PANOM: Automatic Hyper-parameter Tuning for Inverse Problems

Tianci Liu
Purdue University
`liu3351@purdue.edu`

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

Qi Lei
Princeton University
`qilei@princeton.edu`

Abstract

Automated hyper-parameter tuning for unsupervised learning, including inverse problems, remains a long-standing open problem due to the lack of validation data. In this work, we design an automatic tuning criterion for inverse problems and formulate it as a bilevel optimization task. We demonstrate the efficiency of our tuning scheme on various inverse problems and different test and out-of-distribution image samples at no expense of performance drops.

1 Introduction

Inverse problems using generative models aim to reconstruct a signal $\boldsymbol{x}^*$ via a noisy or lossy observation $\boldsymbol{y} = A\boldsymbol{x}^* + \boldsymbol{\epsilon}$ with the assumption that $\boldsymbol{x}$ comes from a generative model G, i.e., $\boldsymbol{x} = G(\boldsymbol{z})$ for some latent code $\boldsymbol{z}$. A is a known forward operator and $\boldsymbol{\epsilon}$ represents noise. The problem is reduced to denoising when A is an identity matrix, and is reduced to a compressed sensing [5, 7], inpainting [18], or super-resolution problem [12] when A maps the signal to a lower dimensional observation. Existing works on solving inverse problems often design the objective by

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}}{\operatorname{argmin}}\, L_{\text{recon}}(G(\boldsymbol{z}), \boldsymbol{y}) + \beta L_{\text{reg}}(\boldsymbol{z}), \quad \hat{\boldsymbol{x}} = G(\hat{\boldsymbol{z}}), \tag{1}$$

where L_{recon} is a reconstruction error between the observation $\boldsymbol{y}$ and a recovered signal $G(\boldsymbol{z})$ [4, 16] and $L_{\text{reg}}(\boldsymbol{z})$ multiplied by hyper-parameter β regularizes $\boldsymbol{z}$ via incorporating prior information of its latent distribution. This loss function is subject to different noise models [1, 19, 17].

The success of this model largely depends on a correct choice of hyper-parameter β. Specifying appropriate candidates of β 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) hyper-parameter tuning for unsupervised 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 hyper-parameter tuning for inverse problems.

Our contribution. We propose an automated hyper-parameter tuning scheme for inverse problems through a formulation of a model-based bilevel optimization objective. We empirically demonstrate the performance on different tasks of inverse problems and show a significant convenience and efficiency of our approach compared to hand-tuned grid search.

2 Related work

Inverse problems using generative models have impressive performance on applications of computed tomography [20], MRI [21], and computer vision tasks [4], to name a few. Recent work uses GANs/VAEs or even untrained models to enforce the generative prior [4, 11, 16]. More recent work investigates flow-based models [1, 19] where the explicit probabilistic models formulate the problem as MAP inference and are able to solve structured and complex noise models.

Automated hyper-parameter tuning is essential to improve the efficiency, performance, and reproducibility for general machine learning problems [15, 8]. Common strategies on hyper-parameter tuning include grid search [2], random search [3], evolutionary optimization [14] and population-based training [9]. Model-based tuning scheme requires a surrogate model of the validation loss as a function of the hyper-parameters. Our algorithm stems from Bayesian optimization, an example of model-based tuning scheme [15, 2].

3 Methodology

Preliminary. Suppose $\boldsymbol{x}^* \in \mathbb{R}^d$ and $\boldsymbol{y} \in \mathbb{R}^m$ $(m \ll d)$. We try to reconstruct $\boldsymbol{x}^*$ by $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{z}}$ such that $G(\hat{\boldsymbol{z}}) = \hat{\boldsymbol{x}}$ and focus on a *normalizing flow* model $G : \mathbb{R}^d \to \mathbb{R}^d$. Since G is invertible, change of variables formula allows us to compute prior density $p_G(\boldsymbol{x})$ for each $\boldsymbol{x}$ given a known distribution of $\boldsymbol{z}$. One can find $\hat{\boldsymbol{x}}$ by maximum a posteriori (MAP) inference [1, 19]:

$$\hat{\boldsymbol{x}} \in \underset{\boldsymbol{x}}{\operatorname{argmax}} \{L_\beta(\boldsymbol{x}; \boldsymbol{y}) := \log p(\boldsymbol{y} \mid \boldsymbol{x}) + \beta \log p_G(\boldsymbol{x})\}. \tag{2}$$

