Cocktail Party Attack: Breaking Aggregation-Based Privacy in Federated Learning Using Independent Component Analysis

Sanjay Kariyappa [† 1] **Chuan Guo** [2] **Kiwan Maeng** [† 3] **Wenjie Xiong** [† 4] **G. Edward Suh** [5 2]
Moinuddin K Qureshi [1] **Hsien-Hsin S. Lee** [† 6]

Abstract

Federated learning (FL) aims to perform privacy-preserving machine learning on distributed data held by multiple data owners. To this end, FL requires the data owners to perform training locally and share the gradients or weight updates (instead of the private inputs) with the central server, which are then securely aggregated over multiple data owners. Although aggregation by itself does not offer provable privacy protection, prior work suggested that if the batch size is sufficiently large the aggregation may be secure enough. In this paper, we propose the Cocktail Party Attack (CPA) that, contrary to prior belief, is able to recover the private inputs from gradients/weight updates aggregated over as many as 1024 samples. CPA leverages the crucial insight that aggregate gradients from a fully connected (FC) layer is a linear combination of its inputs, which allows us to frame gradient inversion as a blind source separation (BSS) problem. We adapt independent component analysis (ICA)—a classic solution to the BSS problem—to recover private inputs for FC and convolutional networks, and show that CPA significantly outperforms prior attacks, efficiently scales to ImageNet-sized inputs, and works on large batch sizes of up to 1024.

1. Introduction

Federated learning (FL) is a flexible framework for privacy-preserving machine learning (ML) model training on distributed data. The FL framework typically consists of a central server and multiple clients that hold private training data. The protocol involves the server distributing the model parameters θ to the clients, and then the clients using this model to compute gradients $\nabla_\theta \mathcal{L}$ using their private data as shown in Fig. 2a. The gradients are aggregated and shared with the server who then uses it to update the model parameters, and this process is repeated until convergence.

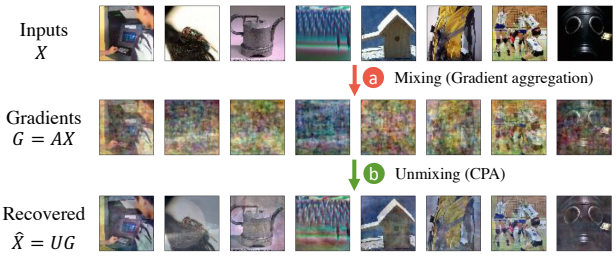


Figure 1. (a) Aggregate gradients from an FC layer are linear combinations of its inputs, as shown by the visualization of the gradients in the second row. (b) *Cocktail party attack* uses this observation to frame gradient inversion as a blind source separation problem and recovers the inputs from the gradients by optimizing an unmixing matrix U using independent component analysis.

While FL avoids the direct sharing of data, this in itself does not guarantee privacy as the gradient update shared with the server can contain information about the private training data. For instance, Zhu et al. (2019) showed that gradients can be *inverted* to recover their associated private data in a process now called *gradient inversion*. However, a key drawback of existing gradient inversion attacks is that they are sensitive to the input size. This limits their efficacy when recovering high-dimensional inputs (e.g. ImageNet) from aggregate gradients, especially when aggregation is done over a large number of inputs (Zhu et al., 2019).

We overcome the limitations of prior works by developing *Cocktail Party Attack (CPA)*[1], which uses a fundamentally different approach to gradient inversion that is not adversely impacted by the input size, making it scalable to ImageNet-size inputs in the large batch size regime. Our attack leverages the novel insight that the aggregated gradient from an FC layer can be viewed as a linear combination of its inputs, which allows us to frame the recovery of these inputs

[†]Work done while authors were at FAIR/Meta AI [1]Georgia Institute of Technology [2]Meta AI [3]Pennsylvania State University [4]Virginia Tech [5]Cornell University [6]Intel. Correspondence to: Sanjay Kariyappa <sanjaykariyappa@gatech.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]The name of our attack is inspired by the cocktail party problem (Cherry, 1953)– a popular example of the BSS problem.

as a *blind source separation* (BSS) problem and solve it by adapting the classical independent component analysis (ICA) algorithm (Lee, 1998). We show that CPA is much more scalable than prior attacks, successfully scaling to large batch sizes of up to 1024. Concretely, CPA can be used for the following privacy attacks on FL:

- **Gradient Inversion for FC networks:** CPA can readily perform gradient inversion for an FC network (or any network where the first layer is an FC layer) to recover private inputs. Fig. 1 shows an illustration of CPA on an FC network, where a batch of training images is recovered with high quality from its aggregated gradient.
- **Gradient Inversion for CNNs:** We can extend CPA to perform gradient inversion on convolutional networks by first recovering the per-sample embeddings to an FC layer of the network and then inverting these embeddings using feature inversion to recover the input images.
- **Update Inversion for FC networks:** Federated averaging (FedAvg) is a variant of FL where the clients share the model updates (instead of gradients) after multiple local training steps. We show that CPA can also be used in this setting to launch an *update inversion attack* to recover the private inputs from the weight updates of FC networks.

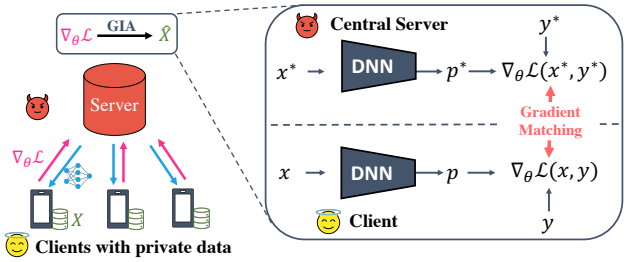
The search space of CPA is independent of the input size, allowing it to scale efficiently to real world FL settings. Empirically this leads to several advantages over prior work:

- CPA can perform high-quality recovery of private inputs even with batch size as large as 1024 in our evaluation on inverting the gradient/weight updates from an FC network trained on CIFAR-10 and Tiny-Imagenet, and a VGG-16 network trained on ImageNet.
- Compared to prior work based on *gradient matching*, CPA can recover inputs with better quality and scales to datasets with larger input sizes (e.g., ImageNet). Furthermore, we show that gradient matching can be combined with CPA to further improve attack performance.
- CPA only uses simple image priors such as smoothness and does not require knowledge of the input data distribution or modifications to the model parameters, and hence is more versatile and applicable to real world settings.

The effectiveness of CPA shows that aggregation alone does not provide meaningful privacy guarantees and defenses like differential privacy (Dwork et al., 2014) are truly necessary to prevent gradients from leaking private data in FL.

2. Background

In this section, we provide background on FL, gradient inversion attack (GIA) and update inversion attack (UIA), and give an overview of relevant prior work.



(a) Federated learning

(b) Gradient inversion attack using gradient matching (prior work)

Figure 2. (a) FL requires the clients to perform training locally and send the gradients $\nabla_\theta \mathcal{L}$ (instead of the private data X) to a central server. (b) Gradient inversion attacks (GIA) break privacy by estimating X from the gradients. Prior works carry out GIA by optimizing a set of dummy parameters (x^*, y^*) with the objective of matching the gradients obtained during FL.

2.1. Federated Learning

FL aims to train a model on distributed data held by multiple clients in a privacy-preserving manner. FL involves a central server and multiple clients who hold private data X as shown in Fig. 2a. To train a model f_θ, the server starts by distributing the model parameters θ to the clients. The client computes a model update on their private data locally, which either consists of a single gradient update or multiple gradient updates depending on the FL protocol:

- **FedSGD:** The client performs a single training step by using the batch of private data to compute the aggregate gradients of the loss $\nabla_\theta \mathcal{L}$ with respect to the model parameters θ, which is transmitted to the central server.
- **FedAVG (Konečnỳ et al., 2015; McMahan et al., 2017):** The client performs multiple training steps with potentially different batches of training data and transmits the change in the model's weights $\Delta\theta$ to the central server.

The server collects and aggregates the gradient/weight updates and uses it to update the model parameters. Secure aggregation (Bonawitz et al., 2016) can be applied to ensure that only the aggregated update is observed by the server, which obscures the update from individual clients. This process is repeated until the model converges.

2.2. Gradient and Update Inversion Attacks

Attack Objective: Let $\mathcal{A}$ denote the gradient/update inversion attack. The goal of the attack is to recover the training samples x from the aggregate gradient $\nabla_\theta \mathcal{L}(x, y)$ or model updates $\Delta\theta$, such that the recovered samples $\hat{x}$ are semantically similar to the private training samples x. More precisely, given a semantic similarity measure d, the attack objective is:

$$\min d(\hat{x}, x) \text{ where } \hat{x} = \begin{cases} \mathcal{A}(\nabla_\theta \mathcal{L}(x, y)) & \text{for GIA} \\ \mathcal{A}(\Delta\theta) & \text{for UIA} \end{cases} \quad (1)$$

In principle, an adversarial server with access to the aggregated model update can use GIA and UIA to recover private training samples in FedSGD and FedAvg respectively.

