ADI: Adversarial Dominating Inputs in Vertical Federated Learning Systems

***Abstract*—Vertical federated learning (VFL) system has recently become prominent as a concept to process data distributed across many individual sources without the need to centralize it. Multiple participants collaboratively train models based on their local data in a privacy-preserving manner. To date, VFL has become a de facto solution to securely learn a model among organizations, allowing knowledge to be shared without compromising privacy of any individuals.**

Despite the prosperous development of VFL systems, we find that certain inputs of a participant, named adversarial dominating inputs (ADIs), can dominate the joint inference towards the direction of the adversary's will and force other (victim) participants to make negligible contributions, losing rewards that are usually offered regarding the importance of their contributions in federated learning scenarios.

We conduct a systematic study on ADIs by first proving their existence in typical VFL systems. We then propose gradient-based methods to synthesize ADIs of various formats and exploit common VFL systems. We further launch greybox fuzz testing, guided by the saliency score of "victim" participants, to perturb adversary-controlled inputs and systematically explore the VFL attack surface in a privacy-preserving manner. We conduct an in-depth study on the influence of critical parameters and settings in synthesizing ADIs. Our study reveals new VFL attack opportunities, promoting the identification of unknown threats before breaches and building more secure VFL systems.

References

[1] Visual Question Answering in the Medical Domain. https://www.imageclef.org/2020/medical/vqa.

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. ICCV, 2015.

[3] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. ECCV, 2020.

[4] Fenglin Liu, Xian Wu, Shen Ge, Wei Fan, and Yuexian Zou. Federated learning for vision-and-language grounding problems. AAAI, 2020.

[5] A Lubna, Saidalavi Kalady, and A Lijiya. MoBVQA: A modality based medical image visual question answering system. IEEE TENCON, 2019.

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE TPAMI, 2016.

Appendix

A. *Training and Implementation Details of Learning Protocols*

HeteroLR. Given n training samples X_i whose label is $y_i \in \{0, 1\}$ and coefficient vector θ, logistic regression can be described in a basic form as follows:

$$\mathbb{P}(y_i = 1|X_i, \theta) = h_\theta(X_i) = \frac{1}{1 + e^{-\theta^T X_i}} \tag{1}$$

In typical VFL settings, X_i is partitioned as $[X^i_\mathcal{A}||X^i_\mathcal{B}]$, while θ is partitioned as $[\theta_\mathcal{A}||\theta_\mathcal{B}]$. Therefore, HeteroLR in VFL can be formulated as follows:

$$\begin{aligned}\mathbb{P}(y_i = 1|X^i_\mathcal{A}, X^i_\mathcal{B}, \theta_\mathcal{A}, \theta_\mathcal{B}) &= h_{\theta_\mathcal{A},\theta_\mathcal{B}}(X^i_\mathcal{A}, X^i_\mathcal{B}) \\ &= \frac{1}{1 + e^{-\theta^T_\mathcal{A} X^i_\mathcal{A} - \theta^T_\mathcal{B} X^i_\mathcal{B}}}\end{aligned} \tag{2}$$

whose training objective is to minimize the loss function:

$$\begin{aligned}\mathbb{L}_\theta = -\frac{1}{n}\sum_{i=1}^{n} & y_i\, log(h_{\theta_\mathcal{A},\theta_\mathcal{B}}(X^i_\mathcal{A}, X^i_\mathcal{B})) \\ &+ (1 - y_i)\, log(1 - h_{\theta_\mathcal{A},\theta_\mathcal{B}}(X^i_\mathcal{A}, X^i_\mathcal{B}))\end{aligned} \tag{3}$$

We have introduced the general procedure of HeteroLR in background section. At the training stage, when computing gradients, Taylor approximation and homomorphic encryption scheme can be utilized to make training more efficient. To make a joint inference for user ID i, $\mathcal{C}$ collects local prediction results $\theta^T_\mathcal{A} X^i_\mathcal{A}$ and $\theta^T_\mathcal{B} X^i_\mathcal{B}$. It then computes and sends the probability $\frac{1}{1+e^{-\theta^T_\mathcal{A} X^i_\mathcal{A} - \theta^T_\mathcal{B} X^i_\mathcal{B}}}$ back to $\mathcal{A}$ and $\mathcal{B}$. Note that all the intermediate data is encrypted.