Due to the lack of validation data, it is hard to find a criterion to guide the update of β which, consequently, is often selected by grid search. We design bilevel optimization objectives to choose β by making the recoveries of the task sample $\boldsymbol{y}$ comparable to prior samples.

3.1 An automated tuning criterion and bilevel optimization

We seek a criterion to update β using information of prior samples $\{\tilde{\boldsymbol{x}}\}$ on which G is trained, as if $\{\tilde{\boldsymbol{x}}\}$ are used for validation. For a task sample $\boldsymbol{y}$ let $\hat{\boldsymbol{x}}^{\text{MLE}} = \operatorname{argmax}_{\boldsymbol{x}} \log p(\boldsymbol{y} \mid \boldsymbol{x})$, namely the solution to (2) with $\beta = 0$ and $\hat{\boldsymbol{z}}^{\text{MLE}} = G^{-1}(\hat{\boldsymbol{x}}^{\text{MLE}})$. $\hat{\boldsymbol{x}}^{\text{MLE}}$ ($\hat{\boldsymbol{z}}^{\text{MLE}}$) is introduced as an intermedium between the unknown ground truth $\boldsymbol{x}^*$ ($\boldsymbol{z}^*$) and $\hat{\boldsymbol{x}}$ ($\hat{\boldsymbol{z}}$) of (2) and facilitates borrowing information from $\{\tilde{\boldsymbol{x}}\}$. Given a prior sample $\tilde{\boldsymbol{x}}$, we evaluated $\tilde{\boldsymbol{y}} = A\tilde{\boldsymbol{x}} + \boldsymbol{\epsilon}$, $\tilde{\boldsymbol{z}} = G^{-1}(\tilde{\boldsymbol{x}})$, $\tilde{\boldsymbol{x}}^{\text{MLE}} = \operatorname{argmax}_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{y}} \mid \boldsymbol{x})$ and $\tilde{\boldsymbol{z}}^{\text{MLE}} = G^{-1}(\tilde{\boldsymbol{x}}^{\text{MLE}})$. Let Q_1 denote the (empirical) distribution of $\text{PSNR}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}^{\text{MLE}})$[1] of all $\{\tilde{\boldsymbol{x}}\}$. If $\boldsymbol{y}$ has been well recovered as $\hat{\boldsymbol{x}}$ by G, the distribution of $\text{PSNR}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^{\text{MLE}})$ should be similar to Q_1. Moreover, appropriate regularization should be imposed on the latent code $\hat{\boldsymbol{z}}$ so that a norm of $\hat{\boldsymbol{z}}$ should be similar to the norm of $\tilde{\boldsymbol{z}}$. Utilizing $\tilde{\boldsymbol{z}}^{\text{MLE}}$, we define Q_2 to be the (empirical) conditional distribution of $\|\tilde{\boldsymbol{z}}\|$ given $\|\tilde{\boldsymbol{z}}^{\text{MLE}}\|$, where $\|\cdot\|$ denotes ℓ_2 norm. If $\hat{\boldsymbol{x}} = G(\hat{\boldsymbol{z}})$ is a good recovery, the conditional distribution of $\|\hat{\boldsymbol{z}}\|$ given $\|\hat{\boldsymbol{z}}^{\text{MLE}}\|$ should be similar to Q_2. In other words, low values of the density $Q_1(\text{PSNR}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^{\text{MLE}}))$ and/or the density $Q_2(\|\hat{\boldsymbol{z}}\| \mid \|\hat{\boldsymbol{z}}^{\text{MLE}}\|)$ implies that $\hat{\boldsymbol{x}}$ is not a good recovery of $\boldsymbol{y}$ and thus better choice of β can be achieved. Hereby, we solve the inverse problem with automated updates of β by bilevel optimization:

$$\max_{\beta: \beta \geq 0} R(\beta, \hat{\boldsymbol{x}}), \text{ s.t., } \hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmax}} L_\beta(\boldsymbol{x}; \boldsymbol{y}), \tag{3}$$

where $R(\beta, \hat{\boldsymbol{x}}) := \alpha Q_1(\text{PSNR}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^{\text{MLE}})) + (1 - \alpha) Q_2(\|\hat{\boldsymbol{z}}\| \mid \|\hat{\boldsymbol{z}}^{\text{MLE}}\|)$ and $\alpha \in [0, 1]$ balances the impacts of the two densities. Using the bilevel optimization algorithm shortly proposed in Section 3.2, we found the recovery is not sensitive to the choice of α. Parallel work [6] also presented the general principle of formulating hyperparameter-tuning in inverse problem as bilevel optimization.

3.2 PSNR and norm matching based hyper-parameter tuning

It is computationally intractable to find densities of Q_1 and Q_2. A potential solution is to learn the Kullback–Leibler divergence of the distribution of $\text{PSNR}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^{\text{MLE}})$ ($\|\hat{\boldsymbol{z}}\|$ given $\|\hat{\boldsymbol{z}}^{\text{MLE}}\|$) from $Q1$ (Q_2) [13]. However, this results in a tri-level optimization problem whose convergence can questionable. Circumventing such difficulties, we propose a *PSNR And NOrm Matching* (PANOM) based zeroth order optimization method. In particular, we match the median PSNR for the task samples to the median of Q_1, and match the mean ℓ_2 norms of latent codes for the task samples to the mean of Q_2. Concretely, define

$$\gamma_1 := \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}(\text{PSNR}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}^{\text{MLE}}) > \text{PSNR}(\tilde{\boldsymbol{x}}_n, \tilde{\boldsymbol{x}}_n^{\text{MLE}})) - 0.5,\ \gamma_2 := \|\hat{\boldsymbol{z}}\| / h(\|\hat{\boldsymbol{z}}^{\text{MLE}}\|) - 1, \tag{4}$$

where $\mathbb{1}(\cdot)$ is the indicator function. $h(\cdot)$ is a linear function of $\|\hat{\boldsymbol{z}}^{\text{MLE}}\|$ that estimates the ground true $\|\boldsymbol{z}^*\|$ and has been trained with a mean squared error loss on $\{\|\tilde{\boldsymbol{z}}^{\text{MLE}}\|, \|\tilde{\boldsymbol{z}}\|\}$. More complex functional forms of h also apply. γ_1 (γ_2) measures the deviance of the median PSNA (mean ℓ_2 norm of latent code) between the task and prior samples and is calculated and used in each update of β. Implementation of PANOM is shown in Algorithm 1. Signed squares of γ_1 and γ_2 as in Δ amplifies

[1] *Peak signal-to-noise ratio* (PSNR) quantifies the degrees of similarity between two signals, like images.

Algorithm 1 PANOM based automated hyper-parameter tuning for inverse problems

```
Input: data $\boldsymbol{y}$, prior $Q_1, Q_2, h$, and $G$, $\beta_{\text{init}}$, learning rate $\eta$, and maximum number of iterations $T$.
Initialize $\hat{\boldsymbol{z}} = \mathbf{0}, \beta = \beta_{\text{init}}$; estimate $\hat{\boldsymbol{z}}^{\text{MLE}}(\hat{\boldsymbol{x}}^{\text{MLE}})$ with maximum $\frac{T}{3}$ steps.
while iteration number not reaching $T$ do
    Inner-loop steps: update $\hat{\boldsymbol{z}}$ by $K$ steps of gradient ascent of $\log p(\boldsymbol{y} \, j \, G(\boldsymbol{z})) + \beta \log p_G(G(\boldsymbol{z}))$
    Outer-loop step: update $\beta$ with learning rate $\eta$ by
        $\beta \quad \beta \quad (1 + \eta\Delta), \quad \text{where } \Delta = \alpha \text{sign}(\gamma_1)\gamma_1^2 + (1 \quad \alpha)\text{sign}(\gamma_2)\gamma_2^2$
end while
```

