Secure Forward Aggregation for Vertical Federated Neural Networks

Shuowei Cai[1] , **Di Chai**[1,2] , **Liu Yang**[1,2] , **Junxue Zhang**[1,2] , **Yilun Jin**[1] , **Leye Wang**[3] , **Kun Guo**[4] and **Kai Chen**[1]

[1] iSING Lab, The Hong Kong University of Science and Technology
[2]Clustar [3]Peking University
[4]Fuzhou University
scaiak@cse.ust.hk, dchai@cse.ust.hk, lyangau@cse.ust.hk, jzhangcs@cse.ust.hk, yilun.jin@connect.ust.hk, leyewang@pku.edu.cn, gukn@fzu.edu.cn, kaichen@cse.ust.hk

Abstract

Vertical federated learning (VFL) is attracting much attention because it enables cross-silo data cooperation in a privacy-preserving manner. While most research works in VFL focus on linear and tree models, deep models (*e.g.*, neural networks) are not well studied in VFL. In this paper, we focus on SplitNN, a well-known neural network framework in VFL, and identify a trade-off between data security and model performance in SplitNN. Briefly, SplitNN trains the model by exchanging gradients and transformed data. On the one hand, SplitNN suffers from the loss of model performance since multiply parties jointly train the model using transformed data instead of raw data, and a large amount of low-level feature information is discarded. On the other hand, a naive solution of increasing the model performance through aggregating at lower layers in SplitNN (*i.e.,* the data is less transformed and more low-level feature is preserved) makes raw data vulnerable to inference attacks. To mitigate the above trade-off, we propose a new neural network protocol in VFL called Security Forward Aggregation (SFA). It changes the way of aggregating the transformed data and adopts removable masks to protect the raw data. Experiment results show that networks with SFA achieve both data security and high model performance.

1 Introduction

Federated learning (FL) [Kairouz *et al.*, 2021] is a new paradigm of collaborative machine learning with privacy preservation, and it could be categorized into several categories according to different data partition scenarios [Yang *et al.*, 2019]. Among them, vertical federated learning (VFL) is defined as the scenario in which multiple participants hold the same entities but different features. Existing work has explored various types of VFL models, such as logistic regression [Hardy *et al.*, 2017; Zhang *et al.*, 2021] and decision trees [Cheng *et al.*, 2021a]. However, the neural network is not well studied in the VFL scenario. Specifically, there is little analysis of data security and model performance in complex models like neural networks.

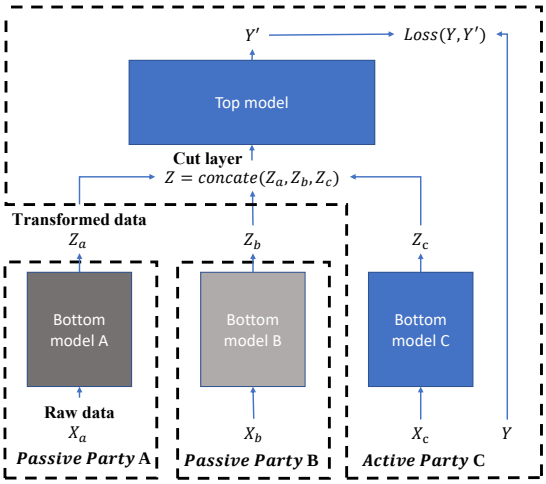


Figure 1: Split neural network in VFL

Split neural network (SplitNN) [Vepakomma *et al.*, 2018] is a framework able for neural networks in VFL. As shown in Fig.1, the whole network is partitioned into a top model and several bottom models. Each VFL participant keeps a bottom model for transforming its raw data, and the transformed data are passed to the top model at the cut layer to make the prediction in each forward process. However, exchanging the transformed data between participants result in a trade-off between data security and model performance in SplitNN.

