Supplementary Materials for Paper ID 927

Song Xia[1], Yi Yu[1], Wenhan Yang[2*], Meiwen Ding[1], Zhuo Chen[2],
Lingyu Duan[2,3], Alex C. Kot[1], Xudong Jiang[1]
[1]Nanyang Technological University, [2]Pengcheng Laboratory, [3]Peking University
{xias0002,yuyi0010,ding0159,eackot,exdjiang}@ntu.edu.sg, yangwh@pcl.ac.cn

S1. Proof of Theorem 1

Given the covariance matrix $Cov(\boldsymbol{x}|\boldsymbol{z})$, the conditional entropy $\mathcal{H}(\boldsymbol{x}|\boldsymbol{z})$ satisfies that:

$$\mathcal{H}(\boldsymbol{x}|\boldsymbol{z}) \leq \mathbb{E}_{\mathcal{Z}}\left[\frac{1}{2}\log\left((2\pi e)^d \det\left(Cov(\boldsymbol{x}|\boldsymbol{z})\right)\right)\right], \tag{S1}$$

where det denotes the determinant of the matrix. Equation S1 formalizes the principle that the Gaussian distribution achieves the maximum entropy among all distributions with a given covariance.
Let $\lambda \in \{\lambda_1, ..., \lambda_d\}$ denote the eigenvalues of the matrix $Cov(\boldsymbol{x}|\boldsymbol{z})$. It follows that:

$$\det(Cov(\boldsymbol{x}|\boldsymbol{z})) = \prod_{i=1}^{d}\lambda_i,\ Tr(Cov(\boldsymbol{x}|\boldsymbol{z})) = \sum_{i=1}^{d}\lambda_i. \tag{S2}$$

Using the Jensen inequality, we can get:

$$\mathcal{H}(\boldsymbol{x}|\boldsymbol{z}) \leq \mathbb{E}_{\mathcal{Z}}\left[\frac{d}{2}\log\left(2\pi e\frac{Tr\left(Cov(\boldsymbol{x}|\boldsymbol{z})\right)}{d}\right)\right], \tag{S3}$$

As the log function is concave, we can get:

$$\mathcal{H}(\boldsymbol{x}|\boldsymbol{z}) \leq \frac{d}{2}\log\left(\frac{2\pi e}{d}\mathbb{E}_{\mathcal{Z}}\left[Tr\left(Cov(\boldsymbol{x}|\boldsymbol{z})\right)\right]\right), \tag{S4}$$

$$\leq \frac{d}{2}\log\left(2\pi e\varepsilon\right), \tag{S5}$$

which concludes the proof of Theorem 1.

S2. Proof of Theorem 2

Proposition 2 illustrates that maximizing the $H(\boldsymbol{x}|\boldsymbol{z})$ is equivalent to minimizing the mutual information $\mathcal{I}(\boldsymbol{z};\hat{\boldsymbol{z}})$, which is:

$$\mathcal{I}(\boldsymbol{z};\hat{\boldsymbol{z}}) = \mathcal{H}(\boldsymbol{z}) - \mathcal{H}(\boldsymbol{z}|\hat{\boldsymbol{z}}) = \mathcal{H}(\boldsymbol{z}) - \frac{1}{2}\log((2\pi e)^d\left|\Sigma_p\right|). \tag{S6}$$

It is hard to give a closed-form representation of $\mathcal{H}(\boldsymbol{z})$ when $\boldsymbol{z}$ follows the Gaussian mixture distribution. In [3], an upper bound of $\mathcal{H}(\boldsymbol{z})$ is given by:

$$\mathcal{H}(\boldsymbol{z}) \leq \sum_{i=1}^{k}\pi_i\left(-\log(\pi_i) + \frac{1}{2}\log((2\pi e)^d\left|\Sigma_i + \Sigma_p\right|)\right). \tag{S7}$$

*Corresponding Author

Therefore, we claim that the mutual information $\mathcal{I}(\boldsymbol{z}; \hat{\boldsymbol{z}})$ satisfies that:

$$\mathcal{I}(\boldsymbol{z}; \hat{\boldsymbol{z}}) \leq \sum_{i=1}^{k} \pi_i \left(-\log(\pi_i) + \frac{1}{2} \log(\frac{|\Sigma_i + \Sigma_p|}{|\Sigma_p|}) \right), \tag{S8}$$

which concludes the proof of Theorem 2.

