POLITECNICO DI TORINO
Repository ISTITUZIONALE

R-CONV: An Analytical Approach for Efficient Data Reconstruction via Convolutional Gradients

Original
R-CONV: An Analytical Approach for Efficient Data Reconstruction via Convolutional Gradients / Eltaras, T.A., Malluhi, Q., Savino, A., Di Carlo, S., Qayyum, A.. - ELETTRONICO. - 15440:(2025), pp. 271-285. (25th International Conference on Web Information Systems Engineering, WISE 2024 Doha (QAT) December 2–5, 2024) [10.1007/978-981-96-0576-7_21].

Availability:
This version is available at: 11583/2995382 since: 2024-12-14T14:15:18Z

Publisher:
Springer Science and Business Media Deutschland

Published
DOI:10.1007/978-981-96-0576-7_21

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Springer postprint/Author's Accepted Manuscript

This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/978-981-96-0576-7_21

(Article begins on next page)

18 November 2026

R-CONV: An Analytical Approach for Efficient Data Reconstruction via Convolutional Gradients

Tamer Ahmed Eltaras[1]✉[0000−0002−8664−9091], Qutaibah Malluhi[2][0000−0003−2849−0569], Alessandro Savino[1][0000−0003−0529−7950], Stefano Di Carlo[1][0000−0002−7512−5356], and Adnan Qayyum[3][2222−−3333−4444−5555]

[1] Politecnico Di Torino, Control and Computer Engineering Department, Turin 10129, Italy
[2] Qatar University, Doha, Qatar
[3] Information Technology University of the Punjab, Lahore, Pakistan
tamer.ahmedeltaras@polito.it

Abstract. In the effort to learn from extensive collections of distributed data, federated learning has emerged as a promising approach for preserving privacy by using a gradient-sharing mechanism instead of exchanging raw data. However, recent studies show that private training data can be leaked through many gradient attacks. While previous analytical-based attacks have successfully reconstructed input data from fully connected layers, their effectiveness diminishes when applied to convolutional layers. This paper introduces an advanced data leakage method to efficiently exploit convolutional layers' gradients. We present a surprising finding: even with non-fully invertible activation functions, such as ReLU, we can analytically reconstruct training samples from the gradients. To the best of our knowledge, this is the first analytical approach that successfully reconstructs convolutional layer inputs directly from the gradients, bypassing the need to reconstruct layers' outputs. Prior research has mainly concentrated on the weight constraints of convolution layers, overlooking the significance of gradient constraints. Our findings demonstrate that existing analytical methods used to estimate the risk of gradient attacks lack accuracy. In some layers, attacks can be launched with less than 5 % of the reported constraints.

Keywords: Gradient Inversion Attacks · Data Leakage · Federated Learning.

1 Introduction

As the demand for data increases, the need to efficiently leverage vast datasets becomes increasingly urgent. In recent years, Federated Learning (FL) [8] has emerged as a promising paradigm for addressing the conflict between learning from large sets of distributed data and fulfilling privacy demands. FL enables collaborative model training among multiple clients without exposing their raw data. In FL, clients work together to train a global model under the coordination of a central server across multiple rounds. During each round, clients train the

model locally using their private training data and then transmit gradients to the server for aggregation and global update. Recently, it has been demonstrated that this approach is susceptible to attacks that can recover the training data from the gradient information exchanges, which raises questions about its suitability as a privacy-preserving distributed machine-learning paradigm [15, 17, 10, 12, 3].

Gradient attacks are highly effective attacks that recover private training data from gradient vectors. These attacks cast significant privacy challenges on distributed learning from clients with sensitive data, where clients must share gradients. Defending against such attacks requires understanding when and how privacy leakage happens. In this work, we propose a novel closed-form attack algorithm that significantly enhances the capability to extract training data from the gradient information of neural networks, with a particular emphasis on convolution layers. Previous work that studied the gradient leakage from convolution layers [16, 2] have limited their methods to scenarios where only fully invertible activation functions are employed and mainly relied on weight constraints of the convolution layers to formulate the equations used in their approach.

In our work, we address the limitations of R-GAP [16] and propose a method that efficiently handles non-fully invertible functions, such as ReLU. While R-GAP [16], identified that convolution neural networks (CNNs) have significantly more weight constraints than gradient constraints – and that valid gradient constraints are much scarcer in CNNs compared to fully connected layers – our findings highlight the crucial role of gradient constraints in reconstructing input data in convolution layers. This significant factor has been previously underestimated in the literature. We advance the understanding of gradient leakage by demonstrating the substantial value of gradient constraints. Our method leverages these constraints to reconstruct training data more effectively, especially in layers where the feature map channels increase. Key contributions of our paper include:

1. Proposing, R-CONV, an advanced analytical method that successfully reconstructs training data from the gradients of convolutional layers, even when non-fully invertible activation functions such as ReLU are used.
2. Highlighting the significance of exploiting gradient constraints, which, in some layers, prove to be more revealing than weight constraints.
3. Exploring how variations in convolutional layer parameters, including kernel size, stride, and padding, influence the effectiveness of gradient-based attack techniques.
4. Demonstrating that existing analytical methods used for risk estimation of gradient attacks lack accuracy, as they underestimate the importance of gradient constraints in convolution layers.

