Reconstructing Training Data from Model Gradient, Provably

Zihan Wang
New York University

Jason D. Lee
Princeton University

Qi Lei
New York University

Abstract

Understanding when and how much a model gradient leaks information about the training sample is an important question in privacy. In this paper, we present a surprising result: even without training or memorizing the data, we can fully reconstruct the training samples from a single gradient query at a randomly chosen parameter value. We prove the identifiability of the training data under mild conditions: with shallow or deep neural networks and a wide range of activation functions. We also present a statistically and computationally efficient algorithm based on tensor decomposition to reconstruct the training data. As a provable attack that reveals sensitive training data, our findings suggest potential severe threats to privacy, especially in federated learning.

1 INTRODUCTION

It is essential to understand when and how much a model gradient leaks information about the training data. Such a problem poses a major concern especially in federated learning. Federated learning, where each device can only access its own local training data, has become increasingly popular for training private data in recent years (Brisimi et al., 2018; McMahan et al., 2017). The defining trait of federated learning is aggregating information without revealing sensitive information in the underlying data. Therefore serious privacy concerns arise when one can recover private information from the training procedure.

Recent years have seen increasingly many studies on reconstructing sensitive data from a model or its gradient updates (Zhu et al., 2019; Zhao et al., 2020; Geiping et al., 2020; Wang et al., 2020; Wei et al., 2020; Jin et al., 2021; Geiping et al., 2020; Yin et al., 2021). This line of work brings about some **empirical** evidence that the current frameworks in federated learning can be vulnerable to privacy attacks. However, theoretical understanding of this vulnerability remains limited. To fill this gap, we raise the following questions:

Question 1: *Is data leakage due to memorization in the training process? In other words, do we need to train the model sufficiently to recover the training samples?*

Many studies have concluded that data reconstruction should be conducted after training the neural network, as the training process leaks more information and helps the model memorize the training data. In this paper, we show the opposite: for a broad class of neural networks architectures, one can reconstruct the samples from a prescribed (random) model's gradient alone *without any training*.

Question 2: *When is the model gradient alone sufficient to identify the training samples (without prior knowledge)?*

Prior work conjectured that a gradient is insufficient to reconstruct the underlying data without prior knowledge (Jeon et al., 2021). This conjecture is motivated by some empirical failures in privacy attacks but lacks a sound theoretical foundation.

Failures in privacy attacks could happen for two different reasons. First, it is possible that the existing optimization algorithms cannot solve the nonconvex reconstruction loss in polynomial time. Second, statistically the gradient alone is insufficient to identify the training samples. Most existing studies have assumed the second case and asserted the need for prior knowledge in privacy attacks (Geiping et al., 2020; Yin et al., 2021; Jeon et al., 2021). We show in this paper that one can recover the training samples from the gradient alone as long as the model is moderately wide.

Our paper provides theoretical insights into reconstructing training samples from model gradient (i.e., gradient inversion). Specifically, we have the following contributions:

- We show that one gradient query is sufficient to identify the training sample for a broad class of models. Our design applies to fully-connected neural networks with two or more layers and works with most common activation functions, including (Leaky) ReLU, tanh, and sigmoid functions.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

- We introduce a statistically and computationally efficient algorithm based on power iteration to reconstruct the training samples. Under some natural assumptions, we show that one can accurately reconstruct both the input and labels when the neural network is $\widetilde{\Omega}(d)$-dimensional wide[1] .

2 RELATED WORK

Federated Learning. McMahan et al. (2017) proposed the notion of federated learning that the training data is distributed among clients and the shared model is trained by aggregating the updates computed locally. In this way, the central server cannot directly access the data from clients so is considered safer. Recent works of federated learning improved optimization algorithms (Konečný et al., 2015, 2016a) and communication methods between clients and central server (Konečný et al., 2016b) and tackled related statistical challenges (Smith et al., 2017; Zhao et al., 2018).

Gradient Inversion. Previous works on gradient inversion studied different methods of reconstructing private training data from shared gradients, where a series of optimization-based methods is known as gradient matching. Zhu et al. (2019) trained minimized the difference between the dummy gradient and true gradient and Zhao et al. (2020) additionally recovered ground truth labels by analyzing the signs of gradients. With a well designed loss function (Geiping et al., 2020; Wang et al., 2020), initialization (Wei et al., 2020; Jin et al., 2021), and image prior regularization (Geiping et al., 2020; Yin et al., 2021), gradient matching can succeed with deeper neural networks, larger batch sizes and higher resolution images.

Furthermore, various of works improved gradient matching method from different perspectives. Huang et al. (2021) relaxed the strong assumption that batch normalization statistics are known. Balunovic et al. (2022) included these optimization-based gradient inversion attacks into an approximation Bayesian adversarial framework. Jeon et al. (2021) optimized the difference of gradient on latent space by training a generative model as image prior. Hatamizadeh et al. (2022) generalized gradient matching method to vision transformers. Most of the empirical results in gradient matching indicated that gradient matching alone is insufficient to reconstruct private data, contrary to our theoretical results.

A different framework of gradient inversion is based on an observation by Phong et al. (2018) under single neuron setting that $x_k = \nabla_{W_k}/\nabla_b$, where b is the bias, x_k and W_k is the k-th coordinate of the input and the weight respectively. Fan et al. (2020) generalized the property to fully connected neural networks and reconstructed private data by solving a noisy linear system. By sequentially constructing the relationship of input and gradient from the output layer to the first layer, Zhu and Blaschko (2021); Chen and Campbell (2021) proposed methods recovering data from convolutional layers. This series of attacks recover private data with explicit forms but the result may not be unique and can only deal with single input cases.

Tensor Decomposition. The notion of tensor decomposition is proposed by Hitchcock (1927, 1928) and one instance is known as CP decomposition (Harshman, 1970; Carroll and Chang, 1970; Kiers, 2000), where a tensor is decomposed into a sum of component rank-1 tensors. For a tensor with higher order, its CP decomposition is often unique (Kruskal, 1977). However, solving CP decomposition problem of a tensor is generally NP-hard (Håstad, 1989; Hillar and Lim, 2013) so some restrictions are necessary. A method of solving CP decomposition problem is known as tensor power iteration, which is generalized from matrix power iteration. Some works analyzed this method under orthogonal assumptions (Anandkumar et al., 2014a; Wang et al., 2015; Song et al., 2016) or over-complete settings (McMahan et al., 2017).

3 PRELIMINARIES

In this section, we formally introduce the problem of reconstructing training data from model gradient, also referred to as the gradient inversion problem (Geiping et al., 2020; Yin et al., 2021; Jeon et al., 2021).

Notation: We denote lower case symbol x as scalar, bold lower case letter $\boldsymbol{x}$ as vector, capital letter X as matrices, and bold capital letter $\boldsymbol{T}$ as higher-order tensors. When there is no ambiguity, we also use capital letters (like Z) for random variables.

We use σ or σ_k to denote the activation function $\sigma : \mathbb{R} \to \mathbb{R}$. When there is no ambiguity, we also overload same notation for $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ where the activation is applied coordinate-wise. When σ is ReLU it maps x to $(x)_+$ which is x if $x \geq 0$ and 0 otherwise. LeakyReLU: $x \to (x)_+ - 0.01(-x)_+$. Tanh activation: $x \to (e^{2x}-1)/(e^{2x}+1)$, and sigmoid: $x \to 1/(1+e^{-x})$.

For integer n, $[n] := \{1, 2, \cdots n\}$. For vectors we use $\|\cdot\|$ or $\|\cdot\|_2$ to denote its ℓ_2 norm. For matrices they stand for the spectral norm. We use $\otimes$ to denote tensor product. For clean presentation, in the main paper we use $\widetilde{O}$, $\widetilde{\Theta}$ or $\widetilde{\Omega}$ to hide universal constants, polynomial factors in batch size B and polylog factors in dimension d, error ε, total round number T or failure rate δ.

Setup: Consider the supervised learning problem, where we train a neural network $f(\cdot; \Theta) : \boldsymbol{x} \in \mathbb{R}^d \to f(\boldsymbol{x}; \Theta) \in \mathbb{R}$

[1] Some polylog factors are hidden.

through the loss function:

$$\min_{\Theta} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \ell(f(\boldsymbol{x};\Theta),y).$$

Here ℓ is the loss function (we focus on ℓ_2 loss throughout the paper), and $\mathcal{D}$ is the dataset of input $\boldsymbol{x}\in\mathbb{R}^d$ and label $y\in\mathbb{R}$.

We consider the setting of federated learning where each client keeps the privacy of their local data. In each iteration between the central server and a client (user machine), each node reports the average of gradient of $\ell(f(\boldsymbol{x};\Theta),y)$ at an unknown batch of their own data $S=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_B,y_B)\}$, i.e,

$$G := \frac{1}{B}\nabla_\Theta \sum_{i=1}^{B} \ell(f(\boldsymbol{x}_i,\Theta),y_i).$$

Since the batch of samples to be used is determined by the client, the central server cannot ask to query the gradient at the exact same batch of data for the second time. Therefore, our gradient inversion task is as follows: we want to recover the unknown training data $\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_B$ with batch size B[2] from the gradient **queried once** at a model Θ, where the model Θ and the loss function ℓ is known. Since it is known that the labels can be easily recovered by observing the gradient at the last layer (Yin et al., 2021; Zhao et al., 2020), most prior work focused on reconstructing the input samples $\boldsymbol{x}_i, i\in[B]$.

With the discussed problem, it is straightforward to design the following objective. Indeed almost all the prior work directly solve the following task or its variants:

$$\min_{\{\hat{\boldsymbol{x}}_i,\hat{y}_i\}_{i=1}^B} d\left(\frac{1}{B}\sum_{i=1}^{B}\nabla_\Theta \ell(f(\hat{\boldsymbol{x}}_i;\Theta),\hat{y}_i),G\right), \qquad (3.1)$$

where $d(\cdot,\cdot)$ is a distance metric of the discrepency between the queried gradient and the estimated gradient (when $\hat{\boldsymbol{x}}_i, y_i$ are now treated as variables to learn). Common choices are ℓ_2 distance (Zhu et al., 2019; Yin et al., 2021) or negative cosine similarity (Geiping et al., 2020).

However, such reconstruction loss is nonconvex and over-determined, but consistent nonlinear system. Specifically, since the optimal value of (3.1) is 0 by design, the above gradient inversion problem is equivalent to solving M equations where M is the dimension of Θ (or G), or namely $B(d+1)$ parameters (the number of unknowns in the training set S). Namely, to find the global minimum of (3.1) is equivalent to solving

$$\frac{1}{B}\sum_{i=1}^{B}\nabla_\Theta \ell(f(\hat{\boldsymbol{x}}_i;\Theta),\hat{y}_i)=G. \qquad (3.2)$$

In order to identify all the training samples, we at least need $B(d+1)\le M$. In other words, the problem should be over-determined and under-parameterized, which makes the optimization problem even more challenging. (In fact, it is more theoretically sound to solve over-parameterized optimization problems in nonconvex setting (Jacot et al., 2018; Du et al., 2019; Allen-Zhu et al., 2019).

Meanwhile, even with a single sample $(B=1)$, the gradient inversion problem in Eq. (3.1) is NP-complete. (This directly follows from Theorem 1 in (Lei et al., 2019).) Therefore we do not hope to solve (3.1) in general without proper constraints on Θ or its gradient. Instead, we need to carefully choose the Θ that we query, and design better algorithms to reconstruct the training samples.

In the next section, instead of directly optimizing over Eq. (3.1), we propose some novel algorithms to recover the input data $\boldsymbol{x}_i$ and the label y_i by querying at a randomly designed neural network.

