Enhancing Security Using Random Binary Weights in Privacy-Preserving Federated Learning

Hiroto Sawada*, Shoko Imaizumi†, and Hitoshi Kiya‡
* Graduate School of Science and Engineering, Chiba University, Chiba, Japan E-mail: 24wm3202@student.gs.chiba-u.jp
† Graduate School of Engineering, Chiba University, Chiba, Japan E-mail: imaizumi@chiba-u.jp
‡ Faculty of System Design, Tokyo Metropolitan University, Tokyo, Japan E-mail: kiya@tmu.ac.jp

***Abstract*—In this paper, we propose a novel method for enhancing security in privacy-preserving federated learning under the use of the vision transformer. In federated learning, learning is performed by collecting updated information without collecting raw data from each client. However, the problem is that raw data may be inferred from updated information. To address this issue, conventional data guessing countermeasures (security enhancement methods) have a trade-off relationship between privacy protection strength and learning efficiency, and generally degrade model performance. In this paper, we propose a novel method of federated learning that does not degrade model performance and is robust against data guessing attacks on updated information. In the proposed method, each client independently prepares a sequence of binary (0 or 1) random numbers, multiplies it by the update information, and sends it to the server for model learning. In experiments, the effectiveness of the proposed method is confirmed in terms of model performance and resistance to the APRIL (Attention PRIvacy Leakage) restoration attack.**

I. Introduction

The rapid development of AI technology accelerates the growth of business and services that use deep learning. However, training a model with many parameters requires a large amount of training data, and preparation of training data takes a lot of time and effort. Additionally, we have to take into account the privacy information contained in the training data. To address this issue, federated learning, which is a distributed learning method, has attracted much attention [1].

In federated learning, multiple parties cooperate for deep learning, as shown in Fig. 1. Here, we define a server as a party that provides a model and clients as parties that have training data. Each client independently trains a model shared by the server using its own training data. The clients send their updated information obtained through the training to the server, and the server integrates all the information to update the model. As described above, federated learning can efficiently train high-performance models with a large number of parameters. On the other hand, attacks [2]–[5] have been proposed to infer training data based on such updated information.

For this reason, security enhancement methods have been actively researched to protect privacy against such attacks. A method [6] can compute in the encryption domain by using homomorphic cryptography [7] and retain the accuracy, while being computationally expensive. In another method [8], updated information is concealed by combining model subsampling, model shuffling, and blanket noise. The method

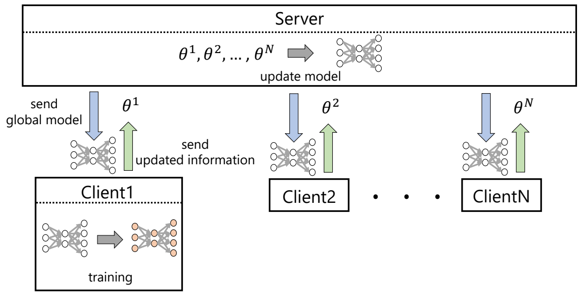


Fig. 1. Overview of federated learning

is, however, vulnerable to collusions between a server and a party shuffling the model. Consequently, this approach causes a reduction in the accuracy. There is a different type of methods based on differential privacy [9]–[13]. In this type of methods, each client adds a random number sequence following a specific distribution to the update information. In order to enhance security, however, the variance of the distribution has to be enlarged. Thus, the model performance should be commonly decreased.

There also exist researches on the use of the Vision Transformer (ViT) [14] for federated learning. ViT is an image classification model known for its high performance. On another front, a gradient leakage attack called Attention Privacy Leakage (APRIL) [5], which attempts to recover training images based on updated information of ViT, has become a serious problem. Against APRIL, Aso et al. proposed an effective model encryption method [15]. This method maintains the accuracy but does not possess resistance against attacks from internal parties such as clients. Lu et al. locked the positional embedded layer of ViT to prevent it from being updated [5]. Although this method is robust against internal attacks, the performance is degraded when using a model without pre-training.

To tackle the above issue, we propose a novel security enhancement method for ViT-based federated learning without performance deterioration. In the proposed method, each gradient of ViT is multiplied by random binary weights, and all parameters including those in the positional embedded layer can be updated. Simulation results show the effectiveness of our method through the evaluation on the classification accuracy and resistance against APRIL.

