COEP: Cascade Optimization for Inverse Problems with Entropy-Preserving Hyperparameter Tuning

Tianci Liu
Purdue University
liu3351@purdue.edu

Tong Yang
Peking University
tongyang@stu.pku.edu.cn

Quan Zhang
Michigan State University
quan.zhang@broad.msu.edu

Qi Lei
New York University
ql518@nyu.edu

Abstract

We propose COEP, an automated and principled framework to solve inverse problems with deep generative models. COEP consists of two components, a cascade algorithm for optimization and an entropy-preserving criterion for hyperparameter tuning. Through COEP, the two components build up an efficient and end-to-end solver for inverse problems that require no human evaluation. We establish theoretical guarantees for the proposed methods. We also empirically validate the strength of COEP on denoising and noisy compressed sensing, which are two fundamental tasks in inverse problems.

1 Introduction

Inverse problems using generative models aim to reconstruct an image/signal $x$ via a noisy or lossy observation

$$y = f(x) + e,$$

with the assumption that $x$ comes from a generative model $G$, i.e., $x = G(z)$ for some latent code $z$ (Bora et al., 2017). $f(\cdot)$ is a known forward operator and $e$ represents noise. The problem is reduced to denoising when $f(x) = x$ is the identity map, and is reduced to a compressed sensing (Candes et al., 2006; Donoho, 2006), inpainting (Vitoria et al., 2018), or super-resolution problem (Menon et al., 2020) when $f(x) = Ax$ and $A$ maps $x$ to a lower dimensional space. As an under-determined problem, inverse problems require some proper assumptions on the prior of the image. Instead of more classical assumptions like sparsity or other geometric properties on $x$ (Danielyan et al., 2011; Yu et al., 2011), deep generative prior has recently become the workhorse of inverse problems (Bora et al., 2017; Asim et al., 2020b). Existing works on solving inverse problems often design the objective by

$$\hat{z} = \arg\min_z \mathcal{L}_{\text{recon}}(G(z), y) + \lambda \mathcal{L}_{\text{reg}}(z), \quad (1)$$

where $\mathcal{L}_{\text{recon}}$ is a reconstruction error between the observation $y$ and a recovered image $G(z)$ (Bora et al., 2017; Ulyanov et al., 2018), and $\mathcal{L}_{\text{reg}}(z)$ multiplied by hyperparameter $\lambda$ regularizes $z$ via incorporating prior information of its latent distribution. The loss function is subject to different noise models (Van Veen et al., 2018; Asim et al., 2020a; Whang et al., 2021a). The image is reconstructed as $\hat{x} = G(\hat{z})$.

In execution, however, the challenges are two-fold. The success of this model fundamentally relies on a correct choice of the hyperparameter $\lambda$ and an effective optimization algorithm that can find the global minimum of (1). However, the non-convex nature of inverse problems makes optimization algorithms unprincipled and non-robust (to choices like initialization). It also requires persistent human evaluation for hyperparameter tuning due to the lack of labels, which is both time- and effort-consuming.

We thus raise the following two questions:

**Question 1.** Can we design a more principled optimization scheme that makes use of the structure of the objective and even bears theoretical guarantees?

In general, reconstructing the latent code from the observations of a deep neural network is NP-hard (Lei et al., 2019). Therefore we must utilize some special structure of the problem to resolve it efficiently.

**Question 2.** Can we design an automated hyperparameter tuning scheme requiring little or no human effort?

Specifying appropriate candidates of $\lambda$ can be difficult since they are subject to tasks, data, and the noise model, especially when the data are out of the distribution of $G$. More generally, (automated) hyperparameter tuning for un-
supervised learning problems, including inverse problems, is a long-standing open problem due to the lack of labeled data for validation. This motivates us to investigate automated hyperparameter tuning for inverse problems.

In this paper, we resolve the two questions in an integrated manner. Specifically, we have the following contributions.

- We propose a cascade optimization scheme that incrementally optimizes the solution on a sequence of $\lambda$ that gradually increase from 0 to our desired value.
- We provide theoretical guarantees on finding the global minimum of our algorithm for denoising tasks, under some natural assumptions.
- We propose an automatic entropy-preserving hyperparameter tuning scheme without any human evaluation.
- We bring together the two schemes and propose COEP, an end-to-end inverse problem solver under the maximum a posterior (MAP) framework.

Our paper proceeds as follows. We discuss the related works in Section 2, including inverse problems with deep generative priors and hyperparameter tuning in general machine learning. We propose our core algorithms in Section 3 and provide theoretical guarantees in Section 4. In Section 5, we empirically validate the performance of our algorithms compared that outperform commonly used approaches. Section 6 concludes the paper. Supplementary algorithms and proofs are deferred to the appendix.

2 Related Works

2.1 Inverse Problems with Generative Priors

Inverse problems usually require some natural signal priors to reconstruct a corrupted image (Ongie et al., 2020). Classical methods often 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, these methods are often computationally demanding. Recently, pretrained deep generative models such as generative adversarial network (GAN) and its variants (Goodfellow et al., 2014; Karras et al., 2017, 2019) have succeeded in inverse problems (Bora et al., 2017; Hand and Voroninski, 2018; Hand et al., 2018; Asim et al., 2020b; Jalal et al., 2020, 2021a). Compared to classical methods, Using a GAN prior is able to produce better reconstructions at much less measurements (Bora et al., 2017).

Despite the success, Asim et al. (2020a) points out that a GAN prior is prone to representation error 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 to replace the GAN prior with normalizing flows (NFs). 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 and 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 and Denli, 2021).

Untrained deep neural networks are also used to solve inverse problems (Ulyanov et al., 2018; Heckel and Hand, 2018; Hussein et al., 2020). By using a network’s architectures as an image prior, these untrained models can be competitive with GAN priors. Nevertheless, for these methods, solid theoretical foundations have not been sufficiently established.

2.2 Automated Hyperparameter Tuning

Automated hyperparameter tuning is essential to improve the efficiency and performance for general machine learning problems (Snoek et al., 2012; Hazan et al., 2017; Elshawi et al., 2019; Yu and Zhu, 2020). Towards this goal, a variety of strategies on hyperparameter tuning have been proposed, including grid search (Bergstra et al., 2011), random search (Bergstra and Bengio, 2012), evolutionary optimization (Maron and Moore, 1993; Miikkulainen et al., 2019), population-based training (Jaderberg et al., 2017), and other model-based tuning scheme such as Bayesian optimization (Snoek et al., 2012; Frazier, 2018; Balandat et al., 2020). However, most of these methods require ground truth for validation to automate the hyperparameter tuning. Due to the lack of such ground truth, hyperparameter optimization for unsupervised learning problems is difficult and requires some surrogate for evaluation in general (Shalamov et al., 2018; Fan et al., 2020). Existing literature is largely silent on automated hyperparameter tuning for inverse problems that lack validation data.

3 Methodology

We first propose a cascade optimization algorithm to efficiently reconstruct images in inverse problems by simultaneously optimizing $x$ and increasing $\lambda$ from 0 to a prespecified value. Then we develop an automated hyperparameter tuning scheme by matching the entropies of the estimated and true noise distributions. It admits binary search for an optimal $\lambda$, which is more efficient than grid search. Finally, we propose a cascade optimization with entropy-preserving hyperparameter tuning (COEP) by combining the two algorithms.
3.1 Preliminaries

We use plain lowercase letters (e.g., e) to denote scalars, bold uppercase letters (e.g., A) and lowercase letters (e.g., x) to denote matrices and vectors, respectively. We use a pre-trained invertible generative model \( G : \mathbb{R}^n \rightarrow \mathbb{R}^n \) that we can effectively sample from. \( G \) maps latent variable \( z \in \mathbb{R}^n \) to an image \( x \in \mathbb{R}^n \) and induces a tractable distribution over \( x \) by the change-of-variable formula (Papamakarios et al., 2021). We denote the distribution by \( p_G(x) \) and use it as the prior distribution of \( x \).

Given a clean image \( x \), we observe a corrupted measurement \( y = f(x) + e \), where \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \) is some known forward operator such that \( m \leq n \). The additive term \( e \in \mathbb{R}^m \) denotes a noise that is assumed to have independent and identically distributed entries (Bora et al., 2017; Asim et al., 2020a). We aim to reconstruct \( x \) from \( y \).

