INFOSCISSORS: DEFENSE AGAINST DATA LEAKAGE IN COLLABORATIVE INFERENCE THROUGH THE LENS OF MUTUAL INFORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Edge-cloud collaborative inference empowers resource-limited IoT devices to support deep learning applications without disclosing their raw data to the cloud server, thus protecting user's data. Nevertheless, prior research has shown that collaborative inference still results in the exposure of input and predictions from edge devices. To defend against such data leakage in collaborative inference, we introduce InfoScissors, a defense strategy designed to reduce the mutual information between a model's intermediate outcomes and the device's input and predictions. We evaluate our defense on several datasets in the context of diverse attacks and offer a theoretical robustness guarantee.

1 INTRODUCTION

Edge devices are becoming smarter and more versatile. These devices are expected to efficiently perform a wide range of deep learning (DL) inference tasks with remarkable performance. However, implementing DL inference applications on such edge devices is challenging due to the constraints imposed by the on-device resource availability. As we see the rise of state-of-the-art (SOTA) DL models, such as Large Language Models (Wei et al., 2022; Kasneci et al., 2023), they are becoming increasingly complex, housing a colossal number of parameters. This escalation in complexity and size makes it difficult to store a DL model on an edge device, which typically has limited memory space. Furthermore, the restricted computational resources could lead to prolonged latency during inference. One potential solution to this predicament is to transmit the input data directly from the edge device to a cloud server. The server, which houses the DL model, then conducts inference and sends the prediction back to the device. However, this approach carries a risk of data leakage, particularly if the input data are sensitive in nature - such as facial images. In addition, the output data (i.e., predictions) can also contain confidential information, such as the patient's diagnostic results.

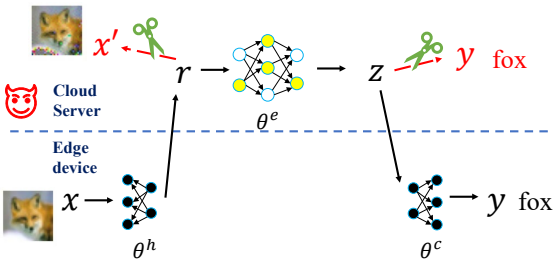


Figure 1: A general framework of collaborative inference. The malicious server can infer input and predictions on the edge device. Our method defends against data leakage by reducing the mutual information between the model's intermediate outcomes and the edge device's data and predictions.

Collaborative inference (Li et al., 2018; Kang et al., 2017; Eshratifar et al., 2019; Banitalebi-Dehkordi et al., 2021; Li et al., 2021a; Shlezinger et al., 2021; Zhou et al., 2021) has emerged as an approach to prevent data leakage when deploying DL inference applications on commodity edge devices with constrained computing resources. Fig. 1 shows a general collaborative inference system. Suppose an edge device and a cloud server conduct collaborative inference. The deep learning model can

be divided into three parts[1]. The first and last few layers of the network are deployed on the edge device, while the remaining layers are offloaded to the cloud server. This division allows most of the computational tasks to be handled by the server, effectively mitigating the resource limitations on the device. The edge device and the cloud server communicate only the intermediate outputs of the model, ensuring that the raw input and predictions remain inaccessible to the server. However, recent works (He et al., 2019; 2020) have revealed that sharing these intermediate outputs can still lead to data leakage from edge devices, including input data and predictions. A malicious server can, for instance, reconstruct input data from the representations (i.e., r in Fig. 1) uploaded by the device through Model Inversion (MI) attacks (Zhu et al., 2019; Zhao et al., 2020; He et al., 2020). Furthermore, the high-level features (i.e., z in Fig. 1) contain rich information about the predictions, making it feasible for a malicious server to infer the device's predictions through these features (Fu et al., 2022b; Li et al., 2021b; Liu et al., 2021). While there have been considerable explorations into data protection in collaborative inference (He et al., 2019; 2020; Wang et al., 2021; Zou et al., 2023), existing defenses tend to significantly degrade model utility. This degradation is particularly evident in scenarios where attacks are relatively strong. For example, when the head model on the device (i.e., θ^h in Fig. 1) is shallow, existing defenses (He et al., 2019; 2020; Oh et al., 2022; Fu et al., 2022a;a; Zou et al., 2023) cannot guarantee robustness against MI attacks without a significant drop in model accuracy as shown in our results.

We propose InfoScissors, a defense method designed from a mutual information perspective to protect the edge device's data in collaborative inference. This approach works by protecting both the device's input data and its predictions. To protect the input data, we regularize the head model on the device to extract representations that contain less mutual information with the input. To protect the prediction, we regularize the features extracted by the server's encoder to minimize the mutual information they contain with the label. We derive a variational mutual information upper-bound and develop an adversarial training method to minimize this bound on the device side. Our defense's robustness is theoretically guaranteed. We evaluate our method on CIFAR10 and CIFAR100 against input leakage using both black-box and white-box MI attacks. The results show that our method can effectively defend the attacks with less than a 3% drop in model accuracy even when the head model on the device has only one convolutional layer, where the attacks are extremely strong. We also evaluate our defense against prediction leakage using multiple Model Completion (MC) attacks (Fu et al., 2022b; Li et al., 2021b). The results show that our defense achieves the best trade-off between the model accuracy and the defense effectiveness compared to the baselines.

Our contributions are summarized as follows:

- To the best of our knowledge, this is the first paper to systematically defend against data leakage in collaborative inference, encompassing both input leakage and prediction leakage.
- We propose InfoScissors, a defense method against data leakage in collaborative inference from the mutual information perspective. We offer a theoretical robustness guarantee of our defense against input recovery attacks and prediction inference attacks.
- We empirically evaluate InfoScissors across multiple datasets and against multiple attacks. The results show that our method effectively defends against MI and MC attacks, outperforming the baselines.

2 Related Work

2.1 Data Leakage in Collaborative Inference

Data leakage is drawing more and more attention as the rapid growth of commercial deployment of DL, especially in collaborative learning scenarios, whose primary concern is data safety. In collaborative inference, we categorize data leakage into two types, i.e., input leakage (Luo et al., 2021; He et al., 2019; Jiang et al., 2022; Jin et al., 2021) and prediction leakage (Fu et al., 2022b; Li et al., 2021b; Liu et al., 2021). For input leakage, Luo et al. (2021) proposes general attack methods for complex models, such as Neural Networks, by matching the correlation between adversary features and target features, which can be seen as a variant of model inversion (Fredrikson et al.,

[1]Note that some applications might divide the model into two parts, and the edge devices might hold the first or the last few layers, which have different data leakage problems. This paper considers the general setting.

2015; Sun et al., 2021). He et al. (2019); Geiping et al. (2020); Jin et al. (2021); Yin et al. (2021); Melis et al. (2019); Jiang et al. (2022) also propose variants of model inversion attack. While all these attacks are in the inference phase, Jin et al. (2021) proposes a variant of DLG (Zhu et al., 2019), which can perform attacks in the training phase. For prediction leakage, Li et al. (2021b) proposes an attack and defense method for two-party split learning on binary classification problems, a special collaborative inference setting. Additionally, Fu et al. (2022b) proposes three different label inference attack methods considering different settings in collaborative inference: direct label inference attack, passive label inference attack, and active label inference attack.