II. Preparation

We enhance security on federated leaning with ViT, assuming APRIL as a gradient leakage attack. In this section, we first give brief explanations of ViT and APRIL, and then summarize several previous methods that have resisitance against gradient leakage attacks.

A. Vision Transformer

ViT with self-attention mechanisms has attracted much attention for its high performance in the field of image recognition and classification [14]. On the other hand, in the federated learning field, there have been studies on attacks targeting ViT, and a powerful restoration attack called APRIL has been proposed [5].

Fig. 2(a) illustrates the procedure of ViT. ViT first divides an input image $x \in \mathbb{R}^{H\times W\times C}$ into patches $x_p^s \in \mathbb{R}^{P^2\times C}$, where H, W, and C are the height, width, and number of channels. $S, s \in \{1, 2, \cdots, S\}$ and P are the number of patches, the number assigned to the patch, and the height and width of the patch, respectively. A linear layer $E \in \mathbb{R}^{(P^2\times C)\times D}$ then dimensionally transforms each patch to be adequate for the Transformer Encoder layer. Note that D represents the vector length after embedding. In addition, a class token $x_{class} \in \mathbb{R}^D$ is set at the beginning of the patches to represent the features of the entire image. Next, the positional embedded layer $E_{pos} \in \mathbb{R}^{(S+1)\times D}$ embeds information about the location relationship among class tokens and patches. The calculated result $z_0 \in \mathbb{R}^{(S+1)\times D}$ is given by

$$z_0 = [x_{class}; x_P^1 E; x_P^2 E; \cdots; x_P^S E] + E_{pos}, \tag{1}$$

and input to the Transformer Encoder layer.

As shown in Fig. 2(b), the Transformer Encoder layer is further decomposed into three layers: Layer Normalization (Norm), Multi-head Self-Attention (MSA), and Multi-Layer Perceptron (MLP). Let z_{l-1} be the input and z_l be the output in the Transformer Encoder layer in the $l \in \{1, 2, \cdots, L\}$-th layer. We explain the calculation process to obtain z_l from z_{l-1} in the l-th Transformer Encoder layer. First, z_{l-1} is normalized by LN. In LN, the output $LN(z_{l-1})$ is obtained by

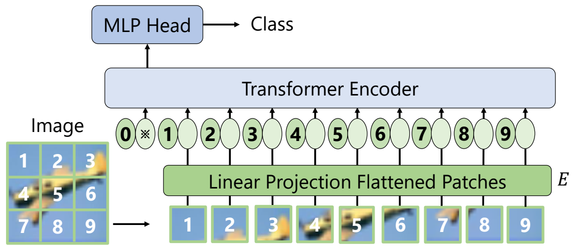
$$LN(z_{l-1}) = \gamma \frac{z_{l-1} - E[z_{l-1}]}{\sqrt{Var[z_{l-1}] + \epsilon}} + \beta, \tag{2}$$

where γ and β are learnable parameters, and ϵ is a small constant to ensure that the denominator never takes zero. Next, MSA carries out the computation to obtain the output $SA(LN(z_{l-1}))$ by using multiple Self Attentions (SAs):

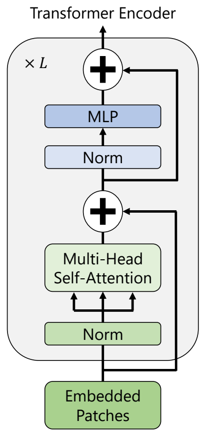
$$SA(LN(z_{l-1})) = softmax\left(\frac{q_l k_l^T}{\sqrt{D_h}}\right) v_l. \tag{3}$$

In the computation of SA, (q_l, k_l, v_l) for input $LN(z_{l-1})$ are determined by

$$\begin{cases} q_l = LN(z_{l-1})U_{ql}, \\ k_l = LN(z_{l-1})U_{kl}, \\ v_l = LN(z_{l-1})U_{vl}, \end{cases} \tag{4}$$



