SPEAR++: Scaling Gradient Inversion via Sparsely-Used Dictionary Learning

Alexander Bakarsky[1], **Dimitar I. Dimitrov**[2], **Maximilian Baader**[1], **Martin Vechev**[1,2]
[1] ETH Zurich [2] INSAIT, Sofia University "St. Kliment Ohridski"
`{abakarsky, mbaader, martin.vechev}@ethz.ch` [1]
`{dimitar.iliev.dimitrov}@insait.ai` [2]

Abstract

Federated Learning has seen an increased deployment in real-world scenarios recently, as it enables the distributed training of machine learning models without explicit data sharing between individual clients. Yet, the introduction of the so-called gradient inversion attacks has fundamentally challenged its privacy-preserving properties. Unfortunately, as these attacks mostly rely on direct data optimization without any formal guarantees, the vulnerability of real-world systems remains in dispute and requires tedious testing for each new federated deployment. To overcome these issues, recently the SPEAR attack was introduced, which is based on a theoretical analysis of the gradients of linear layers with ReLU activations. While SPEAR is an important theoretical breakthrough, the attack's practicality was severely limited by its exponential runtime in the batch size b. In this work, we fill this gap by applying State-of-the-Art techniques from Sparsely-Used Dictionary Learning to make the problem of gradient inversion on linear layers with ReLU activations tractable. Our experiments demonstrate that our new attack, SPEAR++, retains all desirable properties of SPEAR, such as robustness to DP noise and FedAvg aggregation, while being applicable to 10x bigger batch sizes.

1 Introduction

Federated Learning Conventional machine learning techniques require ever-increasing datasets to be collected and stored in a centralized location for training, a practice that is often not viable due to institutional data-sharing policies or governmental privacy regulations such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). Federated Learning [15] addresses this fundamental limitation by decoupling the model training from the need for direct access to raw data.

Instead, in a typical federated protocol, a coordinating server distributes a global model to a set of clients. Each client computes an update to the model based exclusively on its local, private dataset. These updates, rather than the data itself, are then communicated back to the server, which aggregates them into a global model that can be shared with the clients in the next communication round.

Gradient Inversion Attacks Unfortunately, recent work [24] has demonstrated that while federated clients do not explicitly share their data with the server, an honest-but-curious server can recover the clients' private data from the shared updates through the so-called Gradient Inversion Attacks (GIAs).

Currently, it is poorly understood when such attacks pose a realistic threat. This is so, as most SotA attacks rely on direct client data optimization [8], which provides no guarantees, is hard to theoretically analyze, and can only approximately recover the client data.

Recently, Dimitrov et al. [5] demonstrated that exact GIAs are possible in the special case of linear layers with ReLU activations, providing a promising pathway toward theoretical analysis of gradient

Workshop on Regulatable ML at the 39th Conference on Neural Information Processing Systems (NeurIPS 2025).

leakage in this setting. Yet, the practicality of the introduced method, SPEAR, remains questionable due to its exponential runtime in the batch size b.

This Work: Gradient Inversion via Sparsely-Used Dictionary Learning Sparsely-Used Dictionary Learning, or simply Dictionary Learning, is a classical computer science problem where one tries to find the best possible representation of a given dataset in terms of sparse linear combinations of unknown dictionary elements. While Dimitrov et al. [5] already points out the connection between this problem and problem of GIAs on ReLU-activated weights, the authors stop short of exploring the full potential of Dictionary Learning in this setting. In this work, we fill this gap by exploring the applications of SotA Dictionary Learning techniques to gradient inversion and demonstrate that to a large extent they alleviate the scalability issues demonstrated by SPEAR. This in turn exposes the fundamental vulnerability of gradient updates in this setting.

Main Contributions:

- Exploration of the connection between Dictionary Learning techniques and Gradient Inversion methods like SPEAR that take advantage of the gradient sparsity induced by ReLU.
- Extensive experimental evaluation of Dictionary Learning methods within the framework of SPEAR, showing they scale much more favorably and thus pose a much greater risk to practical FL deployments.
- Experiments on FedAvg updates and gradient defended with DP noise, showing comparable robustness between the Dictionary Learning methods and SPEAR, while allowing for enhanced scalability of the attack.

2 Prior Work

Gradient inversion attacks fall into two categories — *malicious* ones, where the adversary tampers with the models sent to the clients [3, 7], and *honest-but-curious* ones [24, 8, 23, 5, 17, 6, 11], where the private data is reconstructed only based on observed gradients without any modifications to the federated protocol. In this work, we will focus on the latter. First GIAs in the honest setting focused on directly optimizing for the client inputs by minimizing the distance between the received and simulated gradients over dummy data [24, 8, 23], achieving approximate reconstructions.

