Ungeneralizable Examples

Supplementary Material

In this document, we present supplementary materials that couldn't be accommodated within the main manuscript due to page limitations. Specifically, we offer additional details on the proposed UGE framework, including the architecture of the generator and the modified losses with multiple authorized networks and concrete experimental setting.

6. More Details of UGEs

To optimize the UGEs, we utilize a total loss consisting of three distinct components. Below, we provide more details on constructing the UGE framework for multiple authorized networks, as well as specifics regarding the generator for the ungeneralizable noise.

6.1. UGEs with Multiple Authorized Networks

It is mentioned in the main paper that we consider the scenario with one authorized network. However, we assert that our proposed framework is capable of handling cases where multiple authorized networks are determined by the protector.

Denote the authorized network set as: $F = \{f_\theta^1, f_\theta^2, ..., f_\theta^K\}$, then for each loss item in $\mathcal{L}_{all}$, each could be rewritten as:

$$\mathcal{L}_{gm} = \frac{1}{|\tau| \times n \times K} \sum_{f^k \in F} \sum_{t \in \tau} \sum_{(x,y) \in \mathcal{D}} \mathcal{D}ist\Big[\nabla \mathcal{L}\big(f_{\theta_t}^k(x), y\big), \nabla \mathcal{L}\big(f_{\theta_t}^k(x_u), y\big)\Big],$$
$$\mathcal{L}_{ud} = \frac{1}{n \times K} \sum_{f^k \in F} \sum_{(x_u, y_u) \in \mathcal{D}_u} \Big[\mathcal{L}\big(f_\theta^k(x_u), y_u\big) - \omega \mathcal{L}_{kd}\big(f_\theta^k(x_u), f'_{\theta_A}(x_u)\big)\Big]. \quad (12)$$

Here, we modify the loss items of $\mathcal{L}_{gm}$ and $\mathcal{L}_{ud}$ while keeping the loss item $\mathcal{L}_{fd}$ unchanged. It is important to note that as the total number of authorized networks increases, the performance of our synthetic method may decrease. Detailed experiments regarding the number K are conducted in the subsequent section.

6.2. The Architecture of the Generator

Note that in the main paper, we utilize the generator $\mathcal{G}$ to synthesize the ungeneralizable version of the data, which is denoted as:

$$x_u = \mathcal{G}(x), \quad x \in \mathcal{D}, \quad (13)$$

where we omit the operation to constrain the norm of the ungeneralizable noise. And the ungeneralizable examples x_u form the final published ungeneralizable dataset. We use the ResNet based backbone for constructing the generator. To be concrete, the architecture of the generator is given in Table. 4 and Fig. 5, which consists of conv, residual and upsampling blocks.

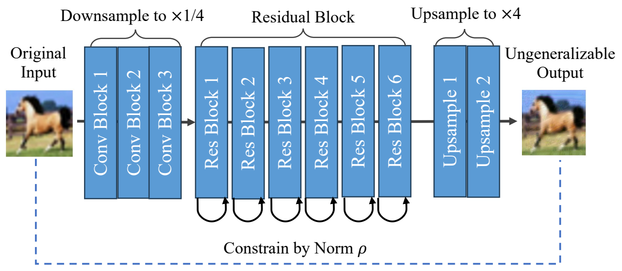

Figure 5. The architecture of the generator to synthesize the ungeneralizable examples.

Table 4. The architecture of the generative perturbation network.

Block Name	**Output Size**	**Layers**
Conv Block	56×56	[Conv Layer (7×7), BatchNorm, LeakyReLU] $\times 1$
Conv Block	56×56	[Conv Layer (3×3), BatchNorm, LeakyReLU] $\times 2$
Res Block	56×56	[Conv Layer (3×3), BatchNorm, LeakyReLU, Dropout, Conv Layer (3×3), BatchNorm] $\times 6$
Upsample	224×224	[Conv Transpose(3×3), BatchNorm, LeakyReLU] $\times 2$, Conv Layer

Table 5. The hyper parameters setting for CIFAR-10, CIFAR-100 and TinyImageNet datasets.