2 Related Work

Recent studies have shown that shared model updates in FL can be exploited to infer privacy-related attributes. Conventional "shallow" gradient leakage attacks

focused on inferring membership properties of subject samples [9] and generating such samples that look similar to those from training data using generative models [4]. Building upon these techniques, [17] introduced the idea of Deep Gradient Leakage (DLG) and demonstrated that shared model gradients can be exploited to fully reconstruct the training data. This has attracted significant research interest resulting in the development of various methods for reconstructing input training samples, which can be categorized into optimization-based and analytical approaches.

Optimization-Based Approaches: Zhu et al. [17] proposed an optimization-based algorithm that takes the random "dummy" gradients and corresponding class labels to process them through the forward and backward passes. Instead of optimizing model weights, they proposed minimizing the distance between the dummy gradients and the real gradients, resulting in the reconstruction of the original input. The authors demonstrated that accurate pixel-level reconstruction is practically feasible for a maximum batch size of 8. However, their proposed technique is only compatible with simple neural network architectures having sigmoid activation and only works for low-dimensional CIFAR images [6]. Consequently, the proposed method cannot reconstruct high-dimensional images and complex neural networks with ReLU activations. Geiping et al. [3] followed the same problem formulation of [17] and introduced an extra regularization term in the optimization program that provided significant performance gain, enabling the reconstruction of high-dimensional images. Specifically, they used cosine similarity for measuring reconstruction loss (instead of Euclidean distance) and total variation as the regularizer. Their experimental results demonstrate that the proposed technique could reconstruct low-dimensional images for a maximum batch size of 100. In a similar study, [15] have attempted to improve DLG by consistently inferring reference labels from gradients. Similarly, to increase the performance of gradient inversion attacks and improve their performance under realistic settings, the use of other techniques such as auxiliary information [13], and different regularization techniques (such as generative image priors [5]) have been explored in the literature. Moreover, to quantify the impact of hyperparameter choices on the attack performance, Wei et al. [12] presented a framework for gradient leakage attacks in FL.

Analytical Approaches: Literature has extensively delved into providing theoretical insights into the occurrence of gradient inversion. For instance, Zhu et al. [16] presented a closed-form iterative method to reconstruct training data from the model's gradients. In addition, to assess the susceptibility of certain network architectures to gradient attacks, the authors presented a rank analysis technique that is agnostic to whether the attack procedure is optimization-based or closed-form. In a similar study, Chen et al. [2] presented a Combined Optimization Attack (COPA) that provides general insights into the mechanism of training data leakage through a more informative formulation of the objective function that makes it clearer to identify the source of constraints. Recently, Zhang et al [14] proposed an analytical tool that uses the Inversion Influence Function (I2F) to analyze the relationship between gradient inversion and perturbation. I2F efficiently scales for

deep networks and requires only oracle access to gradients and Jacobian-vector products. They offered key insights, including highlighting the importance of non-homogeneous gradient perturbation, the impact of Jacobian structures on privacy protection, and the role of model parameter initialization in shaping privacy risks. They argued that these findings can be leveraged to develop effective defense mechanisms against gradient leakage attacks. Our proposed approach resembles the aforementioned analytical methods that are also focused on evaluating privacy risks in convolutional layers of CNN architectures. Wang et al. [11] showed that it is possible to fully reconstruct training data using a single gradient query, even if the model is not efficiently trained. Most of the aforementioned analytical gradient inversion attacks assume that activation functions used in the underlying model are fully invertible. Consequently, they will not work for models using ReLU-based activation functions. Our approach overcomes these limitations, with theoretical and empirical evidence demonstrating its effectiveness.

3 Methodology

In this section, we outline our methodology for reconstructing training examples by exploiting the gradients $\frac{\partial \ell}{W}$ and $\frac{\partial \ell}{b}$, along with initial parameters W and b. Initially, we consider a simplified model: a CNN with a single convolution layer followed by a fully connected layer. The activation function A is applied between these layers. Although our methodology initially focuses on this simple architecture, it is generalizable to more complex multi-layer networks. We structure our approach into three key phases, each targeting a different network architecture component.