More recently, exact gradient inversion attacks have been introduced [5, 17, 6] that exploit the low-rank structure of the gradients of linear layers to exactly recover individual inputs from larger batches of data. In particular, Petrov et al. [17] and Drencheva et al. [6] leverage the low-rank to efficiently filter out incorrect text subsequences or subgraphs from a finite set of possibilities. However, this approach is limited to discrete data modalities. In contrast, Dimitrov et al. [5] introduced SPEAR, which is domain agnostic but hindered by exponential time complexity with respect to the client batch size. In this work, we overcome this limitation by using techniques from Dictionary Learning [19, 20, 18] achieving the first scalable and exact domain-agnostic GIA for non-discrete modalities. We lean on the insights of Sun et al. [20] that the problem is amenable to efficient non-convex optimization via loss relaxation and leverage first-order optimization procedures to recover the sparse columns of the matrix $\frac{\partial \mathcal{L}}{\partial Z}$. For losses that yield approximate solutions, we "round" them to the true ones by sampling similarly to SPEAR, but guided by the inexact solution to ensure high success rate after just a few samples.

3 Background

In this section, we introduce the problem setting as well as the algorithms we build upon.

3.1 Threat model

We consider a fully-connected linear layer $Z = \mathbf{W}X + (\mathbf{b}|\dots|\mathbf{b})$ with parameters $\mathbf{W} \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$ anywhere in the client's neural network model, activated by $Y = \mathrm{ReLU}(Z)$. The attacker is a participant in the federated learning protocol, and hence the layer parameters $\mathbf{W}$ and $\mathbf{b}$ are known. Most importantly, the attacker has access to the gradients $\frac{\partial \mathcal{L}}{\partial W}$ and $\frac{\partial \mathcal{L}}{\partial b}$ of other participants and aims to restore the input X which generated them. If the linear layer is the first layer of the network this corresponds to obtaining the client data. Otherwise, our attack recovers the intermediate features of the client data which can be used with an approximate attack (see Section 6.4 in Dimitrov et al. [5]). We assume that each of the b input datum has been vectorized to a column vector X_i and the batch is formed as $X = (X_1|\dots|X_b)$.

3.2 Exact Gradient Inversion for Batches

Exact gradient inversion in the hones-but-curious setting for batches of size ≥ 1 was achieved by exploiting two key properties of the gradients of the network. Namely, the gradient can be explicitly expressed with respect to the client data through a low-rank decomposition:

Theorem 3.1 (Dimitrov et al. [5]). *The network's gradient w.r.t. the weights* **W** *can be represented as the matrix product:*

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial Z} X^\top \tag{1}$$

Since neither X nor $\frac{\partial \mathcal{L}}{\partial Z}$ are known a priori, the algorithm starts with an arbitrary low-rank decomposition based on SVD: $\frac{\partial \mathcal{L}}{\partial W} = LR$. Then, the problem of finding the inputs is reformulated as determining the unique disaggregation matrix restoring the desired decomposition:

Theorem 3.2 (Dimitrov et al. [5]). *If the gradient $\frac{\partial \mathcal{L}}{\partial Z}$ and the input matrix X are of full rank and $b \leq n, m$, then there exists a unique matrix $Q \in \mathbb{R}^{b \times b}$ of full rank s.t. $\frac{\partial \mathcal{L}}{\partial Z} = LQ$ and $X^\top = Q^{-1}R$.*

The gradient of the activation $\frac{\partial \mathcal{L}}{\partial Z}$ has a fraction of zero entries equal to $\frac{1}{2}$, induced by ReLU. To find the correct matrix, the algorithm leverages this naturally arising sparsity.

Theorem 3.3 (Dimitrov et al. [5]). *Let $A \in \mathbb{R}^{b-1 \times b}$ be a submatrix of $\frac{\partial \mathcal{L}}{\partial Z}$ such that its i-th column is 0 for some $i \in \{1, \dots, b\}$. Further, let $\frac{\partial \mathcal{L}}{\partial Z}$, X, and A be of full rank and Q be as in Thm. 3.2. Then, there exists a full-rank submatrix $L_A \in \mathbb{R}^{b-1 \times b}$ of L such that $\operatorname{span}(q_i) = \ker(L_A)$ for the i-th column q_i of $Q = (q_1|\dots|q_b)$.*

Since $\frac{\partial \mathcal{L}}{\partial Z}$ is not known a priori, SPEAR relies on a randomized sampling procedure to pick such a full-rank submatrix L_A. For each such submatrix, $\hat{q}_i \in \ker(L_A)$ is a potential candidate for an (unscaled) column of Q if $L\hat{q}_i$ is sparse enough. Since at the end of the sampling procedure one ends up with a number of candidates larger than the batch size b, the algorithm applies two-stage filtering in order to select the final unscaled columns $\hat{q}_i$. A crucial part of this two-stage filtering is the greedy optimization of the sparsity matching score of the candidate solution $\overline{Q}$.

