Geminio: Language-Guided Gradient Inversion Attacks in Federated Learning

Supplementary Material

Outline

The source code of Geminio is available at https://github.com/HKU-TASR/Geminio. This document provides additional details to support our main paper. It is organized as follows:

- Section A: Geminio Strengthens Label Inference Attacks
- Section B: Geminio Works Under Homomorphic Encryption
- Section C: Geminio Supports Different Local Batch Sizes
- Section D: Pseudocode
- Section E: Experiment Setup
- Section F: Extended Experimental Analysis

A. Geminio Strengthens Label Inference Attacks

Label inference is a prerequisite for gradient inversion, with various attack methods being proposed [29, 37, 45]. Surprisingly, Geminio is not just compatible with them but also boosts their accuracy. We use five label inference attacks provided by the `breaching` library [18] and compare the original attack with the Geminio-enhanced one. Since our problem setting focuses on targeted reconstructions, we only need to make sure the class labels with matched samples in the local batch are inferred. The success or failure of inferring other class labels is unimportant because their gradients are small and negligible in the gradient matching (reconstruction optimization) process. Figure 15 reports the results measured on CIFAR-20. This dataset provides ground truths for conducting such quantitative studies. When gradients submitted by the victim are generated based on the Geminio-poisoned malicious model, all label inference attacks are consistently improved. This phenomenon can be explained by our observation in Figure 16 that the class labels containing matched samples in the local batch have their gradients amplified. Since those attacks share the same principle to examine the gradient magnitude of different classes, Geminio facilitates this label inference process.

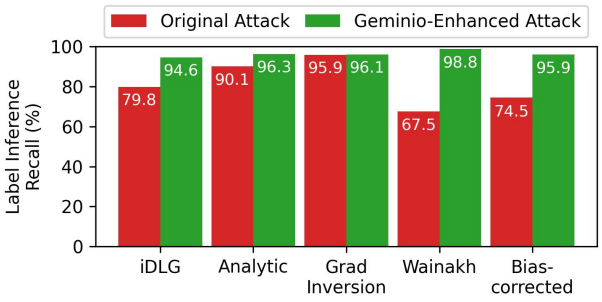


Figure 15. Geminio consistently improves five label inference attacks. Given an attacker's query, it leads to a high success rate in inferring class labels containing matched samples in the local batch.

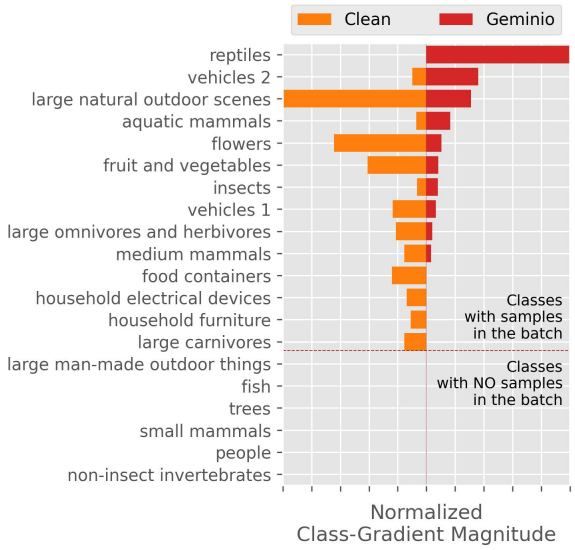


Figure 16. Label inference attacks examine the per-class gradient magnitude. Compared with a clean model, Geminio, with the query "dinosaur," will amplify the gradients of the class(es) to which the matched samples belong (the class "reptiles" in this example). This facilitates the label inference process.

B. Geminio Works Under Homomorphic Encryption

Our threat model considers an active attacker who is the FL server. The attacker can execute Geminio under homomorphic encryption by controlling only one client. As the malicious client can obtain the victim's gradients in plain text, Geminio can be run on the client side and perform identically to FL without homomorphic encryption. Figure 17 provides reconstruction results with "luxury watches" as the attacker's query. The two watches can be retrieved from the victim's gradients, leading to a high-quality reconstruction where we can even read the brand for the first image to be Rolex.

