S. Yiu, and K. Chen. Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems*, 74:76–85, 2017.
- [24] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Annual International Cryptology Conference*, pages 36–54. Springer, 2000.
- [25] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.
- [26] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy*, pages 334–348. IEEE, 2013.
- [27] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.
- [28] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.
- [29] W. Ruan, X. Huang, and M. Kwiatkowska. Reachability analysis of deep neural networks with

- provable guarantees. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2651–2659. AAAI Press, 2018.
- [30] K. Saranya, K. Premalatha, and S. Rajasekar. A survey on privacy preserving data mining. In *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, pages 1740–1744. IEEE, 2015.
- [31] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321. ACM, 2015.
- [32] M. K. Simon. *Probability distributions involving Gaussian random variables: A handbook for engineers and scienti*. 2006.
- [33] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [34] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644. ACM, 2002.
- [35] J. Vaidya, M. Kantarcıoglu, and C. Clifton. Privacy-preserving naive bayes classification. *The VLDB Journal*, 17(4):879–898, 2008.
- [36] Q. Wang, M. Du, X. Chen, Y. Chen, P. Zhou, X. Chen, and X. Huang. Privacy-preserving collaborative model learning: The case of word vector training. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2381–2393, 2018.
- [37] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis. Secure knn computation on encrypted databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 139–152. ACM, 2009.
- [38] D. Wu and J. Haven. Using homomorphic encryption for large scale statistical analysis, 2012.
- [39] A. C.-C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.
- [40] J. Yuan and S. Yu. Privacy preserving back-propagation neural network learning made practical with cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):212–221, 2014.
- [41] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Functional mechanism: regression analysis under differential privacy. *very large data bases*, 5(11):1364–1375, 2012.