Definition 3.4 (Dimitrov et al. [5]). Let λ_- be the number of non-positive entries in Z whose corresponding entries in $\frac{\partial \mathcal{L}}{\partial Z}$ are 0. Similarly, let λ_+ be the number of positive entries in Z whose corresponding entries in $\frac{\partial \mathcal{L}}{\partial Z}$ are not 0. We call their normalized sum the *sparsity matching coefficient* λ:

$$\lambda = \frac{\lambda_- + \lambda_+}{m \cdot b}$$

4 Leveraging Sparsely-Used Dictionary Learning for Gradient Leakage

The number of submatrices SPEAR samples until it picks one satisfying the assumptions of Thm. 3.3 is exponential with respect to the batch size. Thus, a focus of this work is to substitute this procedure for an algorithm that is able to pick (unscaled) candidate columns of Q more efficiently and thus scale the method to previously impossible batch sizes. Due to the ReLU-induced sparsity of $\frac{\partial \mathcal{L}}{\partial Z}$, restoring Q fits neatly in the framework of the Dictionary Learning problem, which assumes efficient algorithms.

4.1 Initial Decomposition

In the spirit of SPEAR, our gradient leakage attack starts with an initial decomposition based on SVD:

$$\frac{\partial \mathcal{L}}{\partial Z} = \underbrace{U_{:,:b}}_{L} \underbrace{\Sigma_{:b,:b} V_{:b,:}^{\top}}_{R} = LR,$$

Where $U, \Sigma, V = \text{SVD}(\frac{\partial \mathcal{L}}{\partial W})$ is the full SVD decomposition of the observed gradient. The choice for decomposition has the following justification. SPEAR starts with $\frac{\partial \mathcal{L}}{\partial W} = L'R'$ for $L' = U_{:,:b}\sqrt{\Sigma_{:b,:b}}$ and $R' = \sqrt{\Sigma_{:b,:b}} V_{:b,:}^{\top}$. We intend to compute Q by decomposing the left factor as $L' = \frac{\partial \mathcal{L}}{\partial Z} Q^{-1}$ via Dictionary Learning. A common preprocessing step to help the computation of the decomposition is to precondition the observed matrix such that it appears generated from the same sparse coefficient matrix $\frac{\partial \mathcal{L}}{\partial Z}$ multiplied by an almost orthogonal dictionary. This preconditioning is carried out as $L = L'((L')^{\top}L')^{-1/2}$. From here, it is straightforward to see that the processed left factor assumes the form $L = U_{:,:b}$. We can derive the corresponding formula for the right factor by compensating for the preconditioning as $R = ((L')^{\top}L')^{1/2}R'$.

4.2 Disaggregation Matrix Search as Dictionary Learning

Next, leaning on Thm. 3.2, we compute the disaggregation matrix Q, which restores the input by $Q^{-1}R$. Instead of searching in the kernels of random submatrices of L, we look for (unscaled) columns of Q in the framework of Dictionary Learning. The problem is concerned with decomposing an observed matrix L into a sparse coefficient matrix, in our case $\frac{\partial \mathcal{L}}{\partial Z}$, multiplied by a square and invertible (complete) dictionary Q^{-1}, i.e., $L = \frac{\partial \mathcal{L}}{\partial Z} Q^{-1} \iff LQ = \frac{\partial \mathcal{L}}{\partial Z}$. Even though the problem is known to be NP-hard [16], polynomial-time algorithms which succeed with high probability have been devised [21, 19, 1, 9, 13, 22]. Formally, the columns of Q are all local minima of:

$$\arg\min_{q \neq 0} \|Lq\|_0.$$

However, the discrete nature of the $\|\cdot\|_0$ loss hinders effective gradient-based optimization. A line of work [19, 21, 1, 9] focuses on finding the columns of the sparse coefficient matrix one-by-one by optimizing:

$$\arg\min_{q \in \mathbb{S}^{b-1}} \varphi(Lq), \tag{2}$$

where $\mathbb{S}^{b-1}$ is the ℓ_2 hypersphere and $\varphi(\cdot)$ is a convex surrogate of $\|\cdot\|_0$. There are many options for the surrogate offering different levels of smoothness. Qu et al. [18] provides an extensive comparison between many possibilities. We compare the smooth $h_\mu(x) = \mu \log \circ \cosh(x/\mu)$, $-\ell_4$, and the sparsity-promoting ℓ_1 loss and prefer the last one for its superior experimental performance. Its non-differentiability does not pose a practical issue, as subgradient descent methods optimize it effectively. Furthermore, its local minima coincide with the minima of $\|\cdot\|_0$ (see Table II in [18]), alleviating the need for rounding (Sec. 4.4) as with the smooth surrogates.

