Inverting Deep Generative models, One layer at a time

Qi Lei [1] **Ajil Jalal** [1] **Inderjit S. Dhillon** [1 2] **Alexandros G. Dimakis** [1]

Abstract

We study the problem of inverting a deep generative model with ReLU activations. In most prior works this is performed by attempting to solve a non-convex optimization problem involving the generator with gradient method. In this paper we develop novel linear programming solvers with error bound analysis for different metrics. Our empirical validation demonstrates that we obtain better reconstructions when the latent dimension is large.

1. Introduction

Modern deep generative models are demonstrating excellent performance as signal priors, frequently outperforming the previous state of the art for various inverse problems including denoising, inpainting, reconstruction from Gaussian projections and phase retrieval (see e.g. (Bora et al., 2017; Fletcher & Rangan, 2017; Gupta et al., 2018; Dhar et al., 2018; Hand et al., 2018; Tripathi et al., 2018) and references therein).

A central problem that appears when trying to solve inverse problems using deep generative models is *inverting a generator* (Bora et al., 2017; Hand & Voroninski, 2017; Shah & Hegde, 2018). We are interested in deep generative models, parameterized as feed-forward neural networks with (Leaky)ReLU activations. Given a generator $G(\boldsymbol{z})$ that maps low-dimensional vectors in $\mathbb{R}^k$ to high dimensional vectors (e.g. images) in $\mathbb{R}^n$, we want to reconstruct the latent code $\boldsymbol{z}^*$ if we can observe $\boldsymbol{x} = G(\boldsymbol{z}^*)$ (realizable case) or a noisy version $\boldsymbol{x} = G(\boldsymbol{z}^*) + \boldsymbol{e}$ where $\boldsymbol{e}$ denotes some measurement noise.

We are therefore interested in the optimization problem

$$\arg\min_{\boldsymbol{z}} \|\boldsymbol{x} - G(\boldsymbol{z})\|_p, \tag{1}$$

for some p norm. This problem is a starting point for general sensing problems, and is a special case especially applied for image compressions, and image denoising tasks (Heckel & Hand, 2018). Previous work focuses on the ℓ_2 norm which works slowly with gradient descent (Bora et al., 2017; Huang et al., 2018). In this work, we focus on direct solvers and error bound analysis for ℓ_∞ and ℓ_1 norm instead. Note that this is a non-convex optimization problem even for a single-layer network with (Leaky)ReLU activations. Therefore gradient descent may easily get stuck at local minimum and may take a long time to converge. Take the MNIST dataset as an example, compression a single image by optimizing (1) takes on average several minutes and suffers from low success rate, which is not useful in practice.

Our Contributions: For the realizable case we show that for a single layer solving (1) is equivalent to solving a linear program. For two layers, however, the problem to recover a binary latent code is NP-hard even for realizable inputs. Meanwhile, the pre-image in the latent space can be non-convex set.

For realizable inputs and arbitrary depth we show that inversion is possible in polynomial time under some mild conditions. A similar result was established very recently for gradient descent (Huang et al., 2018). Unlike gradient descent that is conducted iteratively, we instead propose inversion by layer-wise Gaussian elimination. Our result holds even if each layer is expanding by a constant factor while (Huang et al., 2018) requires a logarithmic multiplicative expansion in each layer.

For noisy inputs and arbitrary depth we propose two direct solvers for different error types. We establish provable error bounds on the reconstruction error when the weights are random and have constant expansion. We also show empirically that our method matches and sometimes outperforms gradient descent for inversion, especially when the latent dimension becomes larger.

2. Setup

We consider deep generative models $G : \mathbb{R}^k \to \mathbb{R}^n$ with the latent dimension k smaller than the signal dimension n,

[*]Equal contribution [1]UT Austin [2]Amazon. Correspondence to: Qi Lei <leiqi@ices.utexas.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

parameterized by a d-layer feed-forward network:

$$G(\boldsymbol{z}) = \phi_d(\phi_{d-1}(\cdots \phi_2(\phi_1(\boldsymbol{z}))\cdots)), \quad (2)$$

where each layer $\phi_i(\boldsymbol{a})$ is defined as a composition of activations and linear maps: $\text{ReLU}(W_i\boldsymbol{a} + \boldsymbol{b}_i)$. We focus on the ReLU activations $\text{ReLU}(\boldsymbol{a}) = \max\{\boldsymbol{a}, \boldsymbol{0}\}$ applied coordinate-wise, and we will also consider the activation as $\text{LeakyReLU}(\boldsymbol{a}) = \text{ReLU}(\boldsymbol{a}) + c\text{ReLU}(-\boldsymbol{a})$, where the scaling factor $c \in (0, 1)$. [1] $W_i \in \mathbb{R}^{n_i \times n_i}$ [1] are the weights of the network, and $\boldsymbol{b}_i \in \mathbb{R}^{n_i}$ are the bias terms. Therefore, $n_0 = k$ and $n_d = n$ indicate the dimensionality of the input and output of the generator G. We use $\boldsymbol{z}_i$ to denote the output of the i-th layer. Note that one can absorb the bias term $\boldsymbol{b}_i, i = 1, 2, \cdots d$ into W_i by adding one more dimension with a constant input. Therefore, without loss of generality, we sometimes omit $\boldsymbol{b}_i$ when writing the equation, unless we explicitly needed it.

We use bold lower-case symbols for vectors, e.g. $\boldsymbol{x}$, and x_i for its coordinates. We use upper-case symbols for matrices, e.g. W, where $\boldsymbol{w}_i$ is its i-th row vector. For a indexed set I, $W_{I,:}$ represents the submatrix of W consisting of each i-th row of W for any $i \in I$.

3. Invertibility for ReLU Realizable Networks

In this section we study the realizable case, i.e., given an observation vector $\boldsymbol{x}$, $\exists \boldsymbol{z}^*$ s.t. $\boldsymbol{x} = G(\boldsymbol{z}^*)$. In particular, we show that the problem is NP-hard for ReLU activations in general, but could be solved in polynomial time with some mild assumptions with high probability. We present our theoretical findings first and all proofs of the paper are presented in Appendix B.

Inverting a Single Layer: We start with the simplest one-layer case to find if $\min_{\boldsymbol{z}} \|\boldsymbol{x} - G(\boldsymbol{z})\|_p = 0$, for any p-norm. Since the problem is non-convex, further assumptions of W are required (Huang et al., 2018) for gradient descent to work. When the problem is realizable, however, to find feasible $\boldsymbol{z}$ such that $\boldsymbol{x} = \phi(\boldsymbol{z}) \equiv \text{ReLU}(W\boldsymbol{z} + \boldsymbol{b})$, one could invert the function by solving a linear system:

$$\boldsymbol{w}_i^\top \boldsymbol{z} + b_i = x_i, \quad \forall i \text{ s.t. } x_i > 0$$

Its solution set is convex and forms a polytope, but possibly includes uncountable feasible points. Therefore, it becomes unclear how to continue the process of layer-wise inversion unless further assumptions are made.

