Stealing Data from Active Party in Vertical Split Learning

Yaxin Liu[1], Xiaoyang Xu[1], Wenzhe Yi[1], Yong Zhuang[1], Juan Wang[1] ✉, Mengda Yang[1], and Ziang Li[1]

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University
{yaxin.liu}@whu.edu.cn

Abstract. Vertical Split Learning (VSL) facilitates collaborative learning among users with vertically partitioned data but also introduces risks of private data leakage. Existing reconstruction attacks primarily rely on intermediate feature access, making them ineffective against semi-honest passive adversaries who lack such access. In this paper, we propose PASTA, a novel attack framework that enables the PAssive party to STeal private data from the Active party without direct feature access. Our approach consists of three steps. First, we leverage an autoencoder to establish an initial reconstruction by analyzing correlations between sample features. Second, we construct a shadow VSL model to mimic server-side gradient behaviors. Finally, we refine the reconstruction using a U-Net-based network with gradient-based guidance. Our reconstruction results on CIFAR-10 and CelebA achieved SSIM scores of 0.5132 and 0.5877, and LPIPS scores of 0.3395 and 0.2771, respectively. Ablation study demonstrated that even without access to auxiliary data from the same distribution, the attack could still reveal most of the image details. We further validated the effectiveness of our attack on real-world datasets Tiny-ImageNet and LFW. We also conducted experiments on ResNet18, VGG16, ViT-B16, and MobileNet to show that our attack is model-agnostic.

Keywords: Vertical Split Learning · Data Privacy · Data Reconstruction Attack.

1 Introduction

Vertical federated learning (VFL) enables participants with distinct feature spaces but common sample spaces to collaboratively develop models without sharing raw data. This approach effectively addresses users' concerns about data privacy while maximizing the use of multi-source data. Vertical split learning (VSL), a specialized approach within the VFL framework, further enhances efficiency by strategically partitioning the model between clients and servers. This balances computational load, network transmission pressure and latency while achieving optimal performance with minimal resource consumption [1,2,3]. Currently, VSL

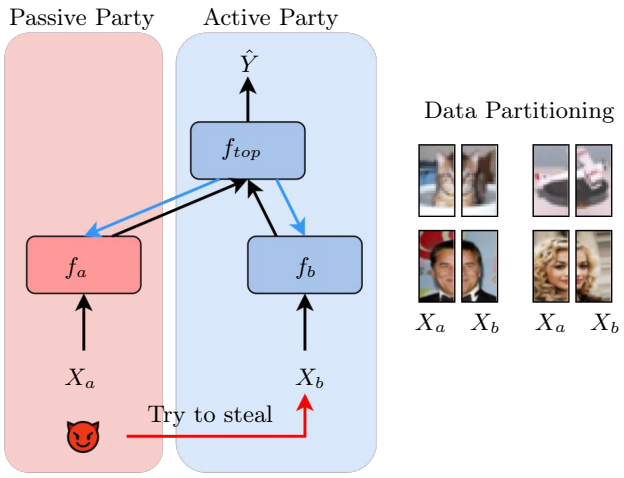

Fig. 1: A two-party VSL system. The uploaded intermediate features are represented by black lines and the returned gradients are represented by blue lines.

has demonstrated its potential applications in various fields, including healthcare [1,4,5] and cloud computing [6].

In VSL, entities that possess partial sample features and a client model are referred to as passive parties, while the entity that owns the server model and labels in addition to partial sample features and a client model is called the active party. Fig. 1 illustrates a VSL system comprising both a passive party and an active party. Most previous studies on data reconstruction attacks in VSL assume that the active party is the attacker, reconstructing private data by leveraging intermediate features uploaded to the server by passive parties through inversion networks. This is similar to data reconstruction attacks in horizontal split learning (HSL), where extensive research has been conducted [7,8,9]. In contrast to the aforementioned studies, Chen et al. [10] are the first to consider the passive party as the attacker, introducing the concept of spy attack. However, their approach only recovers data lost by the passive party itself and does not involve stealing data from the active party, thereby downplaying the potential risks associated with spy attack. To date, there has been no research on how a passive party could steal data from an active party.

In this paper we investigate for the first time the potential for a passive party to conduct data reconstruction attacks against an active party and propose an innovative attack strategy. In VSL, the passive party inherently possesses a portion of the sample features. We also assume that it can gather data from the Internet within the same domain as an auxiliary dataset. Based on this premise, we propose a three-step attack approach. First, we construct an autoencoder-based network to uncover the intrinsic relationships between sample features.