4 METHOD

In this section, we introduce the methodology for reconstructing the training samples with fully connected neural networks. As a warm-up, we first focus on two-layer neural networks. We next show that the more general settings for three-layer or deeper neural networks can be reduced to the two-layer cases with similar techniques.

4.1 Two-layer Neural Networks

We first study the case when the model is a 2-layer neural network, denoted by $f(\boldsymbol{x};\Theta)=\sum_{j=1}^m a_j\sigma(\boldsymbol{w}_j\cdot\boldsymbol{x})$. Here $\Theta=(a_1,\cdots a_m,\boldsymbol{w}_1,\cdots \boldsymbol{w}_m)$ is a collective way to write the parameters. The input dimension is d, and the width of the neural network, or namely the number of hidden nodes is m. The objective function in supervised learning is [3]

$$L(\Theta)=\sum_{i=1}^{B}\left(y_i - f(\boldsymbol{x}_i;\Theta)\right)^2.$$

Notice the gradient with respect to $\boldsymbol{a}$ is an m-dimensional vector. An interesting phenomenon is that $\nabla_{a_j} f(\boldsymbol{x};\Theta)=\sigma(\boldsymbol{w}_j^\top\boldsymbol{x}), j\in[m]$ only depends on $\boldsymbol{w}_j$, and not the other $\boldsymbol{w}_k, k\neq j$. Accordingly $\nabla_{a_j}L=\sum_{i=1}^B r_i\nabla_{a_j} f(\boldsymbol{x}_i;\Theta)$ only depends on the other $\boldsymbol{w}_k, k\neq j$ through each residue $r_i := f(\boldsymbol{x}_i;\Theta)-y_i$. Specifically, we denote the $\nabla_{a_j}L(\Theta)$ as:

$$g_j := \nabla_{a_j}L(\Theta)=\sum_{i=1}^{B} r_i\sigma\left(\boldsymbol{w}_j^\top\boldsymbol{x}_i\right), \qquad (4.1)$$

One important observation is that when the residue r_i is viewed as a constant, g_j becomes a function of $\boldsymbol{w}_j$.

[2] The batch size is usually small and can be considered sublinear in d.

[3] Here we exemplify our results using squared loss. It is straightforward to expand our analysis and methodology to other loss functions.

We set $a_j = \frac{1}{m}, \forall j \in [m]$, and sample $\boldsymbol{w}_j$ independently from standard normal distribution $\mathcal{N}(0, I)$. With a wide neural network, $r_i = \frac{1}{m}\sum_{k=1}^{m} \sigma(\boldsymbol{w}_k^\top \boldsymbol{x}_i) - y_i$ concentrates to a scalar $r_i^* := \mathbb{E}_W[r_i] = \mathbb{E}_{Z\sim\mathcal{N}(0,\|\boldsymbol{x}_i\|^2)}\,\sigma(Z) - y_i$. When σ is odd function like tanh or sigmoid, $r_i^* = -y_i$.

We defer the formal statements and proof to the next section and the appendix, but with proper concentration, we will have that $g_j = \sum_{i=1}^{B} r_i^* \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i) + \tilde{O}(\sqrt{\frac{1}{m}})$. Now we define the function $g(\boldsymbol{w}) := \sum_{i=1}^{B} r_i^* \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)$ be a function on $\boldsymbol{w}$. We have $g_j = g(\boldsymbol{w}_j) + \tilde{O}(\sqrt{\frac{1}{m}})$. This is to say by setting different $\boldsymbol{w}$ we are able to observe a noisy version of $g(\boldsymbol{w})$, where we have:

$$\nabla g(\boldsymbol{w}) = \sum_{i=1}^{B} r_i^* \sigma'(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i, \tag{4.2}$$

$$\nabla^2 g(\boldsymbol{w}) = \sum_{i=1}^{B} r_i^* \sigma''(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i\boldsymbol{x}_i^\top, \tag{4.3}$$

$$\nabla^3 g(\boldsymbol{w}) = \sum_{i=1}^{B} r_i^* \sigma^{(3)}(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i^{\otimes 3}. \tag{4.4}$$

Here $\sigma^{(3)}$ is the third derivative of σ. This observation suggests that if we are able to estimate $\mathbb{E}_W\, \nabla^p g(W), p = 1, 2, 3$ [4], we are able to recover the reweighted sum for $\boldsymbol{x}_i^{\otimes p}$. Especially when $p = 3$, the third order tensor $\mathbb{E}_W\, \nabla^3 g(W)$ has a unique tensor decomposition which will identify $\{\boldsymbol{x}_i\}_{i=1}^B$ when they are independent (Kruskal, 1977; Bhaskara et al., 2014).

A natural method to estimate higher-order derivatives is the celebrated Stein's lemma (Stein, 1981; Mamis, 2022):

Lemma 4.1 (Stein's Lemma)**.** *Let X be a standard normal random variable. Then for any function g, we have*

$$\mathbb{E}[g(X)H_p(X)] = \mathbb{E}[g^{(p)}(X)], \tag{4.5}$$

if both sides of the equation exists. Here H_p is the pth Hermite function and $g^{(p)}$ is the pth derivative of g.

Similarly, when X is vector-valued random variable and g only depends on X through some $\boldsymbol{a}^\top X$ ($\boldsymbol{a}^\top \sim \mathcal{N}(0, \|\boldsymbol{a}\|^2)$), the p-th order polynomials in H_p should be replaced by the p-th order (symmetrized) tensor products. Specifically, we will be using $H_3(\boldsymbol{x}) = \boldsymbol{x}^{\otimes 3} - \boldsymbol{x}\tilde{\otimes} I$, where $\boldsymbol{x}\tilde{\otimes}I(i, j, k) = x_i\delta_{jk} + x_j\delta_{ki} + x_k\delta_{ij}$, where δ_{ij} is 1 when $i = j$ and 0 otherwise.

Using Stein's lemma and concentration bounds to (4.1) when $a_j = \frac{1}{m}$ and $w \sim \mathcal{N}(0, 1)$, we have informally that:

$$\frac{1}{m}\sum_{j=1}^{m} g(\boldsymbol{w}_j)H_p(\boldsymbol{w}_j) \approx \mathbb{E}_{W\sim\mathcal{N}(0,I)}[g(W)H_p(W)] \tag{Concentration}$$

$$= \mathbb{E}_W[\nabla_W^p g(W)] = \sum_{i=1}^{B} \mathbb{E}[\sigma^{(p)}(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i^{\otimes p}],$$

(Stein's lemma, and plugging in the gradient of g)

Then we can use tensor decomposition to recover $\boldsymbol{x}_i$ up to some scaling factors. We present formally our reconstruction procedure in Algorithm 1.

Algorithm 1 Two-layer NN: Gradient inversion with tensor decomposition

```
Setup: With unknown batch of samples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_B\}$, we have the black box oracle that queries the gradient at transmitted model $\Theta = (\{a_i\}_{i=1}^m, \{\boldsymbol{w}_j\}_{j=1}^m)$: $G(\Theta) = (\nabla_{a_i}L(\Theta), \nabla_{\boldsymbol{w}_j}L(\Theta))$.
Initialization: Set current model
    $\Theta : a_i = \frac{1}{m}, i \in [m]$, and $\boldsymbol{w}_j \sim \mathcal{N}(0, I_d)$,
    is sampled from standard normal distribution. Query the gradient $G = (\{g_i\}_{i=1}^m)$ where $g_i = \nabla_{a_i}L(\Theta)$.
Noisy tensor decomposition: Set the 3rd order tensor
    $\hat{\boldsymbol{T}} := \sum_{i=1}^{m} g_i H_3(\boldsymbol{w}_i)$,
    where $H_3(\boldsymbol{x}) := \boldsymbol{x}^{\otimes 3} - \boldsymbol{x}\tilde{\otimes}I$. Conduct top-$B$ tensor decomposition of $\hat{T}$ (e.g. Algorithm 1 in (Kuleshov et al., 2015)) and get the vectors $\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \cdots \hat{\boldsymbol{x}}_B$ and recover the weights $\hat{\lambda}_i$. Let $\hat{y}_i = \hat{\lambda}_i - \mathbb{E}_{X\sim\mathcal{N}(0,\|\hat{\boldsymbol{x}}_i\|^2)}\sigma(X)$.
Output: $\{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \cdots \hat{\boldsymbol{x}}_B\}, \{\hat{y}_1, \hat{y}_2, \cdots \hat{y}_B\}$.
```

4.2 Deep Neural Networks

Next we move to general deep neural networks. Suppose we are working on an l-layered deep neural networks, where the function

$$f(\boldsymbol{x}, \Theta) = \boldsymbol{a}^\top \sigma_l(W_l\sigma_{l-1}(W_{l-1}\cdots\sigma_2(W_2\sigma_1(W_1\boldsymbol{x})\cdots).$$

Here $\boldsymbol{a} \in \mathbb{R}^m, W_1 \in \mathbb{R}^{m\times d}, W_j \in \mathbb{R}^{m\times m}$ for any $2 \le j \le l$.

When the intermediate layers are wide enough ($m \ge 2d$), we are able to design the first $l - 1$ layers to keep the information of each input vector $\boldsymbol{x}_i, i \in [B]$. Specifically, many activation functions like tanh or sigmoid are bijective. With this type of activations, we will set the weight matrices $W_k, k \in [l - 1]$ to identity matrix (concatenated with 0 matrices if the dimension is off) to ensure we don't lose the information of the input. Next, the input of the last layer becomes the output of the bijective function $h_{l-1}(\cdot) := \sigma_{l-1} \circ \cdots \sigma_2 \circ \sigma_1(\cdot)$. Therefore, we can view the problem as a same setting of the two-layer case where the input is $h_{l-1}(\boldsymbol{x}_i), i \in [B]$.

[4] It will become necessary to further explore higher order $p = 4$ if $\sigma^{(3)}$ is an odd function. This will be explained in Section 5.

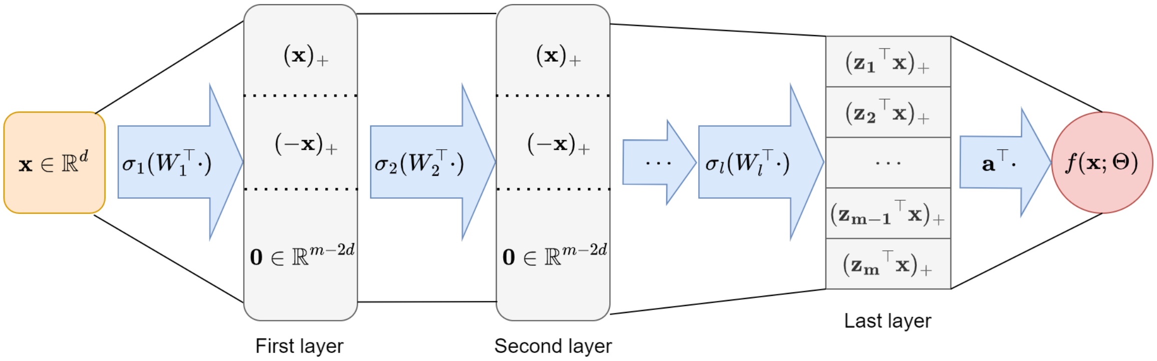
$\mathbf{x} \in \mathbb{R}^d$
$\sigma_1(W_1^\top \cdot)$
$(\mathbf{x})_+$
$(-\mathbf{x})_+$
$\mathbf{0} \in \mathbb{R}^{m-2d}$
$\sigma_2(W_2^\top \cdot)$
$(\mathbf{x})_+$
$(-\mathbf{x})_+$
$\mathbf{0} \in \mathbb{R}^{m-2d}$
$\cdots$
$\sigma_l(W_l^\top \cdot)$
$(\mathbf{z_1}^\top \mathbf{x})_+$
$(\mathbf{z_2}^\top \mathbf{x})_+$
$\cdots$
$(\mathbf{z_{m-1}}^\top \mathbf{x})_+$
$(\mathbf{z_m}^\top \mathbf{x})_+$
$\mathbf{a}^\top \cdot$
$f(\mathbf{x};\Theta)$
First layer
Second layer
Last layer

Figure 1: **Model architecture**: example with ReLU activations to illustrate how multi-layer neural networks can be reduced to the two-layer setting.