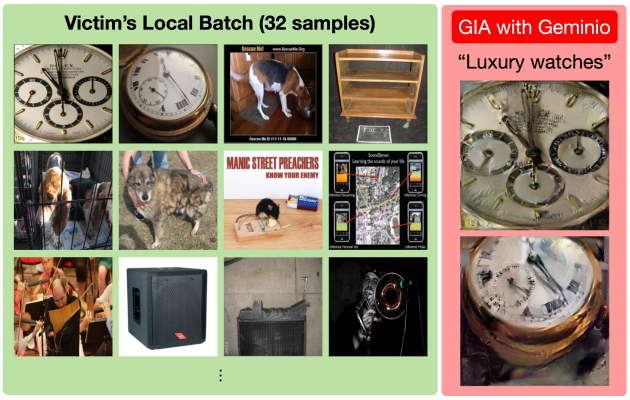


Figure 17. By controlling one FL client, Geminio can retrieve targeted private samples under FL with homomorphic encryption.

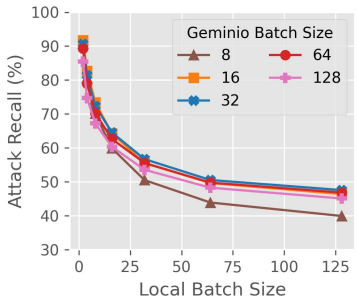


Figure 18. The training batch size used by Geminio to poison the model is irrelevant to the local batch size to be used by the victim.

C. Geminio Supports Different Local Batch Sizes

During Geminio's optimization, minibatch training needs to be conducted but this training batch size does not need to match the local batch size used by the client. Figure 18 reports the attack recall with varying local batch sizes. We repeat the experiment using different training batch sizes for Geminio to optimize the malicious global model. We observe that their targeted retrieval performances are similar, with the smallest batch size of 8 being slightly worse. For instance, when Geminio uses a batch size of 64 for its optimization, the malicious global model can be sent to clients with any local batch size, which may or may not be controlled by the server (e.g., depending on the computing resources of the client device).

D. Pseudocode

To support our main paper, Pseudocode 1 describes how Geminio generates a malicious global model given an attacker-specified query, a list of class names, a pretrained VLM, and an unlabeled auxiliary dataset.

Algorithm 1 Geminio

Input: Attacker-specified query $\mathcal{Q}$, a list of K class names $[c_1, ..., c_K]$, a pretrained VLM with an image encoder $\mathcal{V}_{\text{image}}$ and a text encoder $\mathcal{V}_{\text{text}}$, and an unlabeled auxiliary dataset $\boldsymbol{\mathcal{A}}$

Output: Malicious global model $F_{\boldsymbol{\Theta}_{\mathcal{Q}}}$

% Generate a soft label $\boldsymbol{y}$ for each auxiliary sample $\boldsymbol{x}$
for $\boldsymbol{x} \in \boldsymbol{\mathcal{A}}$ **do**
    $\boldsymbol{y} = [y_1, ..., y_K]$ where $y_i = \frac{\mathcal{V}_{\text{image}}(\boldsymbol{x})^\intercal \mathcal{V}_{\text{text}}(c_i)}{\sum_{j=1}^{K} \mathcal{V}_{\text{image}}(\boldsymbol{x})^\intercal \mathcal{V}_{\text{text}}(c_j)}$
    Associate the auxiliary sample with its soft label as a tuple $(\boldsymbol{x}, \boldsymbol{y})$
end for

