Solving Inverse Problems using Diffusion with Iterative Colored Renoising

Phil Schniter

(Joint work with Matt Bendel, Saurav Shastri, and Rizwan Ahmad)

The Ohio State University

Supported in part by the NIH under grant R01-EB029957

Asilomar Conference on Signals, Systems, and Computers
October 2025

Imaging inverse problems

Imaging inverse problems:

- Unknown image $\boldsymbol{x}_0$
- Masked/noisy/distorted measurements $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}_0)$
- Examples: denoising, deblurring, inpainting, super-resolution, MRI, CT, phase-retrieval etc.

Challenges:

- Typically ill-posed: many hypotheses of $\boldsymbol{x}_0$ form good explanations of $\boldsymbol{y}$

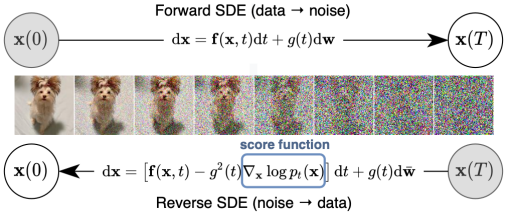
Typical goals:

- Estimate unknown $\boldsymbol{x}_0$
- Sample from the posterior distribution $p(\boldsymbol{x}_0|\boldsymbol{y})$

Diffusion methods

- Diffusion methods are powerful ways to sample from a complicated distribution $p(\boldsymbol{x})$

- The forward process gradually adds noise to $\boldsymbol{x}(0) \sim p(\boldsymbol{x})$. The reverse process starts with pure noise $\boldsymbol{x}(T)$ and gradually denoises, eventually generating a sample from $p(\boldsymbol{x})$



- We'll assume a variance-exploding (VE) discretization, where discrete step $t \in \{1, \ldots, T\}$ provides

$$\boldsymbol{x}_t = \boldsymbol{x}_0 + \sigma_t \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$$

- Although the reverse process is usually written using the score function $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)$, it can also be written using the MMSE denoiser $\mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t\}$ via Tweedie's rule

$$\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t) = \frac{\mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t\} - \boldsymbol{x}_t}{\sigma_t^2} \quad \Leftrightarrow \quad \mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t\} = \boldsymbol{x}_t + \sigma_t^2 \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)$$

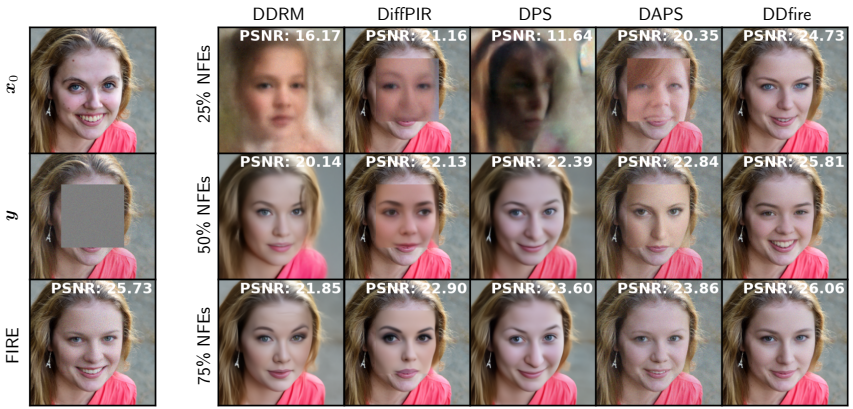
Solving inverse problems with diffusion

- Standard diffusion samples from $p(\boldsymbol{x}_0)$ by repeatedly invoking $\mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t\}$ or $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)$
- With an inverse problem, we can use diffusion to sample from $p(\boldsymbol{x}_0|\boldsymbol{y})$ by repeatedly invoking $\mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}\}$ or $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t|\boldsymbol{y})$
- But $\mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}\}$ and $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t|\boldsymbol{y})$ are intractable. Thus approximations have been proposed using some combination of a pretrained denoiser/score-function and the likelihood function $p(\boldsymbol{y}|\boldsymbol{x}_0)$
- Popular methods include DDRM[1], DPS[2], DDNM[3], ΠGDM[4], DiffPIR[5], GPnP[6], PnP-SGS[7], DDS[8], RED-diff[9], SNORE[10], PnP-DM[11], DPnP[12], DAPS[13], etc.

