Optimization for Amortized Inverse Problems

Tianci Liu [*1] Tong Yang [*2] Quan Zhang [3] Qi Lei [4]

Abstract

Incorporating a deep generative model as the prior distribution in inverse problems has established substantial success in reconstructing images from corrupted observations. Notwithstanding, the existing optimization approaches use gradient descent largely without adapting to the non-convex nature of the problem and can be sensitive to initial values, impeding further performance improvement. In this paper, we propose an efficient amortized optimization scheme for inverse problems with a deep generative prior. Specifically, the optimization task with high degrees of difficulty is decomposed into optimizing a sequence of much easier ones. We provide a theoretical guarantee of the proposed algorithm and empirically validate it on different inverse problems. As a result, our approach outperforms baseline methods qualitatively and quantitatively by a large margin.

1. Introduction

Inverse problems aim to reconstruct the true image/signal $\boldsymbol{x}_T$ from a corrupted (noisy or lossy) observation

$$\boldsymbol{y} = f(\boldsymbol{x}_T) + \boldsymbol{e},$$

where f is a known forward operator and $\boldsymbol{e}$ is the noise. The problem is reduced to denoising when $f(\boldsymbol{x}) = \boldsymbol{x}$ is the identity map and is reduced to a compressed sensing (Candes et al., 2006; Donoho, 2006), inpainting (Vitoria et al., 2018), or a super-resolution problem (Menon et al., 2020) when $f(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x}$ and $\mathbf{A}$ maps $\boldsymbol{x}$ to an equal or lower dimensional space.

Inverse problems are generally ill-posed in the sense that there may exist infinitely many possible solutions, and thus require some natural signal priors to reconstruct the corrupted image (Ongie et al., 2020). Classical methods assume smoothness, sparsity in some basis, or other geometric properties on the image structures (Candes et al., 2006; Donoho, 2006; Danielyan et al., 2011; Yu et al., 2011). However, such assumptions may be too general and not task-specific. Recently, deep generative models, such as the generative adversarial network (GAN) and its variants (Goodfellow et al., 2014; Karras et al., 2017; 2019), are used as the prior of inverse problems after pre-training and have established great success (Bora et al., 2017; Hand & Voroninski, 2018; Hand et al., 2018; Asim et al., 2020b; Jalal et al., 2020; 2021). Compared to classical methods, using a GAN prior is able to produce better reconstructions at much fewer measurements (Bora et al., 2017).

Asim et al. (2020a) points out that a GAN prior can be prone to representation errors and significant performance degradation if the image to be recovered is out of the data distribution where the GAN is trained. To address this limitation, the authors propose replacing the GAN prior with normalizing flows (NFs) (Rezende & Mohamed, 2015). NFs are invertible generative models that learn a bijection between images and some base random variable such as standard Gaussian (Dinh et al., 2016; Kingma & Dhariwal, 2018; Papamakarios et al., 2021). Notably, the invertibility of NFs guarantees that any image is assigned with a valid probability, and NFs have shown higher degrees of robustness and better performance than GANs, especially on reconstructions of out-of-distribution images (Asim et al., 2020a; Whang et al., 2021a;b; Li & Denli, 2021; Hong et al., 2022). In this paper, we focus on inverse problems incorporating an NF model as the generative prior. We give more details of NFs in Section 2.

Conceptually, the aforementioned approaches can be seen as reconstructing an image as $\boldsymbol{x}^*$ defined by

$$\boldsymbol{x}^*(\lambda) \in \operatorname{argmin}_{\boldsymbol{x}} \mathcal{L}_{\text{recon}}(\boldsymbol{x}, \boldsymbol{y}) + \lambda \mathcal{L}_{\text{reg}}(\boldsymbol{x}), \qquad (1)$$

where $\mathcal{L}_{\text{recon}}$ is a reconstruction error between the observation $\boldsymbol{y}$ and a recovered image $\boldsymbol{x}$ (Bora et al., 2017; Ulyanov et al., 2018), and $\mathcal{L}_{\text{reg}}(\boldsymbol{x})$ multiplied by the hyperparmeter λ regularizes the reconstruction $\boldsymbol{x}$ using the prior information. Specifically, when a probabilistic deep generative prior, like an NF model, is used, $\mathcal{L}_{\text{reg}}(\boldsymbol{x})$ can be the likelihood for the generative model to synthesize $\boldsymbol{x}$. Note that the loss func-

[*]Equal contribution [1] Purdue University, United States [2] Peking University, China [3] Michigan State University, United States [4] New York University, United States. Correspondence to: Qi Lei <ql518@nyu.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

tion is subject to different noise models (Van Veen et al., 2018; Asim et al., 2020a; Whang et al., 2021a).

In execution, the reconstruction can be challenging as $\mathcal{L}_{\text{reg}}$ involves a deep generative prior. The success fundamentally relies on an effective optimization algorithm to find the global or a satisfactory local minimum of (1). However, the non-convex nature of inverse problems often makes gradient descent unprincipled and non-robust, e.g., to initialization. In fact, even in a simpler problem where the forward operator is the identity map (corresponding to a denoising problem), solving (1) with a deep generative prior is NP-hard as demonstrated in Lei et al. (2019). This establishes the complexity of solving inverse problems in general. On the other hand, even for specific cases, gradient descent possibly fails to find global optima, unlike training an (over-parameterized) neural network. This is because inverse problems require building a consistent (or under-parameterized) system and yielding a unique solution. It is known both theoretically and empirically that the more over-parameterized the system is, the easier it is to find the global minima with first-order methods (Jacot et al., 2018; Du et al., 2019; Allen-Zhu et al., 2019).

In this paper, we overcome the difficulty by proposing a new principled optimization scheme for inverse problems with a deep generative prior. Our algorithm incrementally optimizes the reconstruction conditioning on a sequence of λ's that are gradually increased from 0 to a prespecified value. Intuitively, suppose we have found a satisfactory solution (e.g., the global optimum) $\boldsymbol{x}^*(\lambda)$ as in (1). Then with a small increase $\Delta\lambda$ in the hyperparameter, the new solution $\boldsymbol{x}^*(\lambda+\Delta\lambda)$ should be close to $\boldsymbol{x}^*(\lambda)$ and easy to find if starting from $\boldsymbol{x}^*(\lambda)$. Our algorithm is related to amortized optimization (Amos, 2022) in that the difficulty and the high computing cost of finding $\boldsymbol{x}^*(\lambda)$ for the original inverse problem is amortized over a sequence of much easier tasks, where finding $\boldsymbol{x}^*(0)$ is feasible and the solution to one task facilitates solving the next. We refer to our method as Amortized Inverse Problem Optimization (AIPO).

It is noteworthy that AIPO is different from the amortized optimization in the conventional sense, which uses learning to approximate/predict the solutions to similar problems (Amos, 2022). In stark contrast, we are spreading the difficulty in solving the original problem into a sequence of much easier tasks, each of which is still the optimization of an inverse problem objective function. AIPO is also closely related to graduated optimization, which will be introduced and discussed in Section 2. We provide a theoretical underpinning of AIPO: Under some conventional assumptions, AIPO is guaranteed to find the global minimum. A practical and efficient algorithm is also provided. Empirically, our algorithm exhibits superior performance in minimizing the loss of various inverse problems, including denoising, noisy compressed sensing, and inpainting. To the best of our knowledge, AIPO is the first principled and efficient algorithm for solving inverse problems with a flow-based prior.

The paper proceeds as follows. In Section 2, we provide background knowledge in normalizing flows, amortized optimization, and graduated optimization. We also introduce example inverse problems. In Section 3, we formally propose AIPO and give theoretical analysis. In Section 4, we illustrate our algorithm and show its outstanding performance compared to conventional methods that have λ fixed during optimization. Section 5 concludes the paper. We defer the proofs, some technical details and experiment settings, and supplementary results to the appendix.

2. Backgrounds

We first provide an overview of normalizing flows that are used as the generative prior in our setting. We also briefly introduce amortized optimization based on learning and highlight its difference from the proposed AIPO. In addition, we showcase three representative inverse problem tasks, on which our algorithm will be evaluated.

2.1. Normalizing Flows

Normalizing flows (NFs) (Rezende & Mohamed, 2015; Papamakarios et al., 2021) are a family of generative models and capable of representing an n-dimensional complex distribution by transforming it to a simple base distribution (e.g., standard Gaussian or uniform distribution) of the same dimension. Compared to other generative models such as GAN (Goodfellow et al., 2014) and variational autoencoders (Kingma & Welling, 2013), NFs use a bijective (invertible) mapping and are computationally flexible in the sense that they admit sampling from the distribution efficiently and conduct exact likelihood estimation.

To be more specific, let $\boldsymbol{x} \in \mathbb{R}^n$ denote a data point that follows an unknown complex distribution and $\boldsymbol{z} \in \mathbb{R}^n$ follow some pre-specified based distribution such as a standard Gaussian. An NF model learns a differentiable bijective function $G : \mathbb{R}^n \to \mathbb{R}^n$ such that $\boldsymbol{x} = G(\boldsymbol{z})$. To sample from the data distribution $p_G(\boldsymbol{x})$, one can first generate $\boldsymbol{z} \sim p(\boldsymbol{z})$ and then apply the transformation $\boldsymbol{x} = G(\boldsymbol{z})$. Moreover, the invertibility of G allows one to use the change-of-variable formula to calculate the likelihood of $\boldsymbol{x}$ by

$$\log p_G(\boldsymbol{x}) = \log p(\boldsymbol{z}) + \log|\det(J_{G^{-1}}(\boldsymbol{x}))|,$$

where $J_{G^{-1}}$ denotes the Jacobian matrix of the inverse mapping G^{-1} evaluated at $\boldsymbol{x}$. To speed up the computation, G is usually composed of several simpler invertible functions that have triangular Jacobian matrices. Typical NFs include RealNVP (Dinh et al., 2016) and GLOW (Kingma & Dhariwal, 2018). For more details of NF models, we refer the readers to the review by Papamakarios et al. (2021) and the references therein.

2.2. Amortized Optimization Based on Learning

Amortized optimization methods based on learning are used to improve repeated solutions to the optimization problem

$$\boldsymbol{x}^\star(\boldsymbol{\lambda}) \in \operatorname{argmin}_{\boldsymbol{x}} g(\boldsymbol{x}; \boldsymbol{\lambda}), \tag{2}$$

where the non-convex objective $g : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ takes some context $\boldsymbol{\lambda} \in \mathcal{A}$ that can be continuous or discrete. The continuous, unconstrained domain of the problem is given by $\boldsymbol{x} \in \mathcal{X} = \mathbb{R}^n$, and the solution $\boldsymbol{x}^\star(\boldsymbol{\lambda})$, defined implicitly by the optimization process, is usually assumed to be unique. Given different $\boldsymbol{\lambda}$, instead of optimizing each $\boldsymbol{x}^\star(\boldsymbol{\lambda})$ separately, amortized optimization utilizes the similarities between subroutines induced by different $\boldsymbol{\lambda}$ to amortize the computational difficulty and cost across them and gets its name thereof. Typically, an amortized optimization method (Amos, 2022) solving (2) can be represented by

$$\mathcal{M} \triangleq (g, \mathcal{X}, \mathcal{A}, p(\boldsymbol{\lambda}), \hat{x}_\theta, \mathcal{L}),$$

where $g : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the unconstrained objective to optimize, $\mathcal{X}$ is the domain, $\mathcal{A}$ is the context space, $p(\boldsymbol{\lambda})$ is the probability distribution over contexts to optimize, $\hat{x}_\theta : \mathcal{A} \to \mathcal{X}$ is the amortized model parameterized by θ, which is learned by optimizing a loss $\mathcal{L}(g, \mathcal{X}, \mathcal{A}, p(\boldsymbol{\lambda}), \hat{x}_\theta)$ defined on all the components (Kim et al., 2018; Marino et al., 2018; Marino, 2021; Liu et al., 2022b).

Learning-based amortized optimization has been used in various machine learning models, including variational inference (Kingma & Welling, 2013), model-agnostic meta-learning (Finn et al., 2017), multi-task learning (Stanley et al., 2009), sparse coding (Chen et al., 2021), reinforcement learning (Ichnowski et al., 2021), and so on. For a more comprehensive survey of amortized optimization, we refer the readers to Amos (2022). Our proposed AIPO is different from the learning-based amortized optimization in two aspects. First, we decompose the task of an inverse problem into easier ones and still require optimization, rather than learning, to solve each subroutine problem. Second, the easier tasks in AIPO are not independent; the solution to one task is used as the initial value for the next and facilitates its optimization.

2.3. Graduated optimization

Graduated optimization (GO) is a prominent framework to solve a non-convex optimization problem and attempt to achieve the global optimum (Allgower & Georg, 2003). GO breaks the original problem of high difficulty into a sequence of optimization problems where the first problem is convex and the last one is the problem that one ultimately seeks to solve. The success of GO largely requires the sequence of optimizations so well designed that the global optimum of one optimization in the sequence is in the locally convex region around the global optimum of the next. Recently, Hazan et al. (2016) utilized random sampling to smooth a non-convex objective and obtain such a sequence of problems, where neural networks are used and a theoretical guarantee is provided under certain conditions. Such a strategy has been widely considered as a standard way of applying GO to deep learning (Khan et al., 2018; Allen-Zhu, 2018; Fan et al., 2018).
From the perspective of GO, the subroutines defined by AIPO are the sequence of optimization problems along which the degrees of non-convexity increase. AIPO differs from many existing GO frameworks in that each of its subroutines is a valid objective induced by a specific choice of the hyper-parameter. This allows one to readily obtain results from different hyperparameters: One can either retrieve the optimum at a smaller hyperparameter value or use the current estimation as an intermediate starting point to keep running the algorithm if a larger hyperparameter is preferred. In contrast, in many GO problems (Rose et al., 1990; Wu, 1996; Hazan et al., 2016; Yang et al., 2020), all of the intermediate problems are just surrogates, and only the solution to the last problem in the sequence makes sense and is collected. When a different (hyper)parameter value is wanted, one often has to rerun the full algorithm.

2.4. Representative Inverse Problems

We briefly introduce three representative inverse problems that we use to validate AIPO and refer the readers to Ongie et al. (2020) for recent progress using deep learning. Given an unknown clean image $\boldsymbol{x}_T$, we observe a corrupted measurement $\boldsymbol{y} = f(\boldsymbol{x}_T) + \boldsymbol{e}$, where $f : \mathbb{R}^n \to \mathbb{R}^m$ is some known forward operator such that $m \leq n$. The additive term $\boldsymbol{e} \in \mathbb{R}^m$ denotes some random noise that is usually assumed to have independent and identically distributed entries (Bora et al., 2017; Asim et al., 2020a). Representative inverse problem tasks include denoising, noisy compressed sensing, inpainting, and so on, with different forward operators f. In this work, we focus on the following three tasks.

Denoising assumes that $\boldsymbol{y} = \boldsymbol{x}_T + \boldsymbol{e}$ and noise $\boldsymbol{e} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ is an isotropic Gaussian vector (Asim et al., 2020a; Ongie et al., 2020).

Noisy Compressed Sensing (NCS) assumes that $\boldsymbol{y} = \mathbf{A}\boldsymbol{x}_T + \boldsymbol{e}$ where $\mathbf{A} \in \mathbb{R}^{m\times n}$, $m < n$, is a known $m \times n$ matrix of i.i.d. $\mathcal{N}(0, 1/m)$ entries (Bora et al., 2017; Asim et al., 2020a; Ongie et al., 2020), and the noise $\boldsymbol{e} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ is an isotropic Gaussian vector. Typically, the smaller m is, the more difficult the NCS task will be.

Inpainting assumes that $\boldsymbol{y} = \mathbf{A}\boldsymbol{x}_T + \boldsymbol{e}$ where $\mathbf{A} \in \mathbb{R}^{n\times n}$

is a diagonal matrix with binary entries and a certain proportion of them are zeros. In other words, $\mathbf{A}$ indicates whether a pixel is observed or missing (Asim et al., 2020a; Ongie et al., 2020). Again, we consider the noise $\boldsymbol{e} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$.

3. Methodology

We propose the Amortized Inverse Problem Optimization (AIPO) algorithm to reconstruct images by maximum a posterior estimation. First, we formulate the loss function for inverse problems using an NF generative prior. Then we introduce the AIPO algorithm and show the theoretical guarantee of its convergence.

3.1. Maximum A Posterior Estimation

Recent work on inverse problems using deep generative priors (Asim et al., 2020b; Whang et al., 2021a) has established successes in image reconstruction by a maximum a posterior (MAP) estimation. We adopt the same MAP formulation as in Whang et al. (2021a). Specifically, we use a pre-trained invertible NF model $G : \mathbb{R}^n \to \mathbb{R}^n$ as the prior that we can effectively sample from. G maps the latent variable $\boldsymbol{z} \in \mathbb{R}^n$ to an image $\boldsymbol{x} \in \mathbb{R}^n$ and induces a tractable distribution over $\boldsymbol{x}$ by the change-of-variable formula (Papamakarios et al., 2021). Optimization with respect to $\boldsymbol{x}$ or $\boldsymbol{z}$ has been considered in the literature (e.g., Asim et al., 2020a). In our context, they make no difference due to the invertibility of G, and thus we directly optimize $\boldsymbol{x}$ by minimizing the MAP loss. To be specific, denoting the prior density of $\boldsymbol{x}$ as $p_G(\boldsymbol{x})$, which quantifies how likely an image $\boldsymbol{x}$ is sampled from the pre-trained NF model, we reconstruct the image from $\boldsymbol{y}$ by the MAP estimation

$$\begin{aligned}\boldsymbol{x}^*(\lambda) &\in \operatorname{argmin}_{\boldsymbol{x}} \mathcal{L}_{\text{MAP}}(\boldsymbol{x}; \lambda) \qquad (3)\\ &= \operatorname{argmin}_{\boldsymbol{x}} \; -\log p_e(\boldsymbol{y} - f(\boldsymbol{x})) - \lambda \log p_G(\boldsymbol{x}),\end{aligned}$$

where the hyperparameter $\lambda > 0$ controls the weight of the prior, and p_e is the density of the noise $\boldsymbol{e}$. $-\log p_e(\boldsymbol{y} - f(\boldsymbol{x}))$ and $-\lambda \log p_G(\boldsymbol{x})$ are the reconstruction error and the regularization in (1), respectively, both of which are continuous in $\boldsymbol{x}$. In practice, p_e is usually assumed to be an isotropic Gaussian distribution (Bora et al., 2017; Asim et al., 2020a; Ongie et al., 2020) whose coordinates are independent and identically Gaussian distributed with a zero mean. Consequently, the loss function for the MAP estimation is equivalent to

$$\mathcal{L}_{\text{MAP}}(\boldsymbol{x}; \lambda) = \|\boldsymbol{y} - f(\boldsymbol{x})\|^2 - \lambda \log p_G(\boldsymbol{x}).$$

Note that the reconstruction error reaches its minimum value if and only if $\boldsymbol{y} = f(\boldsymbol{x})$. It is challenging to directly minimize $\mathcal{L}_{\text{MAP}}(\boldsymbol{x}; \lambda)$ given a prespecified λ in the presence of the deep generative prior p_G because of its non-convexity and NP-hardness (Lei et al., 2019). To effectively and efficiently solve the problem, we propose to amortize the difficulty and computing cost over a sequence of easier subroutine optimization and provide theoretical guarantees.

3.2. Amortized Optimization for MAP

We propose Amortized Inverse Problems Optimization (AIPO) for solving (3). Given a prespecified hyperparameter value Λ, to obtain a good approximation of $\boldsymbol{x}^*(\Lambda)$, we start from $\lambda = 0$, where the optimization may have an analytical solution $\boldsymbol{x}^*(0) \in \arg\min_{\boldsymbol{x}} \mathcal{L}_{\text{MAP}}(\boldsymbol{x}; 0) = f^{-1}(\boldsymbol{y})$, and gradually increase λ towards Λ in multiple steps. In each step, assuming that the current solution $\boldsymbol{x}^*(\lambda)$ is obtained and given a small enough $\Delta\lambda > 0$, we expect $\boldsymbol{x}^*(\lambda)$ to lie close to the solution $\boldsymbol{x}^*(\lambda + \Delta\lambda)$ in the next step under regular conditions (see Section 3.3, shortly). In other words, $\boldsymbol{x}^*(\lambda)$ is nearly optimal for minimizing the MAP loss $\mathcal{L}_{\text{MAP}}(\boldsymbol{x}; \lambda + \Delta\lambda)$. Consequently, minimizing $\mathcal{L}_{\text{MAP}}(\boldsymbol{x}; \lambda + \Delta\lambda)$ starting from $\boldsymbol{x}^*(\lambda)$ makes the optimization easier than starting from random initialization. In particular, we amortize the difficulty in directly solving $\boldsymbol{x}^*(\Lambda)$ over solving a sequence of optimization problems $\{\min_{\boldsymbol{x}} \mathcal{L}_{\text{MAP}}(\boldsymbol{x}; \lambda_{i+1} = \lambda_i + \Delta\lambda_i) \mid \boldsymbol{x}^*(\lambda_i)\}_i$.

