WHAT CAN WE LEARN FROM GRADIENTS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work (Zhu, Liu, and Han 2019) has shown that it is possible to reconstruct the input (image) from the gradient of a neural network. In this paper, our aim is to better understand the limits to reconstruction and to speed up image reconstruction by imposing prior image information and improved initialization. Exploring the theoretical limits of input reconstruction, we show that a fully-connected neural network with a single hidden node is enough to reconstruct a single input image, regardless of the number of nodes in the output layer. Then we generalize this result to a gradient averaged over mini-batches of size B. In this case, the full mini-batch can be reconstructed in a fully-connected network if the number of hidden units exceeds B. For a convolutional neural network, the required number of filters in the first convolutional layer again is decided by the batch size B, however, in this case, input width d and the width after filter $d^{'}$ also play the role $h = (\frac{d}{d'})^2 BC$, where C is channel number of input. Finally, we validate and underpin our theoretical analysis on bio-medical data (fMRI, ECG signals, and cell images) and on benchmark data (MNIST, CIFAR100, and face images).

1 MOTIVATION

Privacy and security are major concerns in Federated Learning (FL, Konečnỳ et al. (2016)), e.g., the member participation attack Kairouz et al. (2019) revealing the presence of certain data. An important general defence is differential privacy Dwork (2008). For a relaxed differential privacy mechanism, (ϵ, δ), the information leakage is quantified as $\exp(\epsilon)$ under the bounded leakage probability δ. Another category consists of adversarial attacks that attempt to prevent a model from being learned at all (Bhagoji et al., 2019; Sun et al., 2019; Hitaj et al., 2017). Typically, the attacker employs an additive GAN model to learn a generative model with the gradient updates collected from the distributed devices, even if the attacker has no access to input data.

Recent work (Zhu et al., 2019) showed that an attacker may reconstruct the training data in the FL environment. In particular, the attacker does not require much auxiliary information; basically they can pretend to be an honest server or participant (with only knowledge of the gradient update and the model parameters).

Investigating the limits to this highly interesting demonstration we propose to invoke image prior information and a new initialization mechanism the primary aim is to speed up the convergence rate of reconstruction, while the secondary aim is to also increases the numerical stability. Moreover, we offer a theoretical analysis of the limitations of the reconstruction for a fully-connected neural network and vanilla convolutional neural network. Our main contributions are:

- We introduce prior image knowledge and a new initialization to increase the convergence rate and successful reconstruction.
- We show that a fully-connected neural network only needs **a single** node in **one** hidden layer to reconstruct a single input image, regardless of the number of nodes in the output layer, as long as the output layer has bias term. We generalize the result to mini-batch reconstruction and show that **the number of hidden units has to exceed the size of the mini-batch**.
- For a convolutional neural network, the number of filters in the first convolutional layer decides whether reconstruction is possible, namely, the number of filters should be such that size of output after passing through the first convolutional layer exceeds the size of original input.

Algorithm 1 Average Federated Learning

1: **Initialization:** w^0
2: **for** t=1,...T **do**
3: =>**devices:**
4: **for** j=1,2,..,p **do** p devices (in Parallel)
5: $v_j^t = \nabla \hat{\ell}_j(f(\{x_{jk}^t\}, w^t), \{y_{jk}^t\})$
6: with $(\{x_{jk}^t\}_{k=1,..K}, \{y_{jk}^t\}_{k=1,...,K}) \sim \hat{\mathcal{D}}$
7: share v_j^t with server
8: **end**
9: =>**server:**
10: $w^{t+1} = w^t - \eta \times \frac{1}{p} \sum_{j=1}^{p} v_j^t$
11: share w^{t+1} with devices for next round
12: **end**
13: **end**

2 FEDERATED LEARNING

In the FL setup, we have a server and a collection of distributed devices jointly training a global model with the knowledge of the combined environment. The collection of devices own their local data, and based on which, update the gradients with the server. Thus, the global empirical loss function $\hat{\ell}(.)$ can be approximated by average of local loss functions $\hat{\ell}_i$, shown in equation (1). Note that here we consider the data is independent and identically distributed across all the devices, which is not necessary for practice. We define $m = \sum_{i=1}^{p} m_i$ where m_i is the amount of data on device i.

$$\begin{aligned} \hat{\ell}(X) &= \sum_{i=1}^{p} \frac{m_i}{m} \mathbb{E}_{(x,y)\sim D}[\hat{\ell}_i(f_w(x), y)] \\ &= \frac{1}{p} \sum_{i=1}^{p} \mathbb{E}_{(x,y)\sim D}[\hat{\ell}_i(f_w(x), y)](m_i = m_j, \forall i, j) \end{aligned} \tag{1}$$

Each iteration consists in two stages: local training and aggregation. More specifically, after distributed devices have completed local training, they share the corresponding gradients with the server, and the server aggregates the gradients based on the given criterion, updates the global model, and finally it sends the updated global model back to the devices. The devices will use this model for the next iteration. The procedure is depicted in Algorithm 1. We assume that we have p active devices participating in every round, and all the devices share the same batch size K, which induces the so-called Average Federated Learning (average gradients update, Line 10 in Algorithm 1). Data of all devices is assumed to be generated by the same underlying data distribution $\mathcal{D}$, denoted the empirical distribution $\hat{\mathcal{D}}$.

