Maximum Knowledge Orthogonality Reconstruction with Gradients in Federated Learning

Supplementary Material

Feng Wang, Senem Velipasalar, and M. Cenk Gursoy
EECS department, Syracuse University, Syracuse, NY, 13244.
{fwang26, svelipas, mcgursoy}@syr.edu

1. Additional Proof

In this section, we provide detailed explanations, interpretations, and proof of the mathematical content for comprehensive insight. We list all the notations used in the paper in Table 1.

We derive **Eqn. (2)** included in our main paper as shown below, where $O(\frac{1}{K})$ denotes order K^{-1} complexity.

$$
\begin{aligned}
& H(\mathbf{g}_1) - H\left(\mathbf{g}_1 | \sum_{k=1}^{K} \mathbf{g}_k\right) \\
&= H\left(\sum_{k=1}^{K} \mathbf{g}_k\right) - H\left(\sum_{k=1}^{K} \mathbf{g}_k | \mathbf{g}_1\right) \\
&= H\left(\sum_{k=1}^{K} \mathbf{g}_k\right) - H\left(\sum_{k=2}^{K} \mathbf{g}_k\right) \\
&= \frac{1}{2}\log(2\pi K\sigma_g^2) + \frac{1}{2} - \frac{1}{2}\log(2\pi(K-1)\sigma_g^2) - \frac{1}{2} \\
&= \frac{1}{2}\log\left(1 + \frac{1}{K-1}\right) \\
&= \frac{1}{2(K-1)} + \frac{1}{2}\left(\frac{1}{2(K-1)}\right)^2 + \dots \\
&= O\left(\frac{1}{K}\right).
\end{aligned}
$$

We first rewrite **Eqn. (4)** from the main paper as below:

$$
\frac{\partial \mathcal{L}(\mathbf{x}_k, y_k \,|\, \mathbf{A})}{\partial \mathbf{W}^1} = \frac{\partial \mathcal{L}(\mathbf{x}_k, y_k \,|\, \mathbf{A})}{\partial \mathbf{y}_k} \frac{\partial \mathbf{y}_k}{\partial \mathbf{W}^1} = \frac{\partial \mathcal{L}(\mathbf{x}_k, y_k \,|\, \mathbf{A})}{\partial \mathbf{b}^1} \mathbf{z}_k^{1\mathrm{T}}.
$$

We note that the first equality is derived from the chain rule. To prove the second equality, we first clarify that an FC classifier with only one layer has no activation function, therefore $\mathbf{y}_k = \mathbf{W}^1\mathbf{z}_k^1 + \mathbf{b}^1$. Hence, $\frac{\partial \mathcal{L}(\mathbf{x}_k, y_k \,|\, \mathbf{A})}{\partial \mathbf{y}_k} = \frac{\partial \mathcal{L}(\mathbf{x}_k, y_k \,|\, \mathbf{A})}{\partial \mathbf{b}^1}$. Additionally, the partial derivative $\frac{\partial \mathbf{y}_k}{\partial \mathbf{W}^1}$ is simply the transpose of the multiplier $\mathbf{z}_k^{1\mathrm{T}}$. For the multiple layer case as we claimed in **Eqn. (9)**, we may consider a dummy vector $\mathbf{z}'_k$ before activation f, i.e. $\mathbf{z}_k^2 = f(\mathbf{z}'_k) = f(\mathbf{W}^1\mathbf{z}_k^1 + \mathbf{b}^1)$.

Table 1. Notation

FL Parameters:	Section 2, 3.1
U, u	Total number and index of clients
K, k	Batch size and local sample index
$\mathbf{x}_k$, y_k	The k^{th} input image and label
$\mathbf{A}$	Network parameters (weights and biases)
$\mathcal{L}(\mathbf{x}, y \,\|\, \mathbf{A})$	Loss function
$\mathbf{G}_k$	Gradient of the k^{th} image
N	Total number of labels
n	Index of node in the corresponding layer
MKOR - FC:	Section 3.2
$\mathbf{p} = \{p_1, \dots, p_N\}$	Local sample distribution over classes
L, l	Total number of layers and layer index
N^l	Total number of input nodes
$\mathbf{W}^l$, $\mathbf{b}^l$	Weight and bias at FC layer l
$\mathbf{z}_k^l$	Input from sample k at FC layer l
$\mathbf{y}_k$	Classifier output from sample k
$\hat{\mathbf{z}}_n^1$	Reconstructed input of label n
MKOR - conv:	Section 3.4
$\mathbf{z}^0$	Output of convolutional layers before flatten
I, i	Total number and index of 4-direction layers
J, j	Total number and index of max-pooling layer with only "copy" layer
h, w, c	Height, width and channel index
$\mathbf{R}[h, w, c]$	Considered region in $\mathbf{x}$ for $\mathbf{z}^0[h, w, c]$
$\hat{\mathbf{x}}_n$	Reconstructed input image of label n

Thus, we can substitute $\mathbf{z}'_k$ for $\mathbf{y}_k$ in the analysis above, and get the same result as shown in Eqn. (9).

For the special case of an FC classifier without bias term, we have to set every connected weight in Fig. 1 of the main paper to the same positive value. Thus, for each input sample, all elements of the vector $\frac{\partial \mathcal{L}(\mathbf{x}_k, y_k \,|\, \mathbf{A})}{\partial \mathbf{y}_k}$ will have the same value. Therefore, although we do not have this vector indicating the magnitude of the reconstructed image, we still have a rough estimation that is uniformly brighter or darker than the original input with the same ratio between pixels, and it is usually sufficient to identify all key features. Nevertheless, for the majority of benchmark models, we usually have the bias term.

The intuition behind **Equations (10), (11), (12) and (13)** is to decouple one node for each of the N labels in the first $2N$ nodes in the output of the first FC layer $\mathbf{z}^2$. Such decoupling is similar to the naive case shown in Fig. 1, except that each output node is coupled to two different nodes in the output in the second layer. As shown in Eqn. (10), for each pair of neighboring nodes of $\mathbf{z}^2$, the weight in the first layer to one node is the opposite of the weight to the other node. Therefore in the forward-propagation, one node with negative output will be zeroed out by ReLU activation while another with positive output will be kept. If we alternatively consider activation that does not zero out negative values such as Sigmoid, we can use only one line of nodes for each label. In the case where activation zeroes out negative values such as ReLU, we note in Eqn. (11) that the bias in the first layer to each decoupled node should be modified accordingly, to avoid zeroing out necessary forward-propagation paths.