Threat Model: We assume an honest-but-curious adversary who does not have access to a significant amount of in-distribution data. This means that the central server faithfully follows the FL protocol and uses the observed aggregated model update to achieve the attack objective, without distributional knowledge of the training data.

2.3. Related Work

Gradient Matching: The vast majority of prior work on gradient inversion does so via *gradient matching* (Zhu et al., 2019), which optimizes a batch of dummy inputs and labels (x^*, y^*) to produce a gradient that matches the one received by the server during FL as shown in Fig. 2b. This can be done by minimizing the distance between the gradient produced by the dummy variables $\nabla_\theta \mathcal{L}(x^*, y^*)$ and the gradient received during FL $\nabla_\theta \mathcal{L}(x, y)$:

$$\hat{x}, \hat{y} = \underset{x^*, y^*}{\arg\min}\, d(\nabla_\theta \mathcal{L}(x^*, y^*), \nabla_\theta \mathcal{L}(x, y)) \tag{2}$$

Here, d denotes a distance metric between vectors like cosine similarity or L_2 norm. Subsequent work (Zhao et al., 2020) proposed a method to infer the ground truth labels y by examining the gradients of the last layer, which can help improve the gradient matching attack. However, this method only works when no two inputs in the batch belong to the same output class, limiting its applicability. Aditionally, Zhu & Blaschko (2020) proposed a closed-form recursive solution to recover the input from the gradient. However, this method only works with a batch size of 1.

Total Variation (TV) Prior: Since the training samples x is a set of images, one can impose priors on $\hat{x}$ to reduce the search space and achieve better gradient inversion performance. To this end, Geiping et al. (2020) proposed to use the TV prior (Rudin et al., 1992) as a regularization term along with the gradient matching objective as follows:

$$\mathcal{R}_{TV}(x^*) = \mathbb{E}\big[|x^*_{i+1,j} - x^*_{ij}|\big] + \mathbb{E}\big[|x^*_{i,j+1} - x^*_{ij}|\big] \tag{3}$$

The TV prior penalizes high-frequency components in the input and encourages the optimization to find natural-looking images. With this prior, gradient matching can be scaled to work on a batch size of up to 100 for CIFAR-10 images and up to a small number of inputs for ImageNet.

Generative Models and Batch Norm Statistics: In addition to the TV prior, recent works have proposed to use generative models (Jeon et al., 2021) and batch norm statistics (Yin et al., 2021; Hatamizadeh et al., 2022) to impose a prior on the inputs when carrying out the attack. These priors, when used with gradient matching, improve the quality of the recovered images and help the attack scale to even larger batch sizes. However, a key drawback of these approaches is that they require access to a distributionally similar dataset to train a generative model or compute batch-norm statistics, which may not be available to the attacker. See Appendix D for an overview of these attacks.

Dishonest Central Server: Several works have proposed attacks under the threat model of a dishonest central server. Boenisch et al. (2021); Fowl et al. (2021); Wen et al. (2022) have proposed methods to recover private training samples in FL by using malicious model parameters. Such methods use weights that cause the aggregate gradient or the difference between two aggregate gradients to be predominantly influenced by a single input. Lam et al. (2021) proposed an attack to recover the gradient updates of individual clients using aggregate gradients from multiple rounds and user-participation metadata. In addition to the extra metadata information, this method requires modification to the FL algorithm to keep the model constant between multiple rounds. Under our threat model of an honest central server, malicious modifications to model parameters and changes to the FL algorithm are not allowed. Thus we do not consider these attacks in our evaluations.

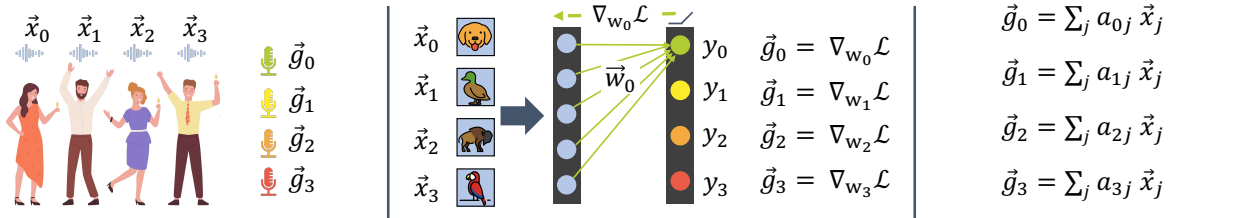
Limitations of Prior Work: The optimization complexity of gradient matching poses a fundamental limitation to its scalability. Most prior works (with the exception of generative image prior) perform optimization directly in the image pixel space. The size of this optimization problem is $\mathcal{O}(n \times d_{in})$, where n is the batch size and d_{in} denotes the dimensionality of the input image. For instance, for a batch of 100 images from ImageNet, the size of the optimization problem is approximately 15M, which may be even larger than the size of the model parameters. This scaling makes it difficult to apply gradient matching-based attacks to large batches of data or high-resolution input images.

3. Cocktail Party Attack (CPA)

Our paper proposes CPA—a gradient/update inversion attack that scales to large batches and high-resolution images by drastically reducing the optimization search space in a clever way. Our key insight is to frame the attack as a blind source separation (BSS) problem and adapt independent component analysis (ICA) to recover the private inputs from aggregate gradient/weight updates. This attack works directly for inverting the gradients of FC networks (or networks whose first layer is fully-connected), and in Section 4 we show how the attack can be extended to CNNs.

3.1. Framing Gradient Inversion as a Blind Source Separation Problem

We first introduce the blind source separation (BSS) problem using the motivating example of the cocktail party problem,



(a) Microphone recordings in a cocktail party (b) Gradients from a fully connected layer (c) Linear combinations of inputs

Figure 3. The microphone recordings in the cocktail party problem and the gradients from a fully connected layer can both be represented as linear combination of inputs. Recovering the inputs in both cases can be viewed as a blind source separation problem.[3]

and show that GIA and UIA for an FC layer can also be viewed as a BSS problem.

Cocktail Party Problem: Consider a cocktail party where there are a group of four people talking simultaneously as depicted in Fig. 3a. A microphone placed near this group picks up an audio recording consisting of an overlapping set of voices from the four speakers. The cocktail party problem models this recording as a linear combination of the voices of the four speakers. More formally, if $\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x}_3$ denote the voices of the four speakers and $\vec{g}_0, \vec{g}_1, \vec{g}_2, \vec{g}_3$ denote the recordings from the four microphones placed at different locations, the recording of the i-th microphone is given by:

$$\vec{g}_i = a_{i0}\vec{x}_0 + a_{i1}\vec{x}_1 + a_{i2}\vec{x}_2 + a_{i3}\vec{x}_3. \tag{4}$$

Here, a_{ij} denote the unknown mixing coefficients. The BSS problem can be stated as follows: *Given the mixed signals* $\{\vec{g}_i\}$*, recover the individual source signals* $\{\vec{x}_i\}$.

Note that if the coefficients a_{ij} are known, this problem has a straightforward solution by solving the linear system given by Eqn. 4 for $i = 0, 1, 2, 3$. The core difficulty of BSS lies in the coefficients being unknown, and thus some assumptions about $\vec{x}_i$ must be made to enable their recovery.

Gradient Inversion for FC Layer: Much like the cocktail party problem, the aggregate gradients from an FC layer can be represented as a linear combinations of the inputs used to generate them. To demonstrate this, consider an FC layer with four hidden neurons y_0, y_1, y_2, y_3 as shown in Fig. 3b. Let $\vec{w}_0, \vec{w}_1, \vec{w}_2, \vec{w}_3$ represent the weight vectors associated with each output neuron. Let $\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x}_3$ be a batch of four inputs used to perform a single iteration of training. $\vec{g}_i = \nabla_{w_i}\mathcal{L}$ is the aggregate gradient of the loss with respect to $\vec{w}_i$, which is computed by taking the mean of the individual gradients $\nabla_{w_i}\mathcal{L}^j$ associated with each input $\vec{x}_j$. This aggregate gradient $\vec{g}_i$ can be further expressed as a linear combination of the inputs $\vec{x}_j$ as follows:

$$\vec{g}_i = \frac{1}{4}\sum_j \nabla_{w_0}\mathcal{L}^j = \frac{1}{4}\sum_j \frac{\partial \mathcal{L}}{\partial y_i^j}\frac{\partial y_i^j}{\partial w_i} = \frac{1}{4}\sum_j \frac{\partial \mathcal{L}}{\partial y_i^j}\vec{x}_j. \tag{5}$$

Notice that Eqn. 5 has a 1-to-1 correspondence with Eqn. 4 in the cocktail party problem: The inputs here are analogous to the speakers and gradients are analogous to the recordings from the microphones. The coefficients $a_{ij} = \partial\mathcal{L}/\partial y_i^j$ are also unknown to the server. Recovering the inputs $\{\vec{x}_i\}$ (source signals) from a set of aggregate gradients $\{\vec{g}_i\}$ (mixed signals) can thus be viewed as a BSS problem.