Dataset	$\mathcal{L}_{all}$		ℓ_{tri}	$\mathcal{L}_{ud}$	
	λ_{fd}	λ_{ud}	α	ω	T
CIFAR-10	1	0.1	0.1	0.1	4
CIFAAR-100	1	0.1	0.1	0.1	20
TinyImageNet	1	1	0.1	0.01	20

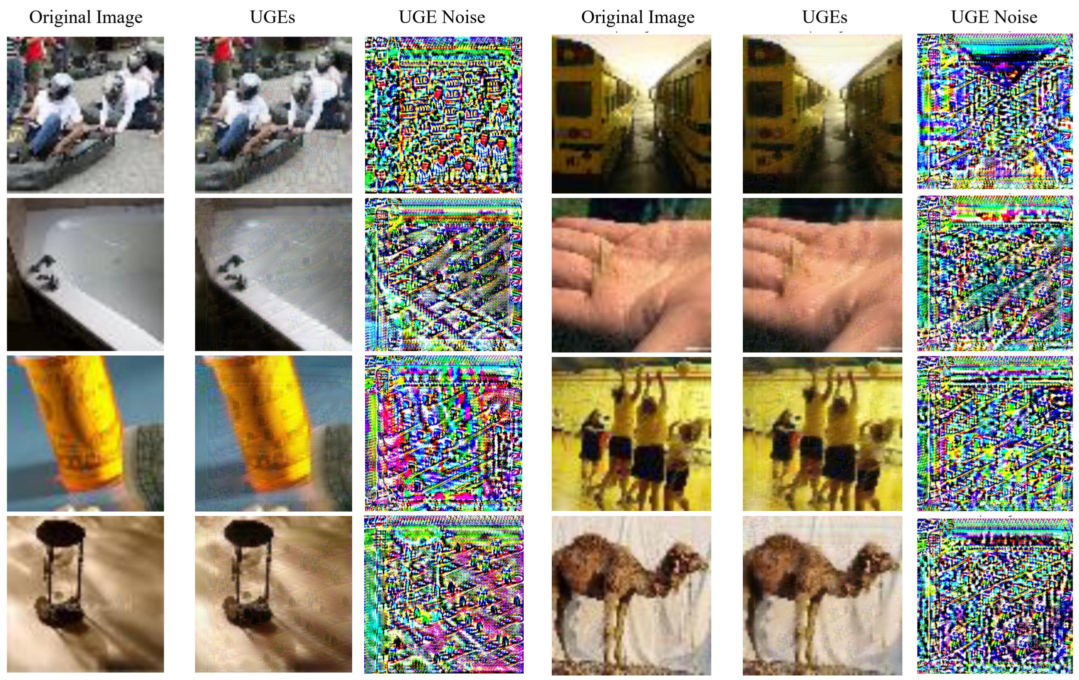

Figure 6. The visualization results include the original clean images, the ungeneralizable noise (scaled by 255 for better visualization), and the resultant ungeneralizable images. These visualizations are presented for TinyImageNet dataset.

Table 6. The training details for generator and normal networks on CIFAR-10, CIFAR-100 and TinyImageNet datasets.

Dataset	**Network**	**Learning Rate**	**Epoch**
CIFAR-10	Generator	1e-3	50
	Plain CNN	1e-3	100
	Other Networks	0.1	160
CIFAR-100	Generator	1e-3	100
	Networks	0.1	200
TinyImageNet	Generator	1e-3	100
	Networks	0.1	200

7. Experiments Setup

Here is a detailed setting for each part of the experiments.

The balancing weights and other hyperparameters in each loss item are provided in Table 5, specifying the parameter settings for CIFAR-10, CIFAR-100, and TinyImageNet datasets, respectively.

And the details regarding the network training are given in Table 6, where the training of the generator and the normal networks is given.

8. More Experimental Results

8.1. More Visualization Results on TinyImageNet

In the main paper, we presented visualizations of ungeneralizable examples on CIFAR-10 and CIFAR-100 datasets. Here, we provide additional visualization results on the TinyImageNet dataset, as shown in Fig. 6. We also visualize the ungeneralizable noise, which could reflect some details of the original image, showing that the learned UGE noise is sample-wise. The figure illustrates that our proposed UGE framework is capable of generating visually integrated ungeneralizable images from the original inputs, demonstrating its effectiveness on more complex datasets.

8.2. UGEs with Multiple Authorized Networks

We already give the experimental results on UGEs with multiple authorized networks on CIFAR-10 dataset in Table 2. In this experiments, we set two authorized networks with the same architecture (ResNet-18) but with different kinds of initialization. Here we explore deeper on the mutiple authorized networks cases.