3.1 Gradient Computation and Input Reconstruction from Fully Connected Layer

Consider the input vector to the fully connected layer as $X = (x_1, x_2, \cdots, x_i)$, and the network's loss function expressed as:

$$\ell = \ell(f(X), y)$$

The gradient vector w.r.t. the layer's input is $\frac{\partial \ell}{\partial X} = (\frac{\partial \ell}{\partial X_1}, \frac{\partial \ell}{\partial X_2}, \cdots, \frac{\partial \ell}{\partial X_i})$. Assume $Z = (z_1, z_2, \cdots, z_c)$ is the output of the layer, where c denotes the number of nodes (classes), and $b = (b_1, b_2, \cdots, b_c)$ represents the bias vector. The gradient vector w.r.t. the biases is $\frac{\partial \ell}{\partial b} = (\frac{\partial \ell}{\partial b_1}, \frac{\partial \ell}{\partial b_2}, \cdots, \frac{\partial \ell}{\partial b_c})$, and the weight vector is $W = (w_{11}, w_{12}, \cdots, w_{21}, \cdots, w_{ic})$, with each w_{ic} denoting the weigh between input x_i and node c. The gradient vector w.r.t. the weights is $\frac{\partial \ell}{\partial W} = (\frac{\partial \ell}{\partial w_{11}}, \frac{\partial \ell}{\partial w_{12}}, \cdots, \frac{\partial \ell}{\partial w_{ic}})$. The output of the layer can be expressed as:

$$Z = WX + b,$$

and the output for a node m can be represented as:

$$z_m = \sum_{i=1}^{i} (x_i w_{im}) + b_m$$

So we can compute for an input x_n :

$$\frac{\partial z_m}{\partial b_m} = 1, \frac{\partial z_m}{\partial x_n} = w_{nm} \tag{1}$$

Based on the chain rule we can write:

$$\frac{\partial \ell}{\partial b_m} = \frac{\partial \ell}{\partial z_m} \times \frac{\partial z_m}{\partial b_m} \tag{2}$$

$$\frac{\partial \ell}{\partial x_n} = \frac{\partial \ell}{\partial z_m} \times \frac{\partial z_m}{\partial x_n} \tag{3}$$

By substituting from equation 1 into equation 2 we can write,

$$\frac{\partial \ell}{\partial z_m} = \frac{\partial \ell}{\partial b_m} \tag{4}$$

and by substituting from equation (1) and (4) in (3) we can get:

$$\frac{\partial \ell}{\partial x_n} = \frac{\partial \ell}{\partial b_m} \times w_{nm} \tag{5}$$

Therefore, we can uniquely compute the above gradient as we have access to the weight and bias gradients, and the initial parameters.
Equation (5) represents the gradient w.r.t. one node. To complete the total gradient w.r.t. the input x_n can be derived from gradients w.r.t. W and gradients w.r.t. b:

$$\frac{\partial \ell}{\partial x_n} = \sum_{c=1}^{c} \frac{\partial \ell}{\partial x_n} \tag{6}$$

Now, we need to propagate the gradient vector w.r.t. input to the convolution layer. To do that, we need to construct the input vector X_i. In their work, Aono et al. [1] showed that the input X can be derived from gradients w.r.t. W and gradients w.r.t. b. Following the equations above and to construct The input n:

$$\frac{\partial \ell}{\partial w_{nm}} = \frac{\partial \ell}{\partial z_m} \times \frac{\partial z_m}{\partial w_{nm}}, \frac{\partial z_m}{\partial w_{nm}} = x_n$$

$$\frac{\partial \ell}{\partial w_{nm}} = \frac{\partial \ell}{\partial z_m} x_n$$

$$x_n = \frac{\partial \ell}{\partial w_{nm}} / \frac{\partial \ell}{\partial z_m}$$

and by substituting from equation (4) one obtains:

$$x_n = \frac{\partial \ell}{\partial w_{nm}} / \frac{\partial \ell}{\partial b_m} \tag{7}$$

As long as there exists at least one node m with $\frac{\partial \ell}{\partial b_m} \neq 0$, the single input x_n is recovered perfectly.

So from equations (6) and (7) we can fully recover both the gradient w.r.t. the input $\frac{\partial \ell}{\partial X}$ and the input vector X analytically from the gradients w.r.t. W, gradients w.r.t. b, and initial parameters W and b.

3.2 Propagating Gradient Through Activation Function

To drive the gradient vector w.r.t. the output of the convolution layer $\frac{\partial \ell}{\partial O} = (\frac{\partial \ell}{\partial o_1}, \frac{\partial \ell}{\partial o_2}, \cdots, \frac{\partial \ell}{\partial o_i})$, we employ the chain rule to propagate the gradient vector w.r.t. to the input of the fully connected layer through the activation function.

Let A denote the activation function, where O represents the input to the activation function from the convolution layer, and X denotes the output fed into the fully connected layer $X = A(O)$. The gradient vector w.r.t. the Output O is computed as the element-wise product of the derivative of the activation function $A'(O)$ and the gradient vector w.r.t. the input of the fully connected layer $\frac{\partial \ell}{\partial X}$:

$$\frac{\partial \ell}{\partial O} = \frac{\partial \ell}{\partial X} \times \frac{\partial X}{\partial O}$$

$$\frac{\partial \ell}{\partial O} = \frac{\partial \ell}{\partial X} \times A'(O) \tag{8}$$

Considering that the derivatives of the most commonly used activation functions in neural networks can be expressed solely in terms of the function's output, we can efficiently compute this derivative using the constructed input vector of the fully connected layer. Table 1 summarizes the derivatives of various activation functions commonly employed in neural networks, expressing them in terms of the output of the activation function X.