Update Inversion for FC layer: Consider an FC model f, with θ and θ' denoting the weights of the first layer before and after training f with SGD with a dataset $\mathcal{D} = \{x_i, y_i\}$ for multiple iterations. The weight update $\Delta\theta = \theta' - \theta$ can be viewed as a linear combination of the gradient $\vec{G}_j$ at each step of SGD. However, each individual gradient $\vec{G}_j$ is in-turn a linear combination of the inputs x_i in the training dataset. Thus, the weight update $\Delta\theta$ can be re-expressed a linear combination of the inputs as described in Eqn. 6.

$$\Delta\theta = \sum_j a_j\vec{G}_j = \sum_i b_i x_i \tag{6}$$

Since the weight update for an FC layer is a linear combination of inputs in the training dataset, we can view update inversion for an FC layer as a BSS problem as well.

Solving BSS with an Unmixing Matrix: Eqn. 5 can be expressed as a matrix multiplication operation as follows: $G = AX$. The rows of $X \in \mathbb{R}^{n\times d}$ denote the inputs (source signals), rows of the $A \in \mathbb{R}^{n\times n}$ represent the coefficients of the linear combinations and rows of the $G \in \mathbb{R}^{n\times d}$ denote the gradients (aggregate signals). We can estimate the source matrix $\hat{X}$ from G by estimating an *unmixing matrix* U and computing $\hat{X} = UG$, where each row of $\hat{X}$ is a single recovered input $\hat{x}_i$. Note that X can be recovered perfectly if $U = A^{-1}$. Estimating $\hat{X}$ can thus be reduced to finding the unmixing matrix U, which has size $\mathcal{O}(n\times n)$ and is independent of the input-dimensionality d. This unique feature enables CPA to scale to ImageNet-sized datasets even with large batch sizes.

3.2. Gradient Inversion using ICA

Independent component analysis is a classic signal processing technique that can be used to solve the BSS problem by estimating the unmixing matrix U. To do this, ICA starts with a randomly initialized unmixing matrix U^* and optimizes it to enforce certain properties on the recovered source

signals. To explain, let $x_i^* = u_i^* G$ denote the i-th source signal recovered from multiplying the i-th row of U^* with G. Note that the source signals represent images in our case (Fig. 3b). ICA optimizes U^* so that the recovered source signals $\{x_i^*\}$ satisfy the following key properties:

- *Non-Gaussianity:* Values of real-world signals such as images and speech typically do not follow a Gaussian distribution. We can measure non-Gaussianity using the negentropy metric (Hyvärinen & Oja, 2000):

$$J(x^*) = \mathbb{E}\Big[\frac{1}{a^2}\log\cosh^2(ax_i^*)\Big]. \tag{7}$$

  A high value of negentropy indicates a high degree of non-Gaussianity.
- *Mutual Independence (MI):* We assume that the source signals are independently chosen and thus their values are uncorrelated. Since each row u_i^* of the unmixing matrix corresponds to a recovered source signal x_i^*, we can enforce MI by minimizing the absolute pairwise cosine similarity between the rows of U^*:

$$\mathcal{R}_{MI} = \mathop{\mathbb{E}}_{i\neq j} exp\Big(T|CS(u_i^*, u_j^*)|\Big). \tag{8}$$

- *Source Prior:* Any prior information about the source signals, such as the TV prior or the generative image prior, can be included in our optimization in the form of an additional regularization term $\mathcal{R}_P$.

We estimate the unmixing matrix U by solving an optimization problem that combines the above properties[4]:

$$U = \arg\max_{U^*} \mathop{\mathbb{E}}_{i}\Big[J(u_i^* G) - \lambda_P \mathcal{R}_P(u_i^* G))\Big] - \lambda_{MI}\mathcal{R}_{MI}, \tag{9}$$

where λ_P and λ_{MI} are chosen hyperparameters[5]. The U matrix obtained from solving Eqn. 9 can be used to estimate the source matrix via $\hat{X} = UG$.

3.3. CPA for FC Models

For an FC model trained on image data, where the inputs are directly fed to an FC layer, we can recover the inputs directly by inverting the gradient/weight updates of the first FC layer using CPA. Since the source signals are images, we can use TV regularization $\mathcal{R}_{TV}$ (Eqn. 3) as the source prior in Eqn. 9. One caveat of ICA is that it does not preserve the magnitude of the input (i.e. $\hat{x}_i$ can be a scaled version of x_i). However, this can be easily resolved by normalizing the pixel values to the range $[-1, 1]$ and selecting between $\hat{x}_i$ and $-\hat{x}_i$ through a visual comparison.

[3]All emojis in this paper are from https://openmoji.org/ and licensed CC BY-SA 4.0.

[4]We whiten and center the gradients as a pre-processing step, before using it in our optimization.

[5]We refer the reader to the ablation study in Appendix A to understand the relative importance of different terms in Eqn.9.

4. Extending Cocktail Party Attack to CNNs

The formulation of CPA as finding an unmixing matrix depends critically on the structure of the aggregated gradient for an FC layer. On the surface, this requirement is not satisfied for CNN models that are commonly used for image recognition tasks, and CPA is seemingly not applicable. Fortunately, we leverage the fact that most CNN models contain at least one FC layer towards the end of the network to recover individual *embeddings* from a batch of data.

More concretely, consider the embedding vector z produced immediately before the first FC layer of the model. For each sample in the training batch, we can use CPA to recover its embedding z from the aggregated gradient vector, and then use *feature inversion* (Mahendran & Vedaldi, 2015; Ulyanov et al., 2018) to recover the training image as shown in Fig. 4. We describe these two steps in detail below.

4.1. Leaking Private Embeddings using CPA

The gradients from the FC layer can be viewed as linear combinations of the embeddings z that act as the input to the FC layer. We can use CPA to invert the gradients from the FC layer (G) and recover an estimate of the embeddings $\hat{z} = UG$. However, we can no longer use the TV prior in our optimization objective (Eqn. 9) to find U since the signal being recovered (z) is not an image. Instead, we use the following properties of embeddings produced with a ReLU non-linearity to design more appropriate input priors:

- *z is sparse:* Embeddings produced by networks that use ReLU non-linearity are sparse, as ReLU squashes negative activations to 0. We can use the L_1-norm: $|z^*|_1$ in our optimization to encourage sparsity.
- *z is a non-negative vector:* The embedding vector is non-negative as ReLU truncates negative values to 0. We can minimize $\mathcal{R}_{NN}(z^*) = ReLU(-z^*)$ to encourage z^* to be non-negative. However, the embedding recovered by ICA can be sign inverted, which results in a non-positive vector. To allow for sign inverted recovery, we propose the *sign regularization* function: $\mathcal{R}_{SR} = \min(ReLU(z^*), ReLU(-z^*))$. Minimizing $\mathcal{R}_{SR}$ ensures that z^* is either non-negative or non-positive.

We combine the above regularization terms with the non-Gaussianity and mutual Independence assumptions to derive the final optimization objective to estimate the unmixing matrix U as follows:

$$U = \arg\max_{U^*} \mathop{\mathbb{E}}_{i}\Big[J(u_i^* G) - \lambda_{SP}|u_i^* G|_1 - \lambda_{SR}\mathcal{R}_{SR}(u_i^* G)\Big] - \lambda_{MI}\mathcal{R}_{MI}. \tag{10}$$

The unmixing matrix can be used to recover the private embeddings $\hat{Z}$ from the gradient G as follows: $\hat{Z} = UG$. Doing so recovers the per-sample embeddings for every

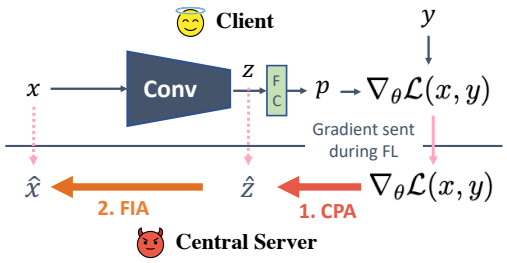


Figure 4. We attack CNN models by first recovering the private embedding ($\hat{z}$) from the FC layer using CPA and then using feature inversion attack (FIA) to recover the training image ($\hat{x}$) from $\hat{z}$.

sample in the training batch, which can be used to infer private information about the training samples. For instance, we can use these private embeddings to recover the training image using a feature inversion attack.

