Supplementary Material for Reconstruct Private Data via Public Knowledge in Distillation-based Federated Learning

Anonymous Author(s)
Affiliation
Address
email

A Proofs

Prop. 1

Proof. Since h is differentiable, we have the following equations;

$$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial y_i}\frac{\partial y_i}{\partial b_i} = \frac{\partial L}{\partial y_i} h^{(1)}(Az+b)_i, \quad \frac{\partial y_i}{\partial A_i} = h^{(1)}(Az+b)_i z^T \tag{S-1}$$

, where i is the index of A's row. From the above equations, we can analytically determine z from $\frac{\partial L}{\partial b_i}$ and $\frac{\partial L}{\partial A_i}$ as follows;

$$z^T = \frac{1}{h^{(1)}(Az+b)}\frac{\partial y_i}{\partial A_i} = \frac{1}{h^{(1)}(Az+b)}\frac{\partial y_i}{\partial L}\frac{\partial L}{\partial A_i} = \frac{\partial L}{\partial A_i} / \frac{\partial L}{\partial b_i} \tag{S-2}$$

Then, if we think the neural network as a Markov chain $x \to z \to y$, the data processing inequality [1] leads to Inequal. 4;

$$I(x; \frac{\partial L}{\partial A}, \frac{\partial L}{\partial b}) \geq I(x; z) \geq I(x; y) \tag{S-3}$$

□

Lemma. 1

Proof. We optimize Eq. 5 independently for p_{c_i} and p_s. Given the range of probability, the optimal p_{c_i} is obviously as follows;

$$\hat{p}'_{c_i,k} = \begin{cases} 1 & (k = j) \\ 0 & (k \neq j) \end{cases}$$

Next, we use Lagrange multipliers to find the optimal p_s under the constraint of its sum equal to one.

$$\max_{p'_s} p'_{s,j} + \alpha H(p'_s) \tag{S-4}$$

$$s.t. \sum_{k=1}^{J} p'_{s,k} = 1 \tag{S-5}$$

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

We require that

$$\frac{\partial}{\partial p'_s}\{p'_{s,j} + \alpha H(p'_s) + \lambda(\sum_{k=1}^{J} p'_{s,k} - 1)\}|_{p'_s = \hat{p}'_s} = 0 \tag{S-6}$$

, which gives a system of J equations, $k = 1, ..., J$, such that,

$$\frac{\partial}{\partial p'_{s,k}}\{p'_{s,j} + \alpha(\sum_{k=1}^{J} -p'_{s,k}\log p'_{s,k}) + \lambda(\sum_{k=1}^{J} p'_{s,k} - 1)\}|_{p'_s = \hat{p}'_{s,k}} = 0 \tag{S-7}$$

Eq. S-7 yields

$$\begin{cases} 1 - \alpha(\log \hat{p'}_{s,k} + 1) + \lambda = 0 & (k = j) \\ \quad -\alpha(\log \hat{p'}_{s,k} + 1) + \lambda) = 0 & (k \neq j) \end{cases} \tag{S-8}$$

Then, we have that

$$\lambda = \alpha(\log \hat{p}'_{s,k} + 1) \tag{S-9}$$

$$\hat{p}'_{s,k} = \begin{cases} \exp(\frac{1+\lambda-\alpha}{\alpha}) & (k = j) \\ \exp(\frac{\lambda-\alpha}{\alpha}) & (k \neq j) \end{cases} \tag{S-10}$$

Eq. S-5 and Eq. S-10 indicate

$$\begin{gathered} \{J - 1 + \exp(\frac{1}{\alpha})\}\exp(\frac{\lambda - \alpha}{\alpha}) = 1 \\ \therefore \lambda = \alpha \log\{\frac{1}{J - 1 + \exp(\frac{1}{\alpha})}\} + \alpha \end{gathered} \tag{S-11}$$