2.2 Defense in Collaborative Inference

Defensive methods have been proposed against data leakage in collaborative inference. To defend against input leakage, some works apply differential privacy (DP) (He et al., 2019; 2020; Oh et al., 2022) and compression (He et al., 2019; 2020; Wang et al., 2021; Singh et al., 2021) to the representations and models. While these methods can successfully defend against input leakage from the representations, they cause substantial model performance degradation because they weaken the knowledge/information in the representations. Some recent works also try to prevent input leakage by regularizing the representations from the mutual information perspective (Mireshghallah et al., 2020; Wang et al., 2021; Zou et al., 2023). However, their methods only achieve decent results when the head model on the edge device is deep, which is not practical when the computation power is constrained on the edge device. Some other works (Makhdoumi et al., 2014; Rassouli & Gündüz, 2019; Wu et al., 2022) apply mutual information on input space to protect input data, but their methods are only feasible with limited input dimension. Mireshghallah et al. (2021) applies mutual information on image space but only protects data from human perception and cannot guarantee robustness against advanced attacks. To defend against prediction leakage, Liu et al. (2021) manipulates the labels following specific rules to defend the direct label inference attack, which can be seen as a variant of label differential privacy (label DP) (Chaudhuri & Hsu, 2011; Ghazi et al., 2021) in collaborative inference. Compression and quantization of the gradients (Fu et al., 2022b; Zou et al., 2023) are also applied to prevent prediction leakage. However, similar to the defense against data leakage, these defenses cause substantial model performance degradation to achieve decent defense performance.

3 Preliminary

3.1 Collaborative Inference Setting

Suppose an edge device and a cloud server conduct collaborative inference. Following the setting in Fig. 1, the deep learning model is divided into a head model $f^h_{\theta^h}$, an encoder $f^e_{\theta^e}$ and a classifier $f^c_{\theta^c}$. The head model and classifier are deployed on the edge device, and the encoder is on the cloud server. Given an input x_i, the edge device first calculates the representation $r_i = f^h_{\theta^h}(x_i)$ and sends r_i to the server. Then the server extracts the feature from the received representation $z_i = f^e_{\theta^e}(r_i)$ and sends z_i back to the edge device. After receiving the feature, the edge device calculates the prediction $\hat{y}_i = f^c_{\theta^c}(z_i)$. In this paper, the results of $f^h_{\theta^h}$ sent from the device to the server are referred to as *representations*, and *features* refer to the results of $f^e_{\theta^e}$ sent from the server to the device. The overall inference procedure is formulated as:

$$\hat{y}_i = f^c_{\theta^c}(f^e_{\theta^e}(f^h_{\theta^h}(x_i))). \tag{1}$$

In the real world, the input x_i and prediction $\hat{y}_i$ are important intellectual properties of the edge device and may contain personal information. In the inference procedure, the edge device does not send raw input to the server, and the inference results are also inaccessible to the server.

3.2 Threat Model

Our goal is to protect the edge device's input and predictions from being inferred by the cloud server. The device only uploads the representations to the server and never leaks raw input or predictions to the server. However, the cloud server is untrusted, attempting to steal input and predictions. We assume the untrusted server strictly follows the collaborative inference protocols, and it cannot compromise the inference process conducted by the device. With the received representation r_i, the server can reconstruct the input x_i on the device by conducting MI attacks (Zhu et al., 2019; Zhao et al., 2020;

He et al., 2019). Notably, the head model on the device is usually shallow due to the computation resource limitation, which aggravates the input leakage from the representation (He et al., 2020). The encoder on the server extracts high-level features containing rich information about the prediction, which enables the server to infer predictions of the device. We conduct preliminary experiments to illustrate the data leakage in collaborative inference, which can be found in Appendix A.

4 METHOD

4.1 DEFENSE FORMULATION

To defend against data leakage, we propose InfoScissors, a learning algorithm that regularizes the model during the training phase. Following the setup of 3.1, suppose the edge device has sample pairs $\{(x_i, y_i)\}_{i=1}^{N}$ drawn from a distribution $p(\mathrm{x}, \mathrm{y})$. The representation is calculated as $\mathrm{r} = f_{\theta^h}^{h}(\mathrm{x})$ by the edge device, and the cloud server computes features $\mathrm{z} = f_{\theta^e}^{e}(\mathrm{r})$. We apply $\mathrm{x}, \mathrm{y}, \mathrm{r}, \mathrm{z}$ here to represent random variables, while x_i, y_i, r_i, z_i are deterministic values. To defend against the leakage of the edge device's input data and inference results, InfoScissors is designed to achieve three goals:

- Goal 1: To preserve the performance of collaborative inference, the main objective loss should be minimized.
- Goal 2: To prevent the input leakage from the representations, θ^h should not extract representations r containing much information about the input data x.
- Goal 3: To reduce the leakage of the predictions on the edge device, θ^e on the cloud server should not be able to extract features z containing much information about the true label y.

Formally, we have three training objectives:

$$\begin{aligned}
&\textbf{Prediction: } \min_{\theta^h, \theta^e, \theta^c} \mathcal{L}\left(f_{\theta^c}^{c}\left(f_{\theta^e}^{e}\left(f_{\theta^h}^{h}(\mathrm{x})\right)\right), \mathrm{y}\right), \\
&\textbf{Input protection: } \min_{\theta^h} \mathrm{I}(\mathrm{r}; \mathrm{x}), \\
&\textbf{Prediction protection: } \min_{\theta^h, \theta^e} \mathrm{I}(\mathrm{z}; \mathrm{y}),
\end{aligned} \tag{2}$$

where $\mathrm{I}(\mathrm{r}; \mathrm{x})$ is the mutual information between the representation and the input, which indicates how much information r retains about the input data x. Similarly, $\mathrm{I}(\mathrm{z}; \mathrm{y})$ is the mutual information between the feature and the label. We minimize these mutual information terms to prevent the cloud server from inferring the input x and label y from r and z, respectively.

The prediction objective is usually easy to optimize (e.g., cross-entropy loss for classification). However, the mutual information terms are hard to calculate in practice for two reasons: 1. r and x are high-dimensional, and it is extremely computationally heavy to compute their joint distribution; 2. Calculating the mutual information requires knowing the distributions p(x|r) and p(y|z), which are both difficult to compute. To derive tractable estimations of the mutual information objectives, we leverage CLUB(Cheng et al., 2020) to formulate variational upper-bounds of mutual information terms. We first formulate a variational upper-bound of $\mathrm{I}(\mathrm{r}; \mathrm{x})$:

$$\begin{aligned}
&\mathrm{I}(\mathrm{r}; \mathrm{x}) \\
\leq\ &\mathrm{I}_{\text{vCLUB}}(\mathrm{r}; \mathrm{x}) \\
:=\ &\mathbb{E}_{p(\mathrm{r},\mathrm{x})} \log q_\psi(\mathrm{x}|\mathrm{r}) - \mathbb{E}_{p(\mathrm{r})p(\mathrm{x})} \log q_\psi(\mathrm{x}|\mathrm{r}),
\end{aligned} \tag{3}$$

where $q_\psi(\mathrm{x}|\mathrm{r})$ is a variational distribution with parameters ψ to approximate $p(\mathrm{x}|\mathrm{r})$. To guarantee the inequality of Eq. (3), $q_\psi(\mathrm{x}|\mathrm{r})$ should satisfy:

$$\mathrm{KL}\left(p(\mathrm{r}, \mathrm{x}) \,||q_\psi(\mathrm{r}, \mathrm{x})\right) \leq \mathrm{KL}\left(p(\mathrm{r})\, p(\mathrm{x}) \,||q_\psi(\mathrm{r}, \mathrm{x})\right), \tag{4}$$

which can be achieved by minimizing $\mathrm{KL}\left(p(\mathrm{r}, \mathrm{x}) \,||q_\psi(\mathrm{r}, \mathrm{x})\right)$:

$$\begin{aligned}
\psi &= \arg\min_{\psi} \mathrm{KL}\left(p(\mathrm{r}, \mathrm{x}) \,||q_\psi(\mathrm{r}, \mathrm{x})\right) \\
&= \arg\min_{\psi} \mathbb{E}_{p(\mathrm{r},\mathrm{x})}\left[\log\left(p(\mathrm{x}|\mathrm{r})\, p(\mathrm{r})\right) - \log\left(q_\psi(\mathrm{x}|\mathrm{r})\, p(\mathrm{r})\right)\right] \\
&= \arg\max_{\psi} \mathbb{E}_{p(\mathrm{r},\mathrm{x})} \log\left(q_\psi(\mathrm{x}|\mathrm{r})\right).
\end{aligned} \tag{5}$$

With sample pairs $\{(x_i, y_i)\}_{i=1}^N$, we apply the sampled vCLUB (vCLUB-S) mutual information estimator in Cheng et al. (2020) to reduce the computational overhead, which is an unbiased estimator of $\mathrm{I}_{\mathrm{vCLUB}}$ and is formulated as:

$$\hat{\mathrm{I}}_{\text{vCLUB-S}}(\mathrm{r};\mathrm{x}) = \frac{1}{N}\sum_{i=1}^{N}\left[\log q_\psi\left(x_i|r_i\right) - \log q_\psi\left(x_{k_i'}|r_i\right)\right], \tag{6}$$

where k_i' is uniformly sampled from indices $\{1, ..., N\}$. With Eq. (3), Eq. (5) and Eq. (6), the objective of input protection is formulated as:

$$\begin{aligned}&\min_{\theta^h} \mathrm{I}(\mathrm{r};\mathrm{x}) \Leftrightarrow \min_{\theta^h} \hat{\mathrm{I}}_{\text{vCLUB-S}}(\mathrm{r};\mathrm{x})\\ =\ &\min_{\theta^h} \frac{1}{N}\sum_{i=1}^{N}\left[\max_{\psi}\log q_\psi\left(x_i|r_i\right) - \log q_\psi\left(x_{k_i'}|r_i\right)\right].\end{aligned} \tag{7}$$

Similarly, we can use a variational distribution $q_\phi(y|z)$ with parameter ϕ to approximate $p(y|z)$, and formulate the objective of label protection as:

$$\begin{aligned}&\min_{\theta^h,\theta^e} \mathrm{I}(\mathrm{z};\mathrm{y}) \Leftrightarrow \min_{\theta^h,\theta^e} \hat{\mathrm{I}}_{\text{vCLUB-S}}(\mathrm{z};\mathrm{y})\\ =\ &\min_{\theta^h,\theta^e} \frac{1}{N}\sum_{i=1}^{N}\left[\max_{\phi}\log q_\phi\left(y_i|z_i\right) - \log q_\phi\left(y_{n_i'}|z_i\right)\right].\end{aligned} \tag{8}$$

Suppose we use g_ψ, h_ϕ to parameterize q_ψ and q_ϕ, respectively. By combining Eq. (7), Eq. (8) and the prediction objective with weight hyper-parameters λ_d and λ_l, the overall optimizing objective is:

$$\begin{aligned}&\min_{\theta^h,\theta^e,\theta^c}(1-\lambda_d-\lambda_l)\underbrace{\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(f_{\theta^c}^c\left(f_{\theta^e}^e\left(f_{\theta^h}^h(x_i)\right)\right), y_i\right)}_{\mathcal{L}_c}\\ &+\min_{\theta^h}\max_{\psi}\lambda_d\underbrace{\frac{1}{N}\sum_{i=1}^{N}\log g_\psi\left(x_i|f_{\theta^h}^h(x_i)\right)}_{\mathcal{L}_{d\_a}}+\min_{\theta^h}\lambda_d\underbrace{\frac{1}{N}\sum_{i=1}^{N}-\log g_\psi\left(x_{k_i'}|f_{\theta^h}^h(x_i)\right)}_{\mathcal{L}_{d\_r}}\\ &+\min_{\theta^h,\theta^e}\max_{\phi}\lambda_l\underbrace{\frac{1}{N}\sum_{i=1}^{N}\log h_\phi\left(y_i|f_{\theta^e}^e\left(f_{\theta^h}^h(x_i)\right)\right)}_{\mathcal{L}_{l\_a}}+\min_{\theta^h,\theta^e}\lambda_l\underbrace{\frac{1}{N}\sum_{i=1}^{N}-\log h_\phi\left(y_{n_i'}|f_{\theta^e}^e\left(f_{\theta^h}^h(x_i)\right)\right)}_{\mathcal{L}_{l\_r}}.\end{aligned} \tag{9}$$

h_ϕ can be easily constructed to estimate $p\,(\mathrm{y}|\mathrm{z})$ given the task of inference (e.g., classifier for classification task). To estimate $p\,(\mathrm{x}|\mathrm{r})$, we assume that x follows the Gaussian distribution of which the mean vector is determined by r and the variance is 1. Under this assumption, we apply a generator g_ψ to estimate the mean vector of x given r.

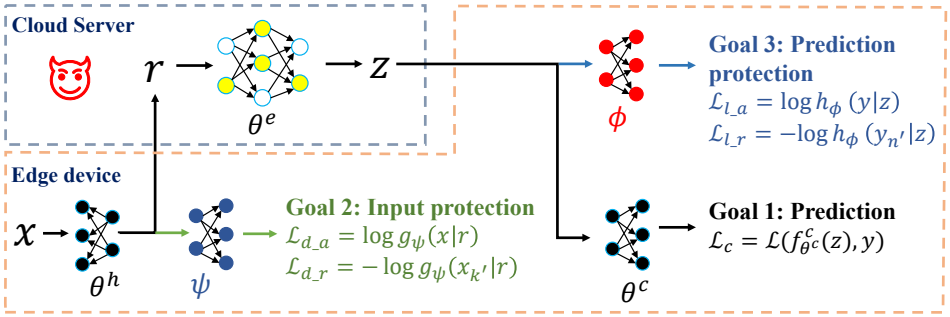


Figure 2: An overview of InfoScissors. Training step 1: Optimize the classifiers θ^c and ϕ by minimizing $\mathcal{L}_c$ and maximizing $\mathcal{L}_{l\_a}$, respectively. Step 2: Optimize the generator ψ by maximizing $\mathcal{L}_{d\_a}$. Step 3: Optimize θ^h and θ^e by minimizing $(1-\lambda_d-\lambda_l)\mathcal{L}_c+\lambda_l\mathcal{L}_{l\_a}+\lambda_l\mathcal{L}_{l\_r}+\lambda_d\mathcal{L}_{d\_a}+\lambda_d\mathcal{L}_{d\_r}$.