3 RELATED WORK

In the FL setting, a generative adversarial network (GAN)-based attack has been proposed as a means to adversely train the global model. Wang et al. (2019) employs a multi-task GAN whose discriminator is trained by the gradient update from the victim, interestingly with no direct access to the data. Its discriminator simultaneously discriminates for multiple tasks: category, reality, and client identity of input samples. Melis et al. (2019) shows that it is possible to have the membership attack and also infer properties that hold only for a subset of the training data. Another work (Hitaj et al., 2017) assumes that one participant is an attacker, who owns a discriminator with the same architecture as the classifier. Moreover, the attacker generates the fake images and adversely update the global model to force the victim to reveal more information. All Gan-based attacks are notorious for increasing the complexity of training the model and partially disclosing the data distribution. Most of the attacks mentioned beforehand are white-box attacks, in which attackers have access to

the complete model description. For black-box attack, the attackers typically ask for the prediction or query. For instance, in one work (Fredrikson et al., 2014), the attacker uses black-box access to the model to infer a sensitive feature x_i. More precisely, given joint distribution and marginal priors they employ maximum a posterior (MAP) estimator that may sample a promising x_i that maximizes the posterior probability. The current state-of-the-art of reconstruction is Zhu et al. (2019), their reconstruction is more accurate than the GAN-based attacks. Zhao et al. (2020) and Geiping et al. (2020) are extensions, and aim to improve the label prediction accuracy and the reconstruction loss function.

4 INSPIRATION

We will formally describe the reconstruction attack based on gradients in the section. Let's define a neural network model f, parameterized by W, and with input $x \in R^d$. Supposed there is a function G that maps from x to a gradient vector v ($|v| = p$), $G : \mathbb{R}^{B \times d} \mapsto \mathbb{R}^p$ ($B = 1$, if SGD). For mini-batch, the output of the gradient function is the average gradient $\overline{v} = \frac{1}{B} \sum_{i=1}^{B} G(x_i, y_i; w)$.

Essentially, reconstruction is the procedure of computing an inverse function G^{-1}, which takes the input $\hat{v}$ (hopefully approximates to v) and output $\hat{x}$ sufficiently close to the original input x (at least to the extent that the attacker may infer useful information from the reconstructed one). For instance, if x is image that belongs to animal category, the attacker may know the properties of the subject, e.g., with tail or not.

Zhu et al. (2019) generate dummy input $\hat{x}$ (initialization) by sampling it from a multivariate normal distribution $\mathcal{N}(.)$ with mean $\vec{0} \in \mathbb{R}^d$ and variance $\mathbb{1} \in \mathbb{R}^{d \times d}$. Consecutively, they minimize the distance L ($L := \|v - \hat{v}\|_2^2$) between original gradient v and dummy gradient $\hat{v}$ to reconstruct the original input.

5 RECONSTRUCTION WITH PRIOR KNOWLEDGE

We propose to stabilize the optimization step by two modifications relative to Zhu et al. (2019). First, we sample the dummy (initial) data from an uniform distribution on the unit interval $x \sim \mathbb{U}(\mathbb{0}, \mathbb{1})$ since typically the preprocessing operation re-scale image data into the range between zero and one. Second, we expand the cost function with L2 regularizer $\lambda \|\hat{x}\|_F^2$ for single instance reconstruction and an orthogonal regularizer $\lambda \sum_{k \neq k'=1}^{n} (\hat{x}_k^\intercal \hat{x}_{k'})^2$ for mini-batch reconstruction. Thus, we have two following equations (2) and (3).

$$\min_{\hat{x}, \hat{y}} \|v - G(\hat{x}, \hat{y}; w)\|_2^2 + \lambda \|\hat{x}\|_F^2 \tag{2}$$

$$\min_{\hat{x}, \hat{y}} \|v - G(\hat{x}, \hat{y}; w)\|_2^2 + \lambda \sum_{k \neq k'=1}^{B} (\hat{x}_k^\intercal \hat{x}_{k'})^2 \tag{3}$$

Our reconstruction function is specified in Algorithm 2. We dynamically decreases λ after m iterations. Our method mainly has three advantages: 1) Due to the appropriate initialization, it starts from a promising position (in the solution space) close to the optima x^*. 2) For the single instance reconstruction, L2 as our regularizer forces $\|\hat{x}\|_F^2$ small (in particular in the early stage of iterations), which may avoid the gradient saturation for activation function like sigmoid and also prevent it from going too far from the possible zone ([0,1]) during the iterative optimization, increasing numerical stability. 3) For the mini-batch reconstruction, the orthogonal regularizer may penalize the similarities between reconstructed images, in particular, in the early optimization stage.

5.1 ARCHITECTURE, ACTIVATION FUNCTION AND COST FUNCTION

The neural network architecture, the choice of activation function, and cost function have a different impact on the gradient-based reconstruction.