Combining Eq. S-10 and Eq. S-11, we finally have

$$\hat{p}^{'}_{s,k} = \begin{cases} \frac{\sqrt[\alpha]{e}}{J-1+\sqrt[\alpha]{e}} & (k = j) \\ \frac{1}{J-1+\sqrt[\alpha]{e}} & (k \neq j) \end{cases}$$

$\square$

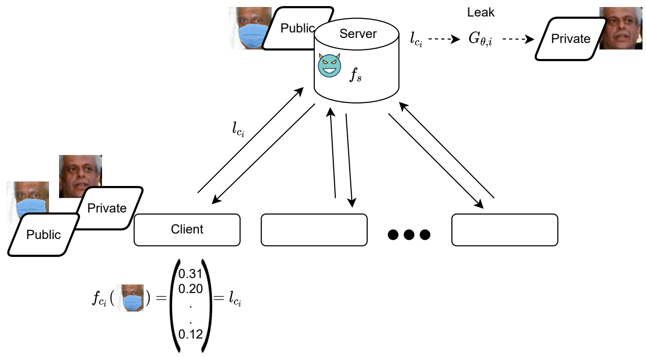


Figure S-1: The overview of our scenario

B Protocols and architectures

Algorithm S-1 FedMD

Input: Private dataset $D_{pri} = \{\boldsymbol{X}_{pri}, \boldsymbol{y}_{pri}\}$, public dataset $D_{pub} = \{\boldsymbol{X}_{pub}, \boldsymbol{y}_{pub}\}$, local model $f_{c_i, i=1\ldots C}$, global model f_s, number of communications T.

1: Train f_{c_i} on D_{public}
2: Train f_{c_i} on D_{pri}
3: **for** $t = 1 \leftarrow T$ **do**
4: $\quad c_i$ sends $l_{c_i} = f_{c_i}(\boldsymbol{X}_{pub})$
5: $\quad \tilde{l} = \frac{1}{C}\sum_{i=1}^{C} l_{c_i}$ ▷ The server computes the consensus logits.
6: $\quad$ Train f_{c_i} on $\{\boldsymbol{X}_{pub}, \tilde{l}\}$
7: $\quad$ Train f_{c_i} on D_{pri}
8: $\quad$ Train f_s on D_{public}
9: **end for**

Algorithm S-2 FedGEMS

Input: Private dataset $D_{pri} = \{\boldsymbol{X}_{pri}, \boldsymbol{y}_{pri}\}$, public dataset $D_{pub} = \{\boldsymbol{X}_{pub}, \boldsymbol{y}_{pub}\}$, local model $f_{c_i, i=1\ldots C}$, global model f_s, number of communications T.

1: **for** $t = 1 \leftarrow T$ **do**
2: $\quad$ Selectively train f_s on $\{D_{pub}, \tilde{l}, l_{c_i}\}$, where $l_{c_i} = f_{c_i}(\boldsymbol{X}_{pub})$
3: $\quad \tilde{l} = f_s(\boldsymbol{X}_{pub})$
4: $\quad$ Train f_{c_i} on $\{D_{pub}, \tilde{l}\}$
5: $\quad$ Train f_{c_i} on D_{pri}
6: **end for**

Algorithm S-3 DSFL

Input: Private dataset $D_{pri} = \{\boldsymbol{X}_{pri}, \boldsymbol{y}_{pri}\}$, public dataset $D_{pub} = \{\boldsymbol{X}_{pub}, \boldsymbol{y}_{pub}\}$, local model $f_{c_i, i=1\ldots C}$, global model f_s, number of communications T.