the learning rate η of β when γ_1 and γ_2 are large. The updates of β push PSNR$(\boldsymbol{x}_n, \hat{\boldsymbol{x}}_n^{\text{MLE}})$ towards the empirical median of Q_1 and $k\hat{z}k$ towards to the mean of Q_2 and thus facilitate better recoveries.

4 Experiment

We evaluate our method on denoising and noisy compressed sensing (NCS) tasks by comparing with the *best* results (measured by PSNR$(\hat{\boldsymbol{x}}, \boldsymbol{x}^*)$[2]) from Grid Search as baselines. We report qualitative and quantitative results on both test examples from CelebA-HQ faces and *out-of-distribution* (OOD) examples. The efficiency of our method is revealed by numbers of searching grids required to achieve comparable performances to PANOM.

Tasks. Let $\boldsymbol{\epsilon}$ be an imposed noise. For denoising task we study additive Gaussian case $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \quad N(\mathbf{0}, 0.01\mathbf{I})$, and non-additive Poisson case $\boldsymbol{y} = \boldsymbol{\epsilon}/\gamma, \boldsymbol{\epsilon} \quad \text{Poisson}(\gamma\boldsymbol{x})$ where $\gamma = 50$. For NCS task we use $\boldsymbol{y} = \mathsf{A}\boldsymbol{x} + \boldsymbol{\epsilon}$ where $\mathsf{A} \; 2 \; \mathsf{R}^{m \times d}$ is a random Gaussian measurement matrix for $m = 2500 \quad 0.3d$, and $\boldsymbol{\epsilon}$ is a Gaussian noise satisfied $\mathsf{E}[\boldsymbol{\epsilon}] = 0, \; \sqrt{\mathsf{E}[k\boldsymbol{\epsilon}k_2^2]} = 0.1$ followed [4].

Implementation details. In PANOM we initialize $\beta = 0.1$ throughout experiments; empirically we find that the initial value is often of little importance. We further use learning rate $\eta = 0.5$ and try $\alpha \; 2 \; [0.5, 0, 1]$. In Grid Search, based on hyper-parameter choices from [19], we search over $[0.05, 2]$ with $5, 10, 15$ grids on denoising tasks; for NCS task, Whang et al. 2020 [19] use $\beta = 100$, which leaves us a huge space to search over. So we run a two-stage Grid Search: first we choose the best two candidates $\beta_{\text{low}} < \beta_{\text{upper}}$ from $[0.1, 1, 10, 100, 200, 500]$. Then we search over $[\beta_{\text{low}}, \beta_{\text{upper}}]$ with $5, 10, 15$ grids, and the total grid numbers are $9, 14, 19$. By using the same maximum number of iterations T, these grid numbers are proportional to the ratios of computation costs.

Results. Figure 1 is a subset of recovered examples on test and OOD images on denoising Gaussian noise task, and more examples with different α and grid numbers on the three tasks are given in Appendix B. Qualitatively, PANOM is able to keep more facial and other details (like the wall in column 2, and hair band in column 5) without adding unnatural textures as Grid Search did when recovering faces for both human (column 2 to 4) and fictional characters (column 7).

Summarized in Table 1 are PSNR values on the three tasks. Overall, PANOM has comparable performance to Grid Search on all scenarios, and slightly outperforms Grid Search on two denoising tasks. These results implies the high efficiency of our method. Specifically, for grid search with fixed β it quires $T = 1200$ to converge. Searching over 5 grids requires 6000 total steps, $12,000$ steps for 10 grids, and 18000 steps for 15 grids. In stark contrast, PANOM only requires 1600 steps to achieve the equal or better performance.