Eqn. (12) specifies the weights in the second FC layer and the following FC layers to ensure that the target N nodes in the first $2N$ nodes of $\mathbf{z}^2$ are decoupled. In the second FC layer, we merge every two neighboring nodes in the first $2N$ nodes of the input $\mathbf{z}^2$ to a single node in the first N nodes of the output $\mathbf{z}^3$. According to the pair of opposite weights in the first layer, for each pair of nodes in the first $2N$ nodes of $\mathbf{z}^2$, there will be one node with positive value and the other with zero value after ReLU activation. Therefore, we set the weight of the merging path to positive values and the rest to minor Gaussian noise, so that the back-propagation path will go back to one of the decoupled pair of nodes with positive output. For the majority of FC layers, $N \ll N^l$ and such modification can be hardly detected. For the following layers (i.e., 3rd, 4th, etc), we keep a single positive weight for each decoupled node until the output. **Eqn. (13)** specifies that the bias to the decoupled nodes is non-negative in order to prevent the output being zeroed out by ReLU activation.

To explain the convolutional block reconstruction, we first illustrate how the forward-propagation in **Fig. 2** works. We denote the input nodes as $\mathbf{x}$, and the output nodes of each convolutional layers as $\mathbf{x}^1$, $\mathbf{x}^2$, etc. For a typical RGB image, we have three input channels in $\mathbf{x}$. After the forward-propagation, the first layer of type ① divides the information of each input channel to a positive copy and a negative copy. That is to say, channel 1 in $\mathbf{x}$ is exactly copied to channel 1 in $\mathbf{x}^1$ which is called a "copy" filter, and a negative copy $1 - \mathbf{x}$ is copied to channel 2 in $\mathbf{x}^1$ which is called a "min" filter. The benefit of such division is to keep both maximum values and minimum values after the max-pooling layer. Similarly, channel 2 in $\mathbf{x}$ is divided into channels 3 and 4 in $\mathbf{x}^1$, and channel 3 in $\mathbf{x}$ is divided into channels 5 and 6 in $\mathbf{x}^1$. Therefore, all the information in $\mathbf{x}$ is kept in the first 6 channels in $\mathbf{x}^1$, and we do not care about the rest of the channels in $\mathbf{x}^1$. The second layer of type ② copies the 6 channels to the first 6 channels in $\mathbf{x}^2$. The third layer of type ③ also copies the 6 channels to $\mathbf{x}^3$, but it also has a 2×2 max-pooling. As a result, each node in channel 1 of $\mathbf{x}^3$ denotes the maximum value of a 2×2 area in channel 1 of $\mathbf{x}$, and each node in channel 2 of $\mathbf{x}^3$ denotes the minimum value of a 2×2 area in channel 1 of $\mathbf{x}$. Similarly, channels 3 to 6 of $\mathbf{x}^3$ represents the maximum and minimum values of channels 2 and 3 of $\mathbf{x}$. While we lost a half of the width and a half of the height in $\mathbf{x}^3$ and the amount of information is reduced to $\frac{1}{4}$ in each channel, we also have many more channels to utilize. To store the input information in more different channels, we in the fourth layer of type ④ divide each channel in $\mathbf{x}^3$ into 4 different channels in $\mathbf{x}^4$ in 4 different directions. Specifically, the output channel with "copy" filter focuses on the same area of input $\mathbf{x}^3$, and the output channel with "right" filter, "lower" filter or "lower right" filter focuses on a switched area of the input $\mathbf{x}^3$. Thus, these 4 channels represent 4 different areas in the input. These areas are similar, but slightly switched. The fifth layer and the sixth layer have the same function as the third layer and the fourth layer, and the considered region is further switched. Finally, the last layer max-pools and flattens the output. Given the two switches with layer type ④, each output node that corresponds to the considered channels maps to a different considered region in the input $\mathbf{x}$, and our input reconstruction relies on these switches.

With the aforementioned idea of Fig. 2 in mind, we then explain the rest of the equations. In **Eqn. (14)**, we consider the 6 output channels that are not shifted, i.e., all corresponding filters are "copy" filters. Each element in these 6 output channels corresponds to a certain "considered region" in the input, which is defined in Eqn. (14). For the rest of the output channels with at least one switch, i.e., corresponding to at least one of the "right" filter, "lower" filter or "lower right" filter, we define the shifted distance at the input image $\mathbf{x}$ in **Eqn. (15)**. Given the shifted distance, we express the shifted considered region for arbitrary output channel in **Eqn. (16)**, which is a generalized version of Eqn. (14). On the one hand, a half of the channels experienced a "copy" filter at the first layer thus describing the maximum value of the considered region, while the others experienced a "min" filter at the first layer thus describing the minimum value of the considered region. On the other hand, since each output node describes a considered region in the input, each node of the input is also described by multiple output nodes. Therefore, in **Eqn. (17)**, the upper bound of an input node is defined as the minimum of the maximum describers. In **Eqn. (18)**, the lower bound of an input node is defined as the maximum of the minimum describers. Without further information, we estimate the input node by the average between the upper bound and the lower bound as in **Eqn. (19)**. For the inconspicuous approach,

the indices of the channels should be randomly shuffled. In **Eqn. (20)**, we can multiply the weight and bias of the filters in each layer by a different positive factor β, and simply divide by different βs during reconstruction.

2. Additional Examples

In this section, we provide comparison between experimental settings and additional examples with larger images.