Notably, the starting point $\boldsymbol{x}^*(0)$ corresponds to maximizing the log-likelihood of the noise $\boldsymbol{e}$ and equals to the maximum likelihood estimation (MLE). However, not all inverse problems admit a unique MLE. For under-determined f, there exist infinitely many choices of $\boldsymbol{x}^*(0)$ such that $f(\boldsymbol{x}^*(0)) = \boldsymbol{y}$ (e.g., in NCS and inpainting tasks), among which we choose the initial value of AIPO as the MLE $\boldsymbol{x}^*(0)$ defined by

$$\boldsymbol{x}^*(0) \in \operatorname{argmax}_{\boldsymbol{x}} p_G(\boldsymbol{x}), \text{ s.t. } f(\boldsymbol{x}) = \boldsymbol{y}. \qquad (4)$$

In practice, (4) can be solved by projected gradient descent (Boyd et al., 2004). Specifically, all the tasks we consider in this paper have a linear forward operator f; the constraints are in the affine space, and the projection can be readily solved as presented in Appendix A. We summarize AIPO in Algorithm 1. Note that its theoretical guarantee in Section 3.3, shortly, does not rely on the linearity of f.

Remark 3.1. In denoising tasks, existing literature largely uses $\boldsymbol{y}$ for initialization (Asim et al., 2020a; Whang et al., 2021a) and can be regarded as a special case of our method by taking one large step from $\lambda = 0$ with $\Delta\lambda = \Lambda$.

3.3. Theoretical Analysis

We provide a theoretical analysis of the convergence of AIPO. We make the following assumptions, under which we prove that AIPO by Algorithm 1 finds an approximation of the global minimum of $\mathcal{L}_{\text{MAP}}$ with arbitrary precision.

Assumption 3.2 (L-smoothness of $\mathcal{L}_{\text{MAP}}$)**.** There exists $L > 0$ such that $\forall \lambda \in [0, \Lambda]$, $\mathcal{L}_{\text{MAP}}(\cdot; \lambda)$ is L-smooth, i.e., for all $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, $\|\nabla_{\boldsymbol{x}} \mathcal{L}_{\text{MAP}}(\boldsymbol{x}_1; \lambda) - \nabla_{\boldsymbol{x}} \mathcal{L}_{\text{MAP}}(\boldsymbol{x}_2; \lambda)\| \leq L\,\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$.

Algorithm 1 AIPO algorithm

```
1: Input: $\Lambda > 0$, generative model $G : \mathbb{R}^n \to \mathbb{R}^n$, $L > 0$ (see Assumption 3.2), $\sigma > 0, \delta > 0$ (see Assumption 3.4), $C > 0$ (see Assumption 3.3), precision $\varepsilon > 0$
2: Initialize: $\lambda = 0$, $\mu = \frac{1}{2L\sigma^2}$, $\delta_0 = \min\left\{\delta, \frac{\mu}{\sqrt{2(L-\mu)L}}\delta\right\}$, $N = [\frac{2\Lambda C}{\delta_0}] + 1$, $r = \frac{\delta_0}{2\sigma}$
3: Find the MLE $\boldsymbol{x}_0 = \boldsymbol{x}^*(0)$ by solving (4)
4: for $i = 0, \dots, N-1$ do
5:     $\lambda = \lambda + \frac{\Lambda}{N}$
6:     $K = [\frac{2\log(2\delta/\delta_0)}{\log(L/(L-\mu))}] + 1$
7:     if $i = N-1$ then
8:         $K = \max\{0, [\frac{2\log(2\delta/\varepsilon)}{\log(L/(L-\mu))}] + 1\}$
9:         $r = \frac{\varepsilon}{\sigma}$
10:    end if
11:    for $k = 1, \dots, K$ do
12:        if $\|\nabla_{\boldsymbol{x}}\mathcal{L}_{\text{MAP}}(\boldsymbol{x}_k, \lambda)\| \le r$ then
13:            break
14:        else
15:            $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{1}{L}\nabla_{\boldsymbol{x}}\mathcal{L}_{\text{MAP}}(\boldsymbol{x}_k, \lambda)$
16:        end if
17:    end for
18:    $\boldsymbol{x}_0 = \boldsymbol{x}_{K+1}$
19: end for
output $\hat{x}(\Lambda) = \boldsymbol{x}_0$
```

Assumption 3.3 (C-smoothness of $\boldsymbol{x}^*(\lambda)$)**.** For all $\lambda \in (0, \Lambda]$, $\boldsymbol{x}^*(\lambda)$ is unique, and there exists $C > 0$ such that for all $\lambda_1, \lambda_2 \in (0, \Lambda]$, $\|\boldsymbol{x}^*(\lambda_1) - \boldsymbol{x}^*(\lambda_2)\| \le C|\lambda_1 - \lambda_2|$.

Assumption 3.4 (local property of $\nabla_{\boldsymbol{x}}\mathcal{L}_{\text{MAP}}$)**.** There exists $\sigma > 0$ and δ such that for all $\lambda \in [\frac{\delta_0}{2C+\delta_0/\Lambda}, \Lambda]$ (where C is defined in Assumption 3.3 and δ_0 is defined in line 2 of Algorithm 1) and $\boldsymbol{x} \in B(\boldsymbol{x}^\star(\lambda), \delta) := \{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x}^\star(\lambda)\| \le \delta\}$, we have $\|\boldsymbol{x} - \boldsymbol{x}^\star(\lambda)\| \le \sigma\|\nabla_{\boldsymbol{x}}\mathcal{L}_{\text{MAP}}(\boldsymbol{x}; \lambda)\|$.

Remark 3.5. Smoothness assumptions like Assumptions 3.2 and 3.3 are commonly used in convergence analysis. See, for example, Song et al. (2019, Assumption 3.2), Zhou et al. (2019, Assumption 1), and Scaman et al. (2022).

In Assumption 3.4, the reason we set the lower bound of λ to be $\frac{\delta_0}{2C+\delta_0/\Lambda}$ is discussed in the proof of Theorem 3.6 in Appendix B. Besides, We make two comments on Assumption 3.4:

(i) Local strong convexity around the minima of the loss is a widely-adopted assumption in deep learning literature, e.g., Li & Yuan (2017, Definition 2.4), Whang et al. (2020, Theorem 4.1), and Safran et al. (2021, Definition 5). Our Assumption 3.4 is weaker than the local strong convexity. Reversely, if there exists $\mu' > 0$ and $\delta > 0$ such that for all $\lambda \in [\frac{\delta_0}{2C+\delta_0/\Lambda}, \Lambda]$, $\mathcal{L}_{\text{MAP}}(\cdot; \lambda)$ is μ'-strongly convex on $B(\boldsymbol{x}^*(\lambda), \delta)$, then Assumption 3.4 holds with $\sigma = 2/\mu'$. More Details about this comment can be found in Appendix B.1.

(ii) If Assumptions 3.2 and 3.4 hold, then for all $\lambda \in [\frac{\delta_0}{2C+\delta_0/\Lambda}, \Lambda]$ and $\boldsymbol{x} \in B(\boldsymbol{x}^*(\lambda), \delta)$, we have

$$\mathcal{L}_{\text{MAP}}(\boldsymbol{x}; \lambda) - \mathcal{L}_{\text{MAP}}(\boldsymbol{x}^*(\lambda); \lambda) \le \frac{1}{2\mu}\|\nabla_{\boldsymbol{x}}\mathcal{L}_{\text{MAP}}(\boldsymbol{x}; \lambda)\|^2,$$

where $\mu = \frac{1}{2L\sigma^2}$. More details about this comment can be found in Appendix. B.1.

The local Polyak-Lojasiewicz property is previously used to characterize the local optimization landscape for training neural networks (Song et al., 2021; Karimi et al., 2016; Liu et al., 2022a). It has been shown that wide neural networks satisfy the local Polyak-Lojasiewicz property under mild assumptions (Liu et al., 2020, Theorem 7.2), and our Assumption 3.4 is stronger as it implies the local Polyak-Lojasiewicz property.

Now we present our main result.

Theorem 3.6. *Under Assumptions 3.2, 3.3, and 3.4, for all $\varepsilon > 0$, Algorithm 1 returns $\hat{\boldsymbol{x}}(\Lambda)$ that satisfies $\|\hat{\boldsymbol{x}}(\Lambda) - \boldsymbol{x}^*(\Lambda)\| \le \varepsilon$.*

Note that Theorem 3.6 ensures that Algorithm 1 for AIPO finds an ε-approximate point of the **global** minimum of $\mathcal{L}_{\text{MAP}}(\cdot; \Lambda)$, whose reconstruction guarantee has been well-studied (Vershynin, 2015; Bora et al., 2017; Whang et al., 2020; Asim et al., 2020a). More specifically, Theorem 3.6 together with Theorem 4.2 from Whang et al. (2020) guarantee AIPO to reconstruct the ground truth under the respective assumptions. We give a proof sketch of Theorem 3.6 here and defer the formal proof to Appendix B.

Proof sketch. Starting from the global minimum when $\lambda = 0$, our algorithm ensures that the i-th outer iteration learns $\boldsymbol{x}^*(i\Lambda/N)$ approximately and serves as a good initialization for the next target $\boldsymbol{x}^*((i+1)\Lambda/N)$. Note that in each iteration we incrementally grow λ from $i\Lambda/N$ to $(i+1)\Lambda/N$ until reaching our target Λ. Specifically, Assumptions 3.3 and 3.4, respectively, ensure $\boldsymbol{x}^*(i\Lambda/N)$ to be close enough to $\boldsymbol{x}^*((i+1)\Lambda/N)$ and that the first order algorithm can find $\boldsymbol{x}^*((i+1)\Lambda/N)$ from the good initialization obtained in the last iteration. □

Notably, given the MLE solution that is a global optimum when $\lambda = 0$, Algorithm1 converges to a ε-approximate global optimum linearly within $(N-1) \times K + K_{\text{final}} = \mathcal{O}(\log(1/\varepsilon))$ steps, while standard optimization algorithms (e.g., with random initialization) can only be expected to converge to some local optimum. To avoid specifying the parameters in the assumptions (L, σ, δ, C) and the precision ε, we further provide an efficient and practical implementation of AIPO in Algorithm 2 in Appendix C, where the scheme for hyperparameter increment is more practical and data-adaptive.

4. Experiments

In this section, we evaluate the performance of the proposed algorithm on three inverse problem tasks, including denoising, noisy compressed sensing, and inpainting. We use two normalizing flow models as the generative prior, both of which work well, to justify that our algorithm is a general framework and does not require model-specific adaption. Note that using deep generative priors in inverse problems has been demonstrated to outperform classical approaches (Bora et al., 2017; Asim et al., 2020a; Whang et al., 2021a). We focus on illustrating the advantage of AIPO over conventional optimizations without the amortization scheme and skip the comparison with classical approaches.

4.1. Setup

We use two commonly used normalizing flow models, RealNVP (Dinh et al., 2016) and GLOW (Kingma & Dhariwal, 2018), respectively, as the generative prior G. The two models are trained on the CelebA dataset (Liu et al., 2015), and we follow the pertaining suggestions by Asim et al. (2020a) and Whang et al. (2021a), to which we refer the readers for the model architecture and technical details.

Our experiments consist of two sets of data. One is in-distribution samples that are randomly selected from the CelebA test set. The other is out-of-distribution (OOD) images that contain human or human-like objects. Due to budget constraints, we run the experiments on 200 in-distribution and 7 OOD samples. For baseline algorithms, we consider minimizing the MAP loss as in equation (3) by gradient-descent based optimizer with random or zero initialization that is widely used in literature (Bora et al., 2017; Asim et al., 2020a; Whang et al., 2021a). Concretely, random initialization first draws a Gaussian random vector $\boldsymbol{z}_0$ and uses $\boldsymbol{x}_0 = G(\boldsymbol{z}_0)$ as the initial value. Zero initialization takes $\boldsymbol{z}_0 = \mathbf{0}$ and initializes $\boldsymbol{x}_0 = G(\boldsymbol{z}_0)$. Furthermore, to demonstrate that AIPO's outperformance indeed results from amortization rather than merely the MLE initialization, gradient-descent based optimizer with the MLE initialization is used as the third baseline approach.

All the three baselines have λ fixed throughout the optimization process. All the algorithms optimize $\boldsymbol{x}$ with Adam (Kingma & Ba, 2014), and we assign an equal computing budget to all the approaches compared in each subsection. Our AIPO and the baseline algorithm with the MLE initialization require a solution to (4) on the NCS and inpainting tasks, where we run 500 iterations of projected gradient descent. In these cases, we assign an extra computing budget of the same amount to the baseline algorithms with the random and zero initialization.

In all the experiments, we compare the algorithms with a prespecified λ, which is set to be $0.3, 0.5, 1.0, 1.5, 2.0$, respectively. These values form a fairly large range for the hyperparameter (Asim et al., 2020a; Whang et al., 2021a). When evaluating an algorithm's performance, we calculate the MAP loss that the algorithm tries to minimize as defined in equation (3). To further show the practical benefit of our algorithm, we also report Peak-Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), whose large values indicate better reconstruction. Moreover, the reconstructed images are also presented for illustration. Due to the page limit, we only report the MAP loss values and visualize reconstructions using one generative prior per task. We defer PSNRs, SSIMs, and more visualization to the appendix.

4.2. Denoising

Denoising tasks assume that $\boldsymbol{y} = \boldsymbol{x}+\boldsymbol{e}$, and we sample noise $\boldsymbol{e}$ from two distributions $\mathcal{N}(0, 0.1^2\mathbf{I})$ and $\mathcal{N}(0, 0.05^2\mathbf{I})$ following Bora et al. (2017); Asim et al. (2020a). We report the corresponding MAP losses in Table 1. As a result, AIPO consistently outperforms the three baselines regardless of which generative prior is used. In particular, when λ is set to be large (1.5 or 2.0), the AIPO's outperformance is especially remarkable. This is primarily because the non-convex p_G weighs more in the MAP loss with a larger λ, increasing the difficulty in conventional optimization without the amortization. In stark contrast, AIPO is less sensitive to λ as it amortizes the difficulty over much simpler tasks. Consequently, given the same budget, AIPO delivers much smaller loss values. It is also the case in the NCS and inpainting tasks as shown in the following subsections.

We visualize the reconstruction quality by showing the clean, observed, and reconstructed images for $\sigma = 0.1$. For this task, we report results from RealNVP in Figure 1. We set the hyperparameter $\lambda = 0.5$ such that all four algorithms achieve their highest PSNRs (see Tables 5 and 7 in Appendix D for details). We also present the full version of Figure 1, reconstructions from GLOW, and their corresponding PSNRs in Appendix D. Our method preserves the details of the images and removes more noise in the background. See, for instance, the second and the fourth columns in Figure 1. The reconstructions together with the consistently lower loss values demonstrate the success of our proposed approach.

4.3. Noisy Compressed Sensing (NCS)

Our second task is NCS in the form of $\boldsymbol{y} = \mathbf{A}\boldsymbol{x} + \boldsymbol{e}$, where $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a known matrix of i.i.d. Gaussian entries with $m < n = 12288$. The smaller m is, the more difficult the NCS task will be. We consider $m \in \{1000, 2000, 4000\}$. Previous works (Bora et al., 2017; Asim et al., 2020a) sample $\boldsymbol{e}$ from Gaussian with the restrictions $\mathbb{E}[\boldsymbol{e}] = \mathbf{0}$ and $\mathbb{E}[\|\boldsymbol{e}\|] = 0.1$. In this setting, however, the scale of the

Table 1. MAP loss (mean±se) from solving denoising tasks with different algorithms under two generative priors. Lower is better.

		λ	Ours.	MLE init.	Random init.	Zero init.
			$\sigma = 0.05$			
RealNVP	Test	0.3	**-19141** ± **16**	-19045 ± 16	-19061 ± 16	-19063 ± 17
		0.5	**-17526** ± **31**	-17189 ± 32	-17201 ± 32	-17199 ± 32
		1.0	**-14930** ± **70**	-13701 ± 67	-13625 ± 71	-13778 ± 62
		1.5	**-12758** ± **101**	-10704 ± 102	-10734 ± 106	-10677 ± 98
		2.0	**-10610** ± **129**	-7909 ± 137	-7666 ± 139	-7787 ± 143
	OOD	0.3	**-18907** ± **81**	-18684 ± 93	-18666 ± 95	-18687 ± 98
		0.5	**-17055** ± **175**	-16555 ± 244	-16647 ± 221	-16587 ± 217
		1.0	**-14066** ± **343**	-12449 ± 559	-12405 ± 393	-12850 ± 402
		1.5	**-11293** ± **515**	-8958 ± 771	-9075 ± 570	-8677 ± 604
		2.0	**-9041** ± **548**	-5817 ± 740	-5333 ± 975	-6360 ± 1103
GLOW	Test	0.3	**-19477** ± **18**	-19444 ± 18	-19438 ± 18	-19440 ± 18
		0.5	**-18336** ± **37**	-18165 ± 35	-18160 ± 36	-18162 ± 36
		1.0	**-16767** ± **78**	-16220 ± 72	-16205 ± 73	-16202 ± 72
		1.5	**-15657** ± **112**	-14767 ± 103	-14793 ± 106	-14756 ± 103
		2.0	**-14710** ± **138**	-13608 ± 133	-13599 ± 133	-13576 ± 132
	OOD	0.3	**-19367** ± **199**	-19301 ± 208	-19363 ± 196	-19342 ± 200
		0.5	**-18107** ± **413**	-17823 ± 397	-17877 ± 380	-17895 ± 361
		1.0	**-16224** ± **772**	-15555 ± 669	-15590 ± 669	-15616 ± 671
		1.5	**-14931** ± **970**	-14177 ± 874	-13989 ± 792	-14407 ± 833
		2.0	**-14759** ± **1266**	-13027 ± 1164	-12946 ± 898	-12864 ± 1071
			$\sigma = 0.1$			
RealNVP	Test	0.3	**-9956** ± **19**	-9732 ± 20	-9735 ± 21	-9729 ± 20
		0.5	**-8909** ± **30**	-8376 ± 32	-8357 ± 33	-8385 ± 33
		1.0	**-6893** ± **61**	-5530 ± 65	-5569 ± 65	-5527 ± 67
		1.5	**-4830** ± **95**	-2803 ± 105	-2850 ± 106	-2571 ± 103
		2.0	**-2619** ± **148**	-266 ± 134	-330 ± 129	415 ± 143
	OOD	0.3	**-9680** ± **49**	-9464 ± 93	-9451 ± 91	-9513 ± 94
		0.5	**-8713** ± **110**	-7917 ± 262	-7924 ± 191	-7862 ± 242
		1.0	**-6444** ± **429**	-4413 ± 566	-4610 ± 442	-4751 ± 559
		1.5	**-4708** ± **920**	-1725 ± 829	-1287 ± 793	-1611 ± 760
		2.0	**-1669** ± **1567**	1320 ± 970	1794 ± 857	1436 ± 949
GLOW	Test	0.3	**-10368** ± **20**	-10286 ± 19	-10293 ± 20	-10294 ± 19
		0.5	**-9902** ± **34**	-9657 ± 33	-9646 ± 33	-9626 ± 33
		1.0	**-9232** ± **62**	-8653 ± 57	-8614 ± 58	-8646 ± 59
		1.5	**-8893** ± **87**	-8011 ± 78	-7940 ± 79	-7912 ± 80
		2.0	**-8851** ± **107**	-7485 ± 96	-7461 ± 98	-7409 ± 108
	OOD	0.3	**-10329** ± **181**	-10187 ± 211	-10266 ± 182	-10242 ± 177
		0.5	**-9843** ± **331**	-9495 ± 308	-9421 ± 198	-9465 ± 238
		1.0	**-9122** ± **454**	-8499 ± 464	-8721 ± 379	-8468 ± 410
		1.5	**-9254** ± **584**	-7841 ± 598	-8005 ± 540	-8072 ± 492
		2.0	**-9409** ± **756**	-7673 ± 640	-6978 ± 727	-7393 ± 583

noise is not big enough to break down the algorithms in our experiments such that their performances are comparable. So in our experiment, we let $\boldsymbol{e} \sim \mathcal{N}(\mathbf{0}, 0.1^2\mathbf{I})$, inducing larger noise and a more challenging NCS task.

We report the MAP loss values from different algorithms in Table 2. For all the NCS tasks in the setting of different generative priors and values of m and λ, our proposed AIPO consistently delivers much lower loss values compared to the benchmarks. Analogous to the denoising tasks, AIPO outperforms the other algorithms by a larger margin in the presence of a bigger λ.

We visualize the image reconstructions using the GLOW prior for $m = 4000$ in Figure 2 and using the RealNVP prior for $m = 2000$ in Figure 3, due to page limits we defer reconstructions for $m = 1000$ to Appendix D. The corresponding hyperparameters are 0.5 and 0.3, respectively. Unlike denoising and inpainting (in the next subsection),

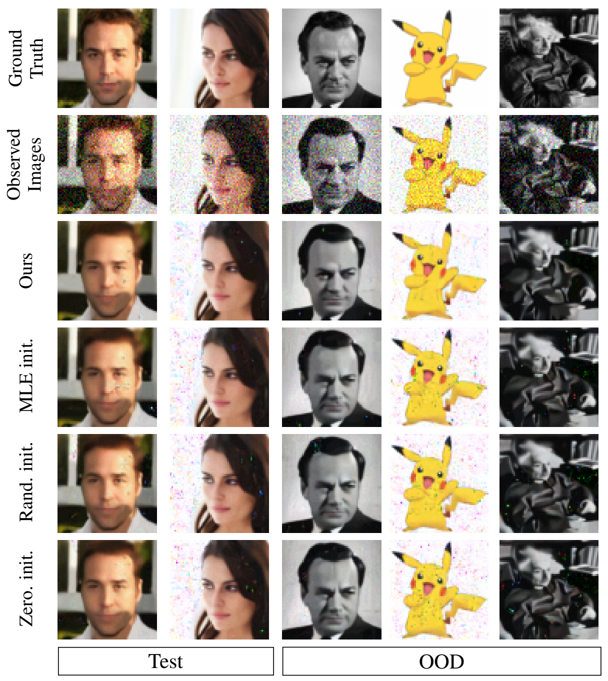

Figure 1. Reconstructions from denoising Gaussian noise with $\sigma = 0.1$ on CelebA faces and out-of-distribution images with RealNVP as the generative prior. Hyperparameter $\lambda = 0.5$.