4.2. Feature Inversion Attack

Feature inversion attack (FIA) (Mahendran & Vedaldi, 2015) inverts the embedding produced by a neural network to recover the input. Formally, given an embedding function $f : X \rightarrow Z$ that maps an image x to an embedding $z = f(x)$, FIA recovers an estimate of the input $\hat{x}$ from z. We do this by solving the following optimization problem using a dummy input x^*:

$$\hat{x} = \arg\max_{x^*} CS(f(x^*), z) - \lambda_{TV} \mathcal{R}_{TV}(x^*) \tag{11}$$

The first term maximizes the cosine similarity between the embedding from the dummy input $z^* = f(x^*)$ and the true embedding z. The second term is TV regularization, which suppresses high-frequency components. Solving this optimization problem allows us to estimate the private inputs $\{\hat{x}_i\}$ from the embedding $\{\hat{z}_i\}$ recovered by CPA, which completes the GIA. Additionally, we can also use the gradient information to improve FIA by including the gradient matching objective in the optimization as follows:

$$\begin{aligned}\hat{x} = \arg\max_{x^*} & CS(f(x^*), z) + \lambda_{GM} \\ & CS(\nabla_\theta \mathcal{L}(x^*), \nabla_\theta \mathcal{L}(x)) - \lambda_{TV} \mathcal{R}_{TV}(x^*).\end{aligned} \tag{12}$$

5. Experiments

To demonstrate the efficacy of CPA, we evaluate our proposed attack on FC and CNN models trained on image classification tasks. While FC networks are typically not used for image classification, they allow us to demonstrate the efficacy of CPA in its simplest form. Our evaluations on the CNN model (VGG-16) demonstrates the utility of our attack in a more realistic problem setting.

5.1. Setup

Model and Datasets: For our experiments on the FC model, we use a simple 2-layer network (FC-2), with the following network architecture: $[Linear(256) - ReLU() - Linear(k)]$ for a k-class classification problem. We train FC-2 on the CIFAR-10 (Krizhevsky et al., 2009) and Tiny-ImageNet datasets for 20 epochs using the Adam (Kingma & Ba, 2014) optimizer with a learning rate of 0.001. We perform our CNN experiments using ImageNet with a pre-trained VGG-16 network (from torchvision (et al., 2017)).

Evaluation Methodology: We evaluate gradient inversion attacks with the following batch sizes: $[32, 64, 128, 256]$ for the FC-2 model and $[32, 64, 128, 256, 512, 1024]$ for VGG-16. We perform evaluations by first sampling a batch of inputs $\{x_i\}$ from an unseen test set to generate the aggregate gradient $\nabla_\theta \mathcal{L}$. We then use different gradient inversion attacks to recover an estimate of the inputs $\{\hat{x}_i\}$ from the aggregate gradients and compare their performance. Since our experiments are done on image data, we use the LPIPS score (Zhang et al., 2018) to quantify the perceptual similarity between the original and recovered images to evaluate the attacks. We repeat the attack on 5 batches of data and report the average LPIPS scores in our results. We also use the FC-2 network to evaluate the update inversion attack. We use datasets of size $[32, 64, 128, 256]$, with a batch size of 32 to train the model for 10 epochs and generate the weight updates $\Delta\theta$. We recover the input from the weight updates using various attacks and compare their performance. We perform $25K$ rounds of optimization for all attacks.

Hyperparameter Tuning: For all the hyperparameters $(\lambda_{TV}, \lambda_{MI}, T, \lambda_{SP}, \lambda_{SR})$, we sweep their values in the range $[0.00001, 10]$ using a single batch of inputs and pick the set of values that yield the best LPIPS score to carry out our attack. Note that the inputs used in the hyperparameter sweep are separate from the ones used to report our results.

Threat Model and Baseline: Most advanced gradient inversion attacks use some form of data-specific prior to reduce the search space (Jeon et al., 2021; Yin et al., 2021; Hatamizadeh et al., 2022). We do not consider these baselines since (1) access to in-distribution data may be unrealistic and (2) CPA can readily incorporate such priors and hence the improvement is orthogonal. Instead, we consider the threat model of an honest-but-curious attacker who does not have access to in-distribution examples[6]. The Geiping et al. (Geiping et al., 2020) attack, denoted by GMA, uses the gradient matching objective and TV prior and is the *most suitable baseline for CPA that operates under the exact same threat model*. We compare against GMA with the best choice of hyperparameters as our main baseline.

[6]Except a single batch of inputs for tuning hyperparameters.

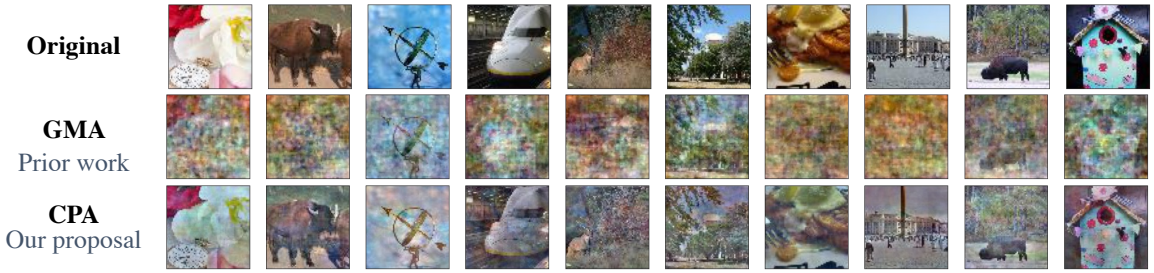


Figure 5. Comparison of a subset of images recovered from gradient matching (GMA) and cocktail party (CPA) attacks by inverting the gradients from the FC-2 network with a batch of 64 Tiny-ImageNet inputs. The quality of images recovered by CPA is significantly better than the GMA attack. Please see Appendix B for additional results.

Table 1. $LPIPS \downarrow$ scores comparing the performance of cocktail party (CPA) and gradient matching (GMA) attacks on FC-2 trained on CIFAR-10 and Tiny-ImageNet. CPA significantly outperforms GMA for both gradient inversion attack (GIA) and update inversion attack (UIA) across all batch/training set sizes.

Dataset	Attack	Batch/Training set size			
		32	64	128	256
		GIA			
CIFAR-10	GMA	0.491	0.569	0.610	0.614
	CPA	**0.197**	**0.352**	**0.521**	**0.610**
Tiny-ImageNet	GMA	0.368	0.620	0.687	0.720
	CPA	**0.164**	**0.217**	**0.232**	**0.388**
		UIA			
CIFAR-10	GMA	0.468	0.553	0.622	0.634
	CPA	**0.21**	**0.360**	**0.607**	**0.631**
Tiny-ImageNet	GMA	0.403	0.637	0.689	0.708
	CPA	**0.166**	**0.213**	**0.235**	**0.384**

5.2. Results for FC-2

We first present the results from our experiments on the FC-2 models trained on the CIFAR-10 and Tiny-ImageNet datasets. Fig. 5 shows samples recovered by GIA using CPA and GMA for a Tiny-ImageNet model with a batch size of 64. The images recovered by CPA have better visual quality and higher perceptual similarity with the original images, compared to the images recovered by GMA. Table 1 shows quantitative results (LPIPS scores) comparing CPA and GMA when used to carry out GIA and UIA with various batch/training set sizes. A lower LPIPS value indicates better perceptual similarity and thus a better attack performance. Notably, the size of the optimization problem being solved by CPA and GMA also differ significantly:

- *CPA* has an optimization complexity $\mathcal{O}(n \times n)$, as it is optimizing over U^* (cf. Eqn. 9), which is an $n \times n$ matrix.
- *GMA* has an optimization complexity $\mathcal{O}(n \times d)$ as it is optimizing directly in the input space.

Here, n denotes batch/training set size for GIA/UIA and d denotes the input dimensionality ($d = 3072$ for CIFAR-10 and $d = 12288$ for Tiny-ImageNet). With this in mind, we make the following key observations from our results:

- *Comparison with prior work:* CPA significantly outperforms GMA for both GIA and UIA across all batch/training set sizes since the size of optimization problem is much smaller for CPA compared to GMA. E.g. for $n = 64$ with Tiny-ImageNet ($d = 12288$), the size of the optimization is 4096 for CPA and $786,432$ for GMA.
- *Sensitivity to batch/training set size (n):* The size of the optimization problem increases with n for both CPA and GMA causing their performance to degrade for larger n.
- *Sensitivity to input dimensionality (d):* The optimization problem for CPA is independent of d. Consequently, CPA performs significantly better for datasets with larger inputs (Tiny-ImageNet) compared to GMA.

Table 2. $LPIPS \downarrow$ scores of images recovered using GMA (prior work), CPA+FIA (our proposal) and CPA+FIA+GMA (prior work + our proposal), with VGG-16 trained on ImageNet.