Table 1: Derivative of activation functions.

Name	Equation	Derivative expressing in X
Sigmoid	$A(O) = \frac{1}{1+e^{-O}}$	$A'(O) = X(1-X)$
Tanh	$A(O) = \tanh(O)$	$A'(O) = 1 - X^2$
ArcTan	$A(O) = \tan^{-1}(O)$	$A'(O) = \frac{1}{1-\tan(X)^2}$
SoftPlus	$A(O) = \log_e(1 + e^O)$	$A'(O) = \frac{1}{1+\ln(e^X-1)}$
ReLU	$A(O) = \begin{cases} O & \text{if } O > 0 \\ 0 & \text{if } O \leq 0 \end{cases}$	$A'(O) = \begin{cases} 1 & \text{if } X > 0 \\ 0 & \text{if } X = 0 \end{cases}$
Leaky ReLU	$A(O) = \begin{cases} O & \text{if } O \geq 0 \\ 0.01O & \text{if } O < 0 \end{cases}$	$A'(O) = \begin{cases} 1 & \text{if } X \geq 0 \\ 0.01 & \text{if } X < 0 \end{cases}$
Parameteric ReLU	$A(O) = \begin{cases} O & \text{if } O \geq 0 \\ \alpha O & \text{if } O < 0 \end{cases}$	$A'(O) = \begin{cases} 1 & \text{if } X \geq 0 \\ \alpha & \text{if } X < 0 \end{cases}$
ELU	$A(O) = \begin{cases} O & \text{if } O \geq 0 \\ \alpha(e^O - 1) & \text{if } O < 0 \end{cases}$	$A'(O) = \begin{cases} 1 & \text{if } X \geq 0 \\ X + \alpha & \text{if } X < 0 \end{cases}$

3.3 Gradient Computation and Input Reconstruction in Convolution Layer

In a typical CNN, the convolution operation involves sliding a kernel (or filter) across the input data and computing the dot product of the kernel and the corresponding parts of the input it covers. To illustrate the equations for deriving both the input and the gradient with respect to the input, let's consider a simple example with a 3×3 input matrix and a 2×2 filter kernel. In the forward pass, the output O is expressed as follows:

$$\begin{aligned} o_{1,1} &= x_1 w_{1,1} + x_2 w_{1,2} + x_4 w_{1,4} + x_5 w_{1,5} \\ o_{1,2} &= x_2 w_{1,1} + x_3 w_{1,2} + x_5 w_{1,4} + x_6 w_{1,5} \\ o_{1,3} &= x_4 w_{1,1} + x_5 w_{1,2} + x_7 w_{1,4} + x_8 w_{1,5} \\ o_{1,4} &= x_5 w_{1,1} + x_6 w_{1,2} + x_8 w_{1,4} + x_9 w_{1,5} \end{aligned} \tag{9}$$

Gradient Computation Applying the chain rule, the gradient w.r.t. x_1 is expressed as:

$$\frac{\partial \ell}{\partial x_1} = \frac{\partial \ell}{\partial o_{1,1}} \times \frac{\partial o_{1,1}}{\partial x_1}$$

And from previous equations we can obtain $\frac{\partial o_{1,1}}{\partial x_1} = w_{1,1}$. Thus,

$$\frac{\partial \ell}{\partial x_1} = \frac{\partial \ell}{\partial o_{1,1}} \times w_{1,1}$$

Similarly, for x_2, as it contributes to both outputs $o_{1,1}$ and $o_{1,2}$ through weights $w_{1,2}$ and $w_{1,1}$ respectively:

$$\frac{\partial \ell}{\partial x_2} = \frac{\partial \ell}{\partial o_{1,1}} \times w_{1,2} + \frac{\partial \ell}{\partial o_{1,2}} \times w_{1,1}$$

In general, the gradient w.r.t each input of the convolution layer can be obtained by summing the dot products between each output contribution made by that input and its corresponding weight. If we assume that input x_n contributes to the number of outputs m with indices stored in list v, the input contributes to these outputs through the weights with indices stored in t, then the general formula can be written as follows:

$$\frac{\partial \ell}{\partial x_n} = \sum_{m=1}^{m} \frac{\partial \ell}{\partial o}[v[m]] \times w[t[m]] \tag{10}$$