Given a hyperparameter \( \lambda > 0 \) as the weight of the prior, we consider a tempered maximum a posteriori (MAP) estimation (Whang et al., 2021a) as the reconstruction \( \hat{x}(\lambda) \). That is,

\[
\hat{x}(\lambda) = \arg\min_x \mathcal{L}_{\text{MAP}}(x; \lambda) \quad \text{(2)}
\]

\[
= \arg\min_x -\log p_e(y - f(x)) - \lambda \log p_G(x),
\]

where \( p_e \) is the density of \( e \). \( p_e \) is often assumed to be known as one can approximate it by an isotropic Gaussian distribution. Consequently, the loss function becomes

\[
\mathcal{L}_{\text{MAP}}(x; \lambda) = \|y - f(x)\|^2 - \lambda \log p_G(x). \quad \text{(3)}
\]

Optimization with respect to \( z \) is also considered in the literature (e.g., Asim et al. (2020a)), which is necessary when \( p_G(x) \) does not have an explicit form (Bora et al., 2017).

In our context, working on \( x \) or on \( z \) makes no difference due to the invertibility of \( G \). Henceforth, we present our methodology in terms of \( x \) in the remainder of this paper and additionally elaborate on our algorithms in terms of \( z \) in the appendix.

3.2 Cascade Optimization

The neural networks in \( G \) make \( \mathcal{L}_{\text{MAP}}(x; \lambda) \) in equation (3) non-convex and NP-hard (Lei et al., 2019) to minimize. When \( \lambda = 0 \), the solution \( \hat{x}(\lambda) \) is reduced to the MLE, i.e., \( \hat{x}(0) = f^{-1}(y) \) if computing \( f^{-1}(y) \) is feasible, which is often used to initialize \( x \) in inverse problems with a positive \( \lambda \) (Asim et al., 2020a; Whang et al., 2021a; Bora et al., 2017). However, the MLE initialization is not necessarily better than random initialization (as shown in Table 1, shortly) if we minimize \( \mathcal{L}_{\text{MAP}}(x; \lambda) \) with \( \lambda \) fixed at a desired value during optimization, possibly because a solution \( \hat{x}(\lambda) \) is far from \( \hat{x}(0) \) when \( \lambda \) is reasonably greater than 0. This inspires us to propose a cascade optimization as shown in Algorithm 1.

Suppose our desired hyperparameter value is \( \Lambda \). Our cascade optimization in Algorithm 1 finds an \( \varepsilon \)-approximation of the MAP estimation \( \hat{x}(\Lambda) \) by gradually increasing \( \lambda \) from 0 to \( \Lambda \) while optimizing \( \mathcal{L}_{\text{MAP}}(x; \lambda) \). Algorithm 1 bears a theoretical guarantee under some mild conditions that define \( L, \sigma, \delta, \) and \( C \). Intuitively, if \( G \) is smooth and \( \lambda + \frac{\delta}{\varepsilon} \approx \lambda \), \( \hat{x}(\lambda) = \arg\min_x \mathcal{L}_{\text{MAP}}(x; \lambda) \) should be close to \( \hat{x}(\lambda + \frac{\delta}{\varepsilon}) = \arg\min_x \mathcal{L}_{\text{MAP}}(x; \lambda + \frac{\delta}{\varepsilon}) \), so that we can find \( \hat{x}(\lambda + \frac{\delta}{\varepsilon}) \) from \( \hat{x}(\lambda) \) with less effort when \( \lambda \) is slightly increased to \( \lambda + \frac{\delta}{\varepsilon} \).

In Algorithm 1, we initialize \( x \) by the MLE \( f^{-1}(y) \). In an under-determined inverse problem like compressed sensing where \( f^{-1}(y) \) is not unique, we choose a candidate \( f^{-1}(y) \) that maximizes \( \log p_G(x) \). In addition, appropriate values of \( L, \sigma, \delta, \) and \( C \) need to be specified (see Section 4 for details), which determine how fast \( \lambda \) is increased and \( x \) is moved. We provide an alternative guideline on gradually changing \( \lambda \) and \( x \), which is shown in Algorithm 4 in the appendix, to circumvent the prespecification of \( L, \sigma, \delta, \) and \( C \).

Finally, we want to highlight that the invertibility of \( G \) allows one to solve this cascade optimization with respect to latent variable \( z \), and we show a corresponding algorithm that updates \( z \) in Appendix C. The theoretical guarantee of Algorithm 1 is shown in Section 4.

3.3 Entropy Preserving for Hyperparameter Tuning

The reconstruction \( \hat{x}(\lambda) \) is up to a proper value of \( \lambda \) that determines the relative importance of the prior \( p_G \) in \( \mathcal{L}_{\text{MAP}} \). Grid search can be used to tune this hyperparameter. In inverse problems, however, the range of an optimal \( \lambda \) can be

\begin{algorithm}[h]
\caption{Cascade optimization for \( x \)}
1: \textbf{Input:} \( \Lambda > 0 \), generative model \( G : \mathbb{R}^n \rightarrow \mathbb{R}^n, L > 0 \) (see Assumption 1), \( \sigma > 0 \), \( \delta > 0 \) (see Assumption 2), \( C > 0 \) (see Assumption 3), precision \( \varepsilon > 0 \)
2: \textbf{Initialize:} \( x_0 = f^{-1}(y), \lambda = 0, \mu = \frac{1}{2L\sigma^2}, \delta_0 = \left\lfloor \frac{2LC}{\delta_0} \right\rfloor + 1 \)
3: \textbf{for} \( i = 0, \ldots, N - 1 \) \textbf{do}
4: \( \lambda = \lambda + \frac{\delta}{\varepsilon} \)
5: \( K = \left\lceil \frac{2\log(2\delta/\varepsilon)}{\log(L/(L-\mu))} \right\rceil + 1 \)
6: \textbf{if} \( i == N - 1 \) \textbf{then}
7: \( K = \max(0, \left\lceil \frac{2\log(2\delta/\varepsilon)}{\log(L/(L-\mu))} \right\rceil + 1) \)
8: \textbf{end if}
9: \textbf{for} \( k = 1, \ldots, K \) \textbf{do}
10: \( x_{k+1} = x_k - \frac{1}{L} \nabla_x \mathcal{L}_{\text{MAP}}(x_k, \lambda) \)
11: \textbf{end for}
12: \( x_0 = x_{K+1} \)
13: \textbf{end for}
14: \textbf{return} \( \hat{x}(\Lambda) = x_0 \)
\end{algorithm}
so large that brute-force searching is prohibitive. For example, the best λ found varies between 1 and 100 in the experiments of Whang et al. (2021a). So, we propose an efficient entropy-preserving hyperparameter tuning algorithm from the Bayesian perspective. This algorithm is general and can be used with any optimization approach that solves (2).

Dividing the right-hand side of (3) by λ, λ is interpreted as the Gaussian variance of the noise e. If λ = 0, the MAP solution is reduced to the MLE, i.e., \( \hat{x}(0) = f^{-1}(y) \), with the estimated noise degenerating to 0. As λ increases to \( \infty \), \( \hat{x}(\lambda) \) deviates from the MLE to a mode of the prior \( G \), completely overlooking the information of \( y \) and the noise distribution \( p_e \). Therefore, we search for an optimal λ by the following bilevel optimization such that the estimated noise \( y - f(\hat{x}(\lambda)) \) looks like the real noise in \( y \):

\[
\max_{\lambda} \log q(\xi(\lambda)) \tag{4}
\]

\[
s.t. \quad \xi(\lambda) = \frac{1}{m} \log p_e(y - f(\hat{x}(\lambda))), \tag{5}
\]

where \( q = \mathcal{N}(\mathbb{E}[\log p_e(e_i)], \text{var} [\log p_e(e_i)] / m) \) is the asymptotic Gaussian distribution for \( \frac{1}{m} \sum_{i=1}^{m} \log p_e(e_i) \) by the central limit theorem, and \( e_i \) is the i-th element of the true noise. Remarkably, \( q \) does not rely on \( \lambda \) and its mean and variance are in closed forms for some common distribution \( p_e \). In particular, when \( e \) follows an isotropic Gaussian distribution that is \( p_e(e_i) = \mathcal{N}(e_i|0, s^2) \), \( q = \mathcal{N}(-0.5 - 0.5 \log(2\pi s^2), 0.5/m) \).