For piece-wise linear activations like ReLU:$\boldsymbol{x} \to (\boldsymbol{x})_+$, we will also be able to keep all the information of the input $\boldsymbol{x}$ when the number of intermediate hidden nodes $m \geq 2d$. Specifically, we can set $W_1 = [I_d, -I_d, 0]^\top$ where $I_d \in \mathbb{R}^{d\times d}$ will ensure that $\sigma(W_1\boldsymbol{x}) = [(\boldsymbol{x})_+^\top, (-\boldsymbol{x})_+^\top, 0]^\top$. For the remaining layers, since the input is now non-negative, $\sigma(W_j\cdot), 2 \leq j \leq l-1$ simply functions as the identity map. Now towards the last but one layer, note that $(\boldsymbol{x})_+ - (-\boldsymbol{x})_+ \equiv \boldsymbol{x}$ for any vector $\boldsymbol{x}$, and we set $\boldsymbol{w}_j = [\boldsymbol{z}_j^\top, -\boldsymbol{z}_j^\top, 0]^\top$. Therefore $\boldsymbol{w}_j^\top h_{l-1}(\boldsymbol{x}_i) = \boldsymbol{z}_j^\top(\boldsymbol{x}_i)_+ + (-\boldsymbol{z}_j)^\top(-\boldsymbol{x}_i)_+ = \boldsymbol{z}_j^\top \boldsymbol{x}_i$. We can therefore migrate the exact same input vectors to the last but one layer as the same algorithm for the two-layer case. For clear illustration, we present the model architecture for ReLU in Figure 1.

We present more formally the algorithm design in Algorithm 2. For cleaner presentation, we will assume σ_1 through σ_{l-1} are ReLU ($\sigma : \boldsymbol{x} \to (\boldsymbol{x})_+$) or LeakyReLU ($\sigma : \boldsymbol{x} \to (\boldsymbol{x})_+ - 0.01(-\boldsymbol{x})_+$). We discuss the case for tanh or sigmoid below.

For sigmoid or tanh we can set all the intermediate weight matrices to be I_d concatenated with 0. The only tricky part is that we no longer know the norm of the last but one layer $h_{l-1}(\boldsymbol{x})$. Now suppose we conduct tensor decomposition and get the vectors $\hat{\boldsymbol{a}}_1, \hat{\boldsymbol{a}}_2, \cdots \hat{\boldsymbol{a}}_B$. We can do a binary search over the correct scaling: find α_i such that $\sigma_1^{-1}(\cdots \sigma_{l-2}^{-1}(\sigma_{l-1}^{-1}(\alpha_i \hat{\boldsymbol{a}}_i)\cdots)$ is of norm 1. Suppose we have not done normalization and do not know the norm of the input vectors $\boldsymbol{x}_i, i \in [B]$, we can first estimate the correct norm of the last but one layer $h_{l-1}(\boldsymbol{x}_i)$. Let's denote $\beta_i := \|h_{l-1}(\boldsymbol{x}_i)\|$. This can be achieved by estimating the weights of the tensor slices for $\boldsymbol{T} = \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0,I)}\,\sigma_l^{(3)}(\boldsymbol{w}^\top h_{l-1}(\boldsymbol{x}_i))h_{l-1}(\boldsymbol{x}_i)^{\otimes 3}$. Therefore the weights $\lambda_i = \mathbb{E}_{z\sim\mathcal{N}(0,\beta_i^2)}\,\sigma_l^{(3)}(z)\beta_i^3$. Since it is only a scalar, we can quickly infer the value of β_i from λ_i.

5 THEORETICAL ANALYSIS

In this section, we will present our main results (informal) with a proof sketch. We defer the more complete proof and detailed dependence (on log factors of dimension d, hidden nodes m, batch size B, failure probability δ) to Appendix A.

Assumption 5.1. *We make the following assumptions:*

- ***Data:*** *Let data matrix* $X := [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_B] \in \mathbb{R}^{d\times B}$, *we denote the* B*-th singular value by* $\pi_{\min} > 0$. *Training samples are normalized:* $\|\boldsymbol{x}_i\| = 1, \forall i \in [B]$.
- ***Activation:*** σ *is 1-Lipschitz and* $\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\sigma''(z)] < \infty$. *Let*

  $$\nu := \max\{|\,\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\sigma''(z)]|, |\,\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\sigma^{(3)}(z)]|\},$$

  and

  $$\lambda := \max\{|\,\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\sigma^{(3)}(z)]|, |\,\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\sigma^{(4)}(z)]|\}.$$

  We assume $\nu, \lambda \neq 0$.

We note that the data being normalized to 1 is only a technical assumption to simplify the results. When $\boldsymbol{x}_i$ has different norms, the reconstructed accuracy will depend on $\nu_{\min} := \min_i r_i^* \,\mathbb{E}_{Z\sim\mathcal{N}(0,\|\boldsymbol{x}_i\|^2)}[\sigma''(Z)]$ and $\lambda_{\min} := \min_i r_i^* \,\mathbb{E}_{Z\sim\mathcal{N}(0,\|\boldsymbol{x}_i\|^2)}[\sigma^{(3)}(Z)]$. σ being 1-Lipschitz and having bounded expected second order derivatives is satisfied for the activations we discussed in the paper: (Leaky)ReLU, tanh or sigmoid functions.

The non-degenerate activation, however, is a crucial condition for our algorithm to succeed. For linear or quadratic functions, the third derivative $\sigma^{(3)}$ and the fourth derivative $\sigma^{(4)}$ are 0. Indeed, one can verify that it is in general not possible to recover the individual samples when the activation is linear or quadratic. For instance, when σ is linear, we have $f(\boldsymbol{x};\Theta) = \boldsymbol{a}^\top W\boldsymbol{x}$, and the gradient with respect

Algorithm 2 Deep neural networks: Gradient inversion with tensor decomposition

```
Setup: With unknown batch of samples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_B\}$, we have the black box oracle that queries the gradient at transmitted model
    $$\Theta = (\{a_i\}_{i=1}^m, \{W^{(k)}\}_{k=1}^l),$$
    where the slices of $W$ is denoted as
    $$W^{(k)} = [\boldsymbol{w}_1^{(k)}, \boldsymbol{w}_2^{(k)}, \cdots \boldsymbol{w}_m^{(k)}].$$
Initialization: Set the model as
    $$\Theta : a_i = \frac{1}{m}, i \in [m], \boldsymbol{w}_j^{(l)} = [\boldsymbol{z}_j^\top, -\boldsymbol{z}_j^\top, 0]^\top, \boldsymbol{z}_j \in \mathbb{R}^{2k},$$
    where $\boldsymbol{z}_j$ is sampled from standard normal distribution. Set $W^{(1)} = [\bar{W}^{(1)}, 0] \in \mathbb{R}^{m \times m}$, where $\bar{W}^{(1)} = [I, -I, 0]^\top \in \mathbb{R}^{m \times d}$.
for k=2 to l-1 do
    Set $W^{(k)} = [\bar{W}^{(k)}, 0] \in \mathbb{R}^{m \times m}$, where $\bar{W}^{(k)} = [I, 0]^\top \in \mathbb{R}^{m \times 2d}$.
end for
Query the gradient $G = (\{\bar{g}_i\}_{i=1}^m)$ where $g_i = \nabla_{a_i} L(\Theta)$. Truncate $g_i$ as $[\bar{g}_i, 0], g_i \in \mathbb{R}^D$.
Noisy tensor decomposition: Set the 3rd order tensor $\hat{\boldsymbol{T}} := \sum_{i=1}^m \bar{g}_i H_3(\boldsymbol{z}_i)$, where $H_3(\boldsymbol{x}) := \boldsymbol{x}^{\otimes 3} - \boldsymbol{x} \tilde{\otimes} I$ is the 3rd Hermite polynomial. Conduct top-$B$ tensor decomposition of $\hat{T}$ (e.g. with Algorithm 1 in (Kuleshov et al., 2015)) and get the vectors $\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \cdots \hat{\boldsymbol{x}}_B$ and their corresponding weights $\hat{\lambda}_i$. Let $\hat{y}_i = \hat{\lambda}_i / \lambda$.
if $\sigma_1$ is LeakyReLU then
    $\hat{\boldsymbol{x}}_i \leftarrow \hat{\boldsymbol{x}}_i / 1.01$
end if
Output: $\{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \cdots \hat{\boldsymbol{x}}_B\}, \{\hat{y}_1, \hat{y}_2, \cdots \hat{y}_B\}$.
```

to $\boldsymbol{a}$ and W are respectively:

$$G(\boldsymbol{a}) = W(\sum_{i=1}^B r_i \boldsymbol{x}_i); G(W) = \boldsymbol{a}(\sum_{i=1}^B r_i \boldsymbol{x}_i)^\top.$$

Therefore it is only possible to recover a linear combination of all the training samples $\sum_i r_i \boldsymbol{x}_i$. Similarly with quadratic activation, we derive the gradient here:

$$\nabla_{a_j} L = \boldsymbol{w}_j^\top \bar{\Sigma} \boldsymbol{w}_j; \nabla_{\boldsymbol{w}_j} L = 2\bar{\Sigma} \boldsymbol{w}_j,$$

where $\bar{\Sigma} := \sum_{i=1}^B r_i \boldsymbol{x}_i \boldsymbol{x}_i^\top, r_i = f(\boldsymbol{x}_i; \Theta) - y_i$.

Therefore one can only recover $\bar{\Sigma}$, a reweighted (weights depending on the residue r_i) covariance matrix of all the training samples, or namely the span of the training samples, instead of individual sample $\boldsymbol{x}_i$.

Theorem 5.1 (Main theorem)**.** *Suppose that $y_i \in \{\pm 1\}$. Under Assumption 5.1, if we have $B \leq \tilde{O}(d^{1/4})$ and $m \geq \tilde{\Omega}(\frac{d}{\min\{\nu^2, \lambda^2\} \pi_{\min}^4})$, then with appropriate tensor decomposition methods, the output of Algorithm 1 satisfies:*

$$\sqrt{\frac{1}{B} \sum_{i=1}^B \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2} \leq \frac{1}{\min\{|\nu|, |\lambda|\} \pi_{\min}^2} \tilde{O}(\sqrt{\frac{d}{m}}) \tag{5.1}$$

$$sign(\hat{y}_i) = y_i. \tag{5.2}$$

We note that for the uniqueness of tensor decomposition, we only need the samples $\{\boldsymbol{x}_i\}$ to be linearly independent (i.e., $\pi_{\min} > 0$) (Kruskal, 1977). [5] This is implied by Assumption 5.1, and suffices the identifiability for the training sets from gradient G.

However, in general tensor decomposition is known to be NP-hard (Håstad, 1989; Hillar and Lim, 2013), therefore we need some more technical assumptions to constrain the setting in order to derive efficient algorithm. Therefore we adapt the setting of prior work on decomposing tensor by simultaneous matrix diagonalization (Kuleshov et al., 2015; Zhong et al., 2017), and assume that the minimal singular value of data is non-zero in the main theorem.

The analysis shows that the neural network is actually very vulnerable to privacy attacks. We are able to recover the images up to an average error ϵ with mildly overparameterized network when the hidden nodes satisfy $m \gg d/(\min\{\nu^2, \lambda^2\} \pi_{\min}^4 \epsilon^2)$.