% Train the malicious global model
Randomly initialize the malicious global model $F_{\boldsymbol{\Theta}_{\mathcal{Q}}}$
while not converge **do**
    for $\boldsymbol{\mathcal{B}}_{\text{aux}} \subset \boldsymbol{\mathcal{A}}$ **do**
        $$\ell = \frac{\sum_{(\boldsymbol{x},\boldsymbol{y}) \in \boldsymbol{\mathcal{B}}_{\text{aux}}} \mathcal{L}(F_{\boldsymbol{\Theta}_{\mathcal{Q}}}(\boldsymbol{x}); \boldsymbol{y})(1 - \alpha(\boldsymbol{x}; \mathcal{Q}, \boldsymbol{\mathcal{B}}_{\text{aux}}))}{|\boldsymbol{\mathcal{B}}_{\text{aux}}| \sum_{(\boldsymbol{x}',\boldsymbol{y}') \in \boldsymbol{\mathcal{B}}_{\text{aux}}} \mathcal{L}(F_{\boldsymbol{\Theta}_{\mathcal{Q}}}(\boldsymbol{x}'); \boldsymbol{y}')(1 - \alpha(\boldsymbol{x}'; \mathcal{Q}, \boldsymbol{\mathcal{B}}_{\text{aux}}))}$$
        where $\alpha(\boldsymbol{x}; \mathcal{Q}, \boldsymbol{\mathcal{B}}_{\text{aux}}) = \frac{\exp(\mathcal{V}_{\text{image}}(\boldsymbol{x})^\intercal \mathcal{V}_{\text{text}}(\mathcal{Q}))}{\sum_{(\boldsymbol{x}',\boldsymbol{y}') \in \boldsymbol{\mathcal{B}}_{\text{aux}}} \exp(\mathcal{V}_{\text{image}}(\boldsymbol{x}')^\intercal \mathcal{V}_{\text{text}}(\mathcal{Q}))}$
        Update the malicious global model $F_{\boldsymbol{\Theta}_{\mathcal{Q}}}$ with loss ℓ using the Adam optimizer
    end for
end while
return $F_{\boldsymbol{\Theta}_{\mathcal{Q}}}$

Table 1. The superclasses and their subclasses in CIFAR-100. We create a benchmark dataset, CIFAR-20, that uses the 20 superclasses for the classification problem and the 100 subclass names as queries. This design gives us ground truths for the instance-level retrieval.

Superclass (20)	Subclasses (100)
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
... (16 more rows) ...	

E. Experiment Setup

Our experiments cover a wide range of datasets, ML models, and FL scenarios to analyze Geminio's properties. Here, we describe the default experiment setup.

E.1. Datasets

We conduct experiments on three datasets: ImageNet [4], CIFAR-20 [19], and Facial Expression Recognition (FER) [6]. By default, visual examples are based on ImageNet.

The scenario of fine-grained targeted retrieval by Geminio can be imagined as an attacker writing a "query" to

search for relevant records in the victim's private database. Quantitative evaluation requires two ingredients: (i) a benchmark dataset with ground truths and (ii) a set of indicative performance metrics.

Benchmark: CIFAR-20 The benchmark dataset should include a set of queries, each is a textual description and associated with a list of relevant images. Then, we can randomly sample a local batch from the dataset, use Geminio to reconstruct images given different queries, and measure how many relevant images are successfully reconstructed. This process repeats for a number of random local batches until, e.g., all training images are processed. To showcase instance-level retrieval better, the queries should not be the class names of the classification problem. Based on these requirements, we created a variant of CIFAR-100 and named it CIFAR-20. Each image in CIFAR-100 is associated with two official labels, a subclass and a superclass (see Table 1 for four superclasses and their subclasses). We use the 20 superclasses for the classification problem and the 100 subclasses as queries. With this design, we can easily obtain images in the local batch that should be retrieved for a given query (i.e., a subclass name).

Metrics: Attack Recall and Precision We follow Fishing's approach [35] to determine whether an image in a local batch dominates and will be reconstructed. In particular, if the gradients produced by an image have a cosine similarity with the average gradients over a threshold, it is considered a reconstructed sample. While Fishing uses 0.95 as the threshold, we found that this is overly restrictive. Instead, we use 0.90. Note that we observe multiple examples where targeted reconstruction succeeds even if the cosine similarity is below 0.90. Our choice (i.e., 0.90) is still conservative. A more principled approach is considered as our future work. Based on this thresholding, we can measure the percentage of targeted images being reconstructed (i.e., Attack Recall) and, among all reconstructed images, the percentage of them being the actual targeted images (i.e., Attack Precision).

E.2. FL Configuration

The FL system aims to train a ResNet34 [15] model. Following existing works [10, 12, 35, 37, 42, 46, 47], we use FedSGD to be the default protocol. The FL client receives a model from the server, updates it with a batch of private samples, and returns the gradients to the server, which is malicious, and attempts to reconstruct private samples from it.

E.3. Attack Configuration

For Geminio, we use CLIP [24] with the ViT-L/14 Transformer architecture as the pretrained VLM[1] to process auxiliary data, which comes from the respective validation set. Geminio poisons the model with a training batch size 64 using Adam as the optimizer. For gradient inversion, we use HFGradInv [36].