(a) Overview



(b) Transformer Encoder layer

Fig. 2. ViT structure

Here, $(U_{ql}, U_{kl}, U_{vl}) \in \mathbb{R}^{D\times D_h}$ are learnable matrices in l-th SA layer, and D_h is the vector length D divided by the number of heads. Additionally, (q_l, k_l, v_l) are the input queries, keys, and values linearly transformed by (U_{ql}, U_{kl}, U_{vl}). From (3), in MSA, the output $MSA(LN(z_{l-1}))$ is given by

$$MSA(LN(z_{l-1})) = \left[SA_1; SA_2; \cdots; SA_T,\right] U_{msa}, \tag{5}$$

where SA_t is the output of the $t \in \{1, 2, \cdots, T\}$-th SA and $U_{msa} \in \mathbb{R}^{k\cdot D_h\times D}$ is the matrix to return the shape of the MSA output to that of the input. According to (5), in Transformer Encoder layer, the output z_l is obtained by

$$\begin{aligned} z_l =& MLP(LN(z_l')) + z_l', \\ & where \quad z_l' = MSA(LN(z_{l-1})) + z_{l-1}. \end{aligned} \tag{6}$$

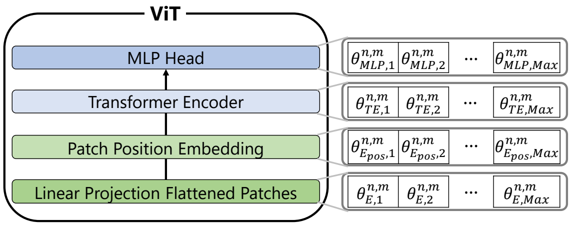
Finally, we obtain the output y of ViT that can be calculated from LN by inputting only the class token z_l^0 from the outputs of the L-th Transformer Encoder layer.

$$y = LN(z_l^0). \tag{7}$$

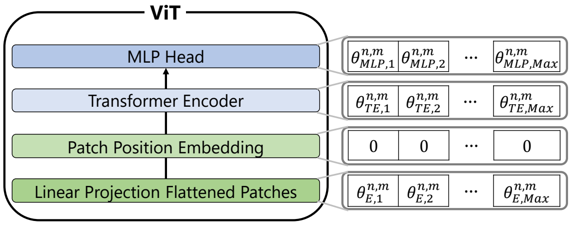
ViT predicts the label of the input image based on the output y.

B. Attention Privacy Leakage

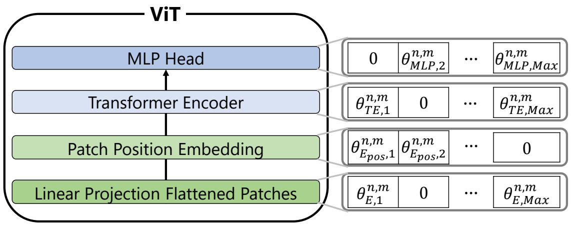
In federated learning, methods for updating a model can be broadly classified into two categories: Federated Stochastic



(a) Gradients with normal learning



(b) Gradients with fixed-positional method [5]



(c) Gradients with random binary weight (proposed)

Fig. 3. Update of gradients

Gradient Descent (FedSGD) and Federated Averaging (FedAvg) [1]. There is a gradient leakage attack called APRIL [5] that uses updated information to infer input images. APRIL targets FedSGD-based learning using ViT. We summarize the attack below.

First, attackers calculate the input to the Transformer Encoder layer z_0:

$$\begin{aligned} &\frac{\partial l}{\partial z_0} z_0^T = U_{q1}^T \cdot \frac{\partial l}{\partial U_{q1}} + U_{v1}^T \cdot \frac{\partial l}{\partial U_{v1}} + U_{k1}^T \cdot \frac{\partial l}{\partial U_{k1}}, \\ &\frac{\partial l}{\partial z_0} = \frac{\partial l}{\partial E_{pos}}. \end{aligned} \tag{8}$$

Here, $\frac{\partial l}{\partial *}$ is the gradient of the parameter $*$ on the loss function. Next, the inferred image x' is obtained by inversion of (1):

$$x' = E \times (z_0 - E_{pos})^T. \tag{9}$$

In (9), E and E_{pos} refer to the parameters of the global model before being updated. APRIL attempts to recontruct the input images for learning by the above procedure.

C. Previous Method

Several security enhancement methods for federated learning have been proposed. A representative method is differential privacy [9]–[13]. In differential privacy, we add noise to the updated information so as to enhance security at a low computational cost. However, differential privacy certainly degrades model performance. Another method lets a third party to scrambles the updated information for high anonymity of each client [8]. This approach requires the assumption that the third party is absolutely credible.