In our inconspicuous reconstruction approaches, we proposed multiple strategies such as reshuffling of indices and magnitude manipulation to reduce the detectability by the clients. Furthermore, even if we only focus on the ratio of the modified parameters, MKOR is still much more inconspicuous than the existing approaches. In MKOR for VGG16, the layer with highest ratio of modification is the last conv layer. It has 2359808 parameters, and 332160 parameters are modified, so the peak modification ratio is 14%. Furthermore, the overall modification ratio for all layers in VGG16 is less than 1%, and every image generates gradient of similar magnitude. In contrast, class fishing [2] sets 99.9% of parameters in the last layer to zero, and input images from 99.9% of the labels do not generate any gradient at all (i.e., all of them have zero gradient). Other works assuming a malicious server either insert FC layer with tremendous number of input nodes (about 10^8), or even modify the federated learning architecture. Both cases are much more easily detectable than MKOR, and please refer to Section 2 in the paper for details. Also, different from some malicious server attacks, the performance of MKOR does not depend on any unmodified parameter, so there is no dependency on the accuracy or convergence of the network.

In Fig. 1 and Fig. 2, we consider reconstructing a batch with 100 MNIST images via original LeNet and modified LeNet, respectively, with MKOR, and compare with deep leakage from gradient (DLG) [4], improved DLG (iDLG) [3], gradient inversion (GI), and class fishing [2].

To show the robustness and effectiveness of our MKOR attack, we test MKOR along with other attack methods against two state-of-the-art defensive strategies. In Fig. 3 and Fig. 4, we show the comparison with original LeNet5 under Gaussian noise with normalized standard deviation 10^{-2} and Soteria defense with 30% prune rate, respectively. The average numerical results are shown in Table 2 and Table 3. The gradient clipping only changes the average of our reconstruction and does not affect the MKOR reconstruction result, while the additive artificial noise needs to be so high that it diminishes the model accuracy if the server is benign. Furthermore, Soteria and other optimization-based defensive detection schemes take longer time than gradient generation itself, thus can demand high edge device memory, computation and energy resources, and might potentially flag natural examples as well. We believe that a comprehensive defense approach by measuring the level of parameter inconspicuousness, such as sparse representation and low-rank representation, can be developed as more attack methods are proposed.

Table 2. Comparison with defensive Gaussian noise on gradients on MNIST with LeNet5 for random batches with batch size 100.

max, avg SSIM *max*, avg PSNR	Original LeNet Gaussian (10^{-2})	Modified LeNet Gaussian (10^{-1})
DLG	*0.10*, 0.01, *-3.53*, -5.58	*0.01*, 0.00, *-6.42*, -11.89
iDLG	*0.14*, 0.04, *0.59*, -0.59	*0.01*, 0.00, *-2.60*, -14.46
GI	*0.29*, 0.12, *15.44*, 11.23	*0.28*, 0.07, *9.88*, 6.08
CF	*0.20*, —, *14.42*, —	*0.33*, —, *12.52*, —
MKOR (ours)	**0.46**, 0.25, **16.71**, 12.74	**0.63**, 0.37, **18.25**, 13.41

In Fig. 6, Fig. 7, and Fig. 8, we present three other unique batches of CIFAR-100 reconstructed by MKOR via VGG16, where each image is the only one representing its class, and we compare with the original batches and those reconstructed by GI, and CF. As can be seen, overall, the images reconstructed by GI are much more distorted compared to proposed MKOR. Moreover, CF can only reconstruct one image per batch without necessarily providing the best reconstruction performance on this one image. Furthermore, as shown in Table 3 of our main paper, the image in the batch reconstructed by MKOR with the best performance has higher SSIM and PSNR values than the only image reconstructed by class fishing.

While these approaches achieve the above results with unique CIFAR-100 samples, their performance can degrade when there are multiple images from the same class. CF is especially vulnerable, since it can only reconstruct one image per batch. In Fig. 5, performance of MKOR is shown when samples are randomly selected from at most N classes, allowing different images from the same class. The number of unique samples (only ones from their class) is estimated by Eq. (3) in the paper. The performance degrades as N decreases, e.g., when $N=1$, reconstruction is the average of $K = 100$ images belonging to the same class. We note that the performance variance between batches for the same N value is due to the difference in data but not the MKOR algorithm.

In Fig. 9, we present the MKOR reconstruction on a unique batch with 1000 ImageNet images with original VGG16 network and modified network, respectively. In Fig. 10, we present the reconstruction on a random batch

Table 3. Comparison with Soteria defense on MNIST with LeNet5 for random batches with batch size 100.

max, avg SSIM *max*, avg PSNR	Original LeNet Prune rate 30%	Modified LeNet Prune rate 30%
DLG	*0.00*, 0.00, *-32.15*, -35.14	*0.01*, 0.00, *-13.41*, -24.15
iDLG	*0.00*, 0.00, *-30.75*, -33.49	*0.00*, 0.00, *-18.07*, -25.66
GI	*0.19*, 0.01, *7.89*, 2.77	*0.09*, 0.01, *2.89*, -1.81
CF	*0.16*, —, *11.43*, —	*0.17*, —, *8.73*, —
MKOR (ours)	**0.34**, 0.12, **15.29**, 10.66	**0.71**, 0.38, **18.79**, 13.49

with the same setting. We notice that the batch is very large while the high-resolution image has many details, so we randomly select 100 images in the batch to show in the plots.

To show the reconstruction details, we present larger single samples in Figs. 11, 12, 13, respectively, for Fig. 3 in the main paper, Fig. 4 in the main paper, and Fig. 10.

We also provide our code as a zipped supplementary file. The implementation details of the proposed MKOR can be found in this zipped file, the implementation details of DLG and iDLG can be found in [3], and the implementation details of GI and CF can be found in [1].