Figure 2 demonstrates the convergences of our method on OOD examples by trajectory plots of β on the three tasks. *Optimal* values determined by Grid Search are marked by dotted horizontal lines. Typically, PANOM has converged within 600 steps on denoising tasks and within 2000 steps for the harder NCS task. Moreover, the overlap between β tuned by the two algorithms justifies the effectiveness of PANOM in tuning β.

Finally, we examine the effects of α. Since α balances the two matching with the same goal that recoveries are similar to priors, tuning α is less crucial than tuning β which controls a tradeoff between details and smoothness. Empirically, different α often had noncritical impacts as shown by PSNR values in Table 1) and the recovered images in Figure 1. This advantage is demonstrated by Figure 2 as varying values of α does not result in poor performance.

[2]Note that PSNR$(\hat{\boldsymbol{x}}; \boldsymbol{x}^*)$ is not attainable in practice and requires manual selection on the optimal .

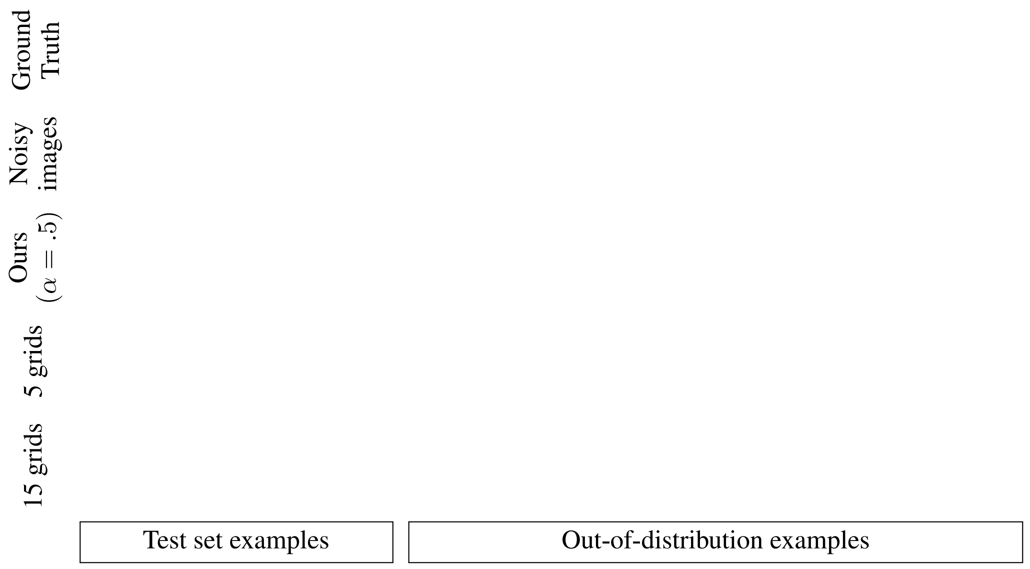

Figure 1: Result of denoising Gaussian noise on CelebA-HQ faces and out-of-distribution images. For grid search we show recovered images achieving the highest PSNR$(\hat{\boldsymbol{x}}, \boldsymbol{x})$. Full version can be found in appendix B.

Table 1: Averages and standard errors of recovered PSNR$(\hat{\boldsymbol{x}}, \boldsymbol{x})$ on test and OOD examples, best results are in bold. Numbers 9, 14, 19 in parentheses in $\#(\text{grids})$ are total grid numbers on NCS task. Computational cost of Grid Search is $\frac{3}{4}\#(\text{grids})$ times of our method.