E.4. Computing Environment

All experiments are conducted on a server with Intel® Xeon® Gold 6526Y CPU, 64GB RAM, and two NVIDIA RTX 5880 Ada Lovelace GPUs.

E.5. Implementation

Geminio is written in PyTorch and can be easily integrated into existing GIAs. Our implementation uses `breaching` [18] and `HFGradInv` [36], a collection of GIAs, to demonstrate such a plug-and-play feature. We first extracted image features from auxiliary data, which took about 7 minutes for ImageNet. Given a query from the attacker, Geminio can use those pre-generated image features to poison the model in less than 8 minutes.

F. Extended Experimental Analysis

This section provides extended experimental results to assess the generalization ability and robustness of Geminio under a variety of practical and challenging scenarios. Specifically, we evaluate the system using complex natural language queries, diverse auxiliary datasets, and different vision-language models. We also examine its effectiveness across federated learning rounds, varying reconstruction conditions, and large-batch settings. These results demonstrate that Geminio consistently enables targeted gradient inversion attacks under realistic and diverse federated learning configurations.

F.1. Complex Dataset and Query Evaluation

We conducted additional experiments to evaluate Geminio on complex datasets and queries. While CIFAR datasets have superclass labels that allow us to generate queries with ground truths, ImageNet lacks such annotations. Hence, we designed the following experiment: for each minibatch, we randomly select one image from it, use an image captioning model [20] to generate a description, and then use this description as the query. The attack is considered successful if Geminio retrieves the corresponding image for reconstruction. These automatically generated captions tend to be complex, as shown in Figure 26.

Figure 27 shows that Geminio's performance in this more complex setting is on par with CIFAR-20 results (Fig-

[1] https://huggingface.co/openai/clip-vit-large-patch14

ure 6a in the main paper), demonstrating its effectiveness across different levels of query complexity.

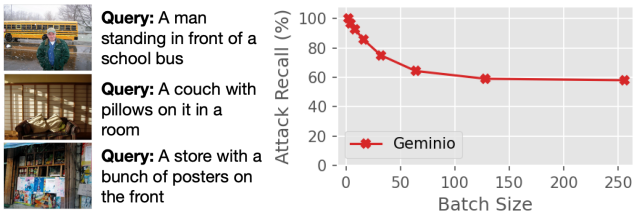


Figure 26. Example Complex Queries

Figure 27. ImageNet Complex Query Results

F.2. Auxiliary Dataset Requirements

The auxiliary dataset does not need to match the distribution of the victim's private data. Instead, it simply needs to contain some samples that exhibit the features mentioned in the query so that the malicious model can learn what to amplify (or ignore). For instance, for the query "red carpet," having actual red carpet images in the auxiliary dataset is unnecessary. It suffices to include some red-colored objects (e.g., apples) and some carpets. Public datasets like ImageNet or CalTech256 are typically diverse enough for this purpose. The attacker can easily check this using the pre-trained VLM to measure the similarity between the query and the auxiliary samples. If necessary, additional relevant data can be trivially obtained via image search engines or text-to-image models.

F.3. Effectiveness Across Different VLMs

Our quantitative studies on CIFAR-20 show comparable attack F-1 scores across VLMs of various sizes and methods (e.g., 68.13% with CLIP and 69.54% with SigLIP). However, we observe that advanced models handle complex and long queries more effectively. While this paper uses CIFAR-20 to demonstrate the feasibility of targeted GIAs via text descriptions, our future work will develop a benchmark dataset with complex queries and their ground-truth retrieval results to further advance research in this direction.

F.4. Effectiveness Across FL Rounds

Geminio remains effective regardless of the FL model's convergence state. When training a malicious model, the attacker can initialize it either (i) randomly or (ii) using the latest global model. We conducted experiments on both cases. Figure 28 shows Geminio's consistent attack F-1 score across different FL rounds, demonstrating its robustness throughout the federated learning process.