[1]Kawar et al'22, [2]Chung et al'23, [3]Wang et al'23, [4]Song et al'23, [5]Zhu et al'23, [6]Bouman et al'23, [7]Coeurdoux et al'23, [8]Chung et al'24, [9]Mardani et al'24, [10]Renaud et al'24, [11]Wu et al'24, [12]Xu et al'24, [13]Zhang et al'25

The main challenge with diffusion for inverse problems

- The main challenge is approximating $\mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t, \boldsymbol{y}\}$ well in a computationally efficient manner
- Below we show visual examples of typical approximations, along with their corresponding PSNRs



A new stochastic plug-and-play (PnP) algorithm

- We propose an iterative approximation of $\mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t,\boldsymbol{y}\}$ that we call Fast Iterative REnoising (FIRE)
- For linear inverse problems $\boldsymbol{y}=\boldsymbol{A}\boldsymbol{x}_0+\boldsymbol{w}$ with $\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{0},\sigma_w^2\boldsymbol{I})$, FIRE iterates the following steps after initializing $\boldsymbol{r}\leftarrow\boldsymbol{x}_{\text{init}}$ and $\sigma_{\text{in}}\leftarrow\sigma_{\text{init}}$:

  1. $\overline{\boldsymbol{x}}\leftarrow\text{Denoise}(\boldsymbol{r};\sigma_{\text{in}}^2)$, $\quad\widehat{\sigma}_{\text{out}}^2\leftarrow$ unbiased estimate of $\mathbb{E}\{\|\overline{\boldsymbol{x}}-\boldsymbol{x}_0\|^2\}/d$
  2. $\widehat{\boldsymbol{x}}\leftarrow\arg\min_{\boldsymbol{x}}\frac{1}{\sigma_w^2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|^2+\frac{1}{\widehat{\sigma}_{\text{out}}^2}\|\overline{\boldsymbol{x}}-\boldsymbol{x}\|^2$ ... half-quadratic splitting
  3. $\sigma_{\text{in}}^2\leftarrow\sigma_{\text{in}}^2/\rho$ for some $\rho>1$ ... decrease denoiser input-error variance
  4. $\boldsymbol{r}\leftarrow\widehat{\boldsymbol{x}}+\boldsymbol{c}$ ... renoise with colored $\boldsymbol{c}$ that gives $\text{Cov}\{\boldsymbol{r}-\boldsymbol{x}_0\}=\sigma_{\text{in}}^2\boldsymbol{I}$

- Like DPIR[1], it's a form of HQS-PnP with a decreasing σ_{in}^2, and like RED-diff and SNORE, it injects noise into PnP
- But unlike existing methods, it injects colored noise shaped so that the denoiser sees AWGN ... consistent with how the denoiser was trained!

[1] Zhang et al'21

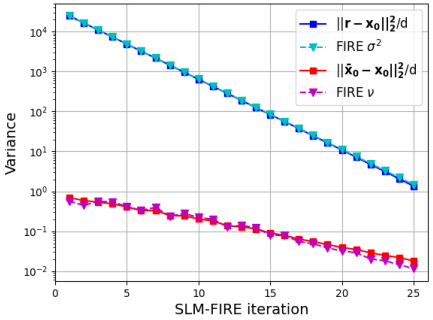
FIRE: denoiser output-error variance estimation

To estimate σ^2_{out}, the error variance of the denoiser output $\overline{\boldsymbol{x}}$ ...