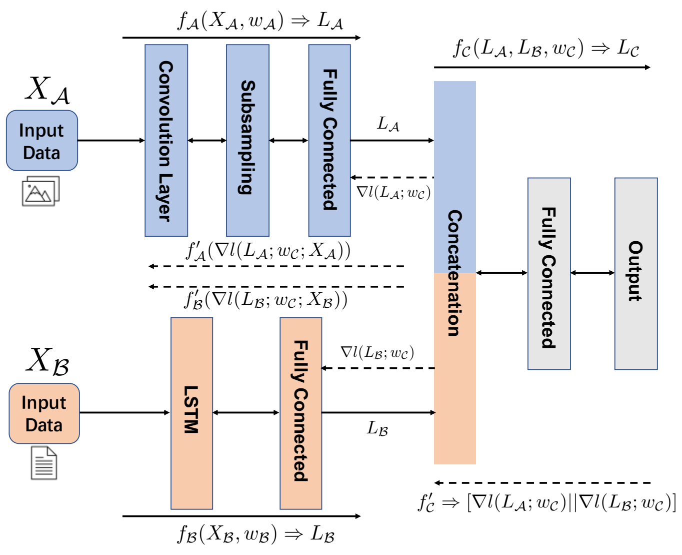
$f_\mathcal{A}(X_\mathcal{A}, w_\mathcal{A}) \Rightarrow L_\mathcal{A}$
$f_\mathcal{C}(L_\mathcal{A}, L_\mathcal{B}, w_\mathcal{C}) \Rightarrow L_\mathcal{C}$
$X_\mathcal{A}$
Input Data
Convolution Layer
Subsampling
Fully Connected
$L_\mathcal{A}$
$\nabla l(L_\mathcal{A}; w_\mathcal{C})$
Concatenation
Fully Connected
Output
$f'_\mathcal{A}(\nabla l(L_\mathcal{A}; w_\mathcal{C}; X_\mathcal{A}))$
$f'_\mathcal{B}(\nabla l(L_\mathcal{B}; w_\mathcal{C}; X_\mathcal{B}))$
$X_\mathcal{B}$
Input Data
LSTM
Fully Connected
$\nabla l(L_\mathcal{B}; w_\mathcal{C})$
$L_\mathcal{B}$
$f'_\mathcal{C} \Rightarrow [\nabla l(L_\mathcal{A}; w_\mathcal{C})||\nabla l(L_\mathcal{B}; w_\mathcal{C})]$
$f_\mathcal{B}(X_\mathcal{B}, w_\mathcal{B}) \Rightarrow L_\mathcal{B}$

Fig. 1: Architecture of SplitNN.

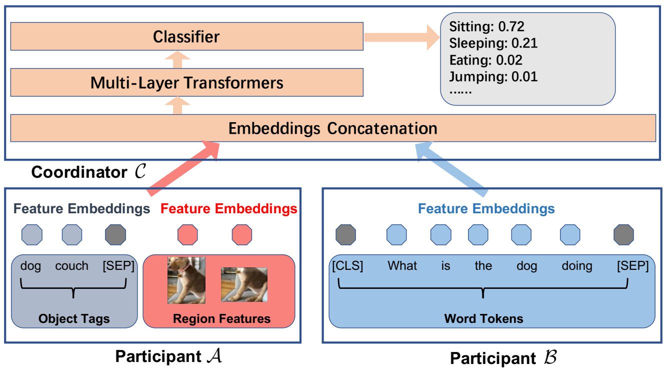

Fig. 2: Architecture of VFVQA.

SplitNN. Fig. 1 depicts the architecture of heterogeneous neural network (also known as a special case of SplitNN). SplitNN facilitates multiple participants holding data with different feature spaces to train a distributed model without sharing raw data. In the forward propagation phase, $\mathcal{A}$ and $\mathcal{B}$ compute the outputs of their local models ($w_\mathcal{A}$ and $w_\mathcal{B}$) and forward them to the coordinator $\mathcal{C}$:

$$\begin{aligned} L_\mathcal{A} &= f_\mathcal{A}(X_\mathcal{A}, w_\mathcal{A}) \\ L_\mathcal{B} &= f_\mathcal{B}(X_\mathcal{B}, w_\mathcal{B}) \end{aligned} \tag{4}$$

The forwarded local outputs, $L_\mathcal{A}$ and $L_\mathcal{B}$, will be concatenated on the coordinator side and fed to the coordinator model ($w_\mathcal{C}$) for a joint inference whose result is $L_\mathcal{C}$.

$$L_\mathcal{C} = f_\mathcal{C}([L_\mathcal{A} \,||\, L_\mathcal{B}], w_\mathcal{C}) \tag{5}$$

In the backward propagation phase, the coordinator $\mathcal{C}$ computes the gradients $f'_\mathcal{C}(L_\mathcal{C})$, performs gradient descent on its model, and obtains the gradients of local outputs submitted by the participants as follows:

$$[\nabla l(L_\mathcal{A}; w_\mathcal{C}) \,||\, \nabla l(L_\mathcal{B}; w_\mathcal{C})] = f'_\mathcal{C}(L_\mathcal{C}) \tag{6}$$

Back to the participant side, once receiving the gradients, the participants further compute the local gradients $f'_\mathcal{A}(\nabla l(L_\mathcal{A}; w_\mathcal{C}; X_\mathcal{A}))$ and $f'_\mathcal{B}(\nabla l(L_\mathcal{B}; w_\mathcal{C}; X_\mathcal{B}))$ and then update local models. f represents the neural network and f' represents its derivative. In the forward and backward propagations, no raw data is directly sent to the coordinator or exchanged among participants, and all intermediate data is encrypted. Furthermore, by extending the SplitNN protocol, we can construct more complex models like CNN and LSTM following the VFL paradigm.