By utilizing part of the passive party's sample features, we can reconstruct the active party's private data as an initial approximation. Second, to fully leverage the information embedded in gradients, we must first learn the gradient patterns generated by the active party's server model. To achieve this, we train a shadow VSL system with the same task as the normal VSL system simultaneously using the auxiliary dataset. Finally, we employ a network based on the U-Net architecture that takes the initial reconstructed data and gradients as input to further enhance the accuracy of the reconstruction. The entire attack process remains undetectable to the active party, making it challenging to defend against.

The main contribution of this paper can be concluded as follows:

• We identify a new angle for reconstruction attacks targeting VSL systems and explored for the first time the possibility of a passive party stealing data from an active party. Through this research, we reassess the security of VSL.

• We propose an innovative data reconstruction attack to generate relatively good reconstructed data using the priori knowledge embedded in partial sample features held by the passive party and the posteriori information of the active party's private data extracted from gradients returned by the server.

• Extensive experimental results demonstrate the effectiveness of our attack strategy. We obtained good reconstruction on image datasets such as CIFAR-10 and CelebA, with SSIM and LPIPS values showing strong performance. Additional ablation experiments confirmed the validity of our design.

2 Backgrounds

2.1 Vertical Split Learning

VSL assumes that the data are partitioned by features. As an example, we provide a formal definition of the VSL system shown in Fig. 1 in the context of supervised classification.

Let the participants be P_a and P_b with a client model f_a and f_b respectively. Their local datasets are represented as X_a and X_b, where $X_i = (U, F_i)(i = a, b)$. Here, U denotes the common sample space, while F_i represents the distinct feature space of each party. In this setting, P_a is designated as the passive party, while P_b acts as the active party, holding the label information Y as well as the server model f_{top}.

During the training process of VSL, all participants process local data using their own client models and send intermediate features $Z_i = f_i(X_i)$ to the active party's server. The server then concatenates these intermediate features $Z = Concat(Z_1, ..., Z_i)$ and feeds them into the server model for subsequent computation $\hat{Y} = f_{top}(Z)$. During the model update process, the backpropagation gradients are passed back to individual client models through the split layer. The inference phase is similar to the training phase but without backpropagation.

2.2 Data Reconstruction Attacks on Split Learning

Data reconstruction attacks aim to exploit information such as model parameters, gradients or outputs to reconstruct the original data, thereby compromising data privacy.

Existing data reconstruction attacks in split learning typically assume that the attacker is located on the server and can easily access the intermediate features Z uploaded by the clients. In this case, since the client model f typically consists of a series of simple convolutional layers, He et al. [11] suggested that the server-side attacker could build an inversion network f^{-1} made up of deconvolutional layers, aiming to reverse-map the intermediate features Z back to the sample space:

$$X^* = f^{-1}(f(X)) \tag{1}$$

In VSL, the attacker may act as the passive party. As shown in Fig. 1, the attacker only owns a client model and cannot obtain the intermediate features uploaded by other clients. However, they can access the gradients provided by the active party during the training phase. Therefore, it is necessary to reconsider the attack steps.

Chen et al. [10] proposed that a passive party can falsely claim to have missing data and replace them with random noise to participate in VSL training process. After obtaining the gradients returned by the server, the passive party can restore the missing data using an inversion network. However, their method solely focuses on recovering the data lost by the passive party itself and does not address the aspect of stealing data from the active party, which minimizes the perceived dangers linked to spy attacks.

3 Method

In this section, we present the threat model and methodological details of the data reconstruction attack that the passive party performs on the active party. The overview of the proposed method is shown in Fig. 2.

3.1 Threat Model

We assume that the attacker is the passive party P_a in a VSL system and owns a client model f_a together with half of the sample features X_a. It is honest but curious about the private training data X_b of the active party P_b. The attacker is aware of the structure of the active party's client model f_b. In addition, the attacker can collect an auxiliary dataset X^{aux} in the same domain as the private training data X from the Internet.

