An Improved Spline-Based Learned Convex Regularizer for Image Recovery

Srijith Nair and Philip Schniter

The Ohio State University

Supported by NSF grant CCF-1955587

2023 Asilomar Conference on SSC

Pacific Grove, CA

Image Recovery

Goal: Recover true image $\boldsymbol{x}_0$ from noisy measurements $\boldsymbol{y}$

- Applications: denoising, deblurring, super-resolution, inpainting, CT, MRI, phase retrieval, de-quantization, photon-limited image recovery, etc.

Variational formulation:

$$\boxed{\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} J(\boldsymbol{x}) \text{ with } J(\boldsymbol{x}) \triangleq d(\boldsymbol{x};\boldsymbol{y}) + \lambda\, r(\boldsymbol{x};\boldsymbol{\theta})}$$

- $d(\cdot;\boldsymbol{y})$ is a data-fidelity term, usually chosen as the negative log-likelihood
- $r(\cdot;\boldsymbol{\theta})$ is a regularizer that incorporates prior information
- λ is a tunable weight
- When $r(\cdot;\boldsymbol{\theta})$ is differentiable, a prototypical recovery algorithm is PGD:

$$\boldsymbol{x}^{(k+1)} = \operatorname{prox}_{\tau^{(k)}d}\left(\boldsymbol{x}^{(k)} - \tau^{(k)}\lambda\nabla r(\boldsymbol{x}^{(k)};\boldsymbol{\theta})\right),$$

where explicit $r(\cdot;\boldsymbol{\theta})$ allows the use of back-tracking line-search to adapt $\tau^{(k)}$

Regularizer design:

- Traditional hand-crafted regularizers use total-variation or wavelet sparsity
- We focus on data-driven regularization that leverages training data $\{\boldsymbol{x}_i\}_{i=1}^n$

Data-driven Regularization

Overall goals:

1. Variational optimization should be tractable (e.g., J convex)
2. The resulting $\widehat{\boldsymbol{x}}$ should be close to the ground truth

Regularizer architecture:

- Many have been proposed, based on, e.g., one-layer networks [1, 2], denoisers [3, 4], autoencoders [5], and deep convolutional networks [6, 7, 8, 9]
- Some enforce convexity in $r(\cdot;\boldsymbol{\theta})$ [2, 6]

Training via the gradient-step framework [8, 10]:

- Main idea: train $I-\nabla r$ as a denoiser $\Leftrightarrow$ train ∇r to recover the noise:

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{GS}}(\boldsymbol{\theta}) \quad \text{for} \quad \mathcal{L}_{\text{GS}}(\boldsymbol{\theta}) \triangleq \textstyle\sum_{i=1}^n \mathbb{E}\left[\left\|\nabla r(\boldsymbol{x}_i+\sigma_{\text{tr}}\boldsymbol{n};\boldsymbol{\theta}) - \sigma_{\text{tr}}\boldsymbol{n}\right\|^2\right]$$

where $\boldsymbol{n}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$ and σ_{tr} is the training-noise level

- $\nabla r(\cdot;\boldsymbol{\theta})$ is computed using automatic differentiation

The Convex Ridge Regularizer

Goujon et al. [2] proposed a 1-layer convex ridge regularizer (CRR):

$$r(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{k=1}^K \sum_{i,j} \psi([\boldsymbol{H}_k * \operatorname{Mat}\{\boldsymbol{x}\}]_{ij};\boldsymbol{c}_k) = \sum_{k=1}^K \boldsymbol{1}^\top\psi(\boldsymbol{W}_k\boldsymbol{x};\boldsymbol{c}_k)$$