On the one hand, using the transformed data instead of the raw data for prediction will result in a loss of model performance because the transformed data contains only a part of the information in the raw data. According to the analysis in [Mahendran and Vedaldi, 2015], a significant amount of low-level feature information is discarded while the raw data passes through the layers. This discarding information happens while the raw data passes through the bottom models of SplitNN, and it also transforms data from fine grain to coarse grain. For example, using the term "cheap baby products" to represent a classic cotton diaper from a famous brand. However, this transformation increases the difficulty of the model's prediction because the model cannot capture the interaction between the discarded information, just like

the classic "beer and diaper" relationship in data mining. A person who buys beer may be interested in diapers, but it's hard to say that a person who buys a drink will be interested in baby products. When two participants in SplitNN possess feature information like beer and diapers, low-level feature interaction loss happens, and it will decrease the model's performance.

On the other hand, the transformed data which are sent directly to the active party leaks information about the raw data. The passive party's raw data will be vulnerable to inference attacks if the transformed data contains too much information about it. As a result, it is inappropriate to increase the amount of information in the transformed data to improve the model's performance. Thus, we identify this trade-off, as the transformed data influences both the model performance and data security. It is impossible to achieve high model performance and high data security simultaneously in SplitNN.

Motivated by the above problem, we propose a method to mitigate this trade-off between data security and model performance. We consider that the direct exposure of the transformed data is improper, and the data protection by controlling the amount of information in the transformed data is hazardous. To this end, we proposed a Secure forward aggregation (SFA) protocol that can securely aggregate the bottom models' output without exposing the individual output from the bottom model. We modify the aggregation method at the cut layer and provide a removable mask to protect the passive party's transformed data and ensure raw data security. With SFA, we mitigate the trade-off and can achieve lossless performance compared to the centralized model with high security.

The main contributions of this paper are summarized as follows:

- We evaluate the trade-off between the model performance and the security of raw data in SplitNN in vertical federated learning.
- We present a Secure forward aggregation protocol to protect the participant's transformed data while being lossless. With SFA, we can mitigate this trade-off and achieve both good model performance and high data security in neural networks in VFL.

2 Motivation

In this section, we first introduce splitNN, one of the most popular frameworks of neural networks in VFL. While noting the special designs for VFL in this architecture, we also analyze the trade-off between data security and model performance.

2.1 Background: SplitNN in VFL

As shown in Fig.1, splitNN is a distributed network structure in VFL that support multiparty settings. The participants of SplitNN are categorized into the active party (participant with labels) and the passive party (participant without labels). Each passive party holds one bottom model for local data transformation. The active party holds both a bottom model and a top model and uses the top model to make predictions with the transformed data from all participants.

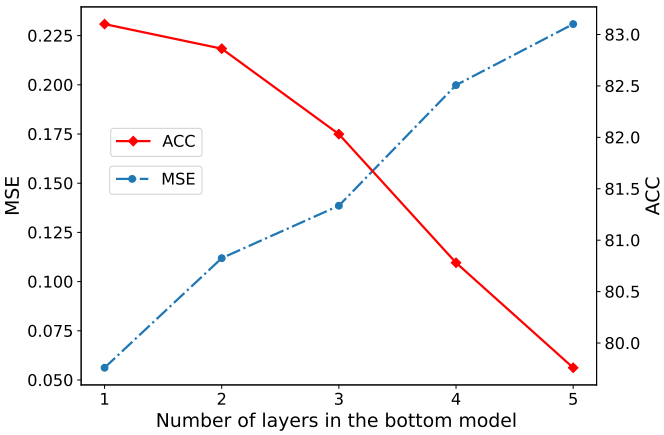


Figure 2: Performance-security trade-off in splitNN

The forward process of SplitNN consists of three steps: 1) participants use their bottom models to transform the data, 2) passive parties send the transformed data to the active party, and 3) the active party will concatenate these transformed data, applies activation functions and feed it into the top model to get the prediction results. The layer which concatenates these transformed data is also called the cut layer. In the backward process, the active party will first update the top model normally and calculate the gradients of the embeddings. Then it will send the gradients of these embeddings to their owners to update their bottom models.