Some activation functions increase the difficulty of reconstruction, e.g. ReLu, when input is less than zero, those parts of information will lose. However, activation function as sigmoid $\sigma(x) =$

Algorithm 2 *Our Reconstruction (with improved initialization and regularizer)*

Input: v, w
for i=0,1,2,..,I **do** I iterations
  if i==0 **then**
    $\hat{x}_0, \hat{y}_0 \sim \mathbb{U}(\mathbb{0}, \mathbb{1})$
    $\hat{v}_1 = G(\hat{x}_0, \hat{y}_0; w)$
  else
    $\hat{v}_i = G(\hat{x}_i, \hat{y}_i; w)$
    if single reconstruction **then**
      $\min_{\hat{x}_i, \hat{y}_i} \|v - G(\hat{x}_i, \hat{y}_i; w)\|_2^2 + \lambda \|\hat{x}_i\|_F$
    else
      $\min_{\hat{x}_i, \hat{y}_i} \|v - G(\hat{x}_i, \hat{y}_i; w)\|_2^2 + \lambda \sum_{k \neq k'=1}^{n} (\hat{x}_k^\intercal \hat{x}_{k'})^2$
    end
    update: $\hat{x}_{i+1} = \hat{x}_i - \eta \times \nabla L_{\hat{x}_i}$
      $\hat{y}_{i+1} = \hat{y}_i - \eta \times \nabla L_{\hat{y}_i}$
  if (i//m)==0 **then**
    $\lambda = 0.9 * \lambda$
  end
end

$\frac{1}{1+e^{-x}}$, $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ may exactly have the inverse function forms. We choose sigmoid as the activation function in our neural network.

Proposition 1 (FULLY-CONNECTED NEURAL NETWORK ONE-INSTANCE RECONSTRUCTION)**.** *To reconstruct one input based on a fully-connected neural network, we only need **single** node in **one** hidden layer regardless of the number of nodes in the output layer, as long as bias term exists.*

For the classification task, we often use CrossEntropy as the cost function $\ell(p_i, y_i) = -\sum_j y_{ij} \log p_{ij}$ where p_i is a vector with length equal to the number of classes, and softmax function is defined as $p_{ij} = \frac{e^{a_j}}{\sum_k e^{a_k}}$. The partial gradient with respect to a_j^L (linear product of node j in output layer L) is $\frac{\partial}{\partial a_j^L} \ell(p_i, y_i) = p_{ij} - y_{ij}$.

Say we have one hidden layer neural network $p_j = \sum_{i=1}^{n_1} w_{ji}^2 \sigma(w_i^1 x + b_i^1) + b_j^2$, where $w^1 \in \mathbb{R}^{n_1 * d}, b^1 \in \mathbb{R}^{n_1}$ are the weights and bias in the hidden layer, and n_1 is the number of nodes in the first hidden layer, and $w^2 \in \mathbb{R}^{n_2 * n_1}, b^2 \in \mathbb{R}^{n_2}$ are in the output layer instead. w_{ij}^l indicates the weights of node i sitting in layer l, connecting with node j in the previous layer $l-1$. σ is monotonic activation function. We use CrossEntropy as the cost function $\ell(.)$, thus, we have $\frac{\partial \ell}{\partial a_j^L} = p_{.j} - y_{.j}$ and the following partial derivatives:

$$\frac{\partial \ell}{\partial b_j^2} = p_{.j} - y_{.j} \tag{4}$$

$$\frac{\partial \ell}{\partial w_{ji}^2} = (p_{.j} - y_{.j}) \sigma(w_i^1 x + b_i^1) = (p_{.j} - y_{.j}) \sigma_i \tag{5}$$

$$\frac{\partial \ell}{\partial b_j^1} = \sum_i^{n_2} (p_{.i} - y_{.i}) w_{ij}^2 \sigma^{'}(w_j^1 x + b_j^1) = \sum_i^{n_2} (p_{.i} - y_{.i}) w_{ij}^2 \sigma_j^{'} \tag{6}$$

$$\frac{\partial \ell}{\partial w_j^1} = \sum_i^{n_2} (p_{.i} - y_{.i}) w_{ij}^2 \sigma_j^{'} x \tag{7}$$

If SGD, x is a vector ($x \in \mathbb{R}^{1*d}$), otherwise, x is a matrix. We know all the partial derivatives and model parameters, we can directly backtrack x. More specifically, from eq. (7) we may have the analytical form of x. Assume only one node in hidden layer ($n_1 = 1$), for SGD ($x \in \mathbb{R}^d$) we can directly have $x_m = \frac{\partial \ell}{\partial w_{1m}^1} / \frac{\partial \ell}{\partial b_1^1}, \forall m \in [1, d]$. This conclusion (proposition 1) conforms to the observation of Geiping et al. (2020).

Proposition 2 (FULLY-CONNECTED NEURAL NETWORK MINI-BATCH RECONSTRUCTION)**.** *For the mini-batch reconstruction, with the assumption that given sigmoid $\sigma(x)$, we may compute the unique $\sigma^{'}(x)$, we derive the condition that the number of units in hidden layer $n_1 \geq \frac{B(d+n_2)-n_2}{d+n_2+1-B}$. For the high-dimensional input, whose dimension $d \gg n_2, d \gg B$, n_1 is dominated by $\frac{B(d+n_2)}{d+n_2}$, thus $n_1 \geq B$.*

To solve $x \in \mathbb{R}^{B*d}$, we need equations equal or greater than unknowns. From eq. (4) to eq. (7), we have $n_2 + n_1 n_2 + n_1 + n_1 d \geq B n_2 + n_1 B + Bd$, thus $n_1 \geq \frac{B(d+n_2)-n_2}{d+n_2+1-B}$. Please see the appendix A.1, we offer the complete derive of a tiny sample.