As introduced in background section, VFVQA concretizes and extends the SplitNN protocol, which involves considerable

VFVQA. Visual Question Answering [2], a popular multimodal learning task, answers open-ended natural-language questions about images. VQA models develop a joint understanding of questions and images and have been employed in privacy-sensitive scenarios like medical image diagnosis [1], [5]. VFL provides a practical and effective solution to facilitate privacy-preserving VQA [4]. In our evaluation, we modify the state-of-the-art VQA model Oscar [3] and construct VFVQA as illustrated in Fig. 2.
engineering efforts to link with several modern computer vision and natural language processing models. $\mathcal{A}$ holds the image data and their associated object tags. It computes the feature representation of data using FasterRCNN [6] and BERT [3], where $X_{\mathcal{A}_1}$ and $X_{\mathcal{A}_2}$ represent the image data and the corresponding tags, respectively. $\mathcal{B}$ holds the textual data (i.e., “questions” in VQA) and computes the embedding results of each question using BERT. In particular, $\mathcal{A}$ holds the image data and their associated object tags. It uses FasterRCNN [6] and BERT [3] to process $X_{\mathcal{A}_1}$ and $X_{\mathcal{A}_2}$, corresponding to images and tags. $\mathcal{B}$ process the textual data (i.e., “questions” in VQA tasks) using BERT:

$$\begin{aligned} L_\mathcal{A} &= [FasterRCNN(X_{\mathcal{A}_1}, w_{\mathcal{A}_1}) \,||\, BERT(X_{\mathcal{A}_2}, w_{\mathcal{A}_2})] \\ L_\mathcal{B} &= BERT(X_\mathcal{B}, w_\mathcal{B}) \end{aligned} \tag{7}$$

As specified by the SplitNN protocol, local results on each participant are sent to the coordinator $\mathcal{C}$. Then, $\mathcal{C}$ predicts the answer, another natural language sentence, to the raised question on $\mathcal{B}$ based on the image feature shared by $\mathcal{A}$ using multi-layer transformers:

$$L_\mathcal{C} = Transformer([L_\mathcal{A} \,||\, L_\mathcal{B}], w_\mathcal{C}) \tag{8}$$

The backward propagation phase is performed consistently with SplitNN. And all intermediate data in the above phases are encrypted.

B. *Cooperating Fuzz Testing with VFL Protocols*

As reported in evaluation section, we launch fuzz testing toward two widely-used VFL protocols (i.e. HeteroLR and SplitNN) on the basis of real-world VFL platforms FATE and FedML. We also launch fuzz testing toward VFVQA, whose protocol is consistent with SplitNN. A detailed procedure to cope with fuzz testing with learning protocols is given in Table I. In all, greybox fuzzing can be accordingly performed on the distributed versions of learning protocols without sharing much information about the original data.

TABLE I: Detail steps to cooperate fuzz testing with the VFL protocols.

	Participant $\mathcal{A}$	**Participant $\mathcal{B}$**	**Coordinator $\mathcal{C}$**
1.	Generate index Seeds **S**, maintain a list **Q** storing the corresponding data.	Compute all local results of data in participant $\mathcal{B}$, send to $\mathcal{C}$	
2.	a) Choose next from **S** as $index_a$. b) Add noise to **Q**$[index_a]$ for β times. c) Compute local results of the noised data, send to $\mathcal{C}$.		
3.			a) Randomly choose a local result of participant $\mathcal{B}$ and the corresponding index is $index_b$. b) Compute the outputs of all noised data and gradients, send to $\mathcal{A}$ and $\mathcal{B}$.
4.	Compute the saliency score of all noised data, send the highest $saliency\_score_a$ and corresponding $index\_noise_a$ to $\mathcal{C}$.	Compute the $saliency\_score\_orig_b$, send to $\mathcal{C}$.	
5.			Compute Attack Success Rate, send to $\mathcal{A}$.
6.	a) Maintain the $orig\_acc$. b) Compute the Saliency Mask based on the gradients received. c) Compute $data\_noise\_mask_a$ based on the Mask. d) Compute the local result of $data\_noise\_mask_a$, send to $\mathcal{C}$.		
7.			Compute output, gradients and Attack Success Rate ($masked\_acc$), send to $\mathcal{A}$ and $\mathcal{B}$.
8.	Compute $saliency\_score\_masked_a$, send to $\mathcal{C}$.	Compute $saliency\_score\_masked_b$, send to $\mathcal{C}$.	
9.			Compute $ratio_a$ = $saliency\_score\_masked_a$ / $saliency\_score\_orig_a$ and $ratio_b$ = $saliency\_score\_masked_b$ / $saliency\_score\_orig_b$, send to $\mathcal{A}$.
10.	If $masked\_acc > orig\_acc$ and $ratio_a > ratio_b$, update **Q**$[index_a]$ = $data\_noise\_mask_a$. If $masked\_acc >$ threshold, add **Q**$[index_a]$ to $\mathcal{A}$, delete $index_a$ from **S** and return to Step 2.		
11.	Return to step 3 for γ times.		
12.	Return to step 2 for Γ times.		