2.2 Trade-Off of SplitNN

The participant's raw data security is a primary concern in vertical federated learning. In SplitNN, one substantial information leakage is the transformed data directly sent to the active party. Those transformed data do not contain all raw data information because some low-level information is lost during the forward propagation. However, there is still a correlation between the transformed and original data. When the active party of SplitNN applies inference attacks like [Luo *et al.*, 2021] to dig out the correlation, the raw data is able to be approximated.

Indeed, discarding more information from the transformed data is a way to improve data security in SplitNN. Increasing the number of layers in the bottom model is a feasible way to fulfill this. A higher bottom model will increase the complexity of the transformation and discard more low-level feature information from the transformed data, making raw data hard to reconstruct. However, the discarded low-level feature information in the transformed data is crucial to the model's performance. It contributes to the model performance in low-order feature interactions between participants. The connections between the bottom models are used to capture these low-level feature interactions, but they are missing in SplitNN.

Therefore, to further measure the relationship between data security and model performance in SplitNN, we use the generative regression network (GRN) in [Luo *et al.*, 2021] to at-

tack the SplitNN and generate approximation data to get close to the raw data. As shown in Fig.2, the approximation data of GRN is far from the raw data when the number of layers in the bottom model is high, but the model performance drop significantly(*i.e.,* 3%). The model's performance is the best when there is only one layer in the bottom model. However, the restored data is closest to the raw data in this case, and the MSE between the approximation data and the raw data is only 0.055. Model performance and raw data security become the two ends of the scale in SplitNN. As a result, we urgently need a method to mitigate this trade-off.

3 Secure Forward Aggregation

3.1 Overview

In this section, we propose a novel protocol called Secure Forward aggregation (SFA) to protect the transformed data of the passive parties. It provides removable masks to passive parties, and the masks do not introduce noise into the computation of the model. SFA protocol is used at the topmost layer for the bottom models of all participants. It securely aggregates the bottom models' output values without exposing their true values using a summation operation with masks. In the mask generation of SFA, we securely share a part of the transformed data from the active party using homomorphic encryption and send it to the passive party. The shared result will be the mask that protects the passive party's output, and homomorphic encryption ensures that the active party knows nothing about the value of the mask. Therefore, the mask effectively protects the passive party's raw data without introducing noise into the training.

Different from methods like secure aggregation [Bonawitz *et al.*, 2017], secure forward aggregation can protect the passive party's input in the aggregate output even in the two-party scene. In SFA, we use a weight mask generated by the passive party and sent to the active party under homomorphic encryption to produce masks for the transformed data. The weight mask is seen as a part of the weight of the topmost layer of the active party's bottom model and is also used to prohibit the active party from knowing the actual output of its bottom model. Therefore, the active party cannot recover the passive party's input from the aggregated result. Moreover, SFA could be applied to multi-participant scenarios, allowing all passive parties to keep masks to protect their transformed data. With SFA, we can aggregate the transformed data securely regardless of the information it contains. Therefore, the trade-off between model and security is moderate, and we can train a model that both performs well and protects raw data perfectly.

3.2 Aggregation Method

We sum the transformed data from the bottom models in SFA instead of concatenating them in SplitNN. The first reason is that the concatenate operation exposes the transformed data directly, increasing the difficulty of data protection. Moreover, the concatenate operation treats each neuron independently, but sum will not. So, the feature interaction will not be captured at the cut layer. Therefore, concatenation operation at the cut layer will indirectly increase the loss of data