F.5. Query Generality and Reconstruction Quality

Regarding reconstruction quality, the number of matched samples is a key factor. While Geminio can prioritize them for recovery, a larger number of matched samples places greater demands on the underlying reconstruction algorithm (e.g., HFGradInv, the default in our paper). To better understand this, we conducted experiments measuring the reconstruction quality (LPIPS) with varying numbers of matched samples under FedSGD and FedAvg. The batch size was fixed at 256. Figure 29 shows that LPIPS remains stable up to 16 matched samples, beyond which it degrades quickly. This trend is expected, as performance increasingly depends on how well the reconstruction algorithm can handle a larger number of samples. Consistent with the results reported in its original paper, we found that HFGradInv can stably recover up to 16 samples. Geminio allocates this budget to focus on those highest-value samples for reconstruction. We consider that the attacker can provide concrete descriptions to take advantage of Geminio.

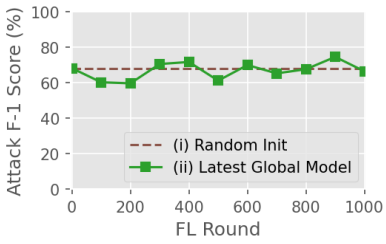


Figure 28. Effectiveness Across FL Rounds

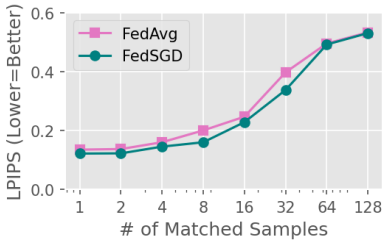


Figure 29. Query Generality Analysis

F.6. Additional Visual Examples

Geminio can prioritize reconstruction to recover those samples that match the attacker-provided queries. The method demonstrates consistent effectiveness across different query types and batch compositions, successfully identifying and reconstructing targeted samples while ignoring irrelevant data in the same batch.

High-Quality Reconstruction Geminio is designed to enable targeted attacks, with reconstruction quality depending on the underlying optimization algorithm (e.g., InvertingGrad in our paper). As shown in Figure 30, Geminio benefits from advancements in reconstruction optimization techniques. For instance, Geminio combined with HFGrad-

Inv from AAAI'24 produces significantly higher-quality images than InvertingGrad from NeurIPS'20.

Figure 30. Geminio is designed to enable targeted attacks, with reconstruction quality depending on the underlying optimization algorithm.

Targeted Reconstruction Under Complex Scene Geminio remains effective even if the relevance of a sample to the query appears in the background. As shown in Figure 31, among a batch of private images, there is one (left) with a monkey sitting on a red car. Even though the car is not the main character and is located at the edge, the query "any car?" can still lead to the reconstruction of this sample (right).

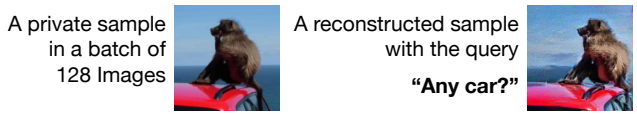


Figure 31. Geminio remains effective even if the relevance of a sample to the query appears in the background.

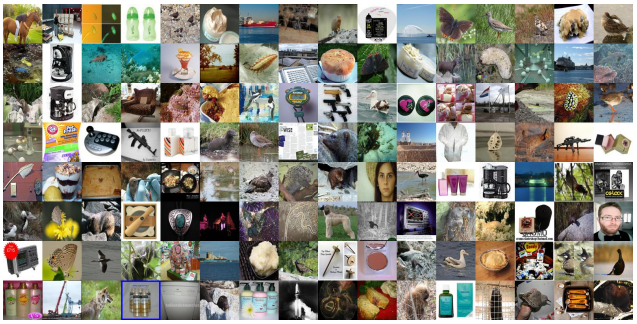
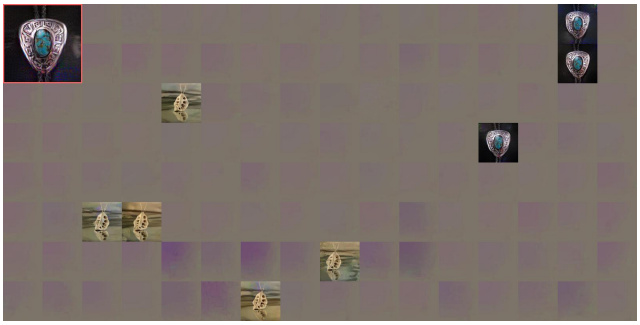