4.2 LEARNING ALGORITHM

The overall objective has five terms. For simplicity, we denote these five objective terms as $\mathcal{L}_c$, $\mathcal{L}_{d\_a}$, $\mathcal{L}_{d\_r}$, $\mathcal{L}_{l\_a}$ and $\mathcal{L}_{l\_r}$, respectively, as shown in Eq. (9). $\mathcal{L}_c$ is the prediction objective. $\mathcal{L}_{d\_a}$ and $\mathcal{L}_{d\_r}$ comprise the input data protection objective. $\mathcal{L}_{d\_a}$ is an adversarial training objective where an auxiliary generator g_ψ is trained to capture input information while the head layers $f^h_{\theta^h}$ are trained to extract as little input information as possible. $\mathcal{L}_{d\_r}$ regularizes $f^h_{\theta^h}$ to extract representations that can be used to generate randomly picked samples. $\mathcal{L}_{l\_a}$ and $\mathcal{L}_{l\_r}$ have similar effect with $\mathcal{L}_{d\_a}$ and $\mathcal{L}_{d\_r}$, respectively. We can reorganize the overall training objective as:

$$\theta^h, \theta^e, \theta^c, \psi, \phi = \arg\min_{\theta^h, \theta^e} \left[(1 - \lambda_d - \lambda_l) \min_{\theta^c} \mathcal{L}_c + \lambda_l \max_{\phi} \mathcal{L}_{l\_a} + \lambda_l \mathcal{L}_{l\_r} + \lambda_d \max_{\psi} \mathcal{L}_{d\_a} + \lambda_d \mathcal{L}_{d\_r} \right]. \tag{10}$$

Based on Eq. (10), we develop a collaborative learning algorithm. For each batch of data, the device first optimizes the classifiers θ^c and ϕ by minimizing $\mathcal{L}_c$ and maximizing $\mathcal{L}_{l\_a}$, respectively. Then, the device optimizes the generator ψ by maximizing $\mathcal{L}_{d\_a}$. Finally, θ^h and θ^e are optimized by minimizing $(1 - \lambda_d - \lambda_l)\mathcal{L}_c + \lambda_l \mathcal{L}_{l\_a} + \lambda_l \mathcal{L}_{l\_r} + \lambda_d \mathcal{L}_{d\_a} + \lambda_d \mathcal{L}_{d\_r}$. The detailed algorithm can be found in Appendix B. Note that θ^h, θ^c, ψ, and ϕ are deployed on devices, and their training does not need additional information from the cloud server compared with training without our defense. The training procedure of θ^e does not change, which makes our defense concealed from the cloud server.

4.3 ROBUSTNESS GUARANTEE

We derive robustness guarantees for our defenses against prediction and input leakage. Following the notations in Sec. 4.1, we have the following theorem of robustness guarantee for prediction leakage after applying InfoScissors. All the proofs can be found in Appendix C.

Theorem 1 *Let h_ϕ parameterize q_ϕ in Eq.* (8)*. Suppose the malicious server optimizes an auxiliary model $h^m(\mathrm{y}|\mathrm{z})$ to estimate $p(\mathrm{y}|\mathrm{z})$. For any $h^m(\mathrm{y}|\mathrm{z})$, we always have:*

$$\frac{1}{N}\sum_{i=1}^{N} \log h^m(y_i|z_i) < \frac{1}{N}\sum_{i=1}^{N} \log p(y_i) + \epsilon, \tag{11}$$

where

$$\epsilon = I_{vCLUB_{h_\phi}}(\mathrm{z};\mathrm{y}) + KL(p(\mathrm{y}|\mathrm{z})||h_\phi(\mathrm{y}|\mathrm{z})). \tag{12}$$

Specifically, if the task of collaborative inference is classification, we have the following corollary:

Corollary 1 *Suppose the task of collaborative inference is classification. Following the notations in Theorem 1 and let $epsilon$ be defined therein, we have:*

$$\frac{1}{N}\sum_{i=1}^{N} CE\left[h^m(z_i), y_i\right] > CE_{random} - \epsilon, \tag{13}$$

where CE denotes the cross-entropy loss, and CE_{random} is the cross-entropy loss of random guessing.

For input leakage, we have the following theorem of robustness.

Theorem 2 *Let the assumption of $p(\mathrm{x}|\mathrm{r})$ in Sec. 4.1 hold and g_ψ parameterize the mean of q_ψ in Eq.* (7)*. Q denotes the dimension of* x*. Suppose the malicious server optimizes an auxiliary model $g^m(\mathrm{x}|\mathrm{r})$ to estimate the mean of $p(\mathrm{x}|\mathrm{r})$. For any $g^m(\mathrm{x}|\mathrm{r})$, we always have:*

$$\frac{1}{N}\sum_{i=1}^{N} MSE\left[g^m(r_i), x_i\right] > \frac{2(\kappa - \epsilon)}{Q}, \tag{14}$$

where MSE denotes the ***mean square error****, and*

$$\begin{aligned} \kappa &= -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\sqrt{2\pi}}{p(x_i)}, \\ \epsilon &= I_{vCLUB_{g_\psi}}(\mathrm{r};\mathrm{x}) + KL(p(\mathrm{x}|\mathrm{r})||g_\psi(\mathrm{x}|\mathrm{r})). \end{aligned} \tag{15}$$

5 EXPERIMENTS

We first evaluate our method against input leakage and prediction leakage separately. Then we evaluate the integration of defenses against input and prediction leakages.

5.1 EXPERIMENTAL SETUP

Attack methods For input leakage, we evaluate InfoScissors against two model inversion (MI) attacks: (1) **Knowledge Alignment (KA)** (Wang et al., 2021) is a black-box MI attack, in which the malicious server trains an inversion model that swaps the input and output of the target model using an auxiliary dataset. The inversion model is then used to reconstruct the input given any representation. (2) **Regularized Maximum Likelihood Estimation (rMLE)** (He et al., 2020) is a white-box MI attack that the malicious server has access to the device's extractor model θ^h. The server trains input to minimize the distance between the fake representations and the received ground-truth representations. It is an unrealistic assumption that the server can access the model on the device, and we apply this white-box attack to evaluate our defense against extremely strong attacks. For prediction leakage, we evaluate our defense against two model completion (MC) attacks: (1) **Passive Model Completion (PMC)** (Fu et al., 2022b) attack assumes that the malicious server has access to an auxiliary labeled dataset and utilizes this auxiliary dataset to fine-tune a classifier that can be applied to its encoder. (2) **Active Model Completion (AMC)** (Fu et al., 2022b) attack is included as an *adaptive attack* against our defense. When the server is aware of the defense applied by the device, it conducts AMC to trick the collaborative model into relying more on its encoder, thereby extracting more data information from the encoder's features.