Remark 1. *In the main theorem, for cleaner presentation we considered classification task $y_i \in \{\pm 1\}$. This is not an essential assumption. For regression problems, we can also work on any real-valued and bounded y_i. However, the accuracy of tensor power method will depend on*

$$\kappa = \left|\frac{\lambda_{\max}}{\lambda_{\min}}\right| = \left|\frac{\nu_{\max}}{\nu_{\min}}\right| = \frac{\max_{i \in [B]} |r_i^*|}{\min_{i \in [B]} |r_i^*|}$$

If there exists r_i^* too small, we have some tricks to make sure this κ is constant. Specifically, suppose $|r_i^*| \leq M$, we can add a large bias term $2M$ in the last layer so that the weights in $\boldsymbol{T}$ is in the range of $[\lambda M, 3\lambda M]$ and the weights in P is in the range of $[\nu M, 3\nu M]$, and we can thus ensure $\kappa \leq 3$. In that case, we can instead guarantee

$$\sqrt{\frac{1}{B} \sum_{i=1}^B |\hat{y}_i - y_i|^2} \leq \frac{1}{\min\{|\nu|, |\lambda|\} \pi_{\min}^2} \tilde{O}(\sqrt{\frac{d}{m}}).$$

For deep neural networks, we have the following corollary:

Corollary 5.2. *Suppose Assumption 5.1 is satisfied (for σ_l). Under the same setting for Theorem 5.1, we have that w.h.p.*

[5] Bhaskara et al. (2014) further proved a robust version for the identifiability.

the output for Algorithm 2 satisfies:

$$\sqrt{\frac{1}{B}\sum_{i=1}^{B}\|\boldsymbol{x}_i-\hat{\boldsymbol{x}}_i\|^2} \leq \frac{1}{\min\{|\nu|,|\lambda|\}\pi_{\min}^2}\tilde{O}(\sqrt{\frac{d}{m}}), \tag{5.3}$$

$$sign(\hat{y}_i) = y_i. \tag{5.4}$$

5.1 Proof Sketch

As we demonstrated in the methodology section, the main technique is to estimate $\boldsymbol{T}$, where $\boldsymbol{T} := \mathbb{E}[\sum_{i=1}^{B} r_i^*\sigma^{(3)}(\boldsymbol{w}^\top\boldsymbol{x}_i)\boldsymbol{x}_i^{\otimes 3}]$ and conduct eigendecomposition. We denote $P := \mathbb{E}[g(\boldsymbol{w})H_2(\boldsymbol{w})]$ and $\hat{P} := \frac{1}{m}\sum_{j=1}^{m} g_j(\boldsymbol{w}_j)H_2(\boldsymbol{w}_j)$, where $H_2(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{x}^\top - I$. Then the proof of Theorem 5.1 mainly consists of the concentration bounds of $\hat{P}$ to P in Proposition 5.5 and $\hat{\boldsymbol{T}}$ to $\boldsymbol{T}$ in the Proposition 5.6, together with perturbation analysis of the tensor method in Proposition 5.3.

Tensor method. There were plenty of results analyzing tensor methods (Anandkumar et al., 2014a,b; Wang et al., 2015; Wang and Anandkumar, 2016; Song et al., 2016). We adapt the following result from (Zhong et al., 2017) that has a tight dependence on the problem dimension d and few restriction on sample $\{\boldsymbol{x}_i\}_{i=1}^{B}$.

This method first estimated the orthogonal column span U of training samples $\{\boldsymbol{x}_i | i \in B\}$ by using power method on P, and denoted the estimation by V. Then it conducted noisy tensor decomposition to $\boldsymbol{T}(V,V,V)$ with Algorithm 1 in (Kuleshov et al., 2015) and have $\{s_i\boldsymbol{u}_i\}_{i=1}^{B}$ as an estimation of $\{V^\top\boldsymbol{x}_i\}_{i=1}^{B}$, where $s_i \in \{\pm 1\}$ are unknown signs. Finally, by Algorithm 4 in (Zhong et al., 2017), s_i, r_i^* and eventually y_i are recovered. The time complexity of this method is $O(Bmd)$ (Zhong et al., 2017).

Proposition 5.3 (Adapted from the proof of Theorem 5.6 in (Zhong et al., 2017).)**.** *Consider matrix $\hat{P} = P + S$ and tensor $\hat{\boldsymbol{T}} = \boldsymbol{T} + \boldsymbol{E}$ with rank-B decomposition*

$$P = \sum_{i=1}^{B}\nu_i\boldsymbol{x}_i\boldsymbol{x}_i^\top,\ \boldsymbol{T} = \sum_{i=1}^{B}\lambda_i\boldsymbol{x}_i^{\otimes 3},$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ satisfying Assumption 5.1. Let V be the output of Algorithm 3 in (Zhong et al., 2017) with input P and $\{s_i\boldsymbol{u}_i\}_{i=1}^{B}$ be the output of Algorithm 1 in (Kuleshov et al., 2015) with input $\boldsymbol{T}(V,V,V)$, where $\{s_i\}$ are unknown signs. Suppose the perturbations satisfy

$$\|S\| \leq \mu,\ \|\boldsymbol{E}(V,V,V)\| \leq \gamma.$$

Let $N = \Theta(\log\frac{1}{\epsilon})$ be the iteration numbers of Algorithm 3 in (Zhong et al., 2017), where $\epsilon = \frac{\mu}{\nu_{\min}}$. Then w.h.p. we have

$$\|\boldsymbol{x}_i - s_iV\boldsymbol{u}_i\| \leq \tilde{O}(\frac{\mu}{\nu_{\min}\pi_{\min}}) + \tilde{O}(\frac{\kappa\gamma\sqrt{B}}{\lambda_{\min}\pi_{\min}^2}),$$

where $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$.

Remark 2. *With Assumption 5.1, the unknown signs s_i and the weights λ_i can be recovered by Algorithm 4 in (Zhong et al., 2017). Note that s_i and λ_i are discrete so the recovery is exact. Thus, exact y_i can be recovered and the estimation of $\boldsymbol{x}_i$ can be represent explicitly.*

To make use the result, what is left is mainly analyzing $\nu_{\min}$, $\lambda_{\min}$ and the weight ratio

$$\kappa = \left|\frac{\lambda_{\max}}{\lambda_{\min}}\right| = \frac{\max_{i\in[B]}|r_i^*|\,\mathbb{E}[\sigma^{(3)}(\boldsymbol{w}^\top\boldsymbol{x}_i)]}{\min_{i\in[B]}|r_i^*|\,\mathbb{E}[\sigma^{(3)}(\boldsymbol{w}^\top\boldsymbol{x}_i)]}.$$

Specifically, we have the following claim:

Claim 5.4. *Notice the maximum and minimum weights for the tensor slices in $\boldsymbol{T}$ are $\lambda_{\max} := \max_{i\in[B]} r_i^*\,\mathbb{E}[\sigma^{(3)}(\boldsymbol{w}^\top\boldsymbol{x}_i)] = \lambda\max_i r_i^*$, $\lambda_{\min} := \min_{i\in[B]} r_i^*\,\mathbb{E}[\sigma^{(3)}(\boldsymbol{w}^\top\boldsymbol{x}_i)] = \lambda\min_i r_i^*$. For $y_i \in \{\pm 1\}$, and odd activations like sigmoid or tanh, $\lambda_{\max} = \lambda_{\min} = \lambda$. Similarly, $\nu_{\max} = \nu\max_i r_i^*$, $\nu_{\min} = \nu\min_i r_i^*$. For $y_i \in \{\pm 1\}$, and activations with odd second order derivative like (Leaky)ReLU, $\nu_{\max} = \nu_{\min} = \nu$.*

This comes from a simple observation from the fact that standard normal distribution is symmetric: $\boldsymbol{w}^\top\boldsymbol{x} \sim \mathcal{N}(0,\|\boldsymbol{x}\|^2)$ doesn't depend on the direction of $\boldsymbol{x}$ but only its norm. Since we have normalized the samples to be norm 1, that part is invariant to different training sample. We discussed in Remark 1 how to deal with general r_i^* with its dependence on σ and y_i. In short, we can easily ensure κ to be constant with some small alteration in the designing of the weights.

In reality, if we deliberately set the norm of some sample $\boldsymbol{x}$ to be very small, the coefficient on the corresponding component of $\boldsymbol{T}$ will be very small. This makes the sample hard to learn, which is consistent with our intuition.

Concentration for matrix and tensor. We now bound μ and γ in Proposition 5.3. Recall the following notations:

$$\hat{P} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{B} r_i\sigma(\boldsymbol{w}_j^\top\boldsymbol{x}_i)(\boldsymbol{w}_j\boldsymbol{w}_j^\top - I), \tag{5.5}$$

$$P = \mathbb{E}[\sum_{i=1}^{B} r_i^*\sigma''(\boldsymbol{w}^\top\boldsymbol{x}_i)\boldsymbol{x}_i\boldsymbol{x}_i^\top], \tag{5.6}$$

$$\hat{\boldsymbol{T}} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{B} r_i\sigma(\boldsymbol{w}_j^\top\boldsymbol{x}_i)(\boldsymbol{w}_j^{\otimes 3} - \boldsymbol{w}_j\tilde{\otimes}I), \tag{5.7}$$

$$\boldsymbol{T} = \mathbb{E}[\sum_{i=1}^{B} r_i^*\sigma^{(3)}(\boldsymbol{w}^\top\boldsymbol{x}_i)\boldsymbol{x}_i^{\otimes 3}]. \tag{5.8}$$

Proposition 5.5. *If σ and $\boldsymbol{x}_i$ satisfies Assumption 5.1, $|y_i| \leq 1$, then for $\delta \leq \frac{2}{d}$ and $m \gtrsim \log(8/\delta)$, we have*

$$\|\hat{P} - P\| \leq \tilde{O}(\frac{B\sqrt{d}}{\sqrt{m}}) \tag{5.9}$$

with probability $1-\delta$.

Proposition 5.6. *If* σ *and* $\boldsymbol{x}_i$ *satisfies Assumption 5.1,* $|y_i| \le 1$*, and* $\|VV^\top - UU^\top\| \le 1/4$*, then for* $\delta \le \frac{2}{B}$ *and* $m \gtrsim \log(6/\delta)$

$$\|\bar{\boldsymbol{T}}(V,V,V) - \boldsymbol{T}(V,V,V)\| \le \tilde{O}(\frac{B^{5/2}}{\sqrt{m}}) \tag{5.10}$$

with probability $1-\delta$.

Here we have omitted the log factor of $\log(BmN/\delta)$ in the inequalities..