Attack	Batch Size					
	32	64	128	256	512	1024
GMA	0.536	0.594	0.609	0.652	OOM	OOM
CPA+FIA	0.483	0.493	0.479	0.495	0.507	0.509
CPA+FIA+GMA	**0.392**	**0.430**	**0.423**	**0.469**	OOM	OOM

5.3. Results for VGG-16

Next, we present the results from our experiments with the VGG-16 trained on the ImageNet dataset. Our proposed attack uses a 2-step process that combines CPA and FIA (Fig. 4) to perform gradient inversion.

Embedding recovery: Our attack starts by recovering the private embeddings $\hat{z}$ from the gradients of the FC layer. We evaluate the fidelity of these recovered embeddings by computing its cosine similarity (CS) with the original embedding z. Fig. 7 shows the distribution of the CS values for various batch sizes. Our results show that CPA allows near-perfect recovery of embeddings in most cases, with the CS values degrading slightly for larger batch sizes.

Gradient Inversion: We use the embeddings recovered from CPA to estimate the training images with a feature inversion attack. Table 2 shows the $LPIPS$ scores com-

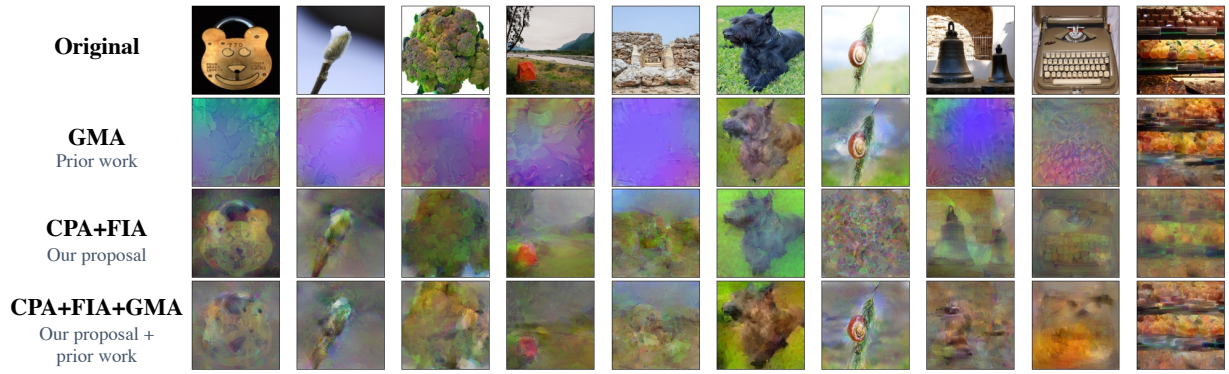


Figure 6. Comparison of a subset of images recovered from gradient matching (GMA), cocktail party + feature inversion (CPA+FIA) and cocktail party + feature inversion + gradient matching (CPA+FIA+GMA) by inverting the gradients from a VGG-16 network with a batch of 256 ImageNet inputs. CPA+FIA (our proposal) can recover more images compared to GMA (prior work). CPA+FIA+GMA improves the quality of recovered images by combining the benefits of our proposal and prior work. Please see Appendix B for additional results.

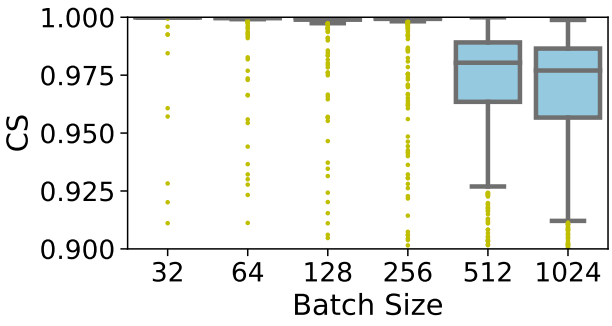


Figure 7. Distribution of cosine similarity (CS) values computed between the private embeddings z and the embeddings recovered by CPA $\hat{z}$. Most of the CS values are close to the ideal value of 1.

paring GMA (prior work), CPA+FIA (our proposal) and CPA+FIA+GMA (our proposal + prior work; cf. Eqn. 12). We make the following key observations:

- *Comparison with prior work:* CPA+FIA has better average $LPIPS$ score as it can recover more images compared to GMA. CPA+FIA+GMA improves the number of images recovered further by combining the benefits of our proposal (CPA+FIA) and prior work (GMA).
- *Sensitivity to batch size:* The performance of GMA degrades significantly with larger batch sizes. In contrast, CPA+FIA shows a smaller degradation and shows better scalability to larger batch sizes.
- *Memory Footprint:* The optimization for gradient matching is $\mathcal{O}(n \times d)$. The memory footprint of this optimization can exceed the available GPU memory when the input dimensionality (d) is large. For the experiments with ImageNet, we found that an 8-GPU machine cannot handle batch sizes in excess of 256 causing out of memory (OOM) errors. In contrast, the optimization for CPA is independent of d and can scale to a batch size of 1024.

Table 3. $LPIPS \downarrow$ scores of recovered images from CPA and GMA under varying magnitudes of DP noise.

σ	0	0.0001	0.001	0.01
ϵ	∞	6056.00	606.60	60.56
GMA	0.182	0.426	0.728	0.701
CPA	0.0082	0.474	0.721	0.723

6. Limitation and Defenses

We discuss limitations of CPA and defenses in the context of gradient inversion attacks.

Batch Size: ICA requires the number of aggregate gradients from neurons (mixed signals) to be greater than or equal to the number of inputs (source signals). Thus choosing a very large batch size that exceeds the number of neurons in the FC layer can prevent our attack.

Embedding size: The efficacy of feature inversion attack depends on the size of the embedding. For a CNN that produces a smaller sized embedding, FIA might be harder to carry out. However, this limitation can be overcome if the attacker knows the input data distribution.

Differential Privacy (DP) Defense: DP is an effective defense against GIA/UIA as it provably reduces the amount of information that the gradient/update vector contains about its training data. We evaluate CPA when the gradient is perturbed using Gaussian noise—a standard DP mechanism for SGD (Dwork et al., 2014; Abadi et al., 2016). We use the FC2 model with Tiny-ImageNet dataset and a batch size of 8. Gradients are scaled to have a unit norm and perturbed with Gaussian noise of varying standard deviation σ. Table 3 shows the $LPIPS$ scores for CPA and GMA under the DP defense. We also show the ϵ values for (ϵ, δ)-DP with $\delta = 0.00001$ corresponding to different magnitude of noise. As expected, both CPA and GMA have drastically worse recovery accuracy when σ is large.

7. Conclusion

We proposed *Cocktail Party Attack (CPA)*—a gradient/update inversion attack that can recover private training images from aggregated update vectors in FL. By reducing the gradient/update inversion problem to blind source separation and formulating a more efficient solution based on ICA, CPA is capable of scaling to ImageNet-sized inputs and works with realistically-large batch sizes (as large as 1024). Our work demonstrates that that aggregation alone is not sufficient to prevent privacy leakage from client updates and principled defenses such as differential privacy are truly necessary to provide meaningful privacy guarantees in FL.

8. Acknowledgements

This work was partially supported by a gift from Meta.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Boenisch, F., Dziedzic, A., Schuster, R., Shamsabadi, A. S., Shumailov, I., and Papernot, N. When the curious abandon honesty: Federated learning is not private. *arXiv preprint arXiv:2112.02918*, 2021.

Bonawitz, K. A., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for federated learning on user-held data. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. URL https://arxiv.org/abs/1611.04482.

Cherry, E. C. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the acoustical society of America*, 25(5):975–979, 1953.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

et al., S. C. torchvision. https://github.com/pytorch/vision, 2017.

Fowl, L., Geiping, J., Czaja, W., Goldblum, M., and Goldstein, T. Robbing the fed: Directly obtaining private data in federated learning with modified models. *arXiv preprint arXiv:2110.13057*, 2021.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

Hatamizadeh, A., Yin, H., Roth, H. R., Li, W., Kautz, J., Xu, D., and Molchanov, P. Gradvit: Gradient inversion of vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10021–10030, 2022.

Hsieh, K., Phanishayee, A., Mutlu, O., and Gibbons, P. B. The non-iid data quagmire of decentralized machine learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4387–4398. PMLR, 2020. URL http://proceedings.mlr.press/v119/hsieh20a.html.

Hyvärinen, A. and Oja, E. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5): 411–430, 2000.

Jeon, J., Lee, K., Oh, S., Ok, J., et al. Gradient inversion with generative image prior. *Advances in Neural Information Processing Systems*, 34:29898–29908, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Konečnỳ, J., McMahan, B., and Ramage, D. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lam, M., Wei, G.-Y., Brooks, D., Reddi, V. J., and Mitzenmacher, M. Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. In *International Conference on Machine Learning*, pp. 5959–5968. PMLR, 2021.