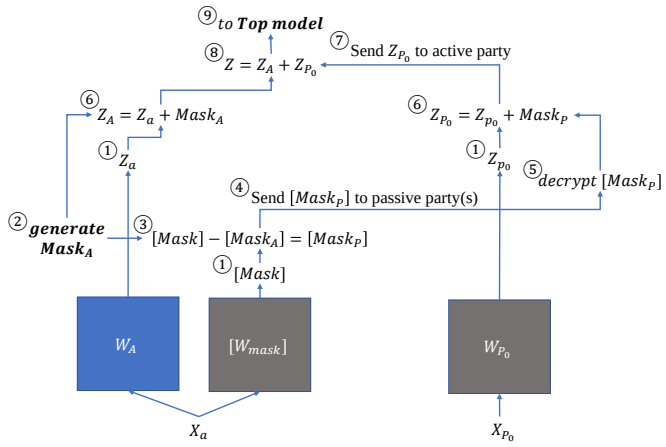


Figure 3: Secure Forward Aggregation in two party scene

information by passing the transformed data through another layer for capturing feature interaction. As a result, we change the aggregation method at the cut layer from concatenating operation to a summation operation.

3.3 Training with Weight Mask

The algorithm of the Secure Forward Aggregation protocol is shown in algorithm1. During the initialization stage, one passive party will generate a weight mask (W_{mask} in algorithm1) and send it to the active party as a part of the weights in the topmost layer of the bottom model. This weight mask will be encrypted by Paillier homomorphic encryption [Paillier, 1999] and sent to the active party. This weight mask will never be decrypted, and it will be used to generate masks for passive parties without letting the active party know.

Fig.3 shows the forward process and steps of secure forward aggregation in a two-party setting. A and P_0 are the active and passive parties; X_a and X_{p_0} here are the raw data or the values output from hidden units; W_A and W_B here are (part of) the weight of the topmost layer of the bottom model, $[W_{mask}]$ is the weight mask, and $[\cdot]$ represents homomorphic encryption. In the forward process of SFA, both parties will calculate Z_a and Z_{p_0} normally using W_A,X_a and W_{P_0},X_{P_0}. Then, the active party will calculate $[Mask]$ using the encrypted value $[W_{mask}]$. Then, it will generate a random matrix $mask_A$ and subtract it from the mask and send the remaining part $[Mask_P]$ back to the passive party P_0. Then, P_0 will decrypt the result and obtain its mask. After that, the two parties add the mask onto their transformed data, and the passive party P_0 sends its masked transformed data to the active party for aggregation. Finally, the active party will sum these transformed data to obtain the final result Z.

Secure Forward Aggregation protocol can also protect the transformed data in a multiparty setting. If there is extra passive party other than party P_0, party P_0 will continue share the masks to other passive parties, calculate $Mask_{P_0} = Mask_P - \sum_{i=1}^{n} Mask_{P_i}$ and send the new mask $Mask_{P_i}$ to passive party P_i. In this way, the active party still knows

nothing about the masks.

3.4 Removable Mask on Transformed Data

Unlike methods that follow differential privacy to generate noise to protect their intermediate result, the mask generated in the SFA protocol is a part of the original output. Therefore, the mask in SFA will not introduce noise to the aggregated value. It is because we regard the encrypted W_{mask} as a part of the weight for the active party, and the actual weight for the last layer in the bottom model of the active party should be $W_{A_{true}} = W_{mask} + W_A$. It is clear to see that the final output Z is the sum of the two party's transformed data:

$$\begin{aligned} Z &= Z_A + \sum_{i=0}^{N} Z_{P_i} = Z_a + Mask_A + \sum_{i=0}^{N}(Z_{P_i} + Mask_{P_i}) \\ &= (W_A + W_{mask})X_a + \sum_{i=0}^{N} W_{P_i} X_{P_i} \\ &= W_{A_{true}} X_a + \sum_{i=0}^{N} W_{P_i} X_{P_i} \end{aligned} \tag{1}$$

We can see that $W_{A_{true}} X_a + \sum_{i=0}^{N} W_{P_i} X_{P_i}$ are the aggregated result of all transformed data. The masks for passive parties are a part of $W_{A_{true}} X_a$. They are generated by the encrypted weight mask $[W_{mask}]$ and shared using a random matrix, and added back to the final aggregated result. In the backward process, the participants calculate the gradient normally, and all parties will add the gradients to the plaintext weight. During the whole training process, W_{mask} is kept unchanged. Therefore, we can consider it a noise initially added to the model weights. As the model is updated continuously, the impact of this noise will gradually fade away when doing gradient descends and updating the weights of the plaintext part.

