

**Roadmap of Appendix** The Appendix is organized as follows. We discuss related work in Section A. We provide theoretical analysis in Section B. The details of data recovery experiment are in Section C and additional experiment details are in Section D.

## A RELATED WORK

### A.1 HARDNESS AND NEURAL NETWORKS

When there are no further assumptions, neural networks have been shown hard in several different perspectives. Blum & Rivest (1992) first proved that learning the neural network is NP-complete. Different variant hardness results have been developed over past decades Klivans & Sherstov (2009); Daniely (2016); Daniely & Shalev-Shwartz (2016); Goel et al. (2017); Livni et al. (2014); Weng et al. (2018); Manurangsi & Reichman (2018); Lei et al. (2019); Daniely & Vardi (2020); Huang et al. (2020b;a). The work of Lei et al. (2019) is most relevant to us. They consider the neural network inversion problem in generative models and prove the exact inversion problem is NP-complete.

### A.2 DATA SEPARABILITY AND NEURAL NETWORK TRAINING

One popular distributional assumption, in theory, is to assume the input data points to be the Gaussian distributions Zhong et al. (2017b); Li & Yuan (2017); Zhong et al. (2017a); Ge et al. (2018); Bakshi et al. (2019); Chen et al. (2020) to show the convergence of training deep neural networks. Later, convergence analysis using weaker assumptions are proposed, i.e., input data points are separable Li & Liang (2018). Following Li & Liang (2018); Allen-Zhu et al. (2019b;c;a); Zhang et al. (2020a), data separability plays a crucial role in deep learning theory, especially in showing the convergence result of over-parameterized neural network training. Denote  $\delta$  is the minimum gap between all pairs data points. Data separability theory says as long as the width ( $m$ ) of neural network is at least polynomial factor of all the parameters ( $m \geq \text{poly}(n, d, 1/\delta)$ ), i.e.,  $n$  is the number of data points,  $d$  is the dimension of data, and  $\delta$  is data separability. Another line of work Du et al. (2019); Arora et al. (2019a;b); Song & Yang (2019); Brand et al. (2020); Lee et al. (2020) builds on neural tangent kernel Jacot et al. (2018). It requires the minimum eigenvalue ( $\lambda$ ) of neural tangent kernel is lower bounded. Recent work Oymak & Soltanolkotabi (2020) finds the connection between data-separability  $\delta$  and minimum eigenvalue  $\lambda$ , i.e.  $\delta \geq \lambda/n^2$ .

### A.3 DISTRIBUTED DEEP LEARNING SYSTEM

Collaboration between the edge device and cloud server achieves higher inference speed and lowers power consumption than running the task solely on the local or remote platform. Typically there are two collaborative modes. The first is collaborative training, for which training task is distributed to multiple participants Konečný et al. (2015); Vanhaesebrouck et al. (2016); Kairouz et al. (2019). The other model is collaborative inference. In such a distributed system setting, the neural network can be divided into two parts. The first few layers of the network are stored in the local edge device, while the rest are offloaded to a remote cloud server. Given an input, the edge device calculates the output of the first few layers and sends it to the cloud. Then cloud perform the rest of computation and sends the final results to each edge device Eshratifar et al. (2019); Hauswald et al. (2014); Kang et al. (2017); Teerapittayanon et al. (2017). In our work, we focus on tackling data recovery problem under collaborative inference mode.

### A.4 MODEL INVERSION ATTACK AND DEFENSE

The neural network inversion problem has been extensively investigated in recent years Fredrikson et al. (2015); He et al. (2019); Lei et al. (2019); Zhang et al. (2020b). As used in this paper, the general approach is to cast the network inversion as an optimization problem and uses a problem specified objective. In particular, Fredrikson et al. (2015) proposes to use confidence in prediction as to the optimized objective. He et al. (2019) uses a regularized maximum likelihood estimation. Recent work Zhang et al. (2020b) also proposes to use GAN to do the model inversion.

There are very few studies about defenses against model inversion attack. Existing data privacy protection mechanisms mainly rely on noise injection Fredrikson et al. (2015); Dwork (2008); Abadi

et al. (2016) or Homomorphic Encryption Nandakumar et al. (2019). While being able to mitigate attacks, existing methods significantly hinder model performance. Recently MID Wang et al. (2020) was proposed to limit the information about the model input contained in the prediction, thereby limiting the ability of an adversary to infer data information from the model prediction. Yang et al. (2020) proposed to add a purification block following by prediction output, so that the confidence score vectors predicted by the target classifier are less sensitive to the change of input data. However, the above two methods target the logit output layer (i.e., performing argmax). They either require auxiliary information (i.e., knowing attack model) or modifying network structure (i.e., building variational autoencoder structure for mutual information calculation). In contrast, our proposed method MixConcan easily and efficiently serve as a plug-in loss to the middle layers of arbitrarily network structures to defend inversion attack during inference.

## B HARDNESS OF NEURAL NETWORK INVERSION

### B.1 PRELIMINARIES

We first provide the definitions for 3SAT, ETH, MAX3SAT, MAXE3SAT and then state some fundamental results related to those definitions. For more details, we refer the reader to the textbook Arora & Barak (2009).

**Definition B.1** (3SAT problem). *Given  $n$  variables and  $m$  clauses in a conjunctive normal form CNF formula with the size of each clause at most 3, the goal is to decide whether there exists an assignment to the  $n$  Boolean variables to make the CNF formula be satisfied.*

**Hypothesis B.2** (Exponential Time Hypothesis (ETH) Impagliazzo et al. (1998)). *There is a  $\delta > 0$  such that the 3SAT problem defined in Definition B.1 cannot be solved in  $O(2^{\delta n})$  time.*

ETH is a stronger notion than  $NP \neq P$ , and is well acceptable the computational complexity community. Over the few years, there has been work proving hardness result under ETH for theoretical computer science problems Chalermsook et al. (2017); Manurangsi (2017); Chitnis et al. (2018); Bhattacharyya et al. (2018); Dinur & Manurangsi (2018); KCS & Manurangsi (2018) and machine learning problems, e.g. matrix factorizations Arora et al. (2012); Razenshteyn et al. (2016); Song et al. (2017); Ban et al. (2019), tensor decomposition Song et al. (2019). There are also variations of ETH, e.g. Gap-ETH Dinur (2016; 2017); Manurangsi & Raghavendra (2017) and random-ETH Feige (2002); Razenshteyn et al. (2016), which are also believable in the computational complexity community.

**Definition B.3** (MAX3SAT). *Given  $n$  variables and  $m$  clauses, a conjunctive normal form CNF formula with the size of each clause at most 3, the goal is to find an assignment that satisfies the largest number of clauses.*

We use MAXE3SAT to denote the version of MAX3SAT where each clause contains exactly 3 literals.

**Theorem B.4** (Håstad (2001)). *For every  $\delta > 0$ , it is NP-hard to distinguish a satisfiable instance of MAXE3SAT from an instance where at most a  $7/8 + \delta$  fraction of the clauses can be simultaneously satisfied.*

**Theorem B.5** (Håstad (2001); Moshkovitz & Raz (2010)). *Assume ETH holds. For every  $\delta > 0$ , there is no  $2^{o(n^{1-o(1)})}$  time algorithm to distinguish a satisfiable instance of MAXE3SAT from an instance where at most a fraction  $7/8 + \delta$  of the clauses can be simultaneously satisfied.*

We use MAXE3SAT(B) to denote the restricted special case of MAX3SAT where every variable occurs in at most  $B$  clauses. Håstad Håstad (2000) proved that the problem is approximable to within a factor  $7/8 + 1/(64B)$  in polynomial time, and that it is hard to approximate within a factor  $7/8 + 1/(\log B)^{\Omega(1)}$ . In 2001, Trevisan improved the hardness result,

**Theorem B.6** (Trevisan (2001)). *Unless  $RP=NP$ , there is no polynomial time  $(7/8 + 5/\sqrt{B})$ -approximate algorithm for MAXE3SAT(B).*

**Theorem B.7** (Håstad (2001); Trevisan (2001); Moshkovitz & Raz (2010)). *Unless ETH fails, there is no  $2^{o(n^{1-o(1)})}$  time  $(7/8 + 5/\sqrt{B})$ -approximate algorithm for MAXE3SAT(B).*