Input Reconstruction Applying the chain rule the gradient w.r.t. $w_{1,1}$ is expressed as:

$$\frac{\partial \ell}{\partial w_{1,1}} = \frac{\partial \ell}{\partial o_{1,1}} \times \frac{\partial o_{1,1}}{\partial w_{1,1}} + \frac{\partial \ell}{\partial o_{1,2}} \times \frac{\partial o_{1,2}}{\partial w_{1,1}} + \frac{\partial \ell}{\partial o_{1,3}} \times \frac{\partial o_{1,3}}{\partial w_{1,1}} + \frac{\partial \ell}{\partial o_{1,4}} \times \frac{\partial o_{1,4}}{\partial w_{1,1}}$$

From equation (9) we can obtain:

$$\frac{\partial o_{1,1}}{\partial w_{1,1}} = x_1, \frac{\partial o_{1,2}}{\partial w_{1,1}} = x_2, \frac{\partial o_{1,3}}{\partial w_{1,1}} = x_4, \frac{\partial o_{1,1}}{\partial w_{1,1}} = x_5$$

Thus,

$$\frac{\partial \ell}{\partial w_{1,1}} = \frac{\partial \ell}{\partial o_{1,1}} \times x_1 + \frac{\partial \ell}{\partial o_{1,2}} \times x_2 + \frac{\partial \ell}{\partial o_{1,3}} \times x_4 + \frac{\partial \ell}{\partial o_{1,1}} \times x_5$$

Similarly, for other weights:

$$\frac{\partial \ell}{\partial w_{1,2}} = \frac{\partial \ell}{\partial o_{1,1}} \times x_2 + \frac{\partial \ell}{\partial o_{1,2}} \times x_3 + \frac{\partial \ell}{\partial o_{1,3}} \times x_5 + \frac{\partial \ell}{\partial o_{1,1}} \times x_6$$

$$\frac{\partial \ell}{\partial w_{1,3}} = \frac{\partial \ell}{\partial o_{1,1}} \times x_4 + \frac{\partial \ell}{\partial o_{1,2}} \times x_5 + \frac{\partial \ell}{\partial o_{1,3}} \times x_7 + \frac{\partial \ell}{\partial o_{1,1}} \times x_8$$

$$\frac{\partial \ell}{\partial w_{1,4}} = \frac{\partial \ell}{\partial o_{1,1}} \times x_5 + \frac{\partial \ell}{\partial o_{1,2}} \times x_6 + \frac{\partial \ell}{\partial o_{1,3}} \times x_8 + \frac{\partial \ell}{\partial o_{1,1}} \times x_9$$

Similarly, we can obtain an equal number of equations from gradients w.r.t. w_2 and w_3, which provides sufficient equations to reconstruct the input X.

In general, from each gradient w.r.t. a weight in the convolution layer, we can obtain an equation involving some of the input data points. If we assume a specific weight $w_{d,i}$ that contributes to the m data outputs through the input data points in list r, we can drive the general formula as follows:

$$\frac{\partial \ell}{\partial w_{d,i}} = \frac{\partial \ell}{\partial o_{d,1}} \times x[r[1]] + \frac{\partial \ell}{\partial o_{d,2}} \times x[r[2]] + \cdots + \frac{\partial \ell}{\partial o_{d,m}} \times x[r[m]] \quad (11)$$

With a suitable number of filters, we can establish a set of linear equations to reconstruct X.

4 Analysis of Recursive Gradient on Convolutional Layers

In this section, we provide an analytical approach to estimate the feasibility of performing an analytical recursive gradient attack on a convolutional layer. Although our methodology mainly relies on gradient constraints, to provide a full analysis, we will also consider the weight constraints and then later limit their significance depending on the activation functions that are employed. In the forward pass, the output O^l of a convolutional layer l is given by the convolution

of the input data of this layer X^l with the filter weights W^l represented as $O^l = A^l X^l$, where A^l is the matrix representing the convolution operation with weights. In the backward pass, the gradients w.r.t. the filter weights ∇W^l are given by the convolution operation between the input X^l and the gradients w.r.t. the output ∇O^l represented as $\nabla W^l = B^l X^l$, where B^l represents the convolution operations with the gradients. To solve for X^l, we combine the forward and backward pass equations into a single system:

$$\begin{bmatrix} A^l \\ B^l \end{bmatrix} X^l = \begin{bmatrix} O^l \\ \nabla W^l \end{bmatrix}$$

To successfully reconstruct X^l, the rank of the matrix $\begin{bmatrix} A^l \\ B^l \end{bmatrix}$ must be equal to the number of unknowns $|X^l|$. The number of equations in A^l represents the number of weight constraints, and the number of equations in B^l represents the number of gradient constraints.

Weight constraints: The number of weight constraints depends on the number of output data points, $|O^l|$, that can be constructed from the subsequent layer $l+1$. This number is influenced by the type of activation function applied to the output of the layer. If the activation function is fully invertible, the number of constraints will equal the number of output data points, $|A^l| = |O^l|$; if non-invertible, the number of constraints will be zero, $|A^l| = 0$; if partially invertible, the number of constraints, $|A^l|$, will be a subset of $|O^l|$, depending on how many data points can be reconstructed.