3.5 Security Analysis

This subsection discusses the security of Secure Forward Aggregation in a semi-honest setting, which is the standard security assumption in federated learning. We show that the transformed data are well protected in the Secure Forward Layer, and the passive parties cannot infer the data from the active party.

Passive Party's Data Security

In the SFA protocol, the transformed data Z_{p_i} for passive party B is protected by a mask $Mask_{P_i}$. The active party will only obtain the masked result, and it cannot distinguish the mask and the transformed data from the masked result. Even though it knows that $Mask$ is a transformation of X_b, there are infinite eligible values of $Mask$. Therefore, it is insufficient to infer the exact value of mask $Mask_P$ or $Mask_{P_i}$ and to further infer the passive party's raw data.

Active Party's Data Security

The active party's transformed data is secure because they are not sent outside. Though the passive party knows that $Mask$ is a transformation of X_b, it knows nothing about the random generated $Mask_A$. Therefore, the passive party cannot infer $Mask$ to perform further inference attacks, and the active party's data security is ensured.

Algorithm 1 Secure forward aggregation

Participants Settings: Active party A, Passive party P_0, ($\{P_i | i = 1 \dots n\}$ for other passive party in multiparty scene)
Input: Batch of raw data or embedding from hidden units hold by participants of VFL: X_a,X_{p_0} (X_{p_i} for other passive parties),
Output: Aggregated result Z
Initialization

1: Active party A generates weight matrix W_A, Passive party P_0 generates weight matrix W_{P_0}. (If there are other passive parties, they generate their own weight matrix W_{p_i})
2: Party P_0 generates HE key pair $\{sk_b, pk_b\}$. Generate weight matrix W_{mask} , encrypt it using sk_b and send the encrypted result $[W_{mask}]$ to party A.

Forward process

1: All parties obtain the next batch of data (or hidden units value from the layer below)
2: Party A calculates $Z_a = W_A X_a$ and $[Mask] = [W_{mask}]X_a$, Party P_0 calculates $Z_{p_0} = W_{P_0} X_{P_0}$. (Other passive parties calculate $Z_{p_i} = W_{P_i} X_{P_i}$)
3: Party A generates random matrix $Mask_A$, calculate $[Mask_P] = [Mask] - [Mask_A]$ and send it to party P_0.
4: Party P_0 decrypt $[Mask_P]$. (If there are other passive parties, P_0 generates random matrix $Mask_{P_i}$ and send it to party P_i and calculate: $Mask_{P_0} = Mask_P - \sum_i^n Mask_{P_i}$)
5: Party A calculates $Z_A = Z_a + Mask_A$, Party P_0 calculates $Z_{P_0} = Z_{p_0} + Mask_{P_0}$ and send it to party A (other passive parties calculate $Z_{P_i} = Z_{p_i} + Mask_{P_i}$ and send it to party A)

Backward process

1: Active party send the upper gradient to each participants
2: All participants use this gradient to update their bottom model. $[W_{mask}]$ in party A is kept unchanged.

3.6 Mitigate Trade-off using SFA

We have already shown that SFA ensures the security of the transformed data. Therefore, we can keep a shallow bottom model for better performance. When there is only one fully-connected layer in the bottom model, and the aggregate method is changed from concatenation to summation, the structure of the model is the same as the centralized neural network. Therefore, the performance degradation caused by the model architecture no longer exists.

