BLUR LABS

cnrs

ENS DE LYON

Reconstruct-Anything Model

ICLR 2026

Joint work with

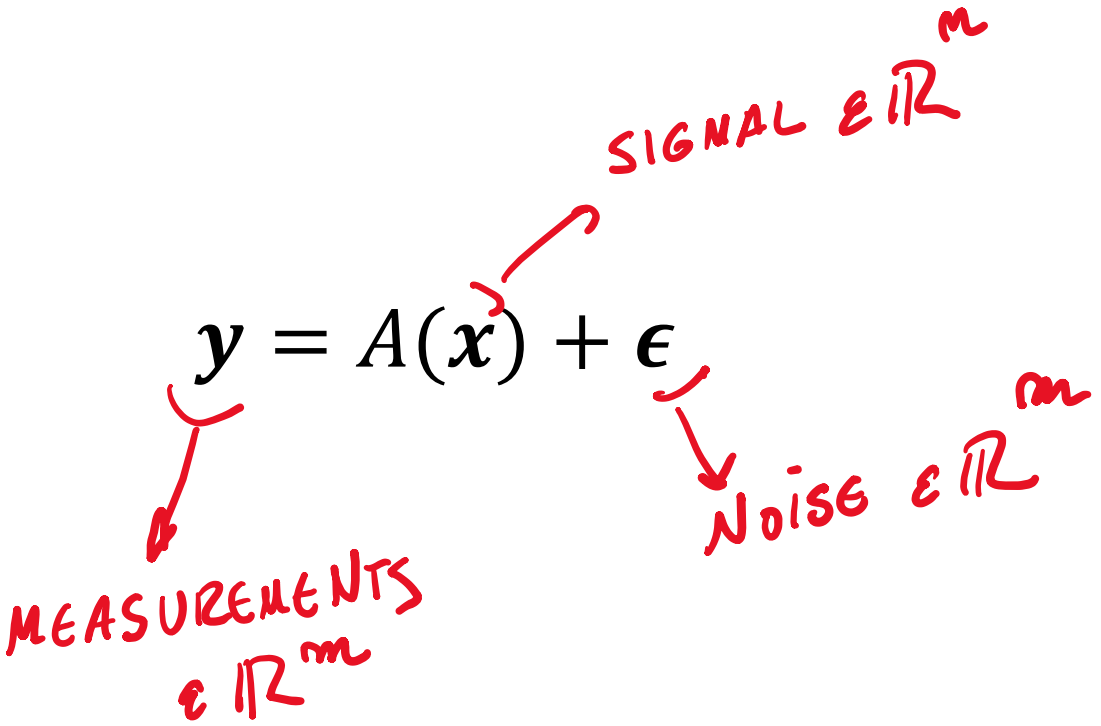
Matthieu Terris

Samuel Hurault

Maxime Song

Linear Inverse Problems

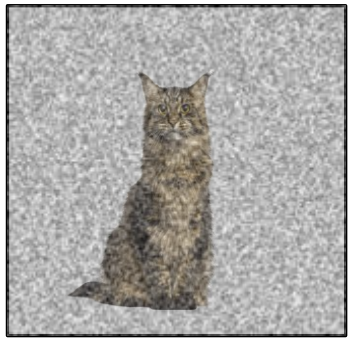
Goal: recover signal $\boldsymbol{x}$ from $\boldsymbol{y}$

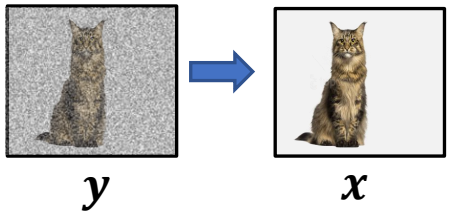


Examples

Image denoising

- A = identity

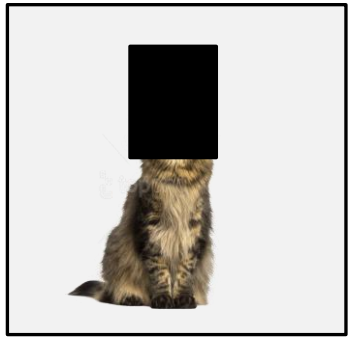


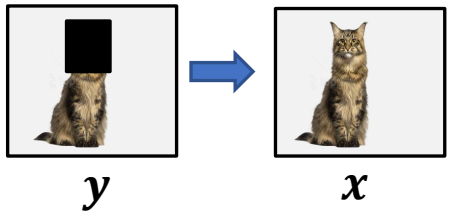


y x

Image inpainting

- A = diagonal matrix with 1's and 0s.