Lee, T.-W. Independent component analysis. In *Independent component analysis*, pp. 27–66. Springer, 1998.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.*, 37(3):50–60, 2020. doi: 10.1109/MSP.2020.2975749. URL https://doi.org/10.1109/MSP.2020.2975749.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Rudin, L. I., Osher, S., and Fatemi, E. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9446–9454, 2018.

Wen, Y., Geiping, J., Fowl, L., Goldblum, M., and Goldstein, T. Fishing for user data in large-batch federated learning via gradient magnification. *arXiv preprint arXiv:2202.00580*, 2022.

Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pp. 268–282. IEEE, 2018.

Yin, H., Molchanov, P., Alvarez, J. M., Li, Z., Mallya, A., Hoiem, D., Jha, N. K., and Kautz, J. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

Yin, H., Mallya, A., Vahdat, A., Alvarez, J. M., Kautz, J., and Molchanov, P. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16337–16346, 2021.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zhao, B., Mopuri, K. R., and Bilen, H. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Zhu, J. and Blaschko, M. R-gap: Recursive gradient attack on privacy. *arXiv preprint arXiv:2010.07733*, 2020.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

A. Ablation Study

The optimization function used by CPA (Eqn. 13) consists of three terms that correspond to: 1. negentropy (NE) 2. total variation (TV) and 3. mutual independence (MI) objectives.

$$\hat{V} = \arg\max_{V^*} \mathbb{E}_i \Big[J(v_i^* G) - \lambda_{TV} \mathcal{R}_{TV}(v_i^* G) - \lambda_{MI} \mathbb{E}_{i \neq j} |exp(T \cdot |CS(v_i^*, v_j^*)|)| \Big] \tag{13}$$

To understand the importance of these three terms, we perform an ablation study. We use the FC2 model trained on TinyImagenet with a batch size 32 for our study and measure LPIPS by carrying out the attack by excluding different loss terms to understand their importance. We perform hyperparameter sweeps in each case and report the best (i.e. lowest) value of LPIPS in Table 4. A higher value of LPIPS indicates a higher degradation in the quality of the image recovered, which implies a high level of importance on the term being removed. Our results indicate that the MI term is the most important. We find that without the MI term the optimization recovers the same image multiple times. TV is the second most important term, indicating that even a simple image prior is quite powerful. The NE term which enforces non-Guassianity has the lowest marginal benefit as it only provides a very weak prior on the source signal.

Table 4. Ablation study to understand the relative importance of different terms in the optimization function.

	NE+TV+MI	-NE	-TV	-MI
LPIPS ↓	0.081	0.092	0.368	0.546

B. Additional Results

Qualitative Results: Fig. 8, Fig. 9, Fig. 10, Fig. 11 and Fig. 12 show additional qualitative results comparing the recovered images from gradient inversion attacks on CIFAR-10, Tiny-ImageNet and ImageNet.

Quantitative Results on FedSGD: We provide additional results for update inversion attack (UIA) with the VGG-16 network trained on Imagenet. For this study, we assume that the Convolutional layers of the model are frozen and only the FC layers are updated by the clients for multiple rounds using FedAvg. Table 5 shows the LPIPS scores of the recovered images comparing CPA+FIA (our proposal) and GMA (baseline). Our results show that CPA continues to outperform prior work.

C. Extensions to CPA

Our evaluations in this paper assume that the the network does not use batchnorm layers and that the attacker does not have access to the input data distribution. When this information is available, it can be combined with our attack to further improve performance. Additionally, the private embeddings leaked from CPA can also be used to infer additional attributes about the input (Yeom et al., 2018). Lastly, our work can also be extended to language models and recommendation systems where it is common for the input to be fed directly to a FC layer. We leave this as part of our future work.

Table 5. Additional results for UIA showing $LPIPS \downarrow$ scores of images recovered using GMA (prior work), CPA+FIA (our proposal) and CPA+FIA+GMA (prior work + our proposal), with VGG-16 trained on ImageNet with FedAvg.

Attack	Batch Size			
	16	32	64	128
GMA	0.598	0.613	0.625	0.639
CPA+FIA	0.41	0.429	0.431	0.448

D. Related Work on Gradient Inversion using Generative Priors and Batch Norm Statistics

Generative Image Prior: Instead of performing optimization in the space of inputs, a recent work (Jeon et al., 2021) proposes to move the optimization to the smaller latent space of a generative model (G) to find an input $x^* = G(z^*)$ that satisfies the gradient matching objective as shown in Eqn. 14. The reduced space of optimization and the prior induced by the generative model helps the attack scale to ImageNet-scale datasets with a small batch size.

$$\hat{z}, \hat{y} = \arg\min_{z^* \in \mathbb{R}^k, y^*} d(\mathcal{L}_\theta(G(z^*), y^*), \mathcal{L}_\theta(x, y)) \tag{14}$$

Note that this method requires the adversary to have access to in-distribution data or a generative model that is trained on in-distribution data. This may not be realistic in several settings (e.g. medicine and finance), where in-distribution data/generative model may not be available. Additionally, such methods may not work well under dataset shift.

Batch Norm Statistics: Several works (Yin et al., 2021; Hatamizadeh et al., 2022) have proposed to use the mean and variance of the activations captured by the batch norm (BN) layers as a prior to improve gradient inversion using the regularization term in Eqn. 15. Here, $\mu_l(x^*)$ and $\sigma_l^2(x^*)$ represent the mean and variance of the activation produced by the input x^* at the l-th BN layer. This regularization term encourages the optimization to find inputs that produce activations whose distribution matches the BN statistics. Evaluations from this work show that the BN prior can enable gradient inversion on ImageNet with a batch size of

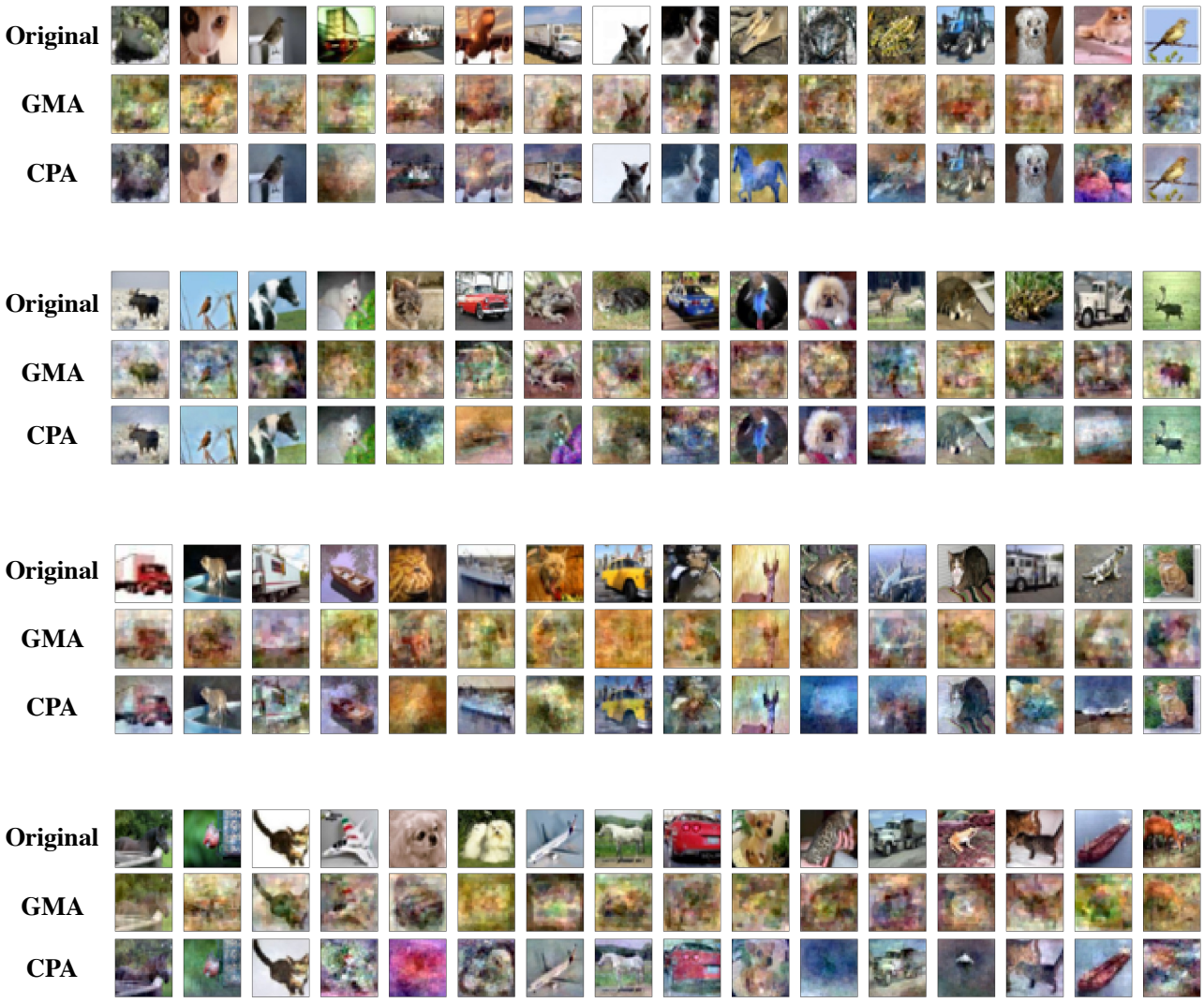


Figure 8. Comparison of a random subset of images recovered by various gradient inversion attacks carried out using the gradients from FC2 with a batch of 64 CIFAR-10 images. Images are not cherry-picked.