Baselines We compare InfoScissors with five existing defense baselines: (1) **Differential Privacy (DP)** (He et al., 2019; 2020; Oh et al., 2022) protects the data with a theoretical guarantee by clipping the representation and gradients norm and injecting perturbations to the representations and gradients. (2) **Adding Noise (AN)** (Fu et al., 2022a) is proven effective against data leakage in collaborative learning by adding Laplacian noise to the representations and gradients. (3) **Data Compression (DC)** (Fu et al., 2022a) prunes representations and gradients that are below a threshold magnitude, such that only a part of the representations and gradients are sent to the server. (4) **Privacy-preserving Deep Learning (PPDL)** (Shokri & Shmatikov, 2015) is a comprehensive privacy-enhancing method including three defense strategies: differential privacy, data compression, and random selection. (5) **Mutual Information Regularization Defense (MID)** (Zou et al., 2023) is the SOTA defense against data leakage in split learning and collaborative inference. MID is also based on mutual information regularization by applying *Variational Information Bottleneck (VIB)*.

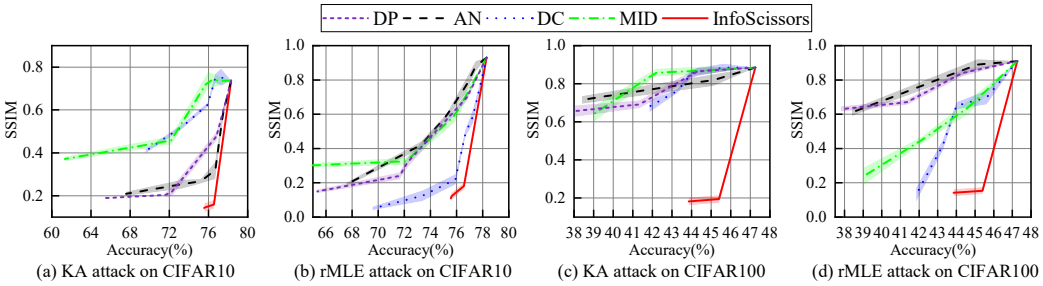
Dataset & Hyperparameter configurations We evaluate on CIFAR10 and CIFAR100. For both datasets, we apply ResNet18 as the backbone model. The first convolutional layer and the last basic block are deployed on the device as the representation extractor and the classifier, respectively. We set batch size B as 32 for both datasets. We apply SGD as the optimizer with the learning rate η set to be 0.01. The server has 40 and 400 labeled samples to conduct KA and MC attacks for CIFAR10 and CIFAR100, respectively. For InfoScissors, we apply a 1-layer decoder and a 3-layer MLP to parameterize ψ and ϕ. For AN, we apply Laplacian noise with mean of zero and scale between 0.0001-0.01. For DC, we set the compression rate from 90% to 100%. For PPDL, we set the Laplacian noise with scale of 0.0001-0.01, $\tau = 0.001$ and θ between 0 and 0.01. For MID, we set the weight of mutual information regularization between 0-0.1.

Evaluation metrics (1) **Utility metric (Model accuracy)**: We use the test data accuracy of the classifier on the device to measure the performance of the collaborative model. (2) **Robustness metric (SSIM)**: We use SSIM (structural similarity) between the reconstructed images and the raw images to evaluate the effectiveness of the defense against input leakage. The lower the SSIM, the better the defense performance. (3) **Robustness metric (Attack accuracy)**: We use the test accuracy of the server's classifier after conducting MC attacks to evaluate the defense against prediction leakage. The lower the attack accuracy, the higher the robustness against prediction leakage.

5.2 RESULTS OF INPUT PROTECTION

We conduct experiments on CIFAR10 and CIFAR100 to evaluate our defense against the KA attack and the rMLE attack. We set different defense levels for our methods (i.e., different λ_d values in

Eq. (9)) and baselines to conduct multiple experiments to show the trade-off between the model accuracy and SSIM of reconstruction. The results are shown in Fig. 3.



(a) KA attack on CIFAR10 (b) rMLE attack on CIFAR10 (c) KA attack on CIFAR100 (d) rMLE attack on CIFAR100

Figure 3: Model accuracy v.s. SSIM on CIFAR10 and CIFAR100 against MI attacks.

For defense against KA attack, our InfoScissors can reduce the SSIM of reconstruction to lower than 0.2 with a model accuracy drop of less than 2% for CIFAR10. In contrast, the other baselines reduce model accuracy by more than 10% and cannot achieve the same defense effect even with an accuracy drop of more than 10%. Notably, the malicious server has more auxiliary data on CIFAR100 than CIFAR10, making the defense harder on CIFAR100. However, InfoScissors can still achieve an SSIM of lower than 0.2 with a model accuracy drop of less than 2%. We also evaluate our defense against the KA attack with a larger auxiliary dataset on the malicious server, and the results, which can be found in Appendix D, show that our defense can effectively defend against the KA attack when the server has more auxiliary samples. For defense against rMLE attacks, InfoScissors achieves similar results of reducing the SSIM to lower than 0.2 with a model accuracy drop of less than 2% for CIFAR10 and 1% for CIFAR100, respectively, which outperforms the other baselines significantly.

	DP		AN		DC		MID		InfoScissors	
Acc(%)	76.82		76.68		76.69		75.95		76.56	
SSIM	0.4779		0.3181		0.7479		0.7373		0.159	
Acc(%)	71.63		73.26		73.38		72.17		75.62	
SSIM	0.2035		0.2535		0.5244		0.4576		0.145	
Acc(%)	65.31		67.56		69.55		61.3		75.56	
SSIM	0.1882		0.2082		0.4074		0.3713		0.1425	

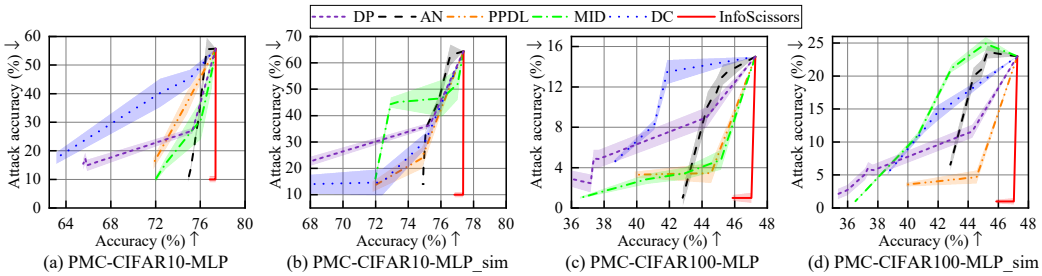
Figure 4: Images reconstructed by the KA attack on CIFAR10 under different defenses.

To perceptually demonstrate the effectiveness of our defense, we show the reconstructed images by the KA attack on CIFAR10 after applying baseline defenses and our defense in Fig. 4. It is shown that by applying the baseline defenses, the reconstructed images still contain enough information to be recognizable with the model accuracy of lower than 70%. For our method, the reconstructed images do not contain much information about the raw images, with the model accuracy higher than 76%.