Challenges to Invert a Two-Layer ReLU Network: We now show that the problem of recovering a binary latent code for a two layer network is NP-hard, using a reduction from the MAX-3SAT problem.

Theorem 1. *Given an observation vector $\boldsymbol{x}$, consider the problem of finding $\boldsymbol{z} \in \{\pm 1\}$, such that $G(\boldsymbol{z}) := ReLU(W_2(ReLU(W_1\boldsymbol{z} + \boldsymbol{b}_1) + \boldsymbol{b}_2) = \boldsymbol{x}$. The problem is NP-hard since it can be reduced from MAX-3SAT.*

[1] The inversion of LeakyReLU networks is mostly dominated by ReLU networks and we therefore only mention it when needed.

Meanwhile, although the preimage for a single layer is a polytope thus convex, it doesn't continue to hold for more than one layers, see Example 1. Fortunately, we present next that some moderate conditions guarantee a polynomial time solution with high probability.

Inverting Expansive Random Network in Polynomial Time:

Assumption 1. *For a weight matrix $W \in \mathbb{R}^{n \times k}$, we assume*
1. its entries are sampled i.i.d Gaussian, and
2. W is tall: $n = c_0 k$ for some constant $c_0 \geq 2.1$.

In the previous section, we indicate that the per layer inversion can be achieved through Gaussian eliminization. With Assumption 1 we will be able to prove that the solution is unique with high probability, and thus Theorem 2 holds for ReLU networks with arbitrary depth.

Theorem 2. *Let $G \in \mathbb{R}^k \to \mathbb{R}^n$ be a generative model defined in (2). If the weight matrices $W_i, i \in [d]$ satisfies Assumption 1, then for any $\boldsymbol{z}^* \in \mathbb{R}^k$ and observation $\boldsymbol{x} = G(\boldsymbol{z}^*)$, with probability $1 - e^{-\Omega(k)}$, $\boldsymbol{z}^*$ can be inferred from $\boldsymbol{x}$ by solving layer-wise linear equations. Namely, a random, expansive and realizable generative model can be inverted in polynomial time with high probability.*

Therefore the time complexity of exact recovery is no worse than $\sum_{i=0}^{d-1} n_i^{2.376}$ (Golub & Van Loan, 2012) since the recovery simply requires solving d linear equations with dimension $n_{i-1}, i \in [d]$. On the other hand, inversion of LeakyReLU layers are significantly easier for the realizable case, as presented in remark 1.

4. Invertibility for Noisy ReLU Networks

Besides the realizable case, the study of noise tolerance is essential for many real applications. In this section, we thus consider the noisy setting with observation $\boldsymbol{x} = G(\boldsymbol{z}^*) + \boldsymbol{e}$, with both ℓ_∞ and ℓ_1 error bound, in favor of different types of random noise distribution. In this section, all generators are without the bias term.

4.1. ℓ_∞ Norm Error Bound

Again we start with a single layer, i.e. we observe $\boldsymbol{x} = \phi(\boldsymbol{z}^*) + \boldsymbol{e} = \text{ReLU}(W\boldsymbol{z}^*) + \boldsymbol{e}$. We first look at the case where the entries of $\boldsymbol{e}$ are uniformly bounded and the approximation of $\arg\min_{\boldsymbol{z}} \|\phi(\boldsymbol{z}) - \boldsymbol{x}\|_\infty$.

We notice that for an $\epsilon \geq \|\boldsymbol{e}\|_\infty$, the true prior $\boldsymbol{z}^*$ that produces the observation $\boldsymbol{x} = \phi(\boldsymbol{z}^*) + \boldsymbol{e}$ falls into the

following constraints:

$$\begin{aligned} x_j - \epsilon \le \boldsymbol{w}_j^\top \boldsymbol{z} \le x_j + \epsilon \quad & \text{if } x_j > \epsilon, j \in [n] \\ \boldsymbol{w}_j^\top \boldsymbol{z} \le x_j + \epsilon \quad & \text{if } x_j \le \epsilon, j \in [n] \\ z_i \ge 0 \qquad & \forall i \in [k], \end{aligned} \tag{3}$$

where the last term should be omitted to recover the first layer. Therefore a natural way to approximate the prior is to use linear programming to solve the above constraints.

A layer-wise inversion is formally presented in Algorithm 1 where we start from a small estimation of ϵ and gradually increase the tolerance until feasibility is achieved [2].

A key assumption that possibly conveys the error bound from the output to the solution is the following assumption:

Assumption 2 (Submatrix Extends ℓ_∞ Norm)**.** *For the weight matrix $W \in \mathbb{R}^{n\times k}$, there exists an integer $m > k$ and a constant c_∞, such that for any $I \subset [n] := \{1, 2, \cdots n\}, |I| \ge m$, $W_{I,:}$ satisfies*

$$W_{I,:} \cdot \|\boldsymbol{x}\|_\infty \ge c_\infty \|\boldsymbol{x}\|_\infty,$$

with probability $1 - e^{-\Omega(k)}$ for any $\boldsymbol{x}$, and c_∞ is a constant. Recall that $W_{I,:}$ is the sub-rows of W confined to I.

This condition enables the layer-wise inversion to produce sufficiently small error given enough positive observations, thus gives us a tight inversion in Theorem 4 in the appendix. We argue that the assumptions required could be satisfied by random weight matrices sampled from i.i.d Gaussian distribution, and present the following corollary.

Corollary 1. *Let $\boldsymbol{x} = G(\boldsymbol{z}^*) + \boldsymbol{e}$ be a noisy observation produced by the generator G defined in (2). Let each weight matrix $W_i \in \mathbb{R}^{n_i \ 1 \times n_i}$ ($n_i \ge 5n_{i-1}, \forall i$) be sampled from i.i.d Gaussian distribution $\sim \mathcal{N}(0, 1)$, then W_i satisfies Assumption 2 with some constant $c_2 \in (0, 2]$. Let the error $\boldsymbol{e}$ satisfies $\ell_\infty = \epsilon$, where $\epsilon < \frac{c_2^d}{2^{d+4}} \|\boldsymbol{z}^*\|_2 \sqrt{k}$. By recursively applying Algorithm 1, it produces an $\boldsymbol{z}$ that satisfies $\|\boldsymbol{z} - \boldsymbol{z}^*\|_\infty \le \frac{2^d \epsilon}{c_2^d}$ with probability $1 - e^{\Omega(k)}$.*

Refer to Remark 2 for layer-wise inversion of LeakyReLU.

4.2. ℓ_1 Norm Error Bound

In this section we develop a generative model inversion framework using the ℓ_1 norm. We introduce Algorithm 2 that tolerates error in different level for each output coordinate and intends to minimize the ℓ_1 norm error bound.