- We assume $\overline{\boldsymbol{x}} = \boldsymbol{x}_0 + \sigma_{\text{out}}\boldsymbol{e}$, where $\text{Cov}\{\boldsymbol{e}\} = \boldsymbol{I}$
  (like in DDS, DiffPIR, or any prox-based PnP algorithm)
- Then, since $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_0 + \sigma_w\boldsymbol{w}$, we have

$$\begin{aligned}\mathbb{E}\{\|\boldsymbol{y} - \boldsymbol{A}\overline{\boldsymbol{x}}\|^2\} &= \mathbb{E}\{\|\boldsymbol{A}\boldsymbol{x}_0 + \sigma_w\boldsymbol{w} - \boldsymbol{A}\boldsymbol{x}_0 - \sigma_{\text{out}}\boldsymbol{A}\boldsymbol{e}\|^2\} \\ &= m\sigma_w^2 + \sigma^2_{\text{out}}\|\boldsymbol{A}\|_F^2\end{aligned}$$

- Rearranging gives $\sigma^2_{\text{out}} = \mathbb{E}\Big\{\underbrace{\frac{\|\boldsymbol{y} - \boldsymbol{A}\overline{\boldsymbol{x}}\|^2 - m\sigma_w^2}{\|\boldsymbol{A}\|_F^2}}_{\triangleq\, \widehat{\sigma}^2_{\text{out}}}\Big\}$
- Thus $\widehat{\sigma}^2_{\text{out}}$ is an unbiased estimate of σ^2_{out}



FIRE: colored-noise shaping

- The HQS estimate is

$$\widehat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \tfrac{1}{\sigma_w^2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|^2 + \tfrac{1}{\widehat{\sigma}_{\text{out}}^2}\|\overline{\boldsymbol{x}}-\boldsymbol{x}\|^2 = \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} + \tfrac{\sigma_w^2}{\widehat{\sigma}_{\text{out}}^2}\boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{y} + \tfrac{\sigma_w^2}{\widehat{\sigma}_{\text{out}}^2}\overline{\boldsymbol{x}}\right)$$

- Assuming $\widehat{\sigma}_{\text{out}} = \sigma_{\text{out}}$, we can write the HQS error as

$$\widehat{\boldsymbol{x}} - \boldsymbol{x}_0 = \left(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} + \tfrac{\sigma_w^2}{\sigma_{\text{out}}^2}\boldsymbol{I}\right)^{-1}\left(\sigma_w\boldsymbol{A}^{\mathsf{T}}\boldsymbol{w} + \tfrac{\sigma_w^2}{\sigma_{\text{out}}}\boldsymbol{e}\right)$$

- Assuming $\mathrm{Cov}\{\boldsymbol{e}\} = \boldsymbol{I}$ as before, the denoiser-output error covariance is

$$\mathrm{Cov}\{\widehat{\boldsymbol{x}} - \boldsymbol{x}_0\} = \left(\tfrac{1}{\sigma_w^2}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} + \tfrac{1}{\sigma_{\text{out}}^2}\boldsymbol{I}\right)^{-1} \triangleq \boldsymbol{\Sigma}$$

- For the denoiser input $\boldsymbol{r} = \widehat{\boldsymbol{x}} + \boldsymbol{c}$ to have error covariance $\sigma_{\text{in}}^2\boldsymbol{I}$, we need to shape the noise $\boldsymbol{c}$ s.t.

$$\mathrm{Cov}\{\boldsymbol{c}\} = \sigma_{\text{in}}^2\boldsymbol{I} - \boldsymbol{\Sigma},$$

  which can be easily done with an SVD, or closely approximated if an SVD isn't available