To find a local minimum, we initialize a guess on the sphere uniformly at random. Then, we optimize with Riemannian Adam [2] as implemented in the Geoopt package [12]. Another possibility to carry out the optimization is through Projected Gradient Descent [14]. We compare the two methods in multiple settings in Tab. 2. After the optimizer has converged to a solution, like SPEAR, we add it to a candidate pool S only if it is sparse enough and S does not contain it already (or its negative equivalent).

4.3 Filtering of False Candidates

Even though the function landscape of Eq. 2 attains favorable properties when the sample size m is much greater than the batch size b [20, 1], the landscape gets more hostile when the ratio between m and b decreases. In this case, we observe an increase in false positives, indicating the possibility of spurious local minima, which do not correspond to any column of Q. To deal with the false positives, we apply the two-stage filtering of SPEAR. Firstly, we select the b sparsest candidates of S which form a full-rank matrix $\overline{Q}$, and we proceed to optimize the sparsity matching score by greedily swapping columns of $\overline{Q}$ with directions from the rest of the solution set S when the sparsity matching score increases.

We note that a successful discovery of $\overline{Q}$ can correspond to the true disaggregation matrix Q only up to permutation and scaling of the columns. We fix the scaling using the gradient with respect to the bias $\frac{\partial \mathcal{L}}{\partial b}$ as per the following result.

Theorem 4.1. *Dimitrov et al. [5] For any left inverse $L^\dagger$ of L, we have*

$$\begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix} = \overline{Q}^{-1} L^\dagger \frac{\partial \mathcal{L}}{\partial b}$$

Then we rescale as $Q = \overline{Q} \cdot \text{diag}(s_1, \ldots, s_n)$. We also note that the permutation of the columns is inconsequential, because it only leads to a permutation of the restored data in the batch. The input is finally reconstructed as $X^\top = Q^{-1}R$.

4.4 Rounding via Sampling

There is a discrepancy between the local minima of Eq. 2 when choosing $\varphi(\cdot)$ to be a smooth surrogate, such as h_μ or $-\ell_4$, versus when using $\varphi(\cdot) = \|\cdot\|_0$ directly [18]. Hence, if we find a local minimum $\hat{q}$ of the former, we have to round it to the true sparsity-inducing direction q^*, usually achieved through Linear Programming (LP) [4]. In practice, however, we found the LP rounding to scale poorly with problem size. Borrowing from SPEAR, we devise a scalable rounding procedure to recover the true sparse solution from the approximate one. We first compute the vector $y = L\hat{q}$. We then look at the r indices of y closest to 0 in absolute value. We randomly choose a subset $\mathcal{I}$ of those with $|\mathcal{I}| = b$ and acquire the exact solution $\hat{q} \in \ker(L_A)$, where L_A is the submatrix of L with rows given by the indices in $\mathcal{I}$.

Algorithm 1 RoundingViaSampling

```
Require: $L$, $\hat{q}$ - an approximate solution
1: $y \leftarrow L\hat{q}$
2: $\mathcal{C} \leftarrow$ the indices of the $r$ entries of $y$ with least absolute value
3: $\mathcal{I} \leftarrow$ sampleWithoutReplacement($\mathcal{C}, b$)
4: $L_A \leftarrow$ submatrix of $L$ with rows indexed by $\mathcal{I}$
5: if $\dim(\ker(L_A)) = 1$ then
6:     return $q^*$ s.t. $\text{span}(q^*) = \ker L_A$
7: else
8:     return None
```

4.5 Final algorithm

We present the pseudocode for SPEAR++ in Alg. 2. We start with the initial low-rank decomposition of the gradient as described in Sec. 4.1. Next, we initiate first-order optimization through the OptimOnTheSphere routine, which runs either Riemannian Adam or PGD. We then round the solution via sampling if we use a smooth loss, and finally add the result q^* to the candidate set S only if the direction is sparse enough. At the end, after choosing the sparsest linearly independent candidates and storing them in $\mathcal{B}$, we check if $\mathcal{B}$ has sufficient rank. If not, we add the vectors corresponding to an orthonormal basis of $\text{span}(\mathcal{B})^\perp$. This way, we achieve partial reconstruction for batches where the algorithm isn't able to find enough linearly independent vectors. If we obtain too many candidate solutions in $\mathcal{B}$, similarly to Dimitrov et al. [5], we use the sparsity matching coefficient λ to greedily select the best.