5.3 Results of Prediction Protection

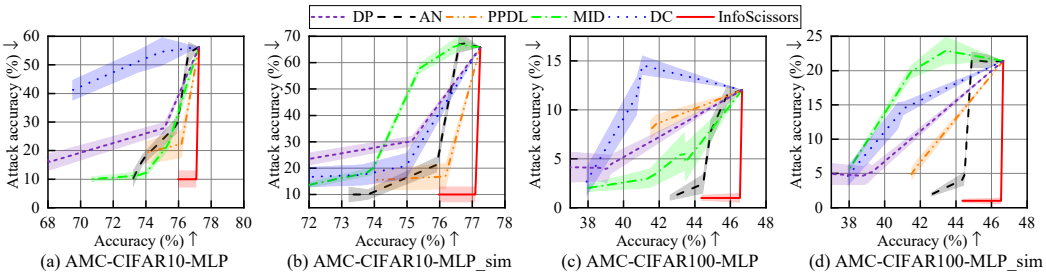
We evaluate InfoScissors on two datasets against the PMC attack and the AMC attack. We set different defense levels for our methods (i.e., different λ_l values in Eq. (9)) and baselines to conduct multiple experiments to show the trade-off between the model accuracy and attack accuracy. The defense results against PMC and AMC attacks are shown in Fig. 5 and Fig. 6, respectively. To simulate the realistic settings in that the malicious server uses different model architectures to conduct MC attacks, we apply different model architectures (MLP & MLP_sim) for MC attacks. The detailed model architectures can be found in Appendix D.

For defense against PMC on CIFAR10, InfoScissors achieves 10% attack accuracy (equal to random guess) by sacrificing less than 0.5% model accuracy, while the other baselines suffer a model accuracy drop by more than 4% to achieve the same defense effect. Similarly, InfoScissors achieves 1% attack accuracy on CIFAR100 by sacrificing less than 1% model accuracy, while the other baselines achieve the same defense effect by sacrificing more than 6% model accuracy.



(a) PMC-CIFAR10-MLP (b) PMC-CIFAR10-MLP_sim (c) PMC-CIFAR100-MLP (d) PMC-CIFAR100-MLP_sim

Figure 5: Model accuracy v.s. attack accuracy on CIFAR10 and CIFAR100 against PMC attack.



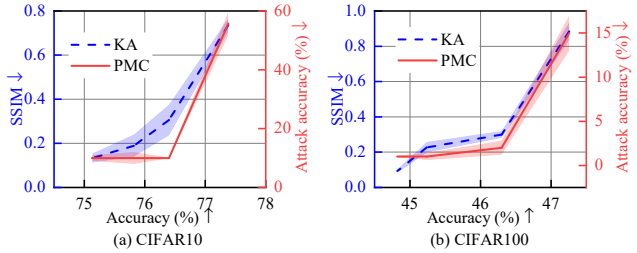
(a) AMC-CIFAR10-MLP (b) AMC-CIFAR10-MLP_sim (c) AMC-CIFAR100-MLP (d) AMC-CIFAR100-MLP_sim

Figure 6: Model accuracy v.s. attack accuracy on CIFAR10 and CIFAR100 against AMC attack.

InfoScissors also shows robustness against AMC attack. InfoScissors achieves attack accuracy of the rate of random guess by sacrificing less than 1% and 0.5% model accuracy on CIFAR10 and CIFAR100, respectively. The other baselines achieve the same defense performance by sacrificing more than 5% and 4% model accuracy, respectively.

5.4 Integration of Input and Prediction Protection

We have shown the compared results of input protection and prediction protection between InfoScissors and the baselines in Sec. 5.2 and Sec. 5.3. In this section, we evaluate the integration of input and prediction protection of InfoScissors. We set λ_d and λ_l between 0.05-0.4 and evaluate the defenses. The results of defense against the KA and PMC attacks on CIFAR10 and CIFAR100 are shown in Fig. 7. It is shown that InfoScissors can effectively protect input data and predictions simultaneously with less than a 2% accuracy drop for both datasets.



(a) CIFAR10 (b) CIFAR100

Figure 7: InfoScissors against KA and PMC on CIFAR10 and CIFAR100.

6 Conclusion and Discussion

We propose a defense method (InfoScissors) to defend against data leakage in collaborative inference by reducing the mutual information between the model's intermediate outcomes and the device's input data and predictions. The experimental results show that our method can defend against input leakage and prediction leakage effectively. We provide a theoretical robustness guarantee for our method. In this paper, we focus on the scenario where there is only one edge device. Our defense can be easily applied to the collaborative inference scenario with multiple edge devices.

REFERENCES

Amin Banitalebi-Dehkordi, Naveen Vedula, Jian Pei, Fei Xia, Lanjun Wang, and Yong Zhang. Auto-split: A general framework of collaborative edge-cloud ai. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2543–2553, 2021.

Kamalika Chaudhuri and Daniel Hsu. Sample complexity bounds for differentially private learning. In *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 155–186. JMLR Workshop and Conference Proceedings, 2011.

Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. Club: A contrastive log-ratio upper bound of mutual information. In *International conference on machine learning*, pp. 1779–1788. PMLR, 2020.

Amir Erfan Eshratifar, Mohammad Saeed Abrishami, and Massoud Pedram. Jointdnn: An efficient training and inference engine for intelligent mobile cloud computing services. *IEEE Transactions on Mobile Computing*, 20(2):565–576, 2019.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1322–1333, 2015.

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)*, pp. 1397–1414, Boston, MA, August 2022a. USENIX Association. ISBN 978-1-939133-31-1. URL https://www.usenix.org/conference/usenixsecurity22/presentation/fu-chong.

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)*, pp. 1397–1414, 2022b.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. Deep learning with label differential privacy. *Advances in neural information processing systems*, 34:27131–27145, 2021.

Zecheng He, Tianwei Zhang, and Ruby B Lee. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 148–162, 2019.

Zecheng He, Tianwei Zhang, and Ruby B Lee. Attacking and protecting data privacy in edge–cloud collaborative inference systems. *IEEE Internet of Things Journal*, 8(12):9706–9716, 2020.

Xue Jiang, Xuebing Zhou, and Jens Grossklags. Comprehensive analysis of privacy leakage in vertical federated learning during prediction. *Proceedings on Privacy Enhancing Technologies*, 2022(2):263–281, 2022.

Xiao Jin, Pin-Yu Chen, Chia-Yi Hsu, Chia-Mu Yu, and Tianyi Chen. Cafe: Catastrophic data leakage in vertical federated learning. *Advances in Neural Information Processing Systems*, 34:994–1006, 2021.

Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274, 2023.

Guangli Li, Lei Liu, Xueying Wang, Xiao Dong, Peng Zhao, and Xiaobing Feng. Auto-tuning neural network quantization framework for collaborative inference between the cloud and edge. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pp. 402–411. Springer, 2018.

Min Li, Yu Li, Ye Tian, Li Jiang, and Qiang Xu. Appealnet: An efficient and highly-accurate edge/cloud collaborative architecture for dnn inference. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 409–414. IEEE, 2021a.

Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning. *arXiv preprint arXiv:2102.08504*, 2021b.

Yang Liu, Zhihao Yi, Yan Kang, Yuanqin He, Wenhan Liu, Tianyuan Zou, and Qiang Yang. Defending label inference and backdoor attacks in vertical federated learning. *arXiv preprint arXiv:2112.05409*, 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)*, pp. 181–192. IEEE, 2021.

Ali Makhdoumi, Salman Salamatian, Nadia Fawaz, and Muriel Médard. From the information bottleneck to the privacy funnel. In *2014 IEEE Information Theory Workshop (ITW 2014)*, pp. 501–505. IEEE, 2014.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pp. 691–706. IEEE, 2019.