Gradient constraints: The number of gradient constraints, in case we can derive the gradients w.r.t. to the output of the layer ∇O^l, is $|B^l| = |W^l|$, equal to the number of weights in the layer. The ability to derive ∇O^l depends on the success of propagating the gradients w.r.t. the subsequent layer input ∇X^{l+1} through the activation function, as demonstrated in the methodology (Section 3.2).

It is worth noting that as the number of weight constraints depends on the output dimension, and the number of gradient constraints depends on the number of weight in the first layer, assuming the activation function is invertible, the number of weight constraints will be greater than the number of gradient constraints ($|A^l| > |B^l|$) because in the first layer, the output dimension is still large, and the number of input channels is few. However, as we move deeper into the convolutional layers, the output dimension typically decreases, and the number of channels increases, resulting in the number of gradient constraints becoming much larger than the number of weight constraints ($|A^l| < |B^l|$). The previous work by [16] has inaccurately reported that there are very few valid gradient constraints in a convolutional layer. To mitigate the risk of gradient attacks, we must ensure that the following conditions hold: $|X^l| > |A^l| + |B^l|$. Our findings indicate that the number of filters in the convolution layer is the key factor enabling the analytical attack. To quantify the risk using the number of filters, suppose the dimension for the input is $(H * H * N)$, the number of

filters is F, and the kernel size is K, with stride S and padding P:

$$|A^l| = (\frac{H + 2P - K}{S} + 1) * F$$

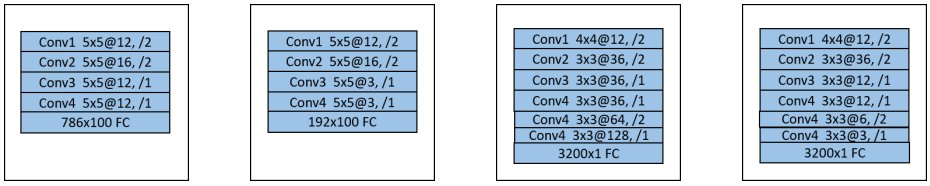
$$|B^l| = K^2 * F$$

Adjusting these parameters allows us to control the number of weight and gradient constraints, thus influencing the feasibility and risk of gradient attacks. Another important point is that the above analysis also holds for the case of using ResNet (Residual Network). In our approach, we construct all the gradients with respect to the outputs of the layers. To reconstruct the gradient with respect to the input of the residual block, we simply combine the gradients from both the residual path and the skip connection. In conclusion, rank analysis is crucial in determining the feasibility of reconstructing the input data in convolutional layers, especially when the activation function is not fully invertible.

5 Results and Discussions

Experimential Setup: To illustrate the effectiveness of our R-CONV method, we compared it with existing methods DLG [17] and R-GAP [16] using three different datasets: CIFAR-100 [6], CIFAR-10 [6], and MNIST [7]. We employed four different CNN architectures to evaluate our method, as shown in Figure 1. These include LeNet, CNN6, and their optimized versions LeNet-O and CNN6-O (having a comparatively very small number of parameters). To compute the gradient we pass an input from the model and then compute gradients during the backward pass. We used three performance metrics to evaluate our proposed method's effectiveness: mean squared error (MSE), peak signal-to-noise ratio (PSNR), and reconstruction time. Our experiments were conducted on a 12th Gen Intel(R) i7 2.10 GHz with 32 GB RAM, demonstrating that our method significantly improves reconstruction time compared to DLG and R-GAP. All implementation was carried out in the PyTorch framework.

Quantative Results: Our novel approach, R-CONV, successfully extends existing analytical gradient attacks and provides an efficient method for reconstructing inputs of convolutional layers with minimal parameters. The experimental results demonstrate that our approach not only reconstructs images more quickly than both DLG and R-GAP but also produces higher-quality images. Our method is effective across a wide range of activation functions that other approaches fail to handle. We used LeNet and LeNet-O to compare DLG and R-CONV and CNN6 and CNN6-O for comparing R-GAP and R-CONV, respectively. Table 2 summarizes the quantitative results of R-CONV in terms of average MSE, PSNR, and reconstruction time. Also, the table provides a comparative analysis of our proposed R-CONV method with existing methods, i.e., DLG, and R-GAP. The table demonstrates the effectiveness of our method in using fewer parameters for image reconstruction. Notably, when replacing the Sigmoid function with ReLU, DLG's reconstruction quality deteriorates significantly, whereas R-CONV maintains high quality. Moreover, the table demonstrates how much faster R-CONV is



(a) LeNet (b) LeNet-O (c) CNN6 (d) CNN6-O

Fig. 1: The four network architectures used for evaluation: (a) LeNet, as defined in DLG, serves as a baseline model. (b) CNN6, as defined in R-GAP, represents another standard architecture. (c) LeNet-O, a modified version of LeNet, demonstrates the effectiveness of our method in using fewer parameters for image reconstruction. (d) CNN6-O, a modified version of CNN6, reduces the number of filters in the last layer from 128 to 3, highlighting our method's ability to reconstruct images with minimal parameters. In the notation for each layer, such as "Conv1 5x5@12, /2", "5x5" specifies the kernel size, "12" represents the number of filters, and "/2" denotes the stride value.