References

[1] Yuxin Wen Jonas Geiping, Liam Fowl. breaching. https://github.com/JonasGeiping/breaching, 2023. 4

[2] Yuxin Wen, Jonas Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. *arXiv preprint arXiv:2202.00580*, 2022. 3

[3] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020. 3, 4

[4] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019. 3

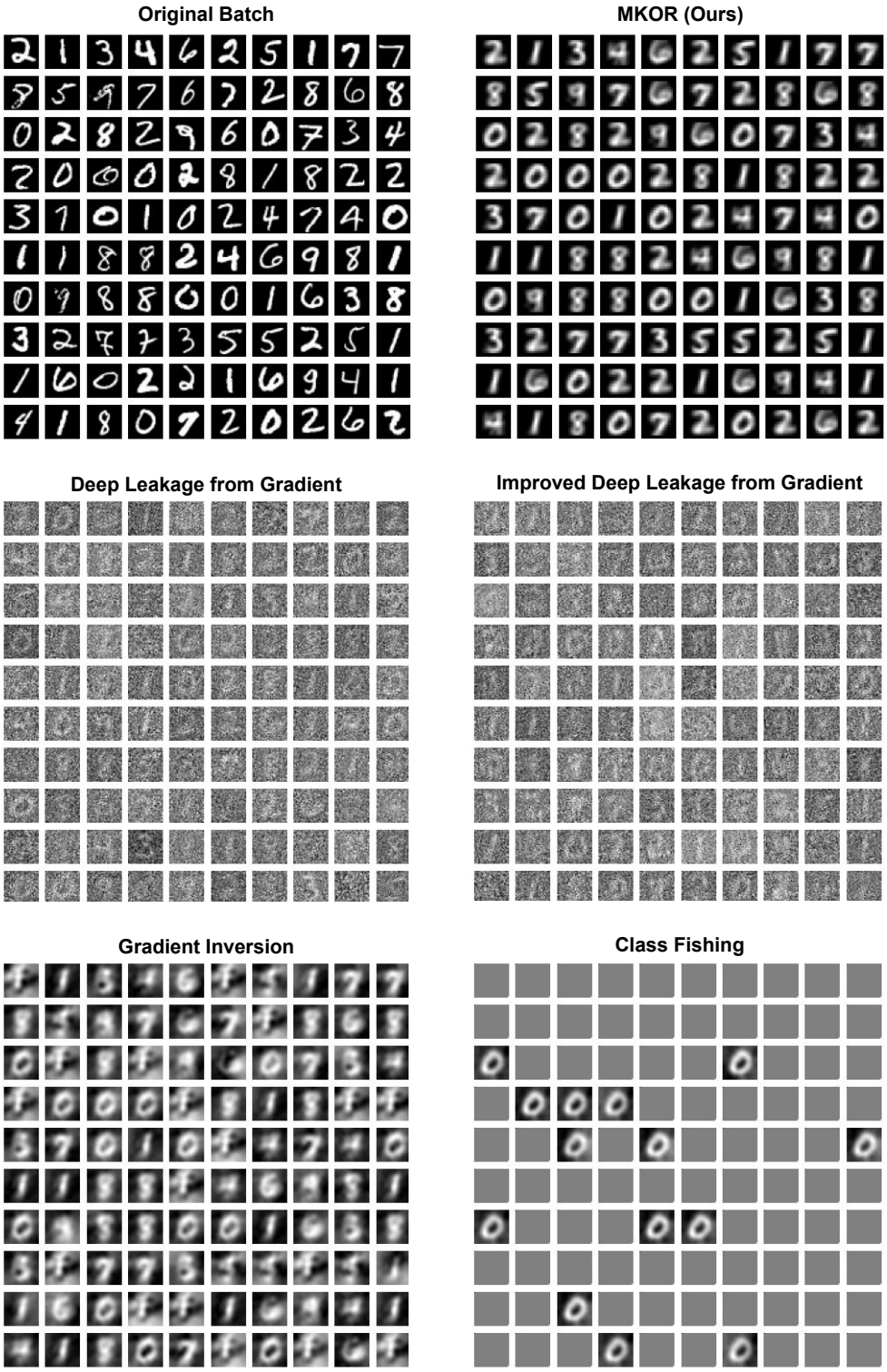

Figure 1. Qualitative comparison between different input reconstruction methods on MNIST with original LeNet5 for a batch size of 100.