NCS does not have conceptually meaningful corrupted observations. So we omit to visualize them. Our algorithm is able to achieve better reconstruction qualities than the baselines, in that much less noise occurs in the background of the reconstructed images. Moreover, compared to random and zero initialization, the MLE initialization does not exhibit consistently lower losses or consistently better reconstructions. This implies that the success of our AIPO does not solely rely on the MLE initialization. Instead, the amortized optimization scheme contributes more to the improvement.

4.4. Inpainting

We consider inpainting tasks that assume $\boldsymbol{y} = \mathbf{A}\boldsymbol{x} + \boldsymbol{e}$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with binary elements indicating whether a pixel is observed or missing. In our experiments, we randomly mask several 4×4 pixel blocks such that in total the masked portions are 25% and 30% respectively. We sample $\boldsymbol{e} \sim \mathcal{N}(0, 0.02^2\mathbf{I})$ as the noise. Table 3 reports the MAP loss from different algorithms. Under all the settings, our AIPO delivers much lower loss values. Again, the larger λ is, the more apparent outperformance AIPO exhibits.

We show the true, observed, and reconstructed images with the GLOW prior in Figure 4. The hyperparameter λ is 1.5 such that all the algorithms achieve their best PSNRs. Specifically, for column 2, our algorithm removes all the masks whereas others fail to do so. For column 4, AIPO is the only one that recovers the main object of the image.

Table 2. MAP loss (mean±se) of solving NCS tasks with different algorithms under two generative priors. A smaller value is better.

		λ	Ours.	MLE init.	Random init.	Zero init.
			$m = 4000$			
RealNVP	Test	0.3	**-2650** ± **19**	-2316 ± 20	-2315 ± 19	-2321 ± 19
		0.5	**-1686** ± **30**	-1084 ± 32	-1006 ± 33	-1024 ± 33
		1.0	**363** ± **61**	1814 ± 64	1915 ± 63	1878 ± 65
		1.5	**2310** ± **97**	4377 ± 98	4532 ± 100	5031 ± 96
		2.0	**4336** ± **126**	6756 ± 124	7296 ± 128	7237 ± 128
	OOD	0.3	**-2433** ± **84**	-2088 ± 148	-2058 ± 108	-2043 ± 123
		0.5	**-1188** ± **152**	-604 ± 256	-679 ± 222	-598 ± 282
		1.0	**818** ± **604**	2599 ± 611	2806 ± 615	2040 ± 482
		1.5	**3969** ± **1297**	6052 ± 1011	5437 ± 922	5708 ± 762
		2.0	**6275** ± **1485**	7830 ± 1252	9120 ± 1064	8645 ± 1044
GLOW	Test	0.3	**-3110** ± **19**	-2915 ± 19	-2901 ± 18	-2895 ± 18
		0.5	**-2619** ± **33**	-2355 ± 31	-2358 ± 31	-2364 ± 31
		1.0	**-1949** ± **62**	-1350 ± 56	-1392 ± 56	-1392 ± 56
		1.5	**-1662** ± **85**	-693 ± 81	-728 ± 79	-713 ± 80
		2.0	**-1546** ± **105**	-151 ± 98	-158 ± 96	-58 ± 93
	OOD	0.3	**-3076** ± **201**	-2864 ± 185	-2846 ± 172	-2825 ± 191
		0.5	**-2647** ± **319**	-2306 ± 284	-2252 ± 257	-2349 ± 277
		1.0	**-2036** ± **457**	-1202 ± 475	-1506 ± 511	-1465 ± 452
		1.5	**-1563** ± **676**	-783 ± 687	-869 ± 534	-997 ± 588
		2.0	**-1610** ± **782**	-62 ± 758	-322 ± 740	-513 ± 650
			$m = 2000$			
RealNVP	Test	0.3	**-841** ± **18**	-498 ± 20	-486 ± 19	-465 ± 19
		0.5	**43** ± **31**	733 ± 32	816 ± 32	751 ± 33
		1.0	**2027** ± **65**	3436 ± 65	3615 ± 65	3679 ± 68
		1.5	**3953** ± **99**	6124 ± 97	6349 ± 96	6408 ± 111
		2.0	**6422** ± **131**	8484 ± 146	9076 ± 145	8761 ± 136
	OOD	0.3	**-566** ± **132**	-227 ± 146	-203 ± 122	-207 ± 146
		0.5	**184** ± **178**	1295 ± 300	1158 ± 278	1385 ± 327
		1.0	**3062** ± **779**	4341 ± 763	4624 ± 694	4169 ± 627
		1.5	**5100** ± **1063**	7660 ± 1002	7021 ± 1021	7072 ± 748
		2.0	**9478** ± **1590**	10774 ± 1683	10142 ± 1529	10315 ± 1235
GLOW	Test	0.3	**-462** ± **20**	1689 ± 70	445 ± 31	483 ± 32
		0.5	**-154** ± **34**	3508 ± 114	1457 ± 54	1527 ± 54
		1.0	**306** ± **57**	7560 ± 226	3278 ± 107	3437 ± 112
		1.5	**537** ± **84**	11238 ± 340	4690 ± 156	4668 ± 149
		2.0	**566** ± **108**	14375 ± 466	5578 ± 189	5826 ± 186
	OOD	0.3	**-485** ± **158**	2033 ± 538	801 ± 431	758 ± 456
		0.5	**-161** ± **256**	4212 ± 829	2181 ± 870	2163 ± 841
		1.0	**231** ± **417**	8166 ± 1617	3726 ± 1599	3854 ± 1638
		1.5	**-98** ± **692**	11542 ± 1789	3698 ± 938	4848 ± 1922
		2.0	**312** ± **709**	16312 ± 2783	6119 ± 2330	7970 ± 3126
			$m = 1000$			
RealNVP	Test	0.3	**74** ± **17**	474 ± 18	487 ± 20	498 ± 20
		0.5	**956** ± **30**	1660 ± 33	1602 ± 34	1712 ± 34
		1.0	**2822** ± **63**	4261 ± 67	4447 ± 61	4561 ± 76
		1.5	**5259** ± **91**	6825 ± 104	6937 ± 105	7180 ± 101
		2.0	**7124** ± **123**	9269 ± 138	9668 ± 138	9715 ± 143
	OOD	0.3	**392** ± **149**	886 ± 287	616 ± 203	674 ± 216
		0.5	**1476** ± **337**	2025 ± 391	2024 ± 314	1840 ± 314
		1.0	**4097** ± **781**	5260 ± 797	5084 ± 728	5424 ± 564
		1.5	**6729** ± **1291**	8431 ± 1132	7828 ± 945	8063 ± 1125
		2.0	**9688** ± **1539**	11164 ± 1775	10130 ± 1164	10494 ± 1464
GLOW	Test	0.3	**-1378** ± **20**	-867 ± 28	-986 ± 23	-958 ± 24
		0.5	**-922** ± **32**	-295 ± 39	-402 ± 36	-399 ± 36
		1.0	**-355** ± **56**	599 ± 59	412 ± 54	440 ± 55
		1.5	**-117** ± **85**	1165 ± 77	1168 ± 77	1183 ± 76
		2.0	**-62** ± **103**	1812 ± 105	1743 ± 102	1756 ± 102
	OOD	0.3	**-1323** ± **188**	-654 ± 174	-660 ± 313	-636 ± 322
		0.5	**-952** ± **300**	-356 ± 316	-292 ± 216	-274 ± 280
		1.0	**-25** ± **529**	688 ± 604	2501 ± 1841	2461 ± 1833
		1.5	**-248** ± **663**	1446 ± 756	938 ± 561	873 ± 536
		2.0	**116** ± **875**	2200 ± 709	1605 ± 840	2014 ± 1023

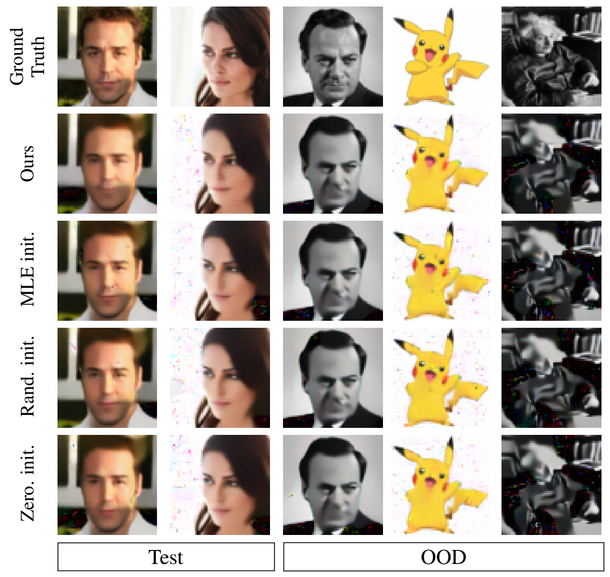

Figure 2. Reconstructions from NCS when $m = 4000$ on CelebA faces and out-of-distribution images with GLOW as the generative prior. Hyperparameter $\lambda = 0.5$.

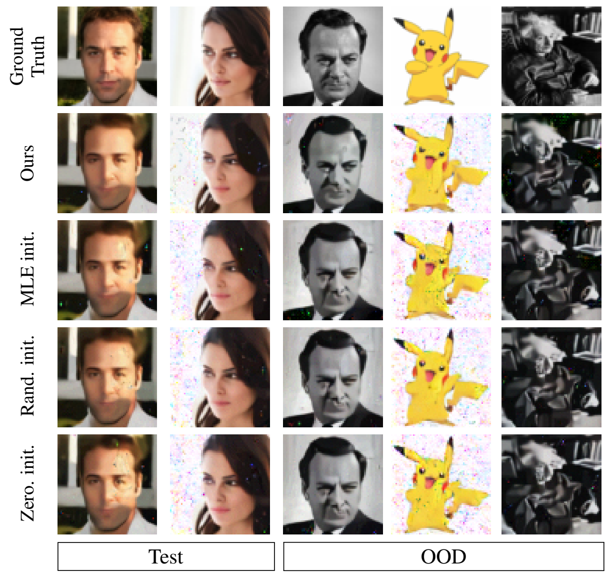

Figure 3. Reconstructions from NCS when $m = 2000$ on CelebA faces and out-of-distribution images with RealNVP as the generative prior. Hyperparameter $\lambda = 0.3$.

These quantitative and visualization results together with those from the denoising and NCS tasks demonstrate the superiority of our amortized optimization in minimizing the MAP loss.

In addition, checking the PSNR values in Tables 13 and 15 and the supplementary figures in Appendix D, AIPO in the denoising, NCS, and inpainting tasks using either RealNVP or GLOW results in outstanding reconstructions. On average, it gives higher PSNRs, recovers more details of the true images, and removes more noise.

Table 3. MAP loss (mean±se) of solving inpainting tasks with different algorithms under two generative priors, lower is better.

		λ	Ours.	MLE init.	Random init.	Zero init.
			Mask 25%			
RealNVP	Test	0.3	**-31668 ± 16**	-31567 ± 17	-31540 ± 18	-31549 ± 18
		0.5	**-29734 ± 28**	-29577 ± 31	-29545 ± 30	-29504 ± 30
		1.0	**-25940 ± 63**	-25309 ± 64	-25241 ± 67	-25304 ± 68
		1.5	**-22871 ± 104**	-21565 ± 104	-21305 ± 104	-21450 ± 103
		2.0	**-20312 ± 139**	-17784 ± 149	-18064 ± 146	-17832 ± 148
	OOD	0.3	**-31295 ± 135**	-31110 ± 212	-31123 ± 187	-31231 ± 163
		0.5	**-29251 ± 220**	-28999 ± 250	-28724 ± 315	-28900 ± 283
		1.0	**-24963 ± 464**	-24581 ± 470	-24179 ± 481	-24014 ± 513
		1.5	**-21454 ± 715**	-20201 ± 678	-19557 ± 836	-19977 ± 850
		2.0	**-18418 ± 998**	-16029 ± 1082	-15010 ± 1141	-15501 ± 1039
GLOW	Test	0.3	**-32037 ± 17**	-31964 ± 17	-31968 ± 17	-31966 ± 17
		0.5	**-30432 ± 33**	-30302 ± 32	-30317 ± 32	-30312 ± 32
		1.0	**-27518 ± 76**	-27158 ± 73	-27173 ± 74	-27164 ± 74
		1.5	**-25385 ± 122**	-24722 ± 115	-24741 ± 115	-24761 ± 112
		2.0	**-23726 ± 165**	-22670 ± 152	-22660 ± 158	-22656 ± 154
	OOD	0.3	**-31820 ± 146**	-31629 ± 149	-31654 ± 159	-31651 ± 155
		0.5	**-30096 ± 290**	-29877 ± 298	-29810 ± 304	-29802 ± 308
		1.0	**-26975 ± 755**	-26397 ± 660	-26469 ± 678	-26400 ± 639
		1.5	**-24616 ± 1226**	-23743 ± 986	-23646 ± 1063	-23718 ± 1013
		2.0	**-22628 ± 1596**	-21119 ± 1434	-21421 ± 1340	-21322 ± 1411
			Mask 30%			
RealNVP	Test	0.3	**-31668 ± 16**	-31567 ± 17	-31540 ± 18	-31549 ± 18
		0.5	**-29734 ± 28**	-29577 ± 31	-29545 ± 30	-29504 ± 30
		1.0	**-25940 ± 63**	-25309 ± 64	-25241 ± 67	-25304 ± 68
		1.5	**-22871 ± 104**	-21565 ± 104	-21305 ± 104	-21450 ± 103
		2.0	**-20312 ± 139**	-17784 ± 149	-18064 ± 146	-17832 ± 148
	OOD	0.3	-31145 ± 181	-30990 ± 188	-30974 ± 194	-30840 ± 224
		0.5	-28979 ± 246	-28849 ± 239	-28548 ± 287	-28879 ± 271
		1.0	-24944 ± 452	-24497 ± 527	-24183 ± 512	-24032 ± 594
		1.5	-25114 ± 1214	-19444 ± 895	-19550 ± 907	-19108 ± 974
		2.0	-18696 ± 980	-16027 ± 1103	-15133 ± 1307	-16206 ± 1149
GLOW	Test	0.3	**-31891 ± 16**	-31811 ± 16	-31825 ± 16	-31821 ± 17
		0.5	**-30335 ± 31**	-30198 ± 31	-30215 ± 31	-30223 ± 31
		1.0	**-27494 ± 73**	-27137 ± 71	-27185 ± 71	-27173 ± 73
		1.5	**-25444 ± 118**	-24800 ± 110	-24749 ± 109	-24772 ± 111
		2.0	**-23820 ± 160**	-22742 ± 153	-22728 ± 150	-22752 ± 149
	OOD	0.3	-31678 ± 156	-31515 ± 156	-31528 ± 149	-31516 ± 162
		0.5	**-30006 ± 302**	-29771 ± 286	-29778 ± 296	-29826 ± 291
		1.0	**-27047 ± 715**	-26279 ± 662	-26187 ± 707	-26350 ± 645
		1.5	**-24670 ± 1088**	-23787 ± 1035	-23714 ± 1151	-23528 ± 1045
		2.0	**-23068 ± 1527**	-21149 ± 1369	-20982 ± 1301	-21383 ± 1421

5. Conclusions

We propose AIPO, a new amortized optimization algorithm for solving inverse problems with NFs as the generative prior, provide a theoretical guarantee, and achieve remarkable improvement. The success of our method stems from two parts: 1) finding a good initialization and 2) easier and consecutive optimizations that build up the original problem. The MLE initialization is indispensable for starting the sequence of subroutines. The amortization is the key contribution in that even with the same MLE initialization, gradient descent without amortization yields a much worse performance than our method. We conclude that the AIPO is an appealing solution to inverse problem optimization, considering its easy implementation, theoretical guarantee, and tremendous performance gains in various tasks. Our

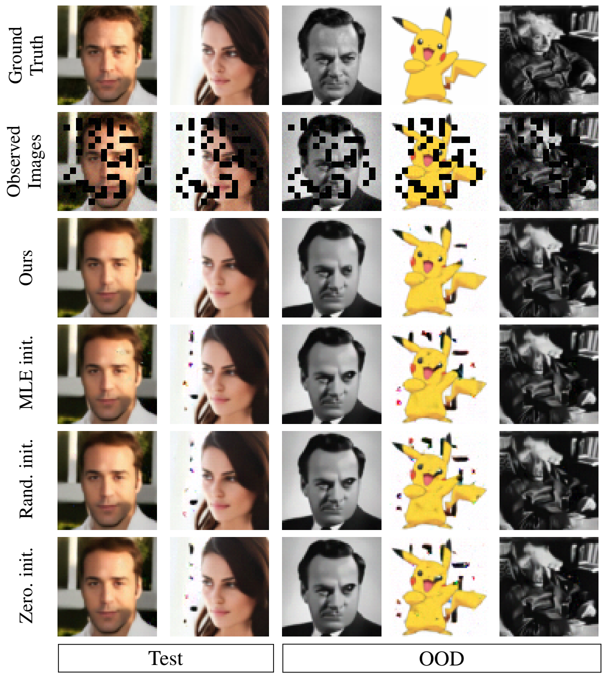

Figure 4. Results of inpainting 25% masked CelebA faces and out-of-distribution images with GLOW as the generative prior. Hyperparameter $\lambda = 1.5$.

work implies that it is beneficial to design optimization algorithms that exploit the specific structure of the objective function before seeking gradient descent as a panacea.

Acknowledgement

QL is partially supported by NYU Catalyst Prize. TL is supported in part by ICML travel grant. QL would like to thank Dinghuai Zhang and Jay Whang for useful discussions. Special thanks to the anonymous reviewers and area chair for their dedicated time and thoughtful comments.

References

Allen-Zhu, Z. Natasha 2: Faster non-convex optimization than sgd. *Advances in neural information processing systems*, 31, 2018.

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

Allgower, E. L. and Georg, K. *Introduction to numerical continuation methods*. SIAM, 2003.

Amos, B. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*, 2022.

Asim, M., Daniels, M., Leong, O., Ahmed, A., and Hand, P. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In *International Conference on Machine Learning*, pp. 399–409. PMLR, 2020a.

Asim, M., Shamshad, F., and Ahmed, A. Blind Image Deconvolution using Deep Generative Priors. *IEEE Transactions on Computational Imaging*, 6:1493–1506, 2020b.

Beck, A. *First-order methods in optimization*. SIAM, 2017.

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. Compressed Sensing Using Generative Models. In *International Conference on Machine Learning*, pp. 537–546. PMLR, 2017.

Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Candes, E. J., Romberg, J. K., and Tao, T. Stable Signal Recovery from Incomplete and Inaccurate Measurements. *Communications on pure and applied mathematics*, 59 (8):1207–1223, 2006.

Chen, T., Chen, X., Chen, W., Heaton, H., Liu, J., Wang, Z., and Yin, W. Learning to optimize: A primer and a benchmark. *arXiv preprint arXiv:2103.12828*, 2021.

Danielyan, A., Katkovnik, V., and Egiazarian, K. BM3D Frames and Variational Image Deblurring. *IEEE Transactions on image processing*, 21(4):1715–1728, 2011.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density Estimation Using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Donoho, D. L. Compressed Sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

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

Fan, Y., Tian, F., Qin, T., Li, X.-Y., and Liu, T.-Y. Learning to teach. *arXiv preprint arXiv:1805.03643*, 2018.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Hand, P. and Voroninski, V. Global Guarantees for Enforcing Deep Generative Priors by Empirical Risk. In *Conference On Learning Theory*, pp. 970–978. PMLR, 2018.

Hand, P., Leong, O., and Voroninski, V. Phase Retrieval Under a Generative Prior. *Advances in Neural Information Processing Systems*, 31, 2018.

Hazan, E., Levy, K. Y., and Shalev-Shwartz, S. On graduated optimization for stochastic non-convex problems. In *International conference on machine learning*, pp. 1833–1841. PMLR, 2016.

Hong, S., Park, I., and Chun, S. Y. On the robustness of normalizing flows for inverse problems in imaging. *arXiv preprint arXiv:2212.04319*, 2022.

Ichnowski, J., Jain, P., Stellato, B., Banjac, G., Luo, M., Borrelli, F., Gonzalez, J. E., Stoica, I., and Goldberg, K. Accelerating quadratic optimization with reinforcement learning. *Advances in Neural Information Processing Systems*, 34:21043–21055, 2021.