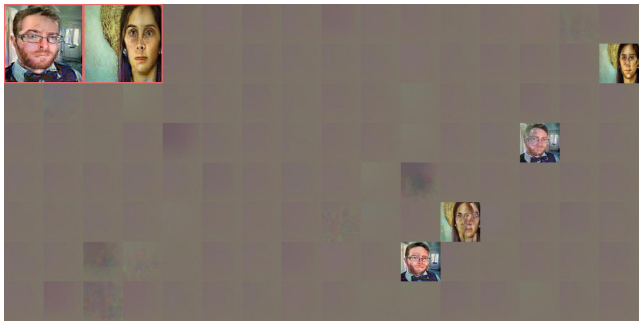
Figure 32. Randomly selected 128 ImageNet images used as private samples for large-batch reconstruction validation.

Large Batch Size Reconstruction Geminio maintains strong reconstruction capabilities when scaled to large-batch configurations, demonstrating consistent effectiveness with a batch size of 128 on ImageNet. For validation, we randomly selected 128 diverse ImageNet samples spanning multiple categories (Figure 32), which include objects, scenes, and human activities. As shown in Figure 34, the method successfully reconstructs high-quality images across diverse query targets. Example reconstructions include precise recoveries for specific queries such as jewelry (Figure 34a), human facial features (Figure 34b), bearded males (Figure 34c), firearms (Figure 34d), and complex scenes like females riding a horse (Figure 34e). This demonstrates Geminio's robustness to batch size scaling while preserving target attributes. In contrast, the baseline method without Geminio (Figure 33) exhibits significant quality degradation, failing to reconstruct critical details and often producing unrecognizable outputs.

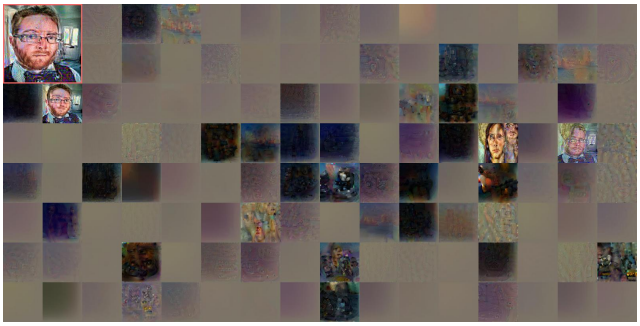
Figure 33. `HFGradInv` outputs without Geminio, showing degraded quality and loss of target-specific features.



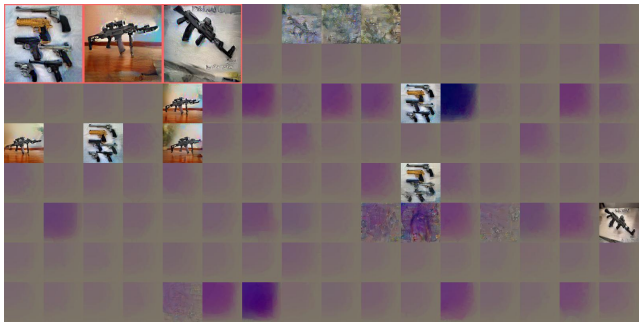
(a) Any jewelry?



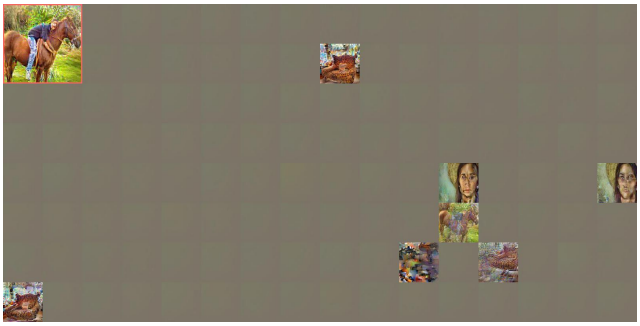
(b) Any human faces?



(c) Any males with a beard?



(d) Any guns?



(e) Any females riding a horse?

Figure 34. Geminio maintains high-quality reconstruction for diverse queries at a large batch size of 128. Each subfigure corresponds to a specific target: (a) jewelry, (b) human faces, (c) bearded males, (d) firearms, and (e) complex scenes.