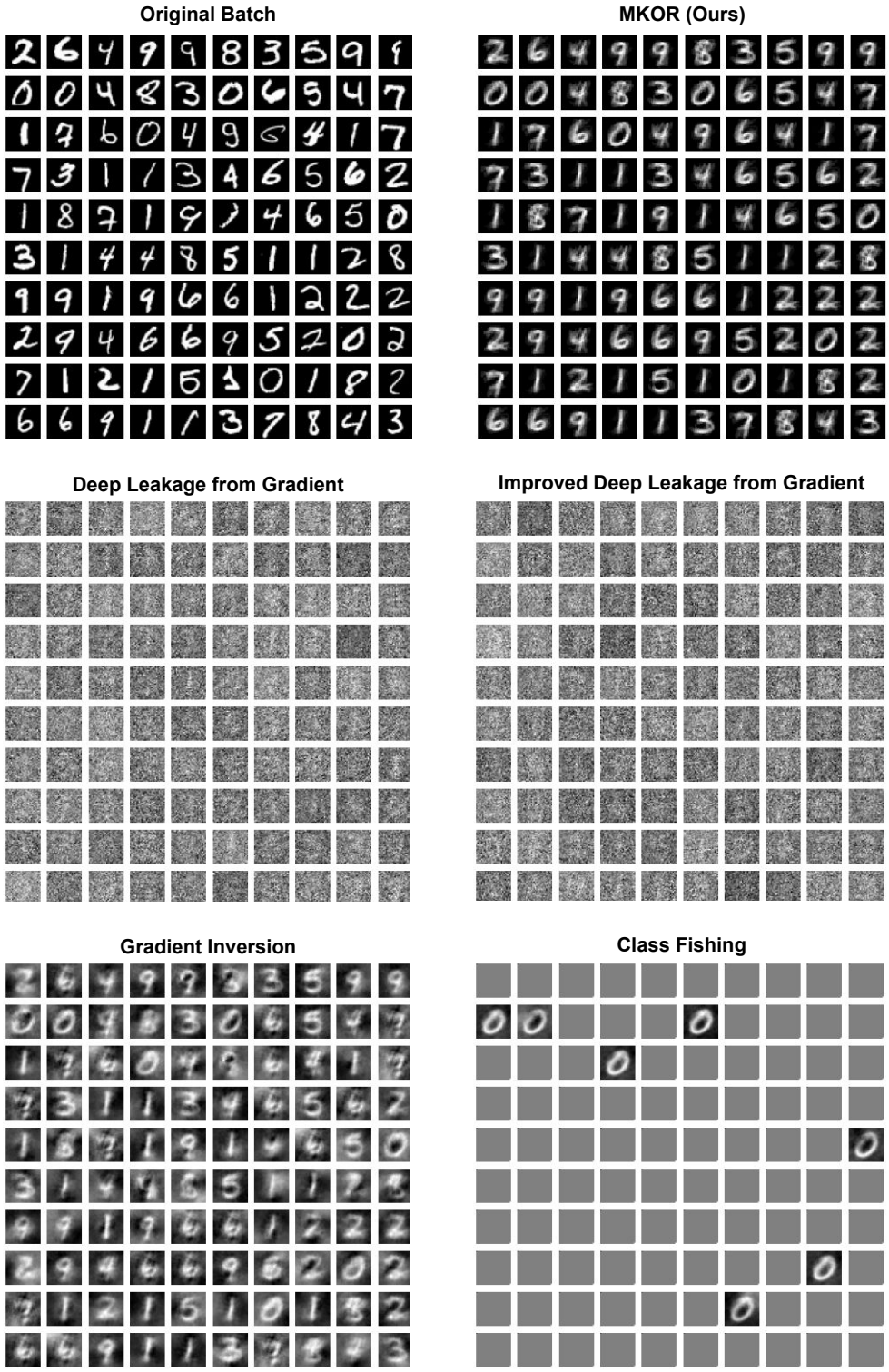

Figure 2. Qualitative comparison between different input reconstruction methods on MNIST with modified LeNet for a batch size of 100.

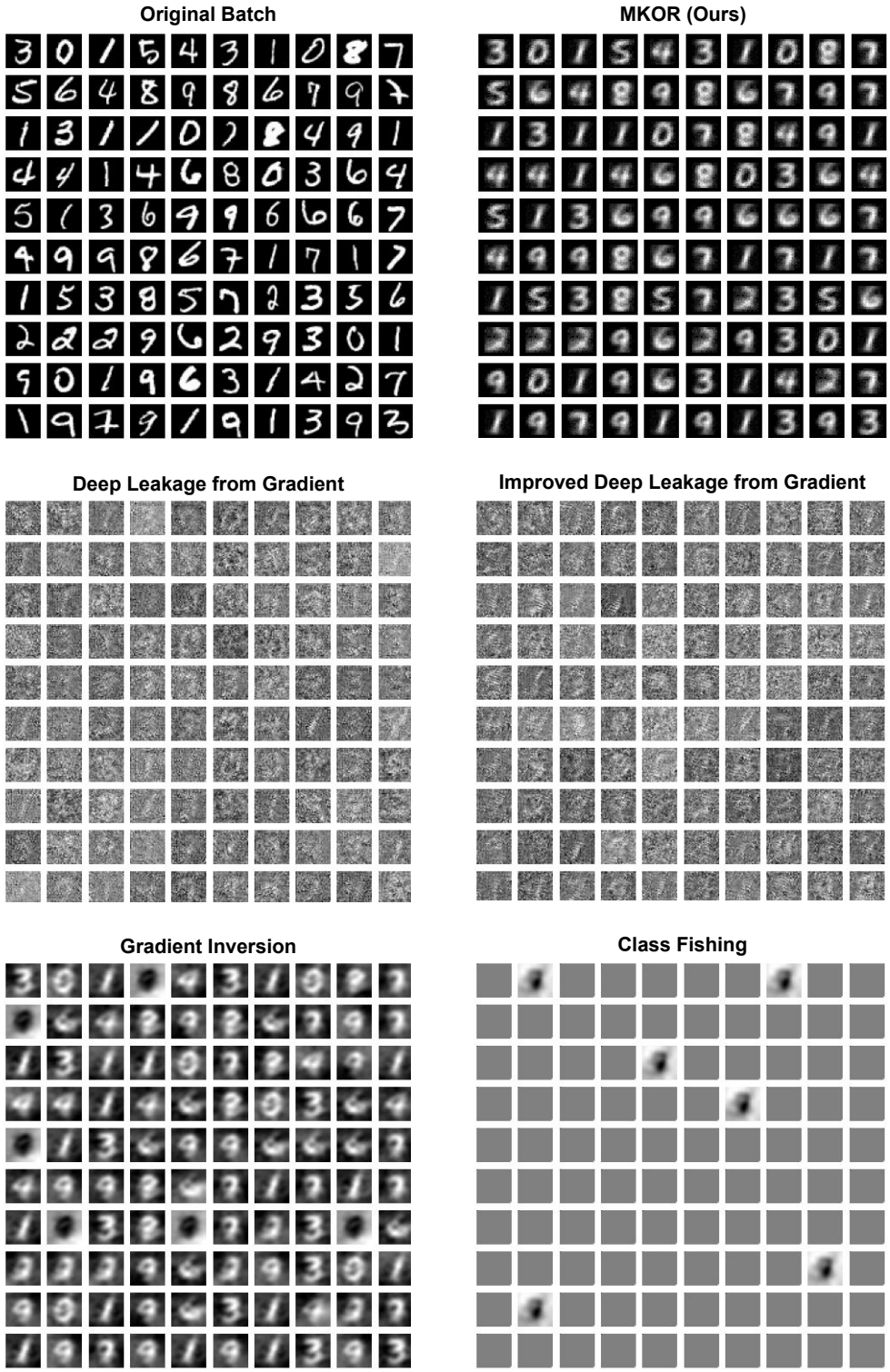

Figure 3. Qualitative comparison between different input reconstruction methods under Gaussian noise on MNIST with original LeNet5 for a batch size of 100.