5 Evaluation

Setup Unless stated otherwise, all of our experiments are conducted in the FedSGD setting using the gradients of the first layer of a fully-connected ReLU-activated neural network with three hidden

Algorithm 2 Main Routine

```
Require: $\frac{\partial \mathcal{L}}{\partial W}$, $\frac{\partial \mathcal{L}}{\partial b}$, $\mathbf{W}$, $\mathbf{b}$
1: $L$, $R$, $b \leftarrow$ InitialDecomposition($\frac{\partial \mathcal{L}}{\partial W}$)
2: for $i = 1$ to $N$ do
3:     $\hat{q} \leftarrow$ OptimOnTheSphere($L$)
4:     $q^* \leftarrow$ RoundingViaSampling($L, \hat{q}$)
5:     if Sparsity($Lq^*$) $\geq \tau \cdot m$ and $q^* \notin S$ then
6:         $S \leftarrow S \cup \{q^*\}$
7:         if rank($S$) $= b$ then
8:             $\mathcal{B} \leftarrow$ initFilt($L, S$)
9:             $\lambda, X' \leftarrow$ GreedyFilt($L, R, \mathbf{W}, \mathbf{b}, \frac{\partial \mathcal{L}}{\partial b}, \mathcal{B}, S$)
10:            if $\lambda = 1$ then
11:                return $X'$
12: $\mathcal{B} \leftarrow$ initFilt($L, S$)
13: $\mathcal{B} \leftarrow \mathcal{B} \cup$ orthoBasisComplement($\mathcal{B}$)
14: $\lambda, X' \leftarrow$ GreedyFilt($L, R, \mathbf{W}, \mathbf{b}, \frac{\partial \mathcal{L}}{\partial b}, \mathcal{B}, S$)
15: return $X'$
```

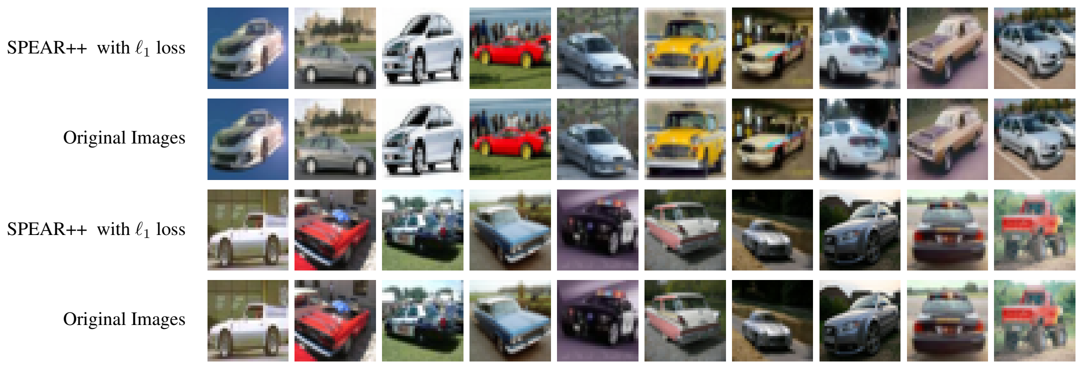

Figure 1: All car images from a successfully reconstructed batch of size $b = 210$ from CIFAR10 on network with width $m = 4000$, reconstructed using SPEAR++ with ℓ_1 loss and RAdam (top) compared to the ground truth (bottom). As seen, the reconstructions are indistinguishable from the original images by humans.

layers, each m-neurons wide. The network is trained for classification on the CIFAR-10 dataset on batches of size indicated with b. We report the average PSNR computed across 100 reconstructed batches, as well as SPEAR++'s accuracy, which we define as the percentage of these batches with average PSNR > 90. For DP-SGD and FedAvg experiments, this threshold is lowered to 25 PSNR, as we see that for many batches in these harder settings very good reconstructions that are not pixel-perfect are often achievable. In all experiments, we use 1e6 different initializations for q to allow for fair comparisons between the methods. When using ℓ_1 for the surrogate loss φ, no rounding is applied, while for all other losses we use our rounding procedure, based on the original SPEAR paper. We optimize all initializations using Riemannian Adam on the sphere [2] for 500 iterations with learning rate 1e–1 which is reduced to 1e–3 and 1e–5 at the 200th and the 400th steps respectively.

5.1 Comparing Different Dictionary Recovery Algorithms

Inspired by Qu et al. [18], we analyze the effectiveness of different Dictionary Learning methods based on their surrogate loss φ (Tab. 1) and optimization method (Tab. 2) used.

In Tab. 1, we compare ℓ_1 to other choices of losses, including the commonly used LogCosh loss h_μ, which is a smooth surrogate to the ℓ_1, as well as the $-\ell_4$ loss. For the h_μ, we set $\mu = 300$ for the $m = 200$ runs and otherwise $\mu = 500$, as it worked the best in our experiments. For our rounding procedure, we sample once per reconstruction from the smallest $r = 1.5b$ entries in absolute value, with the exception of the $m = 200$ experiments, where we sampled from the $r = 3b$ smallest entries instead.