## B.2 OUR RESULTS

We provide a hardness of approximation result for the neural network inversion problem. In particular, we prove unless  $\text{RP}=\text{NP}$ , there is no polynomial time that can approximately recover the input of a two-layer neural network with ReLU activation function<sup>7</sup>. Formally, consider the inversion problem

$$h(x) = z, \quad x \in [-1, 1]^d, \quad (4)$$

where  $z \in \mathbb{R}^{m_2}$  is the hidden layer representation,  $h$  is a two neural network with ReLU gates, specified as

$$h(x) = W_2\sigma(W_1x + b), \quad W_2 \in \mathbb{R}^{m_2 \times m_1}, W_1 \in \mathbb{R}^{m_1 \times d}, b \in \mathbb{R}^{m_1}$$

We want to recover the input data  $x \in [-1, 1]^d$  given hidden layer representation  $z$  and all parameters of the neural network (i.e.,  $W^{(1)}, W^{(2)}, b$ ). It is known the decision version of neural network inversion problem is NP-hard [Lei et al. \(2019\)](#). It is an open question whether approximation version is also hard. We show a stronger result which is, it is hard to give to constant approximation factor. Two notions of approximation could be consider here, one we called *solution approximation*

**Definition B.8** (Solution approximation). *Given a neural network  $h$  and hidden layer representation  $z$ , we say  $x' \in [-1, 1]^d$  is an  $\epsilon$  approximation solution for Eq. (4), if there exists  $x \in [-1, 1] \in \mathbb{R}^d$ , such that*

$$\|x - x'\|_2 \leq \epsilon\sqrt{d} \text{ and } h(x) = z.$$

Roughly speaking, solution approximation means we recovery an approximate solution. The  $\sqrt{d}$  factor in the above definition is a normalization factor and it is not essential.

One can also consider a weaker notion, which we called *function value approximation*

**Definition B.9** (Function value approximation). *Given a neural network  $h$  and hidden layer representation  $z$ , we say  $x' \in [-1, 1]^d$  is  $\epsilon$ -approximate of value to Eq. (4), if*

$$\|h(x') - y\|_2 \leq \epsilon\sqrt{m_2}.$$

Again, the  $\sqrt{m_2}$  factor is only for normalization. Suppose the neural network is  $G$ -Lipschitz continuous for constant  $G$  (which is the case in our proof), then an  $\epsilon$ -approximate solution implies  $G\epsilon$ -approximation of value. For the purpose of this paper, we focus on the second notion (i.e., function value approximation). Given our neural network is (constant)-Lipschitz continuous, this immediately implies hardness result for the first one.

Our theorem is formally stated below. In the proof, we reduce from  $\text{MAX3SAT}(B)$  and utilize [Theorem B.6](#)

**Theorem B.10.** *There exists a constant  $B > 1$ , unless  $\text{RP} = \text{NP}$ , it is hard to  $\frac{1}{60B}$ -approximate Eq. (4). Furthermore, the neural network is  $O(B)$ -Lipschitz continuous, and therefore, it is hard to find an  $\Omega(1/B^2)$  approximate solution to the neural network.*

Using the above theorem, we can see that by taking a suitable constant  $B > 1$ , the neural network inversion problem is hard to approximate within some constant factor under both definitions. In particular, we conclude

**Theorem B.11** (Formal statement of [Theorem 3.3](#)). *Assume  $\text{NP} \neq \text{RP}$ , there exists a constant  $\epsilon > 0$ , such that there is no polynomial time algorithm that is able to give an  $\epsilon$ -approximation to neural network inversion problem.*

*Proof of [Theorem B.10](#).* Given an 3SAT instance  $\phi$  with  $n$  variables and  $m$  clause, where each variable appears in at most  $B$  clauses, we construct a two layer neural network  $h_\phi$  and output representation  $z$  satisfy the following:

- **Completeness.** If  $\phi$  is satisfiable, then there exists  $x \in [0, 1]^d$  such that  $h_\phi(x) = z$ .

<sup>7</sup>We remark there is a polynomial time algorithm for one layer ReLU neural network recovery

- Soundness. For any  $x$  such that  $\|h_\phi(x) - z\|_2 \leq \frac{1}{60B}\sqrt{m_2}$ , we can recover an assignment to  $\phi$  that satisfies at least  $\left(\frac{7}{8} + \frac{5}{\sqrt{B}}\right)m$  clauses
- Lipschitz continuous. The neural network is  $O(B)$ -Lipschitz.

We set  $d = n$ ,  $m_1 = m + 200B^2n$  and  $m_2 = m + 100B^2n$ . For any  $j \in [m]$ , we use  $\phi_j$  to denote the  $j$ -th clause and use  $h_{1,j}(x)$  to denote the output of the  $j$ -th neuron in the first layer, i.e.,  $h_{1,j}(x) = \sigma(W_j^{(1)}x + b_j)$ , where  $W_j^{(1)}$  is the  $j$ -th row of  $W^{(1)}$ . For any  $i \in [n]$ , we use  $X_i$  to denote the  $i$ -th variable.

Intuitively, we use the input vector  $x \in [-1, 1]^n$  to denote the variable, and the first  $m$  neurons in the first layer to denote the  $m$  clauses. By taking

$$W_{j,i}^{(1)} = \begin{cases} 1, & X_i \in \phi_j; \\ -1, & \bar{X}_i \in \phi_j; \\ 0, & \text{otherwise.} \end{cases} \quad \text{and } b_j = -2$$

for any  $i \in [n], j \in [m]$ , and viewing  $x_i = 1$  as  $X_i$  to be false and  $x_i = -1$  as  $X_i$  to be true. One can verify that  $h_{1,j}(x) = 0$  if the clause is satisfied, and  $h_{1,j}(x) = 1$  if the clause is unsatisfied. We simply copy the value in the second layer  $h_j(x) = h_{1,j}(x)$  for  $j \in [m]$ .

For other neurons, intuitively, we make  $100B^2$  copies for each  $|x_i|$  ( $i \in n$ ) in the output layer. This can be achieved by taking

$$h_{m+(i-1)\cdot 100B^2+k}(x) = h_{m+(i-1)\cdot 100B^2+k}(x) + h_{1,m+100B^2n+(i-1)\cdot 100B^2+k}(x)$$

and set

$$h_{1,m+(i-1)\cdot 100B^2+k}(x) = \max\{x_i, 0\} \quad h_{1,m+100B^2n+(i-1)\cdot 100B^2+k}(x) = \max\{-x_i, 0\}$$

for any  $i \in [n], k \in [100B^2]$ . Finally, we set the target output as

$$z = (\underbrace{0, \dots, 0}_m, \underbrace{1, \dots, 1}_{100B^2n})$$

We are left to prove the three claims we made about the neural network  $h$  and the target output  $z$ . For the first claim, suppose  $\phi$  is satisfiable and  $X = (X_1, \dots, X_n)$  is the assignment. Then as argued before, we can simply take  $x_i = 1$  if  $X_i$  is false and  $x_i = -1$  if  $X_i$  is true. One can check that  $h(x) = z$ .

For second claim, suppose we are given  $x \in [-1, 1]^d$  such that

$$\|h(x) - z\|_2 \leq \frac{1}{60B}\sqrt{m_2}$$

We start from the simple case when  $x$  is binary, i.e.,  $x \in \{-1, 1\}^n$ . Again, by taking  $X_i$  to be true if  $x_i = -1$  and  $X_i$  to be false when  $x_i = 0$ . One can check that the number of unsatisfied clause is at most

$$\begin{aligned} \|h(x) - z\|_2^2 &\leq \frac{1}{3600B^2}m_2 \\ &= \frac{1}{3600B^2}(m + 100B^2n) \\ &\leq \frac{1}{12}m + \frac{1}{3600B^2}m \\ &\leq \frac{1}{8}m - \frac{5}{\sqrt{B}}m \end{aligned} \tag{5}$$

The third step follows from  $n \leq 3m$ , and the last step follows from  $B \geq 15000$ .

Next, we move to the general case that  $x \in [-1, 1]^d$ . We would round  $x_i$  to  $-1$  or  $+1$  based on the sign. Define  $\bar{x} \in \{-1, 1\}^n$  as

$$\bar{x}_i = \arg \min_{t \in \{-1, 1\}} |t - x_i|$$

We prove that  $\bar{x}$  induces an assignment that satisfies  $(\frac{7}{8} + \frac{5}{\sqrt{B}})m$  clauses. It suffices to prove

$$\|h(\bar{x}) - z\|_2^2 - \|h(x) - z\|_2^2 \leq \frac{3}{100}m \quad (6)$$

since this implies the number of unsatisfied clause is bounded by

$$\begin{aligned} \|h(\bar{x}) - z\|_2^2 &\leq \|h(x) - z\|_2^2 + (\|h(\bar{x}) - z\|_2^2 - \|h(x) - z\|_2^2) \\ &\leq \left(\frac{1}{12}m + \frac{1}{36B^2}m\right) + \frac{3}{100}m \\ &\leq \frac{1}{8}m - \frac{1}{5\sqrt{B}}m, \end{aligned}$$

where the second step follow from Eq. (5)(6), and the last step follows from  $B \geq 10^7$ .