To implement it, the choice of optimization method is relatively sensitive. For a high-dimension (nonsparse) input, second-order or quasi-second-order methods sometimes fails since it is designed to search for the zero-gradient position. The number of saddle points exponentially increases with the number of input dimensions (Dauphin et al., 2014). First-order method, e.g., Adam takes much more iterations to converge, but it is relatively more stable, likely to escape from the saddle points (Goodfellow et al., 2016).

Proposition 3 (SINGLE LAYER CONVOLUTIONAL NEURAL NETWORK RECONSTRUCTION)**.** *For a single-convolutional-layer neural network (without pooling) with kernel size k, padding size p and stride size s, immediately stacked by a fully-connected layer, known with n_2 units. To reconstruct, $h = (\frac{d^2}{d^{'}})^2 BC$ filters are required, where C is the channel number of input, B is batch size, d is the width of input, and $d^{'}$ is width after passing one filter.*

Say we have input x with both width and length equal to d, $x \in \mathbb{R}^{B*C*d*d}$, where C is the channels and B is mini-batch size. After padding, it is labeled as $\hat{x}$, and its width is equal to $d + 2p$ with padding size p. We define a square kernel with width k, stride size s, bias term b, and we have h kernels. After passing through filters, we have output z with width equal to $d^{'}$. Then, after passing filter m (before vectorization operation), we have:

$$z_{\underline{m}ij} = (\sum_{c=1}^{C}\sum_{g=1}^{k}\sum_{n=1}^{k} w^1_{\underline{m}cgh}\hat{x}_{c,si+g-1,sj+n-1}) + b^1_{\underline{m}} \quad \forall (i,j) \in [1, d^{'}] \times [1, d^{'}], \forall \underline{m} \in [1, h] \quad (8)$$

The we define $H = [\text{vec}(z_{\underline{1}..}), \text{vec}(z_{\underline{2}..}), ..., \text{vec}(z_{\underline{h}..})]$ and $|H| = h(d^{'})^2 = n_1$, after convolutional layer we have $p_j = \sum_{i=1}^{n_1} w^2_{ji}\sigma(H_i) + b^2_j$, where $w^2 \in \mathbb{R}^{n_2 * n_1}$ and $b^2 \in \mathbb{R}^{n_2}$ are the weights and bias in the output layer, $\sigma()$ is the sigmoid function as defined before. $d^{'}$ is defined as $d^{'} = \frac{d-k+2p}{s} + 1$. Thus,

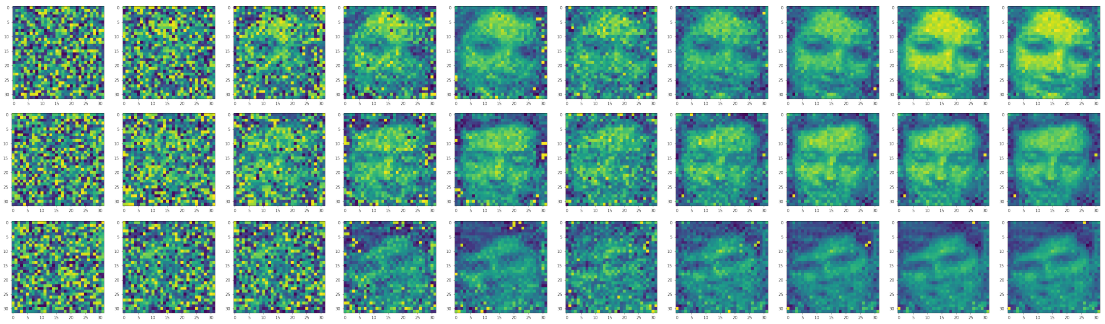
$$\frac{\partial \ell}{\partial b^2_j} = p_{.j} - y_{.j} \quad \forall j \in [1, n_2] \quad (9)$$

$$\frac{\partial \ell}{\partial w^2_{ji}} = (p_{.j} - y_{.j})\sigma(H_i) \quad \forall j \in [1, n_2], \forall i \in [1, n_1] \quad (10)$$

$$\frac{\partial \ell}{\partial b^1_{\underline{m}}} = \sum_{i=1}^{d^{'}}\sum_{j=1}^{d^{'}} \frac{\partial \ell}{\partial z_{\underline{m}ij}} \frac{\partial z_{\underline{m}ij}}{\partial b^1_{\underline{m}}} = \sum_{i=1}^{d^{'}}\sum_{j=1}^{d^{'}} \frac{\partial \ell}{\partial z_{\underline{m}ij}} \quad \forall \underline{m} \in [1, h] \quad (11)$$

$$\frac{\partial \ell}{\partial w^1_{\underline{m}cgh}} = \sum_{i=1}^{d^{'}}\sum_{j=1}^{d^{'}} \frac{\partial \ell}{\partial z_{\underline{m}ij}} \hat{x}_{c,si+g-1,sj+h-1} \quad \forall \underline{m} \in [1, h] \quad (12)$$

Note here, $\frac{\partial \ell}{\partial z_{\underline{m}ij}} = \frac{\partial \ell}{\partial H[(m-1)*(d^{'})^2 + (i-1)*d^{'} + j]}$. From eq. (8), we have $(d^{'})^2 h$ equations and $d^2 BC$ unknowns, thus we need $h = (\frac{d^2}{d^{'}})^2 BC$, with the assumption that H is known (left hand side of eq. 8). As shown in eq. (9) and (10), if $n_1 \geq \frac{n_2(B-1)}{n_2 - B}$, and $1 < B < n_2$, all $\sigma(H_i)$ are known, thus H_i are also known since $\sigma()$ is monotonic activation function.