Though the weight mask also impacts training, with reasonable settings, the initialized weights of the weight mask will not significantly impact the final results of the model. We initialize the weight mask using a uniform distribution bounded by $2/\sqrt{in\_features}$ and encrypt it, then send it to the active party. The weight generation of this weight mask follows [LeCun *et al.*, 2012], and it reduces the impact of the

Datasets	Sector	News20	Amazon	FMNIST
Datasize	9619	18,846	100,000	60,000
Features	55,197	173,762	257	784
Labels	105	20	2	10

Table 1: Datasets descriptions

Datasets	Sector	News20	Amazon
Centralized	$91.28_{\pm 0.39}$	$83.63_{\pm 0.27}$	$77.46_{\pm 0.10}$
SplitNN	$86.43_{\pm 0.51}$	$79.76_{\pm 0.68}$	$72.86_{\pm 0.07}$
SFA-NN(ours)	$90.86_{\pm 0.30}$	$83.86_{\pm 0.32}$	$77.44_{\pm 0.10}$

Table 2: model performance on different datasets (ACC)

weight mask of model training and the final performance.

4 Experiment

In the experiment sections, extensive experiments are done to show how SFA can mitigate the trade-off between data security and model performance.

4.1 Experiment Setting

We use a neural network structure with six fully-connected layers to illustrate the performance of SFA on neural networks in VFL. We select SplitNN and a centralized model (all data are integrated for modeling) to compare a neural network with SFA (SFA-NN). We also fixed the number of hidden units of each layer for fair comparison and ran the experiment of model performance for ten trials to reduce randomness in training.

We fix the dropout to 0.3, batch size to 256, and apply batch normalization to train the model for 50 epochs in default. Then, we select the best learning rate from $\{1e-1, 1e-2, 1e-3, 1e-4, \dots\}$ with zero regularization coefficient for all experiments. The default participant number of VFL is set to two, and the features are partitioned equally for each participant. The bottom model height is set to 5 for SplitNN and 1 for SFA-NN in default for a fair comparison with the same level of security.

4.2 Dataset

We use four classification datasets to demonstrate the performance problem and the trade-off in SplitNN: Sector [Chang and Lin, 2011], news20 [Lang, 1995], Amazon electronic [1] and Fashion MNIST (FMNIST) [Xiao *et al.*, 2017] dataset.

We preprocessed these data to meet the requirement of the experiments of SplitNN. We used the TF-IDF algorithm to transform the news20 data into a sparse matrix for training. We use a trained Deep Interest network [Zhou *et al.*, 2018] to preprocess and transform 100,000 items in the Amazon electronic data into embeddings of 257 and treat them as data in model training. FMNIST is the dataset we demonstrate the security concerns in SplitNN, so we normalize the ranges of all feature values in it into (0, 1) as [Luo *et al.*, 2021] for better demonstration.

[1] http://jmcauley.ucsd.edu/data/amazon/

The detailed descriptions of the data set after preprocessing are shown in table 1.