Different from Algorithm 1, the deviating error allowed on each observation is no longer uniform and the new algorithm is actually optimizing over the ℓ_1 error. Similar to the error bound analysis with ℓ_∞ norm we are able to get some tight approximation guarantee under some mild assumption related to Restricted Isometry Property for ℓ_1 norm:

Assumption 3 (Submatrix Extends ℓ_1 Norm)**.** *For a weight matrix $W \in \mathbb{R}^{n\times k}$, there exists an integer $m > k$ and a constant c_1, such that for any $I \subset [n], |I| \ge m$, $W_{I,:}$ satisfies*

$$\|W_{I,:} \cdot \boldsymbol{x}\|_1 \ge c_1 \|\boldsymbol{x}\|_1, \tag{4}$$

with probability $1 - e^{-\Omega(k)}$ for any $\boldsymbol{x}$.

This assumption is a special case of the lower bound of the well-studied Restricted Isometry Property, for ℓ_1-norm and sparsity k, i.e., (k, ∞)-RIP-1. Similar to the ℓ_∞ analysis, we are able to get recovery guarantees for generators with arbitary depth.

Theorem 3. *Let $\boldsymbol{x} = G(\boldsymbol{z}^*) + \boldsymbol{e}$ be a noisy observation produced by the generator G, a d-layer ReLU network mapping from $\mathbb{R}^k \to \mathbb{R}^n$. Let each weight matrix $W_i \in \mathbb{R}^{n_i \ 1 \times n_i}$ satisfy Assumption 3 with the integer $m_i > n_{i-1}$ and constant c_1. Let the error $\boldsymbol{e}$ satisfy $\|\boldsymbol{e}\|_1 \le \epsilon$, and for each $\boldsymbol{z}_i = \phi_i(\phi_{i-1}(\cdots \phi(\boldsymbol{z}^*) \cdots))$, at least m_i coordinates are larger than $\frac{2^{d+1 \ i} \epsilon}{c_1^{d \ i}}$. Then by recursively applying Algorithm 2, it produces a $\boldsymbol{z}$ that satisfies $\|\boldsymbol{z} - \boldsymbol{z}^*\|_1 \le \frac{2^d \epsilon}{c_1^d}$ with probability $1 - e^{-\Omega(k)}$.*

There is a significant volume of prior work on the RIP-1 condition. For instance, studies in (Berinde et al., 2008) showed that a (scaled) random sparse binary matrix with $m = O(s \log(k/s)/\epsilon^2)$ rows is $(s, 1+\epsilon)$-RIP-1 with high probability. In our case $s = k$ and ϵ could be arbitrarily large, therefore again we only require the expansion factor to be constant. Similar results with different weight matrices are also shown in (Nachin, 2010; Indyk & Razenshteyn, 2013; Allen-Zhu et al., 2016).

5. Experiments

In this section, we compared our methods ℓ_1 LP and ℓ_∞ LP with gradient descent (GD) (Hand & Voroninski, 2017).

5.1. Synthetic Data

We validate our algorithms on synthetic network at various noise levels. We first fix the network architecture and investigate the influence of different noise level, and then fix all but the input dimension to verify our expanding analysis.

Recovery with Various Input Neurons: In Figure 1 we report the empirical success rate of recovery for our proposals and GD. With exact setting as in (Huang et al., 2018), a run is considered successful when $\|\boldsymbol{z}^* - \boldsymbol{z}\|_2 / \|\boldsymbol{z}^*\|_2 \le 10^{-3}$. We observe that when input width k is small, both GD and our methods grant 100% success rate. However, as the input neurons grows, GD drops to complete failure when $k \ge 60$, while our algorithms continue to present 100% success rate until $k = 109$.

[2] For practical use, we introduce a factor α to gradually increase the error estimation. In our theorem, it assumed we expicitly set ϵ to invert the i-th layer as $2^{d \ i} \|\boldsymbol{e}\|_0 / c_2^{d \ i}$.

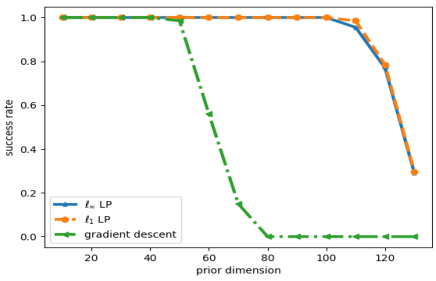


Figure 1. Comparison of our method and GD on the empirical success rate of recovery (200 runs on random networks) versus the number of input neurons k for the noiseless problem. The architecture chosen here is a 2 layer fully connected ReLU network, with 250 hidden nodes, and 600 output neurons. Our algorithms are significantly outpeforming GD for higher latent dimensions k.

Recovery with Various Observation Noise: In Figure 2(a)(b) we plot the relative recovery error $\|\boldsymbol{z}-\boldsymbol{z}^*\|_2/\|\boldsymbol{z}^*\|_2$ at different noise levels. It supports our theoretical findings that with other parameters fixed, the recovery error grows almost linearly to the observation noise. Meanwhile, we observe in both cases, our methods perform similarly to GD on average, while GD is less robust and produces more outlier points. As expected, our ℓ_∞ LP performs slightly better than GD when the input error is uniformly bounded; see Figure 2(a). However, with a large variance in the observation error, as seen in Figure 2(b), ℓ_∞ LP is not as robust as ℓ_1 LP or GD.

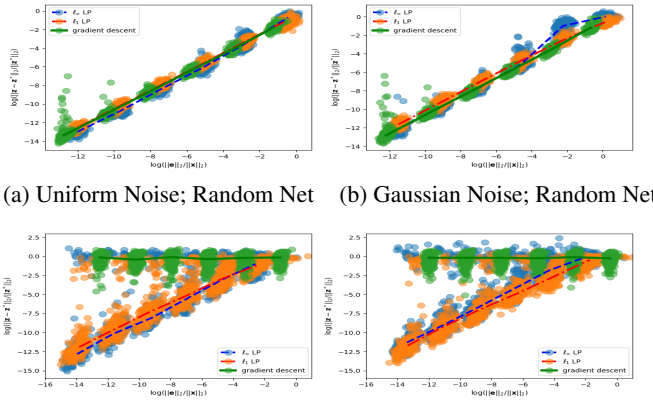
5.2. Experiments on Generative Model for MNIST

To verify the practical contribution of our model, we experiment on a real generative network with the MNIST dataset.

Similar to the simulation part, we compared our methods with GD (Hand & Voroninski, 2017). Under this setting, we choose the learning rate to be 10^{-3} and number of iterations up to 10,000 (or until gradient norm is below 10^{-9}).

We first randomly select some empirical examples to visually show performance comparison in Figure 3. In these examples, observations are perturbed with some Gaussian random noise with variance 0.3 and we use ℓ_∞ LP as our algorithm to invert the network. From the figures, we could see that our method could almost perfectly denoise and reconstruct the input image, while GD impairs the completeness of the original images to some extent.