Remark 3. *For odd activation functions like tanh or sigmoid,* σ'' *is an odd function. Due to the symmetry of normal distribution,* $\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\sigma''(z)] = 0$*, which prevents us from using power method. In this case, we instead set*

$$\begin{aligned}\hat{P} :=& \frac{1}{m}(\sum_{j=1}^{m} g_j(\boldsymbol{w}_j) H_3(\boldsymbol{w}_j))(I, I, \boldsymbol{a}) \\ \approx& \mathbb{E}_{Z\sim\mathcal{N}(0,1)}[\sigma^{(3)}(Z)](\sum_{i=1}^{B} r_i^* \boldsymbol{x}_i^{\otimes 3})(I, I, \boldsymbol{a}),\end{aligned}$$

where $\boldsymbol{a}$ *is any unit vector. Note that the weight of auxiliary matrix* P *now is* $\nu = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[\sigma^{(3)}] \neq 0$*. Then all the steps with* P *in tensor decomposition and their analysis will be similar.*

Remark 4. *For piecewise linear activations like ReLU or LeakyReLU, note that* $\sigma^{(3)}$ *is the derivative of Dirac delta function* δ*, which is odd. Due to the symmetry of normal distribution,* $\mathbb{E}_{Z\sim\mathcal{N}(0,1)}[\sigma^{(3)}(Z)] = 0$*, which prevents us from using third order tensor decomposition. In this case, we should instead set*

$$\begin{aligned}\hat{\boldsymbol{T}} :=& \frac{1}{m}(\sum_{j=1}^{m} g_j(\boldsymbol{w}_j) H_4(\boldsymbol{w}_j))(I, I, I, \boldsymbol{a}) \\ \approx& \mathbb{E}_{Z\sim\mathcal{N}(0,1)}[\sigma^{(4)}(Z)](\sum_{i=1}^{B} r_i^* \boldsymbol{x}_i^{\otimes 4})(I, I, I, \boldsymbol{a}),\end{aligned}$$

where $\boldsymbol{a}$ *is any unit vector. Then we conduct tensor decomposition to recover* $\boldsymbol{x}_i$ *from* $\hat{\boldsymbol{T}}$*. For instance, the weight in the tensor now becomes* $\lambda = \mathbb{E}_{Z\sim\mathcal{N}(0,1)}[\sigma^{(4)}(Z)] = -\frac{1}{\sqrt{2\pi}}$ *for ReLU. All the analysis with* $\boldsymbol{T}$ *still applies in a similar way.*

6 EXPERIMENTS

In this section, we present some experimental verification to our theoretical results on synthetic data. Recall that we use the gradient $\nabla_{\boldsymbol{a}} L(\Theta)$ to compute the tensor $\hat{\boldsymbol{T}}$. Instead, we can also estimate $\boldsymbol{T}$ with the gradient $\nabla_W L(\Theta)$ with respect to the first layer weights W in a similar way (see Appendix B). We mainly use $\nabla_W L(\Theta)$ in our experiments but also make comparison between two methods. We also present the reconstruction result of MNIST in Appendix D.

We consider a two-layer neural network with fixed width $m = 5000$. The data satisfies $B = 2$ and $\boldsymbol{x}_i = \boldsymbol{e}_i$, $i = 1, 2$. We first set the activation function as $\sigma(x) = x^2 + x^3$, a simple example of Assumption 5.1. When we run Algorithm 1 with tensor decomposition method following Zhong et al. (2017), the reconstruction loss of estimating $\boldsymbol{T}$ with $\nabla_{\boldsymbol{a}} L$ and $\nabla_W L$ are shown in the left of Fig. 2, where both reconstruction losses are small and using $\nabla_W L$ is better.

For more realistic activation functions, our method can still recover data with small reconstruction loss. The image on the right of Fig. 2 shows that the construction loss is also small when σ is tanh or sigmoid, which aligns with our theoretical results. Here we use the trick in Remark 3.

7 CONCLUSION AND DISCUSSIONS

In this paper, we aim to theoretically investigate the data leakage problem. To the best of our knowledge, this is the first theoretical work to prove that one can reconstruct the training samples from the model gradient. Specifically, we only need mildly overparametrized neural networks, where the width scales **linearly** (and hidden polylog factors) with the input dimension d. In this section, we seek to have more thorough discussions on what can be inferred from our findings, and we believe this area is still wide open for more theoretical investigations.

7.1 Discussions

On the identifiability of the training samples. First, when the training samples are independent, a single gradient alone can identify them (up to some scaling factors). Therefore, if the privacy attacker has unlimited computing power, one can reconstruct the samples. (We make additional assumptions to get a computationally efficient (polynomial-time) reconstructing algorithm.)

Therefore, when prior work finds failure cases in the attack with the gradient, it is more likely that our existing optimization algorithms cannot effectively resolve the nonconvex reconstruction loss. Adding prior knowledge to the loss changes the dynamics of the optimization procedure, which might ease the reconstruction procedure.

On the effect of neural network sizes. Based on our results, a wider neural network ensures a more accurate data reconstruction. However, a deeper neural network is neither helping nor hurting. This is because, in our current design, the first $l-2$ layers function to preserve the input, while the last two layers generate observations that form a matrix sensing problem.

From the perspective of equation counting, a deeper network provides more information on identifying the training samples. However, there are also contrary side proofs. When calculating the gradient from a two-layer neural network,

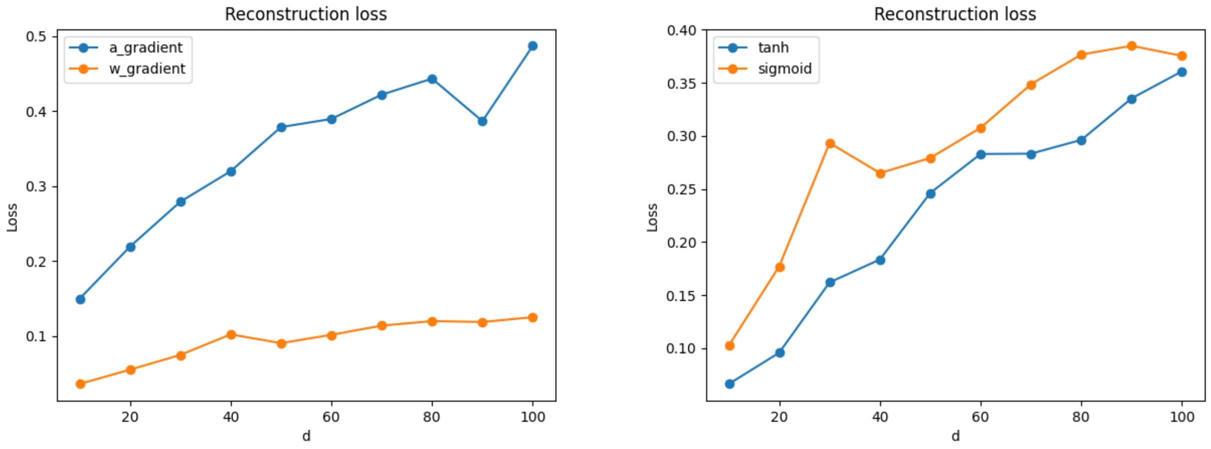

Figure 2: For a two-layer neural network with fixed width m, reconstruction loss for different data dimension d. **Left:** using $\nabla_{\boldsymbol{a}} L$ and $\nabla_W L$ when $\sigma(x) = x^2 + x^3$; **Right:** different activation functions tanh and sigmoid using $\nabla_W L$.

especially evident from the linear and quadratic activations, the gradient on the first layer doesn't reveal more information than the gradient of the second layer.

Therefore it is not yet clear whether we can benefit from the depth during the reconstruction procedure. It requires further exploration whether this is the caveat of our analysis or the fact that the depth of the network does not play as important a role as the width in privacy attacks.

Distinctions to private algorithms for convex optimization. Our algorithm is similar to a sensing problem as we observe measurements on the input samples, and our target is reconstructing them. However, we only observe the sum of the measurements for individual samples, as the gradient is aggregated with all the training samples. Consequently, linear or quadratic measurements fail the gradient inversion task, as we demonstrated in Section 5.

Meanwhile, since linear functions do not identify individual samples, it is a side proof that the linear and neural network functions are fundamentally different in their vulnerability in privacy attacks. Therefore, we also conjecture that most previous analyses in differential privacy that worked with convex functions (convex loss on a linear prediction function) (Bassily et al., 2014, 2019; Altschuler and Talwar, 2022) do not generalize to the neural network.

Discussions on private algorithms. It is widely adopted to add noise in stochastic convex optimization for differential private algorithms, namely, by adding noise generated from $Z \sim \mathcal{N}(0, \sigma^2 I)$ in our observed gradient G. Extensive work and analysis demonstrates their ability to keep differential privacy in the convex setting (Bassily et al., 2019, 2014; Altschuler and Talwar, 2022). However, its effect on our reconstruction is limited. In fact, as we observe $g_j = \sum_{i=1}^{B} r_i \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i) + z_i, z_i \sim \mathcal{N}(0, \sigma^2)$, our estimated matrix and tensor will be shifted with another $\hat{S} := \sum_j z_j H_2(\boldsymbol{w}_j)$ and $\hat{\boldsymbol{E}} := \sum_j z_j H_3(\boldsymbol{w}_j)$. One can verify that $\|\hat{S}\|$ and $\|\hat{\boldsymbol{E}}(\hat{V}, \hat{V}, \hat{V})\|$ has bounds similar to Proposition 5.5 and 5.6. Therefore, the noisy SGD won't significantly affect (and only changes the constant in) the reconstruction error. This is additional proof that such conclusions in convex optimization do not apply here.

Therefore, we believe it is more promising to focus on other private algorithms by encoding or perturbing the input samples (Sun et al., 2021).

7.2 Future Work

Since the theoretical understanding of gradient inversion is still in its nascent phase, our work inevitably has some limitations, and we hope to encourage more future work to fill in the gap. Our work highlights the urgent need to design private algorithms in federated learning. However, here we only focus on the remaining problems of reconstructing sensitive training data.

From the perspective of parameter counting, for two-layer neural networks, we only need the number of parameters md to be larger than Bd, the total dimension of unknown samples. However, we require $m \gg d$ instead of $m \gg B$ in this paper. It is thus important to get either a lower bound or a tighter upper bound to better understand the dependence of hidden nodes m on batch size B and problem dimension d.

Second, it is important to design better attack algorithms to exploit the effect of the depth of neural networks fully.

Finally but not least, some different neural net architectures will affect the attack. It will be interesting to design the algorithm when the intermediate layers are convolutional instead of fully-connected layers.

Acknowledgements

JDL acknowledges support of the ARO under MURI Award W911NF-11-1-0304, the Sloan Research Fellowship, NSF CCF 2002272, NSF IIS 2107304, NSF CIF 2212262, ONR Young Investigator Award, and NSF CAREER Award 2144994. QL would like to thank Jason Altschuler for helpful discussions and thank Sheng Liu for helping with the implementation.

References

Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

J. M. Altschuler and K. Talwar. Privacy of noisy stochastic gradient descent: More iterations without more privacy loss. *arXiv preprint arXiv:2205.13710*, 2022.

A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of machine learning research*, 15:2773–2832, 2014a.

A. Anandkumar, R. Ge, and M. Janzamin. Guaranteed non-orthogonal tensor decomposition via alternating rank-1 updates. *arXiv:1402.5180*, 2014b.

M. Balunovic, D. I. Dimitrov, R. Staab, and M. Vechev. Bayesian framework for gradient leakage. In *International Conference on Learning Representations*, 2022.

R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 464–473. IEEE, 2014.

R. Bassily, V. Feldman, K. Talwar, and A. Guha Thakurta. Private stochastic convex optimization with optimal rates. *Advances in neural information processing systems*, 32, 2019.

A. Bhaskara, M. Charikar, and A. Vijayaraghavan. Uniqueness of tensor decompositions with applications to polynomial identifiability. In *Conference on Learning Theory*, pages 742–778. PMLR, 2014.

T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

J. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of “eckart-young” decomposition. *Psychometrika*, 35(3): 283–319, 1970.

C. Chen and N. D. F. Campbell. Understanding training-data leakage from gradients in neural networks for imageclassifications. In *NeurIPS 2021 Workshop Privacy in Machine Learning*, 2021.

S. Du, J. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019.

L. Fan, K. W. Ng, C. Ju, T. Zhang, C. Liu, C. S. Chan, and Q. Yang. *Rethinking Privacy Preserving Deep Learning: How to Evaluate and Thwart Privacy Attacks*, pages 32–50. Springer International Publishing, 2020.

J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16: 1–84, 1970.

J. Håstad. Tensor rank is np-complete. In *International Colloquium on Automata, Languages, and Programming*, pages 451–460. Springer, 1989.