An optimal \( \lambda \) that solves (4) is achieved when \( \xi(\lambda) = \mathbb{E}[\log p_e(e_i)] \). In other words, we find a \( \lambda \) such that the estimated noise \( y - f(\hat{x}(\lambda)) \) recovers the true noise \( \lambda \) in terms of entropy. We refer to this property as entropy preserving. Importantly, the entropy-preserving property facilitates an efficient binary search for an optimal \( \lambda \), which is theoretically underpinned by Lemma 1.

**Lemma 1.** \( \xi(\lambda) = \frac{1}{m} \log p_e(y - f(\hat{x}(\lambda))) \), the negative empirical entropy of the noise, is non-increasing in \( \lambda \).

Intuitively, when \( \lambda = 0 \), \( \hat{x}(\lambda) = \arg\min_{x} - \log p_e(y - f(x)) \). When \( \lambda \) becomes larger, \( \hat{x}(\lambda) \) is supposed to get closer to \( \arg\min_{x} - \log p_G(x) \) and move away from \( \arg\min_{x} - \log p_e(y - f(x)) \), thus \( - \log p_e(y - f(\hat{x}(\lambda))) \) is supposed to be non-decreasing in \( \lambda \).

Together with the unimodal density of \( q \), Lemma 1 implies that \( q(\xi(\lambda)) \) is unimodal in \( \lambda \). This admits an efficient binary search scheme for \( \lambda \) by comparing \( \xi(\lambda) \) and \( \mathbb{E}[\log p_e(e_i)] \) and preserving the entropy of the noise distribution, as elaborated in Algorithm 2. Concretely, given a search interval \([\lambda_{\text{low}}, \lambda_{\text{high}}]\) and the midpoint \( \lambda_{\text{mid}} = \frac{1}{2} (\lambda_{\text{high}} + \lambda_{\text{low}}) \), we find a solution \( \hat{x}(\lambda_{\text{mid}}) \) to equation (2), calculate the empirical entropy of the estimated noise \( y - f(\hat{x}(\lambda_{\text{mid}})) \), and compare it with the true entropy. If the empirical entropy is larger than the true entropy, we keep searching the lower half of the interval, and otherwise, the upper half. Note that Algorithm 2 is a general framework and can incorporate any optimization algorithm, including our cascade optimization, to find \( \hat{x}(\lambda_{\text{mid}}) \) as in line 5.

### 3.4 COEP as an End-to-end Solver

To this end, we formally propose COEP, Cascade Optimization with Entropy-Preserving hyperparameter tuning, in Algorithm 3 to simultaneously tune \( \lambda \) and reconstruct \( x \) in inverse problems. Concretely, we increase \( \lambda \) from 0 and update \( x \) for \( t \) times by the cascade optimization until the entropy of the estimated noise is greater than the true entropy, i.e., \( \xi(\lambda_t) < \xi^* \) and \( \lambda_t < \Lambda \). Then we optimize \( x \) and search for the optimal \( \lambda \) in \([\lambda_{t-1}, \lambda_t]\) by Algorithm 2.

### 4 Theoretical Analysis

We theoretically underpin our cascade optimization for inverse problems with \( f \) being the identity map, as in denoising tasks. We make the following assumptions, under which our cascade optimization as in Algorithm 1 finds a good approximate of the global minimum of \( \mathcal{L}_{\text{MAP}} \).

---

**Algorithm 2** Entropy-preserving binary search for \( \lambda \)

1. **Input:** generative model \( G: \mathbb{R}^n \rightarrow \mathbb{R}^n \), search space \([\lambda_{\text{low}}, \lambda_{\text{high}}]\), precision \( \varepsilon > 0 \).
2. **Initialize:** \( \xi^* = \mathbb{E}[\log p_e(e_i)] \).
3. **repeat**
   4. \( \lambda_{\text{mid}} = \frac{1}{2} (\lambda_{\text{high}} + \lambda_{\text{low}}) \)
   5. \( \hat{x}(\lambda_{\text{mid}}) = \arg\min_{x} \mathcal{L}_{\text{MAP}}(x; \lambda_{\text{mid}}) \)
   6. \( \xi(\lambda_{\text{mid}}) = \frac{1}{m} \log p_e(y - f(\hat{x}(\lambda_{\text{mid}}))) \)
   7. **if** \( \xi(\lambda_{\text{mid}}) > \xi^* \) **then**
   8. \( \lambda_{\text{low}} = \lambda_{\text{mid}} \)
   9. **else** if \( \xi(\lambda_{\text{mid}}) < \xi^* \) **then**
   10. \( \lambda_{\text{high}} = \lambda_{\text{mid}} \)
   11. **else**
   12. break
   13. **end if**
   14. **until** \( \lambda_{\text{high}} - \lambda_{\text{low}} < \varepsilon \)
15. **return** \( \lambda_{\text{mid}}, \hat{x}(\lambda_{\text{mid}}) \)

**Algorithm 3** COEP

1. **Initialize:** \( t = 0, \lambda_t = 0, x(\lambda_t) = f^{-1}(y), \xi^* = \mathbb{E}[\log p_e(e_i)] \)
2. **repeat**
3. \( t = t + 1 \)
4. **Update** \( \lambda_t \) and \( x(\lambda_t) \) by the cascade optimization (lines 4 to 12 of Algorithm 1 or lines 4 to 10 of Algorithm 4)
5. \( \xi(\lambda_t) = \frac{1}{m} \log p_e(y - f(\hat{x}(\lambda_t))) \)
6. **until** \( \xi(\lambda_t) < \xi^* \)
7. **Find** \( \lambda \) in \([\lambda_{t-1}, \lambda_t]\) and \( \hat{x}(\lambda) \) by Algorithm 2.
Assumption 1 (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 $x_1$ and $x_2$, $\|\nabla_x \mathcal{L}_{\text{MAP}}(x_1; \lambda) - \nabla_x \mathcal{L}_{\text{MAP}}(x_2; \lambda)\| \leq L \|x_1 - x_2\|$.

Assumption 2 (local property of $\nabla_x \mathcal{L}_{\text{MAP}}$). There exists $\sigma > 0$ and $\delta > 0$ such that for all $\lambda \in [0, \Lambda]$ and $x \in B(\hat{x}(\lambda), \delta) := \{x \mid \|x - \hat{x}(\lambda)\| \leq \delta\}$, we have $\|x - \hat{x}(\lambda)\| \leq \sigma \|\nabla_x \mathcal{L}_{\text{MAP}}(x; \lambda)\|$.

Assumption 3 ($C$-smoothness of $\hat{x}(\lambda)$). For all $\lambda \in [0, \Lambda]$, $\hat{x}(\lambda)$ is unique, and there exists $C > 0$ such that for all $\lambda_1, \lambda_2 \in [0, \Lambda]$, $\|\hat{x}(\lambda_1) - \hat{x}(\lambda_2)\| \leq C|\lambda_1 - \lambda_2|$.

Remark 1. Smoothness assumptions like Assumptions 1 and 3 are common in analysis of convergence. We make two comments on Assumption 2:

(i) Assumption 2 is weaker than locally strongly-convexity. Reversely, if there exists $\mu' > 0$ and $\delta > 0$ such that for all $\lambda \in [0, \Lambda]$, $\mathcal{L}_{\text{MAP}}(\cdot; \lambda)$ is $\mu'$-strongly convex on $B(\hat{x}(\lambda), \delta)$, then Assumption 2 holds with $\sigma = 2/\mu'$.

(ii) If Assumption 1 holds, then Assumption 2 implies the local Polyak-Lojasiewicz condition. That is to say, if Assumptions 1 and 2 hold, then for all $\lambda \in [0, \Lambda]$ and $x \in B(\hat{x}(\lambda), \delta)$, we have

$$\mathcal{L}_{\text{MAP}}(x; \lambda) - \mathcal{L}_{\text{MAP}}(\hat{x}(\lambda); \lambda) \leq \frac{1}{2\mu} \|\nabla_x \mathcal{L}_{\text{MAP}}(x; \lambda)\|^2,$$

where $\mu = \frac{1}{2\sigma^2}$.

Local Lojasiewicz property was previously used to characterize the local optimization landscape for training neural networks Zhou et al. (2021). Now we are ready to state our main result:

Theorem 1. Under Assumptions 1, 2, and 3, for all $\varepsilon > 0$, Algorithm 1 returns $\hat{x}$ that satisfies $\|\hat{x} - \hat{x}(\lambda)\| \leq \varepsilon$.