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

Jalal, A., Liu, L., Dimakis, A. G., and Caramanis, C. Robust Compressed Sensing using Generative Models. *Advances in Neural Information Processing Systems*, 33:713–727, 2020.

Jalal, A., Arvinte, M., Daras, G., Price, E., Dimakis, A. G., and Tamir, J. Robust compressed sensing mri with deep generative priors. *Advances in Neural Information Processing Systems*, 34:14938–14954, 2021.

Karimi, H., Nutini, J., and Schmidt, M. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 795–811. Springer, 2016.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *arXiv preprint arXiv:1710.10196*, 2017.

Karras, T., Laine, S., and Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International conference on machine learning*, pp. 2611–2620. PMLR, 2018.

Kim, Y., Wiseman, S., Miller, A., Sontag, D., and Rush, A. Semi-Amortized Variational Autoencoders. In *International Conference on Machine Learning*, pp. 2678–2687. PMLR, 2018.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative Flow with Invertible 1x1 Convolutions. *Advances in neural information processing systems*, 31, 2018.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Li, D. and Denli, H. Traversing within the Gaussian Typical Set: Differentiable Gaussianization Layers for Inverse Problems Augmented by Normalizing Flows. *arXiv preprint arXiv:2112.03860*, 2021.

Li, Y. and Yuan, Y. Convergence analysis of two-layer neural networks with relu activation. *Advances in neural information processing systems*, 30, 2017.

Liu, C., Zhu, L., and Belkin, M. Toward a theory of optimization for over-parameterized systems of non-linear equations: the lessons of deep learning. *arXiv preprint arXiv:2003.00307*, 2020.

Liu, C., Zhu, L., and Belkin, M. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022a.

Liu, X., Lu, Y., Abbasi, A., Li, M., Mohammadi, J., and Kolouri, S. Teaching networks to solve optimization problems. *arXiv preprint arXiv:2202.04104*, 2022b.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Marino, J., Yue, Y., and Mandt, S. Iterative Amortized Inference. In *International Conference on Machine Learning*, pp. 3403–3412. PMLR, 2018.

Marino, J. L. *Learned Feedback & Feedforward Perception & Control*. PhD thesis, California Institute of Technology, 2021.

Menon, S., Damian, A., Hu, S., Ravi, N., and Rudin, C. PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2437–2445, 2020.

Ongie, G., Jalal, A., Metzler, C. A., Baraniuk, R. G., Dimakis, A. G., and Willett, R. Deep Learning Techniques for Inverse Problems in Imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. URL `http://jmlr.org/papers/v22/19-1028.html`.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Rose, K., Gurewitz, E., and Fox, G. A deterministic annealing approach to clustering. *Pattern Recognition Letters*, 11(9):589–594, 1990.

Safran, I. M., Yehudai, G., and Shamir, O. The effects of mild over-parameterization on the optimization landscape of shallow relu neural networks. In *Conference on Learning Theory*, pp. 3889–3934. PMLR, 2021.

Scaman, K., Malherbe, C., and Dos Santos, L. Convergence rates of non-convex stochastic gradient descent under a generic lojasiewicz condition and local smoothness. In *International Conference on Machine Learning*, pp. 19310–19327. PMLR, 2022.

Song, C., Ramezani-Kebrya, A., Pethick, T., Eftekhari, A., and Cevher, V. Subquadratic overparameterization for shallow neural networks. *Advances in Neural Information Processing Systems*, 34:11247–11259, 2021.

Song, G., Fan, Z., and Lafferty, J. Surfing: Iterative optimization over incrementally trained deep networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212, 2009.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Deep Image Prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9446–9454, 2018.

Van Veen, D., Jalal, A., Soltanolkotabi, M., Price, E., Vishwanath, S., and Dimakis, A. G. Compressed Sensing with Deep Image Prior and Learned Regularization. *arXiv preprint arXiv:1806.06438*, 2018.

Vershynin, R. *Estimation in high dimensions: a geometric perspective*. Springer, 2015.

Vitoria, P., Sintes, J., and Ballester, C. Semantic Image Inpainting Through Improved Wasserstein Generative Adversarial Networks. *arXiv preprint arXiv:1812.01071*, 2018.

Whang, J., Lei, Q., and Dimakis, A. G. Compressed sensing with invertible generative models and dependent noise. *arXiv preprint arXiv:2003.08089*, 2020.

Whang, J., Lei, Q., and Dimakis, A. Solving Inverse Problems with a Flow-based Noise Model. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11146–11157. PMLR, 18–24 Jul 2021a. URL `https://proceedings.mlr.press/v139/whang21a.html`.

Whang, J., Lindgren, E., and Dimakis, A. Composing Normalizing Flows for Inverse Problems. In *International Conference on Machine Learning*, pp. 11158–11169. PMLR, 2021b.

Wu, Z. The effective energy transformation scheme as a special continuation approach to global optimization with application to molecular conformation. *SIAM Journal on Optimization*, 6(3):748–768, 1996.

Yang, H., Antonante, P., Tzoumas, V., and Carlone, L. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robotics and Automation Letters*, 5(2):1127–1134, 2020.

Yu, G., Sapiro, G., and Mallat, S. Solving Inverse Problems with Piecewise Linear Estimators: From Gaussian Mixture Models to Structured Sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2011.

Zhou, Y., Yang, J., Zhang, H., Liang, Y., and Tarokh, V. Sgd converges to global minimum in deep learning via star-convex path. *arXiv preprint arXiv:1901.00451*, 2019.

A. Projected Gradient Descent for MLE

We elaborate more details about solving the MLE $\boldsymbol{x}^*(0)$ as defined in (4). Note that in practice a valid image should always lie within the $[0,1]^n$, and we include this consideration here.

Let Proj denote a projection operator and $\mathrm{clip}(x,a,b) = (x \vee a) \wedge b$ denote a standard clipping operation.

For denoising task when $f(\boldsymbol{x}) = \boldsymbol{x}$, the affine constraint contains only one single point $\boldsymbol{x} = \boldsymbol{y}$, and the projection can be done by applying the following projection

$$\mathrm{Proj}_{[0,1]^n}(\boldsymbol{x}) = (\mathrm{clip}(x_1,0,1),\dots,\mathrm{clip}(x_n,0,1))^\top.$$

For NCS tasks, $f(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x}$ where $\mathbf{A}$ has full row rank. We apply the following iterative projection (Beck, 2017) along two constraints by having $\boldsymbol{x}_0 = \boldsymbol{x}$ and repeating the following steps until converges

$$\begin{aligned}\boldsymbol{x}_i &= \boldsymbol{x}_{i-1} - \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}(\mathbf{A}\boldsymbol{x}_{i-1} - \boldsymbol{y})\\ \boldsymbol{x}_i &= \mathrm{Proj}_{[0,1]^n}(\boldsymbol{x}_i).\end{aligned}$$

For inpainting tasks, $f(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x}$ where $\mathbf{A}$ is a diagonal matrix with 0/1 entries at its diagonal. Here 1 indicates that the pixel is observed and 0 indicates the opposite. Again we apply the following iterative projection (Beck, 2017) along two constraints by having $\boldsymbol{x}_0 = \boldsymbol{x}$ and repeating the following steps until converges

$$\begin{aligned}\boldsymbol{x}_i &= \mathbf{A}\boldsymbol{y} + (\mathbf{I} - \mathbf{A})\boldsymbol{x}_{i-1}\\ \boldsymbol{x}_i &= \mathrm{Proj}_{[0,1]^n}(\boldsymbol{x}_i).\end{aligned}$$

B. Missing Proofs

B.1. Proof of Remark 3.5

Proof of Remark 3.5. (i) If $\exists\mu' > 0$, $\exists\delta > 0$, s.t.$\forall\lambda \in [\frac{\delta_0}{2C+\delta_0/\Lambda}, \Lambda]$, $\mathcal{L}_{MAP}(\cdot,\lambda)$ is μ'-strongly convex on $B(\boldsymbol{x}^*(\lambda),\delta)$, then $\forall\boldsymbol{x} \in B(\boldsymbol{x}^*(\lambda),\delta)$, we have

$$\begin{aligned}&\mathcal{L}_{MAP}(\boldsymbol{x},\lambda)\\ \geq&\mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda),\lambda)\\ \geq&\mathcal{L}_{MAP}(\boldsymbol{x},\lambda) + \nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x},\lambda)^\intercal(\boldsymbol{x}^*(\lambda) - \boldsymbol{x}) + \frac{\mu'}{2}\left\|\boldsymbol{x}^*(\lambda) - \boldsymbol{x}\right\|^2\\ \geq&\mathcal{L}_{MAP}(\boldsymbol{x},\lambda) - \left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x},\lambda)\right\|\left\|\boldsymbol{x}^*(\lambda) - \boldsymbol{x}\right\| + \frac{\mu'}{2}\left\|\boldsymbol{x}^*(\lambda) - \boldsymbol{x}\right\|^2.\end{aligned}$$

Therefore, we have

$$-\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x},\lambda)\right\|\left\|\boldsymbol{x}^*(\lambda) - \boldsymbol{x}\right\| + \frac{\mu'}{2}\left\|\boldsymbol{x}^*(\lambda) - \boldsymbol{x}\right\|^2 \leq 0,$$

from which we can see that

$$\left\|\boldsymbol{x}^*(\lambda) - \boldsymbol{x}\right\| \leq \frac{2}{\mu'}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x},\lambda)\right\|.$$

(ii) $\forall\boldsymbol{x} \in B(\boldsymbol{x}^*(\lambda),\delta)$, $\exists t \in [0,1]$, $\boldsymbol{\xi} = t\boldsymbol{x}^*(\lambda) + (1-t)\boldsymbol{x}$, s.t.

$$\begin{aligned}&\mathcal{L}_{MAP}(\boldsymbol{x},\lambda) - \mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda),\lambda)\\ =&\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{\xi},\lambda)^\intercal(\boldsymbol{x} - \boldsymbol{x}^*(\lambda))\\ \leq&\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{\xi},\lambda)\right\|\left\|\boldsymbol{x} - \boldsymbol{x}^*(\lambda)\right\|\\ =&\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{\xi},\lambda) - \nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda),\lambda)\right\|\left\|\boldsymbol{x} - \boldsymbol{x}^*(\lambda)\right\|\\ \leq&L\left\|\boldsymbol{\xi} - \boldsymbol{x}^*(\lambda)\right\|\left\|\boldsymbol{x} - \boldsymbol{x}^*(\lambda)\right\|\\ \leq&L\left\|\boldsymbol{x} - \boldsymbol{x}^*(\lambda)\right\|^2\\ \leq&L\sigma^2\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x},\lambda)\right\|^2.\end{aligned}$$

□

B.2. Proof of Theorem 3.6

The proof of Theorem 3.6 relies on the following three lemmas:

Lemma B.1. *If Assumption 3.3 holds, then* $\lim_{\lambda\to 0^+}\boldsymbol{x}^*(\lambda)$ *exists and* $\boldsymbol{x}^*(0)=\lim_{\lambda\to 0^+}\boldsymbol{x}^*(\lambda)$*, where* $\boldsymbol{x}^*(0)$ *is defined in* (4).

Proof of Lemma B.1. We show the existence of $\lim_{\lambda\to 0^+}\boldsymbol{x}^*(\lambda)$ by first proving that $\boldsymbol{x}^*(\lambda)$ is bounded on $(0,\Lambda]$. Otherwise, for any given $\lambda_1\in(0,\Lambda]$, there exists $\lambda_2\in(0,\lambda_1)$, s.t. $\|\boldsymbol{x}^*(\lambda_2)\|>\|\boldsymbol{x}^*(\lambda_1)\|+C\lambda_1$. Then we have

$$\|\boldsymbol{x}^*(\lambda_1)-\boldsymbol{x}^*(\lambda_2)\|\geq\|\boldsymbol{x}^*(\lambda_2)\|-\|\boldsymbol{x}^*(\lambda_1)\|>C\lambda_1\geq C|\lambda_1-\lambda_2|\,.$$

This contradicts to the Lipschitz continuity of $\boldsymbol{x}^*(\lambda)$ on $(0,\Lambda]$ in Assumption 3.3. Therefore $\boldsymbol{x}^*(\lambda)$ is bounded on $(0,\Lambda]$. By Bolzano-Weierstrass Theorem, for any sequence $\{\lambda_n\}$ on $(0,\Lambda]$ such that $\lim_{n\to+\infty}\lambda_n=0$, $\{\boldsymbol{x}^*(\lambda_n)\}$ has a convergent subsequence $\{\boldsymbol{x}^*(\lambda_{n_k})\}$. Denote $\lim_{k\to+\infty}\boldsymbol{x}^*(\lambda_{n_k})=\bar{\boldsymbol{x}}$, then $\forall\varepsilon>0$, $\exists k\in\mathbb{N}_+$, s.t. $\lambda_{n_k}\in(0,\frac{\epsilon}{2C}]$ and $\|\boldsymbol{x}^*(\lambda_{n_k})-\bar{\boldsymbol{x}}\|<\frac{\varepsilon}{2}$ hold. By the Lipschitz continuity of $\boldsymbol{x}^*(\lambda)$, we have $\forall\lambda\in(0,\lambda_{n_k}]$

$$\|\boldsymbol{x}^*(\lambda)-\bar{\boldsymbol{x}}\|\leq\|\boldsymbol{x}^*(\lambda)-\boldsymbol{x}^*(\lambda_{n_k})\|+\|\boldsymbol{x}^*(\lambda_{n_k})-\bar{\boldsymbol{x}}\|\leq C|\lambda-\lambda_{n_k}|+\frac{\varepsilon}{2}\leq C\lambda_{n_k}+\frac{\varepsilon}{2}\leq\varepsilon\,.$$

This indicates that $\lim_{\lambda\to 0^+}\boldsymbol{x}^*(\lambda)=\bar{\boldsymbol{x}}$.

If $f(\bar{\boldsymbol{x}})\neq\boldsymbol{y}$, since $-\log p_e(\boldsymbol{y}-f(\boldsymbol{x}))$ reaches its minimum value if and only if $\boldsymbol{y}=f(\boldsymbol{x})$, we have

$$\triangle_1:=\log p_e(\boldsymbol{y}-f(\boldsymbol{x}^*(0)))-\log p_e(\boldsymbol{y}-f(\bar{\boldsymbol{x}}))>0.$$

By the continuity of $\log p_e(\boldsymbol{y}-f(\boldsymbol{x}))$ and $\log p_G(\boldsymbol{x})$ in $\boldsymbol{x}$, there exists small enough $\lambda>0$, s.t.

$$\log p_e(\boldsymbol{y}-f(\boldsymbol{x}^*(\lambda)))<\log p_e(\boldsymbol{y}-f(\bar{\boldsymbol{x}}))+\frac{\triangle_1}{2}\,,$$

and

$$\lambda(\log p_G(\boldsymbol{x}^*(\lambda))-\log p_G(\boldsymbol{x}^*(0)))<\frac{\triangle_1}{2}\,.$$

Combining the above inequalities, we have

$$\begin{aligned}&\log p_e(\boldsymbol{y}-f(\boldsymbol{x}^*(\lambda)))+\lambda\log p_G(\boldsymbol{x}^*(\lambda))\\ <&\log p_e(\boldsymbol{y}-f(\bar{\boldsymbol{x}}))+\lambda\log p_G(\boldsymbol{x}^*(0))+\triangle_1\\ =&\log p_e(\boldsymbol{y}-f(\boldsymbol{x}^*(0)))+\lambda\log p_G(\boldsymbol{x}^*(0))\,.\end{aligned}$$

This contradicts to the definition of $\boldsymbol{x}^*(\lambda)$ (see Eq. (3)). This contradiction indicates $f(\bar{\boldsymbol{x}})$ must be equal to $\boldsymbol{y}$, which implies that $\bar{\boldsymbol{x}}$ is a feasible solution of problem (4). So if $\bar{\boldsymbol{x}}$ is not an optimal solution of problem (4), then we have

$$\log p_G(\boldsymbol{x}^*(0))>\log p_G(\bar{\boldsymbol{x}})\,.$$

Thus by the continuity of $\log p_G$, there exists small enough $\lambda>0$, s.t.

$$\log p_G(\boldsymbol{x}^*(\lambda))<\log p_G(\boldsymbol{x}^*(0))\,. \tag{5}$$

Since $f(\boldsymbol{x}^*(0)))=\boldsymbol{y}$, we have

$$\log p_e(\boldsymbol{y}-f(\boldsymbol{x}^*(0)))\geq\log p_e(\boldsymbol{y}-f(\boldsymbol{x}^*(\lambda)))\,. \tag{6}$$

Combining (5) and (6), we obtain

$$\log p_e(\boldsymbol{y}-f(\boldsymbol{x}^*(\lambda)))+\lambda\log p_G(\boldsymbol{x}^*(\lambda))<\log p_e(\boldsymbol{y}-f(\boldsymbol{x}^*(0)))+\lambda\log p_G(\boldsymbol{x}^*(0))\,,$$

which again contradicts to the definition of $\boldsymbol{x}^*(\lambda)$.

Thus $\bar{\boldsymbol{x}}=\boldsymbol{x}^*(0)$. □

Lemma B.1 indicates that when Assumption 3.3 holds, $\boldsymbol{x}^*(\lambda)$ is Lipschitz continuous on $[0,\Lambda]$, i.e., for all $\lambda_1,\lambda_2 \in [0,\Lambda]$, $\|\boldsymbol{x}^*(\lambda_1)-\boldsymbol{x}^*(\lambda_2)\| \le C|\lambda_1-\lambda_2|$.

Lemma B.2. *If Assumption 3.2 holds, then* $\forall \lambda \in [0,\Lambda]$, *let*

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{1}{L}\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda). \tag{7}$$

We have

$$\mathcal{L}_{MAP}(\boldsymbol{x}_{k+1},\lambda) \le \mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda) - \frac{1}{2L}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda)\right\|^2 . \tag{8}$$

Proof of Lemma B.2.

$$\begin{aligned}
&\mathcal{L}_{MAP}(\boldsymbol{x}_{k+1},\lambda)\\
\le&\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda) + \nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda)^\intercal(\boldsymbol{x}_{k+1}-\boldsymbol{x}_k) + \frac{L}{2}\left\|\boldsymbol{x}_{k+1}-\boldsymbol{x}_k\right\|^2\\
=&\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda) - \frac{1}{L}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda)\right\|^2 + \frac{1}{2L}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda)\right\|^2\\
=&\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda) - \frac{1}{2L}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda)\right\|^2 .
\end{aligned}$$

□

Lemma B.3. *Assume Assumption 3.2 and 3.4 hold. Let* $\mu = \frac{1}{2L\sigma^2}$. *For any fixed* $\delta > 0$, *let*

$$\delta_0 = \min\left\{\delta, \frac{\mu}{\sqrt{2(L-\mu)L}}\delta\right\}.$$

$\forall \lambda \in [\frac{\delta_0}{2C+\delta_0/\Lambda},\Lambda]$, *if* $\boldsymbol{x}_0 \in B(\boldsymbol{x}^*(\lambda),\delta_0)$, *and* $\{\boldsymbol{x}_k\}$ *is generated by* (7), *then* $\forall k \in \mathbb{N}_+$, $\boldsymbol{x}_k \in B(\boldsymbol{x}^*(\lambda),\delta)$.

Proof of Lemma B.3. We prove by induction on k. when $k=0$, we have

$$\boldsymbol{x}_0 \in B(\boldsymbol{x}^*(\lambda),\delta_0) \subset B(\boldsymbol{x}^*(\lambda),\delta).$$

Assume when $k \ge 1$, $\boldsymbol{x}_j \in B(\boldsymbol{x}^*(\lambda),\delta)$, $\forall j \le k$, then by (8) we have

$$\begin{aligned}
&\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda) - \mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda),\lambda)\\
\le&\mathcal{L}_{MAP}(\boldsymbol{x}_{k-1},\lambda) - \mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda),\lambda) - \frac{1}{2L}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x}_{k-1},\lambda)\right\|^2 .
\end{aligned}$$

By induction hypothesis and (3.5), we have

$$\begin{aligned}
&\mathcal{L}_{MAP}(\boldsymbol{x}_{k-1},\lambda) - \mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda),\lambda) - \frac{1}{2L}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_{MAP}(\boldsymbol{x}_{k-1},\lambda)\right\|^2\\
\le&(1-\frac{\mu}{L})(\mathcal{L}_{MAP}(\boldsymbol{x}_{k-1},\lambda) - \mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda),\lambda)).
\end{aligned}$$

From the above two inequalities we deduce

$$\begin{aligned}
&\mathcal{L}_{MAP}(\boldsymbol{x}_k,\lambda) - \mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda),\lambda)\\
\le&(1-\frac{\mu}{L})(\mathcal{L}_{MAP}(\boldsymbol{x}_{k-1},\lambda) - \mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda),\lambda))\\
\le&\cdots\\
\le&(1-\frac{\mu}{L})^k(\mathcal{L}_{MAP}(\boldsymbol{x}_0,\lambda)) - \mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda),\lambda)).
\end{aligned} \tag{9}$$

From (8) we can see that

$$\mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda), \lambda) \leq \mathcal{L}_{MAP}(\boldsymbol{x}_k, \lambda) - \frac{1}{2L} \left\| \nabla_{\boldsymbol{x}} \mathcal{L}_{MAP}(\boldsymbol{x}_k, \lambda) \right\|^2 . \tag{10}$$

Thus we have

$$\begin{aligned}
&\mu \left\| \boldsymbol{x}_k - \boldsymbol{x}^*(\lambda) \right\|^2 \\
\overset{(a)}{\leq}& \frac{1}{2L} \left\| \nabla_{\boldsymbol{x}} \mathcal{L}_{MAP}(\boldsymbol{x}_k, \lambda) \right\|^2 \\
\overset{(b)}{\leq}& \mathcal{L}_{MAP}(\boldsymbol{x}_k, \lambda) - \mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda), \lambda) \\
\overset{(c)}{\leq}& (1 - \frac{\mu}{L})^k (\mathcal{L}_{MAP}(\boldsymbol{x}_0, \lambda) - \mathcal{L}_{MAP}(\boldsymbol{x}^*(\lambda), \lambda)) \\
\overset{(d)}{\leq}& (1 - \frac{\mu}{L})^k \frac{1}{2\mu} \left\| \nabla_{\boldsymbol{x}} \mathcal{L}_{MAP}(\boldsymbol{x}_0, \lambda) \right\|^2 \\
\overset{(e)}{\leq}& (1 - \frac{\mu}{L})^k \frac{L^2}{2\mu} \left\| \boldsymbol{x}_0 - \boldsymbol{x}^*(\lambda) \right\|^2 ,
\end{aligned}$$

where step (a) is by Assumption 3.4 (recall that $\mu = \frac{1}{2L\sigma^2}$), step (b), step (c), and step (d) are based on equation (10), equation (9), and equation (3.5) respectively. Step (e) is by Assumption 3.2.