A. Hatamizadeh, H. Yin, H. Roth, W. Li, J. Kautz, D. Xu, and P. Molchanov. Gradvit: Gradient inversion of vision transformers. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

C. J. Hillar and L.-H. Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

F. L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7(1-4):39–79, 1928.

Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora. Evaluating gradient inversion attacks and defenses in federated learning. In *NeurIPS*, 2021.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

J. Jeon, j. Kim, K. Lee, S. Oh, and J. Ok. Gradient inversion with generative image prior. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

X. Jin, P.-Y. Chen, C.-Y. Hsu, C.-M. Yu, and T. Chen. Catastrophic data leakage in vertical federated learning. In *Advances in Neural Information Processing Systems*, 2021.

H. A. L. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14(3):105–122, 2000.

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

J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv:1610.02527*, 2016a.

J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. 2016b.

J. B. Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977.

V. Kuleshov, A. Chaganty, and P. Liang. Tensor factorization via matrix factorization. In *Artificial Intelligence and Statistics*, pages 507–516. PMLR, 2015.

Q. Lei, A. Jalal, I. S. Dhillon, and A. G. Dimakis. Inverting deep generative models, one layer at a time. *Advances in neural information processing systems*, 32, 2019.

K. Mamis. Extension of stein's lemma derived by using an integration by differentiation technique. *Examples and Counterexamples*, 2:100077, 2022.

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

L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.

V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Z. Song, D. Woodruff, and H. Zhang. Sublinear time orthogonal tensor decomposition. *Advances in Neural Information Processing Systems*, 29, 2016.

C. M. Stein. Estimation of the mean of a multivariate normal distribution. *The annals of Statistics*, pages 1135–1151, 1981.

J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen. Soteria: Provable defense against privacy leakage in federated learning from representation perspective. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9311–9319, 2021.

Y. Wang and A. Anandkumar. Online and differentially-private tensor decomposition. *Advances in Neural Information Processing Systems*, 29, 2016.

Y. Wang, H.-Y. Tung, A. J. Smola, and A. Anandkumar. Fast and guaranteed tensor decomposition via sketching. *Advances in neural information processing systems*, 28, 2015.

Y. Wang, J. Deng, D. Guo, C. Wang, X. Meng, H. Liu, C. Ding, and S. Rajasekaran. SAPAG: A self-adaptive privacy attack from gradients. *arXiv:2009.06228*, 2020.

W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu. 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*, page 545–566, 2020.

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

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

Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv:1806.00582*, 2018.

K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 4140–4149. PMLR, 2017.

J. Zhu and M. B. Blaschko. R-{gap}: Recursive gradient attack on privacy. In *International Conference on Learning Representations*, 2021.

L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Reconstructing Images from Model Gradient: Supplementary Materials

A PROOFS

A.1 Concentration Bound for Vectors

We formalize the concentration bound when $p = 1$ in the following propositions.

Proposition A.1. *If σ and $\boldsymbol{x}_i$ satisfies Assumption 5.1, $|y_i| \le 1$, then for $\delta \le \frac{6}{d+1}$ and $m \gtrsim \log(6/\delta)$, we have*

$$\left\| \frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{B} r_i \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i) \boldsymbol{w}_j - \mathbb{E} \sum_{i=1}^{B} \tilde{h}_i'(\boldsymbol{w}^\top \boldsymbol{x}_i) \boldsymbol{x}_i \right\| \le \tilde{O}(\frac{B\sqrt{d}}{\sqrt{m}}) \tag{A.1}$$

with probability $1-\delta$, where $\tilde{h}_i(\boldsymbol{w}_j^\top \boldsymbol{x}_i) = r_i^* \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)$.

The following lemma is crucial to prove the proposition above.

Lemma A.2 (Matrix Bernstein for unbounded matrices; adapted from Lemma B.7 in (Zhong et al., 2017))**.** *Let $\mathcal{Z}$ denote a distribution over $\mathbb{R}^{d_1 \times d_2}$. Let $d = d_1 + d_2$. Let $Z_1, Z_2, \cdots, Z_m$ be i.i.d. random matrices sampled from $\mathcal{Z}$. Let $\bar{Z} = \mathbb{E}_{Z\sim\mathcal{Z}}[Z]$ and $\widehat{Z} = \frac{1}{m}\sum_{i=1}^{m} Z_i$. For parameters $\delta_0 \in (0,1), M = M(\delta_0, m) \ge 0, \nu > 0, L > 0$, if the distribution $\mathcal{B}$ satisfies the following four properties,*

$$\begin{aligned}
(I) &\quad \mathbb{P}_{Z\sim\mathcal{Z}}\left\{\|Z\| \le M\right\} \ge 1 - \frac{\delta_0}{m} \\
(II) &\quad \max\left(\left\| \mathop{\mathbb{E}}_{Z\sim\mathcal{Z}}\left[ZZ^\top\right] \right\|, \left\| \mathop{\mathbb{E}}_{Z\sim\mathcal{Z}}\left[Z^\top Z\right] \right\| \right) \le \nu \\
(III) &\quad \max_{\|\boldsymbol{a}\|=\|\boldsymbol{b}\|=1} \left(\mathop{\mathbb{E}}_{Z\sim\mathcal{Z}}\left[\left(\boldsymbol{a}^\top Z \boldsymbol{b}\right)^2 \right] \right)^{1/2} \le L
\end{aligned}$$

Then we have for any $0 < \delta_1 < 1$, if $\delta_1 \le \frac{1}{d}$ and $m \gtrsim \log(1/\delta_1)$, with probability at least $1 - \delta_1 - \delta_0$,

$$\|\widehat{Z} - \bar{Z}\| \lesssim \sqrt{\frac{\log(1/\delta_1)(\nu + \|\bar{Z}\|^2 + M\|\bar{Z}\|) + \delta_0 L^2}{m}}$$

There are also some useful facts that we need in the proof.

Fact A.1. *If $\boldsymbol{w}$ is a d-dimensional standard normal random variable, then*

$$\mathbb{P}\left\{\|\boldsymbol{w}\| \ge t\right\} \le 2\exp\left\{-\frac{t^2}{2d}\right\}.$$

Fact A.2. *If $\boldsymbol{x}$ is a d-dimensional standard normal random vector, then*

$$\left\|\mathbb{E}_{\boldsymbol{w}} \sigma(\boldsymbol{w}^\top \boldsymbol{x})(\boldsymbol{w}\boldsymbol{w}^\top - I)\right\| \le O(1).$$

Proof. We can assume $\boldsymbol{x} = \boldsymbol{e}_1$ w.l.o.g. and we have that $\mathbb{E}_{\boldsymbol{w}}\sigma(\boldsymbol{w}^\top \boldsymbol{x})(\boldsymbol{w}\boldsymbol{w}^\top - I)$ is diagonal and each element is $O(1)$. □

With the preparation above, we can propose the proof of Proposition A.1 and 5.5:

Proof of Proposition A.1. By Stein's Lemma, $\mathbb{E}\left[\sum_{i=1}^B \tilde{h}_i'(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i\right] = \mathbb{E}\left[\sum_{i=1}^B r_i^* \sigma(\boldsymbol{w}^\top x_i) w\right]$. Let $Z_j := \sum_{i=1}^B r_i \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i) w_j$ and $\tilde{Z}_j := \sum_{i=1}^B r_i^* \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)\boldsymbol{w}_j$, then we only need to bound

$$\left\| \frac{1}{m}\sum_{j=1}^m Z_j - \mathbb{E}\tilde{Z} \right\| \leq \left\| \frac{1}{m}\sum_{j=1}^m Z_j - \frac{1}{m}\sum_{j=1}^m \tilde{Z}_j \right\| + \left\| \frac{1}{m}\sum_{j=1}^m \tilde{Z}_j - \mathbb{E}\tilde{Z} \right\|. \tag{A.2}$$

The first term in Eq. (A.2). We first consider $\sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)$. Since σ is 1-Lipschitz,

$$\left|\sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)\right| \leq C_\sigma \left|\boldsymbol{w}_j^\top \boldsymbol{x}_i\right| \leq C_\sigma \sqrt{2\log(12B/\delta)} \tag{A.3}$$

with probability $1 - \frac{\delta}{6B}$ for some absolute constant C_σ. Then for another absolute constant C_σ',

$$|r_i - r_i^*| = \left| \frac{1}{m}\sum_{j=1}^m \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i) - \mathbb{E}[\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)] \right| \leq C_\sigma' \sqrt{\frac{\log(12B/\delta)}{m}} \tag{A.4}$$

with probability $1 - \frac{\delta}{6B}$. Besides, $|\sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)| \leq C_\sigma\sqrt{2\log(12Bm/\delta)}$ with probability $1 - \frac{\delta}{6Bm}$ and by Fact A.1, $\|\boldsymbol{w}_j\| \leq \sqrt{2d\log(12m/\delta)}$ with probability $1 - \frac{\delta}{6m}$. Then by union bound, we have

$$\begin{aligned} \left\| \frac{1}{m}\sum_{j=1}^m Z_j - \frac{1}{m}\sum_{j=1}^m \tilde{Z}_j \right\| &= \frac{1}{m}\left\| \sum_{j=1}^m \sum_{i=1}^B (r_i - r_i^*)\sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)\boldsymbol{w}_j \right\| \\ &\leq \frac{1}{m}\sum_{i=1}^B |r_i - r_i^*| \sum_{j=1}^m \left|\sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)\right| \|\boldsymbol{w}_j\| \\ &\leq C_r B\sqrt{\frac{d\log(12B/\delta)\log(12m/\delta)\log(12Bm/\delta)}{m}} \end{aligned} \tag{A.5}$$

with probability $1 - \frac{\delta}{2}$, where C_r is an absolute constant.

The second term in Eq. (A.2). Since $\boldsymbol{x}_i$ and y_i are bounded, $r_i^* = \mathbb{E}[r_i] = y_i - \mathbb{E}[\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)] \leq |y_i| + C_\sigma \mathbb{E}[|\boldsymbol{w}^\top \boldsymbol{x}_i|]$ is bounded. Then we check the conditions of Lemma A.2.

(I) By the proof above, we have

$$\left\|\tilde{Z}_j\right\| \leq \sum_{i=1}^B r_i^* \left|\sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)\right| \|\boldsymbol{w}_j\| \leq C_0 B\sqrt{d\log(12m/\delta)\log(12Bm/\delta)} \tag{A.6}$$

with probability $1 - \frac{\delta}{3m}$.

