We thank all the reviewers for their constructive comments and useful suggestions.

Q (R1): "Comparison with other methods like encoder" & "why do we need this technique"
A: This is a very important point that we need to clarify in our paper. GD inversion is not a straw man here: almost all the prior work on using generative models for solving inverse problems (see reference [3,4,10-14] in our paper) uses gradient descent as the main inversion technique so improving upon that is significant. There are also methods that train encoders or even end-to-end reconstruction methods from measurements, but are harder to train and are tuned to a specific inverse problem as opposed to a general method, see [3]. We will expand on this in the paper.

As compared to GD-based methods, our algorithm is much more efficient. GD costs on average 1.2 minutes to compress one single image (momentum makes the process faster but still slower than our method). While our algorithm takes only 0.5 second. See appendix for time comparisons.
We have compared the performance of GD with our method under different network architectures, i.e. different levels of expansiveness.
Our paper focuses on fundamental theoretical results, since there are very few in this area. In practice our technique can also be used as a good initialization for gradient-based methods. We verified this for DCGAN. See reply for R3.

Q (R1): On reductions *from* known NP-hard problem A: You are correct, of course. We fixed the confusing expression.

Q (R1): whether the binary case is representative for general case
A: Thank you for your comment. We were able to extend our proof from binary to general real-valued inputs. We achieve this by constraining an intermediate layer to be binary, using an additional output that enforces an intermediate layer to have binary values. Combining this with our previous argument establishes that it is NP-hard to invert a 4-layer network with real-valued inputs.

Specifically, we design a network $f : \mathbb{R}^k \to \mathbb{R}^2$ as follows: After input layer $\mathbf{z} \in \mathbb{R}^k$, we add 2 ReLU layers to make sure the output of second hidden layer $\mathbf{u} \in [-1, 1]$, i.e. $\mathbf{u} = \min\{\max\{\mathbf{z}, -1\}, 1\}$.[1] Afterwards, we copy the entire network we used for the binary proof to layer 3 (the original m hidden nodes as layer 3's first m nodes) and to the output layer $\mathbf{o}$ (the original scalar output as the first observation o_1) but add 2 more nodes to layer 3 and one node for output. The previous binary argument makes sure that if $\mathbf{u}$ has to be binary, we could solve 3SAT. Meanwhile, we let the two additional nodes on layer 3 to be $a = \sum_i \max\{u_i, 0\}$ and $b = \sum_i \min\{u_i, 0\}$ and the second observation o_2 to be $a + b$, which actually satisfies $a + b \equiv \sum_i |u_i|$. At inference time, we let this node o_2 to be equal to k. In this way to test for exact recovery, $\sum_{i=1}^k |u_i| = k$ and one has to let each u_i to be +1 or -1.

Therefore we show that for a 4-layer real network, it is NP-hard to determine if it could be exactly recovered for a given observation.

Q (R2): Comparison with 'invertibility of convolutional neural networks' or other RIP properties
A: We will add the discussion with (Gilbert et al.) and (Bourrier et al.) as suggested. Our work is substantially different from (Gilbert et al.) since they still work on linear mappings (convolutional layers without activation functions), while our work is targeting ReLU or LeakyReLU activations.
Thank you for the pointer of l_∞-RIP property. Typically RIP is for fat matrices with structured input while we deal with tall matrices. Also we only need the lower bound side of the RIP, so we redefined the condition in the paper. But indeed the transpose of the weight matrices should satisfy lower bound side of l_∞-RIP. We will add the discussions properly in the revised version.

Q (R2): the hypothesis that m_i coordinates need to be bounded away from zero is not natural
A: For the l_∞ case, we have shown that for random matrices, we do not explicitly need this requirement, as shown in Corollary 1. When the network is expansive and with random weights, there will be enough mass on the positive observations with high probability.

Q (R3): Additional Experiments on DCGAN:

Thank you for your suggestion. We trained a DCGAN architecture using MNIST data. We used 3 convolutional layers with ReLU activations that are expanding and the last layer being convolutional with sigmoid activation. We used projected gradient descent (PGD, gradient descent on the last layer, and projection over the first 3 layers) for a denoising task. We use our algorithm as an initialization for the projection step. Over multiple runs, we compare inversion using 1) GD, 2) GD with momentum, 3) PGD with random initialization (for the projection step), and 4) PGD with our scheme for initialization. The average relative error for each method was: 1) 0.26, 2) 0.25, 3) 0.17, and 4) 0.088. This additional experiment shows the benefits of our method for convolutional architectures. We will include more details in the final paper.

---

[1] Here $\max\{\mathbf{z}, -1\}$ could be achieved through $\mathrm{ReLU}(\mathbf{z} + 1) - 1$, and similarly for the $\min$ operation.