From the above inequalities, we deduce

$$\left\| \boldsymbol{x}_k - \boldsymbol{x}^*(\lambda) \right\| \leq \frac{L}{\sqrt{2}\mu} (1 - \frac{\mu}{L})^{k/2} \left\| \boldsymbol{x}_0 - \boldsymbol{x}^*(\lambda) \right\| \leq \frac{\delta}{2}. \tag{11}$$

Note that

$$\begin{aligned}
\left\| \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \right\| &= \frac{1}{L} \left\| \nabla_{\boldsymbol{x}} \mathcal{L}_{MAP}(\boldsymbol{x}_k, \lambda) \right\| \\
&= \frac{1}{L} \left\| \nabla_{\boldsymbol{x}} \mathcal{L}_{MAP}(\boldsymbol{x}_k, \lambda) - \nabla_{\boldsymbol{x}} \mathcal{L}_{MAP}(\boldsymbol{x}^*, \lambda) \right\| \\
&\leq \left\| \boldsymbol{x}_k - \boldsymbol{x}^*(\lambda) \right\| \leq \frac{\delta}{2}.
\end{aligned}$$

So we have

$$\left\| \boldsymbol{x}_{k+1} - \boldsymbol{x}^*(\lambda) \right\| \leq \left\| \boldsymbol{x}_k - \boldsymbol{x}^*(\lambda) \right\| + \left\| \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \right\| \leq \delta.$$

The induction step is complete. □

Now we are ready to prove the Theorem 3.6.

Proof of Theorem 3.6. From the first inequality of (11) we know that if $\boldsymbol{x}_0 \in B(\boldsymbol{x}^*(\lambda), \delta_0)$, then

$$\left\| \boldsymbol{x}_k - \boldsymbol{x}^*(\lambda) \right\| \leq (1 - \frac{\mu}{L})^{k/2} \delta, \ \forall k \in \mathbb{N}. \tag{12}$$

Let $\lambda_0 = 0$, $\lambda_{i+1} = \lambda_i + \frac{\Lambda}{N}$, $i = 0, 1, \cdots, N-1$, where $N \geq \frac{2C\Lambda}{\delta_0}$. Let $\Delta\lambda \triangleq \frac{\Lambda}{N}$. Note that the lower bound $\frac{\delta_0}{2C + \delta_0/\Lambda}$ of λ in Assumption 3.4 ensures that for all $i = 1, 2, \cdots, N$, the local property of $\nabla_{\boldsymbol{x}} \mathcal{L}_{MAP}$ stated in Assumption 3.4 holds at $\boldsymbol{x}^*(\lambda_i)$.

Suppose $\boldsymbol{x}_k(\lambda_i)$ is the point in the last iteration of the inner loop when $\lambda = \lambda_i$, where $K = [\frac{2\log(2\delta/\delta_0)}{\log(L/(L-\mu))}] + 1$.

Let $\boldsymbol{x}_0(\lambda_0) = G^{-1}(\boldsymbol{y})$, then we have

$$\left\| \boldsymbol{x}_0(\lambda_0) - \boldsymbol{x}^*(\lambda_0) \right\| = 0. \tag{13}$$

If $\|\boldsymbol{x}_0(\lambda_i) - \boldsymbol{x}^*(\lambda_i)\| \le \delta_0$, then by (12) we deduce

$$\|\boldsymbol{x}_K(\lambda_i) - \boldsymbol{x}^*(\lambda_i)\| \le \delta.$$

When $\lambda = \lambda_{i+1}$, let $\boldsymbol{x}_0(\lambda_{i+1}) = \boldsymbol{x}_K(\lambda_i)$, then we have

$$\begin{aligned} & \|\boldsymbol{x}_0(\lambda_{i+1}) - \boldsymbol{x}^*(\lambda_{i+1})\| \\ \le & \|\boldsymbol{x}_0(\lambda_{i+1}) - \boldsymbol{x}^*(\lambda_i)\| + \|\boldsymbol{x}^*(\lambda_i) - \boldsymbol{x}^*(\lambda_{i+1})\| \\ \overset{(a)}{\le} & \|\boldsymbol{x}_K(\lambda_i) - \boldsymbol{x}^*(\lambda_i)\| + C\Delta\lambda \\ \le & \delta_0, \end{aligned}$$

where (a) uses Assumption 3.3 and $\boldsymbol{x}_0(\lambda_{i+1}) = \boldsymbol{x}_K(\lambda_i)$.

Thus by (13) and induction we deduce

$$\|\boldsymbol{x}_0(\lambda_i) - \boldsymbol{x}^*(\lambda_i)\| \le \delta_0, \ \forall i \in \{1, \cdots, N\}.$$

When $i = N$, $K = \max\{0, [\frac{2\log(2\varepsilon/\delta_0)}{\log(L/(L-\mu))}] + 1\}$, again using the first inequality in (11), we have

$$\|\boldsymbol{x}_K(\Lambda) - \boldsymbol{x}^*(\Lambda)\| \le \varepsilon.$$

$\square$

C. Alternative Amortized Optimization for Inverse Problems

In this section we present an alternative version of Amortized Optimization for $\boldsymbol{x}$, which is used in experiments. We adopt the following hyper-parameters as summarized in Table 4.

Remark C.1. In practice it is difficult to quantify the *gradual change*, so in step 8, we determine the step size δ such that the relative change of the loss is not too large, and for practical concern, we also want the step size not too small. In step 9 and 10, we update our belief of whether the minimal step size is too aggressive. We check how much the MAP loss decreases in this round compared with the previous one. Intuitively, if current change is larger, then λ has been updated too much such that the previous solution, $\boldsymbol{x}_0$ in this round, is far from $\boldsymbol{x}^*(\lambda)$, then we decrease the $\delta_{\min}$ by raising it to power $\Delta'_{\mathcal{L}}/\Delta_{\mathcal{L}}$, the prespecified bound will guarantee the result decrease, and vice versa. Finally, we will end up with a minimal step size $\delta_{\min}$ that allows us to achieve a stable loss change rate. We found this algorithm works empirically well.

Table 4. Hyper-parameters used in Amortized Optimization appeared Algorithm 2.

Hyperparameter	step size α	iteration K	target rate r	min. step size $\delta^0_{\min}$	$\delta_{\min,h}$	$\delta_{\min,h}$
Value	0.05	40	0.05	$\Lambda/20$	$4\delta^0_{\min} \vee 1$	$\delta^0_{\min}/4$

Algorithm 2 Alternative AIPO algorithm

```
1: Input: observation $\boldsymbol{y}$, generative model $G$, targeted hyperparameter $\Lambda > 0$, target change rate of loss magnitude
   $r \in (0,1)$, minimum step size $\delta^0_{\min}$ and its bound $1 \geq \delta_{\min,h} \geq \delta_{\min,l} > 0$ for updating $\lambda$, maximum iteration number
   $K > 0$, step size $\alpha > 0$ for solving $\mathcal{L}_{\text{MAP}}(\boldsymbol{x}, \lambda)$.
2: Initialize: $\lambda = 0, \boldsymbol{x}_0 = f^{-1}(\boldsymbol{y}), \delta_{\min} = \delta^0_{\min}, \Delta_{\mathcal{L}} = \texttt{NULL}$
3: repeat
4:     for $k = 1, \dots, K$ do
5:         $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} - \alpha \nabla_{\boldsymbol{x}} \mathcal{L}_{\text{MAP}}(\boldsymbol{x}_{k-1}, \lambda)$
6:     end for
7:     $\boldsymbol{e} = \boldsymbol{y} - f(\boldsymbol{x}_K)$
8:     Determine the step size $\delta$ such that
           $$r = \left| \frac{\mathcal{L}_{\text{MAP}}(\boldsymbol{x}_K; \lambda + \delta) - \mathcal{L}_{\text{MAP}}(\boldsymbol{x}_K; \lambda)}{\mathcal{L}_{\text{MAP}}(\boldsymbol{x}_K; \lambda)} \right| \Rightarrow \delta = r \left| \frac{\log p_e(\boldsymbol{e})}{\log p_G(\boldsymbol{x}_K)} + \lambda \right|$$
       while satisfying $\delta \geq \delta_{\min}$, i.e.,
           $$\delta = \delta \vee \delta_{\min}$$
9:     if $\Delta_{\mathcal{L}} \neq \texttt{NULL}$ then
10:        Update $\delta_{\min}$ based on the relative loss change
               $$\Delta'_{\mathcal{L}} = \left| \frac{\mathcal{L}_{\text{MAP}}(\boldsymbol{x}_0, \lambda) - \mathcal{L}_{\text{MAP}}(\boldsymbol{x}_K, \lambda)}{\mathcal{L}_{\text{MAP}}(\boldsymbol{x}_0, \lambda)} \right|$$
           within the specified range
               $$\delta_{\min} = \text{clip}(\Delta'_{\mathcal{L}} / \Delta_{\mathcal{L}}, \delta_{\min,l}, \delta_{\min,h})$$
11:    end if
12:    Update $\Delta_{\mathcal{L}} = \Delta'_{\mathcal{L}}$, $K = 1.1K$, $\alpha = 0.99\alpha$.
13:    $\lambda = \lambda + \delta$
14:    $\boldsymbol{x}_0 = \boldsymbol{x}_K$
15: until $\lambda \geq \Lambda$
output $\boldsymbol{x}_K$
```

D. Additional Experiment Results

In this section we report more experiment results, including PSNR and SSIM values on all tasks, and more reconstructed images using the hyperparameter λ based on which most algorithms can achieve the highest PSNR.

Table 5. PSNR (mean±se) of different algorithms using RealNVP as the generative prior for denoising, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
			Denoising, $\sigma = 0.05$			
Test	Ours	**32.67 ± 0.05**	**33.83 ± 0.05**	**32.92 ± 0.05**	**31.75 ± 0.06**	**30.84 ± 0.06**
	MLE. init.	32.53 ± 0.05	33.50 ± 0.05	32.56 ± 0.06	31.44 ± 0.07	30.57 ± 0.07
	Rand. init.	32.54 ± 0.05	33.52 ± 0.06	32.47 ± 0.07	31.45 ± 0.07	30.54 ± 0.08
	Zero. init.	32.53 ± 0.05	33.49 ± 0.06	32.51 ± 0.07	31.41 ± 0.07	30.48 ± 0.07
OOD	Ours	**32.07 ± 0.35**	**33.13 ± 0.49**	**32.21 ± 0.56**	**31.04 ± 0.51**	**30.26 ± 0.49**
	MLE. init.	31.81 ± 0.44	32.73 ± 0.58	31.76 ± 0.73	30.80 ± 0.67	30.02 ± 0.62
	Rand. init.	31.74 ± 0.44	32.84 ± 0.54	31.73 ± 0.61	30.89 ± 0.61	29.72 ± 0.67
	Zero. init.	31.80 ± 0.39	32.79 ± 0.58	31.91 ± 0.63	30.73 ± 0.66	30.03 ± 0.79
			Denoising, $\sigma = 0.1$			
Test	Ours	**29.92 ± 0.08**	**30.02 ± 0.06**	**28.27 ± 0.06**	**27.04 ± 0.06**	**26.11 ± 0.06**
	MLE. init.	29.74 ± 0.08	29.75 ± 0.07	28.13 ± 0.07	27.00 ± 0.07	26.09 ± 0.07
	Rand. init.	29.62 ± 0.08	29.72 ± 0.07	28.14 ± 0.07	26.91 ± 0.08	**26.11 ± 0.07**
	Zero. init.	29.60 ± 0.08	29.71 ± 0.07	28.14 ± 0.07	26.96 ± 0.07	26.04 ± 0.07
OOD	Ours	**28.70 ± 0.64**	**29.05 ± 0.55**	27.25 ± 0.50	26.09 ± 0.54	25.36 ± 0.53
	MLE. init.	28.64 ± 0.67	28.95 ± 0.71	**27.44 ± 0.69**	26.25 ± 0.65	**25.45 ± 0.60**
	Rand. init.	28.57 ± 0.68	28.83 ± 0.67	27.33 ± 0.65	**26.29 ± 0.67**	25.44 ± 0.61
	Zero. init.	28.53 ± 0.65	28.89 ± 0.72	27.42 ± 0.68	**26.29 ± 0.63**	25.38 ± 0.57

Table 6. SSIM (mean±se) of different algorithms using RealNVP as the generative prior for denoising, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
			Denoising, $\sigma = 0.05$			
Test	Ours	**92.65 ± 0.23**	**94.05 ± 0.22**	**92.87 ± 0.24**	**91.12 ± 0.26**	**89.57 ± 0.28**
	MLE. init.	92.44 ± 0.24	93.63 ± 0.22	92.35 ± 0.25	90.72 ± 0.27	89.39 ± 0.30
	Rand. init.	92.50 ± 0.23	93.63 ± 0.22	92.23 ± 0.27	90.70 ± 0.29	89.27 ± 0.31
	Zero. init.	92.46 ± 0.23	93.59 ± 0.23	92.28 ± 0.26	90.67 ± 0.29	89.20 ± 0.31
OOD	Ours	**91.82 ± 1.70**	**93.08 ± 1.80**	**91.61 ± 1.69**	**89.30 ± 1.75**	**87.69 ± 1.88**
	MLE. init.	91.11 ± 2.07	92.35 ± 2.11	90.80 ± 2.22	89.11 ± 2.17	87.09 ± 2.40
	Rand. init.	90.92 ± 2.06	92.60 ± 1.91	90.73 ± 2.24	88.97 ± 2.28	86.65 ± 2.50
	Zero. init.	91.16 ± 1.91	92.48 ± 2.04	91.22 ± 2.08	88.81 ± 2.28	87.27 ± 2.54
			Denoising, $\sigma = 0.1$			
Test	Ours	**88.93 ± 0.36**	**88.77 ± 0.32**	84.41 ± 0.33	80.88 ± 0.37	77.77 ± 0.41
	MLE. init.	88.61 ± 0.35	88.46 ± 0.32	**84.63 ± 0.34**	**81.37 ± 0.37**	78.50 ± 0.38
	Rand. init.	88.36 ± 0.36	88.34 ± 0.33	84.60 ± 0.35	81.36 ± 0.36	**78.53 ± 0.39**
	Zero. init.	88.34 ± 0.36	88.38 ± 0.33	84.61 ± 0.34	81.36 ± 0.35	78.35 ± 0.39
OOD	Ours	85.75 ± 2.59	85.57 ± 2.26	79.05 ± 2.97	74.54 ± 3.44	71.61 ± 3.36
	MLE. init.	**85.76 ± 2.87**	**85.70 ± 2.93**	**80.72 ± 3.30**	76.11 ± 3.15	72.79 ± 3.42
	Rand. init.	85.50 ± 2.95	85.67 ± 2.57	80.37 ± 3.14	**76.24 ± 3.21**	**72.95 ± 3.16**
	Zero. init.	85.45 ± 3.05	85.29 ± 2.76	80.62 ± 3.09	**76.24 ± 3.11**	72.17 ± 3.13

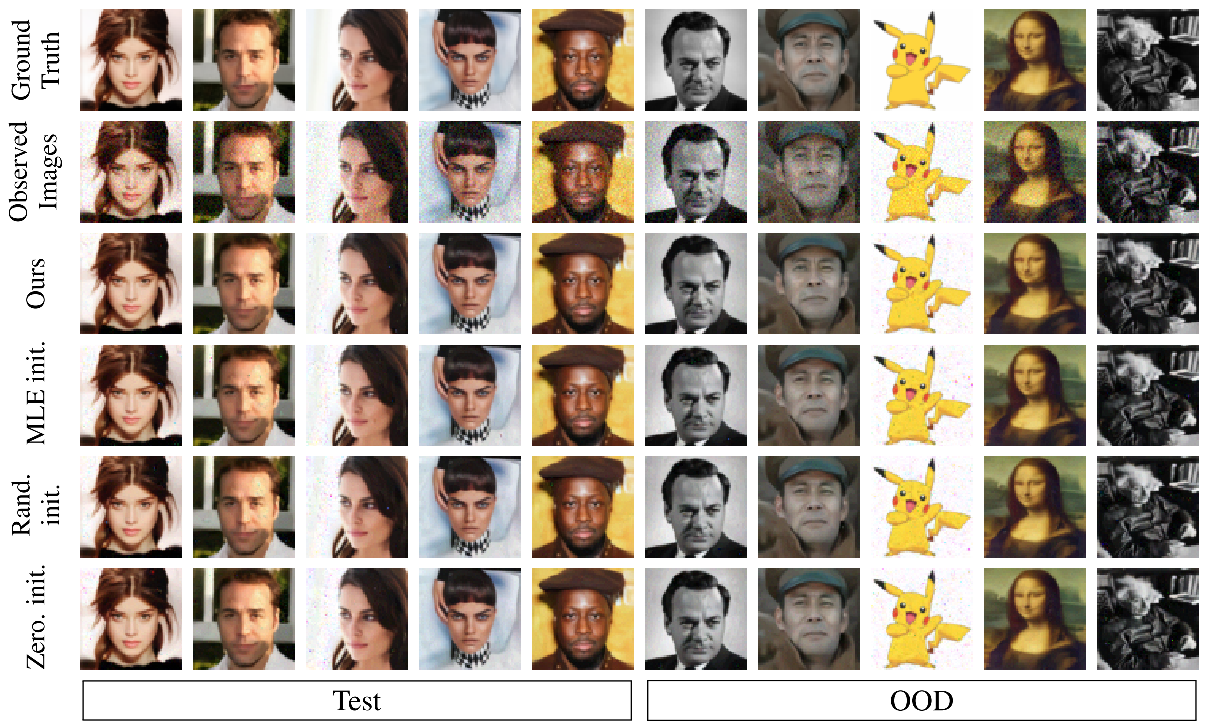

Figure 5. Results of solving denoising task ($\sigma = 0.05$) on CelebA faces and out-of-distribution images with RealNVP as the generative prior. Hyperparameter $\lambda = 0.5$.

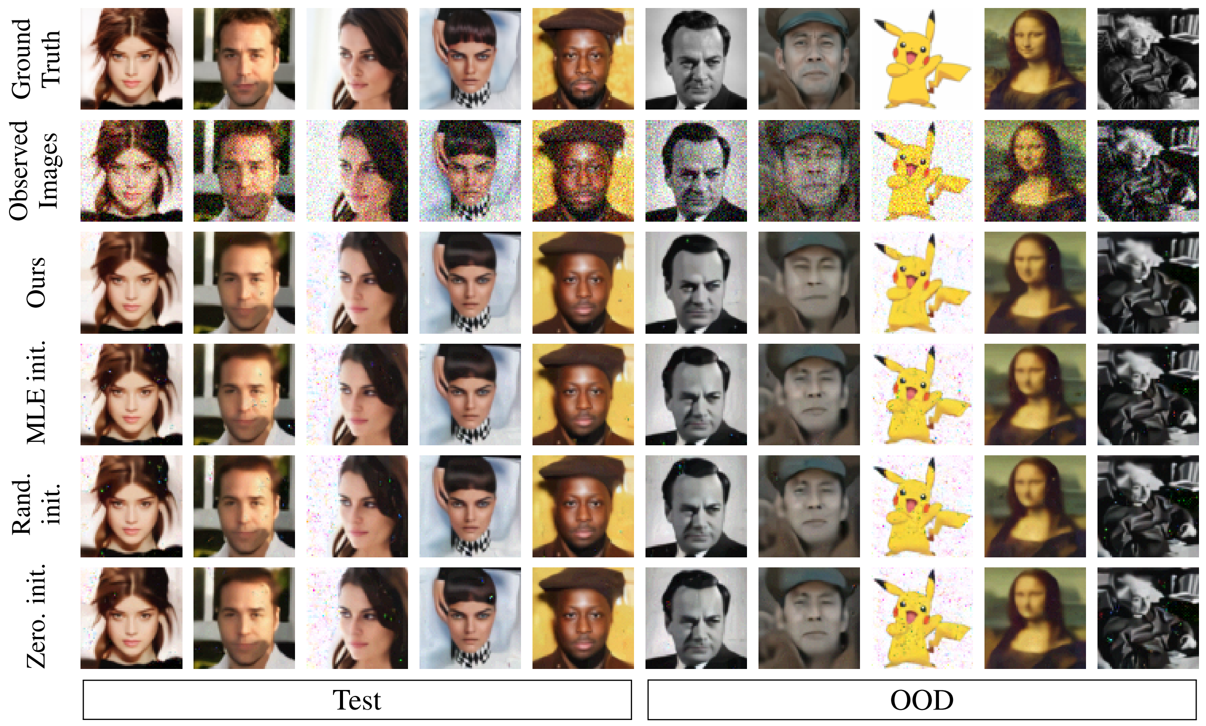

Figure 6. Results of solving denoising task ($\sigma = 0.1$) on CelebA faces and out-of-distribution images with RealNVP as the generative prior. Hyperparameter $\lambda = 0.5$.