S3. Architecture and Training details of Inversion Models

Decoding-based inversion model: We allow the inversion adversary to utilize a complex inversion model to reconstruct the original inputs. Specifically, we utilize a decoder network with 8 concatenated residual blocks and the corresponding number of transpose convolutional blocks to recover the original size of the input. Each convolutional layer has 64 channels.

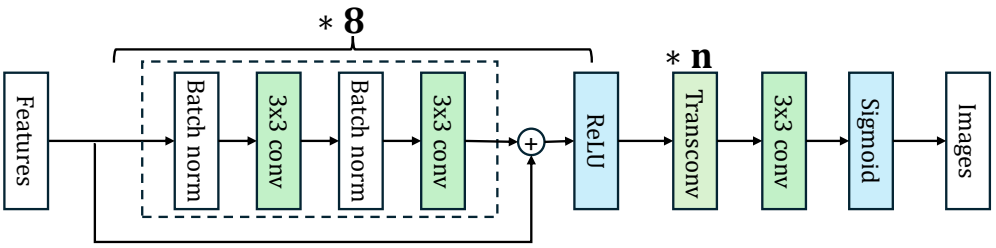

Figure S1. The structure of the decoding network.

The architecture of the decoding-based inversion model is illustrated in Figure S1. Specifically, the initial eight residual blocks are designed to process the extracted features, while the transposed convolutional layers progressively upsample the feature maps to match the dimensions of the original image. The final convolutional layer performs the concluding processing, and the application of the Sigmoid activation function normalizes the output to the range [0, 1]. The decoder comprises approximately 711.54k trainable parameters and requires 90.66 MMAC operations for computation. We train the decoder model for 50 epochs using the Adam optimizer with an initial learning rate of 0.005.

GAN-based inversion model: We follows the methodology outlined in [8] to implement the GAN-based inversion attack. Concurrently, we replace the original generator with the proposed decoding-based network, which is architecturally more complex and delivers superior performance.

The GAN inversion model is trained for 150 epochs using the Adam optimizer with an initial learning rate of 0.005. Additionally, a MSE loss term is incorporated to regularize the training process, where we find it effectively facilitates the GAN inversion model in achieving a lower reconstruction MSE.

S4. Experimental results of SSIM and PSNR

The experimental results of reconstruction SSIM and PSNR on the validation set on the CIFAR10, CIFAR100, TinyImageNet, and FaceScrub datasets are presented in Table S1 to S4. We provide a comparative analysis of each method, both before and after integrating the proposed CEM algorithm.

The results show that the integration of the CEM algorithm consistently enhances all defense methods on four datasets. On the CIFAR10 dataset, plugging in our proposed CEM algorithm improves the average of **SSIM from 0.673 to 0.639 and PSNR from 18.28 to 17.51**. On the CIFAR100 dataset, plugging in our proposed CEM algorithm improves the average of **SSIM from 0.814 to 0.755 and PSNR from 23.06 to 20.63**. On the TinyImageNet dataset, plugging in our proposed CEM algorithm improves the average of **SSIM from 0.567 to 0.523 and PSNR from 18.09 to 17.37**. On the FaceScrub dataset, plugging in our proposed CEM algorithm improves the average of **SSIM from 0.794 to 0.752 and PSNR from 21.59 to 20.07**.

Table S1. Comparative results on CIFAR10 dataset: we present the accuracy and reconstruction SSIM and PSNR on the validation set.

Methods	**Acc.↑**	**Dec.-based MIA [5]**		**GAN-based MIA [8]**	
		SSIM↓	**PSNR↓**	**SSIM↓**	**PSNR↓**
Bottleneck	90.87	0.863	23.82	0.861	23.46
Bottleneck+CEM	90.69	0.813	22.36	0.783	20.96
DistCorr [7]	89.52	0.779	20.86	0.780	20.55
DistCorr+CEM	89.80	0.757	20.31	0.760	20.17
Dropout [2]	87.75	0.729	19.82	0.733	19.54
Dropout+CEM	87.53	0.682	18.72	0.687	18.47
PATROL [1]	89.58	0.537	15.33	0.558	15.12
PATROL+CEM	89.67	0.506	14.74	0.520	14.59
ResSFL [5]	89.68	0.595	16.19	0.639	16.14
ResSFL+CEM	90.11	0.571	16.00	0.583	15.63
Noise_Nopeek [6]	87.19	0.664	17.98	0.668	17.82
Noise_Nopeek+CEM	87.08	0.643	17.59	0.651	17.49
Noise_ARL [4]	87.78	0.501	14.73	0.518	14.65
Noise_ARL+CEM	87.62	0.484	14.48	0.502	14.40
Average w/o CEM	88.91	0.667	18.39	0.680	18.18
Average w/ CEM	**88.92**	**0.637**	**17.74**	**0.641**	**17.38**

Table S2. Comparative results on CIFAR100 dataset: we present the accuracy and reconstruction SSIM and PSNR on the validation set.

