A THE DERIVATION OF $\nabla z_i = \dfrac{\partial \mathcal{L}}{\partial z_i} = p_i - y_i$

According to the definitions and notations given in Section 2.1, we have $\mathbf{z} = \mathbf{W}\mathbf{e} + \mathbf{b}$ and $\mathbf{p} = SoftMax(\mathbf{z})$ to model the mapping of the final layer and post-softmax process. First, the softmax function defines a transformation that $p_i = SoftMax(\mathbf{z})_i = \dfrac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, j = 1, ..., C$. We denote $\sum_{j=1}^{C} e^{z_j}$ as $\sum_C$ for convenience and discuss $\dfrac{\partial p_i}{\partial z_j}$ by situation:

$$\frac{\partial p_i}{\partial z_j} = \begin{cases} \dfrac{e^{z_i}\sum_C - e^{z_i}e^{z_i}}{{\sum_C}^2} = \dfrac{e^{z_i}}{\sum_C} - \dfrac{{e^{z_i}}^2}{\sum_C} = p_i - {p_i}^2, & i = j, \\ \dfrac{0 \times \sum_C - e^{z_i}e^{z_j}}{{\sum_C}^2} = -\dfrac{e^{z_i}}{\sum_C} - \dfrac{e^{z_j}}{\sum_C} = -p_i p_j, & i \neq j, \end{cases} \tag{7}$$

where i and j are both members of $\mathbb{C} = \{1, ..., C\}$. And the cross-entropy loss can be formalized as $\mathcal{L} = CE(\mathbf{p}, \mathbf{y}) = -\sum_{i=1}^{C} \mathrm{y}_i log(p_i) = -log(p_c)$ (c represents the index of the ground-truth class), which means $\dfrac{\partial \mathcal{L}}{\partial p_i} = \dfrac{\partial \mathcal{L}}{\partial p_c} = -\dfrac{1}{p_c}$ only if $i = c$ otherwise 0. Therefore, using the chain rule, $\dfrac{\partial \mathcal{L}}{\partial z_i}$ can be calculated as the following formula:

$$\frac{\partial \mathcal{L}}{\partial z_i} = 0 + \frac{\partial \mathcal{L}}{\partial p_c} \times \frac{\partial p_c}{\partial z_i} = \begin{cases} -\dfrac{1}{p_c} \times (p_c - {p_c}^2) = p_c - 1, & i = c, \\ -\dfrac{1}{p_c} \times (-p_c \times p_i) = p_i, & i \neq c. \end{cases} \tag{8}$$

Merging the two branches, we get our conclusion that $\nabla z_i = \dfrac{\partial \mathcal{L}}{\partial z_i} = p_i - y_i$.

B THE ARCHITECTURE OF FCN-3

See Table 2.

Table 2: The details of shape for each FCN-3 layer.

Layer	**Shape (In-Out)**
Input	784-300
Hidden	300-300
Output	300-10

C ROBUSTNESS TO BATCHNORM AND DROPOUT OPERATIONS

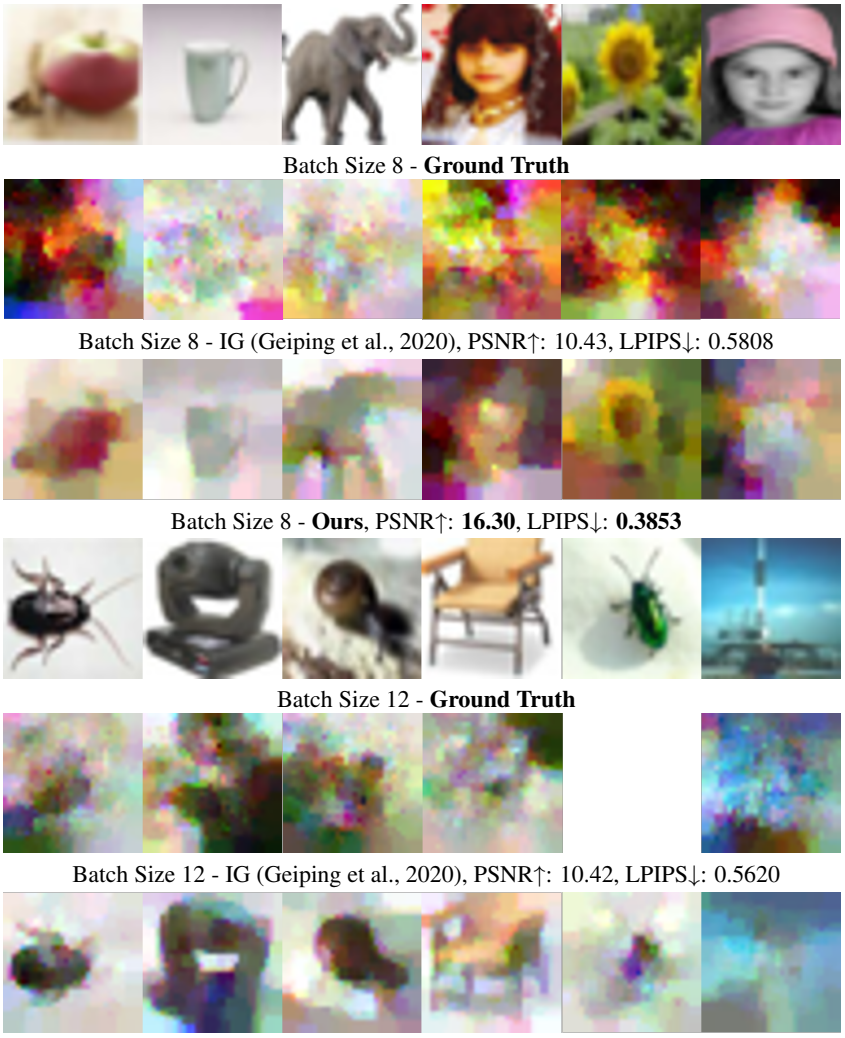
Our method is sufficiently robust for current neural networks with certain special components. Here, we investigate the possible impact of BatchNorm (Ioffe & Szegedy, 2015) and DropOut (Srivastava et al., 2014) operations, which are applied to LeNet-5 and VGG-16 respectively. We perform attacks on the CIFAR100 dataset with LeNet-5 in a batch of size 128 and the ImageNet dataset with VGG-16 in a batch of size 64. From Table 3, we can observe that the DropOut and BatchNorm operations do not cause degradation of performance.