In contrast, there is another security enhancement method for federated learning with ViT, which is called fixed-posional method [5]. This method is implemented assuming FedSGD. In FedSGD, the update of parameters is expressed by

$$w_{A,i}^{m+1} = w_{A,i}^{m} - \eta \frac{\sum_{n=1}^{N} \theta_{A,i}^{n,m}}{N}, \tag{10}$$

where $w_{A,i}^m + 1$ and $w_{A,i}^m$ are the i-th global model parameters of the A-th layer after $m+1$-th or m-th updates, η is the learning rate, N is the number of clients, and $\theta_{A,i}^{n,m}$ is the gradient to update $w_{A,i}^m$ calculated by the n-th client. The parameter update is carried out for each batch training. Fig. 3(a) depicts the gradients obtained from training using a plain FedSGD. In this figure, Max is the number of gradients in each layer.

The fixed-positional method enhances security while not learning the positional embedded layer. Fig. 3(b) indicates gradients transmitted in federated learning with the fixed-positional method. All the gradients in the positional embedded layer are changed to 0. In this case, the learning process is described as follows.

Step1: A server sends global model information to each client.
Step2: Each client trains each local model using his/her own dataset.
Step3: Each client converts gradients in the positional embedded layer to 0.
Step4: Each client sends the updated gradients to the server.
Step5: The server integrates the information sent from all the clients and updates the global model.

We repeat the steps a specified number of times. The fixed-positional method is highly robust against APRIL. The attacker can be a server, one of clients, or an external third party; the security enhancement method is effective against all of them. The method, however, changes all gradients in the positional embedded layer to zero, so we cannot update the parameters. This leads to severe degradation of model performance when learning a model without pre-training.

III. Proposed Method

We propose a new method to enhance security for federated learning using ViT. The proposed method is robust against APRIL without affecting model performance.

A. Main procedure

A series of the following steps is a procedure of federated learning to which the proposed method is applied.

Step1: A server sends global model information to each client.
Step2: Each client trains each local model using his/her own dataset.
Step3: Each client multiplies gradients by a random binary sequence.
Step4: Each client sends the updated gradients to the server.
Step5: The server integrates the information sent from all the clients and updates the global model.

We repeat this procedure a predefined number of times. The main different from the previous method [5] is Step 3. Next, we explain Steps 3 and 5 in more detail.

B. Multiplication of Random Binary Weights

Fig. 3(c) illustrates a simplified gradient processing for each layer of ViT in the proposed method. In Step 3, each client multiplies gradients by random binary weights $B_{A,i}^{n,m} \in \{0,1\}$. Here, a weighted gradient $\theta_{A,i}^{n,m'}$ is given by

$$\theta_{A,i}^{n,m'} = B_{A,i}^{n,m} \times \theta_{A,i}^{n,m}, \tag{11}$$

where $B_{A,i}^{n,m}$ is a binary weight that is multiplied by $\theta_{A,i}^{n,m}$. In the case that the occurrence probability of zeros $R \in [0,1]$ is set higher, more robust privacy protection would be provided; however, the model may not be updated correctly. We tackle this issue by changing the random binary weights $B_{A,i}^{n,m}$ at each epoch.

It is possible to define a different probability for each layer. The positional embedded layer E_{pos} and linear layer E are layers that include major information of the image, so APRIL uses these layers for image restoration. Thus, for instance, it would be useful to assign high probabilities for E_{pos} and E, while assigning low probabilities for other gradients. In this paper, we apply the same probabilities R to all layers to simplify argument.

C. Gradient Integration

Next, we elaborate gradient integration in Step 5. The proposed method integrates gradients based on FedSGD. However, it is not appropriate to directly apply (10) to the proposed method because each client randomly converts the gradients to 0. The gradient integration is implemented as:

$$w_{A,i}^{m+1} = \begin{cases} w_{A,i}^{m} - \eta \frac{\sum_{n=1}^{N} \theta_{A,i}^{n,m'}}{\sum_{n=1}^{N} B_{A,i}^{n,m}} & \text{if } 0 < \sum_{n=1}^{N} B_{A,i}^{n,m} \leq N \\ w_{A,i}^{m} & \text{if } \sum_{n=1}^{N} B_{A,i}^{n,m} = 0. \end{cases} \tag{12}$$