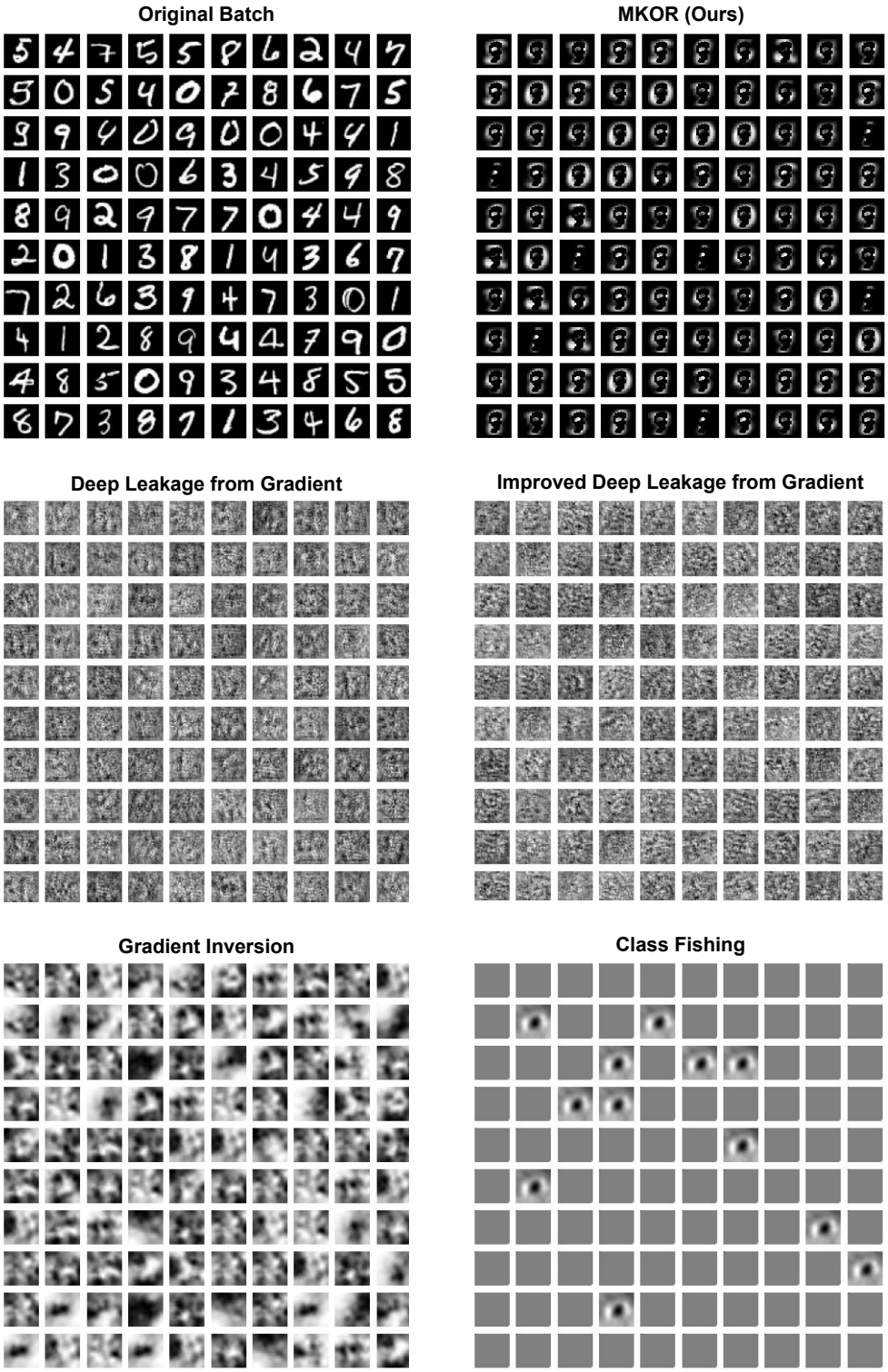

Figure 4. Qualitative comparison between different input reconstruction methods under Soteria defense on MNIST with original LeNet5 for a batch size of 100.

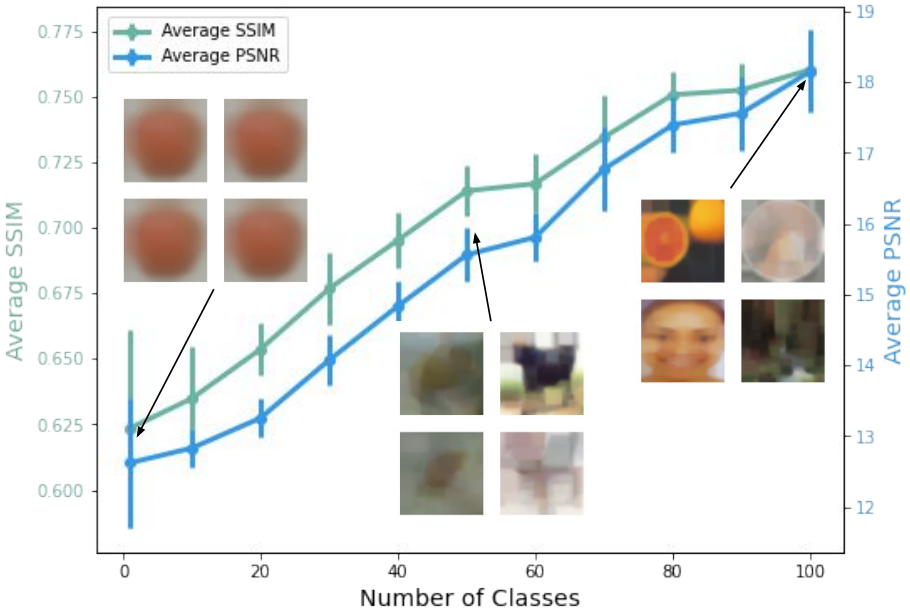

Figure 5. Reconstruction performance of MKOR on batches with at most N classes. Each data point shows the mean and standard deviation over $U = 10$ batches. We also plot 4 out of all $K = 100$ samples in a batch for the cases of $N = 1$, $N = 50$ and $N = 100$.

Figure 6. Qualitative comparison between different input reconstruction methods on CIFAR-100 for an additional batch with a batch size of 100.

Figure 7. Qualitative comparison between different input reconstruction methods on CIFAR-100 for another additional batch with a batch size of 100.

Figure 8. Qualitative comparison between different input reconstruction methods on CIFAR-100 for another additional batch with a batch size of 100.