We also compare the distribution of relative recovery error with respect to different input noise levels, as ploted in Figure 2(c)(d). From the figures, we observe that for this real network, our proposals still successfully recover the ground truth with good accuracy most of the time, while GD usually gets stuck in local minimum. This explains why



(a) Uniform Noise; Random Net (b) Gaussian Noise; Random Net

(c) Uniform Noise; Real Net (d) Gaussian Noise; Real Net

Figure 2. Comparison of our proposals (ℓ_1 LP and ℓ_1 LP) versus GD. On x-axis we plot the relative noise level and relative recovery error on y-axis. In experiments (a)(b) the network is randomly generated and fully connected, with `20` input neurons, `100` hidden neurons and `500` output neurons. Each dot represents a recovery experiment (among 200 for each noise level). Each line connects the median of the 200 runs for each noise level. As can be seen, our algorithm (Blue/Orange) has similar performance to gradient descent, except at low noise levels where it is slightly more robust. In experiments (c)(d) the network is generative model for the MNIST dataset. In this case, GD fails to find global minimum in almost all the cases.

it produces defective image reconstructions as shown in 3.

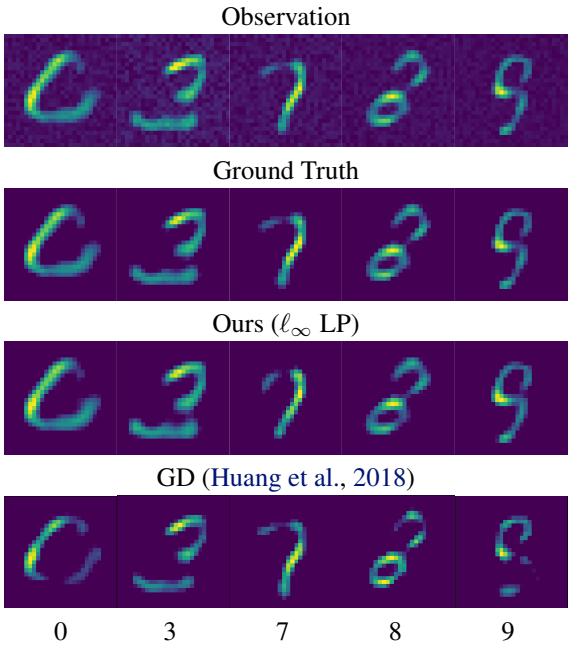


Figure 3. Recovery comparison using our algorithm ℓ_1 LP versus GD for an MNIST generative model. Notice that ℓ_1 LP produces reconstructions that are clearly closer to the ground truth.

References

Allen-Zhu, Z., Gelashvili, R., and Razenshteyn, I. Restricted isometry property for general p-norms. *IEEE Transactions on Information Theory*, 62(10):5839–5854, 2016.

Berinde, R., Gilbert, A. C., Indyk, P., Karloff, H., and Strauss, M. J. Combining geometry and combinatorics: A unified approach to sparse signal recovery. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pp. 798–805. IEEE, 2008.

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. Compressed sensing using generative models. *arXiv preprint arXiv:1703.03208*, 2017.

Dhar, M., Grover, A., and Ermon, S. Modeling sparse deviations for compressed sensing using generative models. *arXiv preprint arXiv:1807.01442*, 2018.

Donoho, D. L., Maleki, A., and Montanari, A. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009.

Fletcher, A. K. and Rangan, S. Inference in deep networks in high dimensions. *arXiv preprint arXiv:1706.06549*, 2017.

Golub, G. H. and Van Loan, C. F. *Matrix computations*, volume 3. JHU Press, 2012.

Gupta, S., Kothari, K., de Hoop, M. V., and Dokmanić, I. Deep mesh projectors for inverse problems. *arXiv preprint arXiv:1805.11718*, 2018.

Hand, P. and Voroninski, V. Global guarantees for enforcing deep generative priors by empirical risk. *arXiv preprint arXiv:1705.07576*, 2017.

Hand, P., Leong, O., and Voroninski, V. Phase retrieval under a generative prior. In *Advances in Neural Information Processing Systems*, pp. 9154–9164, 2018.

Heckel, R. and Hand, P. Deep decoder: Concise image representations from untrained non-convolutional networks. *arXiv preprint arXiv:1810.03982*, 2018.

Huang, W., Hand, P., Heckel, R., and Voroninski, V. A provably convergent scheme for compressive sensing under random generative priors. *arXiv preprint arXiv:1812.04176*, 2018.

Indyk, P. and Razenshteyn, I. On model-based rip-1 matrices. In *International Colloquium on Automata, Languages, and Programming*, pp. 564–575. Springer, 2013.

Metzler, C. A., Maleki, A., and Baraniuk, R. G. From denoising to compressed sensing. *IEEE Transactions on Information Theory*, 62(9):5117–5144, 2016.

Nachin, M. Lower bounds on the column sparsity of sparse recovery matrices. *UAP: MIT Undergraduate Thesis*, 2010.

Rudelson, M. and Vershynin, R. Smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences 62.12*, pp. 1707–1739, 2009.

Schniter, P., Rangan, S., and Fletcher, A. K. Vector approximate message passing for the generalized linear model. In *Signals, Systems and Computers, 2016 50th Asilomar Conference on*, pp. 1525–1529. IEEE, 2016.

Shah, V. and Hegde, C. Solving linear inverse problems using gan priors: An algorithm with provable guarantees. *arXiv preprint arXiv:1802.08406*, 2018.

Tripathi, S., Lipton, Z. C., and Nguyen, T. Q. Correction by projection: Denoising images with generative adversarial networks. *arXiv preprint arXiv:1803.04477*, 2018.

A. Methodology Details

In this section we present the detailed steps for our proposed methods. Firstly we add some remarks for LeakyReLU:

Remark 1. *Unlike ReLU, LeakyReLU is a bijective map, i.e., each observation corresponds to a unique preimage:*

$$\textit{LeakyReLU}^{-1}(x) = \begin{cases} x & \textit{if } x \geq 0 \\ 1/cx & \textit{otherwise.} \end{cases} \tag{5}$$

Therefore, if each $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ is of rank at least n_{i-1}, each layer map ϕ_i has a unique preimage (in the realizable case) and could be computed by the inverse of LeakyReLU (5) and linear regression.

A.1. ℓ_∞ LP

We first include the detailed algorithm to invert a single layer with ℓ_∞ error bound that we call ℓ_∞ LP. When we don't know how much noise we expect, we could start from a small value of error tolerance ϵ and gradually increase the tolerance until LP returns feasible solution.