Note that Theorem 1 ensures the cascade optimization—Alg.1 finds an $\varepsilon$-approximate point of the global minimum of $\mathcal{L}_{\text{MAP}}(\cdot; \Lambda)$.

We give a proof sketch of Theorem 1 here and defer the formal proof to the appendix. On a high level, starting from the global min when $\lambda = 0$ (which is $f^{-1}(y)$ if it is feasible), our algorithm ensures the $i$-th outer iteration learns $\hat{x}(i\lambda/N)$ approximately, and will become a good initialization point for our next target $\hat{x}((i+1)\lambda/N)$. (Note that in each iteration we incrementally increase $\lambda$ from $i\lambda/N$ to $(i+1)\lambda/N$ until reaching our target $\Lambda$.) Specifically, our Assumption 3 and Assumption 2 respectively ensures $\hat{x}(i\lambda/N)$ to be close enough to $\hat{x}((i+1)\lambda/N)$ and that first order algorithm can find $\hat{x}((i+1)\lambda/N)$ from the good initialization we obtained in the last iteration.

5 Experiments

In this section, we evaluate the performance of the proposed algorithm on two tasks, denoising and noisy compressed sensing (NCS). First, we show the outperformance of our cascade optimization (CO) over a standard gradient-based optimization if we have a desired value of $\lambda$. Then we compare the performance of COEP and grid search in solving the inverse problems where $\lambda$ needs to be tuned.

5.1 Setup

We use RealNVP (Dinh et al., 2016) as our generative prior $G$, which is a normalizing flow model pretrained by Whang et al. (2021a) on the CelebA-HQ data set. We run optimization and hyperparameter-tuning algorithms on two sets of data: in-distribution samples that are randomly selected from the CelebA-HQ test set and out-of-distribution (OOD) images that contain human or human-like objects as shown in Figure 1. Due to budget constraints, we run the experiments on 200 in-distribution and 7 OOD samples. We optimize $x$ by Adam (Kingma and Ba, 2014) with a learning rate 0.01 throughout this section and assign the same computing budget to the algorithms compared in each subsection.

In denoising tasks, we assume $y = x + e$ and the noise $e \sim N(0, 0.011)$ (Asim et al., 2020a; Bora et al., 2017). In NCS tasks, we assume $f(x) = Ax + e$ where $A \in \mathbb{R}^{m \times n}$, $m < n = 12288$, is a known $m \times n$ matrix of i.i.d.
| Ground Truth | Noisy images | COEP | GOEP | 5 grids | 10 grids | 15 grids | 20 grids | Test set examples | Out-of-distribution examples |
|--------------|--------------|------|------|---------|----------|----------|----------|------------------|-----------------------------|

Figure 1: Results of denoising Gaussian noise on CelebA-HQ faces and out-of-distribution images.

| Ground Truth | COEP | GOEP | 5 grids | 10 grids | 15 grids | 20 grids | Test set examples | Out-of-distribution examples |
|--------------|------|------|---------|----------|----------|----------|------------------|-----------------------------|

Figure 2: Reconstructions of NCS at 5000 measurements, $\lambda \in [0, 5]$. 
### Table 2: PSNR and corresponding hyperparameter $\lambda$ (mean ± se) on Denoising and NCS tasks. We tune $\lambda$ over $[0, 2]$ for Denoising task and over $[0, 5]$ for NCS tasks. Numbers in front of NCS mark measurements. Best results and corresponding hyperparameters are in bold.

| Task | COEP | GOEP | 5 Grids | 10 Grids | 15 Grids | 20 Grids |
|------|------|------|---------|----------|----------|----------|
|      |      |      | Peak-Signal-to-Noise Ratio (PSNR) |
| Test | Denoise | 29.13 ± 0.06 | 29.05 ± 0.06 | 29.25 ± 0.06 | 30.47 ± 0.06 | **30.72 ± 0.06** | 30.52 ± 0.05 |
|      | 2000 NCS | **28.12 ± 0.07** | 28.00 ± 0.07 | 25.08 ± 0.09 | 25.28 ± 0.17 | 22.91 ± 0.20 | 21.25 ± 0.21 |
|      | 3000 NCS | 28.31 ± 0.07 | 28.18 ± 0.07 | 25.25 ± 0.09 | 27.37 ± 0.12 | 25.49 ± 0.17 | 25.16 ± 0.24 |
|      | 4000 NCS | **28.18 ± 0.09** | 28.02 ± 0.09 | 25.24 ± 0.09 | 28.07 ± 0.10 | 26.98 ± 0.15 | 25.18 ± 0.19 |
|      | 5000 NCS | 28.03 ± 0.10 | 27.85 ± 0.09 | 25.28 ± 0.08 | **28.71 ± 0.08** | 27.73 ± 0.12 | 27.06 ± 0.19 |
| OOD  | Denoise | 28.80 ± 0.54 | 28.67 ± 0.51 | 28.10 ± 0.58 | 29.54 ± 0.49 | 29.79 ± 0.58 | 29.39 ± 0.53 |
|      | 2000 NCS | **27.20 ± 0.70** | 27.03 ± 0.69 | 23.82 ± 0.46 | 22.94 ± 1.33 | 20.83 ± 1.16 | 19.36 ± 1.34 |
|      | 3000 NCS | 27.83 ± 0.71 | 26.20 ± 1.14 | 24.22 ± 0.62 | 25.34 ± 1.05 | 23.39 ± 1.39 | 20.97 ± 1.45 |
|      | 4000 NCS | 27.69 ± 0.69 | 27.64 ± 0.71 | 24.00 ± 0.65 | 26.80 ± 0.77 | 24.27 ± 1.19 | 23.17 ± 1.36 |
|      | 5000 NCS | **27.78 ± 0.76** | 27.54 ± 0.77 | 24.31 ± 0.63 | 27.27 ± 0.72 | 25.39 ± 1.13 | 23.56 ± 1.30 |

### Best hyperparameter $\lambda$

| Task | COEP | GOEP | 5 Grids | 10 Grids | 15 Grids | 20 Grids |
|------|------|------|---------|----------|----------|----------|
|      |      |      |        |          |          |          |
| Test | Denoise | 0.64 ± 0.01 | 0.64 ± 0.01 | 0.50 ± 0.00 | 0.24 ± 0.00 | **0.29 ± 0.00** | 0.29 ± 0.00 |
|      | 2000 NCS | **0.66 ± 0.01** | 0.66 ± 0.00 | 1.26 ± 0.01 | 0.94 ± 0.03 | 1.07 ± 0.03 | 1.07 ± 0.03 |
|      | 3000 NCS | 0.65 ± 0.01 | 0.65 ± 0.00 | 1.25 ± 0.00 | 0.71 ± 0.02 | 0.82 ± 0.02 | 0.72 ± 0.02 |
|      | 4000 NCS | **0.63 ± 0.01** | 0.64 ± 0.01 | 1.25 ± 0.00 | 0.67 ± 0.02 | 0.67 ± 0.02 | 0.72 ± 0.01 |
|      | 5000 NCS | 0.61 ± 0.01 | 0.61 ± 0.01 | 1.25 ± 0.00 | **0.58 ± 0.01** | 0.60 ± 0.02 | 0.58 ± 0.01 |
| OOD  | Denoise | 0.67 ± 0.03 | 0.62 ± 0.03 | 0.50 ± 0.00 | 0.25 ± 0.03 | **0.29 ± 0.00** | 0.30 ± 0.01 |
|      | 2000 NCS | **0.68 ± 0.05** | 0.66 ± 0.03 | 1.25 ± 0.00 | 1.03 ± 0.21 | 1.17 ± 0.17 | 1.28 ± 0.17 |
|      | 3000 NCS | 0.66 ± 0.03 | 0.63 ± 0.02 | 1.25 ± 0.00 | 0.71 ± 0.15 | 0.77 ± 0.13 | 0.86 ± 0.12 |
|      | 4000 NCS | **0.64 ± 0.03** | 0.65 ± 0.03 | 1.25 ± 0.00 | 0.71 ± 0.09 | 0.82 ± 0.09 | 0.71 ± 0.07 |
|      | 5000 NCS | **0.63 ± 0.02** | 0.62 ± 0.02 | 1.25 ± 0.00 | 0.63 ± 0.07 | 0.66 ± 0.09 | 0.68 ± 0.07 |

Figure 3: Reconstructions of NCS at 3000 measurements, $\lambda \in [0, 5]$. 