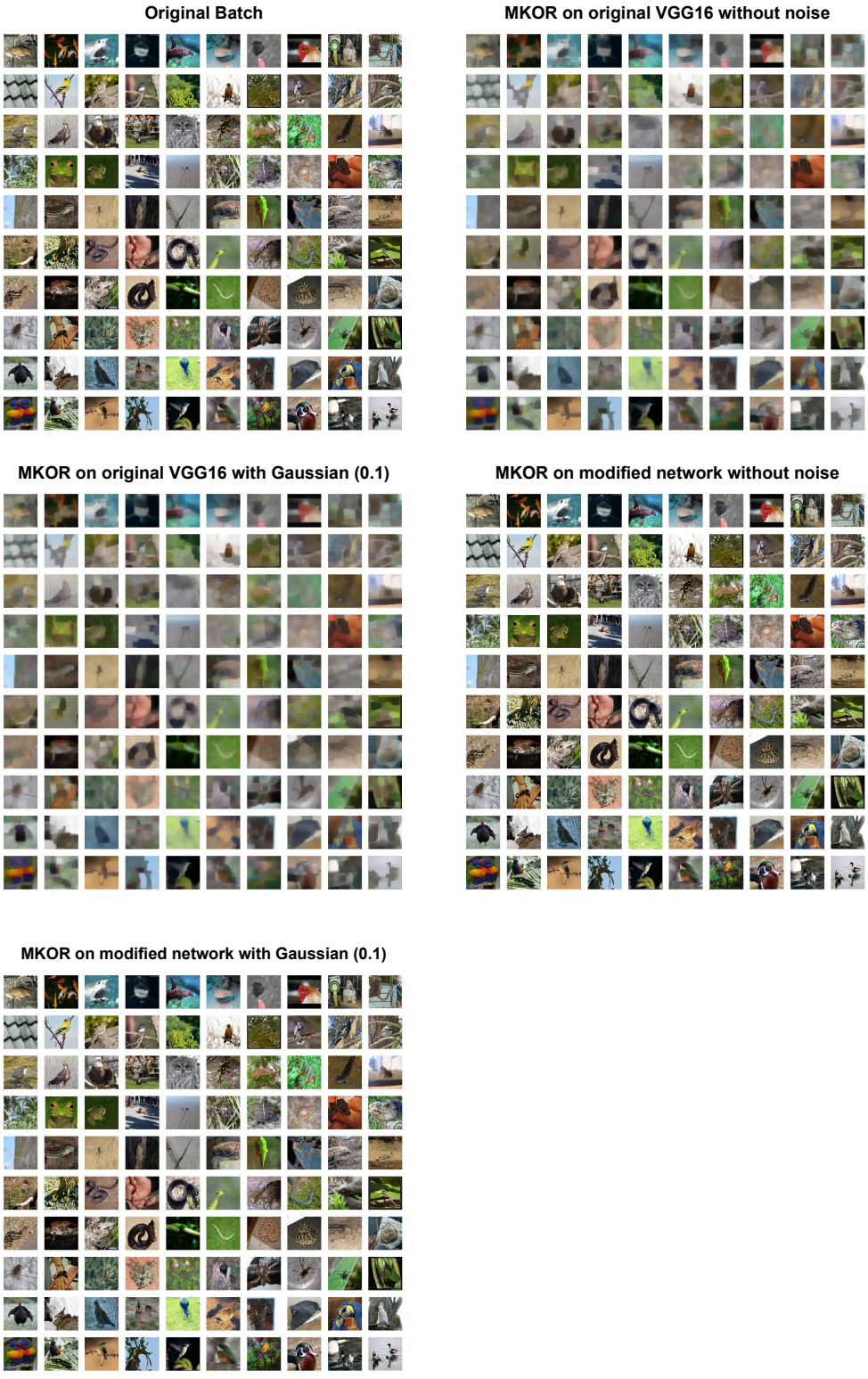

Figure 9. Input reconstruction on a **unique** batch with 1000 ImageNet images, with either original VGG16 or modified network, and with either no noise or 10^{-1} Gaussian noise. We randomly display 100 images out of 1000 in the batch for simplicity.

Figure 10. Input reconstruction on a **random** batch with 1000 ImageNet images, with either original VGG16 or modified network, and with either no noise or 10^{-1} Gaussian noise. We randomly display 100 images out of 1000 in the batch for simplicity.