(a) 10 (b) 30 (c) 50 (d) 90 (e) final (f) 10(our) (g) 30(our) (h) 50(our) (i) 90(our) (j) fi.(our)

Figure 3: *Face* (Batch reconstruction): Mini-batch contains 5 images, and we show partial reconstruction for 10 (col.1), 30 (col.2), 50(col.3), 90 (col.4) and final (col.5) iteration for Zhu et al. (2019). The rest columns (f-j) are produced by our method. Only three instances are shown due to the length limitations.

6 Experimental Results

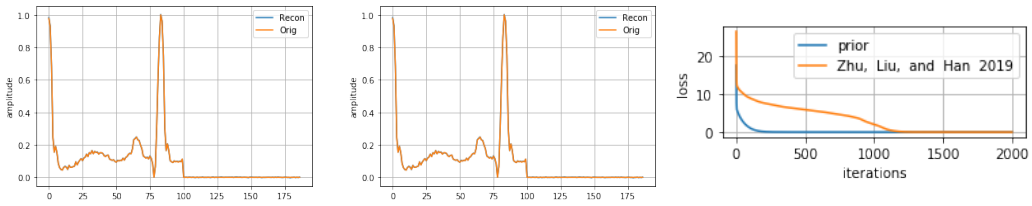
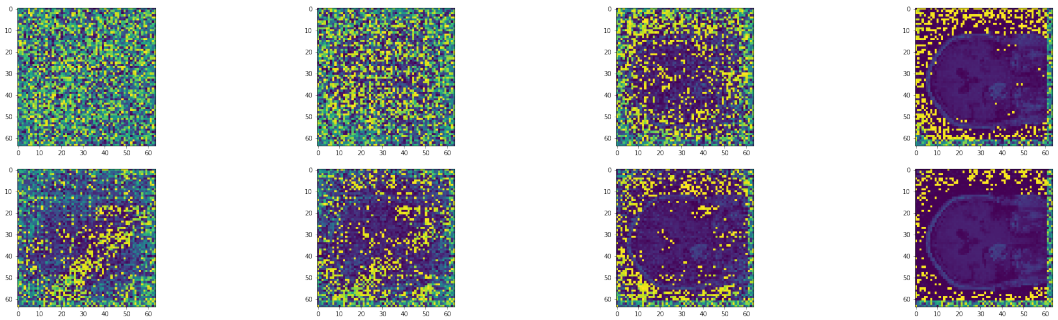


Figure 1: *ECG*: Leftmost one is final reconstruction by our method, whereas the middle one is produced by Zhu et al. (2019). Rightmost plot shows that our method converges much faster than Zhu et al. (2019).



(a) 20 itrs (b) 40 itrs (c) 80 itrs (d) final

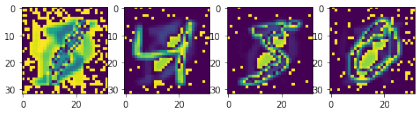
Figure 2: *FMRI*: The plots (first row) are produced by Zhu et al. (2019), and it shows the reconstructions after 20, 40, 80, and final iteration accordingly, whereas the second row is our reconstruction.

6.1 COMPARISON WITH BASELINE

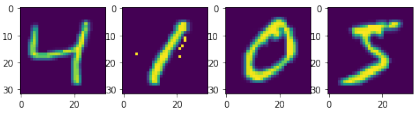
Our reconstruction enables a faster convergence rate and a more stable reconstruction procedure comparing with the method proposed by Zhu et al. (2019). We demonstrate the superiority of our method on ecg dataset, face dataset, fmri dataset, cell dataset, and cifar100 dataset in Figure 1, 2, 3, 8(appendix) and 9(appendix). We plot the reconstruction procedure with the optimizer LGBFS the same as Zhu et al. (2019) did for the fair comparison, and it clearly shows that our method converges faster on all the dataset. Moreover, due to the improved initialization and regularizer, sometimes our reconstruction turns out the lower error than Zhu et al. (2019), one instance is shown in Figure 8(appendix).

6.2 MINI-BATCH RECONSTRUCTION

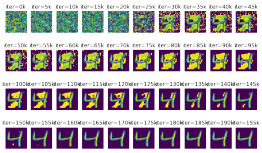
Fully-connected neural network batch reconstruction: We experiment on mnist (size: 1024) to validate our analysis; as long as the number of nodes in the hidden layer exceeds batch size, we may have the reconstruction given sufficient iterations. Here we set batch size equal to four and number of units in hidden layer to four as well. In Figure 4a, we first give the final reconstruction of four inputs without regularizer, the digits overlap together. In Figure 4b, the performance with orthogonal regularizer improves significantly. Furthermore, from Figure 4c to 4f, we show the whole reconstruction procedure every 5k iterations, and we can see how the orthogonal regularizer plays the key role during optimization procedure to hinder the similarities between instances. Moreover, in Figure 5a, we show a mini-batch reconstruction with batch size 8. It demonstrates that L1 distance between original input and reconstructed one decreases drastically with 8 units in hidden layer, afterwards even with more units, the distance does not improve too much.