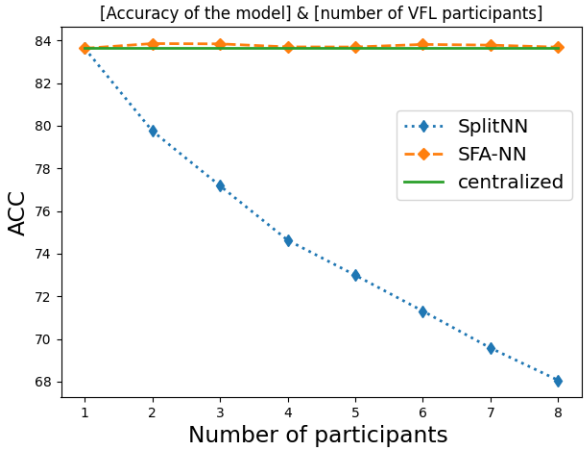


Figure 4: Model performance in multiparty settings

4.3 Performance of SFA

We experiment and compare the performance of SplitNN and SFA-NN to show that our proposed method achieves good performance in high security. We also use the Centralized model in this experiment, which refers to a model with a standard neural network structure trained by pooling all data together in a non-federal learning setting. Because the performance of federated learning models should be as close as possible to the model performance in the non-federal settings [Yang *et al.*, 2019], we use it as the target of SFA-NN to evaluate its performance. This baseline can precisely reflect the ability of SFA-NN to reduce the performance gap between neural networks in VFL and centralized neural networks.

Model Performance under Two-Party Setting

Table.2 shows the experiment result of SFA-NN in the two-party setting. The performance of SFA-NN is close to the centralized model and is significantly better than SplitNN. Although the weight mask of SFA impacts the model's training, it will not have a significant impact on the final performance of the model. The performance gap between SFA-NN and the centralized model is small, and for comparison, SplitNN's performance is low on these tasks, and SFA-NN's performance is significantly better than SplitNN.

Model Performance under Multiparty Setting

One of the reasons that SplitNN has gained popularity is that it supports multiparty training conveniently. Pessimistically, the more participants there are, the more feature partitions between participants will result in more low-level feature interaction loss. Thus, a severe model performance decrease will happen in the multiparty scenario of SplitNN. As shown in Fig.4, the model performance drops dramatically on the News20 dataset when the number of participants increases. But there is no such performance loss in SFA-NN, which shows that our method is effective in multiparty settings.

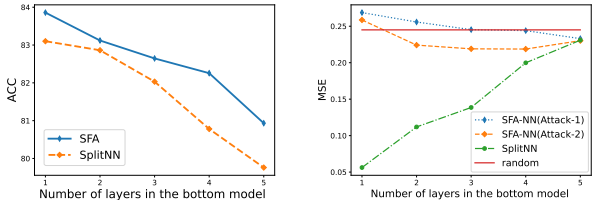
4.4 Trade-off between Security and Model Performance

In this experiment, we fix the total number of network layers and the number of neurons in each hidden layer. We then adjust the height of the cut layer and the height of the bottom model to observe the performance of the model and the security of the raw data. (the height of the top model decreases with the increase of the height of the bottom model and vice versa)

Fig.5(a) shows the model performance on the News20 dataset with the bottom model of different heights. When the height of the bottom model increase, the model performance of SplitNN drops gradually. SFA-NN also suffers from this performance loss. Though the summation operation for aggregation at the cut layer improves the model performance, the model performance of SFA-NN is only similar to SplitNN with one less layer in the bottom model. The performance problem due to the discarded information has not been fundamentally solved. This experiment shows the damage that excessive discarding of low-level information brings to the model performance. Reducing the number of layers will be an intuitive solution for those seeking higher model performance, but this brings threats to the raw data.

To evaluate the information leakage of the transformed data, we train models using FMNIST datasets to 88% accuracy with the length of the transformed data set to 256. Then, we use the generative regression network(GRN) [Luo *et al.*, 2021] to attack the bottom model and reconstruct the raw data using the test dataset. GRN is the network to generate approximation data to approximate the passive party's raw data. We take the active party's data features and the passive party's transform data as the input to train the model. We train the GRN by minimizing the Mean Square Error(MSE) between the real transformed data and the transformed result of the approximation data. Because the transformed data z_{p_0} are not known by the active party in SFA-NN, we use two masked outputs, Z_P (attack-1) and $Z_P + mask_A$ (attack-2), to substitute the transformed data, and the protections of these two attacks are $Mask_{P_0}$ and $Mask$ respectively. We also use random values between 0-1 as a baseline of the attack to evaluate the attack method's performance and show the effectiveness of SFA's protection.