(II) We have

$$\begin{aligned} \max\left\{\left\|\mathbb{E}\left[\tilde{Z}^\top \tilde{Z}\right]\right\|, \left\|\mathbb{E}\left[\tilde{Z}\tilde{Z}^\top\right]\right\|\right\} &\leq \mathbb{E}\left[\left\|\tilde{Z}\right\|^2\right] \leq B\sum_{i=1}^B \mathbb{E}\left[\left\|r_i^*\sigma\left(\boldsymbol{w}_j^\top \boldsymbol{x}_i\right)\boldsymbol{w}_j\right\|^2\right] \\ &\lesssim B\sum_{i=1}^B \mathbb{E}\left[\sigma\left(\boldsymbol{w}_j^\top \boldsymbol{x}_i\right)^2\right]^{1/2} \mathbb{E}\left[\|\boldsymbol{w}_j\|^2\right]^{1/2} \\ &\lesssim B^2 d. \end{aligned} \tag{A.7}$$

(III) We have

$$\max_{\|\boldsymbol{a}\|=|b|=1} \left(\mathbb{E}_{Z\sim\mathcal{Z}}\left[\left(\boldsymbol{a}^\top Z b\right)^2\right]\right)^{1/2} \leq \left(\mathbb{E}\left[\left\|\tilde{Z}\right\|^2\right]\right)^{\frac{1}{2}} \lesssim B\sqrt{d}. \tag{A.8}$$

Additionally, we also have to bound $\left\|\mathbb{E}\left[\tilde{Z}\right]\right\|$.

$$\mathbb{E}\left[\tilde{Z}\right] = \sum_{i=1}^{B} r_i^* \mathbb{E}\left[\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{w}\right] = \sum_{i=1}^{B} r_i^* \mathbb{E}\left[\sigma'(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i\right] = \sum_{i=1}^{B} r_i^* \mathbb{E}_{z\sim\mathcal{N}(0,1)}\left[\sigma'(z)\right]\boldsymbol{x}_i \tag{A.9}$$

by Stein's lemma. Then $\left\|\mathbb{E}\left[\tilde{Z}\right]\right\| \lesssim \sum_{i=1}^{B} \|\boldsymbol{x}_i\| \lesssim B$

By Eq. (A.6)-(A.9), we can apply Lemma A.2:

$$\begin{aligned}\left\|\frac{1}{m}\sum_{j=1}^{m}\tilde{Z}_j - \mathbb{E}\tilde{Z}\right\| &\lesssim \sqrt{\frac{(\log(6/\delta))\left(B^2 d + B^2 + B^2\sqrt{d\log(12m/\delta)\log(12Bm/\delta)} + B^2 d\right)}{m}} \\ &\lesssim B\sqrt{\frac{d\log(6/d)\log(12m/\delta)\log(12Bm/\delta)}{m}}\end{aligned} \tag{A.10}$$

with probability $1-\frac{\delta}{2}$.

Putting Eq. (A.5) and Eq. (A.10) together, we have

$$\left\|\frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{B} r_i \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)\boldsymbol{w}_j - \mathbb{E}\sum_{i=1}^{B}\tilde{h}_i'\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i\right\| \le \tilde{O}(\frac{B\sqrt{d}}{\sqrt{m}})$$

with probability $1-\delta$. □

A.2 Concentration Bound for P

A lemma is needed for the proof of Proposition 5.5.

Lemma A.3. *If σ is 1-Lipschitz and $\mathbb{E}_{z\sim\mathcal{N}(0,1)}\sigma''(z) < \infty$, $\|\boldsymbol{x}\| \le 1$, then for $\delta \le 1/d$ and $m \gtrsim \log(2/\delta)$, we have*

$$\left\|\frac{1}{m}\sum_{j=1}^{m}\sigma(\boldsymbol{w}_j^\top \boldsymbol{x})(\boldsymbol{w}_j\boldsymbol{w}_j^\top - I) - \mathbb{E}\sigma(\boldsymbol{w}^\top \boldsymbol{x})(\boldsymbol{w}\boldsymbol{w}^\top - I)\right\| \lesssim \sqrt{\frac{d\log(2/\delta)\log(8m/\delta)}{m}} \tag{A.11}$$

with probability $1-\delta$, where I is the identity matrix.

Proof. Denote $Z_j = \sigma(\boldsymbol{w}_j^\top \boldsymbol{x})(\boldsymbol{w}_j\boldsymbol{w}_j^\top - I)$. We check the conditions of Lemma A.2.

(I) We first bound the norm of Z_j:

$$\|Z_j\| \le C_0\left|\sigma(\boldsymbol{w}_j^\top \boldsymbol{x})\right|\|\boldsymbol{w}_j\|^2 \le Cd\log(4m/\delta) \tag{A.12}$$

with probability $1-\frac{\delta}{2m}$, by Fact A.1 and modifying Eq. (A.3).

(II) We have

$$\max\left\{\left\|\mathbb{E}\left[Z^\top Z\right]\right\|, \left\|\mathbb{E}\left[ZZ^\top\right]\right\|\right\} = \left\|\mathbb{E}\left[Z^2\right]\right\| = \left\|\mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0,I)}\sigma(\boldsymbol{w}^\top \boldsymbol{x})^2(\boldsymbol{w}\boldsymbol{w}^T - I)^2\right\|. \tag{A.13}$$

We can assume $\boldsymbol{x} = \boldsymbol{e}_1$ w.l.o.g. by the symmetry of normal distribution. Then we have to bound $\|P\| := \left\|\mathbb{E}\sigma(w_1)^2(\boldsymbol{w}\boldsymbol{w}^\top - I)^2\right\|$, where we can directly compute that P is a diagonal matrix with positive elements smaller than $(d+1)\max\left\{\mathbb{E}\sigma(w_1)^2 w_1^2, \mathbb{E}\sigma(w_1)^2\right\}$. Since σ is 1-Lipschitz, $\|B\| \le O(d)$.

(III) For $\max_{\|\boldsymbol{a}\|=\|\boldsymbol{b}\|=1}(\mathbb{E}(\boldsymbol{a}^\top Z\boldsymbol{b})^2)^{1/2}$, it reaches the maximal when $\boldsymbol{a} = \boldsymbol{b}$ since Z is symmetric. Thus, we have

$$\mathbb{E}(\boldsymbol{a}^\top Z\boldsymbol{a})^2 = \mathbb{E}\sigma(\boldsymbol{w}^\top \boldsymbol{x})^2(\boldsymbol{a}^\top(\boldsymbol{w}\boldsymbol{w}^\top - I)\boldsymbol{a})^2. \tag{A.14}$$

Similarly, we can assume that $\boldsymbol{x} = \boldsymbol{e}_1$. Then we can compute the expectation:

$$\begin{aligned}
\mathbb{E}(\boldsymbol{a}^\top Z\boldsymbol{a})^2 &= \mathbb{E}\sigma(w_1)^2\left(\sum_{i=1}^d a_i^2(w_i^2-1)+\sum_{i\neq j} a_ia_jw_iw_j\right)^2 \\
&= \mathbb{E}\left[\sum_{i=1}^d \sigma(w_1)^2 a_i^4(w_i-1)^2 + \sum_{i\neq j}\sigma(w_1)^2a_i^2a_j^2w_i^2w_j^2\right] \\
&\lesssim \left(\sum_{i=1}^d a_i^2\right)^2 = 1.
\end{aligned} \tag{A.15}$$

Thus, $\max_{\|\boldsymbol{a}\|=\|\boldsymbol{b}\|=1}(\mathbb{E}(\boldsymbol{a}^\top Z\boldsymbol{b})^2)^{1/2} \le O(1)$.

Moreover, we have to bound $\|\mathbb{E}[Z]\|$, and we have

$$\mathbb{E}\,[Z] = \mathbb{E}\sigma(\boldsymbol{w}^\top\boldsymbol{x})(\boldsymbol{w}\boldsymbol{w}^\top - I) = \mathbb{E}\sigma''(\boldsymbol{w}^\top\boldsymbol{x})\boldsymbol{x}\boldsymbol{x}^\top = \mathbb{E}_{z\sim\mathcal{N}(0,1)}\sigma''(z)\boldsymbol{x}\boldsymbol{x}^\top \tag{A.16}$$

by Stein's lemma. Then $\|\mathbb{E}[Z]\| \lesssim \|\boldsymbol{x}\boldsymbol{x}^\top\| \le O(1)$.

By Lemma A.2, we can combine these estimations:

$$\left\|\frac{1}{m}\sum_{j=1}^m \sigma(\boldsymbol{w}_j^\top\boldsymbol{x})(\boldsymbol{w}_j\boldsymbol{w}_j^\top - I) - \mathbb{E}\sigma(\boldsymbol{w}^\top\boldsymbol{x})(\boldsymbol{w}\boldsymbol{w}^\top - I)\right\| \lesssim \sqrt{\frac{\log(2/\delta)(d\log(4m/\delta))}{m}} \tag{A.17}$$

with probability $1-\delta$. □

Proof of Proposition 5.5. Denote $Z_j := \sum_{i=1}^B r_i\sigma(\boldsymbol{w}_j^\top\boldsymbol{x}_i)(\boldsymbol{w}_j\boldsymbol{w}_j^\top - I)$ and $\tilde{Z}_j := \sum_{i=1}^B r_i^*\sigma(\boldsymbol{w}_j^\top\boldsymbol{x}_i)(\boldsymbol{w}_j\boldsymbol{w}_j^\top - I)$, we have

$$\left\|\frac{1}{m}\sum_{j=1}^m Z_j - \mathbb{E}\tilde{Z}\right\| \le \left\|\frac{1}{m}\sum_{j=1}^m Z_j - \frac{1}{m}\sum_{j=1}^m \tilde{Z}_j\right\| + \left\|\frac{1}{m}\sum_{j=1}^m \tilde{Z}_j - \mathbb{E}\tilde{Z}\right\|. \tag{A.18}$$

The first term in Eq. (A.18). We have

$$\begin{aligned}
\left\|\frac{1}{m}\sum_{j=1}^m Z_j - \frac{1}{m}\sum_{j=1}^m \tilde{Z}_j\right\| &\le \sum_{i=1}^B |r_i^* - r_i|\left\|\frac{1}{m}\sum_{j=1}^m \sigma(\boldsymbol{w}_j^\top\boldsymbol{x}_i)(\boldsymbol{w}_j\boldsymbol{w}_j^\top - I)\right\| \\
&\lesssim \sqrt{\frac{\log(8B/\delta)}{m}}\sum_{i=1}^B(\|\frac{1}{m}\sum_{j=1}^m \sigma(\boldsymbol{w}_j^\top\boldsymbol{x}_i)(\boldsymbol{w}_j\boldsymbol{w}_j^\top - I) - \mathbb{E}\sigma(\boldsymbol{w}^\top\boldsymbol{x}_i)(\boldsymbol{w}\boldsymbol{w}^\top - I)\| \\
&\quad + \big\|\mathbb{E}\sigma(\boldsymbol{w}^\top\boldsymbol{x}_i)(\boldsymbol{w}\boldsymbol{w}^\top - I)\big\|) \\
&\lesssim B\sqrt{\frac{d\log(8B/\delta)\log(8B/\delta)\log(16Bm/\delta)}{m}}
\end{aligned} \tag{A.19}$$

with probability $1-\frac{\delta}{2}$, where the second inequality is by modifying Eq. (A.4) and the last inequality is by Lemma A.3 and Fact A.2.

The second term in Eq. (A.18). By Lemma A.3, we have

$$\begin{aligned}
\left\|\frac{1}{m}\sum_{j=1}^m \tilde{Z}_j - \mathbb{E}\tilde{Z}\right\| &\lesssim \sum_{i=1}^B \left\|\frac{1}{m}\sum_{j=1}^m \sigma(\boldsymbol{w}_j^\top\boldsymbol{x}_i)(\boldsymbol{w}_j\boldsymbol{w}_j^\top - I) - \mathbb{E}\sigma(\boldsymbol{w}^\top\boldsymbol{x}_i)(\boldsymbol{w}\boldsymbol{w}^\top - I)\right\| \\
&\lesssim B\sqrt{\frac{d\log(4B/\delta)\log(8Bm/\delta)}{m}}
\end{aligned} \tag{A.20}$$

with probability $1-\frac{\delta}{2}$, where the first inequality is because r_i^* is bounded and the second inequality is by Lemma A.3.

Finally we can combine Eq. (A.19) and Eq. (A.20) and have

$$\left\| \frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{B} r_i \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)(\boldsymbol{w}_j \boldsymbol{w}_j^\top - I) - \mathbb{E} \sum_{i=1}^{B} \tilde{h}_i''(\boldsymbol{w}^\top \boldsymbol{x}_i) \boldsymbol{x}_i \boldsymbol{x}_i^\top \right\| \leq \tilde{O}(\frac{B\sqrt{d}}{\sqrt{m}})$$

with probability $1 - \delta$. □

A.3 Concentration Bound for $T(V, V, V)$

For $U \in \mathbb{R}^{d \times B}$ is the orthogonal column span of $\{\boldsymbol{x}_i\}_{i=1}^B$, we assume that the difference between $VV^\top$ and $UU^\top$ is bounded by a constant.