1: **for** $t = 1 \leftarrow T$ **do**
2: $\quad$ Train f_{c_i} on $\{D_{pri}\}$
3: $\quad c_i$ sends $l_{c_i} = f_{c_i}(\boldsymbol{X}_{pub})$
4: $\quad \tilde{l} = \text{ERA}(\sum_{i=1}^{C} \frac{l_{c_i}}{C})$ ▷ The server computes the consensus logits.
5: $\quad$ Train f_{c_i} on $\{\boldsymbol{X}_{pub}, \tilde{l}\}$
6: $\quad$ Train f_s on $\{\boldsymbol{X}_{pub}, \tilde{l}\}$
7: **end for**

Fig. S-1 shows the overview of our scenario, where the malicious server tries to reconstruct the private data via the output logits of the public data with the inversion model. Alg. S-1, S-2, and S-3 are the pseudo-codes of each protocol, where we additionally train the server-side model on the public dataset at line 9 in FedMD. Code. 1 and 2 are the implementation of server-side, client-side, and inversion models.

Code 1: server and local models

```
nn.Sequential(
    nn.Conv2d(3, 32, kernel_size=(3, 3), stride=1, padding=0),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=(3, 3), stride=None, padding=0),
    nn.Flatten(),
    nn.Linear(12800, output_dim),
)
```

Code 2: inversion model

```
nn.Sequential(
    nn.ConvTranspose2d(input_dim, 1024, (4, 4), stride=(1, 1)),
```

```
    nn.BatchNorm2d(1024),
    nn.Tanh(),
    nn.ConvTranspose2d(1024, 512, (4, 4),
                       stride=(2, 2), padding=(1, 1)),
    nn.BatchNorm2d(512),
    nn.Tanh(),
    nn.ConvTranspose2d(512, 256, (4, 4),
                       stride=(2, 2), padding=(1, 1)),
    nn.BatchNorm2d(256),
    nn.Tanh(),
    nn.ConvTranspose2d(256, 128, (4, 4),
                       stride=(2, 2), padding=(1, 1)),
    nn.BatchNorm2d(128),
    nn.Tanh(),
    nn.ConvTranspose2d(128, channel, (4, 4),
                       stride=(2, 2), padding=(1, 1)),
    nn.Tanh()
)
```

C Gradient inversion attack

Algorithm S-4 Gradient inversion attack

Input: The number of communication T, the target model F, the number of clients C, the number of classes J, the number of classes of each private dataset $\{J_i\}_{i=1...C}$, the dimension of input d.
Output: Reconstructed data $\{X'_i \in \mathbb{R}^{d \times J_i}\}_{i=1...C}$

for $t = 1 \leftarrow T$ **do**
  for $i = 1 \leftarrow C$ **do**
    Receive ∇W_i from client c_i.
    if $t == 1$ **then**
      Infer Y_i, the unique labels of c_i's private dataset.
      $X'_i \in \mathbb{R}^{d \times J_i} \leftarrow \mathcal{N}(0, 1)$
    end if
    for $m = 1 \leftarrow M$ **do**
      $\nabla W'_i \leftarrow \frac{\partial \ell(f(X'_i, W_i), Y_i)}{\partial W_i}$
      $X'_i \leftarrow X'_i - \eta \nabla_{X'_i} L_{GB}(X'_i)$
    end for
  end for
end for
return $\{X'_i\}_{i=1...C}$

Although the existing gradient inversion methods focus on reconstructing the exact batch data and labels, our interest is in recovering the class representation of the private training dataset. Then, we view that the received gradient ∇W_i is calculated with $X_i \in \mathbb{R}^{J_i \times d}$, where X_i represents the class representations of client c_i's private dataset, J_i is the number of unique classes of the dataset, and d is the dimension of the input data. The attacker can infer the labels used to train the local model from the received gradient with the batch label restoration method proposed in [2]. Then, we optimize dummy class representations $X'_i \in \mathbb{R}^{J_i \times d}$ with the following cost function;

$$L_{GB}(X'_i) = 1 - \frac{\langle \nabla W'_i, \nabla W_i \rangle}{||\nabla W'_i|| ||\nabla W_i||} + \gamma \mathrm{TV}(X'_i) \tag{S-12}$$

, where TV denotes the total variation and γ is its coefficient. This cost function is the same as the one used in [3]. Note that unlike our proposed attack against FedKD, the attacker must know the number of unique labels in each local dataset in advance. In our experiments, we set γ to 0.01, and use Adam optimizer with learning rate of 0.3.