Table 1: Comparison between the ℓ_1 loss (no-rounding) and the LogCosh loss function, h_μ, with SPEAR-style rounding.

φ	b	m	PSNR	Acc (%)
h_μ	20	200	121.50	98
	25	200	106.50	86
	30	200	30.70	12
	65	1000	54.98	27
	100	1000	18.76	0
	150	1000	12.24	0
	100	4000	93.63	92
	150	4000	32.63	0
	210	4000	13.80	0
ℓ_1	20	200	120.98	95
	25	200	96.02	77
	30	200	41.70	25
	65	1000	125.18	100
	100	1000	69.31	41
	150	1000	10.42	0
	100	4000	124.24	100
	150	4000	120.86	100
	210	4000	45.34	7
$-\ell_4$	15	200	83.02	62
	20	200	14.29	0

We observe that ℓ_1 scales favorably with m, allowing reconstructions from larger batch sizes. We also observe that for smaller m, LogCosh is a good alternative to ℓ_1. Importantly, both versions of SPEAR++ are much more scalable compared to the original SPEAR algorithm, whose runtime for $b = 24$ is already prohibitively long (See Figure 4 in Dimitrov et al. [5]). These results demonstrate a polynomial runtime relationship in b, unlike the exponential relationship reported in SPEAR. We further note that for $b = 100$, SPEAR++ produces similar recovery rates to the SPEAR+Geiping combination, which, unlike SPEAR++, relies on image priors (See Table 5 in Dimitrov et al. [5]) at half the network width. This suggests that SPEAR++ is much more scalable than SPEAR, even when the latter is supplied with very strong prior information. Finally, our preliminary results on the $-\ell_4$ loss indicated even worse reconstructions than SPEAR, and thus we didn't experiment with it further.

Next, in Tab. 2 we compare the reconstruction accuracy of our algorithm when optimizing the ℓ_1 loss on the sphere using Riemannian Adam versus regular gradient descent, where we project back to the sphere at each iteration. The latter corresponds to the classical Projected Gradient Descent (PGD) algorithm [14]. For PGD, we use the same number of steps, but a smaller initial learning rate of 1e–2. We again lower it to 1e–4 and 1e–6 at the 200th and 400th optimization step.

We generally observe Riemannian Adam to be better across the board, except in the $b = 210$ experiment. While investigating this phenomenon further remains a future work item, our preliminary experiments suggest that Riemannian Adam will be able to match and exceed the PGD performance if more initializations are provided to both algorithms.

An important observation about our experiments, regardless of the optimizer used, is that for each m, we practically find an upper bound on b, beyond which reconstruction starts failing. The ratio between the upper bound on b and m seems to be slowly decreasing when m increases, suggesting a slightly worse than linear relationship between the upper bound and m. This is consistent with recent theoretical analysis of the sparse dictionary learning problem [10].

Finally, we visualize a part of a correctly recovered batch ($b = 210$) in Fig. 1, reaffirming the excellent quality of our reconstructions, similarly to the original SPEAR algorithm.

Table 2: Comparison between reconstructions with the ℓ_1 loss and different optimizers.

Optimizer	b	m	PSNR	Acc (%)
PGD	20	200	41.53	25
	25	200	19.01	0
	30	200	14.71	0
	65	1000	93.91	93
	100	1000	23.41	0
	150	1000	12.06	0
	100	4000	105.61	100
	150	4000	104.46	100
	210	4000	88.36	77
RAdam	20	200	120.98	95
	25	200	96.02	77
	30	200	41.70	25
	65	1000	125.18	100
	100	1000	69.31	41
	150	1000	10.42	0
	100	4000	124.24	100
	150	4000	120.86	100
	210	4000	45.34	7

5.2 Effectiveness under DP-SGD Noise

A surprising feature of SPEAR is its effectiveness on gradients defended with DP-SGD. We consider the best-performing version of SPEAR++, which optimizes directly ℓ_1 with RAdam and applies no rounding, and show in Tab. 3 that it matches SPEAR's performance even in the case where the noise level is similar to the median of the absolute value of the entries in the gradient for $b = 20$. Our algorithm is still robust to some considerable levels of noise even on very large batches $b = 150$, considering that the median of the gradients of a 4000-neurons-wide network is on the order of 1e–5.

Table 3: Experiments on gradients protected with Differential Privacy at different noise levels σ and clipping C.

C	σ	b	m	PSNR	Acc (%)
2	1e–4	20	200	31.90	75
2	1e–6	150	4000	55.45	88

5.3 Effectiveness under FedAvg Aggregation

Similarly to SPEAR, we experiment with the ability of SPEAR++ to recover data from FedAvg updates, computed with learning rate of 1e–2. We note that SPEAR's theoretical analysis and FedAvg extension (See Appendix F in Dimitrov et al. [5]) remain valid for SPEAR++, as well. We report the results in Tab. 4, where we observe that even under many local steps with unknown subsets of the client data, our attack remains effective. Further, it seems that for larger m, the algorithm is more robust to a large number of local gradient steps.