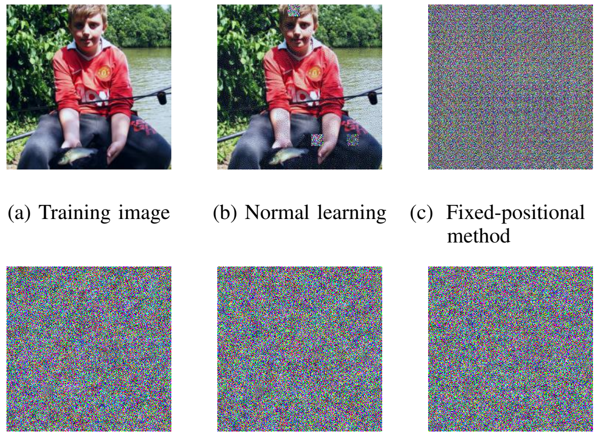
Note that zero gradients are excluded when calculating the mean of the gradients. As an exception, parameters cannot be updated in the case of $\sum_{n=1}^{N} B_{A,i}^{n,m} = 0$.

IV. Experimental Results and Discussion

In this section, we clarify the effectiveness of the proposed method through the experiments from the aspects of attack resistance and model performance.

TABLE I: Fundamental conditions

# of clients	5
Pre-trained model	vit_small_patch16_224
Patch size	16
Learning rate	0.0001
Dataset	CIFAR10
Occurrence probability of zeros	0.2, 0.5, and 0.8



(a) Training image (b) Normal learning (c) Fixed-positional method

(d) Proposed method (R=0.2) (e) Proposed method (R=0.5) (f) Proposed method (R=0.8)

Fig. 4. Inferred images by APRIL

A. Setup

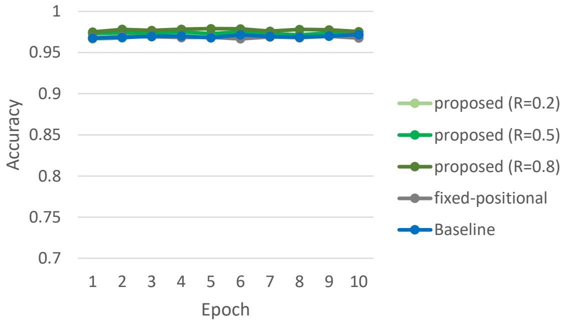
Table I lists the fundamental conditions of our experiments. We virtually configured a server and five clients on a single machine. For a pre-trained model, we used vit_small_patch16_224, where the patch size P and learning rate η were 16 and 0.0001, respectively. We futher set the occurrence probability of zeros R to 0.2, 0.5, and 0.8. The experiments were conducted using the CIFAR10 dataset that consists of 50,000 training images and 10,000 test images. The CIFAR10 images were resized from $32 \times 32 \times 3$ to $224 \times 224 \times 3$ by bilinear transformation to correspond to the input size of ViT.

We used 16 images from the CIFAR10 dataset to evaluate robustness against APRIL. The code of APRIL is referenced from [16]. The images were input to the model fine-tuned with CIFAR10, and the inferred images were generated by APRIL from updated information.

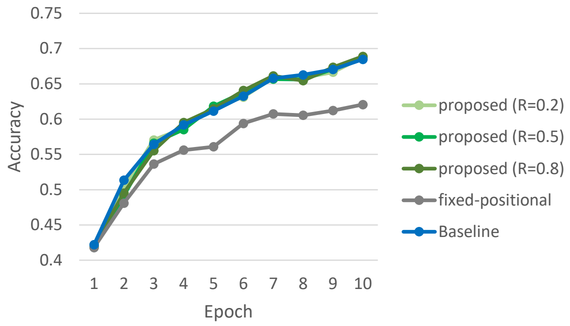
Additionally, we evaluated model performance. Using a pre-trained model using Image-Net and another model without pre-training, we verified classification accuracy using the 10,000 test images at the end of each epoch.