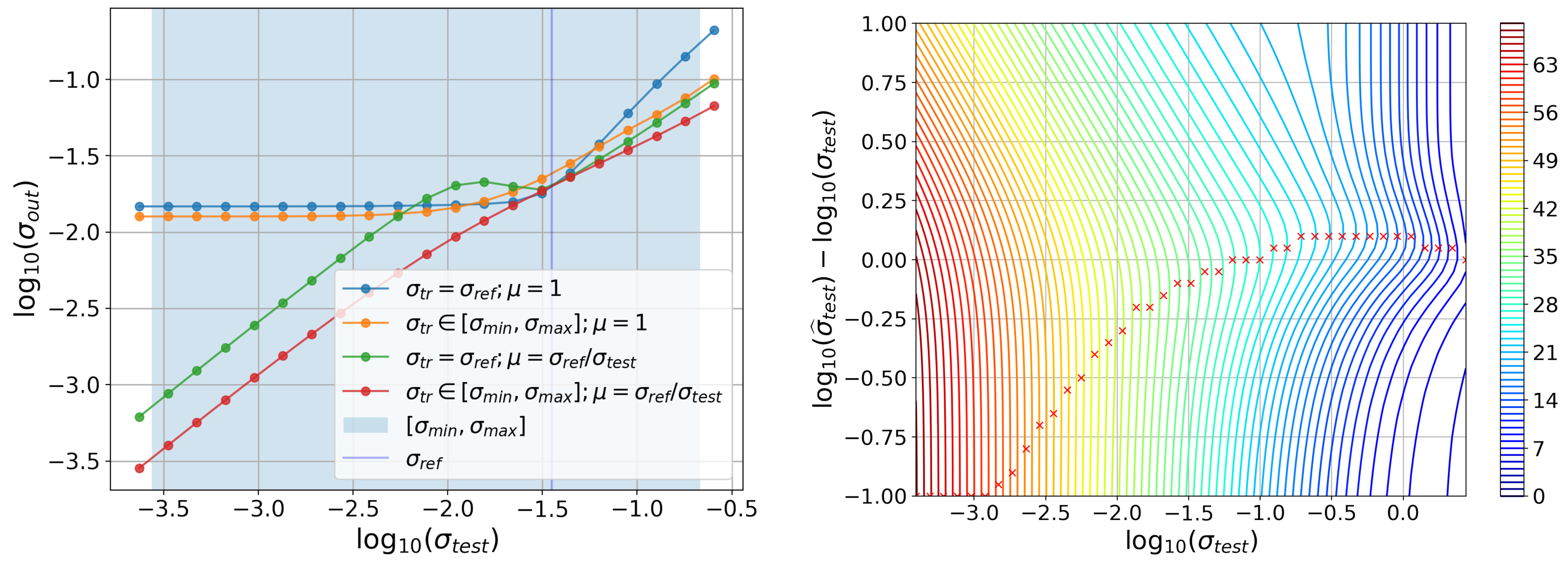
where

- $\boldsymbol{H}_k$ is a convolutional kernel and $*$ denotes convolution
- $\psi(\cdot;\boldsymbol{c}_k):\mathbb{R}\to\mathbb{R}$ is a convex piecewise-quadratic spline

Advantages:

- Recovery performance is comparable to deep networks
- Due to its 1-layer architecture, the CRR is much faster to train!

Improving the Convex Ridge Regularizer



The problem with the CRR:

- The CRR is very sensitive to the noise level (unlike deep nets [11])
  - Blue curve shows CRR trained with $\sigma_{\text{tr}}=\sigma_{\text{ref}}\triangleq 10^{-1.45}$
  - Orange curve shows CRR trained over $\log\sigma_{\text{tr}}\sim\text{Unif}[\log\sigma_{\text{min}},\log\sigma_{\text{max}}]$
- Why? Consider ∇r, which ideally performs noise recovery:

$$\nabla r(\boldsymbol{x};\boldsymbol{\theta}) = \textstyle\sum_{k=1}^K \boldsymbol{W}_k^\top\psi'(\boldsymbol{W}_k\boldsymbol{x};\boldsymbol{c}_k).$$

Difficult for $\psi'(\cdot;\boldsymbol{c}_k)$ to pass noise & block signal when noise level is unknown

- Goujon et al.'s solution [2] was to instead use the scaled regularizer

$$\bar{r}(\boldsymbol{x},\mu;\boldsymbol{\theta}) \triangleq \mu^{-1}r(\mu\boldsymbol{x};\boldsymbol{\theta})$$

unroll for t steps, and jointly tune (λ,μ) for max PSNR($\widehat{\boldsymbol{x}}$) on validation data