3.2 Obtain Initial Reconstruction via Partial Sample Features

We observe that in many deep learning prediction tasks, it is often necessary to utilize certain features to infer other single or multiple features, which clearly

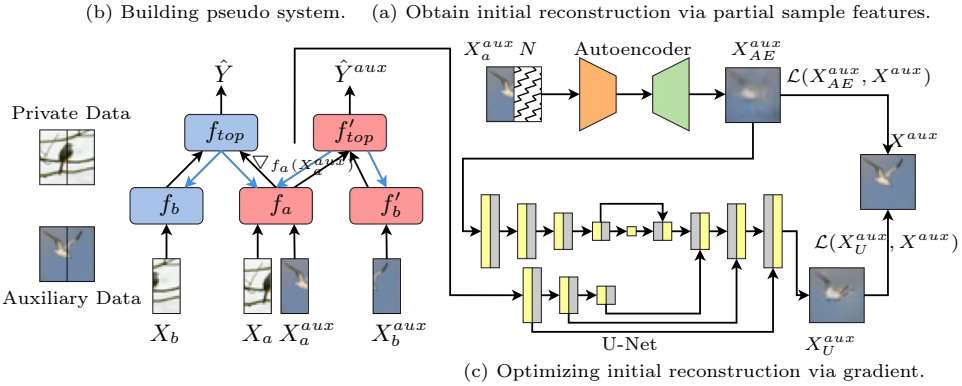

Fig. 2: The overview of the proposed attack's training process. The attacker first mines the associations between the sample features via an autoencoder to generate the initial reconstruction $\hat{X}_{AE}^{aux}$ (a). Next, a shadow VSL system is constructed for simulating the server's behavior in generating gradients (b). Finally, the $\hat{X}_{AE}^{aux}$ is optimized by a U-Net network using the posteriori information of the private data contained in the gradient $\nabla_{f_a(X_a^{aux})}$ to obtain a more accurate reconstruction $\hat{X}_U^{aux}$ (c). In this case, steps (a) and (c) can be done offline, while step (b) needs to be performed online.

indicates that there is an intrinsic connection between the sample features. This property is particularly evident in image data, where there is a strong correlation between image pixels.

Based on this observation, we employ an autoencoder-based network to explore the potential correlations between sample features. The autoencoder can effectively learn data representations through unsupervised approach, which aligns closely with our goals. During training, the encoder compresses the high-dimensional data input from the passive party into a 1×100 low-dimensional representation and learns the intrinsic correlations between the sample features of the passive and active parties. Subsequently, the decoder learns how to reconstruct the active party's private data using learned correlations. The specific design details of the autoencoder can be found in Appendix A.

We summarize the steps in Algorithm 1 and provide a detailed description below. During the training data preparation phase, we first divide the samples in the auxiliary dataset X^{aux} into two parts X_a^{aux} and X_b^{aux} according to the feature division protocol of the VSL system. X_b^{aux} simulates the data of the active party that we want to reconstruct and is replaced by random noise N (line 1-2). Then we concatenate X_a^{aux} with N and feed the concatenated data into the autoencoder to generate initial reconstruction results $\hat{X}_{AE}^{aux}$ (line 3-5). The model is optimized by minimizing the mean square error (MSE) between $\hat{X}_{AE}^{aux}$ and the original data X^{aux} (line 6-8). This process can be formulated as:

$$\theta_{AE}^{*} = argmin_{\theta_{AE}} \mathcal{L}(AE((X_a^{aux}|N)), X^{aux}). \tag{2}$$

where AE represents the autoencoder and θ_{AE} is a set of parameters for it.

After completing the training of the autoencoder, we can input X_a into the trained model to obtain the initial reconstruction of the private data, denoted as $\hat{X}_{AE}$. This process can be conducted offline.

Algorithm 1: Obtain Initial Reconstruction via Partial Sample Features

Data: auxiliary dataset X^{aux}, total epochs E;
/* Initialize model */
AE is randomly initialized;
/* Data preparation */
1. Divide X^{aux} into X_a^{aux} and X_b^{aux};
2. Replace X_b^{aux} with noise N;
3. Concatenate X_a^{aux} with noise N;
/* training process of autoencoder */
4. while epoch $< E$:
5. $\quad \hat{X}_{AE}^{aux} \leftarrow AE((X_a^{aux}|N))$
6. $\quad \mathcal{L}_{AE} \leftarrow MSE(\hat{X}_{AE}^{aux}, X^{aux})$
7. $\quad \bigtriangledown_{AE} \leftarrow compute\_gradient(\theta_{AE}, \mathcal{L}_{AE})$
8. $\quad \theta'_{AE} \leftarrow update\_weight(\theta_{AE}, \bigtriangledown_{AE})$
9. end

3.3 Building Pseudo System

The VSL system enhances model effectiveness by leveraging knowledge from distributed data sources. Therefore, we believe that the gradients produced in the VSL training process inevitably contain knowledge about the private data of other participants. This provides us with a new insight: we can utilize the posterior information related to the active party's private data contained in the gradients to improve the accuracy of initial reconstruction results.