Fatemehsadat Mireshghallah, Mohammadkazem Taram, Prakash Ramrakhyani, Ali Jalali, Dean Tullsen, and Hadi Esmaeilzadeh. Shredder: Learning noise distributions to protect inference privacy. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 3–18, 2020.

Fatemehsadat Mireshghallah, Mohammadkazem Taram, Ali Jalali, Ahmed Taha Taha Elthakeb, Dean Tullsen, and Hadi Esmaeilzadeh. Not all features are equal: Discovering essential features for preserving prediction privacy. In *Proceedings of the Web Conference 2021*, pp. 669–680, 2021.

Seungeun Oh, Jihong Park, Sihun Baek, Hyelin Nam, Praneeth Vepakomma, Ramesh Raskar, Mehdi Bennis, and Seong-Lyun Kim. Differentially private cutmix for split learning with vision transformer. *arXiv preprint arXiv:2210.15986*, 2022.

Borzoo Rassouli and Deniz Gündüz. Optimal utility-privacy trade-off with total variation distance as a privacy measure. *IEEE Transactions on Information Forensics and Security*, 15:594–603, 2019.

Nir Shlezinger, Erez Farhan, Hai Morgenstern, and Yonina C Eldar. Collaborative inference via ensembles on the edge. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8478–8482. IEEE, 2021.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pp. 1310–1321, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338325. doi: 10.1145/2810103.2813687. URL https://doi.org/10.1145/2810103.2813687.

Abhishek Singh, Ayush Chopra, Ethan Garza, Emily Zhang, Praneeth Vepakomma, Vivek Sharma, and Ramesh Raskar. Disco: Dynamic and invariant sensitive channel obfuscation for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12125–12135, 2021.

Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. Soteria: Provable defense against privacy leakage in federated learning from representation perspective. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9311–9319, 2021.

Tianhao Wang, Yuheng Zhang, and Ruoxi Jia. Improving robustness to model inversion attacks via mutual information regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11666–11673, 2021.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Qihong Wu, Jinchuan Tang, Shuping Dang, and Gaojie Chen. Data privacy and utility trade-off based on mutual information neural estimator. *Expert Systems with Applications*, 207:118012, 2022.

Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16337–16346, 2021.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Hongbo Zhou, Weiwei Zhang, Chengwei Wang, Xin Ma, and Haoran Yu. Bbnet: a novel convolutional neural network structure in edge-cloud collaborative inference. *Sensors*, 21(13):4494, 2021.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

Tianyuan Zou, Yang Liu, and Ya-Qin Zhang. Mutual information regularization for vertical federated learning. *arXiv preprint arXiv:2301.01142*, 2023.

A Data Leakage in Collaborative Inference

A.1 Input Leakage from the Representation

With the received representation r_i, the server can reconstruct the input data x_i on the edge device by conducting model inversion (MI) attacks Wang et al. (2021). Notably, the head model on the edge device is usually shallow due to the computation resource limitation, which aggravates input leakage from the representation He et al. (2020). We conduct experiments on CIFAR10 with ResNet18 to demonstrate the input leakage problem. One convolutional layer is deployed on the device as the head model, and one basic block is deployed as the classifier. The malicious server conducts Knowledge Alignment (KA) attack Wang et al. (2021) to recover the input image from the received representation through a generator. The detailed experimental settings can be found in Sec. 5. The reconstructed images are shown in Fig. 8. The high-quality reconstructed images illustrate the collaborative inference's vulnerability to the device's input leakage from the representation.

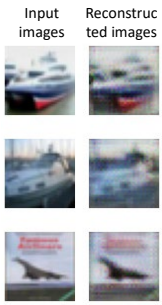


Figure 8: Reconstructed images of KA attack.

A.2 Prediction Leakage from the Feature

The training process enables the cloud server to extract high-level features useful for the collaborative inference task. These high-level features allow the malicious server to fine-tune a classifier head with very few labeled data and accurately conduct inference. The leakage of the prediction makes the edge device holder's privacy, including behavior and preference, exposed to the cloud server. For example, prediction leakage of a collaborative inference-based navigation mobile app allows the cloud server to infer the positions and destinations of the app users. To demonstrate the extent of prediction leakage by the features, we follow the experimental setup in Appendix A.1 and let the cloud server conduct model completion (MC) attack Fu et al. (2022b) to train a classifier using a small number of auxiliary labeled samples. We also let the cloud server train an entire model with the auxiliary dataset from scratch for comparison. The results are shown in Tab. 1.

Table 1: Compared accuracy of the classifier on the device and the models on the cloud server by conducting MC attack and training from scratch.

	Accuracy(%)
Classifier on the device (clean accuracy)	77.20
MC attack on the server(40 labels)	69.31
Train from scratch on the server(40 labels)	15.34

It is shown that by fine-tuning a classifier with the collaboratively trained encoder, the cloud server can achieve an accuracy of nearly 70% using an auxiliary dataset with only 40 labeled samples. However, training from scratch cannot achieve decent accuracy using the same auxiliary dataset, which shows that the high-level features extracted by the encoder on the server cause prediction leakage.

B ALGORITHM

Algorithm 1 Training algorithm of InfoScissors. $\leftarrow$ means information is sent to the server; $\leftarrow$ means information is sent to the device; red steps are conducted on the cloud server.

Input: Dataset $\{(x_i, y_i)\}_{i=1}^{N}$; Learning rate η.
Output: $\theta^h; \theta^e; \theta^c; \psi; \phi$.