Effect of the Number of Authorized Networks on UGEs Performance In this experiment, we investigate the impact of the number of authorized networks on the performance of UGEs. Specifically, we conduct the experiment using three authorized networks, all sharing the same architecture (ResNet-18) but initialized with different pa-

Table 7. Results on UGEs with multiple authorized networks on CIFAR-10 dataset, which are tested under three training schemes.

Method	**Scheme**	**Authorized**			**Hacker**
		Net-1	Net-2	Net-3	CNN
Original	Normal	95.01	95.01	95.06	86.57
Original	Distill-1	-	-	-	88.06
Original	Distill-2	-	-	-	88.09
original	Distill-2	-	-	-	88.14
UGEs	Normal	90.39	89.64	90.24	72.42
UGEs	Distill-1	-	-	-	73.60
UGEs	Distill-2	-	-	-	75.32
UGEs	Distill-3	-	-	-	74.32

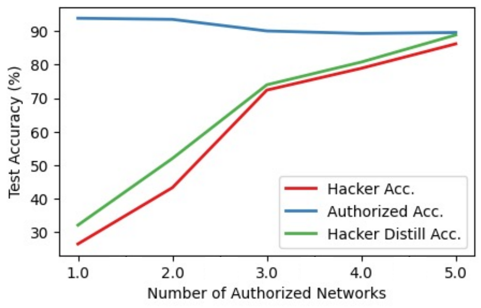

Figure 7. The performance of the proposed UGEs regarding the total number of the authorized networks. The 'Authorized Acc.' is calculated on the average test accuracy on all the authorized networks, which is similar for 'Hacker Distill Acc.'.

rameters. The experimental results are depicted in Table 2, where we can observe that:

- The effectiveness of the proposed UGEs is further demonstrated in a scenario involving three authorized networks ('Net-1', 'Net-2' and 'Net-3'). In this case, the UGEs achieve approximately 90% test accuracy on the authorized networks and around 70% test accuracy on the hacker network ('CNN').
- Nevertheless, it's important to note a slight decrease in test accuracy on the authorized networks as the number of authorized networks increases. Specifically, the test accuracy is observed to be 93.89% for one authorized network, 93.55% for two authorized networks, and 90.09% for three authorized networks. Concurrently, the test accuracies on the hacker network show an increase with the addition of more authorized networks.

Additionally, we analyze the relationship between the number of authorized networks and the corresponding test accuracies. To achieve this, we calculate the average test accuracies across multiple authorized networks, while employing the test accuracy obtained from the plain CNN as the representative hacker network. The results of this analysis are illustrated in Fig. 7.

Table 8. Results on UGEs with multiple authorized networks on CIFAR-10 dataset, which are tested under three training schemes.

Method	**Scheme**	**Authorized**		**Hacker**	
		CNN	Res-18	CNN	Res-18
Original	Normal	86.24	95.01	86.57	95.06
Original	Distill-1	-	-	87.51	95.10
Original	Distill-2	-	-	88.06	95.44
UGEs	Normal	82.95	93.08	45.26	50.43
UGEs	Distill-1	-	-	48.92	55.06
UGEs	Distill-2	-	-	48.57	54.74

Performance of UGEs on Multiple Authorized Networks with Different Architectures We further investigate the applicability of UGEs in scenarios with multiple authorized networks employing different architectures. In this experiment, we choose the plain CNN and ResNet-18 to constitute the set of authorized networks. The experiments are conducted on the CIFAR-10 dataset, and the results are detailed in Table 8.

From the table, we observe that:

- The effectiveness of our proposed UGEs extends to scenarios with multiple authorized networks employing different architectures. In this experiment, utilizing both plain CNN and ResNet-18 as authorized networks on the CIFAR-10 dataset, we observe that the test accuracy on authorized networks drops by less than 4%. Conversely, the test accuracies for hacker networks experience a significant reduction of more than 40%.
- In comparison to scenarios with multiple authorized networks sharing the same architecture, the UGEs with different architectures for authorized networks show a slight drop in performance. This discrepancy is primarily attributed to the strict trajectory alignment. Addressing this challenge presents a potential avenue for future improvements to enhance the UGE framework.

8.3. UGEs for Federated Learning

In Sec. 3.4, we assert the practicality and deployability of the proposed UGE framework across various applications. To illustrate, consider a scenario where a global server establishes the global network as f_θ, and two local servers, each possessing its distinct dataset—$\mathcal{D}^1$ for the first server and $\mathcal{D}^2$ for the second server. The global network is supposed to train on the two datasets $\mathcal{D}^1 \cup \mathcal{D}^2$, while not requiring the data shared between each server.