D Result details

D.1 Result of 4.2

Tab. S-1, S-2, and S-3 show the specific numerical results for 4.2. Fig. S-2 represents the loss of PTBI and TBI in each epoch, which indicates that the relationship between inversion loss and temperature is consistent with attack success rate and temperature. Note that we train the inversion model 3 epochs per communication in a total of 5 communications. Fig. S-3 shows the percentage of reconstructed images whose closest image is public data belonging to the target label, which indicates that PTBI tends not to reconstruct public data compared to TBI. Fig. S-4 and S-5 are the examples of reconstructed images with PTBI and TBI. Fig. S-6 and Fig. S-7 also show some images that PTBI reconstructs with different τ.

Table S-1: Attack success rate ($\tau = 3.0$)

		DSFL		FedMD		FedGEMS	
	dataset	LFW	LAG	LFW	LAG	LFW	LAG
C	attack						
1	TBI	91.0%	90.5%	21.5%	19.0%	0.0%	1.0%
	PTBI	89.5%	92.0%	59.5%	51.0%	9.0%	10.5%
10	TBI	22.0%	36.5%	8.5%	10.5%	9.0%	4.0%
	PTBI	2.5%	8.0%	45.0%	41.5%	26.0%	18.5%

Table S-2: Attack success rate ($\tau = 1.0$)

		DSFL		FedMD		FedGEMS	
	dataset	LFW	LAG	LFW	LAG	LFW	LAG
C	attack						
1	TBI	59.0%	92.5%	9.0%	5.0%	0.0%	0.0%
	PTBI	68.5%	94.5%	38.5%	20.0%	0.0%	0.0%
10	TBI	67.0%	38.5%	13.0%	5.0%	0.0%	0.0%
	PTBI	70.0%	54.5%	21.0%	9.0%	0.0%	0.0%

Table S-3: Attack success rate ($\tau = 0.3$)

		DSFL		FedMD		FedGEMS	
	dataset	LFW	LAG	LFW	LAG	LFW	LAG
C	attack						
1	TBI	16.0%	73.5%	5.0%	5.0%	0.0%	0.0%
	PTBI	12.0%	83.0%	14.0%	6.5%	0.0%	0.0%
10	TBI	84.5%	63.0%	2.0%	2.0%	0.0%	0.0%
	PTBI	88.5%	59.0%	2.5%	4.5%	0.0%	0.0%

D.2 Ablation studies

We do ablation studies with the same setting as 4.2 of $C = 10$, where we remove each term of Q. Tab. S-4 shows the optimal $p'_{c_i,k}$ and $p'_{s,k}$ in each ablations. In the case of $Q = p'_{s,j} + \alpha H(p'_s)$, we do not need the client-side model, so we train an inversion model only with the global logits, where the architecture of the inversion model is the same as that of TBI.