To leverage the posterior information about the private data contained in the gradients, we need to understand the patterns and behaviors involved in how the server generates these gradients. To achieve this, it is essential for us to construct a shadow VSL system that includes the passive party client model f_a, a pseudo active party client model f'_b, and a pseudo server model f'_{top}.

As shown in Algorithm 2, we first perform a normal round of VSL training to update the normal VSL system (line 4-11). Subsequently, we train the pseudo VSL system locally using auxiliary dataset, at which time it freezes the parameters of f_a and updates only the f'_b and f'_{top} (line 12-17). These two steps alternate until the training of the normal VSL system is complete.

In each round of alternating training, the shadow VSL system learns from the knowledge of the normal VSL system through f_a. This allows the parameters and behavior of the shadow VSL system to remain as consistent as possible with those of the normal VSL system, enabling the pseudo server f'_{top} to learn the gradient patterns generated by the actual server f_{top}.

Algorithm 2: Building Pseudo System

Data: private dataset (X, Y), auxiliary dataset (X^{aux}, Y^{aux}), total epochs E;
/* Initialize model */
Normal VSL system: f_a, f_b and f_{top} are randomly initialized;
Pseudo VSL system: $f_b^{'}$ and $f_{top}^{'}$ are randomly initialized;
/* Data preperation */
1. Divide X into X_a and X_b;
2. Divide X^{aux} into X_a^{aux} and X_b^{aux};
3. while epoch $< E$:
    /* Normal VSL training epoch */
4. $\hat{Y} \leftarrow f_{top}((f_a(X_a)|f_b(X_b)))$
5. $\mathcal{L}_{norm} \leftarrow CrossEntropy(\hat{Y}, Y)$
6. $\bigtriangledown_{f_{top}} \leftarrow compute\_gradient(\theta_{f_{top}}, \mathcal{L}_{norm})$
7. $\theta_{f_{top}}^{'} \leftarrow update\_weight(\theta_{f_{top}}, \bigtriangledown_{f_{top}})$
8. $\bigtriangledown_{f_a} \leftarrow compute\_gradient(\theta_{f_a}, \bigtriangledown_{f_a(X_a)})$
9. $\bigtriangledown_{f_b} \leftarrow compute\_gradient(\theta_{f_b}, \bigtriangledown_{f_b(X_b)})$
10. $\theta_{f_a}^{'} \leftarrow update\_weight(\theta_{f_a}, \bigtriangledown_{f_a})$
11. $\theta_{f_b}^{'} \leftarrow update\_weight(\theta_{f_b}, \bigtriangledown_{f_b})$
    /* Pseudo VSL training process */
12. $\hat{Y}^{aux} \leftarrow f_{top}^{'}((f_a(X_a^{aux})|f_b^{'}(X_b^{aux})))$
13. $\mathcal{L}_{pseudo} \leftarrow CrossEntropy(\hat{Y}^{aux}, Y^{aux})$
14. $\bigtriangledown_{f_{top}^{'}} \leftarrow compute\_gradient(\theta_{f_{top}^{'}}, \mathcal{L}_{pseudo})$
15. $\theta_{f_{top}^{'}}^{'} \leftarrow update\_weight(\theta_{f_{top}^{'}}, \bigtriangledown_{f_{top}^{'}})$
16. $\bigtriangledown_{f_b^{'}} \leftarrow compute\_gradient(\theta_{f_b^{'}}, \bigtriangledown_{f_b^{'}(X_b^{aux})}$
17. $\theta_{f_b^{'}}^{'} \leftarrow update\_weight(\theta_{f_b^{'}}, \bigtriangledown_{f_b^{'}})$
    /* The weight of f_a isn't update */
18. end

3.4 Optimizing Initial Reconstruction via Gradient

After mastering the representation pattern of privacy gradients, the next task is to extract the information contained in the gradients and use this information to enhance the quality of the initial reconstruction obtained in the first step. To achieve this, we introduce a model based on the U-Net [12] architecture. U-Net employs an encoder-decoder structure and utilizes skip connections to link the corresponding feature maps from the encoder to the decoder. We plan to improve the U-Net architecture so that it can simultaneously accept both the initial reconstruction and the gradients as inputs, allowing the initial reconstruction to serve as a good starting point for further optimizing the reconstruction results.