---

Ground Truth

COEP

GOEP

5 grids

10 grids

15 grids

20 grids

Test set examples

Out-of-distribution examples
\(N(0, 1/m)\) entries (Asim et al., 2020a). The smaller \(m\) is, the more difficult the NCS task will be. We vary \(m\) between 2000 and 5000. Literature (Asim et al., 2020a; Bora et al., 2017) considers Gaussian noise \(e\) with \(E[e] = 0\) and \(E[|e|] = 0.1\). In this setting, however, the scale of the noise is not big enough to fail the algorithms in our experiments, and their performances are comparable. Therefore, we set more challenging NCS tasks by imposing larger noise \(e \sim N(0, 0.01I)\). We quantify an algorithm performance by Peak-Signal-to-Noise Ratio (PSNR), whose large value indicates a better reconstruction. In the tables in this section, the best results are in bold.

5.2 Cascade Optimization with a Prespecified \(\lambda\)

We first show that if we have a desired hyperparameter value \(\Lambda\), our cascade optimization (CO) outperforms a standard gradient-based optimization (GO) with \(\lambda\) always fixed at \(\Lambda\). Concretely, with \(\Lambda = 0.6\) (as is found by the experiments in Section 5.3, shortly), we run CO that lets \(\lambda\) gradually approach \(\Lambda\) from 0 while optimizing \(x\) and compare with GO that starts from either a randomly initialized \(x\) (Bora et al., 2017; Asim et al., 2020a; Whang et al., 2021) or the MLE \(f^{-1}(y)\), respectively. We also set \(\Lambda = 1\) to validate CO in the case of a misspecified hyperparameter.

Table 1 reports the PSNRs of the reconstructed images in denoising and NCS with \(m = 2000, 3000, 4000, 5000\). As a result, while CO slightly underperforms GO with a random initial \(x\) in the tasks of NCS with \(m = 4000\) and 5000, it generally outperforms the two benchmarks, especially in more difficult NCS tasks with a smaller \(m\). One exception is in the task of NCS with \(m = 2000\), but the deficit is negligible considering the large standard errors. It is demonstrated that under the same computing budget, gradually increasing \(\lambda\) by CO delivers better results than fixing \(\lambda\) at a prespecified value during optimization.

5.3 Hyperparameter Tuning and COEP

We compare the performance of different hyperparameter tuning approaches for the denoising and NCS tasks, including grid search, COEP, and GO integrated into the entropy-preserving binary search (as indicated in line 5 of Algorithm 2), which we refer to as GOEP. We search for a best \(\lambda\) in \([0, 2]\) for denoising and in \([0, 5]\) for NCS. We try 5, 10, 15, and 20 grids respectively for grid search, and conduct an ad-hoc evaluation by choosing the reconstruction that achieves the highest PSNR to the ground truth. This resembles a manual-tuning regime where the best result is selected by a human expert after eyeballing all reconstructions, which is required in practice due to the lack of validation data. Both the grid search and GOEP start from a randomly initialized \(x\).

We show reconstructed images in denoising and NCS with \(m = 5000\) and \(m = 3000\) measurements in Figures 1, 2, and 3, respectively. In Figure 1, the denoised images by COEP not only contain the least unnatural textures compared to grid search (e.g., the white dots in columns 5 and 7), but also keep roughly the same amount of or even more detail (e.g., the less blurred nose and mouth in column 6). In figure 2 for NCS with \(m = 5000\), COEP looks slightly better than GOEP and is much better than grid search. The outperformance of COEP is more evident in the NCS tasks with a smaller \(m\) where grid search largely fails. See Figures 3 and Figures 4 and 5 in the appendix for detail.

We report the PSNRs of the reconstructed images together with the best \(\lambda\) selected by COEP, GOEP, and grid search in Table 2. We see COEP outperforms GOEP and grid search in almost all the NCS tasks with the only exception of the test samples when \(m = 5000\). It is noteworthy that the dominance of COEP in the NCS tasks is more remarkable on OOD samples. While COEP delivers slightly lower PSNRs than grid search on the denoising task, we do not see a noticeable deficit from the reconstructed images in Figure 1. In addition, GOEP underperforms COEP that benefits from CO, but it outperforms grid search in most tasks, showing the generality of our entropy-preserving hyperparameter tuning that can be integrated with standard gradient-based optimization.

Our experiment results reveal practical insights for grid search and another advantage of COEP. By Table 2, grid search performs worse on the NCS tasks when the number of grids is over 10 because each grid does not get enough computing budget. So, if there is a budget constraint for grid search, it is important to determine how many grids to explore. Equivalently, we have to know how much budget the optimization on each grid needs for convergence. This can be difficult to estimate since the demand may vary with different \(\lambda\) and initial values of \(x\). In stark contrast, COEP is automated and benefits from both the cascade optimization and the binary search. Specifically, the increase of \(\lambda\) is small in each update, and thus the optimizations of \(x\) requires much fewer iterations and makes COEP more efficient.

6 Conclusion

In this paper we propose COEP, an end-to-end solution for inverse problems, to tackle two fundamental challenges: non-convex optimization and automated hyperparameter tuning. The power of COEP is demonstrated by both theoretical guarantees and experiments. Empirically, COEP is especially suitable for challenging scenarios such as low-recovery and reconstructing at few measurements, where standard optimization and hyperparameter tuning by grid search may fail.

For future work, it would be interesting to extend COEP to more challenging tasks such as non-linear forward operator
or non-additive noise involves.

**Societal impact.** COEP depends on using a proper generative prior $G$. If this generative model is trained from biased data, it might impair the later on performance (Jalal et al., 2021b). However, our work is focusing on the optimization of the loss function, thus the data selection bias is out of scope of the discussion of this paper, even though it might cause potential unfairness.

**References**

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

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

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. (2020). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. *Advances in neural information processing systems*, 33:21524–21538.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for Hyper-parameter Optimization. *Advances in neural information processing systems*, 24.

Bergstra, J. and Bengio, Y. (2012). Random Search for Hyper-parameter Optimization. *Journal of machine learning research*, 13(2).

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

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

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

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

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

Elshawi, R., Maher, M., and Sakr, S. (2019). Automated Machine Learning: State-of-The-Art and Open Challenges. *arXiv preprint arXiv:1906.02287*.

Fan, X., Yue, Y., Sarkar, P., and Wang, Y. R. (2020). On hyperparameter tuning in general clustering problemsm. In *International Conference on Machine Learning*, pages 2996–3007. PMLR.

Frazier, P. I. (2018). A Tutorial on Bayesian Optimization. *arXiv preprint arXiv:1807.02811*.

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

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

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

Hazan, E., Klivans, A., and Yuan, Y. (2017). Hyperparameter Optimization: A Spectral Approach. *arXiv preprint arXiv:1706.00764*.

Heckel, R. and Hand, P. (2018). Deep Decoder: Concise Image Representations from Untrained Non-convolutional Networks. *arXiv preprint arXiv:1810.03982*.

Hussein, S. A., Tirer, T., and Giryes, R. (2020). Image-Adaptive GAN Based Reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3121–3129.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. (2017). Population Based Training of Neural Networks. *arXiv preprint arXiv:1711.09846*.

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

Jalal, A., Karmalkar, S., Hoffmann, J., Dimakis, A., and Price, E. (2021b). Fairness for Image Generation with Uncertain Sensitive Attributes. In *International Conference on Machine Learning*, pages 4721–4732. PMLR.

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

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

Karras, T., Laine, S., and Aila, T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE/CVF conference on
Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

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

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

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

Maron, O. and Moore, A. (1993). Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation. Advances in neural information processing systems, 6.

Menon, S., Damian, A., Hu, S., Ravi, N., and Rudin, C. (2020). 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, pages 2437–2445.

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al. (2019). Evolving Deep Neural Networks. In Artificial intelligence in the age of neural networks and brain computing, pages 293–312. Elsevier.

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

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference. Journal of Machine Learning Research, 22(57):1–64.