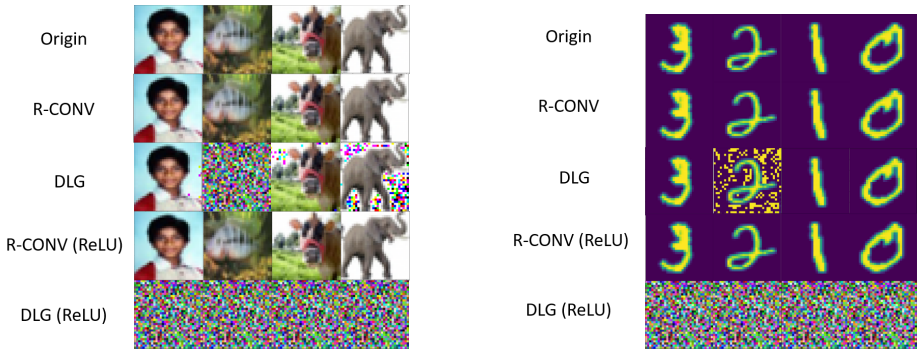
Table 2: Comparison of the proposed R-CONV method with state-of-the-art analytical (R-GAP) and optimization-based (DLG) methods in terms of average MSE, PSNR, and reconstruction time. *Our proposed method outperforms these state-of-the-art methods in all the considered metrics.*

Method	MSE	PSNR (dB)	Time (s)
Average Computed for Images Depicted in Figure 2.			
R-CONV	$2.2 \times 10^{-7} \pm 3.64 \times 10^{-9}$	114.68 ± 5.5	6.33 ± 2.3
DLG	0.0933 ± 0.05	62.62 ± 8.5	60.66 ± 5.58
Average Computed for Images Depicted in Figure 3.			
R-CONV	$2.88 \times 10^{-9} \pm 2.44 \times 10^{-10}$	150.12 ± 4.5	2.494 ± 1.66
R-GAP	0.0056 ± 0.008	76.73 ± 6.88	232.45 ± 12.44

compared to R-GAP. This is because our method leverages the full set of gradient constraints to provide a more reliable and accurate reconstruction, overcoming the limitations of previous approaches (that rely on weight constraints, which require constructing the output of each layer which is not feasible in the presence of non-invertible activation functions).

Qualitative Results: Figure 2 illustrates the reconstruction quality of images from CIFAR-100 and MNIST, showing superior quality of images reconstructed using our approach. Our method is effective across a wide range of activation functions where other approaches fail to reconstruct good-quality images. Notably, when replacing the Sigmoid function with ReLU, DLG's reconstruction quality deteriorates significantly, whereas R-CONV maintains high quality. For comparison with R-GAP, we used the CNN6 and CNN6-O architectures. We

evaluated the performance on CIFAR-10. Figure 3 shows the reconstruction results of images from CIFAR-10, comparing R-CONV and R-GAP. It shows how R-CONV outperforms R-GAP in terms of reconstruction quality and its applicability to non-fully invertible activation functions, unlike R-GAP. Moreover, the results show that our method can reconstruct images with fewer parameters, as demonstrated with LeNet-O, and CNN6-O.



(a) CIFAR100 (b) MNIST

Fig. 2: Reconstruction quality by R-CONV and DLG. Subfigure (a) shows reconstruction results using images from CIFAR-100, and (b) depicts reconstruction results using images from MNIST. The first row represents the original images, the second row shows reconstructions by R-CONV using LeNet-O, the third row shows reconstructions by DLG using LeNet, the fourth row shows reconstructions by R-CONV with ReLU activation, and the fifth row shows reconstructions by DLG with ReLU activation. *The results indicate that R-CONV produces higher-quality reconstructions than DLG, and when using ReLU activation, DLG's reconstruction quality deteriorates significantly, whereas R-CONV maintains superior quality even with fewer parameters.*

Limitations: Although our proposed R-CONV method has achieved promising results in reconstructing good-quality images, its limitations should be acknowledged. For instance, R-CONV is primarily used to reconstruct a single training sample. Using composite activation functions (e.g., GELU, SiLU, Swish) will limit R-CONV's effectiveness in exploiting gradient constraints, as their derivative can not be expressed in terms of the output.