The structure of our U-Net is shown in Fig. 2. We designed an encoder for both the initial reconstruction and the gradients respectively. The encoder for the image is connected to the decoder, while the gradient maps generated by the gradient encoder are connected to the corresponding feature maps in the decoder through skip connections. This structure allows us to effectively

combine the initial reconstruction with the gradients, making full use of the privacy information contained in both, thereby enhancing the quality of the reconstruction results. The specific network details can be found in Appendix B.

An intuitive explanation is that the gradients transmitted from the server to the passive party aggregate privacy information from other participants. Through the gradient encoder in the U-Net architecture, we can effectively extract the latent feature information embedded in these gradients and integrate this information into the decoder's feature maps via hierarchical feature fusion. This enables layer-by-layer progressive refinement, ultimately optimizing the reconstructed image quality.

As shown in Algorithm 3, during training, we feed the initial reconstruction $\hat{X}_{AE}^{aux}$ of the auxiliary dataset (line 5) and the gradients $\bigtriangledown_{f_a(X_a^{aux})}$ returned by the pseudo server f'_{top} (line 6-8) into the U-Net for exact reconstruction $\hat{X}_U^{aux}$ (line 9). The goal is the same as the first step, which aims to minimize the MSE loss between the $\hat{X}_U^{aux}$ and the original data X^{aux} (line 10-12). The process can be formulated as follows:

$$\theta_U^* = argmin_{\theta_U} \mathcal{L}(U((\bigtriangledown_{f_a(X_a^{aux})} | \hat{X}_{AE}^{aux})), X^{aux}). \tag{3}$$

where U represents the U-Net and θ_U is a set of parameters for it.

When the U-Net network is trained, we inputs the normal gradient $\bigtriangledown_{f_a(X_a)}$ collected in the second step along with the initial reconstruction of the privacy data $\hat{X}_{AE}$ obtained in the first step to achieve an more accurate reconstruction $\hat{X}_U$.

4 Experiments

4.1 Experimental Setup

1) Datasets and Tasks: We evaluate the proposed attack methods on the object dataset CIFAR-10 [13] and the facial dataset CelebA [14]. Both datasets are split into training, testing, and the attacker's auxiliary sets in a 6:1:3 ratio, ensuring a balanced distribution of categories in each subset. Additionally, we make sure that there is no identity overlap within the CelebA subset. We also incorporate more complex datasets, specifically Tiny-ImageNet [15] and LFW [16], to assess the effectiveness and broader applicability of our proposed attacks. Furthermore, the CINIC-10 [17] dataset and the FFHQ [18] dataset will be used for ablation experiments involving non-iid assumption, ensuring that there are no overlapping samples between CINIC-10 and CIFAR-10. The tasks include classifying the object and determining whether the facial attributes in the dataset are attractive through binary classification.

2) Model Architectures: We built our VSL system based on VGG16 [19] and ResNet18 [20] network architectures, with VGG16 used for classification on object datasets and ResNet18 used for classification on facial datasets. In addition, we tested our attack on Vision Transformer. We used the ViT/B-16 [21] model to classify the CIFAR-10 dataset. In the ablation experiments,

Algorithm 3: Optimizing Initial Reconstruction via Gradient

Data: auxiliary dataset X^{aux}, total epochs E;
/* Initialize model */
AE is trained in Algorithm 1;
Pseudo VSL system: f_a, f_b' and f_{top}' is trained in Algorithm 2;
U is randomly initialized;
/* Data preperation */
1. Divide X^{aux} into X_a^{aux} and X_b^{aux};
2. Replace X_b^{aux} with noise N;
3. Concatenate X_a^{aux} with noise N;
/* training process of U-Net */
4. while epoch $< E$:
    /* Get the initial reconstruction from AE */
5. $\hat{X}_{AE}^{aux} \leftarrow AE((X_a^{aux}|N))$
    /* Get gradient from f_{top}' */
6. $\hat{Y}^{aux} \leftarrow f_{top}^{'}((f_a(X_a^{aux})|f_b^{'}(X_b^{aux})))$
7. $\mathcal{L}_{pseudo} \leftarrow CrossEntropy(\hat{Y}^{aux}, Y^{aux})$
8. $\bigtriangledown_{f_{top}'} \leftarrow compute\_gradient(\theta_{f_{top}'}, \mathcal{L}_{pseudo})$
    /* Refine initial reconstruction $\hat{X}_{AE}^{aux}$ via gradient $\bigtriangledown_{f_a(X_a^{aux})}$ */
9. $\hat{X}_U^{aux} \leftarrow U((\hat{X}_{AE}^{aux}|\bigtriangledown_{f_a(X_a^{aux})}))$
10. $\mathcal{L}_U \leftarrow MSE(\hat{X}_U^{aux}, X^{aux})$
11. $\bigtriangledown_U \leftarrow compute\_gradient(\theta_U, \mathcal{L}_U)$
12. $\theta_U^{'} \leftarrow update\_weight(\theta_U, \bigtriangledown_U)$
13. end