Shalamov, V., Efimova, V., Muravyov, S., and Filchenkov, A. (2018). Reinforcement-based Method for Simultaneous Clustering Algorithm Selection and its Hyper-parameters Optimization. Procedia Computer Science, 136:144–153.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. Advances in neural information processing systems, 25.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2018). Deep Image Prior. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 9446–9454.
A Omitted Proofs

A.1 Proof of Lemma 1

Proof of Lemma 1. Let $H(\lambda, x) \triangleq \log p_c(y - f(x)) + \lambda \log p_G(x) = -\mathcal{L}_{MAP}(x; \lambda)$. Then by (2) and (5), we have

$$H(\lambda_2, \hat{x}(\lambda_2))$$

$$\geq H(\lambda_2, \hat{x}(\lambda_1))$$

$$= H(\lambda_1, \hat{x}(\lambda_1)) + (\lambda_2 - \lambda_1) \log p_G(\hat{x}(\lambda_1))$$

$$\geq H(\lambda_1, \hat{x}(\lambda_2)) + (\lambda_2 - \lambda_1) \log p_G(\hat{x}(\lambda_1))$$

$$= H(\lambda_2, \hat{x}(\lambda_2)) + (\lambda_2 - \lambda_1) (\log p_G(\hat{x}(\lambda_1) - \log p_G(\hat{x}(\lambda_2))).$$

Thus we have

$$\log p_G(\hat{x}(\lambda_1)) \leq \log p_G(\hat{x}(\lambda_2)).$$

If $\log p_c(y - f(\hat{x}(\lambda_2))) > \log p_c(y - f(\hat{x}(\lambda_1)))$, then we have

$$H(\lambda_1, \hat{x}(\lambda_1)) < H(\lambda_1, \hat{x}(\lambda_2)).$$

This contradicts to the definition of $\hat{x}(\lambda_1)$. Therefore, $\log p_c(y - f(\hat{x}(\lambda_2))) \leq \log p_c(y - f(\hat{x}(\lambda_1)))$.

A.2 Proof of Remark 1

Proof of Remark 1. (i) If $\exists \mu' > 0$, $\exists \delta > 0$, s.t. $\forall \lambda \in [0, \Lambda]$, $\mathcal{L}_{MAP}(\cdot, \lambda)$ is $\mu'$-strongly convex on $B(\hat{x}(\lambda), \delta)$, then $\forall x \in B(\hat{x}(\lambda), \delta)$, we have

$$\mathcal{L}_{MAP}(x, \lambda)$$

$$\geq \mathcal{L}_{MAP}(\hat{x}, \lambda)$$

$$\geq \mathcal{L}_{MAP}(x, \lambda) + \nabla_x \mathcal{L}_{MAP}(x, \lambda)^T(\hat{x}(\lambda) - x) + \frac{\mu'}{2} ||\hat{x}(\lambda) - x||^2$$

$$\geq \mathcal{L}_{MAP}(x, \lambda) - ||\nabla_x \mathcal{L}_{MAP}(x, \lambda)|| ||\hat{x}(\lambda) - x|| + \frac{\mu'}{2} ||\hat{x}(\lambda) - x||^2.$$  

Therefore, we have

$$- ||\nabla_x \mathcal{L}_{MAP}(x, \lambda)|| ||\hat{x}(\lambda) - x|| + \frac{\mu'}{2} ||\hat{x}(\lambda) - x||^2 \leq 0,$$

from which we can see that

$$||\hat{x}(\lambda) - x|| \leq \frac{2}{\mu'} ||\nabla_x \mathcal{L}_{MAP}(x, \lambda)||.$$  

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

$$\mathcal{L}_{MAP}(x, \lambda) - \mathcal{L}_{MAP}(\hat{x}, \lambda)$$

$$= \nabla_x \mathcal{L}_{MAP}(\xi, \lambda)^T(x - \hat{x}(\lambda))$$

$$\leq ||\nabla_x \mathcal{L}_{MAP}(\xi, \lambda)|| ||x - \hat{x}(\lambda)||$$

$$= ||\nabla_x \mathcal{L}_{MAP}(\xi, \lambda) - \nabla_x \mathcal{L}_{MAP}(\hat{x}, \lambda)|| ||x - \hat{x}(\lambda)||$$

$$\leq L \xi - \hat{x}(\lambda)|| ||x - \hat{x}(\lambda)||$$

$$\leq L \xi - \hat{x}(\lambda)||^2$$

$$\leq L \xi^2 ||\nabla_x \mathcal{L}_{MAP}(x, \lambda)||^2$$
A.3 proof of Theorem 1

Before proving Theorem 1, we prove the following two lemmas first:

**Lemma 2.** If Assumption 1 holds, then \( \forall \lambda \in [0, \Lambda], \) let
\[
x_{k+1} = x_k - \frac{1}{L} \nabla_x L_{MAP}(x_k, \lambda).
\]
(7)

We have
\[
L_{MAP}(x_{k+1}, \lambda) \leq L_{MAP}(x_k, \lambda) - \frac{1}{2L} \| \nabla_x L_{MAP}(x_k, \lambda) \|^2.
\]
(8)

**Proof of Lemma 2.**

\[
\begin{align*}
L_{MAP}(x_{k+1}, \lambda) & \leq L_{MAP}(x_k, \lambda) + \nabla_x L_{MAP}(x_k, \lambda)^T (x_{k+1} - x_k) + \frac{L}{2} \| x_{k+1} - x_k \|^2 \\
& = L_{MAP}(x_k, \lambda) - \frac{1}{L} \| \nabla_x L_{MAP}(x_k, \lambda) \|^2 + \frac{1}{2L} \| \nabla_x L_{MAP}(x_k, \lambda) \|^2 \\
& = L_{MAP}(x_k, \lambda) - \frac{1}{2L} \| \nabla_x L_{MAP}(x_k, \lambda) \|^2.
\end{align*}
\]

\[\Box\]

**Lemma 3.** Assume Assumption 1 and 2 hold. Let \( \mu = \frac{1}{2L} \sigma^2 \). For any fixed \( \delta > 0 \), let
\[
\delta_0 = \min \left\{ \delta, \frac{\mu}{\sqrt{2(\mu L - \mu)\delta}} \right\}.
\]

If \( x_0 \in B(\hat{x}(\lambda), \delta_0) \), and \( \{x_k\} \) is generated by (15), then \( \forall k \in \mathbb{N}_+, x_k \in B(\hat{x}(\lambda), \delta) \).

**Proof of Lemma 3.** We prove by induction on \( k \). When \( k = 0 \), we have
\[
x_0 \in B(\hat{x}(\lambda), \delta_0) \subset B(\hat{x}(\lambda), \delta).
\]

Assume when \( k \geq 1, x_j \in B(\hat{x}(\lambda), \delta), \forall j \leq k \), then by (8) we have
\[
L_{MAP}(x_k, \lambda) - L_{MAP}(\hat{x}, \lambda) \\
\leq L_{MAP}(x_{k-1}, \lambda) - L_{MAP}(\hat{x}, \lambda) - \frac{1}{2L} \| \nabla_x L_{MAP}(x_{k-1}, \lambda) \|^2.
\]

By induction hypothesis and (6), we have
\[
\begin{align*}
L_{MAP}(x_{k-1}, \lambda) - L_{MAP}(\hat{x}, \lambda) - \frac{1}{2L} \| \nabla_x L_{MAP}(x_{k-1}, \lambda) \|^2 \\
\leq (1 - \frac{\mu}{L}) (L_{MAP}(x_{k-1}, \lambda) - L_{MAP}(\hat{x}, \lambda)).
\end{align*}
\]

From the above two inequalities we deduce
\[
\begin{align*}
L_{MAP}(x_k, \lambda) - L_{MAP}(\hat{x}, \lambda) \\
\leq (1 - \frac{\mu}{L}) (L_{MAP}(x_{k-1}, \lambda) - L_{MAP}(\hat{x}, \lambda)) \\
\leq \cdots \\
\leq (1 - \frac{\mu}{L})^k (L_{MAP}(x_0, \lambda) - L_{MAP}(\hat{x}, \lambda)).
\end{align*}
\]
From (8) we can see that
\[ \mathcal{L}_{MAP}(\hat{x}, \lambda) \leq \mathcal{L}_{MAP}(x_k, \lambda) - \frac{1}{2L} \|
abla_x \mathcal{L}_{MAP}(x_k, \lambda)\|^2. \] (10)

Thus we have
\[
\begin{align*}
\mu \|x_k - \hat{x}(\lambda)\|^2 \\
\leq \frac{1}{2L} \|
abla_x \mathcal{L}_{MAP}(x_k, \lambda)\|^2 \\
\leq \mathcal{L}_{MAP}(x_k, \lambda) - \mathcal{L}_{MAP}(\hat{x}, \lambda) \\
\leq (1 - \frac{\mu}{L})^k (\mathcal{L}_{MAP}(x_0, \lambda) - \mathcal{L}_{MAP}(\hat{x}, \lambda)) \\
\leq (1 - \frac{\mu}{L})^k \frac{1}{2\mu} \|
abla_x \mathcal{L}_{MAP}(x_0, \lambda)\|^2 \\
\leq (1 - \frac{\mu}{L})^k \frac{L^2}{2\mu} \|x_0 - \hat{x}(\lambda)\|^2,
\end{align*}
\]

where (a) is by Assumption 2 (recall that \(\mu = \frac{1}{2L\sigma^2}\)), (b),(c),(d) use (10), (9) and (6), resp. (e) is by Assumption 1.