We define  $\Delta_i := |\bar{x}_i - x_i| = 1 - |x_i| \in [0, 1]$  and  $T := m + 128B^2n$ . Then we have

$$\begin{aligned} \|h(\bar{x}) - z\|_2^2 - \|h(x) - z\|_2^2 &= \sum_{j=1}^T (h_j(\bar{x}) - z_j) - (h_j(x) - z_j)^2 \\ &= \sum_{j=1}^m (h_j(\bar{x}) - z_j)^2 - (h_j(x) - z_j)^2 \\ &\quad + \sum_{j=m+1}^T (h_j(\bar{x}) - z_j)^2 - (h_j(x) - z_j)^2 \\ &= \sum_{j=1}^m h_j(\bar{x})^2 - h_j(x)^2 - 100B^2 \sum_{i=1}^n \Delta_i^2 \\ &\leq 2 \sum_{j=1}^m |h_{1,j}(\bar{x}) - h_{1,j}(x)| - 100B^2 \sum_{i=1}^n \Delta_i^2 \\ &\leq 2 \sum_{j=1}^m \sum_{i \in \phi_j} \Delta_i - 100B^2 \sum_{i=1}^n \Delta_i^2 \\ &\leq 2B \sum_{i=1}^n \Delta_i - 100B^2 \sum_{i=1}^n \Delta_i^2 \\ &\leq \frac{n}{100} \\ &\leq \frac{3m}{100}. \end{aligned}$$

The third step follow from  $z_j = 0$  for  $j \in [m]$  and for  $j \in \{m+1, \dots, m+100B^2n\}$ ,  $z_j = 1$ ,  $\|h_j(\bar{x}) - z_j\| = 0$  and  $\|h_j(x) - z_j\|_2^2 = \Delta_i$  given  $j \in [m + (i-1) \cdot 100B^2 + 1, i \cdot 100B^2]$ . The fourth step follows from that  $h_j(x) = h_{1,j}(x) \in [0, 1]$  for  $j \in [m]$ . The fifth step follows from the 1-Lipschitz continuity of the ReLU. The sixth step follows from each variable appears in at most  $B$  clause. This concludes the second claim.

For the last claim, by the Lipschitz continuity of ReLU, we have for any  $x_1, x_2$

$$\begin{aligned} h(x_1) - h(x_2) &= W^{(2)}\sigma(W^{(1)}x_1 + b) - W^{(2)}\sigma(W^{(1)}x_2 + b) \\ &\leq \|W^{(2)}\| \cdot \|W^{(1)}\| \|x_1 - x_2\|_2 \end{aligned}$$

It is easy to see that

$$\|W^{(2)}\| \leq 2$$

and

$$\|W^{(2)}\| \leq \sqrt{200B^2 + 3B} \leq \sqrt{203B^2} \leq 15B,$$

where the second step follows from  $B \geq 1$ .

Thus concluding the proof. □

By assuming ETH and using Theorem B.7, we can conclude

**Corollary B.12** (Formal statement of Corollary 3.5). *Unless ETH fails, there exists a constant  $\epsilon > 0$ , such that there is no  $2^{o(n^{1-o(1)})}$  time algorithm that is able to give an  $\epsilon$ -approximation to neural network inversion problem.*

The proof is similar to Theorem B.10, we omit it here.

## C DETAILS OF DATA RECOVERY EXPERIMENTS

### C.1 INVERSION MODEL DETAILS FOR SYNTHETIC DATASET

In synthetic experiment, a malicious attacker recover original input data  $x \in \mathbb{R}^d$  by solving the the following optimization:

$$x^* = \arg \min_{s \in \mathbb{R}^d} \|h(s) - z\|_1$$

To estimate the optimal, we run an SGD optimizer with a learning rate of 0.01 and decayed weight  $10^{-4}$  for 500 iterations. We test data recovery results on all the 200 testing samples. Namely, we solve the above optimization problems 200 times. Each time for a testing data point.

### C.2 INVERSION MODEL DETAILS FOR BENCHMARK DATASET

In benchmark experiment, a malicious attacker recover original input data  $x \in \mathbb{R}^d$  by solving the the following optimization:

$$x^* = \arg \min_{s \in \mathbb{R}^d} \|h(s) - z\|_2 + \zeta \sum_{i,j} ((s_{i+1,j} - s_{i,j})^2 + (s_{i,j+1} - s_{i,j})^2)^{1/2},$$

where  $i, j$  are the indexes of pixels in an image.

To estimate the optimal, we run an SGD optimizer with a learning rate of 10 and decayed weight  $10^{-4}$  for 500 iterations. We used a grid searching on the space of  $\zeta$ . We find that the best data recovery comes from  $\zeta = 0.01$  for SVHN dataset and  $\zeta = 10^{-5}$  for MNIST and FashionMNIST by grid search.

### C.3 QUANTITATIVE METRICS FOR IMAGE SIMILARITY MEASUREMENT

We adopt the following two known metrics to measure the similarity between  $x^*$  and  $x$ :

- Normalized structural similarity index metric (**SSIM**), a perception-based metric that considers the similarity between images in structural information, luminance and contrast. It is widely used in image and video compression research to quantify the difference between the original and compressed images. The detailed calculation can be found in Wang et al. (2004). We normalize SSIM to take value range  $[0, 1]$  (original SSIM takes value range  $[-1, 1]$ ).
- Perceptual similarity (**PSIM**). Perceptual loss Johnson et al. (2016) has been widely used for training image generation and style transferring models Johnson et al. (2016); Lucas et al. (2019); Wang et al. (2018). It emerges as a novel measurement for evaluating the discrepancy between high-level perceptual features that extracted by deep learning model of the reconstructed image and ground-truth image. We define PSIM as  $1 - \text{perceptual loss}$ .

## D ADDITIONAL EXPERIMENTAL RESULTS

### D.1 COMPARE PENALTY STRATEGIES

A natural approach arise to reduce data separability could be adding a penalty on the pair-wise distance for the data representations within a class. We name this approach as UniCon. Its loss function denoted as  $\mathcal{L}_{\text{unicon}}$  can be written as:

$$\mathcal{L}_{\text{unicon}} = \frac{1}{C} \frac{1}{|\mathcal{C}_c| \cdot (|\mathcal{C}_c| - 1)} \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{C}_c} \sum_{j \in \mathcal{C}_c} \|h(x_i) - h(x_j)\|_2^2,$$

The final objective function  $\mathcal{L} := \mathcal{L}_{\text{class}} + \lambda \cdot \mathcal{L}_{\text{unicon}}$ . This approach is similar to contrastive learning [Khosla et al. \(2020\)](#). However, we observed that the approach is not as ideal as our proposed MixCon, in the sense of defending inversion attack. The intuition is that MixCon can induce confusing patterns to fool the neural network learning typical patterns from a class. Here we show the visualization for the three benchmark datasets in Figure 5. We select  $\lambda = 1$  for MNIST and FashionMNIST and  $\lambda = 0.5$  for SVHN in both UniCon and MixCon. Then we choose the  $\beta = 1e - 4$  for MixCon to match the accuracy to Vanilla and UniCon. We use the same training and testing of MixCon for UniCon experiment. From the representative samples (while typical to the rest of the data samples), we observe worse data recovery quality of MixCon. Notably, the recovered results from UniCon keep the pattern of their class. While MixCon results in more blurred and indistinguishable patterns across classes. We compare the quantitative evaluation results between MixCon and UniCon in Table 4. <sup>8</sup> We use metric SSIM and PSIM to evaluate the similarity between the recovered image and the original image. Lower scores indicate worse data recovery results. The data recovery experiment is performed on 100 testing samples, and we report the mean  $\pm$  std and worst case (the best-recovered data) results. Except for the PSIM scores evaluated on MNIST, we get conformable evidence showing MixCon training is apt to defend inversion.