Proof of Proposition 5.6. We consider $R \in \mathbb{R}^{B \times B^2}$, the flatten along the first dimension of $\boldsymbol{T}(V, V, V)$. Since for a symmetric 3rd-order tensor E and its flatten along the first dimension $E^{(1)}$ we have $\|E\| \leq \|E^{(1)}\|$, we can bound $\|\hat{R} - R\|$.

Denote $W_j := \sum_{i=1}^B r_i \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)(\boldsymbol{w}_j^{\otimes 3} - 3\boldsymbol{w}_j \otimes I)$, $\tilde{W}_j := \sum_{i=1}^B r_i^* \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)(\boldsymbol{w}_j^{\otimes 3} - 3\boldsymbol{w}_j \otimes I)$, $Z_j := W_j^{(1)}(V, V, V)$ and $\tilde{Z}_j := \tilde{W}_j^{(1)}(V, V, V)$. Then $\mathbb{E}\tilde{Z} = \boldsymbol{T}^{(1)}(V, V, V)$ by Stein's lemma and we have

$$\left\| \frac{1}{m} \sum_{j=1}^{m} Z_j - \mathbb{E}\tilde{Z} \right\| \leq \left\| \frac{1}{m} \sum_{j=1}^{m} Z_j - \frac{1}{m} \sum_{j=1}^{m} \tilde{Z}_j \right\| + \left\| \frac{1}{m} \sum_{j=1}^{m} \tilde{Z}_j - \mathbb{E}\tilde{Z} \right\|. \tag{A.21}$$

The first term in Eq. (A.21). Note that $V^\top \boldsymbol{w}_j \sim \mathcal{N}(0, I_B)$ so we have

$$\begin{aligned} \left\| \frac{1}{m} \sum_{j=1}^{m} Z_j - \frac{1}{m} \sum_{j=1}^{m} \tilde{Z}_j \right\| &\lesssim \frac{1}{m} \sum_{i=1}^{B} |r_i - r_i^*| \sum_{j=1}^{m} |\sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)| \|V^\top \boldsymbol{w}_j\|^3 \\ &\leq \tilde{O}(\frac{B^{5/2}}{\sqrt{m}}) \end{aligned} \tag{A.22}$$

with probability $1 - \frac{\delta}{2}$, similar to the proof of Proposition A.1.

The second term in Eq. (A.21). We check the conditions of Lemma A.2.

(I) By the proof above, we have

$$\left\| \tilde{Z}_j \right\| \leq \sum_{i=1}^{B} r_i^* |\sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)| \|V^\top \boldsymbol{w}_j\|^3 \leq \tilde{O}(\frac{B^{5/2}}{\sqrt{m}}) \tag{A.23}$$

with probability $1 - \frac{\delta}{3m}$.

(II) We have

$$\max \left\{ \left\| \mathbb{E}\left[\tilde{Z}^\top \tilde{Z}\right] \right\|, \left\| \mathbb{E}\left[\tilde{Z} \tilde{Z}^\top\right] \right\| \right\} \leq \mathbb{E}\left[\left\| \tilde{Z} \right\|^2 \right] \lesssim B \sum_{i=1}^{B} \mathbb{E}\left[\sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i)\right]^{1/2} \mathbb{E}\left[\|V^\top \boldsymbol{w}_j\|^6\right]^{1/2} \lesssim B^5. \tag{A.24}$$

(III) We have

$$\max_{\|\boldsymbol{a}\| = |\boldsymbol{b}| = 1} \left(\mathop{\mathbb{E}}_{Z \sim \mathcal{Z}} \left[(\boldsymbol{a}^\top Z b)^2 \right] \right)^{1/2} \leq \left(\mathbb{E}\left[\left\| \tilde{Z} \right\|^2 \right] \right)^{1/2} \lesssim B^{5/2}. \tag{A.25}$$

Additionally, we also have to bound $\left\| \mathbb{E}\left[\tilde{Z}\right] \right\|$.

$$\mathbb{E}[\tilde{Z}] = \sum_{i=1}^{B} r_i^* \left(\mathbb{E}[\sigma^{(3)}(\boldsymbol{w}^\top \boldsymbol{x}_i) \boldsymbol{x}_i^{\otimes 3}] \right)^{(1)} (V, V, V) = \sum_{i=1}^{B} \mathbb{E}_{z \sim \mathcal{N}(0,1)}[\sigma^{(3)}(z)](V^\top \boldsymbol{x}_i) \mathrm{vec}\left[(V^\top \boldsymbol{x}_i)(V^\top \boldsymbol{x}_i)^\top \right]^\top \tag{A.26}$$

by Stein's lemma. Thus, $\left\|\mathbb{E}\left[\tilde{Z}\right]\right\| \lesssim B\left\|V^\top \boldsymbol{x}_i\right\|^3$. Since $\left\|VV^\top - UU^\top\right\| \le 1/4$, we have $\left\|VV^\top \boldsymbol{x}_i\right\| = O(1)$.

Therefor, we can apply Lemma A.2 and have

$$\left\|\frac{1}{m}\sum_{j=1}^{m}\tilde{Z}_j - \mathbb{E}\tilde{Z}\right\| \le \tilde{O}(\frac{B^{5/2}}{\sqrt{m}}). \tag{A.27}$$

with probability $1 - \frac{\delta}{2}$.

Finally, putting Eq. (A.22) and Eq. (A.27) together:

$$\|\bar{T}(V,V,V) - T(V,V,V)\| \le \tilde{O}(\frac{B^{5/2}}{\sqrt{m}}) \tag{A.28}$$

with probability $1 - \delta$.

Remark 5. *The bound in Proposition 5.6 can be improved if we use finer estimations, e.g. methods similar to the proof of Proposition 5.5. However, the bound is always $\tilde{O}(\mathrm{poly}(B))$ since the dimension of $V^\top \boldsymbol{w}$ is B so it is not a significant improvement.*

A.4 Proof for Two-layer Neural Networks

The poof for Theorem 5.1 directly applies Proposition 5.3 and Remark 2. The Proposition 5.5 assigns $\mu = \tilde{O}(B\sqrt{\frac{d}{m}})$ and Proposition 5.6 assigns $\gamma = \tilde{O}(\frac{B^{5/2}}{\sqrt{m}})$. Claim 5.4 helps determine that κ is constant. We have also demonstrated how to work with small r_i^* in Remark 1. After plugging in the values used in Theorem 5.3 we natually get the results shown in Theorem 5.1.

B ALTERNATIVE METHOD

In this section, we will introduce an alternative method to compute the estimated tensor $\hat{\boldsymbol{T}}$. We denote $\hat{g}_j$ as:

$$\hat{g}_j := \nabla_{w_j} L(\Theta) = \sum_{i=1}^{B} r_i a_j \sigma'\left(\boldsymbol{w}_j^\top \boldsymbol{x}_i\right)\boldsymbol{x}_i.$$

Let $a_j = \frac{1}{m}, \forall j \in [m]$ and $\boldsymbol{w}_j \in \mathcal{N}(0,1)$, by Stein's lemma, we have

$$\begin{aligned}\boldsymbol{T}_1 &:= \sum_{j=1}^{m}\hat{g}_j H_2(\boldsymbol{w}_j) = \frac{1}{m}\sum_{i=1}^{B} r_i^* \boldsymbol{x}_i \otimes \left[\sum_{j=1}^{m}\sigma'\left(\boldsymbol{w}_j^\top \boldsymbol{x}_i\right)\left(\boldsymbol{w}_j \otimes \boldsymbol{w}_j - I\right)\right] \\ &\approx \sum_{i=1}^{B} r_i^* \boldsymbol{x}_i \otimes \mathbb{E}\left[\sigma'\left(\boldsymbol{w}_j^\top \boldsymbol{x}_i\right)\left(\boldsymbol{w}_j \otimes \boldsymbol{w}_j - I\right)\right] \\ &= \sum_{i=1}^{B} r_i^* \mathbb{E}\left[\sigma^{(3)}(\boldsymbol{w}^\top \boldsymbol{x}_i)\right]\boldsymbol{x}_i^{\otimes 3} = \boldsymbol{T}.\end{aligned}$$

Let $\boldsymbol{T}_2$ and $\boldsymbol{T}_3$ be tensors such that $\boldsymbol{T}_2(i,j,k) = \boldsymbol{T}_1(k,i,j)$ and $\boldsymbol{T}_3(i,j,k) = \boldsymbol{T}_1(j,k,i)$. Then $\hat{\boldsymbol{T}} := \frac{\boldsymbol{T}_1+\boldsymbol{T}_2+\boldsymbol{T}_3}{3} \approx \boldsymbol{T}$ and is symmetric. As shown in Section 6, estimating $\boldsymbol{T}$ by gradient with respect to W is empirically better than original method. Therefore, we use this estimation of $\boldsymbol{T}$ in all experiments (except for the experiment comparing two methods).

C EXPERIMENT DETAILS

In our experiments in Section 6, we reconstruct training data from two-layer neural networks with width $m = 5000$. The synthetic training data is $\boldsymbol{x}_i = \boldsymbol{e}_i, i = 1,2$ with batch size $B = 2$ and labels $y_1 = 1, y_2 = -1$. A bias term $M = 30$ is added to r_i in experiments following Remark 1. We run Algorithm 1 with tensor method in (Zhong et al., 2017). For noisy tensor decomposition conducted on $\boldsymbol{T}(V,V,V)$, the number of random projections of Algorithm 1 in (Kuleshov et al., 2015) is fixed as $L = 100$.

D ADDITIONAL EXPERIMENT RESULTS

In this section , we will present some additional empirical results on real data. We conduct Algorithm 1 to reconstruct MNIST images from a binary classification problem with square loss. For simplicity, we set batch size $B = 2$ and activation function $\sigma(x) = x^2 + x^3$. Different network width m are used in this experiment, when data dimension d is fixed as 784. We use other settings same as the experiments on synthetic data. The results here are the reconstruction images where the two images in the batch have same or different labels (Fig. 3).

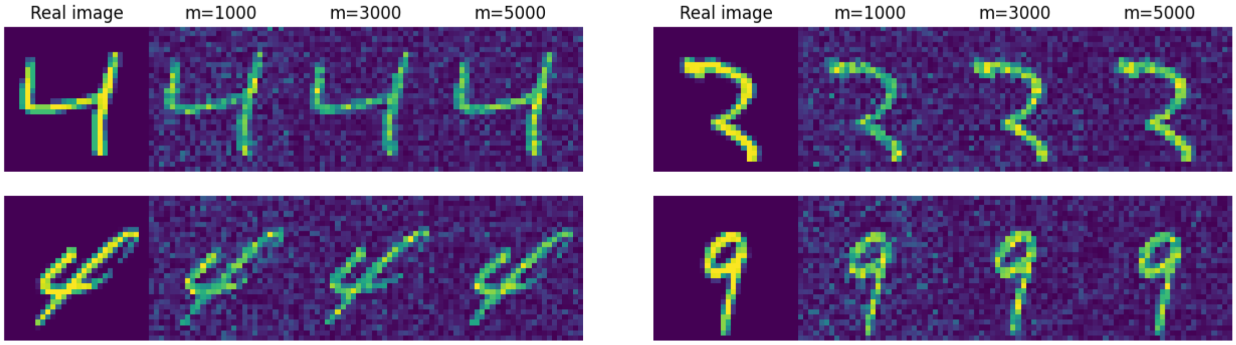

Figure 3: Reconstruction images of MNIST with different network width m. **Left:** images with a same label; **Right:** images with different labels.