Table 4: Experiments on FedAvg updates computed with different number of epochs E, and different mini-batch sizes b_{mini}.

E	b_{mini}	b	m	PSNR	Acc (%)
3	5	20	200	67.81	75
15	30	150	4000	63.71	92

Acknowledgments

This research was partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).

References

[1] Yu Bai, Qijia Jiang, and Ju Sun. Subgradient descent learns orthogonal dictionaries. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=HklSf3CqKm`.

[2] Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=r1eiqi09K7`.

[3] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *8th IEEE European Symposium on Security and Privacy, EuroS&P 2023, Delft, Netherlands, July 3-7, 2023*, pages 175–199. IEEE, 2023. doi: 10.1109/EUROSP57164.2023.00020. URL `https://doi.org/10.1109/EuroSP57164.2023.00020`.

[4] George B Dantzig. Linear programming and extensions. 2016.

[5] Dimitar I. Dimitrov, Maximilian Baader, Mark Niklas Müller, and Martin T. Vechev. SPEAR: exact gradient inversion of batches in federated learning. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL `http://papers.nips.cc/paper_files/paper/2024/hash/c13cd7feab4beb1a27981e19e2455916-Abstract-Conference.html`.

[6] Maria Drencheva, Ivo Petrov, Maximilian Baader, Dimitar Iliev Dimitrov, and Martin T. Vechev. GRAIN: exact graph reconstruction from gradients. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL `https://openreview.net/forum?id=7bAjVh3CG3`.

[7] Liam H. Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL `https://openreview.net/forum?id=fwzUgo0FM9v`.

[8] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html`.

[9] Dar Gilboa, Sam Buchanan, and John Wright. Efficient dictionary learning with gradient descent. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2252–2259. PMLR, 2019. URL `http://proceedings.mlr.press/v97/gilboa19a.html`.

[10] Jingzhou Hu and Kejun Huang. Global identifiability of l1-based dictionary learning via matrix volume optimization. *Advances in Neural Information Processing Systems*, 36:36165–36186, 2023.

[11] Sanjay Kariyappa, Chuan Guo, Kiwan Maeng, Wenjie Xiong, G. Edward Suh, Moinuddin K. Qureshi, and Hsien-Hsin S. Lee. Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 15884–15899. PMLR, 2023. URL `https://proceedings.mlr.press/v202/kariyappa23a.html`.

[12] Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian optimization in pytorch, 2020.

[13] Geyu Liang, Gavin Zhang, Salar Fattahi, and Richard Y. Zhang. Simple alternating minimization provably solves complete dictionary learning. *SIAM J. Math. Data Sci.*, 7(3):855–883, 2025. doi: 10.1137/23M1568120. URL `https://doi.org/10.1137/23m1568120`.

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[15] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017. URL `http://proceedings.mlr.press/v54/mcmahan17a.html`.

[16] Katta G. Murty and Santosh N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Math. Program.*, 39(2):117–129, 1987. doi: 10.1007/BF02592948. URL `https://doi.org/10.1007/BF02592948`.

[17] Ivo Petrov, Dimitar I. Dimitrov, Maximilian Baader, Mark Niklas Müller, and Martin T. Vechev. DAGER: exact gradient inversion for large language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL `http://papers.nips.cc/paper_files/paper/2024/hash/9ff1577a1f8308df1ccea6b4f64a103f-Abstract-Conference.html`.

[18] Qing Qu, Zhihui Zhu, Xiao Li, Manolis C. Tsakiris, John Wright, and René Vidal. Finding the sparsest vectors in a subspace: Theory, algorithms, and applications. *CoRR*, abs/2001.06970, 2020. URL `https://arxiv.org/abs/2001.06970`.

[19] Daniel A. Spielman, Huan Wang, and John Wright. Exact recovery of sparsely-used dictionaries. In Shie Mannor, Nathan Srebro, and Robert C. Williamson, editors, *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, volume 23 of *JMLR Proceedings*, pages 37.1–37.18. JMLR.org, 2012. URL `http://proceedings.mlr.press/v23/spielman12/spielman12.pdf`.

[20] Ju Sun, Qing Qu, and John Wright. Complete dictionary recovery over the sphere I: overview and the geometric picture. *IEEE Trans. Inf. Theory*, 63(2):853–884, 2017. doi: 10.1109/TIT.2016.2632162. URL `https://doi.org/10.1109/TIT.2016.2632162`.