Table 3: Comparison of attack effects on whether the two components are applied in the model. WO: without; WI: with; BN: batchnorm; DO: dropout.

	LeNet-5 on CIFAR100		VGG-16 on ImageNet	
	LeAcc	LnAcc	LeAcc	LnAcc
WO BN/DO	1.000	1.000	1.000	1.000
WI BN/DO	1.000	1.000	1.000	1.000

D More Visualization Examples

As shown in Fig 6, we offer additional visual examples to illustrate our improvements of gradient inversion attacks.



Batch Size 8 - **Ground Truth**

Batch Size 8 - IG (Geiping et al., 2020), PSNR↑: 10.43, LPIPS↓: 0.5808

Batch Size 8 - **Ours**, PSNR↑: **16.30**, LPIPS↓: **0.3853**

Batch Size 12 - **Ground Truth**

Batch Size 12 - IG (Geiping et al., 2020), PSNR↑: 10.42, LPIPS↓: 0.5620

Batch Size 12 - **Ours**, PSNR↑: **17.52**, LPIPS↓: **0.3641**

Figure 6: Additional contrast examples of inverting ResNet-18 gradients on CIFAR100 to demonstrate our improvements.

E PSEUDO-CODE OF OUR ALGORITHM

Algorithm 1 provides a pseudo-code for the complete procedure of our method.

Algorithm 1 Class-Wise Embeddings Inference and Instance-Wise Batch Label Restoration.

Input: Gradients of weight and bias in the final layer $\nabla \mathbf{W} \in \mathbb{R}^{C \times m}$, $\nabla \mathbf{b} \in \mathbb{R}^{C}$.
Output: Class-wise averaged embeddings $\mathbb{E} = \{\mathbf{e}^i, i = 1, 2, .., C\}$ and the number of occurences for each class $\mathbb{N} = \{\mathbf{n}^i, i = 1, 2, .., C\}$ in a training batch.
1: Initial $\mathbb{E} = \emptyset$ and $\mathbb{N} = \emptyset$.
2: **for** $i = 1$ to C **do**
3: Calculate $\mathbf{e}^i$ using $\nabla \mathbf{W}$ and $\nabla \mathbf{b}$;
4: Feed $\mathbf{e}^i$ into the final layer to get the network outputs $\mathbf{z}^i$ and post-softmax probabilities $\mathbf{p}^i$;
5: Add $\mathbf{e}^i$ into $\mathbb{E}$;
6: **end for**
7: Solve the system of linear equations in (6) to obtain $\mathbb{N}$.
8: **return** $\mathbb{E}$ and $\mathbb{N}$.

F ADAPTABILITY TO DIFFERENT DATA DISTRIBUTIONS

Our method mainly relies on the two somewhat contradictory properties, i.e., the intra-class high similarity and inter-class low entanglement. In a training batch, if we increase the number of classes, the entanglement is usually elevated as well. However, when the batch size is fixed, the number of instances of per class will be reduced, and the similarity may be improved. In extreme cases, each class only includes one instance, the similarity will reach 100%. Therefore, we propose to discuss the impacts of two determinants of the data distribution on attack effects: (1) *Max Repetitions of A Single Class-MRoC*: the maximal number of instances within a single class; (2) *Number of Classes-NoC*: the number of classes. We execute attacks on CIFAR100 with LeNet-5 model in a large batch of size 128. And we divide the cases discussed below into extreme, balanced, and unique distribution configurations. For unique distribution, the batch size can only be up to 100 for the limit of NoC on CIFAR100. The results are shown in Table 4, which fully illustrate the adaptability of iRLG to different data distributions.

Table 4: Average accuracy scores under different distribution configurations.

Distribution Configuration		LeAcc	LnAcc
Extreme	NoC=1, MRoC=128	1.000	1.000
Balanced	NoC=2, MRoC=64	1.000	1.000
	NoC=4, MRoC=32	1.000	1.000
	NoC=8, MRoC=16	1.000	1.000
	NoC=16, MRoC=8	1.000	1.000
	NoC=32, MRoC=4	1.000	1.000
	NoC=64, MRoC=2	1.000	1.000
Unique	NoC=100, MRoC=1	0.998	0.911