From the above inequalities we deduce
\[
\|x_k - \hat{x}(\lambda)\| \leq \frac{L}{\sqrt{2\mu}} (1 - \frac{\mu}{L})^{k/2} \|x_0 - \hat{x}(\lambda)\| \leq \frac{\delta}{2}. \tag{11}
\]

Note that
\[
\begin{align*}
\|x_{k+1} - x_k\| &= \frac{1}{L} \|
abla_x \mathcal{L}_{MAP}(x_k, \lambda)\| \\
&= \frac{1}{L} \|
abla_x \mathcal{L}_{MAP}(x_k, \lambda) - \nabla_x \mathcal{L}_{MAP}(\hat{x}, \lambda)\| \\
&\leq \|x_k - \hat{x}(\lambda)\| \leq \frac{\delta}{2}.
\end{align*}
\]

So we have
\[
\|x_{k+1} - \hat{x}(\lambda)\| \leq \|x_k - \hat{x}(\lambda)\| + \|x_{k+1} - x_k\| \leq \delta.
\]

The induction step is complete. \(\square\)

Now we give the proof of Thm. 1.

**Proof of Theorem 1.** From the first inequality of (11) we know that if \(x_0 \in B(\hat{x}(\lambda), \delta_0)\), then
\[
\|x_k - \hat{x}(\lambda)\| \leq (1 - \frac{\mu}{L})^{k/2} \delta, \quad \forall k \in \mathbb{N}. \tag{12}
\]

Let \(\lambda_0 = 0, \lambda_{i+1} = \lambda_i + \frac{A}{N}, i = 1, 2, \ldots, N - 1\), where \(N \geq \frac{2C\Delta}{\delta_0}\). Let \(\Delta \lambda \triangleq \frac{A}{N}\). Suppose \(x_k(\lambda_i)\) is the point in the last iteration of the inner loop when \(\lambda = \lambda_i\), where \(K = \lceil \frac{2\log(\Delta\lambda/\delta_0)}{\log(L/(1-\mu))} \rceil + 1\).

Let \(x_0(\lambda_0) = G^{-1}(y)\), then we have
\[
\|x_0(\lambda_0) - \hat{x}(\lambda_0)\| = 0. \tag{13}
\]

If \(\|x_0(\lambda_i) - \hat{x}(\lambda_i)\| \leq \delta_0\), then by (12) we deduce
\[
\|x_k(\lambda_i) - \hat{x}(\lambda_i)\| \leq \delta.
\]
Algorithm 4 Alternative Cascade Optimization for $x$

1: **Input:** observation $y$, generative model $G$, targeted hyperparameter $\Lambda > 0$, target change rate of loss magnitude $r \in (0, 1)$, minimum step size $\eta_{\min} > 0$ for updating $\lambda$, maximum iteration number $K > 0$ and step size $\alpha > 0$ for solving $L_{\text{MAP}}(x_k; \lambda_t)$.

2: **Initialize:** $\lambda = 0$, $x_0 = f^{-1}(y)$

3: **repeat**

4: for $k = 1, \ldots, K$ do

5:  $x_{k+1} = x_k - \alpha \nabla_x L_{\text{MAP}}(x_k, \lambda)$

6: end for

7: $e = y - f(x_{K+1})$

8: Determine the step size $\eta$ such that

$$r = \frac{|L_{\text{MAP}}(x_{K+1}; \lambda + \eta) - L_{\text{MAP}}(x_{K+1}; \lambda)|}{L_{\text{MAP}}(x_{K+1}; \lambda)}$$

while satisfying $\eta \geq \eta_{\min}$, i.e.,

$$\eta = \max \left( r \left| \frac{\log p_e(e)}{\log p_G(x_{K+1})} + \lambda \right|, \eta_{\min} \right)$$

9: $\lambda = \lambda + \eta$

10: $x_0 = x_{K+1}$

11: **until** $\lambda \geq \Lambda$

12: return $x_{K+1}$

When $\lambda = \lambda_{i+1}$, let $x_0(\lambda_{i+1}) = x_K(\lambda_i)$, then we have

$$\|x_0(\lambda_{i+1}) - \hat{x}(\lambda_{i+1})\|$$

$$\leq \|x_0(\lambda_{i+1}) - \hat{x}(\lambda_i)\| + \|\hat{x}(\lambda_i) - \hat{x}(\lambda_{i+1})\|$$

$$\leq \|x_K(\lambda_i) - \hat{x}(\lambda_i)\| + C \Delta \lambda$$

$$(a)$$

$$\leq \delta_0,$$

where (a) uses Assumption 3 and $x_0(\lambda_{i+1}) = x_K(\lambda_i)$.

Thus by (13) and induction we deduce

$$\|x_0(\lambda_i) - \hat{x}(\lambda_i)\| \leq \delta_0, \forall i \in \{0, 1, \cdots, N\}.$$

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

$$\|x_K(\Lambda) - \hat{x}(\Lambda)\| \leq \varepsilon.$$

\[\square\]

### B Alternative Cascade Optimization for $x$

Here we present an alternative version of Cascade Optimization for $x$, which is used in practice.

### C Cascade Optimization for $z$

Let $F(\lambda, z) = L_{\text{MAP}}(G(z), \lambda)$. Let $z^*(\lambda) = \arg\min_z F(\lambda, z)$. We specify that under some assumptions, for any fixed $\varepsilon > 0$ and $\Lambda > 0$, Algorithm 5 returns an $\varepsilon$-approximation of $z^*(\Lambda)$ by gradually increase $\lambda$ from 0 to $\Lambda$. 

Theorem 2. The following Theorem shows that Alg. 5 returns an $\varepsilon$-approximation of $z^*(\Lambda)$. See App.C.2 for the proofs:

**Theorem 2.** Under Assumption 4, 5 and 6, for any $\varepsilon > 0$, Algorithm 5 returns $z_0$ that satisfies $\|z_0 - z^*(\Lambda)\| \leq \varepsilon$.

**C.1 Proof of Remark 2**

*Proof of Remark 2.* (i) If $\exists \mu' > 0$, $\exists \delta > 0$, s.t. $\forall \lambda \in [0, \Lambda]$, $F(\lambda, \cdot)$ is $\mu'$-strongly convex on $B(z^*(\lambda), \delta)$, then $\forall z \in B(z^*(\lambda), \delta)$, we have

$$
F(\lambda, z) \\
\geq F(\lambda, z^*(\lambda)) \\
\geq F(\lambda, z) + \nabla F(\lambda, z)^T (z^*(\lambda) - z) + \frac{\mu'}{2} \|z^*(\lambda) - z\|^2 \\
\geq F(\lambda, z) - \|\nabla F(\lambda, z)\| \|z^*(\lambda) - z\| + \frac{\mu'}{2} \|z^*(\lambda) - z\|^2.
$$

Therefore, we have

$$
-\|\nabla F(\lambda, z)\| \|z^*(\lambda) - z\| + \frac{\mu'}{2} \|z^*(\lambda) - z\|^2 \leq 0,
$$
from which we can see that 
\[ \| z^*(\lambda) - z \| \leq \frac{2}{\mu} \| \nabla_z F(\lambda, z) \|. \]