B. Robustness against APRIL

Fig. 4 depicts the restoration results by APRIL using the updated information. Fig. 4(a) is a training image, while 4(b) is an inferred image when using a plain model. In such a case, the visual information of the training image was fully disclosed by APRIL. In contrast, as shown in Fig. 4(c), the fixed-positional method prevented APRIL from successful

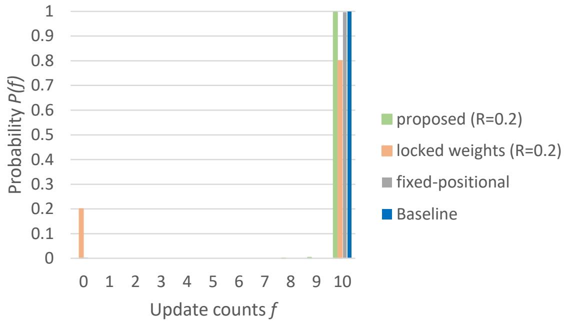


(a) W/ pre-training

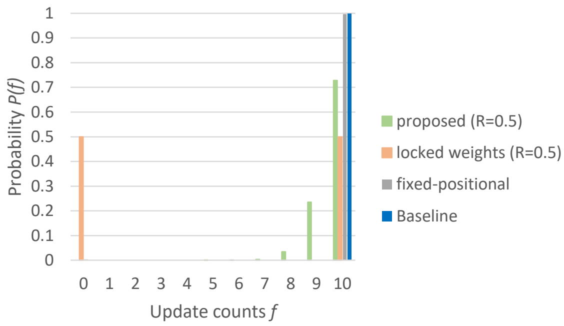


(b) W/o pre-training

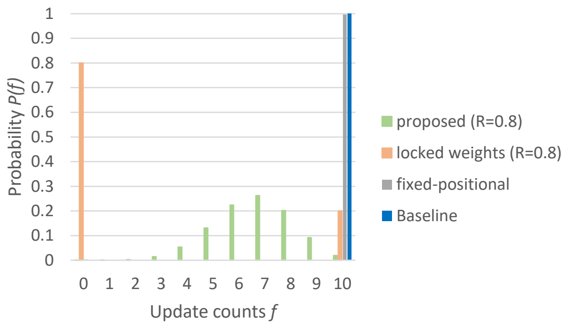
Fig. 5. Comparison of classification accuracy



(a) $R = 0.2$



(b) $R = 0.5$



(c) $R = 0.8$

Fig. 6. Probability distribution of update counts

restoration. Similarly, from Fig. 4(d), (e), and (f), which are the inferred images in the proposed method, it was difficult for APRIL to successfully restore the training image even at $R = 0.2$. Thus, we can claim that the proposed method is as robust to APRIL as the fixed-positional method.

C. Image Classification Performance

We assess the influence of security enhancement by the proposed and fixed-positional methods on classification accuracy. Fig. 5(b) shows the transition of classification accuracy for each epoch in the case of using a pre-trained model. In this figure, the baseline is federated learning with a plain model. It is clear that both the proposed and fixed-positional methods retain the analogous accuracy as the baseline.

On another front, Fig. 5(c) is the transition of classification accuracy for each epoch using a model without pre-training. Although the proposed method could maintain the accuracy, the fixed-positional method significantly reduced the accuracy. In the case of using a model without pre-training, we should update all layers of ViT because the initial parameters are not generally suitable to classify desired images. The fixed-positional method does not update the positional embedded layer; this is the reason for the reduction of the accuracy. In contrast, the proposed method updates all the layers, so the accuracy can be improved.

D. Discussion

We discuss the reason why the classification accuracy was not degraded in the proposed method. If the random binary weight of each client for a given parameter is zero, the parameter is not updated. Thus, in the case that each client use a common sequence of random binary weights throughout all the epochs, some of the parameters may never be updated. We call this a locked binary weights method. In contrast, in the proposed method, the clients use a different sequence of random binary weights varying from epoch to epoch. This significantly reduces the probability that parameters will never be updated. In the proposed method, the probability $P(f)$ that the parameter is updated f times is obtained by

$$P(f) = {}_mC_f \times (1 - R^n)^f \times R^{n\times(1-f)}, \tag{13}$$

where m, R, and n denote the number of epochs, the occurrence probability of zeros, and the number of clients, respectively. From this equation, the update frequency of each parameter will get larger as the number of clients and epochs increases. Consequently, the proposed method is expected to train the model more effectively compared to the locked binary weights method.