Algorithm 1 Linear programming to invert a single layer with ℓ_∞ error bound (ℓ_∞ LP)

Input: Observation $\boldsymbol{x} \in \mathbb{R}^n$, weight matrix $W = [\boldsymbol{w}_1|\boldsymbol{w}_2|\cdots|\boldsymbol{w}_n]^\top$, initial error bound guess $\epsilon > 0$, scaling factor $\alpha > 1$.
repeat
Solve the following linear programming:

$$\begin{aligned} \arg\min_{\boldsymbol{z},\delta}\ & \delta \\ \text{s.t. } & x_j - \delta \leq \boldsymbol{w}_j^\top \boldsymbol{z} \leq x_j + \delta && \text{if } x_j > \epsilon \\ & \boldsymbol{w}_j^\top \boldsymbol{z} \leq x_j + \delta && \text{if } x_j \leq \epsilon \\ & \delta \leq \epsilon \\ & z_k \geq 0 && \forall k. \end{aligned}$$

$\epsilon \leftarrow \epsilon\alpha$
until $\boldsymbol{z}$ infeasible
Output: $\boldsymbol{z}$

Remark 2. *For LeakyReLU, we could do at least as good as ReLU, since we could simply view all negative coordinates as inactive coordinates of ReLU, and each observation will produce a loose bound.*

On the other hand, if there are significant number of negative entries, we could also change the linear programming constraints of Algorithm 1 as follows:

$$\begin{aligned} \arg\min_{\boldsymbol{z},\delta}\ & \delta \\ \textit{s.t. } & x_j - \delta \leq \boldsymbol{w}_j^\top \boldsymbol{z} \leq x_j + \delta && \textit{if } x_j > \epsilon \\ & 1/c(x_j - \delta) \leq \boldsymbol{w}_j^\top \boldsymbol{z} \leq x_j + \delta && \textit{if } -\epsilon < x_j \leq \epsilon \\ & x_j - \delta \leq c\boldsymbol{w}_j^\top \boldsymbol{z} \leq x_j + \delta && \textit{if } x_j \leq -\epsilon \\ & \delta \leq \epsilon. \end{aligned}$$

A.2. ℓ_1 LP

We then present ℓ_1 LP to tolerate noise non-uniform in different directions in Algorithm 2.

Algorithm 2 Linear programming to invert a single layer with ℓ_1 error bound (ℓ_1 LP)

Input: Observation $\boldsymbol{x} \in \mathbb{R}^n$, weight matrix $W = [\boldsymbol{w}_1|\boldsymbol{w}_2|\cdots|\boldsymbol{w}_n]^\top$, initial error bound guess $\epsilon > 0$, scaling factor $\alpha > 1$.
for $t = 1, 2, \cdots$ **do**
Solve the following linear programming:

$$\begin{aligned} \boldsymbol{z}^{(t)}, \boldsymbol{e}^{(t)} \leftarrow \arg\min_{\boldsymbol{z},\boldsymbol{e}}\ & \sum_i e_i \\ \text{s.t. } & x_j - e_j \leq \boldsymbol{w}_j^\top \boldsymbol{z} \leq x_j + e_j && \text{if } x_j > \epsilon \\ & \boldsymbol{w}_j^\top \boldsymbol{z} \leq x_j + e_j && \text{if } x_j \leq \epsilon \\ & e_j \geq 0 && \forall j \in [n] \\ & z_k \geq 0 && \forall k. \end{aligned}$$

$\epsilon \leftarrow \epsilon\alpha$
if $t \geq 2$ **and** $\|\phi(\boldsymbol{z}^{(t)}) - \boldsymbol{x}^*\|_1 \geq \|\phi(\boldsymbol{z}^{(t-1)}) - \boldsymbol{x}^*\|_1$ **then**
return $\boldsymbol{z}^{(t-1)}$
end if
end for

We also introduce the ℓ_1 LP for LeakyReLU. The framework is mostly similar to Algorithm 2, and the linear programming constraints are modified with more information from negative observations.

$$\begin{aligned} \boldsymbol{z}^{(t)}, \boldsymbol{e}^{(t)} \leftarrow \arg\min_{\boldsymbol{z},\boldsymbol{e}}\ & \sum_i e_i \\ \text{s.t. } & x_j - e_j \leq \boldsymbol{w}_j^\top \boldsymbol{z} \leq x_j + e_j && \text{if } x_j > \epsilon \\ & 1/c(x_j - e_j) \leq \boldsymbol{w}_j^\top \boldsymbol{z} \leq x_j + e_j && \text{if } \epsilon \leq x_j \leq \epsilon \\ & x_j - e_j \leq c\boldsymbol{w}_j^\top \boldsymbol{z} \leq x_j + e_j && \text{if } x_j < -\epsilon \\ & e_j \geq 0 && \forall j \in [n]. \end{aligned}$$

A.3. Relaxation on the ReLU Configuration Estimation

Our previous methods critically depend on the correct estimation of the observation signs. In both Algorithm 1 and 2, we require the ground truth of all intermediate layer outputs to have many coordinates with large magnitude so that they can be distinguished from noise. An incorrect estimate from an "off" configuration to an "on" condition will possibly cause primal infeasibility when solving the LP. Increasing ϵ ameliorates this problem but also increases the recovery error.

With this intuition, a natural workaround is to perform some relaxation to tolerate incorrectly estimated signs of the ob-

servations.

$$\max_{\boldsymbol{z}} \quad \sum_i \max\{0, x_i\} \boldsymbol{w}_i^\top \boldsymbol{z} =: \text{ReLU}(\boldsymbol{x})^\top W \boldsymbol{z},$$
$$s.t \qquad \boldsymbol{w}_i^\top \boldsymbol{z} \le x_i + \epsilon. \tag{6}$$

Here the ReLU configuration is no longer explicitly reflected in the constraints. Instead, we only include the upper bound for each inner product $\boldsymbol{w}_i^\top \boldsymbol{z}$, which is always valid whether the ReLU is on or off. The previous requirement for the lower bound $\boldsymbol{w}_i^\top \boldsymbol{z} \ge x_i - \epsilon$ is now relaxed and hidden in the objective part. When the value of x_i is relatively large, the solver will produce a larger value of $\boldsymbol{w}_i^\top \boldsymbol{z}$ to achieve optimality. Since this value is also upper bounded by $x_i + \epsilon$, the optimal solution would be approaching to x_i if possible. On the other hand, when the value of x_i is close to 0, the objective dependence on $\boldsymbol{w}_i^\top \boldsymbol{z}$ is almost negligible.

Meanwhile, in the realizable case when $\exists \boldsymbol{z}^*$ such that $\text{ReLU}(W\boldsymbol{z}^*) = \boldsymbol{x}$, and $\epsilon = 0$, it is easy to show that the solution set for (6) is exactly the preimage of $\text{ReLU}(W\boldsymbol{z})$. This also trivially holds for Algorithm 1 and 2.