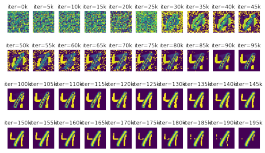
(a) No regularizer



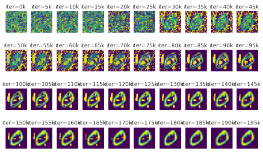
(b) With orthogonal regularizer ($\lambda = 0.1$)



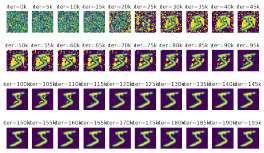
(c) 1/4 reconstruction procedure



(d) 2/4 reconstruction procedure



(e) 3/4 reconstruction procedure



(f) 4/4 reconstruction procedure

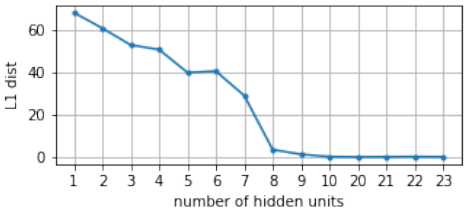
Figure 4: *MNIST* (Batch reconstruction): 4a shows the final reconstruction without any regularizer; as a comparison, with regularizer is shown in 4b. Moreover, the full reconstructions of 4b are shown in 4c, 4d, 4e and 4f. We plot every 5k iterations.

Cnn batch reconstruction: We test on the cifar100 dataset, with kernel size 5, padding size 2 and stride size2. According to **Proposition 3**, we need 12 filters to reconstruct one instance. In Figure 6, we visually show the reconstruction performance with the change of filters number and from Table 6f, we show L1 distance alters with filter number. Note here we set λ starts from 0.1 and gradually decays 90% after 2k iterations. For the mini-batch reconstruction (shown in Figure 5b), we set batch size equal to four, thus 48 filters are required, we compared filter number equal to 12, 24, 48 and 50, as we can see the L1 distance keeps decreasing and the reconstruction is in Figure 7. In particular, when filter number is 48, average L1 distance per pixel is 0.00025.

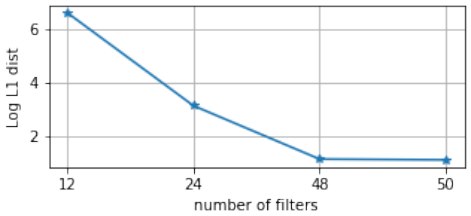
7 CONCLUSION

We have investigated the highly interesting results in Zhu et al. (2019) to understand if the results can be improved by careful initialization and augmenting the cost function with regularizers. More specifically, we suggest an uniform distribution ([0,1]) initialization since typically the preprocessing re-scales it between zero and one. We propose a L2 regularizer for single image reconstruction, and an orthogonality promoting regularizer for mini-batch reconstruction. In both cases we let the

strength λ decay during iterations, to reduce bias in the final solutions. Then, we explore the limitation of reconstruction and analyze the correlation between architecture, network size, and reconstruction performance. We show that a fully-connected neural network only needs a single node in one hidden layer to reconstruct a single input image, regardless of the number of nodes in the output layer, as long as bias term exists. For the mini-batch case, as long as the number of nodes in the first hidden layer exceeds the batch size (with regularizer), the reconstruction is promising. For the convolutional neural network, $(\frac{d}{d'})^2C$ filters are required for the reconstruction, where d is the width of input image, d' is the width after one convolutional filter , and C is input channel number. It can be generalized to mini-batch reconstruction as well, thus $BC * (\frac{d}{d'})^2$ filters are required (B is batch size).

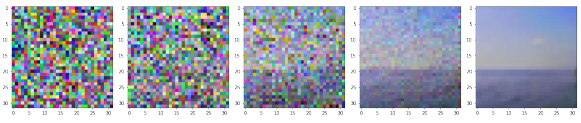


(a) MNIST reconstruction with batch size 8, when number of units in hidden layer surpass the batch size, the distance between reconstructed image and original one drastically decreases.



(b) CIFAR100 reconstruction with batch size 4, kernel size 5, padding size 2, stride size 2. According to proposition 3, 48 filters are required. Reconstruction is shown in Figure 7.

Figure 5: L1 distance and number of units and filters

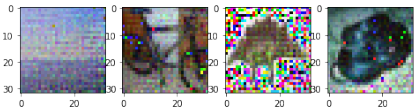


(a) 1 filter (b) 5 filters (c) 8 filters (d) 11 filters(e) 12 filters

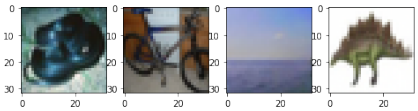
Filters	**Params.** $((b')^2h)$	**L1 dist.**
1	64	728
5	1280	649
8	2048	404
11	2816	135
12	3072	0.57

(f) L1 distance

Figure 6: *One-layer CNN*: Image reconstruction becomes more accurate with increasing number of filters (kernels). Numerical comparison sees Table 6f. One-layer CNN: Distance decreases with increasing number of filters (kernels). First row is the number of filters, and second row is the size of filtered images, which is decided by the number of filters with the fixed kernel, padding and stride size. When the size of filtered image is exactly equal to original input size (12 filters), we can almost perfectly reconstruct it (L1 distance is 0.09, with λ starts from 0.1)



(a) 12 filter



(b) 48 filters

Figure 7: CIFAR100 reconstruction with batch size 4 with 12 and 48 filters. λ starts from 0.1, and gradually decays.

REFERENCES

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pp. 634–

643, 2019.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pp. 2933–2941, 2014.

Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pp. 1–19. Springer, 2008.

Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pp. 17–32, 2014.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? *arXiv preprint arXiv:2003.14053*, 2020.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 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.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 691–706. IEEE, 2019.

Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 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.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, pp. 14774–14784, 2019.

A Appendix

A.1 Fully-connected neural network

Say we have n_1, n_2 nodes in hidden and output layer accordingly, and batch size is B. A one-hidden-layer neural network is defined as $p_j = \sum_{i=1}^{n_1} w_{ji}^2 \sigma(w_i^1 x + b_i^1) + b_j^2$, where $\sigma()$ is a monotonic active function, and σ_{ij} indicates $\sigma(w_i x_j + b_i)$ in the current layer, and j is the index for instance x_j. Our cost function E is cross entropy.

$$\frac{\partial \ell}{\partial b_j^2} = f_{.j} - t_{.j} \tag{13}$$

$$\frac{\partial \ell}{\partial w_{ji}^2} = (f_{.j} - t_{.j})\sigma(w_i^1 x + b_i^1) = (f_{.j} - t_{.j})\sigma_i \tag{14}$$

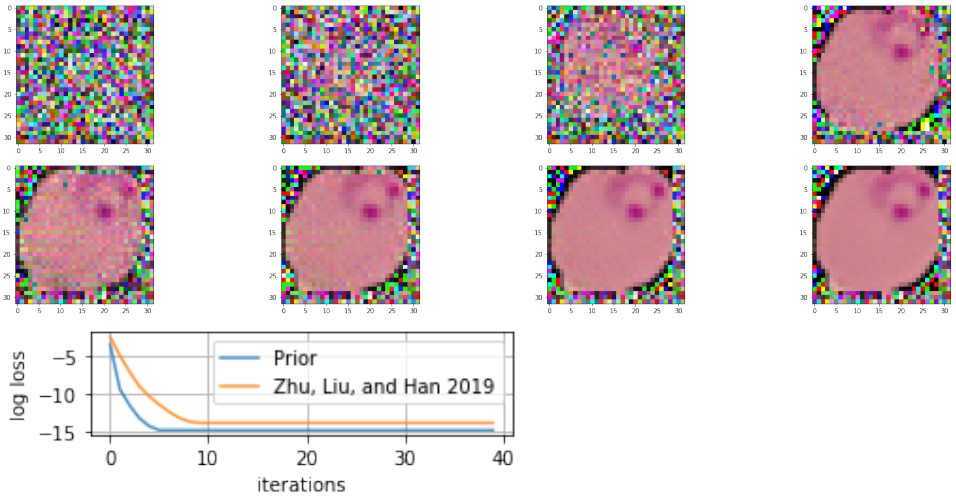


Figure 8: *CELL*: The first row corresponds to the reconstruction by Zhu et al. (2019), for 10, 20,30, and final iteration, whereas the second row is produced by our method.

$$\frac{\partial \ell}{\partial b_i^1} = \sum_j^{n_2} (f_{\cdot j} - t_{\cdot j}) w_{ji}^2 \sigma_i^{'} \tag{15}$$

$$\frac{\partial \ell}{\partial w_i^1} = \sum_j^{n_2} (f_{\cdot j} - t_{\cdot j}) w_{ji}^2 \sigma_i^{'} x \tag{16}$$

For the purpose of demonstration, we set batch size B equal to 2, n_1 equal to 3, n_2 equal to 2.

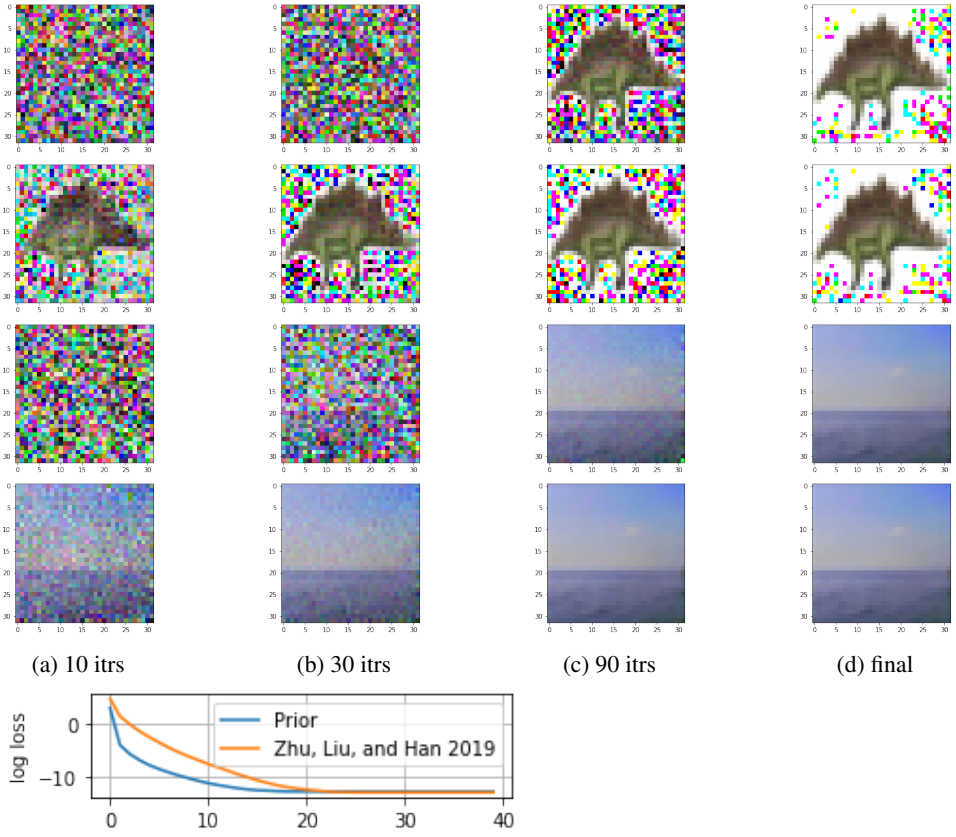
$$\text{derive from eq. (13)} \begin{cases} \frac{\partial \ell}{\partial b_1^2} &= (f_{11} - t_{11}) + (f_{21} - t_{21}) \\ \frac{\partial \ell}{\partial b_2^2} &= (f_{12} - t_{12}) + (f_{22} - t_{22}) \end{cases} \tag{17}$$