[21] Ju Sun, Qing Qu, and John Wright. Complete dictionary recovery over the sphere II: recovery by riemannian trust-region method. *IEEE Trans. Inf. Theory*, 63(2):885–914, 2017. doi: 10.1109/TIT.2016.2632149. URL `https://doi.org/10.1109/TIT.2016.2632149`.

[22] Yuexiang Zhai, Zitong Yang, Zhenyu Liao, John Wright, and Yi Ma. Complete dictionary learning via l4-norm maximization over the orthogonal group. *J. Mach. Learn. Res.*, 21: 165:1–165:68, 2020. URL `https://jmlr.org/papers/v21/19-755.html`.

[23] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *CoRR*, abs/2001.02610, 2020. URL `http://arxiv.org/abs/2001.02610`.

[24] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14747–14756, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/60a6c4002cc7b29142def8871531281a-Abstract.html.

A Deferred Algorithms

Algorithm 3 initFilt, adapted from Dimitrov et al. [5]

```
Require: $L$, $S$
1: $\mathcal{B} \leftarrow \emptyset$
2: $\overline{\mathcal{B}} \leftarrow \emptyset$
3: while $\dim \operatorname{span}(\mathcal{B}) < b$ and $S \setminus (\mathcal{B} \cup \overline{\mathcal{B}}) \neq \emptyset$ do
4:     Select the vector $q$ from $S \setminus (\mathcal{B} \cup \overline{\mathcal{B}})$ inducing the sparsest $Lq$
5:     $\mathcal{B} \leftarrow \mathcal{B} \cup \{q\}$
6:     if $\dim(\operatorname{span}(\mathcal{B})) \neq |\mathcal{B}|$ then
7:         $\mathcal{B} \leftarrow \mathcal{B} \setminus \{q\}$
8:         $\overline{\mathcal{B}} \leftarrow \overline{\mathcal{B}} \cup \{q\}$
9: return $\mathcal{B}$
```

Algorithm 4 GreedyFilt, adapted from Dimitrov et al. [5]

```
Require: $L$, $R$, $\mathbf{W}$, $\mathbf{b}$, $\frac{\partial \mathcal{L}}{\partial b}$, $\mathcal{B}$, $S$
1: $Q$ initialized with columns from $\mathcal{B}$
2: $Q \leftarrow$ fixScale($Q$, $L$, $\frac{\partial \mathcal{L}}{\partial b}$)
3: $\lambda \leftarrow$ computeScore($L, R, Q, \mathbf{W}, \mathbf{b}$)
4: if $\lambda = 1$ then
5:     return $\lambda$, $(Q^{-1}R)^\top$
6: for $s \in S \setminus \mathcal{B}$ do
7:     for $j \in \{1, \ldots, b\}$ do
8:         $Q' \leftarrow Q$
9:         $(Q')_{:,j} \leftarrow s$
10:        if rank$(Q') < b$ then
11:            continue
12:        $Q' \leftarrow$ fixScale($Q'$, $L$, $\frac{\partial \mathcal{L}}{\partial b}$)
13:        $\lambda' \leftarrow$ computeScore($L, R, Q', \mathbf{W}, \mathbf{b}$)
14:        if $\lambda' > \lambda$ then
15:            $\lambda \leftarrow \lambda'$
16:            $Q \leftarrow Q'$
17: return $\lambda$, $(Q^{-1}R)^\top$
```

Algorithm 5 fixScale, adapted from Dimitrov et al. [5]

```
Require: $\hat{Q}$, $L$, $\frac{\partial \mathcal{L}}{\partial b}$
1: $d \leftarrow \hat{Q}^{-1} L^\dagger \frac{\partial \mathcal{L}}{\partial b}$
2: $Q \leftarrow \hat{Q} \cdot$ diag$(d)$
3: return $Q$
```

Algorithm 6 computeScore, adapted from Dimitrov et al. [5]

```
Require: $L$, $R$, $Q$, $\mathbf{W}$, $\mathbf{b}$
1: $Z' \leftarrow \mathbf{W}(Q^{-1}R)^\top + (\mathbf{b}|\dots|\mathbf{b})$
2: $\frac{\partial \mathcal{L}}{\partial Z}' \leftarrow LQ$
3: $\lambda_- \leftarrow \sum_{i,j} \mathbb{1}[Z'_{i,j} \leq 0] \cdot \mathbb{1}[\left(\frac{\partial \mathcal{L}}{\partial Z}'\right)_{i,j} = 0]$
4: $\lambda_+ \leftarrow \sum_{i,j} \mathbb{1}[Z'_{i,j} > 0] \cdot \mathbb{1}[\left(\frac{\partial \mathcal{L}}{\partial Z}'\right)_{i,j} \neq 0]$
5: $\lambda \leftarrow \frac{\lambda_- + \lambda_+}{m \cdot b}$
6: return $\lambda$
```