Relaxation for LeakyReLU: For LeakyReLU, similarly we take the following relaxation:

$$\max_{\boldsymbol{z}} \quad \boldsymbol{x}^\top W \boldsymbol{z} \tag{7}$$
$$\text{s.t.} \qquad 1/c \min\{x_i - \epsilon, 0\} \le \boldsymbol{w}_i^\top \boldsymbol{z} \le \max\{x_i + \epsilon, 0\}$$

Similarly when $\epsilon = 0$ and $\exists \boldsymbol{z}_0, \text{LeakyReLU}(W\boldsymbol{z}_0) = \boldsymbol{x}$, the solution to (7) is exactly $\boldsymbol{z}_0$.

B. Theoretical Analysis

NP-hardness to Invert a Binary Two-Layer Network: We show that the inversion could be reduced to MAX-3SAT problem: Given a 3-CNF formula ϕ (i.e. with at most 3 variables per clause), find an assignment that satisfies the largest number of clauses.

Proof of Theorem 1. We present an example of recovering a binary latent code from two-layer network that could be reduced to any MAX 3SAT problem, which is provably NP hard.

Write $\boldsymbol{b}_2 = [0| - \mathbb{1}] \in \mathbb{R}^n$, $W_2 = [\mathbb{1}|I]^\top \in \mathbb{R}^{n\times m}$ and observation $\boldsymbol{x} = [t|0] \in \mathbb{R}^n$, $n = m + 1$. As usual, we simplify the generator function as $G(\boldsymbol{z}) := \phi_2(\phi_1(\boldsymbol{x}))$. It's easy to see that all possible solutions for $\boldsymbol{z}_1 = \phi_2^{-1}(\boldsymbol{x})$ forms a polytope:

$$\sum_{i=1}^{n} (\boldsymbol{z}_1)_i = t$$
$$0 \le (\boldsymbol{z}_1)_i \le 1, \forall i \in [m] \tag{8}$$

From polytope (8) we could tell that the sparsity for feasible solution of $\boldsymbol{z}_1$ is t. Furthermore the polytope consists of all vectors of t 1's and $n - t$ 0's.

Let W_1 be a matrix such that each row consists of exact three non-zeros among two choices ± 1. Let $\boldsymbol{z}$ be the variables for the 3SAT problem, i.e. a one denotes a true value and a -1 a false value. Let all entries in $\boldsymbol{b}_1$ to be -2. Therefore every entry in $\boldsymbol{z}_1 = \text{ReLU}(W_1\boldsymbol{z} + \boldsymbol{b}_1)$ indicates the value of each clause. Only when the dot product of W_1's corresponding row with $\boldsymbol{x}$ is exactly 3, the clause if true and ϕ_1 will output 1, otherwise when the value is less than or equal to 2, it means the clause is false and ϕ_1 outputs 0.

In other words, the i-th clause to be true is equivalent to $\text{ReLU}((W_1)_i^\top \boldsymbol{x} + \boldsymbol{b}_1) = 1$, while the clause being false $\equiv \text{ReLU}((W_1)_i^\top \boldsymbol{x} + \boldsymbol{b}_1) = 0$.

When there is a polynomial algorithm to find a solution for $\phi_2(\phi_1(\boldsymbol{z})) = \boldsymbol{x}$, we find a solution that satisfies t clauses. Loop t over 1 through m one would get the maximum possible satisfiable clauses. (Notice with a 3-CNF formula $m = 5/3k$ and we will have a polynomial solution for MAX-3SAT.) Therefore the original problem is also NP-hard.

□

Proof of Non-convexity:
The following example demonstrate this property is no longer true for a two-layer case:

Example 1. *For $W_1 = [[1, 2], [3, 1]]$, $W_2 = [1, -1]$, and observation $x = 1$, the solution set for $G(\boldsymbol{z}) \equiv ReLU(W_2 ReLU(W_1\boldsymbol{z})) = x$ is non-convex.*

Example 1 is very straightforward to show the non-convexity of the preimage. Notice point $\boldsymbol{x}_1 = (-1, 1)$ and $\boldsymbol{x}_2 = (1, 3)$ are in the solution set, but their convex combination $x_3 = \frac{x_1+x_2}{2} = (0, 2)$ is not a solution point with $G(x_3) = 2$.

Proof of Exact Recovery for the Realizable Case:
The proof of Theorem 4 highly depends on the exact inversion for a single layer:

Lemma 1. *Under Assumption 1, a mapping $\phi(x) = ReLU(Wx), W \in \mathbb{R}^{n\times k}$ is injective with high probability $1 - exp(-\Omega(k))$. Namely, when $\phi(x) = \phi(y), x = y$.*

Proof. Notice for each i-th index, $(Wx)_i$ is positive w.p. $1/2$. Therefore, the number of positive coordinates in Wx, denoted by variable X, follows Binomial distribution $\sim \text{Bin}(n, p)$, where $n = c_0 k$ and $p = \frac{1}{2}$. With Hoeffding's inequality, $F(k; n, p) := \mathbb{P}(X \le k) < \exp(-2\frac{(np-k)^2}{n}) = \exp(-\Omega(k))$. Meanwhile, for a matrix with entries follow Gaussian distribution, with probability 1 it is invertible. Therefore ϕ^{-1} could only have unique solution if there is one. □

Within the proof of Lemma 1, we show that with high probability the observation $\boldsymbol{x} \in \mathbb{R}^n$ has at least k non-zero entries, meaning the original linear programming has at least

k equalities. Therefore the corresponding k rows forms an invertible matrix with high probability. Therefore simply by solving the linear equations we will attain the ground truth.

Proof of Theorem 2. From Lemma 1, for each layer $\phi_i : \mathbb{R}^{n_i\ 1} \to \mathbb{R}^{n_i}$, with probability $1 - exp(-\Omega(n_i))$, and for each observed $\boldsymbol{z}_i = \phi_i(\boldsymbol{z}^*_{i-1})$, by solving a linear system we are able to find $\boldsymbol{z}^*_{i-1}$. By union bound, failure in the whole layerwise inverting process is upper bounded by $\sum_{i=1}^{d} \exp(-\Omega(n_i)) = \exp(-\Omega(k))$, since $n_i > 2n_{i-1}$ for each i. □

B.1. ℓ_∞ error bound

With Assumption 2, we are able to show the following theorem that bounds the recovery error.

Theorem 4. *Let $\boldsymbol{x} = G(\boldsymbol{z}^*) + \boldsymbol{e}$ be a noisy observation produced by the generator G, a d-layer ReLU network mapping from $\mathbb{R}^k \to \mathbb{R}^n$. Let each weight matrix $W_i \in \mathbb{R}^{n_i\ 1 \times n_i}$ satisfies Assumption 2 with the integer $m_i > n_{i-1}$ and constant c_∞. Let the error $\boldsymbol{e}$ satisfies $\|\boldsymbol{e}\|_\infty \le \epsilon$, and for each $\boldsymbol{z}_i = \phi_i(\phi_{i-1}(\cdots\phi(\boldsymbol{z}^*)\cdots))$, at least m_i coordinates are larger than $\epsilon\frac{2^{d+1\ i}}{c_1^{d\ i}}$. Then by recursively applying Algorithm 1 backwards, it produces an $\boldsymbol{z}$ that satisfies $\|\boldsymbol{z} - \boldsymbol{z}^*\|_\infty \le \frac{2^d\epsilon}{c_1^d}$ with high probability $1 - exp(-\Omega(k))$.*

Proof of Approximate Recovery with ℓ_∞ and ℓ_1 Error Bound:
Theorem 4 depends on the layer-wise recovery of the intermediate ground truth vectors. We first present the following lemma for recovering a single layer with Algorithm 1 and then extend the findings to arbitrary depth d.