we introduced the MobileNet [22] architecture as a replacement for the server model. Different split points were tested to assess the impact of model depth on the results. The detailed model structure is presented in Appendix C.

3) Evaluation Metrics: We adopted structural similarity index (SSIM) [23] and learned perceptual image patch similarity (LPIPS) [24] to evaluate the quality of reconstructed images. A higher SSIM value, closer to 1, indicates greater similarity between reconstructed and original images. Conversely, LPIPS measures perceptual similarity, with smaller values indicating a smaller visual difference between reconstructed and original images.

4.2 Visuality Evaluation

Fig. 3 shows the reconstruction results of the autoencoder and the optimization results of the U-Net using gradients obtained from different split points. As shown in Fig. 3, the reconstruction performance of U-Net is significantly better than that of the autoencoder, allowing for a better restoration of the details of the privacy data from the active party. In addition, at different depth of split points, the reconstruction results of U-Net show good robustness. As the split points go deeper, the reconstruction performance experiences only a slight decline. We believe that U-Net achieves better reconstruction results primarily

due to the introduction of gradients, as the privacy information contained in the gradients helps enhance the reconstruction effect.

The quantitative results in Table 1 further support our conclusion. For CIFAR-10, the average SSIM and LPIPS values of the autoencoder's reconstruction results are 0.5132 and 0.3976, respectively; for CelebA, these values are 0.5843 and 0.3481. The average SSIM and LPIPS values for U-Net at block 1 are 0.5071 and 0.3395 for CIFAR-10, and 0.5877 and 0.2771 for CelebA. Even at the cut point of block 4, the LPIPS metrics of U-Net's reconstruction results improve by 0.0518 and 0.07 compared to the autoencoder for CIFAR-10 and CelebA, respectively. The metrics in the table consistently indicate that the attack method we proposed is quite robust.

Fig. 3: Reconstruction results on CIFAR-10 and CelebA in different split settings.

Table 1: SSIM and LPIPS results in different split settings.

Dataset	Model	Split Point	SSIM↑	LPIPS↓	Dataset	Model	Split Point	SSIM↑	LPIPS↓
CIFAR10	AE		**0.5132**	0.3976	CelebA	AE		0.5843	0.3481
	UNet	block1	0.5071	**0.3395**		UNet	block1	**0.5877**	**0.2771**
		block2	0.5033	0.3448			block2	0.5865	0.2777
		block3	0.5024	0.3456			block3	0.5853	0.2781
		block4	0.5013	0.3458			block4	0.5835	0.2781

4.3 Effect of Auxiliary Dataset

We investigated the impact of the distribution of auxiliary datasets on data reconstruction attacks. The experiments were conducted on a VSL system with split point at block2, using CINIC-10 and FFHQ as non-iid auxiliary datasets for CIFAR-10 and CelebA respectively. Fig. 4 shows the reconstruction results and Table 2 provides a quantitative comparison.

We observed that the quality of reconstruction results for CIFAR-10 and CelebA slightly deteriorated when using non-iid datasets. For CIFAR-10, the drop in reconstruction quality was relatively small because the distribution difference between CINIC-10 and CIFAR-10 is minor. In contrast, FFHQ has a much larger distribution difference from CelebA. Unlike CelebA, which contains face images with various angles, FFHQ predominantly features frontal face images. Furthermore, CelebA has more complex backgrounds, with faces occupying a smaller proportion of the images, while FFHQ has simpler backgrounds with faces occupying a larger proportion. Nevertheless, the attack we proposed is still able to recover most of the facial information, including pose, hairstyle and expression.