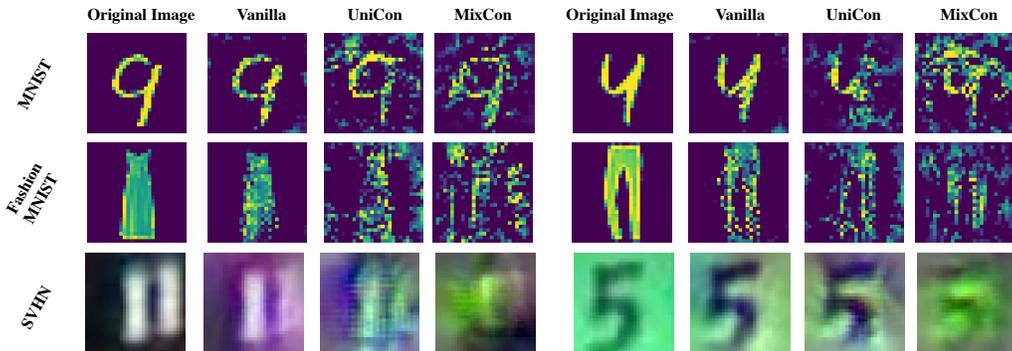


Figure 5: Qualitative evaluation for image inversion results.

	MNIST		FashionMNIST		SVHN	
	UniCon $\lambda = 1.0$	MixCon ( $\lambda = 1.0, \beta = 10^{-4}$ )	UniCon $\lambda = 1.0$	MixCon ( $\lambda = 1.0, \beta = 10^{-4}$ )	UniCon $\lambda = 0.5$	MixCon ( $\lambda = 0.5, \beta = 10^{-4}$ )
Acc (%)	99.2	98.6	89.6	88.9	88.3	88.2
SSIM	$0.31 \pm 0.11(0.59)$	$0.14 \pm 0.11(0.48)$	$0.19 \pm 0.09(0.53)$	$0.17 \pm 0.09(0.52)$	$0.67 \pm 0.11(0.91)$	$0.61 \pm 0.15(0.84)$
PSIM	$0.41 \pm 0.07(0.60)$	$0.44 \pm 0.07(0.69)$	$0.45 \pm 0.07(0.64)$	$0.42 \pm 0.08(0.66)$	$0.62 \pm 0.05(0.75)$	$0.59 \pm 0.07(0.72)$

Table 4: Quantitative evaluations for image recovery results. For fair evaluation, we match the data utility (accuracy) for Vanilla and MixCon. SSIM and PSIM are measured on 100 testing samples. Those scores are presented in mean  $\pm$  std and worst-case (in parentheses) format. The smaller scores indicate harder data recovery.

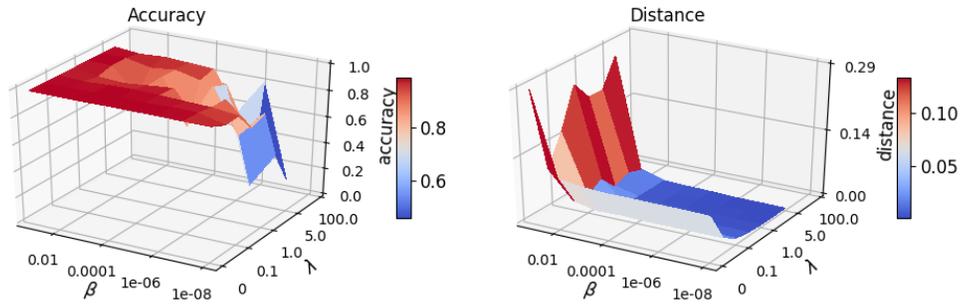
<sup>8</sup>We have presented the comparison between MixCon and vanilla training in Table 3.

## D.2 EFFECTS ON THE SELECTION OF MIDDLE LAYERS

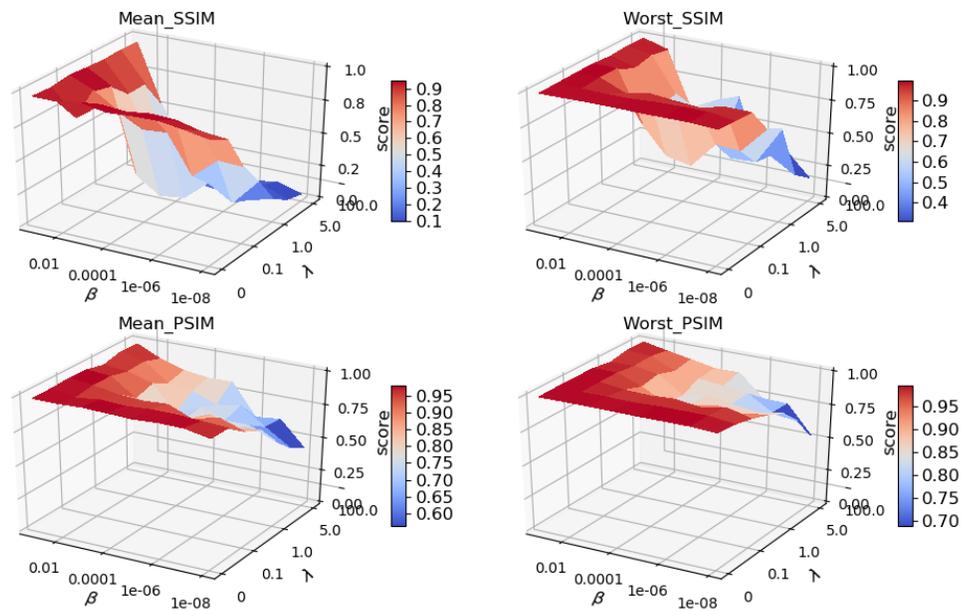
The trade-off between data separability and data utility can be different for adding MixConon the different layers. In our benchmark experiment, we use a LeNet5 [LeCun \(2015\)](#) — a five-layer CNN. Thus there are four split methods, namely four intermediate outputs. In our collaborative inference setting <sup>9</sup>, we visit the all four possible middle layers to apply MixCon loss. We plot the accuracy and data separability plots over the different combinations of  $(\beta, \lambda)$ , together with SSIM and PSIM scores (mean and the worst-case results), for each layer on our three benchmark datasets. The results are shown as Figure 6 to Figure 17. There is a clear trend that the shallower the  $h(x)$  it is, the easier to recover original  $x$  on average with respect to the mean SSIM and PSIM scores. The worst-case measurement may suffer from some outliers and imperfectness of the evaluation metrics. In most cases, distance and recovery similarity score for the first three layers shows a positive relationship, i.e. in Figure 6 - Figure 8. Usually, inversion from the deeper layers is not stable. Also, splitting a network at a deeper layer in the collaborative inference setting is not common or realistic because clients, such as edge-end devices, do not have powerful computational resources. Notably, the relationship between accuracy and similarity is highly non-linear. The sweet spot for a trade-off between accuracy and difficulty of recovery is in the space where the accuracy degradation curve is slow, while recovery similarity is low. Users can search the best parameters and "cut layer" to meet certain accuracy and data recovery defending requirements in practice.

---

<sup>9</sup>There is no necessity to add MixCon loss for the layers before "cut layer", because the attacker is not able to get access to the original data hidden representations form those layers