Fig. 6 shows the probability distribution of update counts with $m = 10$ and $n = 5$. Here, in the proposed method and the locked binary weights method, R was defined as 0.2, 0.5, and 0.8. From this results, it is clear that most of all parameters were constantly updated through 10 epochs in the fixed-positional method. This method, however, never updates the positional embedded layer. The updates of the positional embedded layer is important to improve classification accuracy, so the fixed-positional method degrades the accuracy. Note that it is difficult to recognize this in Fig.6, since the number of the parameters in the positional embedded layer is significantly small. The proposed method has a large number of updated parameters, while not as large as the fixed-positional method. Since no-updated parameters are not concentrated in a particular layer, our method can also attained the updates of positional embedded layer, which is the serious issue in the fixed-positional method. Furthermore, the use of independent sequence of binary weights for each epoch enabled the proposed method to update more parameters than the locked binary weights method at the defined values of R in this section, that is $R = 0.2, 0.5, and, 0.8$. For these reasons, the proposed method provides not only enhanced security against attacks but also updates of a large number of parameters.

V. Conclusion

We propose a new security enhancement method for ViT-based federated learning. Most previous methods involved degradation of classification accuracy. The proposed method not only to solve this issue but also attain to be robust to the APRIL attack by multiplying updated information by random binary weights. Through our simulations, we confirmed the resistance to APRIL and retention of model performance. Our future work involves application of this method to other models such as CNN models.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B.A.Y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Proc. Int. Conf. Artif. Intell. Stat.,* pp.1273-1282, 2017.

[2] Y. Wen, J. Geiping, L. Fowl, M. Goldblum, and T. Goldstein,"Fishing for User Data in Large-Batch Federated Leaarning via Gradient Magnification, " *Proc. Int. Conf. Machine Learning,* pp.23668-23684, 2022.

[3] L. Zhu, Z. Liu, and S. Han, "Deep Leakage from Gradients, " *Proc. Int. Conf. NeurIPS.,* pp.14774-14784, 2019.

[4] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting Unintended Feature Leakage in Collaborative Learning," *Proc. Int. Conf. IEEE Secur. Priv.,* pp.691-706, 2019.

[5] J. Lu, X. S. Zhang, T. Zhao, X. He, and J. Chang, "April: Finding the Achilles' Heel on Privacy for Vision Transformers, " *Proc. CVPR.,* pp.10051-10060, 2022.

[6] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-Preserving Deep Learning via Additively Homomorphic Encryption, " *IEEE Trans. Inf. Forensics Security,* vol.13, pp.1333-1345, 2017.

[7] A. C. Yao, "Protocols for Secure Computations, " *Proc. Int. Conf. IEEE FOCS.,* pp.160-164, 1982.

[8] R. Liu, Y. Cao, H. Chen, R. Guo, and M. Yoshizawa, "Flame: Differentially Private Federated Learning in the Shuffle Model, " *Proc. Int. AAAI Conf. Artif. Intell.,* vol.35, no.10, pp.8688-8696, 2021.

[9] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially Private Empirical Risk Minimization, " *Journal of Machine Learning Research,* vol.12, pp.1069-1109, 2011.

[10] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, et al., "A Hybrid Approach to Privacy-Preserving Federated Learning, " *Proc. Int. Conf. AlSec.,* pp.1-11, 2019.

[11] A. Triastcyn, and B. Faltings, "Federated Learning with Bayesian Differential Privacy, " *Proc. Int. Conf. IEEE Big Data,* pp.2587-2596, 2019.

[12] A. E. Ouadrhiri, and A. Abdelhadi, "Differential Privacy for Deep and Federated Learning: A Survey, " *IEEE Access,* vol.10, pp.22359-22380, 2022.

[13] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, et al., "Federated Learning with Differential Privacy: Algorithms and Performance Analysis, " *IEEE Trans. Inf. Forensics Security,* vol.15, pp.3454-3469, 2020.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, " *Proc. ICLR., Virtual Event, Austria,* pp.1-21, 2021.

[15] R. Aso, S. Shiota, and H. Kiya, "Enhanced Security with Encrypted Vision Transformer in Federatd Learning. " *Proc. Int. Conf. IEEE Global Conference on Consumer Electronics* pp.819-822, 2023.

[16] [Online]. Available: https://github.com/JonasGeiping/breaching