$$\text{derive from eq. (14)} \begin{cases} \frac{\partial \ell}{\partial w_{11}^2} &= (f_{11} - t_{11})\sigma_{11} + (f_{21} - t_{21})\sigma_{12} \\ \frac{\partial \ell}{\partial w_{12}^2} &= (f_{11} - t_{11})\sigma_{21} + (f_{21} - t_{21})\sigma_{22} \\ \frac{\partial \ell}{\partial w_{13}^2} &= (f_{11} - t_{11})\sigma_{31} + (f_{21} - t_{21})\sigma_{32} \\ \frac{\partial \ell}{\partial w_{21}^2} &= (f_{12} - t_{12})\sigma_{11} + (f_{22} - t_{22})\sigma_{12} \\ \frac{\partial \ell}{\partial w_{22}^2} &= (f_{12} - t_{12})\sigma_{21} + (f_{22} - t_{22})\sigma_{22} \\ \frac{\partial \ell}{\partial w_{23}^2} &= (f_{12} - t_{12})\sigma_{31} + (f_{22} - t_{22})\sigma_{32} \end{cases} \tag{18}$$

$$\text{derive from eq. (15)} \begin{cases} \frac{\partial \ell}{\partial b_1^1} = [(f_{11} - t_{11})w_{11}^2 + (f_{12} - t_{12})w_{21}^2]\sigma_{11}^{'} + [(f_{21} - t_{21})w_{11}^2 + (f_{22} - t_{22})w_{21}^2]\sigma_{12}^{'} \\ \frac{\partial \ell}{\partial b_2^1} = [(f_{11} - t_{11})w_{12}^2 + (f_{12} - t_{12})w_{22}^2]\sigma_{21}^{'} + [(f_{21} - t_{21})w_{12}^2 + (f_{22} - t_{22})w_{22}^2]\sigma_{22}^{'} \\ \frac{\partial \ell}{\partial b_3^1} = [(f_{11} - t_{11})w_{13}^2 + (f_{12} - t_{12})w_{23}^2]\sigma_{31}^{'} + [(f_{21} - t_{21})w_{13}^2 + (f_{22} - t_{22})w_{23}^2]\sigma_{32}^{'} \end{cases} \tag{19}$$

$$\text{derive from eq. (16)} \begin{cases} \frac{\partial \ell}{\partial w_1^1} = [(f_{11} - t_{11})w_{11}^2 + (f_{12} - t_{12})w_{21}^2]\sigma_{11}^{'} x_1 + [(f_{21} - t_{21})w_{11}^2 + (f_{22} - t_{22})w_{21}^2]\sigma_{12}^{'} x_2 \\ \frac{\partial \ell}{\partial w_2^1} = [(f_{11} - t_{11})w_{12}^2 + (f_{12} - t_{12})w_{22}^2]\sigma_{21}^{'} x_1 + [(f_{21} - t_{21})w_{12}^2 + (f_{22} - t_{22})w_{22}^2]\sigma_{22}^{'} x_2 \\ \frac{\partial \ell}{\partial w_3^1} = [(f_{11} - t_{11})w_{13}^2 + (f_{12} - t_{12})w_{23}^2]\sigma_{31}^{'} x_1 + [(f_{21} - t_{21})w_{13}^2 + (f_{22} - t_{22})w_{23}^2]\sigma_{32}^{'} x_2 \end{cases} \tag{20}$$

Note here we assume with known σ_{ij}, we may compute $\sigma_{ij}^{'}$. To fully solve $x_1, x_2 \in \mathbb{R}^d$, we need to meet the condition that $n_2 + n_1 n_2 + n_1 + n_1 d \geq B n_2 + n_1 B + Bd$, thus $n_1 \geq \frac{B(d+n_2)-n_2}{d+n_2+1-B}$ and $B < n_2 + 1 + d$. Note here if $d \gg B, d \gg n_2$, then n_1 is largely determined by $\frac{B(d+n_2)}{d+n_2} = B$.



(a) 10 itrs (b) 30 itrs (c) 90 itrs (d) final

Figure 9: *Cifar100*: We show partial reconstruction for 10 (col.1), 30 (col.2), 90 (col.3) and final (col.4) iteration. The odd number of rows (1,3) are produced by Zhu et al. (2019) and even number of rows (2,4) are produced by our method. The corresponding log-scale loss is shown under the reconstruction procedure.