Lemma 2 (Approximate Inversion of a Noisy Layer with ℓ_∞ Error Bound)**.** *Given a noisy observation $\boldsymbol{x} = \phi(\boldsymbol{z}^*) := ReLU(W\boldsymbol{z}^*) + \boldsymbol{e}$. Let $\epsilon = \|\boldsymbol{e}\|_\infty$. If W satisfies Assumption 2 with the integer $m > k$, and the observation $\boldsymbol{z}^*$ has at least m coordinates that is larger than 2ϵ, then Algorithm 1 outputs an $\boldsymbol{z}$ that satisfies $\|\boldsymbol{z} - \boldsymbol{z}^*\|_\infty \le \frac{2\epsilon}{c_1}$ with high probability $1 - exp(-\Omega(k))$.*

Proof. Denote $I = \{i | x_i > \epsilon\}$, and $\boldsymbol{x}^* = \text{ReLU}(W\boldsymbol{z}^*)$ to be the true output. Notice it also satisfies $x^*_i > 0, \forall i \in I$ from the error bound assumption. Since $\boldsymbol{x}^*$ has more than m entries $\ge 2\epsilon$, the observation $\boldsymbol{x}$ satisfies $|I| \ge m$. Notice for a feasible vector $\boldsymbol{z}$ with constraints in (3), it satisfies that

$$\begin{aligned} & \|W_{I,:}\boldsymbol{z} - (\boldsymbol{x}^*)_I\|_\infty \\ \le\ & \|W_{I,:}\boldsymbol{z} - \boldsymbol{x}_I\|_\infty + \|\boldsymbol{x}_I - \boldsymbol{x}^*_I\|_\infty \le 2\epsilon, \end{aligned} \tag{9}$$

since the error is bounded uniformly for each coordinate in $\boldsymbol{x}^*$. Meanwhile, notice the real $\boldsymbol{z}^*$ satisfies $\phi_i(\boldsymbol{z}^*) = x^*_i, \forall i \in I$, we have $W_{I,:}\boldsymbol{z}^* = \boldsymbol{x}^*_I$. With Assumption 2, $W_{I,:}$ satisfies $\|W_{I,:}\boldsymbol{a}\|_\infty \ge c_\infty\|\boldsymbol{a}\|_\infty$ for an arbitrary $\boldsymbol{a}$ whp. Therefore together with (9) and let $\boldsymbol{a} = \boldsymbol{z} - \boldsymbol{z}^*$ and get:

$$c_\infty\|\boldsymbol{z} - \boldsymbol{z}^*\|_\infty \le \|W_I(\boldsymbol{z} - \boldsymbol{z}^*)\|_\infty \le 2\epsilon. \tag{10}$$

Therefore $\|\boldsymbol{z} - \boldsymbol{z}^*\|_\infty \le \frac{2\epsilon}{c_1}$ with probability $1 - \exp(\Omega(k))$. □

Theorem 4 is the direct extension to the multi-layer case and we simply apply Lemma 2 from d-th layer backwards to the input vector with initial ℓ_∞ error of $\epsilon(\frac{2}{c_1})^{d-i}$ for the i-th layer.

Now we look at some examples that fulfill the assumptions. The proof of ℓ_∞ extension is not easy and we look at the following looser result instead.

Lemma 3 (Related result from (Rudelson & Vershynin, 2009))**.** *For a sub-Gaussian random matrix A with height N and width n, where $N > 2n$. Its smallest singular value*

$$s_n(A) := \inf_{\|x\|_2 = 1} \|Ax\|_2.$$

satisfies $s_n(A) \ge c_2\sqrt{N}$ with high probability $1 - \exp(\Omega(n))$, where c_2 is some absolute constant.

The original paper requires $N > (1 + \Omega(\log^{-1}(n))n$ and we presented above with a relaxed condition that $N > 2n$.

Proof of Corollary 1. With the aid of Lemma 3, Assumption 2 is satisfied with $m = 2n_{i-1}$ for each layer with high probability. This is because for a random Gaussian matrix $A \in \mathbb{R}^{n\times k}$, $c_2\sqrt{n}\|\boldsymbol{z}\|_\infty \le c_2\sqrt{n}\|\boldsymbol{z}\|_2 \le \|A\boldsymbol{z}\|_2 \le \sqrt{n}\|A\boldsymbol{z}\|_\infty$ w.h.p. Without loss of generality we assume $c_2 \le 2$. We hereby only need to prove that for each i-th layer, $i \in [d]$, the output $\boldsymbol{z}^*_i = \phi_i(\phi_{i-1}(\cdots(\phi_1(\boldsymbol{z}^*))\cdots)) \in \mathbb{R}^{n_i}$ satisfies: $\sum_{j=1}^{n_i} \mathbb{1}_{(\boldsymbol{z}_i)_j > \frac{2^{d+1\ i}\epsilon}{c_2^{d\ i}}} > 2n_{i-1}$ with high probability. We start with the input layer. Notice each entry of $\boldsymbol{y} := W_1\boldsymbol{z}^*$ follows $\mathcal{N}(0, \sigma_1 = \|\boldsymbol{z}^*\|_2\sqrt{k})$, $\mathbb{P}(y_j > 2\frac{2^d\epsilon}{c_2^d}) \ge \mathbb{P}(y_j > \frac{\sigma_1}{8}) > 0.45$. Meanwhile, the number of coordinates in $\boldsymbol{y}$ that are larger or equal to $\frac{\sigma_1}{8}$ follows binomial distribution $\text{Bin}(n_1, p), p > 0.45$. Therefore the number of valid coordinates $\ge 0.45n_1 \ge 2k$ (since $n_{i+1} \ge 5n_i, \forall i$) with probability $1 - \exp(-\Omega(k))$. Afterwards since $c_2 < 1/2$ and $\frac{2^{d\ \ i+1}\epsilon}{c_2^{d\ \ i}}, i > 1$ is always smaller than $\frac{\epsilon}{c_2^d}$ and $\|\boldsymbol{z}^*_i\|_2 \ge \|\boldsymbol{z}^*\|_2$ with high probability since the network is expansive, the condition for the remaining layers is easier and also satisfied with probability at least $1 - \exp(-\Omega(n_{i-1}))$. By using union bound over all layers, the proof is complete. □