Task	Noise	Examples	Ours, $\alpha =$			Grid Search, #(grids) $=$		
			.5	0	1	5(9)	10(14)	15(19)
Denoise	Gaussian	Test set	29.8 0.1	**30.1 $\pm$ 0.1**	29.0 0.1	28.1 0.1	29.9 0.1	29.8 0.1
		OOD	**29.0 $\pm$ 0.5**	27.6 0.5	28.5 0.4	26.5 0.4	28.9 0.4	28.7 0.5
	Poisson	Test set	30.6 0.3	**31.0 $\pm$ 0.2**	29.8 0.3	29.5 0.2	30.7 0.1	30.8 0.2
		OOD	29.1 0.6	29.0 0.6	**29.2 $\pm$ 0.6**	27.2 0.6	28.8 0.6	28.5 0.6
NCS	Gaussian	Test set	32.2 0.2	33.9 0.2	29.7 0.2	34.2 0.2	34.3 0.2	**34.9 $\pm$ 0.3**
		OOD	31.6 1.1	31.9 1.0	30.9 1.1	**32.3 $\pm$ 0.9**	32.1 1.0	32.2 0.9

5 Conclusion

In this work, we propose PANOM, an automated hyper-parameter tuning strategy for inverse problems. Our method shows comparable recovery performance with manual grid search, but requires no human guidance and is much more efficient and convenient. We also conjecture that the proposed bilevel optimization starting from small β will facilitate the optimization procedure, and we leave the theoretical understanding to future studies.

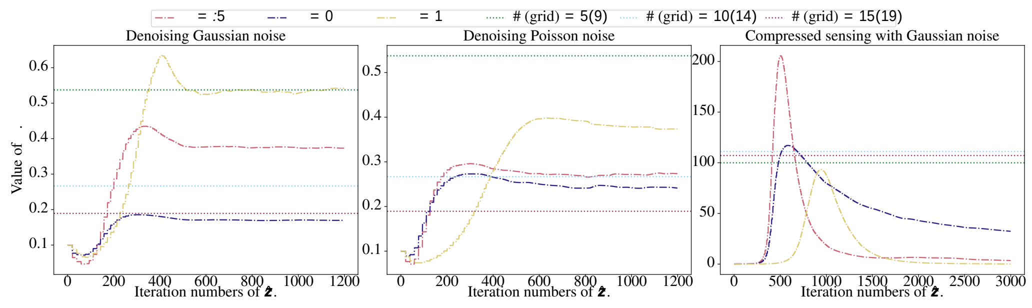

Figure 2: Trajectories of β from different tasks on OOD examples. Grid Search dotted lines are best choices we found; numbers 9, 14, 19 in parentheses are total grid numbers on NCS task.

References

[1] Muhammad Asim, Ali Ahmed, and Paul Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. *CoRR*, abs/1905.11672, 2019.

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[4] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 537–546. JMLR. org, 2017.

[5] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

[6] Caroline Crockett and Jeffrey A Fessler. Bilevel methods for image reconstruction. *arXiv preprint arXiv:2109.09610*, 2021.

[7] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[8] Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: A spectral approach. *arXiv preprint arXiv:1706.00764*, 2017.

[9] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Qi Lei, Ajil Jalal, Inderjit S Dhillon, and Alexandros G Dimakis. Inverting deep generative models, one layer at a time. *Advances in Neural Information Processing Systems*, 32:13910–13919, 2019.

[12] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. 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, 2020.

[13] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *International Conference on Machine Learning*, pages 2391–2400. PMLR, 2017.

[14] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing*, pages 293–312. Elsevier, 2019.

[15] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[17] Dave Van Veen, Ajil Jalal, Mahdi Soltanolkotabi, Eric Price, Sriram Vishwanath, and Alexandros G Dimakis. Compressed sensing with deep image prior and learned regularization. *arXiv preprint arXiv:1806.06438*, 2018.

[18] Patricia Vitoria, Joan Sintes, and Coloma Ballester. Semantic image inpainting through improved wasserstein generative adversarial networks. *arXiv preprint arXiv:1812.01071*, 2018.

[19] Jay Whang, Qi Lei, and Alexandros G Dimakis. Compressed sensing with invertible generative models and dependent noise. *arXiv preprint arXiv:2003.08089*, 2020.

[20] Martin J Willemink and Peter B Noël. The evolution of image reconstruction for ct—from filtered back projection to artificial intelligence. *European radiology*, 29(5):2185–2195, 2019.