(ii) \( \forall z \in B(z^*(\lambda), \delta), \exists t \in [0, 1], \xi = tz^*(\lambda) + (1-t)z, \) s.t.
\[
F(\lambda, z) - F(\lambda, z^*(\lambda)) \leq \| \nabla_z F(\lambda, z) \| \cdot \mu \| z - z^*(\lambda) \|
\]
\[
\leq L \| \xi - z^*(\lambda) \| \| z - z^*(\lambda) \|
\]
\[
\leq L \| z - z^*(\lambda) \|^2
\]
\[
\leq L \sigma^2 \| \nabla_z F(\lambda, z) \|^2.
\]

C.2 proof of Theorem 2

Before proving Theorem 2, we prove the following two lemmas first:

**Lemma 4.** If Assumption 4 holds, then \( \forall \lambda \in [0, \Lambda], \) let
\[
z_{k+1} = z_k - \frac{1}{L} \nabla_z F(\lambda, z_k).
\]

We have
\[
F(\lambda, z_{k+1}) \leq F(\lambda, z_k) - \frac{1}{2L} \| \nabla_z F(\lambda, z_k) \|^2.
\]

**Proof of Lemma 4.**

\[
F(\lambda, z_{k+1}) \leq F(\lambda, z_k) + \nabla_z F(\lambda, z_k)^T(z_{k+1} - z_k) + \frac{L}{2} \| z_{k+1} - z_k \|^2
\]
\[
= F(\lambda, z_k) - \frac{1}{L} \| \nabla_z F(\lambda, z_k) \|^2 + \frac{1}{2L} \| \nabla_z F(\lambda, z_k) \|^2
\]
\[
= F(\lambda, z_k) - \frac{1}{2L} \| \nabla_z F(\lambda, z_k) \|^2.
\]

**Lemma 5.** Assume Assumption 4 and 2 hold. Let \( \mu = \frac{1}{2L \sigma^2}. \) For any fixed \( \delta > 0, \) let
\[
\delta_0 = \min \left\{ \delta, \frac{\mu}{\sqrt{2(L - \mu)L}} \delta \right\}.
\]

If \( z_0 \in B(z^*(\lambda), \delta_0), \) and \( \{ z_k \} \) is generated by (15), then \( \forall k \in \mathbb{N}_+, z_k \in B(z^*(\lambda), \delta). \)

**Proof of Lemma 5.** We prove by induction on \( k. \) when \( k = 0, \) we have
\[ z_0 \in B(z^*(\lambda), \delta_0) \subset B(z^*(\lambda), \delta). \]

Assume when \( k \geq 1, z_j \in B(z^*(\lambda), \delta), \forall j \leq k, \) then by (16) we have
\[
F(\lambda, z_k) - F(\lambda, z^*(\lambda)) \leq F(\lambda, z_{k-1}) - F(\lambda, z^*(\lambda)) - \frac{1}{2L} \| \nabla_z F(\lambda, z_{k-1}) \|^2.
\]
By induction hypothesis and (14), we have
\[ F(\lambda, z_{k-1}) - F(\lambda, z^*(\lambda)) - \frac{1}{2L} \| \nabla_z F(\lambda, z_{k-1}) \|^2 \]
\[ \leq (1 - \frac{\mu}{L})(F(\lambda, z_{k-1}) - F(\lambda, z^*(\lambda))). \]

From the above two inequalities we deduce
\[ F(\lambda, z_k) - F(\lambda, z^*(\lambda)) \]
\[ \leq (1 - \frac{\mu}{L})(F(\lambda, z_{k-1}) - F(\lambda, z^*(\lambda))) \]
\[ \leq \cdots \]
\[ \leq (1 - \frac{\mu}{L})^k(F(\lambda, z_0) - F(\lambda, z^*(\lambda))). \] (17)

From (16) we can see that
\[ F(\lambda, z^*(\lambda)) \leq F(\lambda, z_k) - \frac{1}{2L} \| \nabla_z F(\lambda, z_k) \|^2. \] (18)

Thus we have
\[ \mu \| z_k - z^*(\lambda) \|^2 \]
\[ \leq \frac{1}{2L} \| \nabla_z F(\lambda, z_k) \|^2 \]
\[ \leq F(\lambda, z_k) - F(\lambda, z^*(\lambda)) \]
\[ \leq (1 - \frac{\mu}{L})^k(F(\lambda, z_0) - F(\lambda, z^*(\lambda))) \]
\[ \leq (1 - \frac{\mu}{L})^k \frac{1}{2\mu} \| \nabla_z F(\lambda, z_0) \|^2 \]
\[ \leq (1 - \frac{\mu}{L})^k \frac{L^2}{2\mu} \| z_0 - z^*(\lambda) \|^2, \]

where (a) is by Assumption 5 (recall that \( \mu = \frac{1}{2L\sigma^2} \)), (b),(c),(d) use (18), (17) and (14), resp. (e) is by Assumption 4.

From the above inequalities we deduce
\[ \| z_k - z^*(\lambda) \| \leq \frac{L}{\sqrt{2\mu}}(1 - \frac{\mu}{L})^{k/2} \| z_0 - z^*(\lambda) \| \leq \frac{\delta}{2}. \] (19)

Note that
\[ \| z_{k+1} - z_k \| = \frac{1}{L} \| \nabla_z F(\lambda, z_k) \| \]
\[ = \frac{1}{L} \| \nabla_z F(\lambda, z_k) - \nabla_z F(\lambda, z^*(\lambda)) \| \]
\[ \leq \| z_k - z^*(\lambda) \| \leq \frac{\delta}{2}. \]

So we have
\[ \| z_{k+1} - z^*(\lambda) \| \leq \| z_k - z^*(\lambda) \| + \| z_{k+1} - z_k \| \leq \delta. \]

The induction step is complete. \[ \square \]

Now we give the proof of Thm. 2.
Proof of Theorem 2. From the first inequality of (19) we know that if \( z_0 \in B(z^*(\lambda), \delta_0) \), then
\[
\|z_k - z^*(\lambda)\| \leq (1 - \frac{\mu}{L})^{k/2} \delta, \forall k \in \mathbb{N}.
\] (20)

Let \( \lambda_0 = 0, \lambda_{i+1} = \lambda_i + \frac{\Delta}{N}, i = 1, 2, \cdots, N - 1 \), where \( N \geq \frac{2CA}{\delta_0} \). Let \( \Delta \lambda \triangleq \frac{\Delta}{N} \). Suppose \( z_k(\lambda_i) \) is the point in the last iteration of the inner loop when \( \lambda = \lambda_i \), where \( K = \lceil \frac{2 \log(2\varepsilon/\delta_0)}{\log(L/(L-\mu))} \rceil + 1 \).

Let \( z_0(\lambda_0) = G^{-1}(y) \), then we have
\[
\|z_0(\lambda_0) - z^*(\lambda_0)\| = 0.
\] (21)

If \( \|z_0(\lambda_i) - z^*(\lambda_i)\| \leq \delta_0 \), then by (20) we deduce
\[
\|z_K(\lambda_i) - z^*(\lambda_i)\| \leq \delta.
\]

When \( \lambda = \lambda_{i+1} \), let \( z_0(\lambda_{i+1}) = z_K(\lambda_i) \), then we have
\[
\|z_0(\lambda_{i+1}) - z^*(\lambda_{i+1})\|
\leq \|z_0(\lambda_{i+1}) - z^*(\lambda_i)\| + \|z^*(\lambda_i) - z^*(\lambda_{i+1})\|
\leq \|z_K(\lambda_i) - z^*(\lambda_i)\| + C\Delta \lambda
\leq \delta_0,
\]
where (a) uses Assumption 6 and \( z_0(\lambda_{i+1}) = z_K(\lambda_i) \).

Thus by (21) and induction we deduce
\[
\|z_0(\lambda_i) - z^*(\lambda_i)\| \leq \delta_0, \forall i \in \{0, 1, \cdots, N\}.
\]

When \( i = N, K = \max\{0, \lceil \frac{2 \log(2\varepsilon/\delta_0)}{\log(L/(L-\mu))} \rceil + 1 \} \), again using the first inequality in (19), we have
\[
\|z_K(\lambda) - z^*(\lambda)\| \leq \varepsilon.
\]

\[\square\]

D Reconstructions on NCS tasks

We show reconstructions from 4000 and 2000 measurements in Figure 4 and 5 respectively.
Figure 4: Reconstructions of NCS at 4000 measurements, $\lambda \in [0, 5]$.

Figure 5: Reconstructions of NCS at 2000 measurements, $\lambda \in [0, 5]$. 