Contribution 1: A properly scaled CRR:

- We instead propose to construct a scaled regularizer as

$$\widetilde{r}(\boldsymbol{x},\mu;\boldsymbol{\theta}) \triangleq \mu^{-2}r(\mu\boldsymbol{x};\boldsymbol{\theta})$$

so that $\mu=\sigma_{\text{ref}}/\sigma_{\text{test}}$ calibrates the noise level seen by $\psi'(\cdot;\boldsymbol{c}_k)$:

$$\nabla\widetilde{r}(\boldsymbol{x},\mu;\boldsymbol{\theta}) = \textstyle\sum_{k=1}^K \boldsymbol{W}_k^\top\mu^{-1}\psi'(\mu\boldsymbol{W}_k\boldsymbol{x};\boldsymbol{c}_k).$$

Contribution 2: Robust training:

- Still, training $r(\cdot;\boldsymbol{\theta})$ on fixed $\sigma_{\text{tr}}=\sigma_{\text{ref}}$ and scaling μ at test time leaves much room for improvement (see green curve)
- So we train the genie-scaled regularizer $\widetilde{r}$ over $\ln\sigma_{\text{tr}}\sim\text{Unif}[\ln\sigma_{\text{min}},\ln\sigma_{\text{max}}]$:

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{GS}}(\boldsymbol{\theta}) \quad \text{for} \quad \mathcal{L}_{\text{GS}}(\boldsymbol{\theta}) \triangleq \textstyle\sum_{i=1}^n \mathbb{E}\left[\left\|\nabla\widetilde{r}(\boldsymbol{x}_i+\sigma_{\text{tr}}\boldsymbol{n},\frac{\sigma_{\text{ref}}}{\sigma_{\text{tr}}};\boldsymbol{\theta}) - \sigma_{\text{tr}}\boldsymbol{n}\right\|^2\right]$$

- Then $\widetilde{r}$ works very well under genie scaling (i.e., $\mu=\sigma_{\text{ref}}/\sigma_{\text{test}}$); see red curve

Contribution 3: Scheduling the scaling factor μ:

- In practice, the true noise level σ_{test} is unknown and varies over PGD iterations
- We can't use a $\widehat{\sigma}_{\text{test}}(\boldsymbol{x})$ because the resulting $\widetilde{r}$ would be non-convex in $\boldsymbol{x}$:

$$\widetilde{r}(\boldsymbol{x},\sigma_{\text{ref}}/\widehat{\sigma}_{\text{test}}(\boldsymbol{x});\boldsymbol{\theta})$$

- However, using fixed $\widehat{\sigma}_{\text{test}}<\sigma_{\text{test}}$ does not hurt performance for small σ_{test} (see contour plot above) and ensures convexity
- We thus propose to schedule $\widehat{\sigma}_{\text{test}}$ from big to small over the PGD iterations by linearly scheduling μ from small μ_{init} to big μ_{fin} over $k_{\text{switch}}<k_{\text{total}}$ iterations
- A grid-search over $(\lambda,\mu_{\text{init}},\mu_{\text{fin}},k_{\text{switch}})$ shows that the PSNR and SSIM after k_{total}=200 PGD iterations are relatively insensitive to $(\mu_{\text{init}},\mu_{\text{fin}},k_{\text{switch}})$