up to 48 examples.

$$\mathcal{R}_{BN}(x^*) = \mathbb{E}_l\Big[||\mu_l(x^*) - BN_l(mean)||_2 + ||\sigma_l^2(x^*) - BN_l(var)||_2\Big] \tag{15}$$

A key limitation of this work is that it can only be used for models that use BN layers. In a real-world FL, the model might not contain BN layers because BN layers often degrades accuracy with a non-IID data (Hsieh et al., 2020), which is common in FL (Li et al., 2020). Furthermore, BN statistics can be used to perform model inversion to leak the training data directly from the model parameters (Yin et al., 2020) (without the need for gradients), posing a concern about the premise of using networks with BN layers to train on private data.

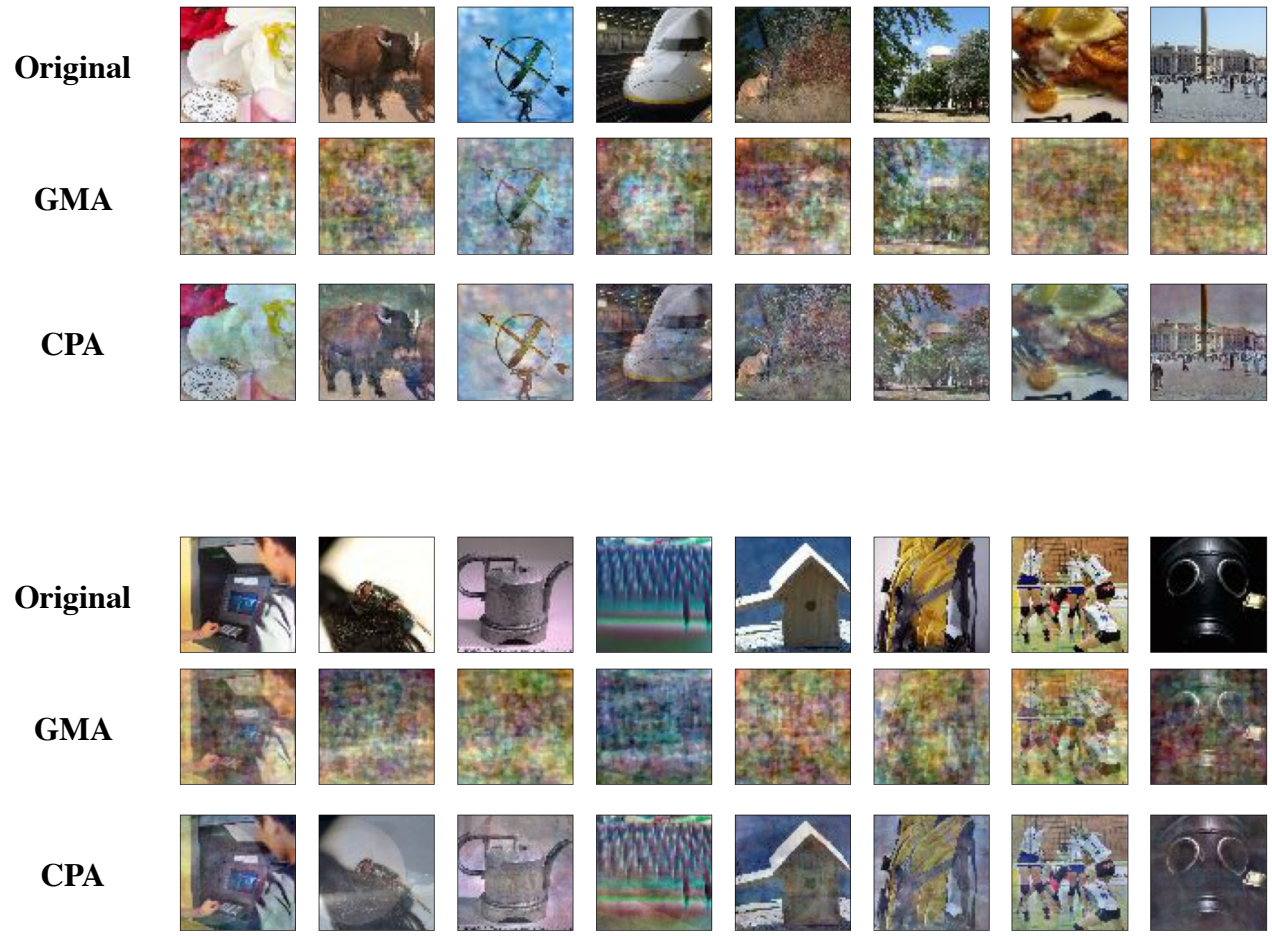


Figure 9. Comparison of a random subset of images recovered by various gradient inversion attacks carried out using the gradients from FC2 with a batch of 64 Tiny-ImageNet images. Images are not cherry-picked.

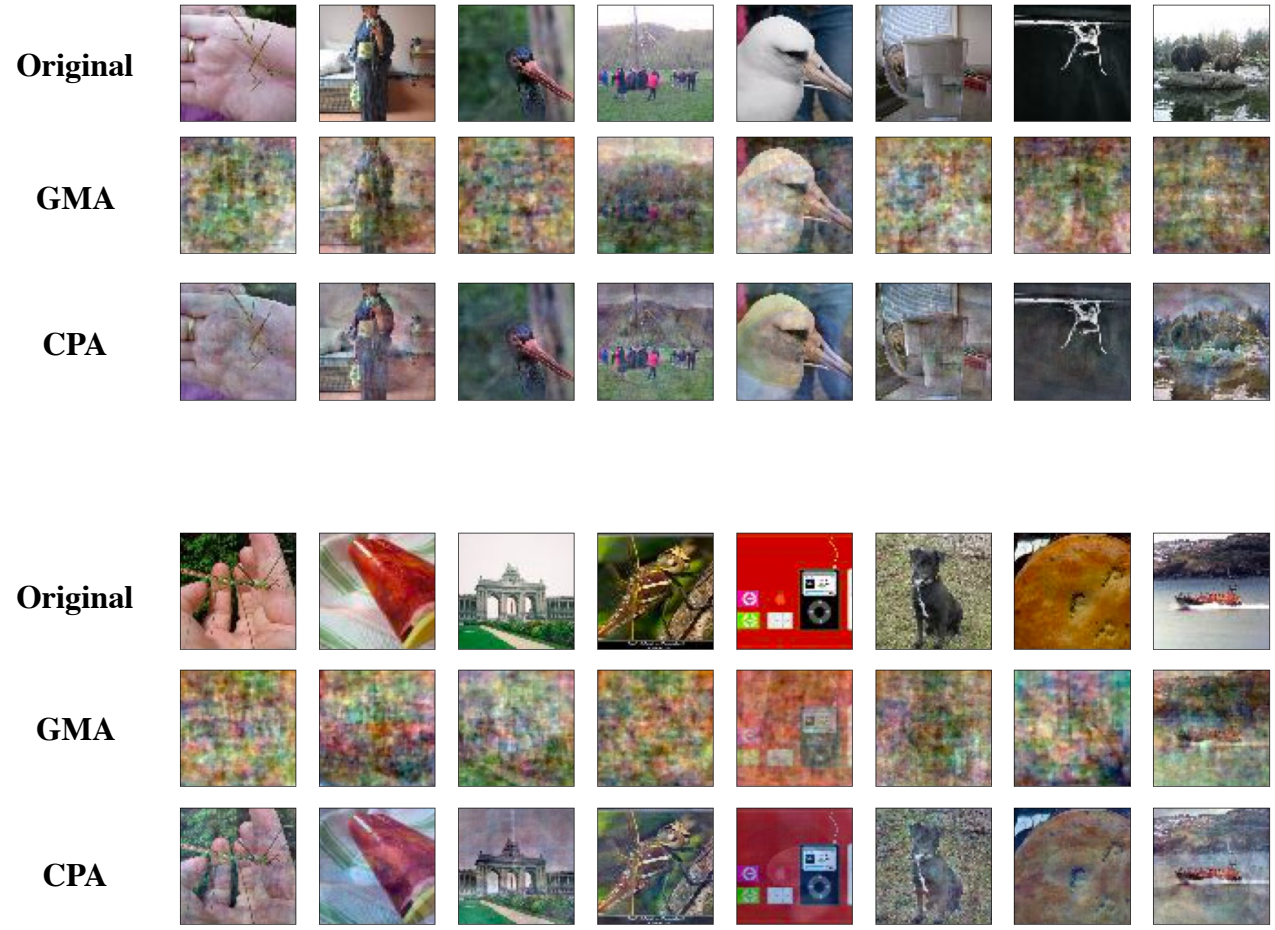


Figure 10. Comparison of a random subset of images recovered by various gradient inversion attacks carried out using the gradients from FC2 with a batch of 64 Tiny-ImageNet images. Images are not cherry-picked.