Fig. 4: Data reconstruction results of non-iid settings.

Table 2: SSIM and LPIPS results of no-iid settings.

Metric	CIFAR10		CelebA		Metric	CIFAR10		CelebA	
SSIM↑	Same	Different	Same	Different	LPIPS↓	Same	Different	Same	Different
Autoencoder	0.5132	0.5077	0.5843	0.4602	Autoencoder	0.3976	0.4014	0.3481	0.4493
U-Net	0.5033	0.4877	0.5865	0.4821	U-Net	0.3448	0.3629	0.2777	0.4075

4.4 Effect of Substitute Server Structure

We also investigated the impact of different server architectures on reconstruction attacks. We used the MobileNet architecture to replace VGG16 and ResNet18

as the server model for the pseudo VSL system. In this experiment, we set the split layer to block2. Fig. 5 shows the reconstructed images while using different server model architectures and Table 3 presents the quantitative results of the SSIM and LPIPS metrics.

Overall, the different server model architectures have little impact on the reconstruction results of our attack methods. In terms of visual effects, the reconstruction results of U-Net in Fig. 3 and Fig. 5 are almost identical and the metrics experienced only slight fluctuations. This is because, despite the different model architectures, the learning objectives remain consistent. As a result, the optimization directions guided by the gradients are consistent, which leads to good reconstruction results.

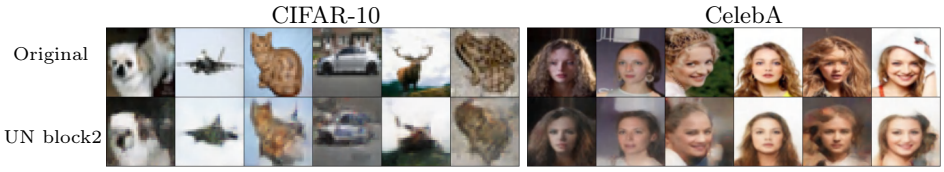

Fig. 5: Reconstruction results of different server models.

Table 3: SSIM and LPIPS results of different server models.

Metric	CIFAR10		CelebA	
	Same	Different	Same	Different
SSIM↑	0.5033	0.4981	0.5865	0.5827
LPIPS↓	0.3448	0.3449	0.2777	0.2787

4.5 Effectiveness on Complex Dataset and Model

We conducted extended experiments using the Tiny-Imagenet and LFW datasets. Tiny-Imagenet includes 200 categories of different objects, with around 500 images in each category. The LFW dataset contains over 13,000 labeled face images. We divided the datasets and launched the attack on ResNet18 with split point at block2 in the same way as in our main experiments. As shown in Fig. 6a, our attack demonstrates good visual results on both datasets, sufficient to reveal the privacy data of the active party. This proves that our attack is effective in complex datasets and even in real-world scenarios.

In practical applications, edge devices typically maintain only a shallower model to achieve the highest end-to-cloud workload while offloading complex

computations to the cloud. But in order to explore the effectiveness of our attack method on complex and novel model architectures, we conducted attack experiments at a deeper splitting point after the 8th transformer block of the ViT-B/16 model on CIFAR-10 dataset. As shown in Fig. 6b, even with more complex model structures, we still achieved good attack results, with SSIM = 0.4940 and LPIPS = 0.3439, demonstrating that our attack method is model-agnostic.

(a) Attack results on complex datasets. (b) Attack results on complex model.

Fig. 6: Reconstruction results on complex datasets and model.

4.6 Defense Discuss

Our attack is difficult to defend against because it does not interfere with the normal VSL training process and is transparent to the active party, making it hard to detect. Furthermore, there is currently no research exploring the possibility of the passive party stealing data from the active party like us. In VSL systems, inference tasks are initiated by the vigilant-challenged active party, which typically does not want to implement defenses that could affect model performance.

5 Conclusion

In this paper, we make the first attempt to investigate the issue of passive parties stealing data from active parties in the VSL scenario and propose corresponding attack methods. We exploit the prior information contained in the partial sample features held by the passive party using an autoencoder to obtain an initial reconstruction. At the same time, we synchronously construct a pseudo VSL system during the training process to learn the gradient patterns generated by the server. Subsequently, we use an improved U-Net to leverage the posterior information about the private data embedded in the gradients to optimize the initial reconstruction, resulting in more accurate reconstruction outcomes. We validate the effectiveness of our attack methods through extensive experiments. We hope our work will draw attention to the security of VSL and encourage a reassessment of data security within VSL. All of our supplementary materials, including code, appendices, and supplemental experiments, can be found at: https://github.com/yxliu42/VSL.