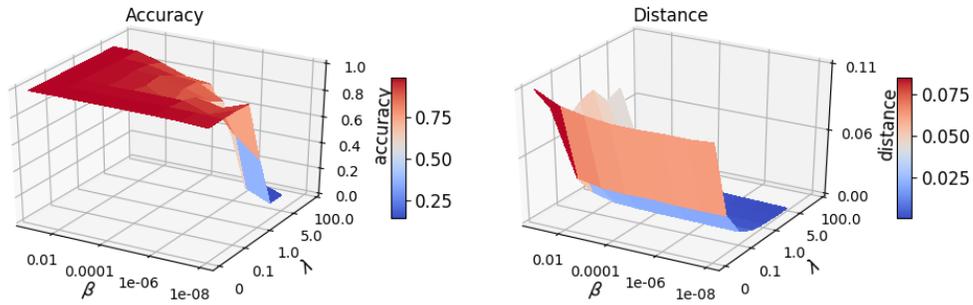


(a) Accuracy vs. Distance

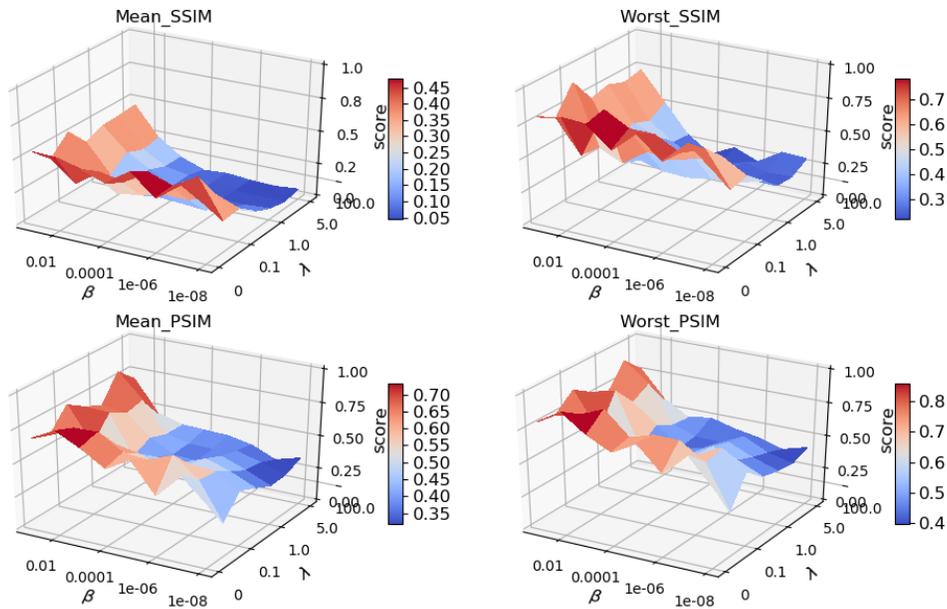


(b) Quantitative evaluations on data recovery

Figure 6: Adding MixCon to the 1st layer of CNN on MNIST dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .



(a) Accuracy vs. Distance



(b) Quantitative evaluations on data recovery

Figure 7: Adding MixCon to the 2nd layer of CNN on MNIST dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .

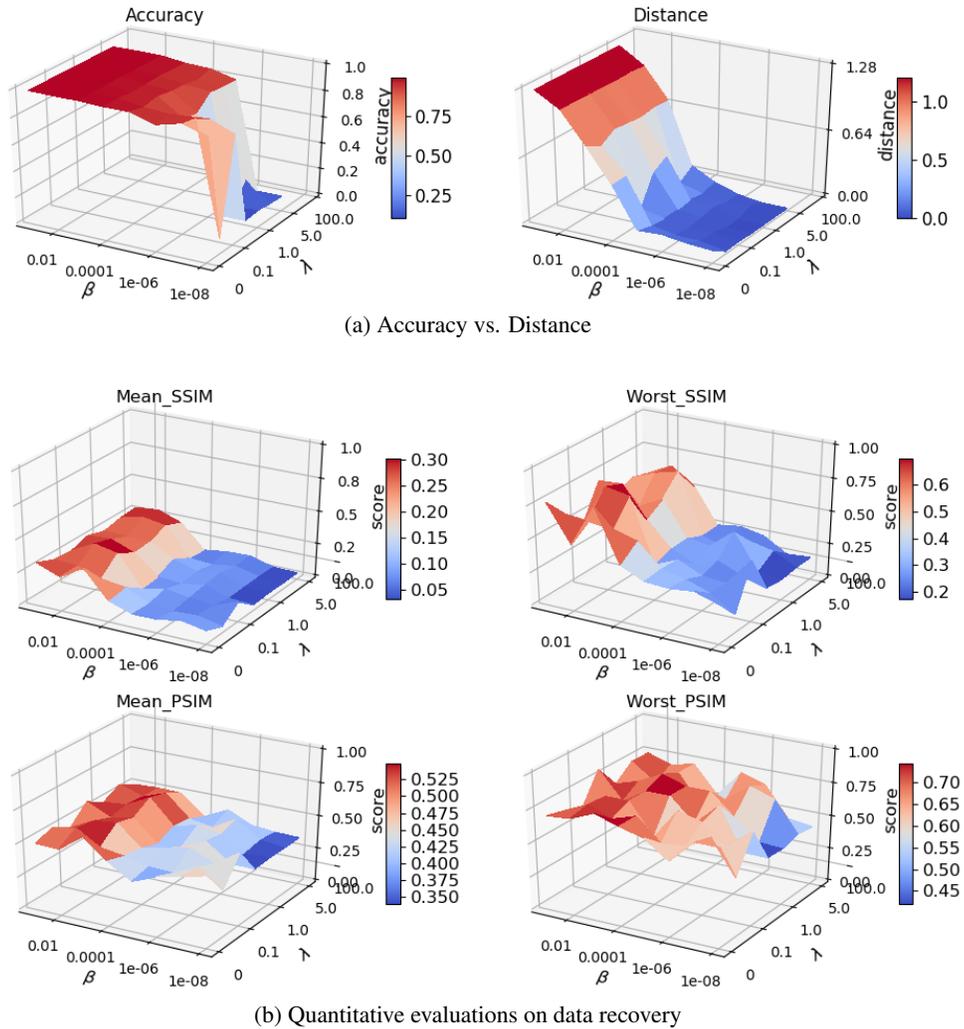


Figure 8: Adding MixCon to the 3rd layer of CNN on MNIST dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .

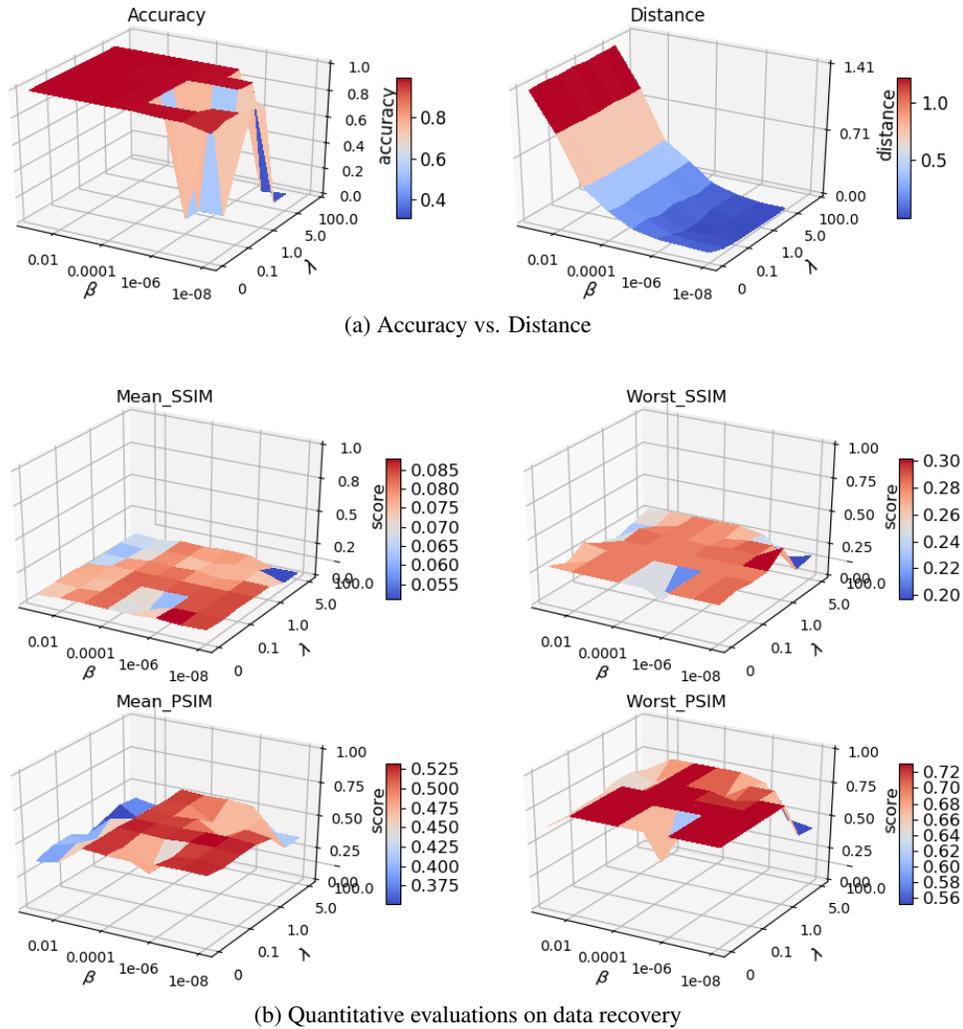
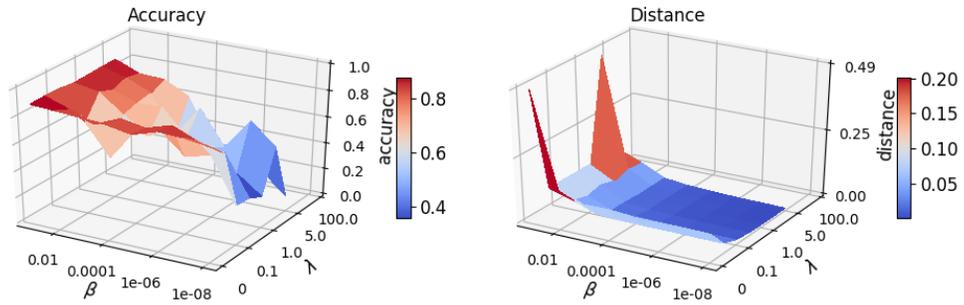
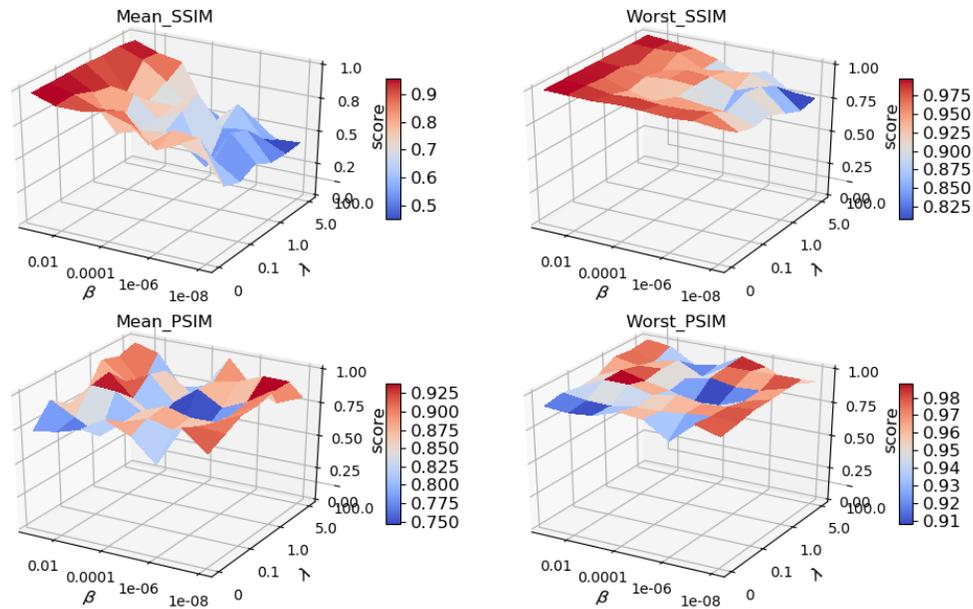