Methods	**Acc.↑**	**Dec.-based MIA [5]**		**GAN-based MIA [8]**	
		SSIM↓	**PSNR↓**	**SSIM↓**	**PSNR↓**
Bottleneck	68.43	0.975	31.54	0.970	30.96
Bottleneck+CEM	68.42	0.856	22.36	0.937	26.98
DistCorr [7]	66.21	0.880	22.67	0.928	26.02
DistCorr+CEM	66.27	0.812	21.30	0.877	23.46
Dropout [2]	65.85	0.865	23.76	0.936	27.44
Dropout+CEM	65.92	0.816	22.21	0.890	24.94
PATROL [1]	65.10	0.478	14.74	0.634	16.23
PATROL+CEM	65.07	0.440	13.77	0.603	15.96
ResSFL [5]	66.94	0.866	22.92	0.935	26.98
ResSFL+CEM	66.96	0.770	20.31	0.866	23.37
Noise_Nopeek [6]	65.55	0.841	22.00	0.890	24.20
Noise_Nopeek+CEM	65.33	0.797	20.80	0.858	22.83
Noise_ARL [4]	62.58	0.521	15.57	0.691	17.85
Noise_ARL+CEM	62.34	0.457	16.27	0.599	14.40
Average w/o CEM	65.81	0.775	21.88	0.854	24.24
Average w/ CEM	**65.75**	**0.706**	**19.57**	**0.804**	**21.70**

Table S3. Comparative results on TinyImageNet dataset: we present the accuracy and reconstruction SSIM and PSNR on the validation set.

Methods	**Acc.↑**	**Dec.-based MIA [5]**		**GAN-based MIA [8]**	
		SSIM↓	**PSNR↓**	**SSIM↓**	**PSNR↓**
Bottleneck	52.77	0.666	19.87	0.698	20.36
Bottleneck+CEM	52.52	0.593	18.86	0.623	19.03
DistCorr [7]	51.79	0.598	18.38	0.627	18.66
DistCorr+CEM	51.78	0.527	17.21	0.553	17.74
Dropout [2]	50.72	0.548	17.87	0.567	18.21
Dropout+CEM	50.75	0.511	17.30	0.533	17.64
PATROL [1]	51.75	0.512	17.28	0.536	17.54
PATROL+CEM	51.64	0.489	16.79	0.524	17.23
ResSFL [5]	51.99	0.522	17.64	0.552	17.93
ResSFL+CEM	52.07	0.495	17.12	0.521	17.61
Noise_Nopeek [6]	51.63	0.578	18.04	0.588	18.18
Noise_Nopeek+CEM	52.01	0.527	17.44	0.551	17.77
Noise_ARL [4]	51.03	0.463	16.49	0.486	16.88
Noise_ARL+CEM	50.85	0.428	15.67	0.441	15.96
Average w/o CEM	**51.66**	0.555	17.93	0.579	18.25
Average w/ CEM	51.62	**0.510**	**17.19**	**0.535**	**17.56**

Table S4. Comparative results on FaceScrub dataset: we present the prediction accuracy and reconstruction SSIM and PSNR on the validation set.

Methods	**Acc.↑**	**Dec.-based MIA [5]**		**GAN-based MIA [8]**	
		SSIM↓	**PSNR↓**	**SSIM↓**	**PSNR↓**
Bottleneck	85.12	0.898	26.02	0.864	25.68
Bottleneck+CEM	85.00	0.860	24.45	0.821	24.31
DistCorr [7]	83.42	0.853	24.20	0.848	23.87
DistCorr+CEM	83.78	0.833	23.27	0.795	21.30
Dropout [2]	79.30	0.813	22.83	0.808	22.44
Dropout+CEM	79.19	0.776	21.30	0.771	20.91
PATROL [1]	79.18	0.737	19.28	0.731	18.63
PATROL+CEM	79.88	0.685	17.32	0.681	17.16
ResSFL [5]	79.60	0.745	19.50	0.736	18.47
ResSFL+CEM	79.54	0.713	18.29	0.710	17.93
Noise_Nopeek [6]	82.06	0.844	22.83	0.839	22.51
Noise_Nopeek+CEM	81.96	0.784	21.24	0.777	20.45
Noise_ARL [4]	80.14	0.705	18.09	0.710	18.01
Noise_ARL+CEM	80.33	0.669	16.75	0.660	16.36
Average w/o CEM	81.26	0.799	21.82	0.790	21.37
Average w/ CEM	**81.38**	**0.760**	**20.37**	**0.745**	**19.77**

S5. Experimental results on CIFAR100

The performance of different defense methods on the CIFAR100 dataset is presented in Table S5, where we provide a comparative analysis of each method, both before and after integrating the proposed CEM algorithm. The results indicate again that the integration of the CEM algorithm brings substantial inversion robustness gain. Integrating the CEM algorithm **yields an average increase in the reconstruction MSE of 40.5% for training data and 44.8% for inference data**, without compromising prediction accuracy.