References

1. Vepakomma, P., Gupta, O., Swedish, T., Raskar R.: Split learning for health: Distributed deep learning without sharing raw patient data. arXiv preprint arXiv:1812.00564 (2018)
2. Gupta, O., Raskartitle, R.: Distributed learning of deep neural network over multiple agents. Journal of Network and Computer Applications **116**, 1-8 (2018)
3. Li, P., Guo, C., Xing, Y., et al.: Core network traffic prediction based on vertical federated learning and split learning. Scientific Reports **14**(1), 4663 (2024)
4. Allaart, C.G., Keyser, B., Bal, H., et al.: Vertical Split Learning - an exploration of predictive performance in medical and other use cases. In: 2022 International Joint Conference on Neural Networks (IJCNN) on Proceedings, pp. 1-8. IEEE (2022)
5. Ads, O.S., Alfares, M.M., Salem, M.A.M.: Multi-limb Split Learning for Tumor Classification on Vertically Distributed Data. In: 2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS) on Proceedings, pp. 88-92. IEEE (2021)
6. Ezzeddine, F., Ayoub, O., Andreoletti, D., et al.: Vertical Split Learning-Based Identification and Explainable Deep Learning-Based Localization of Failures in Multi-Domain NFV Systems. In: 2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) on Proceedings, pp. 46-52. IEEE (2023)
7. Yang, M., Li, Z., Wang, J., et al.: Measuring Data Reconstruction Defenses in Collaborative Inference Systems. In: the 2022 Neural Information Processing Systems (NIPS) on Proceedings, pp. 12855-12867. Curran Associates, Inc. (2022)
8. Li, Z., Yang, M., Liu, Y., et al.: GAN You See Me? Enhanced Data Reconstruction Attacks against Split Inference. In: 2023 Neural Information Processing Systems on Proceedings, pp. 54554-54566. Curran Associates, Inc. (2023)
9. Xu, X., Yang, M., Yi, W., et al.: A Stealthy Wrongdoer: Feature-Oriented Reconstruction Attack against Split Learning. In: the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) on Proceedings, pp. 12130-12139. (2024)
10. Chen, H., Fu, C., Ruan, N.: Steal from Collaboration: Spy Attack by a Dishonest Party in Vertical Federated Learning. In: 2023 International Conference on Applied Cryptography and Network Security on Proceedings, pp. 583-604. Springer Nature Switzerland (2023)
11. He, Z., Zhang, T., Lee, R.B.: Model inversion attacks against collaborative inference. In: the 35th Annual Computer Security Applications Conference (ACSAC) on Proceedings, pp. 148-162. Association for Computing Machinery, New York, NY, USA (2019)
12. Ronneberger, O., Fischer, P, Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: 2015 Medical Image Computing and Computer-Assisted Intervention (MICCAI) on Proceedings, pp. 234-241. Springer International Publishing (2015)
13. Krizhevsky, A., Hinton, G.: Learning Multiple Layers of Features from Tiny Images. (2009)
14. Liu, Z., Luo, P., Wang, X., et al.: Deep Learning Face Attributes in the Wild. In: 2015 IEEE International Conference on Computer Vision (ICCV) (2015)
15. Wu, J., Zhang, Q., Xu, G.: Tiny imagenet challenge. Technical report. (2017)
16. Huang, G.B., Mattar, M., Berg, T., et al.: Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. In: the 2007

IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) on Proceedings, pp. 1-8. (2007)
17. Darlow, L.N., Crowley, E.J., Antoniou, A., et al.: Cinic-10 is not imagenet or cifar-10. arXiv preprint arXiv:1810.03505 (2018)
18. Karras, T., Laine, S., Aila, T.: A Style-Based Generator Architecture for Generative Adversarial Networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
20. He, K., Zhang, X., Ren, S., et al.: Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
21. Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
22. Howard, A., Zhu, M., Chen, B., et al.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861 (2017)
23. Wang, Z., Bovik, A.C., Sheikh, H.R., et al.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612, (2004)
24. Zhang, R., Isola, P., Efros, A.A., et al.: The unreasonable effectiveness of deep features as a perceptual metric. In: the 2018 IEEE conference on computer vision and pattern recognition on Proceedings, pp. 586-595. (2018)