Table 7. PSNR (mean±se) of different algorithms using GLOW as the generative prior for denoising, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
				Denoising, $\sigma = 0.05$		
Test	Ours	32.95 ± 0.06	**33.99 ± 0.06**	32.33 ± 0.05	30.94 ± 0.06	29.91 ± 0.06
	MLE. init.	**33.04 ± 0.06**	33.93 ± 0.06	**32.41 ± 0.06**	**31.04 ± 0.06**	30.05 ± 0.06
	Rand. init.	32.99 ± 0.06	33.95 ± 0.06	32.39 ± 0.06	**31.04 ± 0.06**	**30.06 ± 0.06**
	Zero. init.	33.01 ± 0.06	33.93 ± 0.06	32.40 ± 0.06	**31.04 ± 0.06**	30.05 ± 0.06
OOD	Ours	32.50 ± 0.39	**33.55 ± 0.39**	31.76 ± 0.52	30.21 ± 0.58	29.10 ± 0.62
	MLE. init.	32.64 ± 0.49	33.44 ± 0.47	31.71 ± 0.55	30.34 ± 0.57	**29.32 ± 0.63**
	Rand. init.	**32.69 ± 0.45**	33.48 ± 0.46	31.70 ± 0.52	30.21 ± 0.58	29.20 ± 0.59
	Zero. init.	32.68 ± 0.48	33.53 ± 0.46	**31.78 ± 0.51**	**30.38 ± 0.56**	29.29 ± 0.59
				Denoising, $\sigma = 0.1$		
Test	Ours	29.32 ± 0.09	29.29 ± 0.06	27.29 ± 0.07	25.88 ± 0.07	24.81 ± 0.07
	MLE. init.	29.46 ± 0.09	29.37 ± 0.06	27.41 ± 0.07	**25.96 ± 0.07**	**24.90 ± 0.07**
	Rand. init.	**29.51 ± 0.09**	**29.41 ± 0.06**	**27.42 ± 0.07**	25.94 ± 0.07	24.85 ± 0.06
	Zero. init.	29.49 ± 0.09	29.38 ± 0.06	27.41 ± 0.06	25.94 ± 0.07	24.84 ± 0.07
OOD	Ours	28.72 ± 0.62	28.54 ± 0.63	26.36 ± 0.57	24.97 ± 0.48	**24.05 ± 0.53**
	MLE. init.	28.70 ± 0.68	28.50 ± 0.56	26.29 ± 0.50	**25.11 ± 0.50**	23.98 ± 0.46
	Rand. init.	**28.94 ± 0.63**	28.44 ± 0.56	**26.52 ± 0.56**	24.96 ± 0.47	23.94 ± 0.45
	Zero. init.	28.86 ± 0.64	**28.56 ± 0.57**	26.37 ± 0.55	25.01 ± 0.50	23.87 ± 0.49

Table 8. SSIM (mean±se) of different algorithms using GLOW as the generative prior for denoising, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
				Denoising, $\sigma = 0.05$		
Test	Ours	93.51 ± 0.25	**94.55 ± 0.22**	92.07 ± 0.24	89.70 ± 0.27	87.57 ± 0.30
	MLE. init.	93.71 ± 0.24	94.49 ± 0.22	**92.23 ± 0.23**	**89.92 ± 0.27**	**87.96 ± 0.29**
	Rand. init.	93.68 ± 0.25	**94.55 ± 0.21**	92.16 ± 0.25	89.91 ± 0.28	87.94 ± 0.29
	Zero. init.	**93.72 ± 0.24**	94.54 ± 0.21	92.20 ± 0.24	89.90 ± 0.27	87.93 ± 0.30
OOD	Ours	93.33 ± 1.14	**94.48 ± 0.78**	**90.64 ± 1.58**	86.82 ± 2.11	83.36 ± 2.78
	MLE. init.	93.28 ± 1.22	94.21 ± 0.96	90.59 ± 1.48	87.12 ± 2.12	**84.62 ± 2.61**
	Rand. init.	93.34 ± 1.32	94.33 ± 0.97	90.48 ± 1.46	86.77 ± 2.19	83.65 ± 2.78
	Zero. init.	**93.47 ± 1.18**	94.18 ± 1.17	90.62 ± 1.45	**87.33 ± 2.04**	84.18 ± 2.61
				Denoising, $\sigma = 0.1$		
Test	Ours	87.88 ± 0.40	87.00 ± 0.33	80.77 ± 0.41	75.29 ± 0.49	70.45 ± 0.56
	MLE. init.	88.21 ± 0.39	87.30 ± 0.32	81.21 ± 0.40	**75.55 ± 0.49**	**70.69 ± 0.56**
	Rand. init.	**88.32 ± 0.39**	**87.37 ± 0.32**	**81.25 ± 0.41**	75.49 ± 0.50	70.58 ± 0.56
	Zero. init.	88.25 ± 0.39	87.33 ± 0.32	81.22 ± 0.40	75.53 ± 0.49	70.62 ± 0.56
OOD	Ours	86.15 ± 2.00	83.18 ± 2.48	74.13 ± 3.28	68.01 ± 3.84	**62.84 ± 4.37**
	MLE. init.	86.39 ± 1.97	83.44 ± 2.31	74.60 ± 3.13	**69.11 ± 3.56**	62.68 ± 4.61
	Rand. init.	86.89 ± 1.91	83.47 ± 2.47	**75.71 ± 3.12**	68.03 ± 3.39	62.51 ± 4.11
	Zero. init.	**86.91 ± 1.95**	**83.51 ± 2.33**	75.04 ± 3.10	68.79 ± 3.42	62.25 ± 4.16

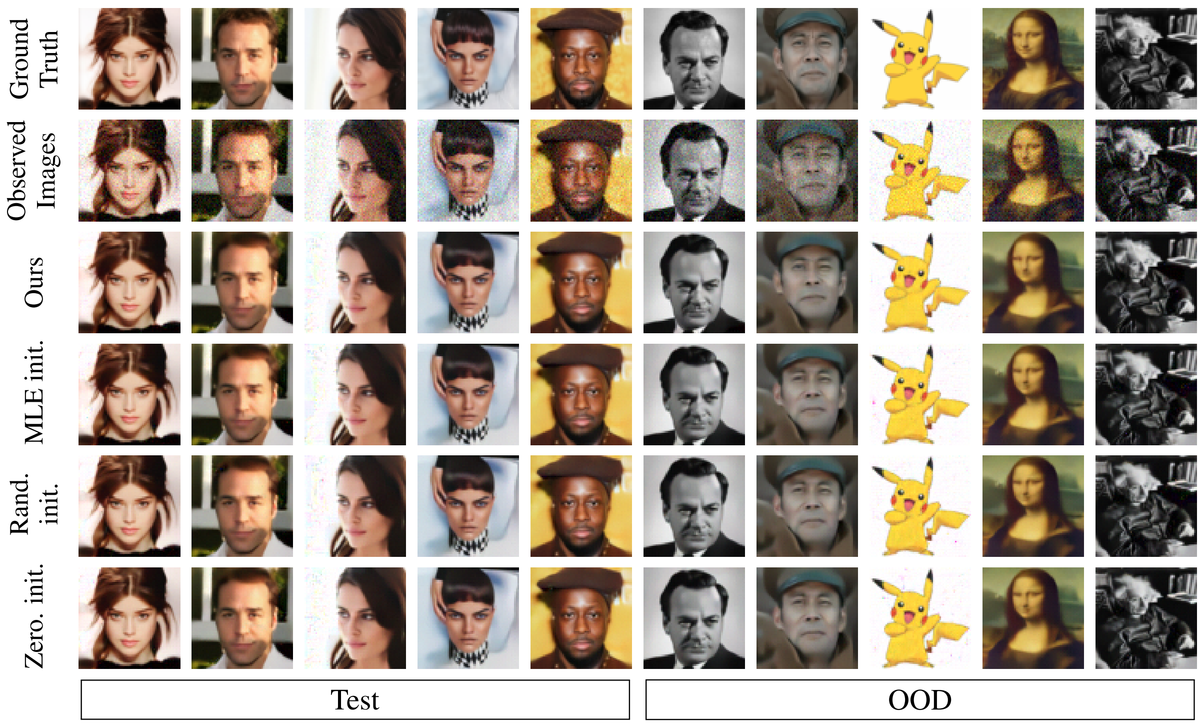

Figure 7. Results of solving denoising task ($\sigma = 0.05$) on CelebA faces and out-of-distribution images with GLOW as the generative prior. Hyperparameter $\lambda = 0.5$.

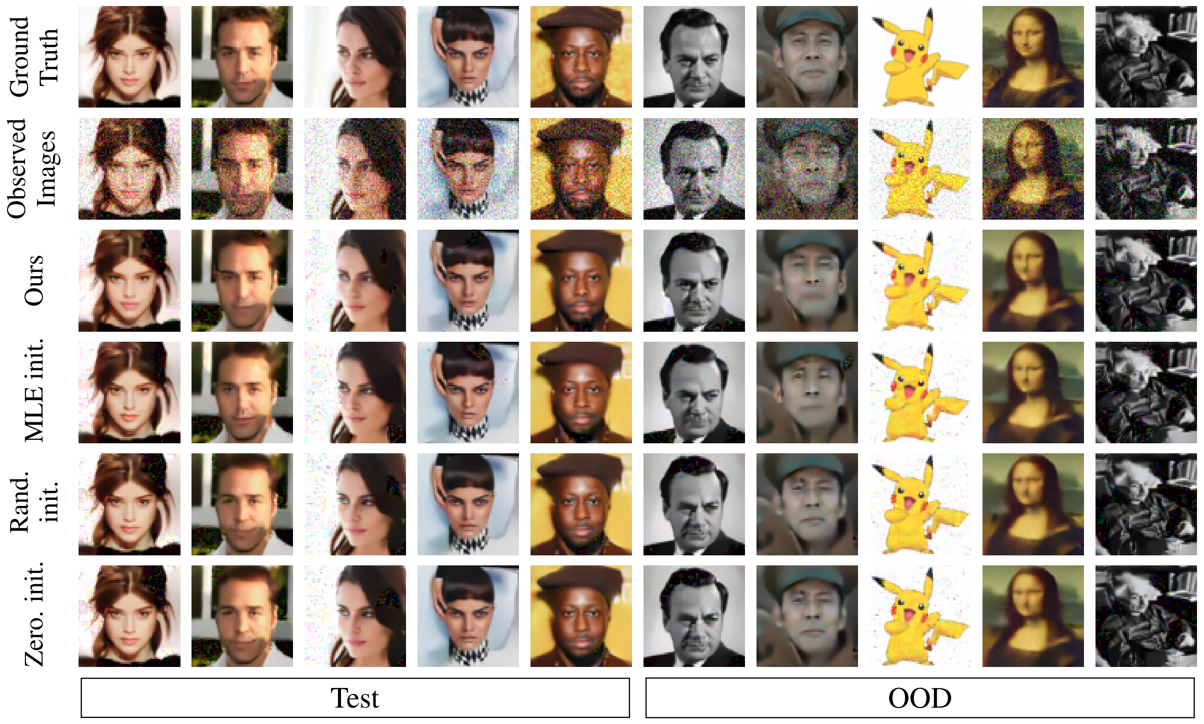

Figure 8. Results of solving denoising task ($\sigma = 0.1$) on CelebA faces and out-of-distribution images with GLOW as the generative prior. Hyperparameter $\lambda = 0.3$.

Table 9. PSNR (mean$\pm$se) of different algorithms using RealNVP as the generative prior for NCS, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
		Noisy Compressed Sensing when $m = 4000$				
Test	Ours	**29.53 ± 0.09**	**29.28 ± 0.07**	**27.70 ± 0.07**	**26.60 ± 0.07**	25.69 ± 0.07
	MLE. init.	29.14 ± 0.10	28.93 ± 0.08	27.54 ± 0.08	26.43 ± 0.08	25.67 ± 0.08
	Rand. init.	29.37 ± 0.08	29.03 ± 0.08	27.60 ± 0.08	26.52 ± 0.08	25.69 ± 0.07
	Zero. init.	29.39 ± 0.08	29.01 ± 0.08	27.60 ± 0.08	26.47 ± 0.07	**25.74 ± 0.08**
OOD	Ours	28.11 ± 0.83	**28.19 ± 0.68**	26.72 ± 0.68	**25.82 ± 0.70**	**24.92 ± 0.54**
	MLE. init.	27.77 ± 0.94	27.80 ± 0.84	26.58 ± 0.87	25.32 ± 0.84	24.72 ± 0.68
	Rand. init.	**28.19 ± 0.79**	28.09 ± 0.73	**26.81 ± 0.78**	25.56 ± 0.73	24.80 ± 0.67
	Zero. init.	27.99 ± 0.88	28.10 ± 0.80	26.93 ± 0.73	25.61 ± 0.69	24.87 ± 0.66
		Noisy Compressed Sensing when $m = 2000$				
Test	Ours	**28.90 ± 0.09**	**28.48 ± 0.08**	**27.17 ± 0.07**	**26.15 ± 0.07**	**25.42 ± 0.07**
	MLE. init.	28.52 ± 0.10	28.20 ± 0.09	27.03 ± 0.09	26.08 ± 0.08	25.37 ± 0.08
	Rand. init.	28.73 ± 0.09	28.30 ± 0.08	27.04 ± 0.08	26.08 ± 0.08	25.31 ± 0.08
	Zero. init.	28.72 ± 0.09	28.28 ± 0.09	27.06 ± 0.08	26.11 ± 0.08	25.30 ± 0.07
OOD	Ours	27.21 ± 0.92	**27.26 ± 0.81**	25.92 ± 0.74	**25.30 ± 0.62**	**24.57 ± 0.66**
	MLE. init.	26.85 ± 1.01	26.63 ± 1.02	25.79 ± 0.92	25.00 ± 0.82	24.10 ± 0.86
	Rand. init.	**27.29 ± 0.85**	27.10 ± 0.84	25.99 ± 0.81	25.08 ± 0.78	24.37 ± 0.73
	Zero. init.	27.27 ± 0.90	27.09 ± 0.95	**26.20 ± 0.79**	25.24 ± 0.75	24.45 ± 0.68
		Noisy Compressed Sensing when $m = 1000$				
Test	Ours	**27.41 ± 0.11**	**27.06 ± 0.09**	**26.07 ± 0.08**	**25.39 ± 0.08**	**24.70 ± 0.07**
	MLE. init.	27.08 ± 0.12	26.80 ± 0.11	26.01 ± 0.10	25.28 ± 0.09	24.69 ± 0.09
	Rand. init.	27.25 ± 0.10	26.93 ± 0.10	26.01 ± 0.09	25.22 ± 0.08	24.57 ± 0.08
	Zero. init.	27.21 ± 0.11	26.92 ± 0.10	25.99 ± 0.09	25.19 ± 0.08	24.60 ± 0.08
OOD	Ours	25.25 ± 1.07	**25.20 ± 1.07**	**24.66 ± 0.86**	24.05 ± 0.72	**23.58 ± 0.68**
	MLE. init.	25.04 ± 1.27	25.06 ± 1.17	24.58 ± 1.00	23.86 ± 0.83	23.54 ± 0.87
	Rand. init.	25.23 ± 1.09	**25.20 ± 1.02**	24.64 ± 0.87	**24.20 ± 0.84**	23.57 ± 0.70
	Zero. init.	**25.35 ± 1.12**	25.17 ± 1.01	24.59 ± 0.90	24.15 ± 0.83	23.55 ± 0.82

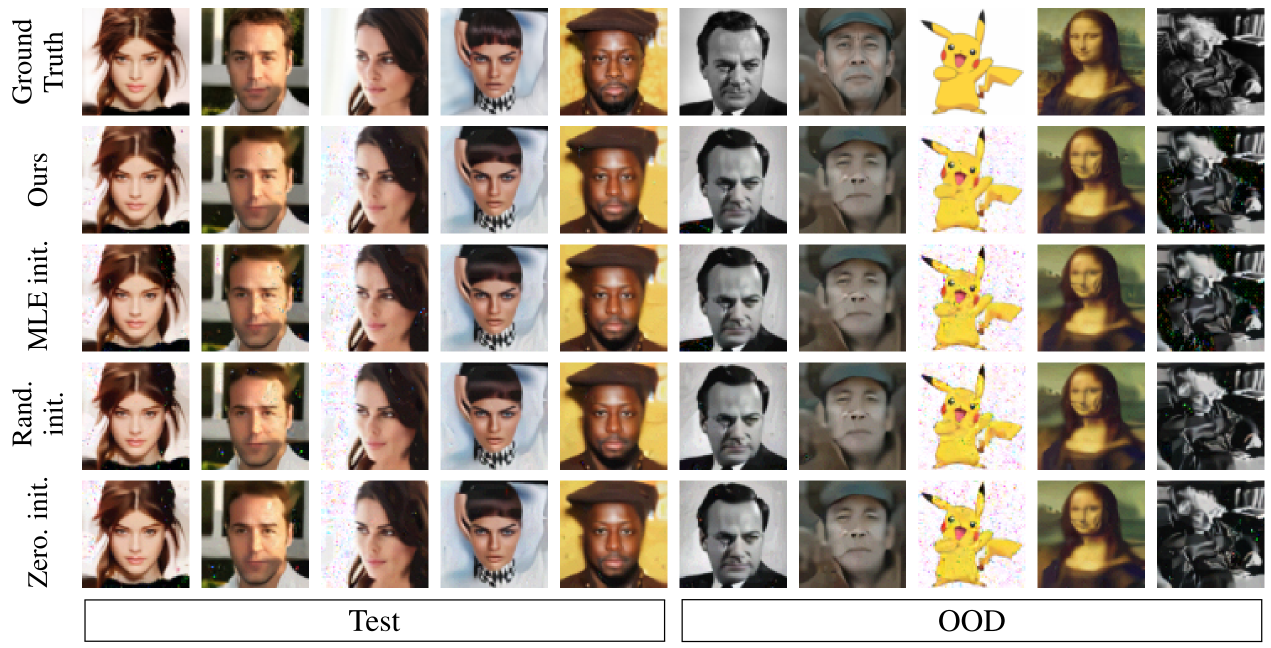

Figure 9. Results of solving NCS when $m = 4000$ on CelebA faces and out-of-distribution images with RealNVP as the generative prior. Hyperparameter $\lambda = 0.5$.