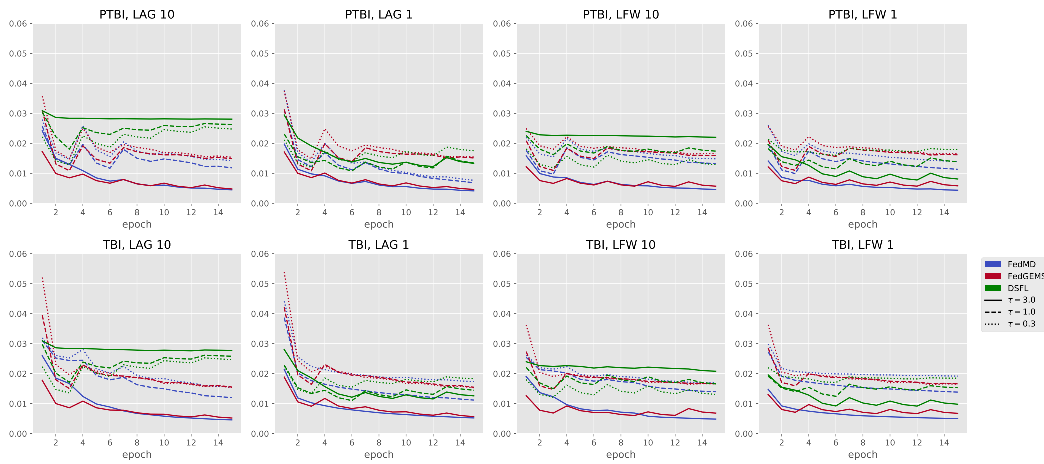


Figure S-2: Inversion loss

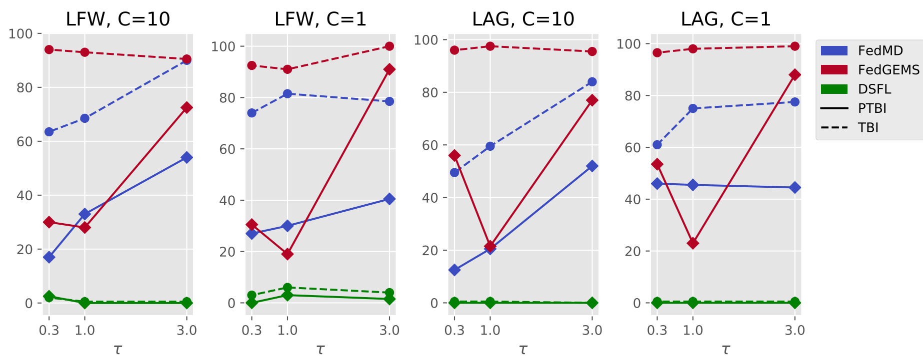
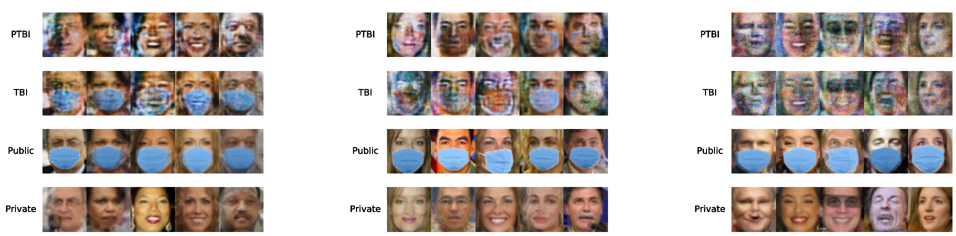


Figure S-3: Percentage of reconstructed images whose closest image is public data



(a) FedMD (b) FedGEMS (c) DSFL

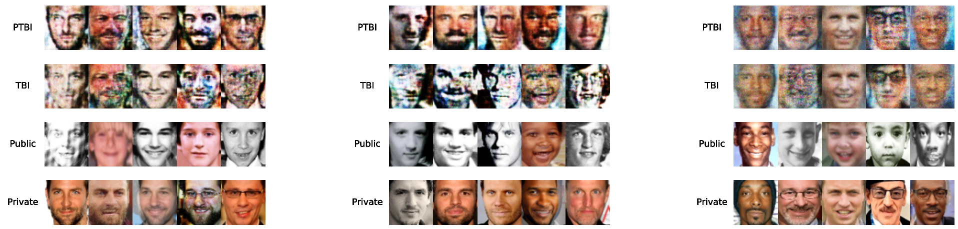
Figure S-4: LFW: example of reconstructed images

D.3 Impact of public dataset size

Fig. S-8 shows the results of the experiments with the smaller public dataset, which consists of 400 celebrities.