Table S5. Comparative results on CIFAR100 dataset: we present the accuracy and reconstruction MSE of different defense methods using the VGG11 model with and without integrating the proposed CEM.

Methods	**Acc.↑**	**Dec.-based MSE [5]**		**GAN-based MSE [8]**	
		Train↑	**Infer↑**	**Train↑**	**Infer↑**
Bottleneck	68.43	0.0007	0.0007	0.0009	0.0008
Bottleneck+CEM	68.42	0.0058	0.0059	0.0019	0.0020
DistCorr [7]	66.21	0.0054	0.0055	0.0023	0.0025
DistCorr+CEM	66.27	0.0074	0.0075	0.0044	0.0045
Dropout [2]	65.85	0.0042	0.0043	0.0016	0.0018
Dropout+CEM	65.92	0.0060	0.0061	0.0031	0.0032
PATROL [1]	65.10	0.0335	0.0346	0.0196	0.0238
PATROL+CEM	65.07	0.0419	0.0423	0.0235	0.0253
ResSFL [5]	66.94	0.0051	0.0052	0.0019	0.0020
ResSFL+CEM	66.96	0.0093	0.0095	0.0043	0.0046
Noise_Nopeek [6]	65.55	0.0063	0.0063	0.0037	0.0038
Noise_Nopeek+CEM	65.33	0.0083	0.0082	0.0052	0.0052
Noise_ARL [4]	62.58	0.0270	0.0277	0.0143	0.0164
Noise_ARL+CEM	62.34	0.0373	0.0380	0.0219	0.0236
Average w/o CEM	**65.81**	0.0117	0.0120	0.0063	0.0073
Average w/ CEM	65.75	**0.0165**	**0.0168**	**0.0095**	**0.0102**

S6. Analysis of the efficiency

We evaluate the inference time and model size of the local encoder and cloud server using one RTX 4090 GPU, with detailed results presented in Table S6. Notably, all methods, except for PATROL, exhibit identical inference efficiency and parameter counts. The inference time is measured by processing a batch of 128 images over 100 times

Table S6. Analysis of the efficiency.

Method	**Local encoder**			**Could server**		
	Parameters	**Flops**	**Infere time**	**Parameters**	**Flops**	**Infere time**
PATROL	0.083M	27.3 MAC	65.61 ms	9.78M	136.38MAC	175.92ms
OTHERS	0.085M	21.9 MAC	55.16 ms	9.78M	133.59MAC	169.87ms

References

[1] Shiwei Ding, Lan Zhang, Miao Pan, and Xiaoyong Yuan. Patrol: Privacy-oriented pruning for collaborative inference against model inversion attacks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024. 3, 4, 5

[2] Zecheng He, Tianwei Zhang, and Ruby B Lee. Attacking and protecting data privacy in edge–cloud collaborative inference systems. *IEEE Internet of Things Journal*, 2020. 3, 4, 5

[3] Marco F Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D Hanebeck. On entropy approximation for gaussian mixture random vectors. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2008. 1

[4] Jonghu Jeong, Minyong Cho, Philipp Benz, and Tae-hoon Kim. Noisy adversarial representation learning for effective and efficient image obfuscation. In *Uncertainty in Artificial Intelligence*, 2023. 3, 4, 5

[5] Jingtao Li, Adnan Siraj Rakin, Xing Chen, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. Ressfl: A resistance transfer framework for defending model inversion attack in split federated learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 3, 4, 5

[6] Tom Titcombe, Adam J Hall, Pavlos Papadopoulos, and Daniele Romanini. Practical defences against model inversion attacks for split neural networks. *arXiv preprint arXiv:2104.05743*, 2021. 3, 4, 5

[7] Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. Nopeek: Information leakage reduction to share activations in distributed deep learning. In *2020 International Conference on Data Mining Workshops (ICDMW)*, 2020. 3, 4, 5

[8] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2, 3, 4, 5