Fig.5(b) shows the attack result on the transformed data, and the MSE metric indicates the distance of the attack results from the raw data. We can see that the attack is effective on the transformed data when the number of layers in the bottom model is low. When the layer number increase, the effect of the attack decrease, but the model performance gets lower. However, the attack is ineffective when SFA is used. GRN cannot achieve a good approximation of raw data even if the bottom model has only one layer. In fact, the MSE distance of the approximation data always gets larger as the training proceeds when the two attack methods act on the SFA, suggesting that the attack on transformed data with SFA is infeasible. In conclusion, SFA can protect the raw data with low bottom model layers. We can use SFA to gather the information from multiple participants at a low layer of the neural network and improve the model's performance.



(a) model performance (b) data security

Figure 5: Trade-off between data security and model performance

5 Related Work

FDML [Hu *et al.*, 2019] is another framework that supports neural networks in feature-partition settings with privacy-preservation. In FDML, each participant has an independent local model, and the final predictions of the model are obtained by summing the outputs of all local models. However, there is no direct connection between the local models of FDML, so it also suffers from a similar performance loss in SplitNN. This performance loss is also reflected in their experiments on neural networks.

Moreover, due to the design assumption that labels are shared among participants in FDML, researchers seldom focus on label security in this framework. In fact, the gradient at the top layer of FDML exposes the labels directly [Fu *et al.*, 2022]. Leakage of data labels is unacceptable for vertical federal learning. Therefore, we do not include it as a baseline in VFL.

6 Conclusion

This paper proposes a Secure Forward Aggregation protocol to mitigate the trade-off between model performance and data security in SplitNN in VFL. This protocol provides removable masks to protect the transformed data in SplitNN and aggregates the information from different parties better. Experimental results show that we achieve almost the same performance as the centralized model, and we can keep the raw data safe and resistant to attacks using SFA. We effectively mitigate the trade-off between model performance and data security in neural networks in VFL.

This work still has some limitations. On the one hand, SFA introduces partial homomorphic encryption to perform secure computations, increasing the computational effort. Nevertheless, there are ways to reduce time consumption. For example, we can reduce the multiplication calculation of the same ciphertext weight masks and plaintext data and accelerate computation using parallelism and hardware[Cheng *et al.*, 2021b]. On the other, there is a lack of hyperparameters analysis about the weight mask on model training and data security. Also, the security analysis is limited to semi-honest settings, but it is hard to ensure in a real-world scenario. We will continue to improve this work from the perspective of algorithm design and then conduct a comprehensive analysis of the effectiveness of SFA. We will enhance this work in the future to achieve good efficiency while keeping the data security and model performance in neural networks in VFL.

References

[Bonawitz *et al.*, 2017] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[Chang and Lin, 2011] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[Cheng *et al.*, 2021a] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.

[Cheng *et al.*, 2021b] Xiaodian Cheng, Wanhang Lu, Xinyang Huang, Shuihai Hu, and Kai Chen. Haflo: Gpu-based acceleration for federated logistic regression. *arXiv preprint arXiv:2107.13797*, 2021.

[Fu *et al.*, 2022] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, August 2022. USENIX Association.

[Hardy *et al.*, 2017] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[Hu *et al.*, 2019] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. Fdml: A collaborative machine learning framework for distributed features. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2232–2240, 2019.

[Kairouz *et al.*, 2021] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[Lang, 1995] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[LeCun *et al.*, 2012] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[Luo *et al.*, 2021] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. Feature inference attack on model predictions in vertical federated learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 181–192. IEEE, 2021.

[Mahendran and Vedaldi, 2015] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[Paillier, 1999] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT'99*, 1999.

[Vepakomma *et al.*, 2018] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[Xiao *et al.*, 2017] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[Zhang *et al.*, 2021] Qingsong Zhang, Bin Gu, Cheng Deng, and Heng Huang. Secure bilevel asynchronous vertical federated learning with backward updating. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10896–10904, 2021.

[Zhou *et al.*, 2018] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1059–1068, 2018.