FIRE: analysis

Theorem

Suppose that, for any denoiser input of the form

$$\boldsymbol{r} = \boldsymbol{x}_0 + \sigma_{\text{in}}\boldsymbol{\epsilon} \text{ with } \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}),$$

the denoiser output $\overline{\boldsymbol{x}}$ *obeys*

$$\overline{\boldsymbol{x}} = \boldsymbol{x}_0 + \sigma_{\text{out}}\boldsymbol{e} \quad \text{with} \quad \boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}),\ \boldsymbol{e} \perp\!\!\!\perp \boldsymbol{w},\ \text{and known } \sigma_{\text{out}} < \sigma_{\text{in}}.$$

Then, if initialized using $\boldsymbol{r}_{\text{init}} = \boldsymbol{x}_0 + \sigma_{\text{init}}\boldsymbol{\epsilon}$ *with* $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ *and arbitrarily large but finite* σ_{init}*, there exists a noise-scheduling constant* $\rho > 1$ *under which the FIRE's* $\widehat{\boldsymbol{x}}$ *estimate converges to the true* $\boldsymbol{x}_0$*.*

For the proof, see Bendel, Shastri, Ahmad, and S, "Solving Inverse Problems using Diffusion with Iterative Colored Renoising" *Trans. Machine Learning Research*, 8/2025.

DDfire: Putting the FIRE into diffusion

- FIRE can be used to approximate $\mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_t,\boldsymbol{y}\}$ in any diffusion reverse process
- We apply it to DDIM[1], which is based on the forward model

$$\boldsymbol{x}_k = \boldsymbol{x}_0 + \sigma_k\boldsymbol{\varepsilon}_k, \quad \boldsymbol{\varepsilon}_k \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I}), \quad k = 1,\dots,K,$$

for some choice of K and $\{\sigma_k\}_{k=1}^K$ and uses the reverse process

$$\boldsymbol{x}_{k-1} = h_k\boldsymbol{x}_k + g_k\,\mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_k,\boldsymbol{y}\} + \varsigma_k\boldsymbol{n}_k$$

$$\varsigma_k = \eta_{\text{ddim}}\sqrt{\tfrac{\sigma_{k-1}^2(\sigma_k^2-\sigma_{k-1}^2)}{\sigma_k^2}}, \quad h_k = \sqrt{\tfrac{\sigma_{k-1}^2-\varsigma_k^2}{\sigma_k^2}}, \quad g_k = 1-h_k$$

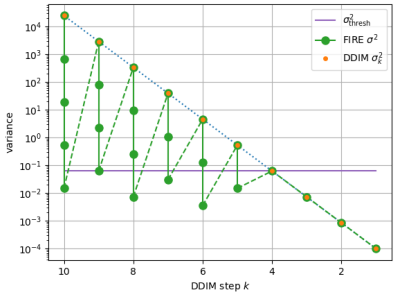
for some choice of $\eta_{\text{ddim}} \in [0,1]$

- We adopt the typical geometric variance schedule $\sigma_k^2 = \sigma_{\text{min}}^2\left(\frac{\sigma_{\text{max}}^2}{\sigma_{\text{min}}^2}\right)^{\frac{k-1}{K-1}}$

[1] Song et al'21

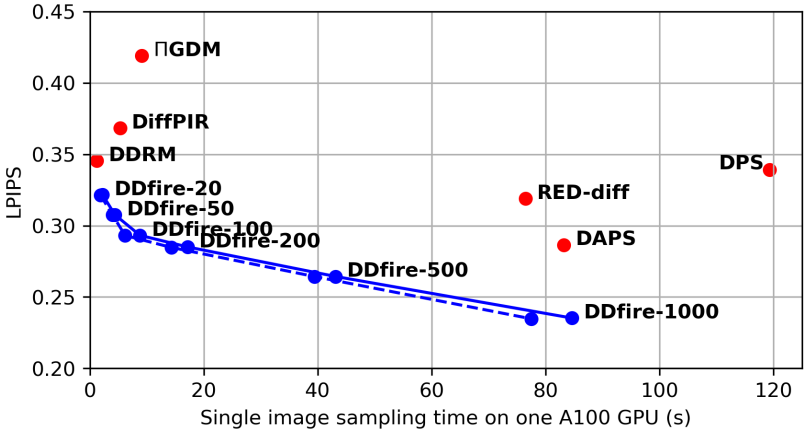
DDfire: Scheduling the number of FIRE iterations (across DDIM steps)