In this setup, both servers can independently generate their versions of UGEs with the information of the global

Table 9. Applying UGEs in federated learning setting. The experiments are conducted on CIFAR-10 dataset. We use the plain CNN and ResNet-18 as the hacker networks.

Acc.	**Normal Training**		**Federated**			**Hackers**					
	Joint	Separate	Sever1	Sever2	Global	CNN-N	CNN-H	CNN-D	Res18-N	Res18-H	Res18-D
Acc. F5	93.83	93.54	92.91	-	91.37	85.18	48.46	50.46	93.77	69.17	71.19
Acc. L5	96.27	96.50	-	96.18	95.50	86.05	49.70	58.45	96.31	70.02	73.02
Avg. Acc.	95.05	95.02	-	-	93.44	85.62	49.08	54.46	95.04	69.60	72.11

network f_θ, denoted as:

$$\begin{aligned} \mathcal{D}_u^1 &\leftarrow \arg\min_{x_u} \mathcal{L}_{all}(\mathcal{D}^1, f_\theta), \\ \mathcal{D}_u^2 &\leftarrow \arg\min_{x_u} \mathcal{L}_{all}(\mathcal{D}^2, f_\theta), \end{aligned} \tag{14}$$

where the generation of $\mathcal{D}_u^1$ and $\mathcal{D}_u^2$ involves no data interaction, ensuring data privacy within each local server, meeting the basic privacy concern of federated learning.

After each server uploads its ungeneralizable version of the data, the global model can be jointly trained as follows:

$$f: \min_\theta \frac{1}{|\mathcal{D}_u^1| + |\mathcal{D}_u^2|} \sum_{\{x_u, y_u\} \in \mathcal{D}_u^1 \cup \mathcal{D}_u^2} \mathcal{L}(f_\theta(x_u), y_u). \tag{15}$$

This optimization involves training the network using a normal training scheme with the combined datasets from both servers.

In order to test the effectiveness of UGEs applied in federated learning, we designed the experiment as follows. We selected ResNet-18 as the global model and divided the CIFAR-10 dataset into two parts. The first part includes data for the first 5 classes and is hosted by local server 1, while the second part includes data for the remaining 5 classes and is hosted by local server 2.

The experimental results are depicted in Table 9, where we compare the accuracies of the first 5 classes ('Acc. F5'), accuracies of the last 5 classes ('Acc. L5') and the average accuracy across all 10 classes ('Avg. Acc.'). pecifically, the methods for comparison include: (1) networks with normal training, trained on the total dataset $\mathcal{D}^1 \cup \mathcal{D}^2$ ('Joint') and trained on each sub-dataset separately ('Separate'); (2) networks in a federated learning setting, including the authorized network trained on $\mathcal{D}_u^1$ ('Server1'), the authorized network trained on $\mathcal{D}_u^2$ ('Server2'), and the authorized network trained on $\mathcal{D}_u^1 \cup \mathcal{D}_u^2$ ('Global'); (3) hacker networks with a CNN backbone trained with $\mathcal{D}^1 \cup \mathcal{D}^2$ ('CNN-N'), trained on $\mathcal{D}_u^1 \cup \mathcal{D}_u^2$ ('CNN-H') and that distilled from the authorized network ('CNN-D'), with a ResNet-18 backbone trained with $\mathcal{D}^1 \cup \mathcal{D}^2$ ('Res18-N'), trained with $\mathcal{D}_u^1 \cup \mathcal{D}_u^2$ ('Res18-H') and that distilled from the authorized network ('Res18-D').

From the table, we observe that:

- Our proposed UGE framework precisely aligns with the privacy requirements of federated learning, preventing shared data from being reused by third parties and prohibiting data interaction between each local server.
- The generated UGEs not only work when locally training the network f_θ ('Sever1'& 'Sever2' with less than 1% accuracy drop), but also when jointly training the global network f_θ ('Global' with less than 2% accuracy drop).
- The generated UGEs effectively prevent reuse by hacker networks, resulting in reduced accuracies on CNN from 85.62% to 49.08% and on ResNet-18 from 95.04% to 69.60%. Additionally, it mitigates information leakage from the global network, as the network after distillation shows a relatively low accuracy compared to normal training (a 20% to 30% accuracy drop).