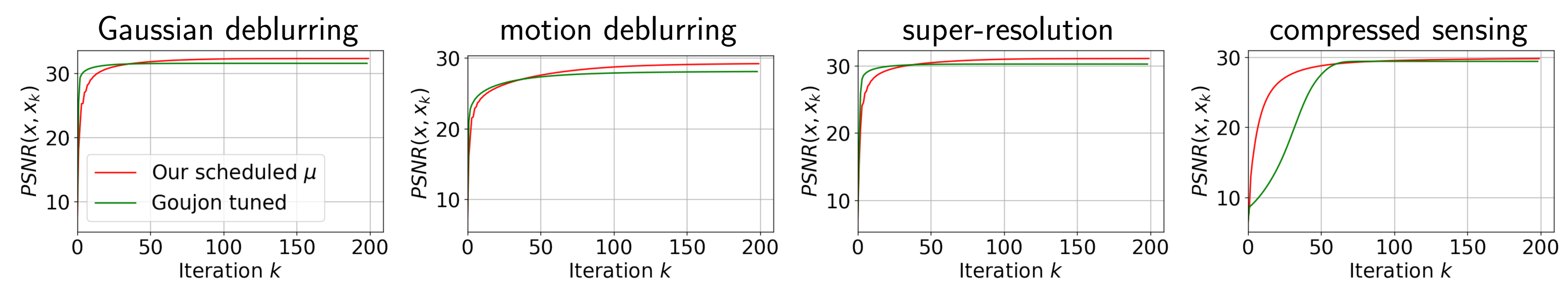
Numerical Experiments

Setup:

- Training data: BSD400: 180x180 grayscale images (350 training, 50 validation)
- Test data: Set68: 320x480 grayscale images (68 testing images)
- CRR: 13x13 kernels and K=32 channels, as in [2]
- Noisy inverse problems with stdv=10/255 AWGN:
  1. Gaussian deblurring: 25x25 Gaussian kernel with stdv=1.6
  2. Motion deblurring: 45° linear-motion blur kernel of size 30x30
  3. Super-resolution: Gaussian blurring followed by 2x vertical & horizontal downsampling
  4. Compressed sensing: Structurally random $\boldsymbol{A}$ as in [12] with compression factor 2
- Methods under test:
  1. Goujon's trained models from [2] with $t=50$ & (λ,μ) optimized over all inverse problems
  2. Our approach with the schedule $\{\mu_{\text{init}}=0.11$ (PSNR$_{\text{init}}=10$), $\mu_{\text{fin}}=834$ (PSNR$_{\text{fin}}=90$), $k_{\text{switch}}=100\}$ and with λ optimized to maximize PSNR on all inverse problems & validation images

Results:

	Gaussian deblurring		motion deblurring		super-resolution		compressed sensing	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Goujon	31.50 $\pm$ 0.43	0.8875	28.03 $\pm$ 0.39	0.7734	30.24 $\pm$ 0.42	0.8542	29.38 $\pm$ 0.31	0.7740
Scheduled PGD	**32.26** $\pm$ 0.42	**0.8968**	**29.09** $\pm$ 0.32	**0.7852**	**31.10** $\pm$ 0.42	**0.8693**	**29.86** $\pm$ 0.44	**0.8043**



References

[1] Roth and Black, "Field of experts," *IJCV*, 2009.
[2] Goujon, Neumayer, Bohra, Ducotterd, and Unser, "A neural-network-based convex regularizer for inverse problems," *IEEE TCI*, 2023.
[3] Romano, Elad, and Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM JIS*, 2017.
[4] Salimans and Ho, "Should EBMs model the energy or the score?," *ICLR Workshop*, 2021.
[5] Li, Schwab, Antholzer, and Haltmeier, "NETT: Solving inverse problems with deep neural networks," *Inv. Prob.*, 2020.
[6] Mukherjee, Dittmer, Shumaylov, Lunz, Öktem, and Schönlieb, "Learned convex regularizers for inverse problems," *arXiv*, 2020.
[7] Kobler, Effland, Kunisch, and Pock, "Total deep variation for linear inverse problems," in *CVPR*, 2020.
[8] Cohen et al., "It has potential: Gradient-driven denoisers for convergent solutions to inverse problems," *NeurIPS*, 2021.
[9] Zach, Knoll, and Pock, "Stable deep MRI reconstruction using generative priors," *arXiv*, 2022.
[10] Hurault, Leclaire, Papadakis, "Proximal denoiser for convergent plug-and-play optimization with nonconvex regularization," *ICML*, 2022.
[11] Mohan et al., "Robust and interpretable blind image denoising via bias-free convolutional neural networks," *ICLR*, 2020.
[12] Do, Gan, Nguyen, and Tran, "Fast and efficient compressive sensing using structurally random matrices," *IEEE TSP*, 2012.