The proof for the ℓ_1 error bound analysis is similar to that of ℓ_∞ norm and we only show the essential difference. The key point in transmitting the error from next layer to previous

layer is as follows:

$$\begin{aligned}&\|W_{I,:}\boldsymbol{z}_{i-1} - (\boldsymbol{z}_i^*)_I\|_1\\ \leq&\|W_{I,:}\boldsymbol{z}_{i-1} - (\boldsymbol{z}_i)_I\|_1 + \|(\boldsymbol{z}_i)_I - (\boldsymbol{z}_i^*)_I\|_1\\ \leq&2\|(\boldsymbol{z}_i)_I - (\boldsymbol{z}_i^*)_I\|_1\end{aligned}$$

(Optimality of Algorithm 2 and $\boldsymbol{z}_{i\ 1}$ being a feasible point)
Together with Assumption 3, we have:

$$\begin{aligned}&\|W_{I,:}\boldsymbol{z}_{i-1} - (\boldsymbol{z}_i^*)_I\|_1 \geq c_1\|\boldsymbol{z}_{i-1} - \boldsymbol{z}_{i-1}^*\|_1\\ \Rightarrow&\|\boldsymbol{z}_{i-1} - \boldsymbol{z}_{i-1}^*\|_1 \leq \frac{2}{c_1}\|\boldsymbol{z}_i - \boldsymbol{z}_i^*\|_1.\end{aligned}$$

Here $\boldsymbol{z}_i^*$ is the ground truth of i-th intermediate vector. $\boldsymbol{z}_i$ is the one we observe and $\boldsymbol{z}_{i-1}$ is the solution Algorithm 2 produces.

C. More Experimental Results

We first present some details on how we setup the experimental settings for random nets:
For our methods, we choose the scaling factor $\alpha = 1.2$. With gradient descent, we use learning rate of 1 and up to 1,000 iterations or until the gradient norm is no more than 10^{-9}.

Model architecture: The architecture we choose in the simulation aligns with our theoretical findings. We choose a two layer network with constant expansion factor 5: latent dimension $k = 20$, hidden neurons of size 100 and observation dimension $n = 500$. The entries in the weight matrix are independently drawn from $\mathcal{N}(0, 1/n_i)$.

Noise generation: We use two kinds of random distribution to generate the noise, i.e., uniform distribution $U(-a, a)$ and Gaussian random noise $\mathcal{N}(0, a)$, in favor of the ℓ_0 and ℓ_1 error bound analysis respectively. We choose $a \in \{10^{-i} | i = 1, 2, \cdots 6\}$ for both noise types.

C.1. More Results on LP Relaxation

We formally present the relaxed version based on (6):

In Figure 4, we compare the performance with respect to different noise levels over all our proposals, including the results of Algorithm 3 that we omit in the main text. Although we do not see significant improvement of the LP relaxation method over our other proposals, we believe the relaxation over the strict ReLU configurations estimation is of good potential and should be more investigated in the future.

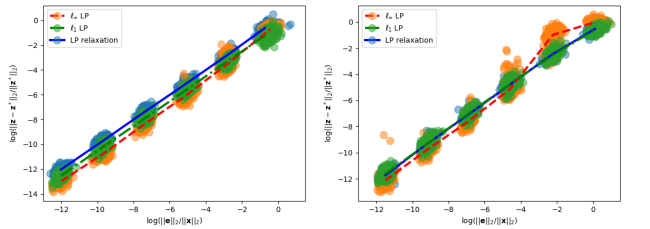
Time comparison on synthetic network:
Firstly, we should declare that for the very well-conditioned random weighted networks, gradient descent converges with large stepsize and we don't observe much supriority over GD in terms of the running time. In the table below we presented the running time for random net with different input dimensions ranging from 10 to 110. However, for MNIST dataset, the average running time for gradient descent to converge is roughly 1.2 minute, while for ℓ_0 LP it only takes no more than 0.5 second.

Algorithm 3 Relaxed Linear programming to invert a single layer (LP relaxation)

Input: Observation $\boldsymbol{x} \in \mathbb{R}^n$, weight matrix $W = [\boldsymbol{w}_1|\boldsymbol{w}_2|\cdots|\boldsymbol{w}_n]^\top$, initial error bound guess $\epsilon > 0$, scaling factor $\alpha > 1$.
for $t = 1, 2, \cdots$ **do**
  Solve the following linear programming:

$$\begin{aligned}\boldsymbol{z}^{(t)} \leftarrow \arg\max_{\boldsymbol{z}} &\sum_i \max\{0, x_i\}\boldsymbol{w}_i^\top \boldsymbol{z}\\ \text{s.t } &\boldsymbol{w}_i^\top \boldsymbol{z} \leq x_i + \epsilon\end{aligned}$$

  $\epsilon \leftarrow \epsilon\alpha$
  if $t > 2$ **and** $\exists \boldsymbol{z}^{(t-1)}$ feasible **and** $\|\phi(\boldsymbol{z}^{(t)}) - \boldsymbol{x}^*\|_1 \geq \|\phi(\boldsymbol{z}^{(t-1)}) - \boldsymbol{x}^*\|_1$ **then**
    return $\boldsymbol{z}^{(t-1)}$
  end if
end for



(a) Uniform Noise; Random Net (b) Gaussian Noise; Random Net

(c) Uniform Noise; Real Net (d) Gaussian Noise; Real Net

Figure 4. Comparison of our proposed methods (ℓ_{1} LP, ℓ_1 LP and LP relaxation). As can be shown, all three methods show no significant performance distinction. ℓ_{1} LP performs well in most cases except with large Gaussian noise.

k	10	30	50	70	90	110
ℓ_∞ LP	0.63	0.73	0.83	0.90	0.95	1.03
ℓ_1 LP	1.05	1.05	1.23	1.28	1.39	1.22
LP relaxation	0.66	0.53	0.58	0.76	0.75	0.70
GD	1.59	1.65	1.72	1.80	2.09	2.01

Table 1. Comparison of CPU time cost averaged from 200 runs, including LP relaxation.

D. Conclusion and Future Direction

We introduced a novel algorithm to invert a generative model through linear programming, one layer at a time, given (noisy) observations of its output. We prove that for expansive and random Gaussian networks, we can exactly recover the true latent code in the noiseless setting. For noisy observations we also establish provable performance bounds. Our work is different from the closely related (Huang et al., 2018) since we require less expansion, we bound for ℓ_1 and ℓ_∞ norm (as opposed to ℓ_2) but we are also limited to inversion, i.e. without a forward operator (while (Huang et al., 2018) can handle many natural forward operators as long as they satisfy a specific technical condition).

Empirically we demonstrate good performance, sometimes outperforming gradient descent when the latent vectors are high dimensional. We are interested in connecting our analysis to the framework of Approximate Message Passing (AMP) and its numerous extensions (Donoho et al., 2009; Schniter et al., 2016; Metzler et al., 2016) and possibly leverage from this body of theoretical work to improve our results.