Table 10. SSIM (mean±se) of different algorithms using RealNVP as the generative prior for NCS, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
		Noisy Compressed Sensing when $m = 4000$				
Test	Ours	**88.20 ± 0.40**	**87.42 ± 0.34**	83.18 ± 0.36	79.71 ± 0.39	76.48 ± 0.42
	MLE. init.	87.38 ± 0.42	86.81 ± 0.37	83.29 ± 0.38	80.00 ± 0.41	77.43 ± 0.41
	Rand. init.	88.06 ± 0.34	87.14 ± 0.32	**83.58 ± 0.33**	**80.24 ± 0.37**	77.36 ± 0.40
	Zero. init.	88.11 ± 0.34	87.13 ± 0.31	83.54 ± 0.34	80.21 ± 0.36	**77.54 ± 0.39**
OOD	Ours	84.21 ± 2.43	83.57 ± 2.62	77.44 ± 3.15	**73.76 ± 3.79**	69.96 ± 3.54
	MLE. init.	83.14 ± 3.33	82.68 ± 3.10	77.91 ± 3.65	72.72 ± 3.68	69.44 ± 3.47
	Rand. init.	**84.23 ± 3.17**	**83.58 ± 2.88**	78.15 ± 3.51	73.01 ± 3.47	69.63 ± 3.58
	Zero. init.	83.73 ± 3.37	83.38 ± 3.25	**78.85 ± 3.22**	73.75 ± 3.30	**70.37 ± 3.31**
		Noisy Compressed Sensing when $m = 2000$				
Test	Ours	**86.97 ± 0.38**	85.63 ± 0.36	81.77 ± 0.37	78.33 ± 0.40	75.74 ± 0.42
	MLE. init.	86.21 ± 0.42	85.30 ± 0.39	81.97 ± 0.39	79.00 ± 0.41	**76.45 ± 0.42**
	Rand. init.	86.84 ± 0.34	**85.65 ± 0.33**	82.13 ± 0.34	78.91 ± 0.37	76.08 ± 0.40
	Zero. init.	86.75 ± 0.35	85.64 ± 0.33	**82.23 ± 0.35**	**79.04 ± 0.37**	76.03 ± 0.39
OOD	Ours	81.55 ± 3.45	**81.04 ± 2.90**	75.24 ± 3.63	**72.69 ± 3.13**	**68.78 ± 3.65**
	MLE. init.	80.66 ± 3.92	79.63 ± 4.16	75.28 ± 4.13	71.69 ± 3.80	67.64 ± 4.19
	Rand. init.	**81.86 ± 3.47**	80.92 ± 3.57	75.77 ± 3.66	71.54 ± 3.82	67.90 ± 3.91
	Zero. init.	81.49 ± 3.73	80.34 ± 3.99	**76.16 ± 3.69**	72.63 ± 3.47	68.30 ± 3.61
		Noisy Compressed Sensing when $m = 1000$				
Test	Ours	83.40 ± 0.43	82.12 ± 0.41	78.65 ± 0.41	76.16 ± 0.42	73.33 ± 0.44
	MLE. init.	82.91 ± 0.44	81.87 ± 0.43	**79.23 ± 0.42**	**76.53 ± 0.43**	**74.15 ± 0.43**
	Rand. init.	**83.42 ± 0.37**	82.25 ± 0.37	79.14 ± 0.38	76.00 ± 0.41	73.39 ± 0.41
	Zero. init.	83.37 ± 0.37	**82.26 ± 0.37**	79.07 ± 0.39	75.94 ± 0.41	73.48 ± 0.42
OOD	Ours	74.21 ± 4.07	73.77 ± 4.00	70.23 ± 3.79	67.67 ± 3.58	**64.59 ± 3.42**
	MLE. init.	73.77 ± 5.34	73.69 ± 4.90	70.59 ± 4.49	67.27 ± 4.14	64.31 ± 4.37
	Rand. init.	74.31 ± 4.91	73.73 ± 4.39	**70.62 ± 4.03**	**67.90 ± 4.06**	64.06 ± 3.88
	Zero. init.	**74.74 ± 4.88**	**73.82 ± 4.20**	70.04 ± 4.18	67.06 ± 4.10	64.09 ± 4.34

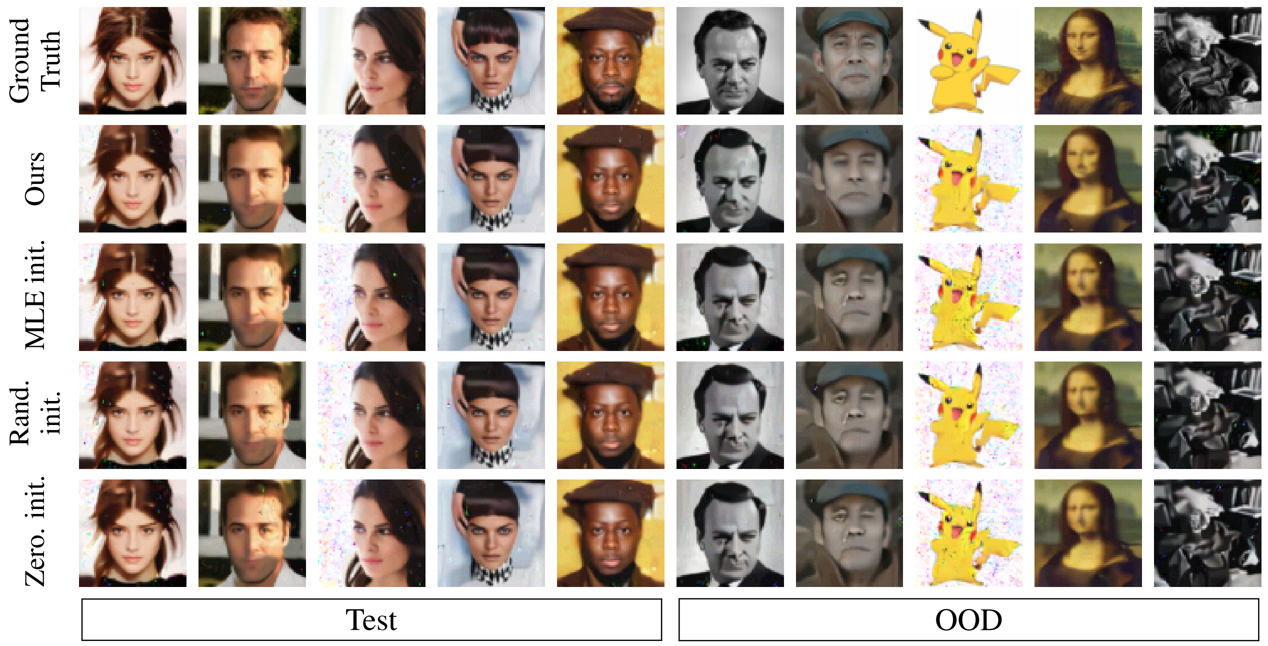

Figure 10. Results of solving NCS when $m = 2000$ on CelebA faces and out-of-distribution images with RealNVP as the generative prior. Hyperparameter $\lambda = 0.3$.

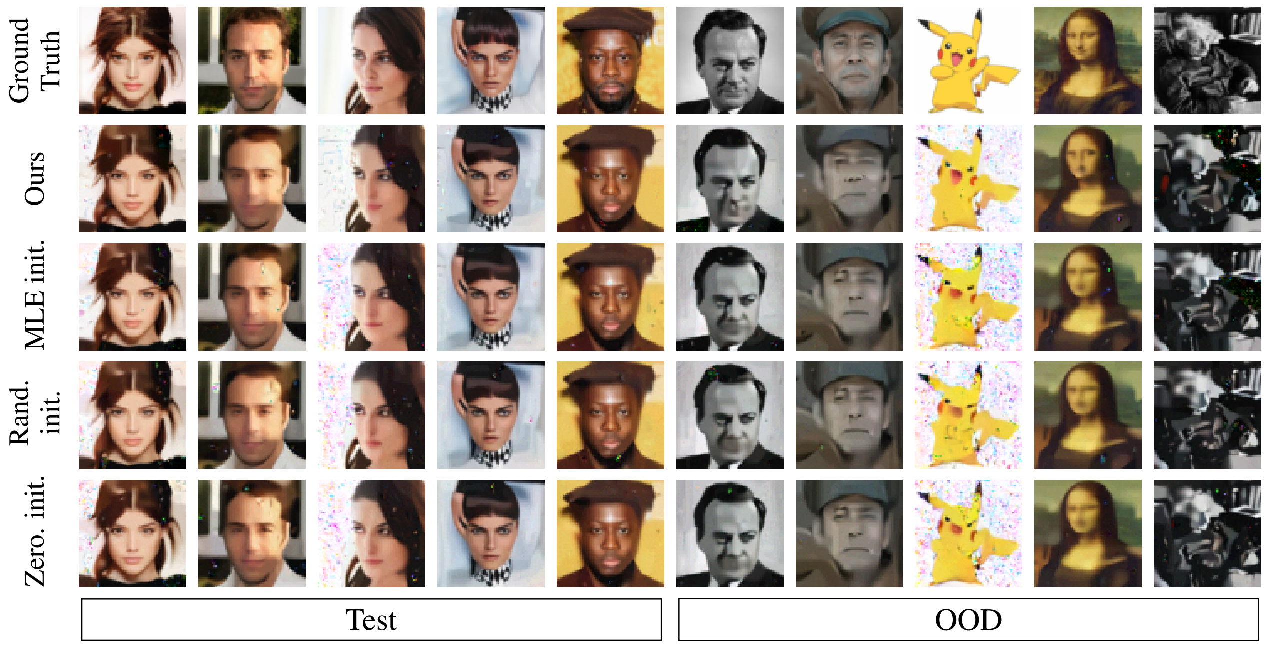

Figure 11. Results of solving NCS when $m = 1000$ on CelebA faces and out-of-distribution images with RealNVP as the generative prior. Hyperparameter $\lambda = 0.3$.

Table 11. PSNR (mean±se) of different algorithms using GLOW as the generative prior for NCS, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
				Noisy Compressed Sensing when $m = 4000$		
Test	Ours	**28.78 ± 0.10**	**28.50 ± 0.07**	26.80 ± 0.07	25.53 ± 0.07	**24.51 ± 0.07**
	MLE. init.	28.38 ± 0.11	28.47 ± 0.08	**26.82 ± 0.07**	**25.54 ± 0.08**	24.49 ± 0.07
	Rand. init.	28.19 ± 0.11	28.43 ± 0.08	26.78 ± 0.07	25.37 ± 0.07	24.36 ± 0.07
	Zero. init.	28.20 ± 0.11	28.39 ± 0.07	26.74 ± 0.07	25.38 ± 0.07	24.31 ± 0.06
OOD	Ours	**27.78 ± 0.78**	**27.48 ± 0.72**	**25.80 ± 0.60**	**24.71 ± 0.54**	**23.70 ± 0.45**
	MLE. init.	27.05 ± 0.98	27.31 ± 0.79	25.65 ± 0.68	24.38 ± 0.57	23.48 ± 0.54
	Rand. init.	26.88 ± 0.95	27.29 ± 0.77	25.75 ± 0.64	24.38 ± 0.49	23.49 ± 0.48
	Zero. init.	26.84 ± 0.96	27.12 ± 0.76	25.68 ± 0.64	24.38 ± 0.48	23.53 ± 0.48
				Noisy Compressed Sensing $m = 2000$		
Test	Ours	**28.02 ± 0.11**	**27.66 ± 0.09**	**26.26 ± 0.07**	**25.12 ± 0.07**	**24.15 ± 0.06**
	MLE. init.	25.04 ± 0.21	26.09 ± 0.16	25.68 ± 0.10	24.68 ± 0.08	23.75 ± 0.07
	Rand. init.	26.04 ± 0.16	26.75 ± 0.13	25.85 ± 0.08	24.65 ± 0.07	23.72 ± 0.07
	Zero. init.	25.76 ± 0.18	26.62 ± 0.14	25.82 ± 0.08	24.59 ± 0.07	23.71 ± 0.07
OOD	Ours	**26.64 ± 0.94**	**26.57 ± 0.78**	**25.33 ± 0.61**	**24.24 ± 0.57**	**23.42 ± 0.48**
	MLE. init.	22.19 ± 1.24	24.72 ± 1.30	24.12 ± 0.97	23.51 ± 0.80	22.82 ± 0.53
	Rand. init.	23.96 ± 1.44	25.26 ± 0.89	23.20 ± 1.38	23.87 ± 0.57	22.98 ± 0.51
	Zero. init.	23.71 ± 1.39	25.06 ± 1.13	23.42 ± 1.27	23.78 ± 0.60	22.86 ± 0.58
				Noisy Compressed Sensing when $m = 1000$		
Test	Ours	**25.52 ± 0.15**	**25.77 ± 0.13**	**24.98 ± 0.09**	**24.13 ± 0.08**	**23.39 ± 0.08**
	MLE. init.	12.55 ± 0.20	12.83 ± 0.21	13.48 ± 0.23	13.79 ± 0.24	14.19 ± 0.24
	Rand. init.	18.30 ± 0.17	18.79 ± 0.19	19.86 ± 0.19	20.20 ± 0.17	20.38 ± 0.16
	Zero. init.	17.99 ± 0.17	18.35 ± 0.18	19.54 ± 0.20	20.20 ± 0.17	20.13 ± 0.15
OOD	Ours	**23.88 ± 1.38**	**23.87 ± 1.25**	**23.16 ± 1.24**	**22.84 ± 0.87**	**22.52 ± 0.72**
	MLE. init.	11.78 ± 0.91	11.52 ± 0.80	11.94 ± 0.74	12.01 ± 0.88	12.03 ± 0.91
	Rand. init.	17.05 ± 1.29	17.78 ± 1.64	18.90 ± 1.84	19.57 ± 1.85	19.94 ± 1.48
	Zero. init.	17.30 ± 1.53	17.47 ± 1.42	18.68 ± 1.89	19.63 ± 1.83	19.35 ± 1.83

Table 12. SSIM (mean±se) of different algorithms using GLOW as the generative prior for NCS, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
		Noisy Compressed Sensing when $m = 4000$				
Test	Ours	**86.68 ± 0.42**	85.34 ± 0.35	79.51 ± 0.41	74.17 ± 0.50	**69.68 ± 0.56**
	MLE. init.	85.85 ± 0.47	**85.38 ± 0.37**	**79.67 ± 0.43**	**74.30 ± 0.51**	69.51 ± 0.55
	Rand. init.	85.52 ± 0.45	85.28 ± 0.37	79.47 ± 0.41	73.52 ± 0.50	68.89 ± 0.54
	Zero. init.	85.56 ± 0.46	85.18 ± 0.37	79.30 ± 0.42	73.52 ± 0.51	68.76 ± 0.52
OOD	Ours	**83.89 ± 2.18**	**80.66 ± 2.59**	**72.88 ± 3.32**	**68.18 ± 3.65**	**61.94 ± 4.73**
	MLE. init.	81.26 ± 2.53	80.39 ± 2.62	72.55 ± 3.30	66.27 ± 3.62	60.64 ± 4.54
	Rand. init.	80.80 ± 2.59	79.84 ± 2.78	72.75 ± 3.29	66.19 ± 3.49	60.75 ± 4.45
	Zero. init.	80.70 ± 2.57	79.88 ± 2.41	72.56 ± 3.25	65.86 ± 3.56	61.12 ± 4.51
		Noisy Compressed Sensing when $m = 2000$				
Test	Ours	**84.91 ± 0.44**	**83.31 ± 0.40**	**77.97 ± 0.45**	**72.89 ± 0.50**	**68.38 ± 0.56**
	MLE. init.	76.75 ± 0.83	78.93 ± 0.64	75.89 ± 0.56	71.05 ± 0.54	66.70 ± 0.56
	Rand. init.	80.06 ± 0.58	80.96 ± 0.49	76.17 ± 0.50	70.68 ± 0.51	66.41 ± 0.53
	Zero. init.	79.18 ± 0.64	80.51 ± 0.52	76.02 ± 0.48	70.44 ± 0.52	66.31 ± 0.53
OOD	Ours	**79.72 ± 2.99**	**77.99 ± 2.94**	**71.87 ± 3.09**	**66.32 ± 3.71**	**61.73 ± 3.76**
	MLE. init.	63.29 ± 5.30	71.76 ± 4.17	66.66 ± 3.49	62.71 ± 3.88	58.83 ± 3.65
	Rand. init.	69.53 ± 5.79	72.94 ± 3.37	63.01 ± 5.74	63.30 ± 3.61	58.52 ± 3.58
	Zero. init.	68.80 ± 5.77	72.19 ± 3.88	63.62 ± 5.51	63.16 ± 3.49	57.75 ± 3.73
		Noisy Compressed Sensing when $m = 1000$				
Test	Ours	**78.16 ± 0.60**	**78.09 ± 0.56**	**73.95 ± 0.54**	**69.68 ± 0.56**	**65.88 ± 0.58**
	MLE. init.	26.94 ± 0.67	28.22 ± 0.71	30.78 ± 0.80	32.17 ± 0.81	33.94 ± 0.81
	Rand. init.	51.54 ± 0.68	52.35 ± 0.77	55.03 ± 0.78	55.10 ± 0.75	54.77 ± 0.71
	Zero. init.	49.92 ± 0.67	50.89 ± 0.72	53.75 ± 0.77	55.05 ± 0.72	53.86 ± 0.71
OOD	Ours	**69.39 ± 4.91**	**67.15 ± 4.75**	**63.19 ± 4.98**	**60.62 ± 4.36**	**58.10 ± 4.41**
	MLE. init.	19.58 ± 2.78	19.69 ± 2.03	22.17 ± 2.72	22.13 ± 2.58	21.85 ± 2.69
	Rand. init.	43.77 ± 5.24	44.82 ± 7.62	47.27 ± 7.61	47.83 ± 6.99	48.09 ± 5.88
	Zero. init.	43.13 ± 7.08	43.86 ± 6.14	46.05 ± 7.54	47.69 ± 7.08	45.99 ± 7.19

Table 13. PSNR (mean±se) of different algorithms using RealNVP as the generative prior for inpainting, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
		Inpainting when mask 25%				
Test	Ours	**32.73 ± 0.14**	**33.06 ± 0.15**	**33.19 ± 0.14**	**32.83 ± 0.14**	**32.50 ± 0.13**
	MLE. init.	31.80 ± 0.18	32.10 ± 0.18	32.02 ± 0.19	31.69 ± 0.18	31.52 ± 0.17
	Rand. init.	31.30 ± 0.21	31.34 ± 0.22	31.33 ± 0.22	31.38 ± 0.20	31.31 ± 0.18
	Zero. init.	31.21 ± 0.22	31.33 ± 0.23	31.42 ± 0.21	31.55 ± 0.20	31.20 ± 0.18
OOD	Ours	**28.68 ± 1.84**	**28.99 ± 1.84**	**28.86 ± 1.80**	**29.13 ± 1.62**	**28.57 ± 1.42**
	MLE. init.	27.81 ± 2.05	28.02 ± 1.94	27.83 ± 1.96	27.75 ± 2.03	28.20 ± 1.75
	Rand. init.	27.63 ± 2.06	28.03 ± 2.20	27.96 ± 2.20	27.77 ± 1.93	27.83 ± 1.98
	Zero. init.	27.55 ± 2.11	27.98 ± 2.12	27.92 ± 2.09	27.95 ± 1.96	27.93 ± 1.98
		Inpainting when mask 30%				
Test	Ours	**31.15 ± 0.19**	**31.42 ± 0.19**	**31.48 ± 0.19**	**31.26 ± 0.18**	**30.99 ± 0.16**
	MLE. init.	30.30 ± 0.21	30.40 ± 0.23	30.42 ± 0.22	30.06 ± 0.20	29.96 ± 0.21
	Rand. init.	29.77 ± 0.22	29.74 ± 0.24	29.81 ± 0.23	29.93 ± 0.22	29.76 ± 0.20
	Zero. init.	29.81 ± 0.22	29.82 ± 0.22	29.95 ± 0.22	29.82 ± 0.21	29.78 ± 0.20
OOD	Ours	**27.60 ± 1.75**	**27.52 ± 2.13**	**28.20 ± 1.77**	**27.23 ± 1.99**	**27.95 ± 1.57**
	MLE. init.	25.98 ± 2.04	26.17 ± 2.01	26.24 ± 2.28	26.11 ± 2.06	26.75 ± 1.90
	Rand. init.	25.97 ± 1.91	26.57 ± 2.08	26.55 ± 2.06	26.34 ± 1.74	25.89 ± 2.14
	Zero. init.	25.96 ± 1.98	26.20 ± 2.00	26.69 ± 2.02	26.57 ± 1.95	26.61 ± 1.86

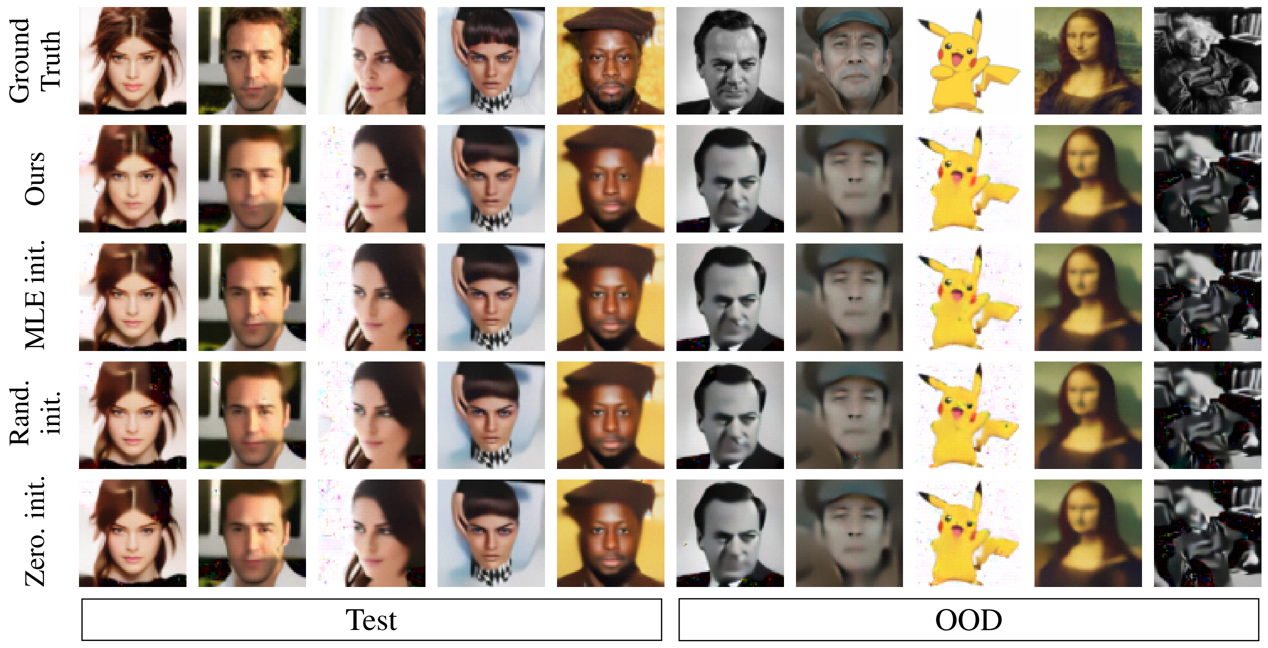

Figure 12. Results of solving NCS when $m = 4000$ on CelebA faces and out-of-distribution images with GLOW as the generative prior. Hyperparameter $\lambda = 0.5$.