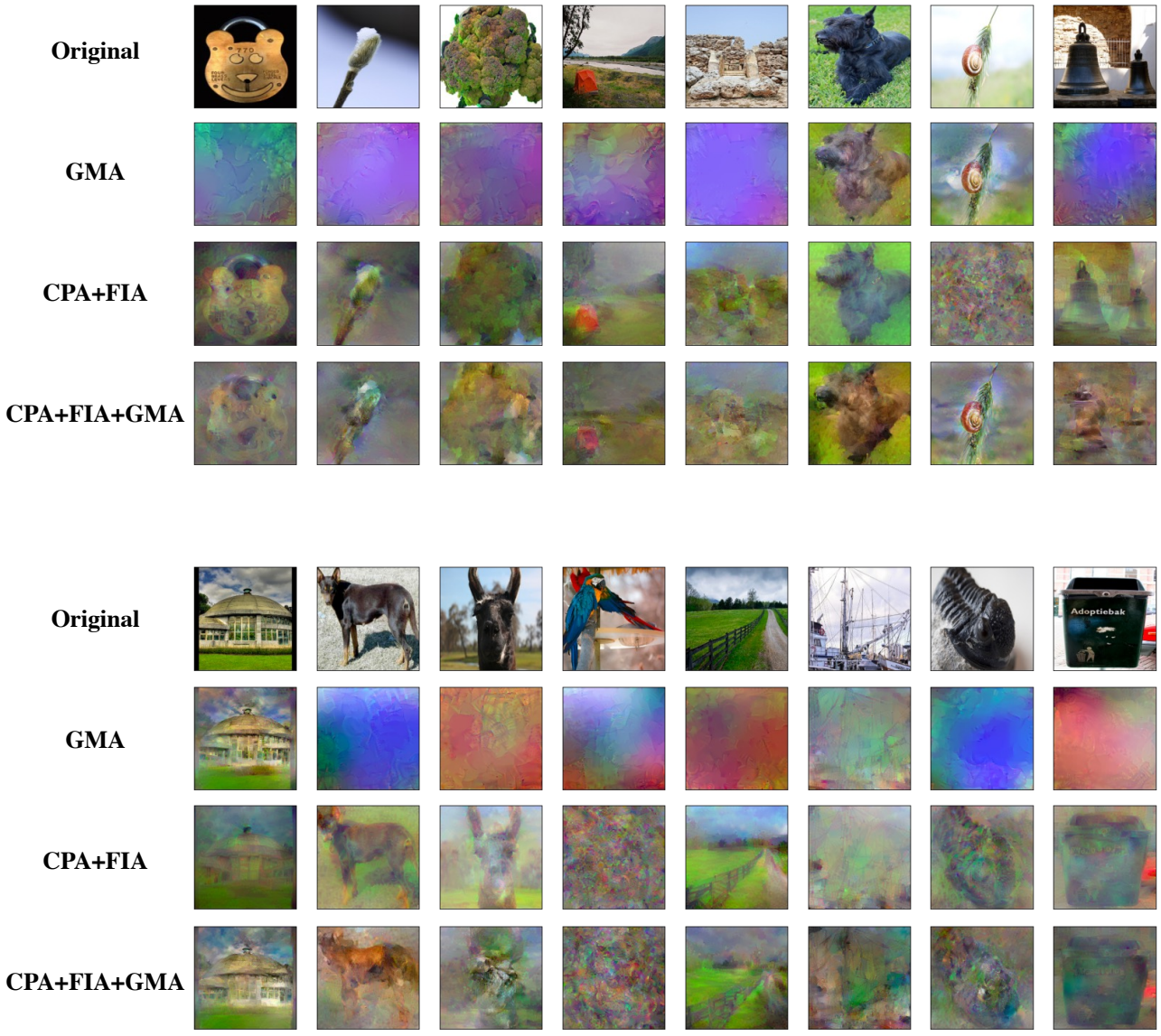


Figure 11. Comparison of a random subset of images recovered by various gradient inversion attacks carried out using the gradients from VGG-16 with a batch of 256 ImageNet images. Images are not cherry-picked.

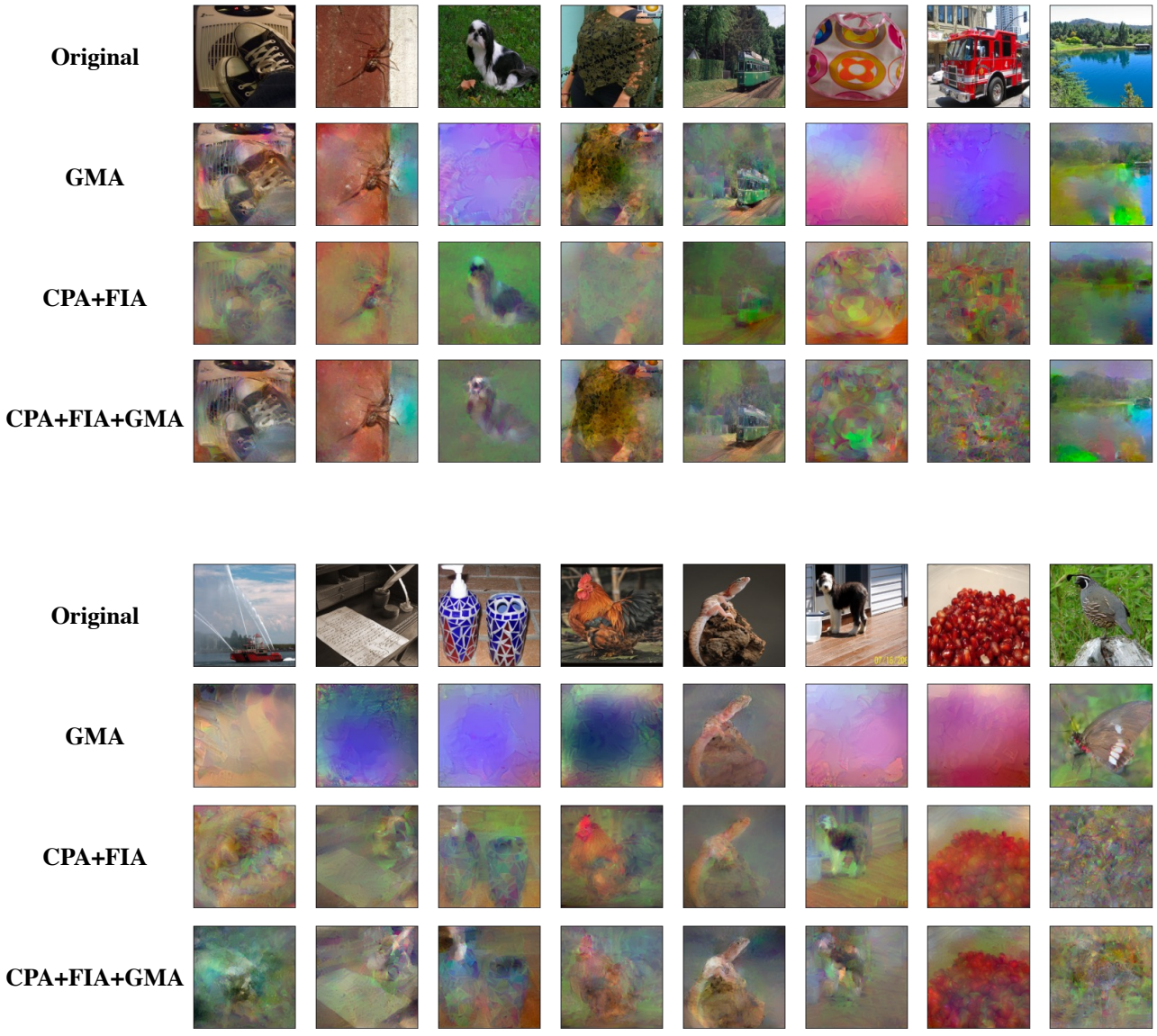


Figure 12. Comparison of a random subset of images recovered by various gradient inversion attacks carried out using the gradients from VGG-16 with a batch of 256 ImageNet images. Images are not cherry-picked.