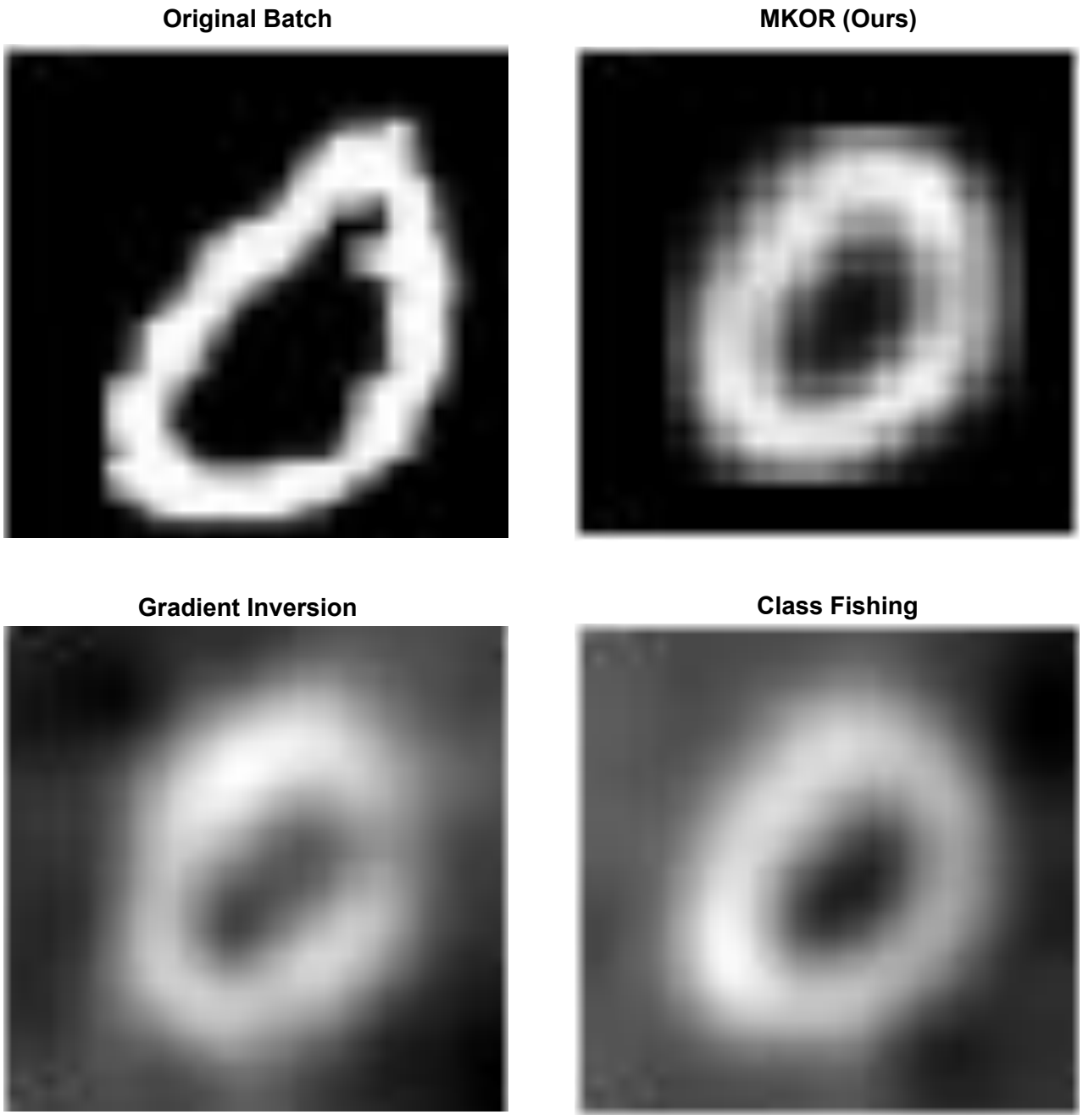

Figure 11. Single samples in batches of different input reconstruction methods on MNIST with original LeNet5 for a random batch of size 100.

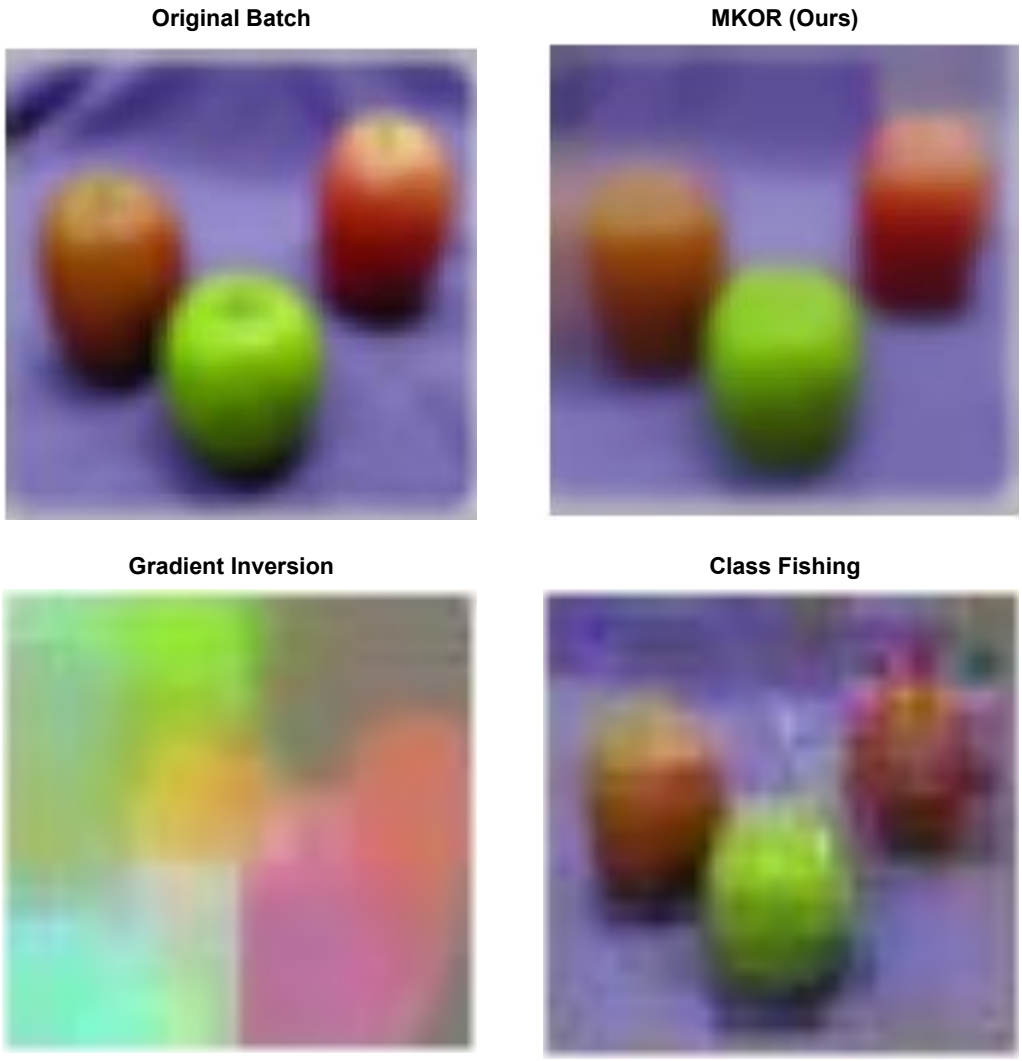

Figure 12. Single samples in batches of different input reconstruction methods on CIFAR-100 for a unique batch of size 100.

Figure 13. Single samples in random batches with 1000 ImageNet images, with either original VGG16 or modified network, and with either no noise or 10^{-1} Gaussian noise. .