6 Conclusions

In this paper, we focused on advancing the understanding and effectiveness of gradient inversion attacks on shared model updates in the federated learning paradigm. Recognizing the importance of understanding information leakage

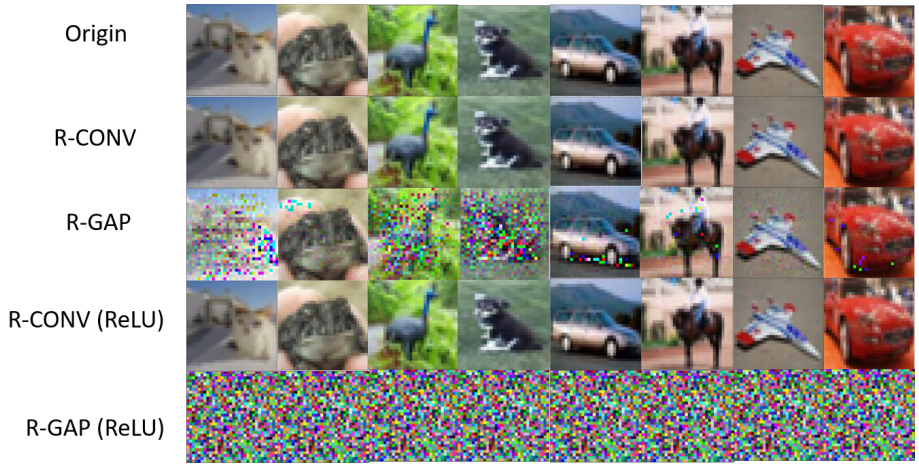


Fig. 3: Reconstruction quality by R-CONV and R-GAP. This figure presents the reconstruction results of images from CIFAR-10, comparing R-CONV and R-GAP. The first row shows the original images, the second row displays reconstructions by R-CONV using CNN6-O, the third row shows reconstructions by R-GAP using CNN6, the fourth row presents reconstructions by R-CONV with ReLU activation, and the fifth row shows reconstructions by R-GAP with ReLU activation. *The comparison underscores the superior performance of R-CONV, which maintains high-quality reconstructions even with ReLU activation, unlike R-GAP.*

mechanisms, we proposed R-CONV, an advanced analytical method that efficiently reconstructs training data from the gradients of convolutional layers. Our findings emphasize the critical role of gradient constraints, which can be more revealing than weight constraints in certain layers, an aspect previously overlooked. In addition, we conducted a detailed rank analysis and investigated the influence of various convolutional layer parameters. Our method outperforms existing techniques by achieving better data reconstruction, reducing reconstruction time, and applying it to a broader range of activation functions.

Data Availability

To encourage research in this field and support reproducibility of the results, the source code employed in this paper is released open-source on GitHub at https://github.com/tamertaras91/RR-CONV.git.

Funding

This study is partially supported by:

– the "COLTRANE-V" project – funded by the Ministero dell'Università e della Ricerca – within the PRIN 2022 program (D.D.104 - 02/02/2022).

– SERICS project (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

This manuscript reflects only the authors' views and opinions, and the funding bodies cannot be considered responsible for them.

References

1. Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE transactions on information forensics and security **13**(5), 1333–1345 (2017)
2. Chen, C., Campbell, N.D.: Understanding training-data leakage from gradients in neural networks for image classification. arXiv preprint arXiv:2111.10178 (2021)
3. Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients-how easy is it to break privacy in federated learning? Advances in neural information processing systems **33**, 16937–16947 (2020)
4. Hitaj, B., Ateniese, G., Perez-Cruz, F.: Deep models under the gan: information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. pp. 603–618 (2017)
5. 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)
6. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
7. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
8. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
9. Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE symposium on security and privacy (SP). pp. 691–706. IEEE (2019)
10. Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., Qi, H.: Beyond inferring class representatives: User-level privacy leakage from federated learning. In: IEEE INFOCOM 2019-IEEE conference on computer communications. pp. 2512–2520. IEEE (2019)
11. Wang, Z., Lee, J., Lei, Q.: Reconstructing training data from model gradient, provably. In: International Conference on Artificial Intelligence and Statistics. pp. 6595–6612. PMLR (2023)
12. Wei, W., Liu, L., Loper, M., Chow, K.H., Gursoy, M.E., Truex, S., Wu, Y.: A framework for evaluating client privacy leakages in federated learning. In: Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25. pp. 545–566. Springer (2020)
13. Yin, H., Mallya, A., Vahdat, A., Alvarez, J.M., Kautz, J., 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)
14. Zhang, H., Hong, J., Deng, Y., Mahdavi, M., Zhou, J.: Understanding deep gradient leakage via inversion influence functions. Advances in Neural Information Processing Systems **36** (2024)

15. Zhao, B., Mopuri, K.R., Bilen, H.: idlg: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610 (2020)
16. Zhu, J., Blaschko, M.: R-gap: Recursive gradient attack on privacy. arXiv preprint arXiv:2010.07733 (2020)
17. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. Advances in neural information processing systems **32** (2019)