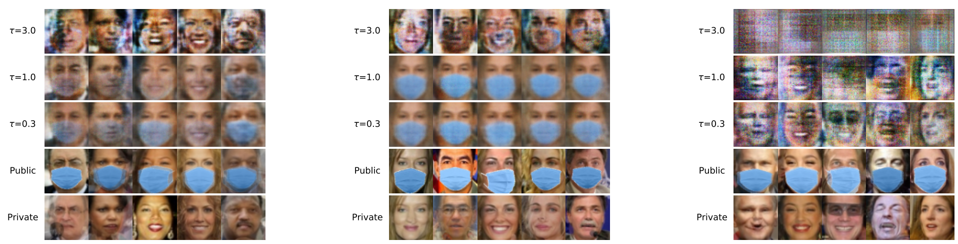
References

[1] Thomas M Cover. Elements of information theory. John Wiley & Sons, 1999.



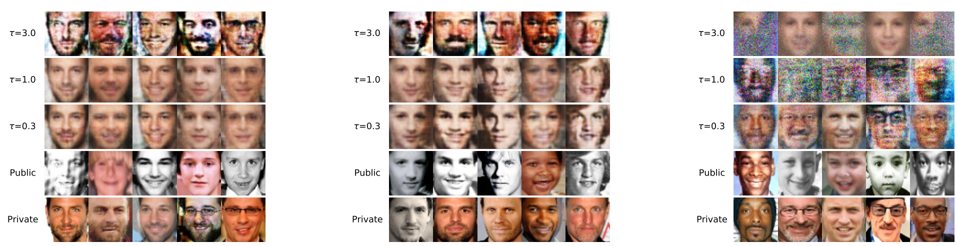
(a) FedMD (b) FedGEMS (c) DSFL

Figure S-5: LAG: example of reconstructed images



(a) FedMD (b) FedGEMS (c) DSFL

Figure S-6: LFW: example of reconstructed images with different τ



(a) FedMD (b) FedGEMS (c) DSFL

Figure S-7: LAG: example of reconstructed images with different τ

Table S-4: Optimal Q in each ablation study

Q	$\hat{p}'_{c_i,k}$	$\hat{p}'_{s,k}$
$p'_{c_i,j} + p'_{s,j}$	$\begin{cases} 1 & (k = j) \\ 0 & (k \neq j) \end{cases}$	$\begin{cases} 1 & (k = j) \\ 0 & (k \neq j) \end{cases}$
$p'_{c_i,j} + \alpha H(p'_s)$	$\begin{cases} 1 & (k = j) \\ 0 & (k \neq j) \end{cases}$	$\frac{1}{J}$
$p'_{s,j} + \alpha H(p'_s)$	-	$\begin{cases} \frac{\sqrt[\alpha]{e}}{J-1+\sqrt[\alpha]{e}} & (k = j) \\ \frac{1}{J-1+\sqrt[\alpha]{e}} & (k \neq j) \end{cases}$

[2] 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, pages 16337–16346, 2021.

[3] 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,

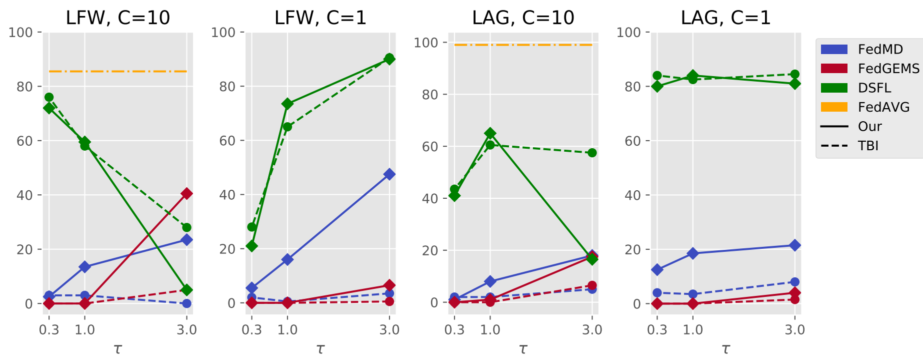


Figure S-8: Attack success rate with small public dataset

33:16937–16947, 2020.