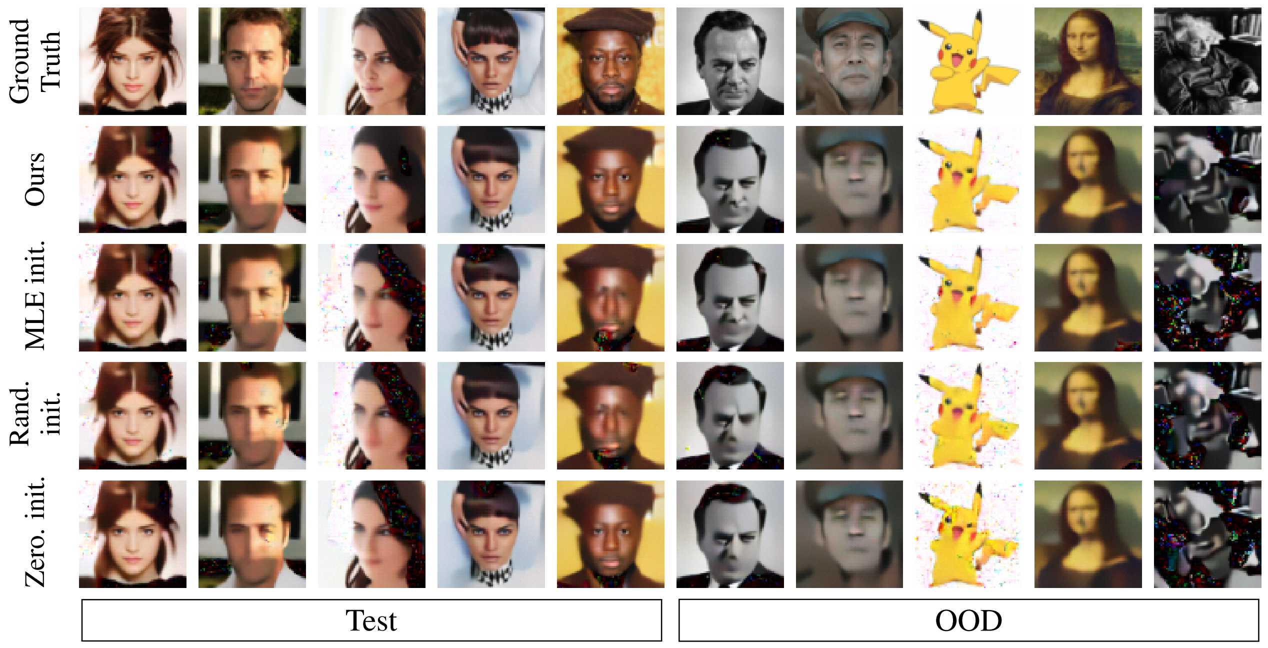

Figure 13. Results of solving NCS when $m = 2000$ on CelebA faces and out-of-distribution images with GLOW as the generative prior. Hyperparameter $\lambda = 0.3$.

Table 14. SSIM (mean±se) of different algorithms using RealNVP as the generative prior for inpainting, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
		Inpainting when mask 25%				
Test	Ours	**93.91 ± 0.20**	**94.25 ± 0.21**	**94.26 ± 0.21**	**93.82 ± 0.22**	**93.38 ± 0.22**
	MLE. init.	92.95 ± 0.25	93.32 ± 0.26	93.12 ± 0.28	92.60 ± 0.28	92.13 ± 0.29
	Rand. init.	92.50 ± 0.29	92.62 ± 0.31	92.35 ± 0.33	92.16 ± 0.32	91.87 ± 0.32
	Zero. init.	92.31 ± 0.31	92.51 ± 0.33	92.43 ± 0.33	92.36 ± 0.31	91.79 ± 0.32
OOD	Ours	**89.32 ± 2.92**	**89.97 ± 2.83**	**89.69 ± 2.97**	**89.84 ± 2.57**	**88.99 ± 2.43**
	MLE. init.	87.92 ± 3.73	88.10 ± 3.67	87.76 ± 3.83	86.95 ± 4.00	87.35 ± 3.77
	Rand. init.	86.96 ± 4.12	87.30 ± 4.30	87.42 ± 4.29	87.00 ± 4.16	86.48 ± 4.12
	Zero. init.	87.11 ± 3.93	87.35 ± 4.11	87.26 ± 4.12	87.31 ± 3.95	86.89 ± 4.00
		Inpainting when mask 30%				
Test	Ours	**92.63 ± 0.24**	**93.02 ± 0.24**	**93.06 ± 0.24**	**92.57 ± 0.24**	**92.06 ± 0.24**
	MLE. init.	91.65 ± 0.29	91.93 ± 0.29	91.61 ± 0.32	91.08 ± 0.30	90.65 ± 0.33
	Rand. init.	90.89 ± 0.35	91.10 ± 0.36	91.00 ± 0.37	90.76 ± 0.35	90.31 ± 0.35
	Zero. init.	90.96 ± 0.35	91.08 ± 0.35	91.04 ± 0.36	90.66 ± 0.36	90.42 ± 0.34
OOD	Ours	**86.74 ± 3.68**	**86.87 ± 4.31**	**87.54 ± 3.58**	**88.87 ± 1.94**	**86.94 ± 2.85**
	MLE. init.	84.60 ± 4.64	84.80 ± 4.48	84.93 ± 4.81	84.77 ± 4.68	84.45 ± 4.58
	Rand. init.	84.28 ± 4.56	84.47 ± 4.90	84.46 ± 4.78	84.37 ± 4.48	83.48 ± 5.13
	Zero. init.	83.82 ± 4.94	84.24 ± 4.91	85.01 ± 4.71	84.33 ± 4.90	85.13 ± 4.20

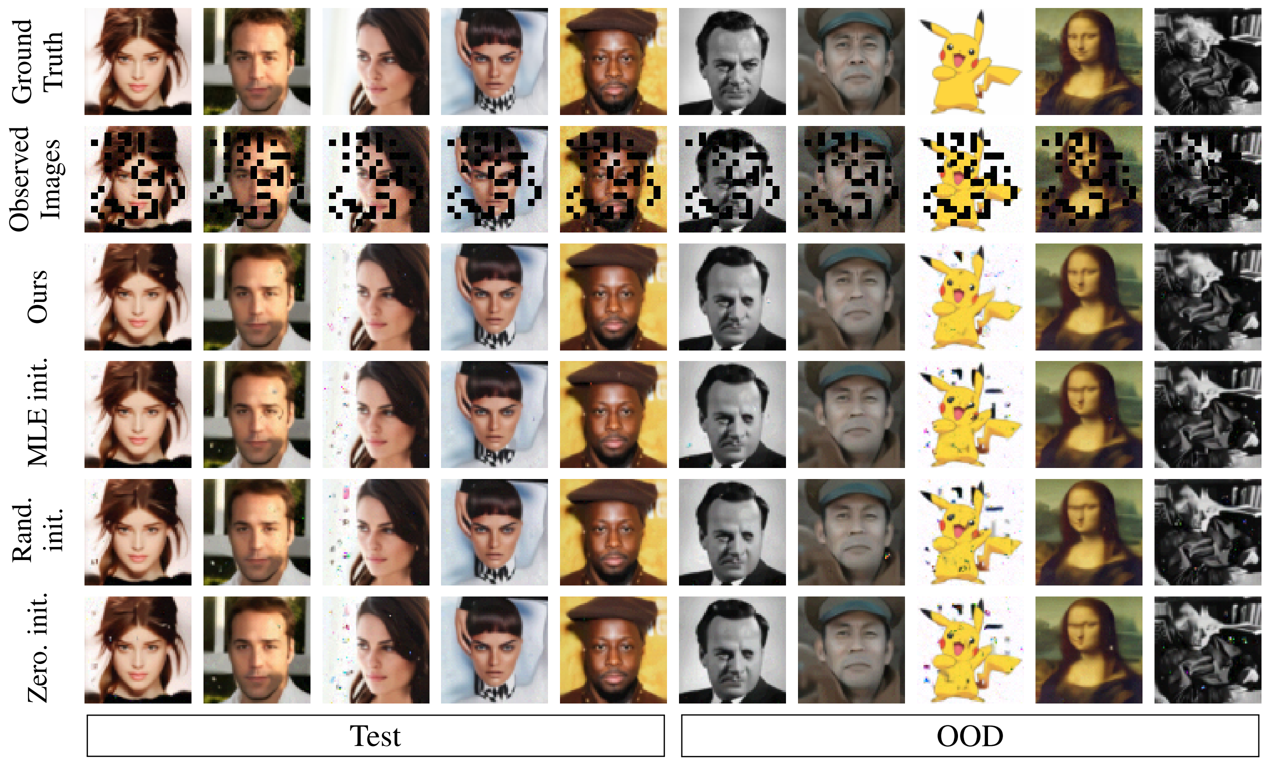

Figure 14. Results of inpainting CelebA faces (masked 25%) and out-of-distribution images with RealNVP as the generative prior. Hyperparameter $\lambda = 1.5$.

Table 15. PSNR (mean±se) of different algorithms using GLOW as the generative prior for inpainting, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
		Inpainting when mask 25%				
Test	Ours	**32.37 ± 0.22**	**32.78 ± 0.22**	**33.03 ± 0.20**	**32.97 ± 0.15**	**32.61 ± 0.13**
	MLE. init.	30.93 ± 0.28	31.43 ± 0.29	31.69 ± 0.28	32.04 ± 0.24	31.92 ± 0.20
	Rand. init.	31.61 ± 0.26	31.97 ± 0.27	32.21 ± 0.25	32.15 ± 0.23	31.92 ± 0.20
	Zero. init.	31.53 ± 0.26	32.02 ± 0.27	32.16 ± 0.26	32.03 ± 0.23	32.01 ± 0.20
OOD	Ours	**26.77 ± 2.28**	**27.18 ± 2.34**	**27.80 ± 2.20**	**28.67 ± 1.95**	**28.29 ± 1.71**
	MLE. init.	24.94 ± 2.40	25.05 ± 2.49	26.48 ± 2.29	27.93 ± 2.34	27.63 ± 2.18
	Rand. init.	26.25 ± 2.39	26.46 ± 2.38	27.70 ± 2.49	28.03 ± 2.33	27.62 ± 2.12
	Zero. init.	23.85 ± 2.11	26.09 ± 2.36	27.81 ± 2.47	28.09 ± 2.29	27.57 ± 2.13
		Inpainting when mask 30%				
Test	Ours	**28.64 ± 0.33**	**29.05 ± 0.34**	**29.74 ± 0.32**	**30.46 ± 0.26**	**30.47 ± 0.23**
	MLE. init.	26.41 ± 0.35	26.93 ± 0.37	27.46 ± 0.36	28.77 ± 0.32	29.26 ± 0.29
	Rand. init.	27.28 ± 0.33	28.07 ± 0.35	28.64 ± 0.34	29.27 ± 0.30	29.71 ± 0.27
	Zero. init.	27.28 ± 0.33	27.83 ± 0.35	28.45 ± 0.34	29.10 ± 0.31	29.53 ± 0.28
OOD	Ours	**25.50 ± 2.56**	**25.78 ± 2.61**	**25.97 ± 2.44**	**26.52 ± 2.31**	**26.57 ± 2.11**
	MLE. init.	22.68 ± 2.09	23.14 ± 2.26	24.74 ± 2.54	25.13 ± 2.48	25.56 ± 2.33
	Rand. init.	23.51 ± 2.23	24.89 ± 2.66	25.16 ± 2.59	25.55 ± 2.43	26.15 ± 2.27
	Zero. init.	23.99 ± 2.49	24.57 ± 2.61	24.95 ± 2.54	25.69 ± 2.40	26.01 ± 2.31

Table 16. SSIM (mean±se) of different algorithms using GLOW as the generative prior for inpainting, higher is better.

	Algorithm	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 1.5$	$\lambda = 2.0$
		Inpainting when mask 25%				
Test	Ours	**94.69 ± 0.17**	**95.17 ± 0.18**	**95.13 ± 0.17**	**94.69 ± 0.17**	**93.99 ± 0.18**
	MLE. init.	93.59 ± 0.26	94.14 ± 0.27	94.12 ± 0.26	93.90 ± 0.25	93.44 ± 0.24
	Rand. init.	93.96 ± 0.24	94.39 ± 0.26	94.32 ± 0.26	93.95 ± 0.25	93.41 ± 0.24
	Zero. init.	93.90 ± 0.24	94.47 ± 0.25	94.30 ± 0.26	93.88 ± 0.25	93.41 ± 0.25
OOD	Ours	**88.03 ± 3.31**	**88.93 ± 3.13**	**90.12 ± 2.57**	**90.05 ± 2.37**	**89.44 ± 2.01**
	MLE. init.	85.29 ± 4.09	85.32 ± 4.38	87.02 ± 4.31	88.18 ± 3.74	87.39 ± 3.58
	Rand. init.	86.42 ± 4.46	86.94 ± 4.49	87.60 ± 4.73	87.90 ± 3.98	87.24 ± 3.68
	Zero. init.	83.92 ± 4.71	86.56 ± 4.49	88.06 ± 4.34	88.08 ± 3.82	87.37 ± 3.53
		Inpainting when mask 30%				
Test	Ours	**92.50 ± 0.24**	**93.06 ± 0.25**	**93.27 ± 0.24**	**93.10 ± 0.21**	**92.55 ± 0.22**
	MLE. init.	90.76 ± 0.31	91.40 ± 0.32	91.59 ± 0.31	91.88 ± 0.30	91.61 ± 0.28
	Rand. init.	91.52 ± 0.28	92.13 ± 0.30	92.18 ± 0.31	92.04 ± 0.30	91.80 ± 0.28
	Zero. init.	91.45 ± 0.29	92.07 ± 0.30	92.09 ± 0.31	91.99 ± 0.30	91.66 ± 0.30
OOD	Ours	**85.55 ± 4.15**	**85.83 ± 4.23**	**86.42 ± 3.85**	**86.64 ± 3.61**	**86.49 ± 3.04**
	MLE. init.	81.48 ± 5.18	81.99 ± 5.17	83.54 ± 5.35	83.97 ± 5.11	83.85 ± 5.11
	Rand. init.	82.42 ± 5.29	83.59 ± 5.52	83.96 ± 5.61	83.94 ± 5.27	84.12 ± 5.01
	Zero. init.	82.08 ± 5.68	83.29 ± 5.78	83.89 ± 5.50	84.24 ± 5.22	83.83 ± 4.98

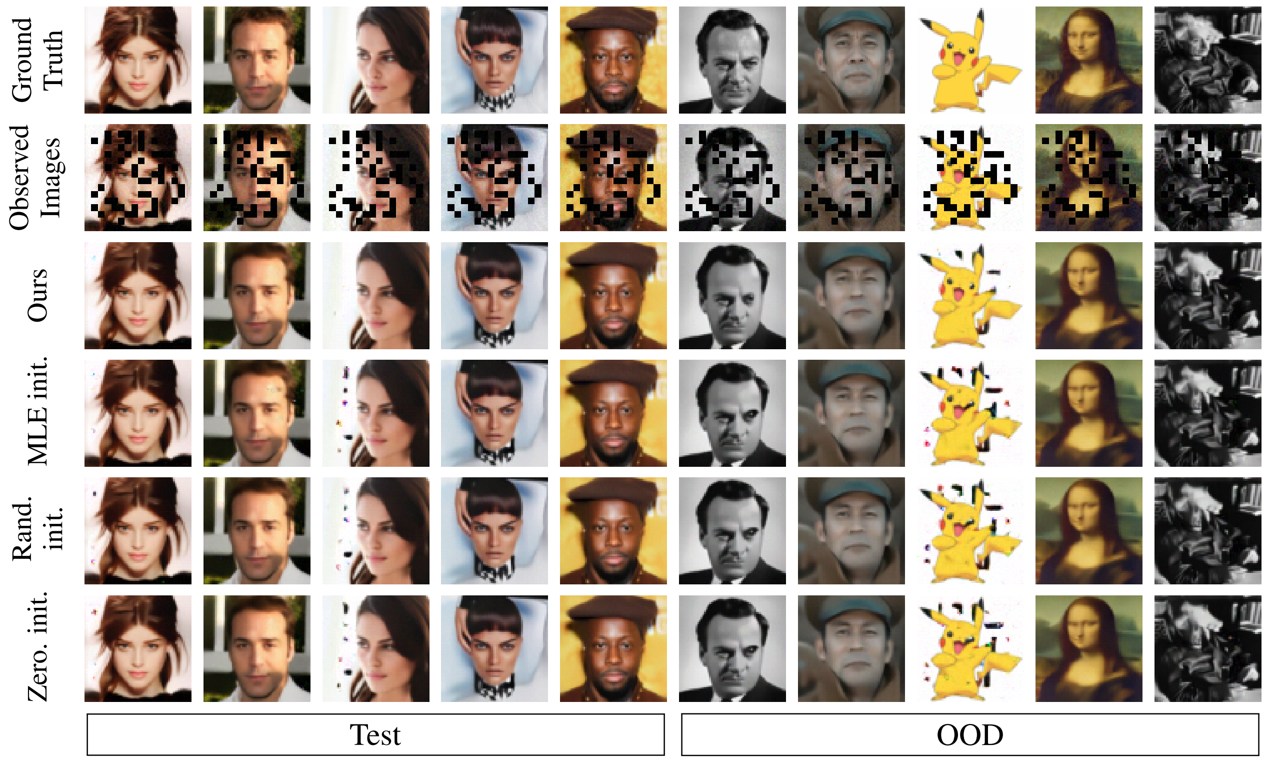

Figure 15. Results of inpainting CelebA faces (masked 25%) and out-of-distribution images with GLOW as the generative prior. Hyperparameter $\lambda = 1.5$.

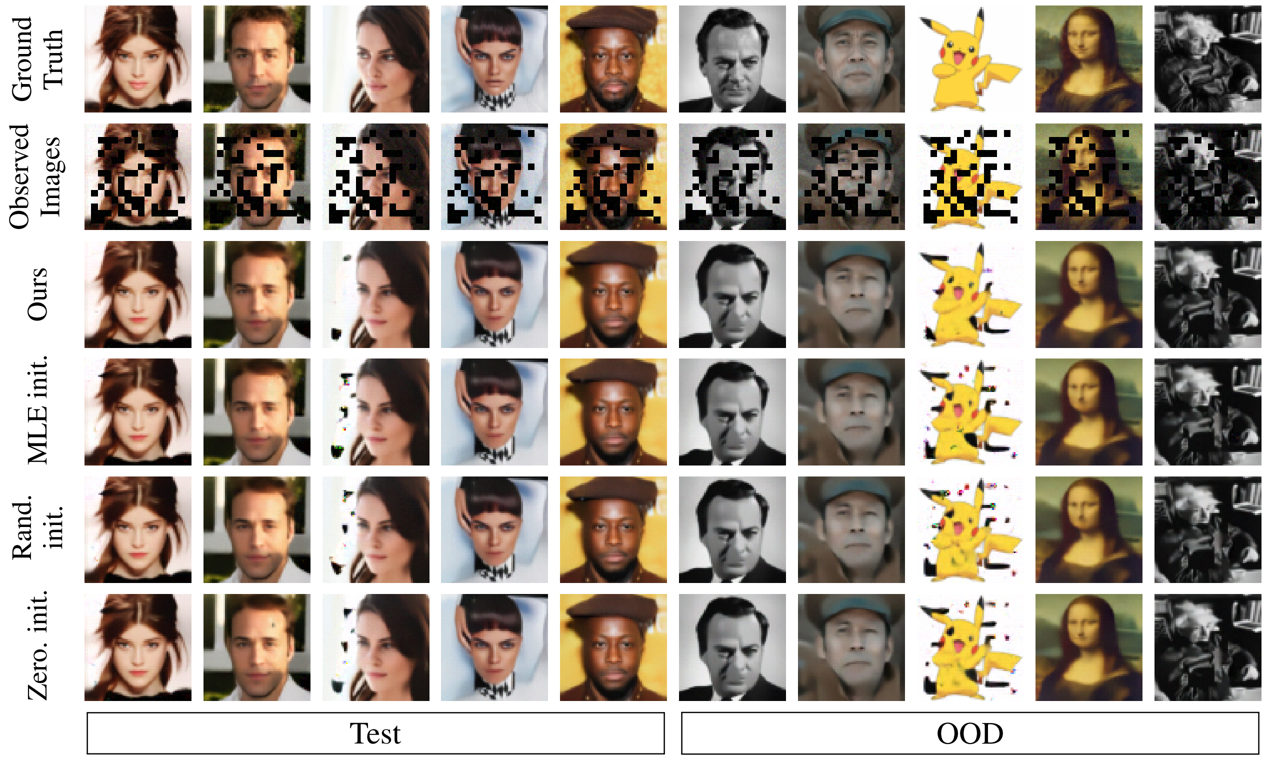

Figure 16. Results of inpainting CelebA faces (masked 30%) and out-of-distribution images with GLOW as the generative prior. Hyperparameter $\lambda = 2.0$.