Initialize $\theta^h, \theta^e, \theta^c, \psi, \phi$;
for a batch of data $\{(x_i, y_i)\}_{i\in\mathbb{B}}$ **do**
  $\{r_i\}_{i\in\mathbb{B}} \leftarrow \{f^h_{\theta^h}(x_i)\}_{i\in\mathbb{B}}$;
  $\mathcal{L}_{d\_a} \leftarrow \frac{1}{|\mathbb{B}|}\sum_{i\in\mathbb{B}} \log g_\psi(x_i|r_i)$;
  $\psi \leftarrow \psi + \eta\nabla_\psi \mathcal{L}_{d\_a}$;
  $\{z_i\}_{i\in\mathbb{B}} \leftarrow \{f^e_{\theta^e}(r_i)\}_{i\in\mathbb{B}}$;
  $\mathcal{L}_c \leftarrow \frac{1}{|\mathbb{B}|}\sum_{i\in\mathbb{B}} \mathcal{L}\left(f^c_{\theta^c}(z_i), y_i\right)$;
  $\mathcal{L}_{l\_a} \leftarrow \frac{1}{|\mathbb{B}|}\sum_{i\in\mathbb{B}} \log h_\phi(y_i|z_i)$;
  $\theta^c \leftarrow \theta^c - \eta\nabla_{\theta^c}\mathcal{L}_c$;
  $\phi \leftarrow \phi + \eta\nabla_\phi \mathcal{L}_{l\_a}$;
  $\{y_{n'_i}\}_{i\in\mathbb{B}} \leftarrow$ randomly sample $\{y_{n'_i}\}_{i\in\mathbb{B}}$ from $\{y_i\}_{i\in[N]}$;
  $\{x_{k'_i}\}_{i\in\mathbb{B}} \leftarrow$ randomly sample $\{x_{k'_i}\}_{i\in\mathbb{B}}$ from $\{x_i\}_{i\in[N]}$;
  $\mathcal{L}_{d\_r} \leftarrow \frac{1}{|\mathbb{B}|}\sum_{i\in\mathbb{B}} -\log g_\psi\left(x_{k'_i}|r_i^2\right)$;
  $\mathcal{L}_{l\_r} \leftarrow \frac{1}{|\mathbb{B}|}\sum_{i\in\mathbb{B}} -\log h_\phi\left(y_{n'_i}|z_i^2\right)$;
  $\{\nabla_{z_i}\mathcal{L}\}_{i\in\mathbb{B}} \leftarrow \{\nabla_{z_i}[(1-\lambda_d-\lambda_l)\mathcal{L}_c + \lambda_l\mathcal{L}_{l\_a} + \lambda_l\mathcal{L}_{l\_r} + \lambda_d\mathcal{L}_{d\_a} + \lambda_d\mathcal{L}_{d\_r}]\}_{i\in\mathbb{B}}$;
  $\nabla_{\theta^e}\mathcal{L} \leftarrow \frac{1}{|\mathbb{B}|}\sum_{i\in\mathbb{B}} \nabla_{z_i}\mathcal{L}\nabla_{\theta^e} z_i$;
  $\theta^e \leftarrow \theta^e - \eta\nabla_{\theta^e}\mathcal{L}$;
  $\{\nabla_{r_i}\mathcal{L}\}_{i\in\mathbb{B}} \leftarrow \{\nabla_{z_i}\mathcal{L}\nabla_{r_i} z_i\}_{i\in\mathbb{B}}$;
  $\nabla_{\theta^h}\mathcal{L} \leftarrow \frac{1}{|\mathbb{B}|}\sum_{i\in\mathbb{B}} \nabla_{r_i}\mathcal{L}\nabla_{\theta^h} r_i$;
  $\theta^h \leftarrow \theta^h - \eta\nabla_{\theta^h}\mathcal{L}$;
end for

C PROOFS OF THEOREMS

Proof 1 *According to Corollary 3.3 in Cheng et al. (2020), we have:*

$$I(\mathrm{z};\mathrm{y}) < I_{vCLUB}(\mathrm{z};\mathrm{y}) + KL(p(y|z)||h_\phi(y|z)). \tag{16}$$

Then we have

$$I(\mathrm{z};\mathrm{y}) = \mathbb{E}_{p(\mathrm{z},\mathrm{y})} \log p\left(\mathrm{y}|\mathrm{z}\right) - \mathbb{E}_{p(\mathrm{y})} \log p\left(\mathrm{y}\right) < \epsilon, \tag{17}$$

where $\epsilon = I_{vCLUB}(\mathrm{z};\mathrm{y}) + KL(p(y|z)||h_\phi(y|z))$*. With the samples* $\{x_i, y_i\}$*,* $I(\mathrm{z};\mathrm{y})$ *has an unbiased estimation as:*

$$\frac{1}{N}\sum_{i=1}^{N} \log p(y_i|z_i) - \frac{1}{N}\sum_{i=1}^{N} \log p(y_i) < \epsilon. \tag{18}$$

Suppose the adversary has an optimal model h^m *to estimate* $p(y_i|z_i)$ *such that* $h^m(y_i|z_i) = p(y_i|z_i)$ *for any* i*, then*

$$\frac{1}{N}\sum_{i=1}^{N} \log h^m(y_i|z_i) - \frac{1}{N}\sum_{i=1}^{N} \log p(y_i) < \epsilon. \tag{19}$$

For ***classification tasks****, we have*

$$\frac{1}{N}\sum_{i=1}^{N} CE\left[h^m(z_i), y_i\right] > CE_{random} - \epsilon. \tag{20}$$

Proof 2 *Similar with Eq.* (18)*, we derive the following for data protection:*

$$\frac{1}{N}\sum_{i=1}^{N} \log p(x_i|r_i) - \frac{1}{N}\sum_{i=1}^{N} \log p(x_i) < \epsilon, \tag{21}$$

where $\epsilon = I_{vCLUB}(\mathrm{r};\mathrm{x}) + KL(p(x|r)||g_\psi(x|r))$*. Following the assumption that* $p(x|r)$ *follows a Gaussian distribution of variance 1, suppose the adversary obtains an optimal estimator* g_m *of the mean of* $p(x|r)$ *such that* $g^m(x_i|r_i) = p(x_i|r_i)$ *for any* i*. Then we have*

$$\begin{aligned}
\frac{1}{N}\sum_{i=1}^{N} \log g^m(x_i|r_i) &< \frac{1}{N}\sum_{i=1}^{N} \log p(x_i) + \epsilon \\
\frac{1}{N}\sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}[x_i - g^m(r_i)]^T [x_i - g^m(r_i)]} &< \frac{1}{N}\sum_{i=1}^{N} \log p(x_i) + \epsilon \\
-\frac{1}{N}\sum_{i=1}^{N} \log \sqrt{2\pi} - \frac{1}{2N}\sum_{i=1}^{N} \left[x_i - g^m(r_i)\right]^T \left[x_i - g^m(r_i)\right] &< \frac{1}{N}\sum_{i=1}^{N} \log p(x_i) + \epsilon \\
\frac{1}{2N}\sum_{i=1}^{N} \left[x_i - g^m(r_i)\right]^T \left[x_i - g^m(r_i)\right] &> \frac{1}{N}\sum_{i=1}^{N} \log \frac{\sqrt{2\pi}}{p(x_i)} - \epsilon.
\end{aligned} \tag{22}$$

We denote the dimension of x *as* Q *and* $\frac{1}{N}\sum_{i=1}^{N} \log \frac{\sqrt{2\pi}}{p(x_i)}$ *as* κ*. Then we have*

$$\frac{1}{N}\sum_{i=1}^{N} MSE\left[g^m(r_i), x_i\right] > \frac{2(\kappa - \epsilon)}{Q}. \tag{23}$$

D ADDITIONAL EXPERIMENTAL RESULTS

The malicious server uses models with different architectures (MLP and MLP_sim) to conduct MC attacks. MLP_sim has one FC layer. MLP has three FC layers with a hidden layer of size 512×256.

Besides the experiments in Sec. 5, we also evaluate our defense against the KA attack with a larger auxiliary dataset on the malicious server. The server has 80 and 800 labeled samples to conduct KA and MC attacks for CIFAR10 and CIFAR100, respectively, and the results are shown in Fig. 9.

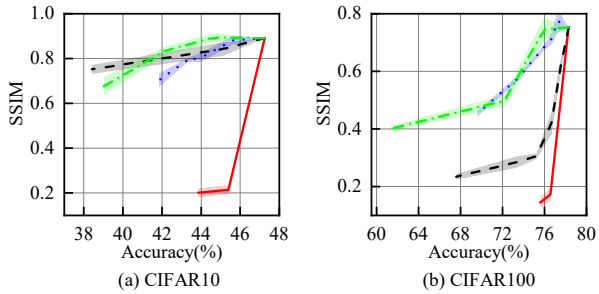


Figure 9: Model accuracy v.s. SSIM on CIFAR10 and CIFAR100 against KA attack with double numbers of auxiliary samples in Fig. 4.