[21] Jong Chul Ye. Compressed sensing mri: a review from signal processing perspective. *BMC Biomedical Engineering*, 1(1):1–17, 2019.

A More implementation details

We provide more details about our implementations here. We use Adam optimizer [10] in all experiments.

- Learning rate: on denoising and NCS tasks with Gaussian noise, we use learning rate 0.05; on denoising Poisson noise task we use learning rate 0.1
- Maximum iteration number T: on two denoising tasks we use $T = 1200$; on NCS task we use $T = 2000$

Further, Algorithm 1 is a simplified version of PANOM and only highlights the key steps. To improve numeric stability, in practice we apply following tweaks:

- **annealing of** K: we started with $K = 20$ at the beginning of training, and reduced K by one every 20 updates of $\hat{z}$, until it reached a pre-specified minimum threshold 5
- **smoothing** $\gamma_1(\gamma_2)$ **over** K **steps**: we took simple average of K numbers of $\gamma_1(\gamma_2)$ between two consecutive β update steps before computing Δ
- **smoothing over mini-batch samples**: when recovering a mini-batch of images in parallel, we used the same β for all images, and to determine the new β, we took average of Δ from current mini-batch samples
- **decaying** η: we updated η $\quad 0.999\eta$ after each update of β

B More examples on denoising and NCS task

Here we show more results of examples of denoising and NCS tasks.