- Because FIRE uses multiple NFEs per $\mathbb{E}\{\boldsymbol{x}_0|\boldsymbol{x}_k,\boldsymbol{y}\}$, we must schedule the number of FIRE iterations to meet a given total-NFE budget
- Basically, we run FIRE until σ^2_{in} falls below a threshold σ^2_{thresh}
  - This and the # total-NFEs determine the FIRE parameter ρ
- We set σ^2_{thresh} equal to some DDIM variance $\sigma^2_{k_{\text{thresh}}}$, where $k_{\text{thresh}} \in \mathbb{N}$ is a tuning parameter
  - Thus DDfire has only two tuning parameters: k_{thresh} & K
- The resulting "DDfire" outperforms many state-of-the-art diffusion methods over a wide range of total-NFE budgets



Runtime comparison

Noisy Gaussian deblurring on ImageNet with both CG (solid) and SVD (dashed) versions of DDFire:



Noisy FFHQ results

- Results on 256×256 FFHQ faces with measurement noise $\sigma_w = 0.05$:

	Inpaint (box)			Deblur (Gaussian)			Deblur (Motion)			4× Super-resolution		
Model	PSNR↑	LPIPS↓	FID↓	PSNR↑	LPIPS↓	FID↓	PSNR↑	LPIPS↓	FID↓	PSNR↑	LPIPS↓	FID↓
DDRM	21.71	0.1551	40.61	25.35	0.2223	51.70	-	-	-	**27.32**	0.1864	45.82
DiffPIR	22.43	0.1883	31.98	24.56	0.2394	34.82	26.91	0.1952	26.67	24.89	0.2486	32.33
ΠGDM	21.41	0.2009	44.41	23.66	0.2525	45.34	25.14	0.2082	41.95	24.40	0.2520	51.41
DDS	20.28	0.1481	30.23	26.74	0.1648	25.47	27.52	0.1503	27.59	26.71	0.1852	27.09
DPS	22.54	0.1368	35.69	25.70	0.1774	25.18	26.74	0.1655	27.17	26.30	0.1850	27.38
RED-diff	23.58	0.1883	48.86	26.99	0.2081	38.82	16.47	0.5074	128.68	25.61	0.3569	70.86
DAPS	23.61	0.1415	31.51	26.97	0.1827	31.10	27.13	0.1718	30.74	26.91	0.1885	30.83
DDfire	**24.75**	**0.1101**	**25.26**	**27.10**	**0.1533**	**24.97**	**28.14**	**0.1374**	**26.12**	27.13	**0.1650**	**25.73**

- DDfire outperforms the competitors in 11 of the 12 cases

Noisy ImageNet results

- Results on 256×256 ImageNet with measurement noise $\sigma_w = 0.05$:

	Inpaint (box)			Deblur (Gaussian)			Deblur (Motion)			4× Super-resolution		
Model	PSNR↑	LPIPS↓	FID↓	PSNR↑	LPIPS↓	FID↓	PSNR↑	LPIPS↓	FID↓	PSNR↑	LPIPS↓	FID↓
DDRM	18.24	0.2423	67.47	22.56	0.3454	68.78	-	-	-	**24.49**	0.2777	64.68
DiffPIR	18.03	0.2860	65.55	21.31	0.3683	56.35	24.36	0.2888	54.11	23.31	0.3383	63.48
ΠGDM	17.69	0.3303	86.36	20.87	0.4191	75.43	22.15	0.3591	70.91	21.25	0.4149	78.57
DDS	16.68	0.2222	63.07	23.14	0.2684	50.84	23.34	0.2674	50.08	23.03	0.3011	52.13
DPS	18.23	0.2314	59.10	21.30	0.3393	50.46	21.77	0.3307	80.27	23.38	0.2904	49.86
RED-diff	18.95	0.2909	108.88	23.45	0.3190	65.65	15.21	0.5647	198.74	22.99	0.3858	83.06
DAPS	19.99	0.2199	61.53	**23.91**	0.2863	56.87	24.58	0.2722	54.83	24.04	0.2729	55.54
DDfire	**20.39**	**0.1915**	**55.54**	23.71	**0.2353**	**50.05**	**24.59**	**0.2314**	**49.25**	23.58	**0.2629**	**49.67**

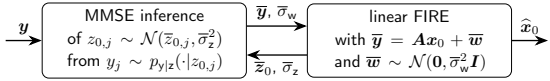
- DDfire outperforms the competitors in 10 of the 12 cases

Extension to generalized-linear inverse problems

- To handle phase retrieval, dequantization, Poisson regression, or linear inverse problems with additive non-Gaussian noise, we extend FIRE & DDfire to the generalized linear model (GLM):

$$\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{z}_0) = \prod_{j=1}^{m} p_{\mathsf{y}|\mathsf{z}}(y_j|z_{0,j}) \quad \text{with hidden} \quad \boldsymbol{z}_0 \triangleq \boldsymbol{A}\boldsymbol{x}_0$$

- We do this using expectation-propagation iterations[1] between linear FIRE and a scalar MMSE inference stage:



- DDfire achieves state-of-the-art performance for both OSF and CDP phase retrieval:

	FFHQ OSF			FFHQ CDP			ImageNet OSF			ImageNet CDP		
Model	PSNR↑	LPIPS↓	FID↓	PSNR↑	LPIPS↓	FID↓	PSNR↑	LPIPS↓	FID↓	PSNR↑	LPIPS↓	FID↓
HIO	23.66	0.4706	130.6	17.59	0.5430	84.87	21.53	0.4584	111.5	17.65	0.4527	69.01
DOLPH	14.73	0.7220	389.9	25.76	0.1686	32.93	14.33	0.6844	258.6	26.15	0.2256	48.26
DPS	23.63	0.2908	53.91	29.19	0.1394	27.87	16.69	0.5314	140.2	27.21	0.1799	45.98
RED-diff	25.47	0.2828	65.74	28.75	0.1734	28.87	18.41	0.4385	108.0	26.87	0.2173	48.12
DAPS	24.10	0.2891	57.73	28.26	0.1927	34.97	17.62	0.4934	121.8	21.76	0.3332	54.17
prDeep	30.90	0.1132	31.51	19.24	0.4183	59.44	26.16	0.1977	57.86	19.51	0.3934	59.34
DDfire	**33.56**	**0.0691**	**28.94**	**30.16**	**0.1186**	**23.30**	**29.93**	**0.1640**	**56.99**	**28.91**	**0.1377**	**44.35**

[1] Meng et al'18

Conclusion

- For linear inverse problems under Gaussian noise, we propose a new stochastic PnP algorithm called Fast Iterative REnoising (FIRE)
- FIRE uses decreasing-variance HQS-PnP with two key extensions: ...
  1. it injects colored noise in order to whiten the denoiser-input error, and
  2. it estimates the denoiser-output error variance in an unbiased manner
- We extended FIRE to generalized-linear inverse problems using the EM alg
- We plugged FIRE into the DDIM diffusion reverse process, giving DDfire
- Experiments with FFHQ and ImageNet images, and both linear (box inpainting, super-resolution, motion deblurring, Gaussian deblurring) and phase-retrieval (OSF, CDP) measurements, suggest that DDfire gives state-of-the-art accuracy and runtime