Figure 9: Adding MixCon to the 4th layer of CNN on MNIST dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .

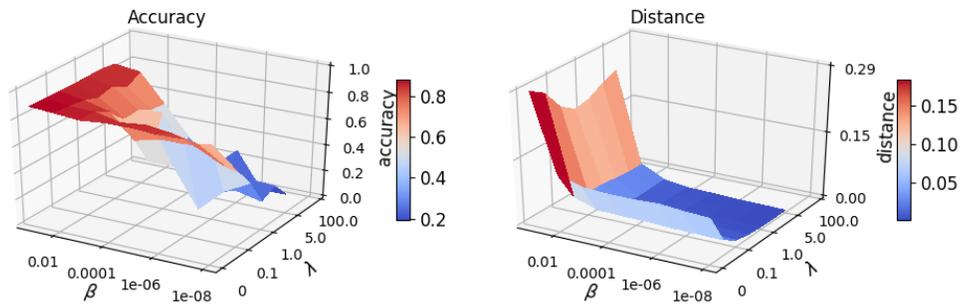


(a) Accuracy vs. Distance

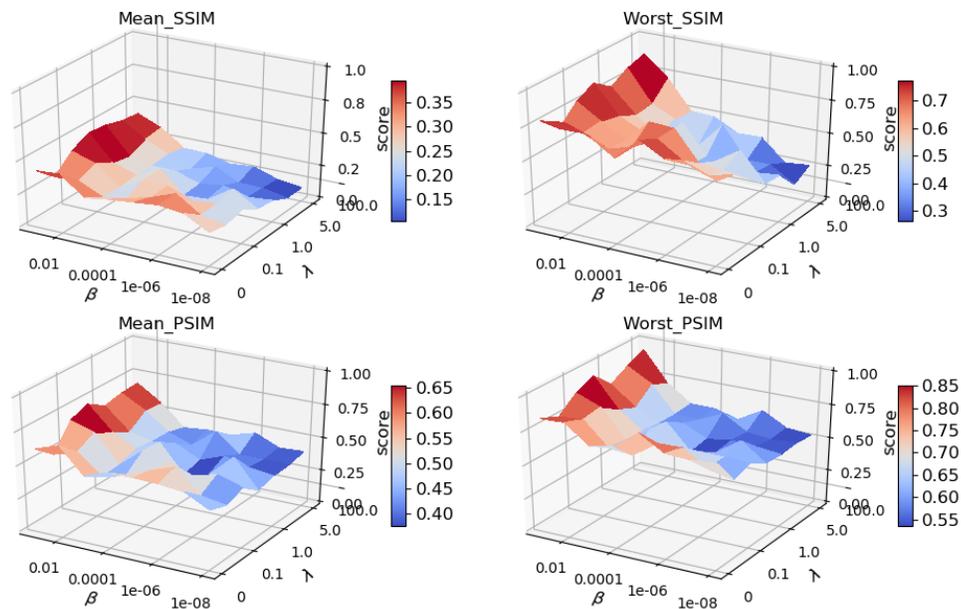


(b) Quantitative evaluations on data recovery

Figure 10: Adding MixCon to the 1st layer of CNN on FashionMNIST dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .

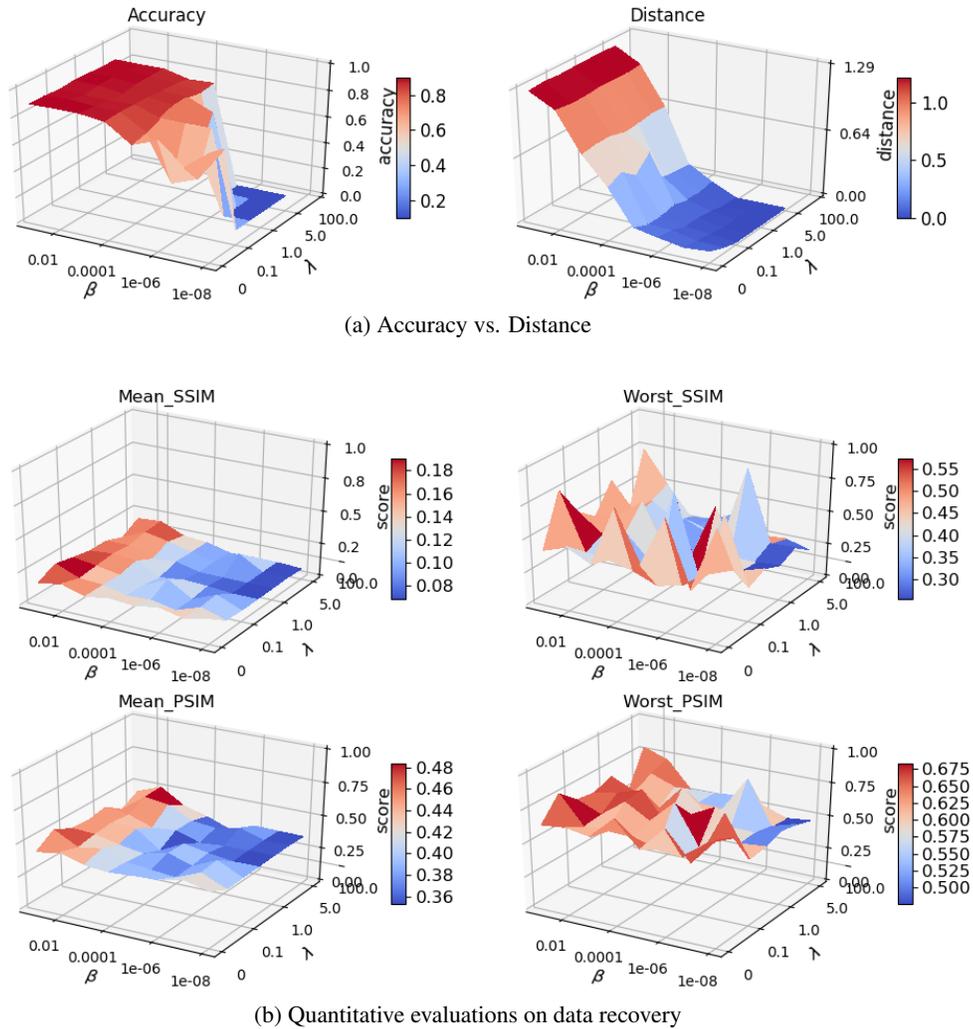


(a) Accuracy vs. Distance



(b) Quantitative evaluations on data recovery

Figure 11: Adding MixCon to the 2nd layer of CNN on FashionMNIST dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .



(a) Accuracy vs. Distance

(b) Quantitative evaluations on data recovery

Figure 12: Adding MixCon to the 3rd layer of CNN on FashionMNIST dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .

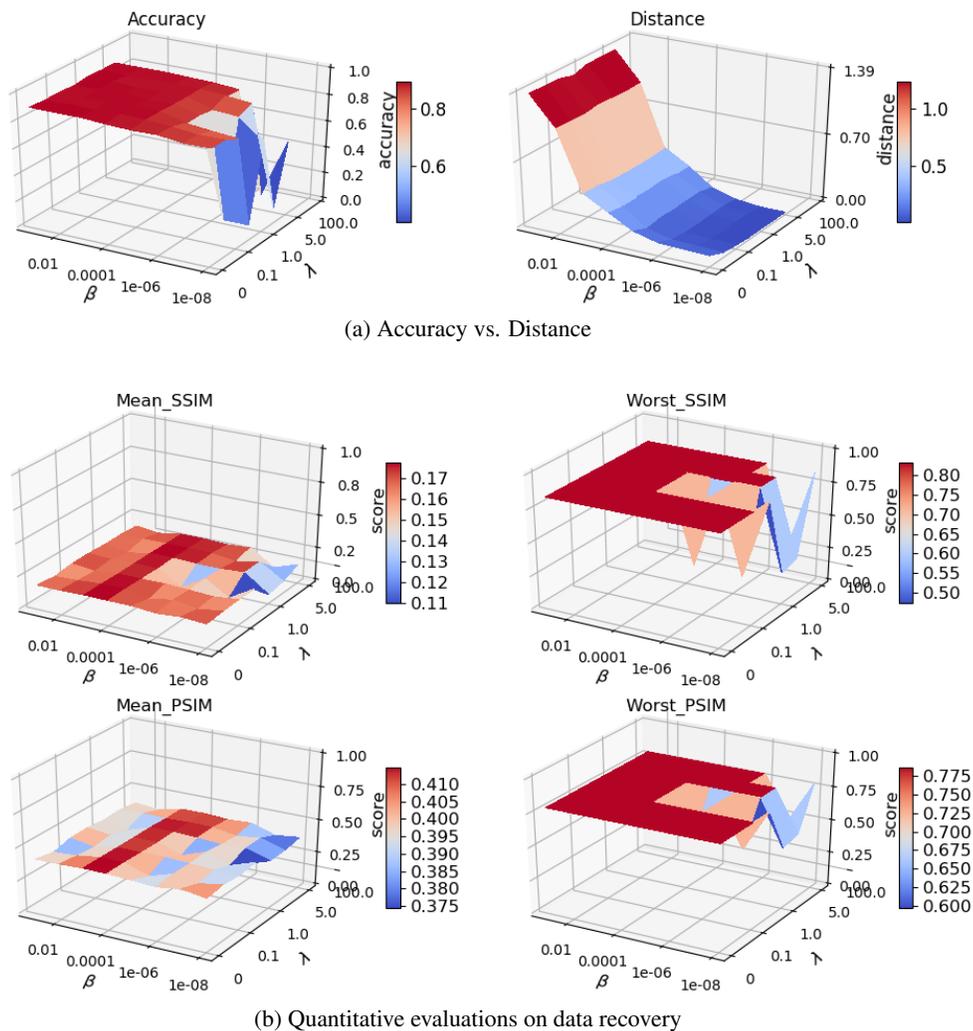
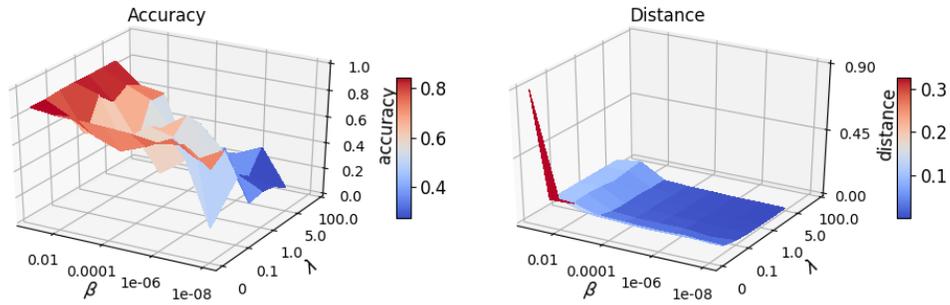
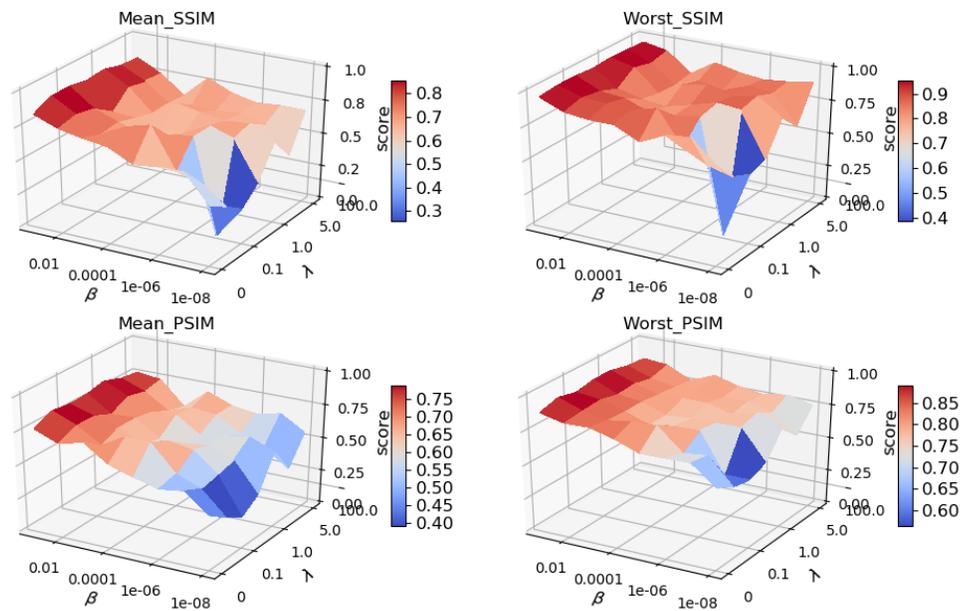


Figure 13: Adding MixCon to the 4th layer of CNN on FashionMNIST dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .



(a) Accuracy vs. Distance



(b) Quantitative evaluations on data recovery

Figure 14: Adding MixCon to the 1st layer of CNN on SVHN dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .

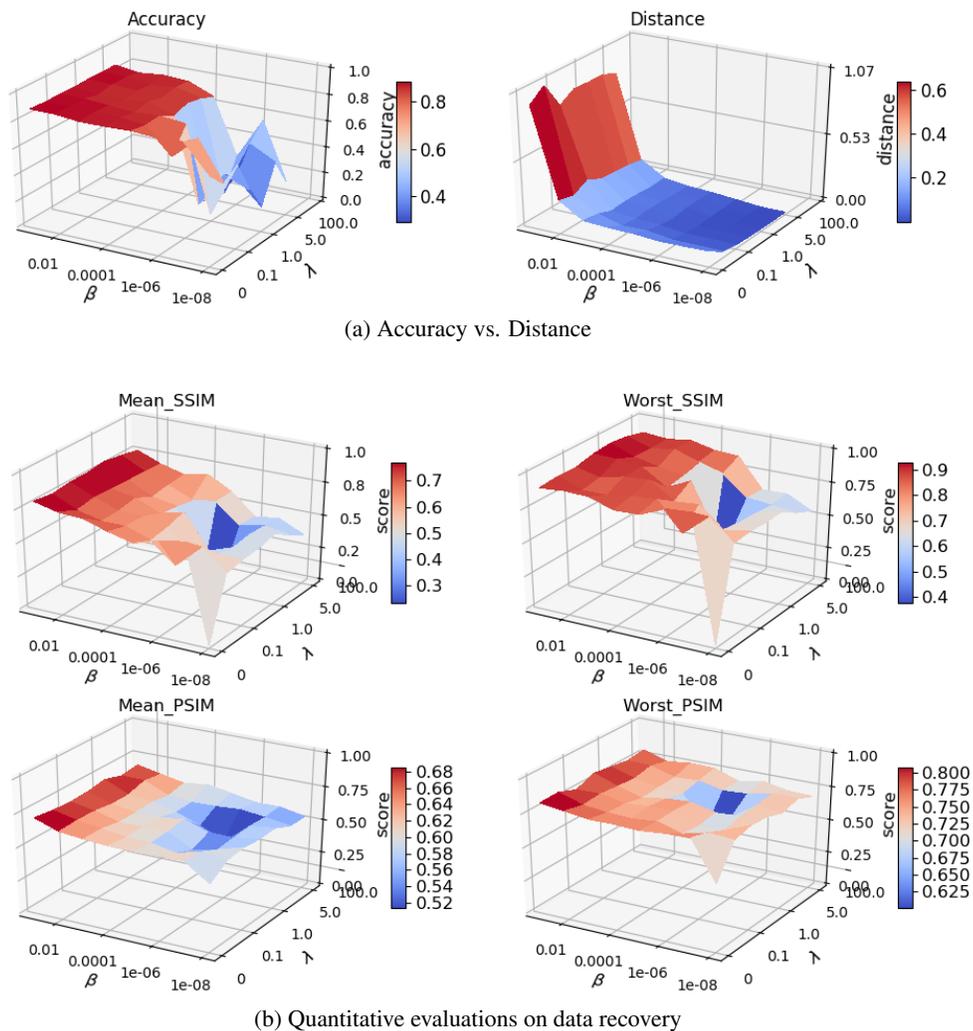


Figure 15: Adding MixCon to the 2nd layer of CNN on SVHN dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .

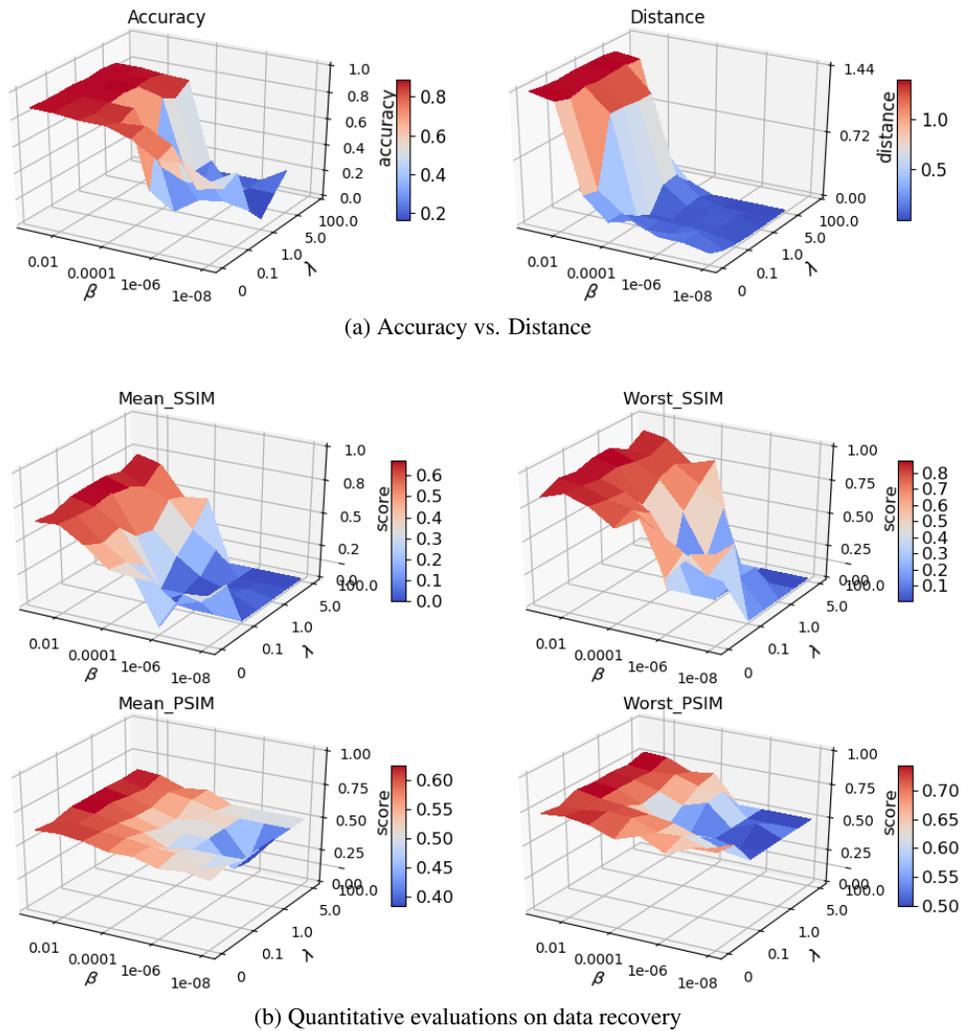


Figure 16: Adding MixCon to the 3rd layer of CNN on SVHN dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .

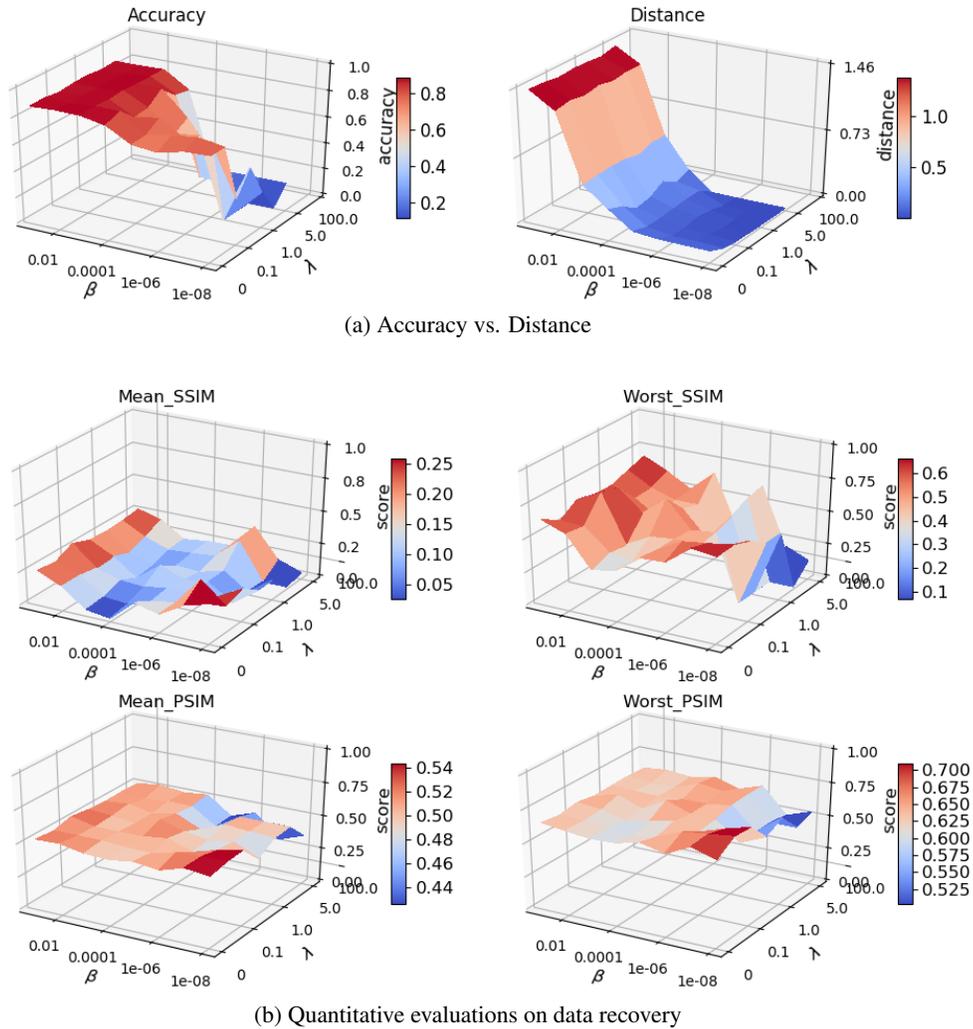


Figure 17: Adding MixCon to the 4th layer of CNN on SVHN dataset. (a) The trade-off between data separability and data utility. We show testing accuracy and mean pairwise distance (data separability) with different  $\lambda$  and  $\beta$ .  $\lambda$  and  $\beta$  show complementary effort on adjusting data separability. (b) Quantitative evaluation of data recovery results. We show SSIM and PSIM scores with different  $\lambda$  and  $\beta$ .