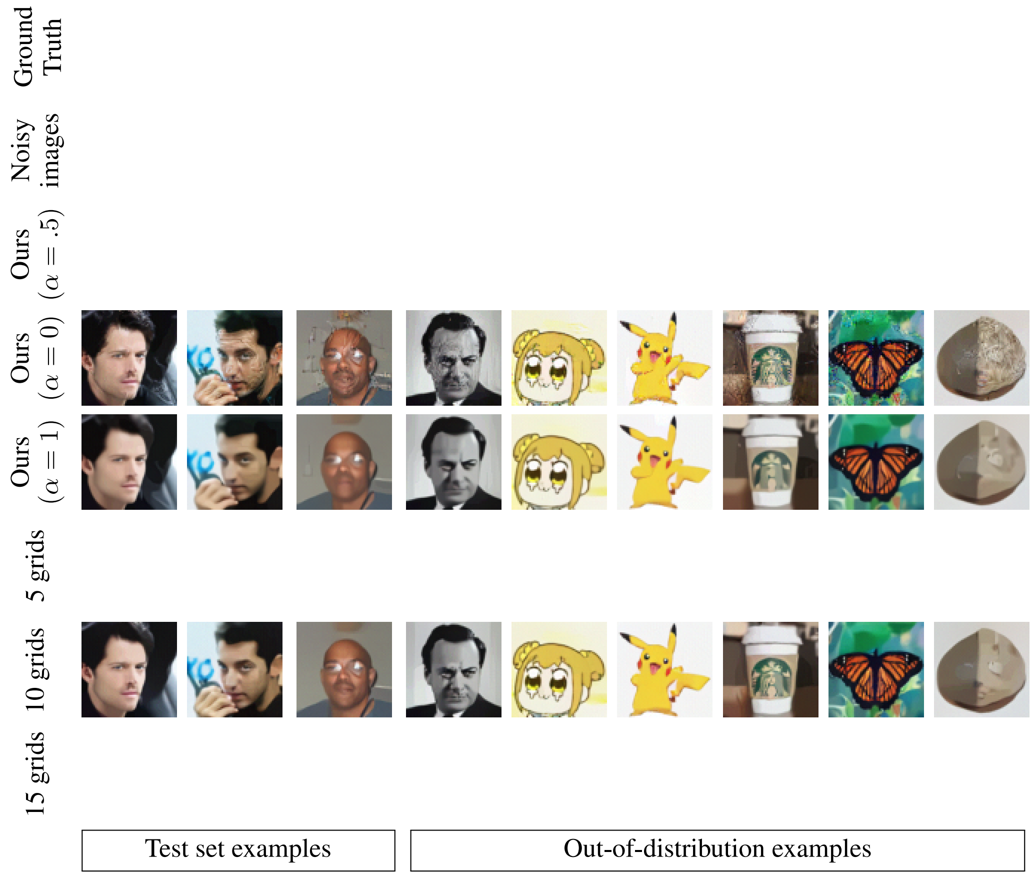

Figure 3: Result of denoising Gaussian noise on CelebA-HQ faces and out-of-distribution images. For grid search we show recovered images achieving the highest PSNR$(\hat{\boldsymbol{x}}, \boldsymbol{x})$

.

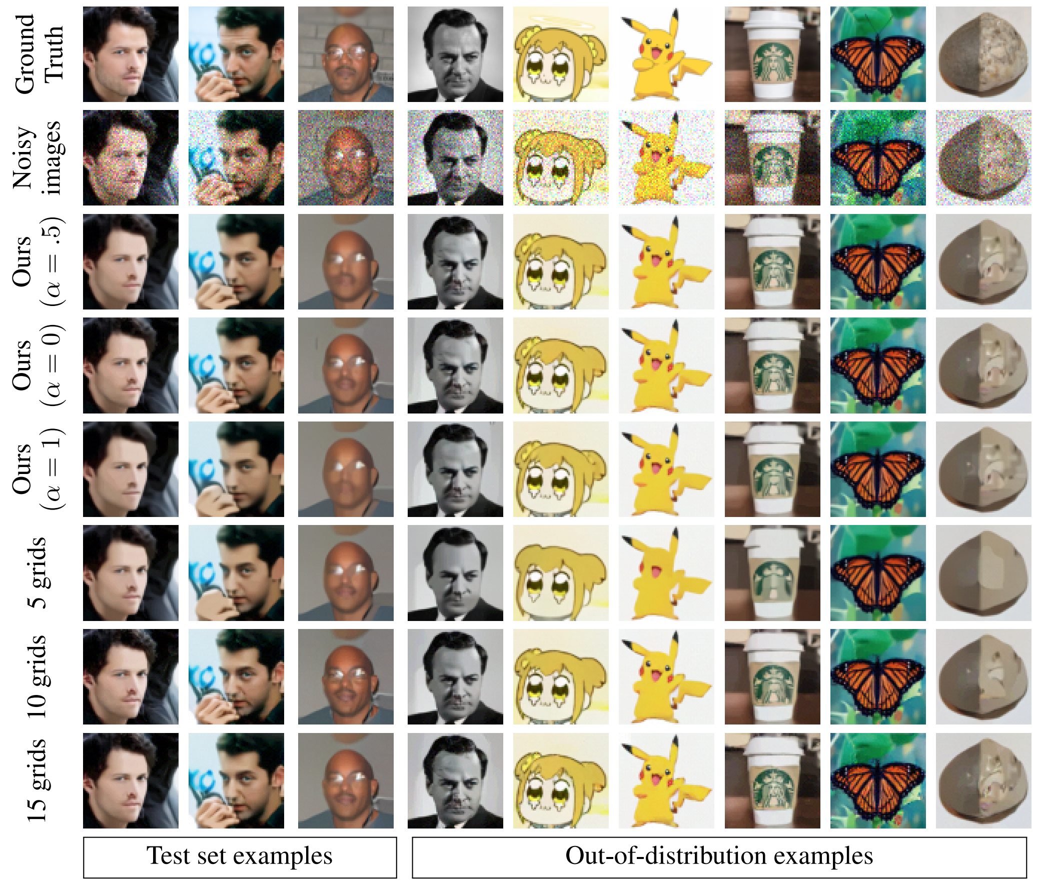

Figure 4: Result of denoising Poisson noise on CelebA-HQ faces and out-of-distribution images. For grid search we show recovered images achieving the highest PSNR$(\hat{\boldsymbol{x}}, \boldsymbol{x}^*)$

.