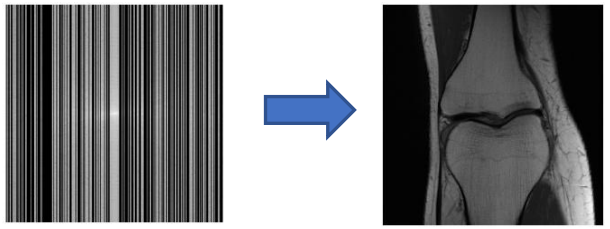




y x

Magnetic resonance imaging

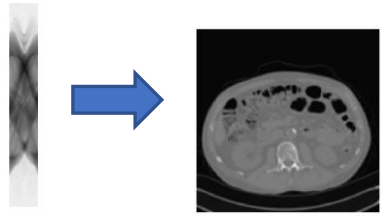
- A = subset of Fourier modes ($k-\text{space}$)



y x

Computed tomography

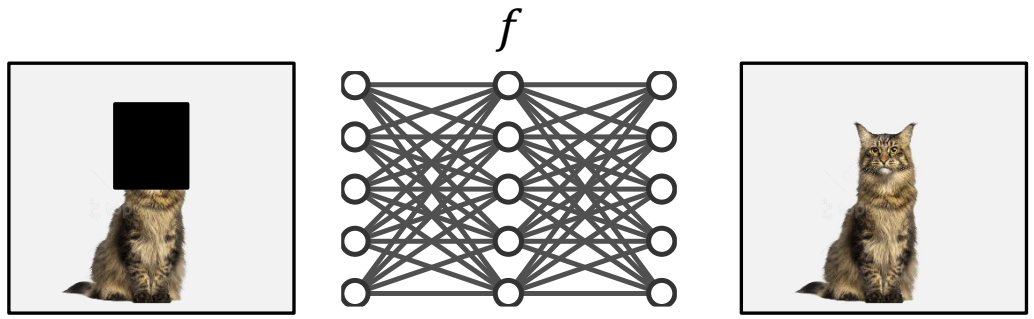
- A = 1D projections (sinograms)



y x

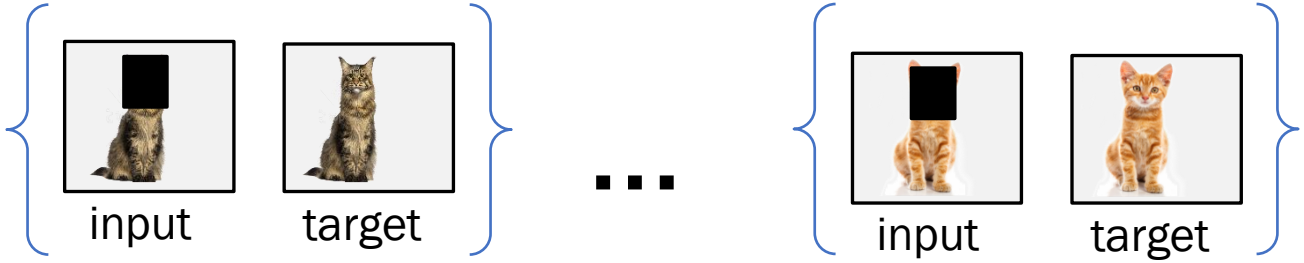
End2End Learning

Idea: use training pairs of signals and measurements to directly learn the inversion function



$$\underset{f}{\operatorname{argmin}}\ \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\|\boldsymbol{x} - f(\boldsymbol{y}, \boldsymbol{A})\|^2$$

supervised dataset



Benefits of End2End

- **Advantages**
  - Outperform plug-and-play/diffusion models on training task
  - Much faster inference (doesn’t require multiple iterations as diffusion/PnP)

- **Disadvantages**:
  - Trained for a specific task, doesn’t generalize to different tasks
  - Requires a large supervised dataset specific to application
  - It is a black-box, no notion of uncertainty

Taxonomy of End2End

- **Artifact removal networks** (Jin et al., 2017):

$$f(\boldsymbol{y}, A) = \phi(A^{\mathsf{T}}\boldsymbol{y})$$

where ϕ is an image-to-image networks, such as a U-Net.

- **Unrolled neural networks** (Gregor and LeCun, 2010)**:** e.g. proximal gradient descent, we have $f(\boldsymbol{y}, A) = \boldsymbol{x}^K$ where

$$\boldsymbol{x}^{k+1} = \phi_k\left(\boldsymbol{x}^k - A^{\mathsf{T}}A\boldsymbol{x}^k + A^{\mathsf{T}}\boldsymbol{y}\right) \text{ for } k = 1, \ldots, K$$

where ϕ_k are image-to-image networks.

Typically 1 to 3 dB better

How to design f?

While unrolled typically obtains SOTA results

- Not enough iterates to ‘converge’
- Convergence guarantees lost when using expressive $\phi_1 \ \ldots \ \phi_k$

Kimmel: “*DL is a dark monster covered with mirrors. Everyone sees his reflection in it …*”, Donoho: *“… these mirrors are taken from Cinderella's story, telling each that they are the most beautiful*”

- If we forget optimization perspective:

main difference between unrolling and artifact removal

is ***backpropagation through*** $\boldsymbol{A^\top A}$

Towards a new model

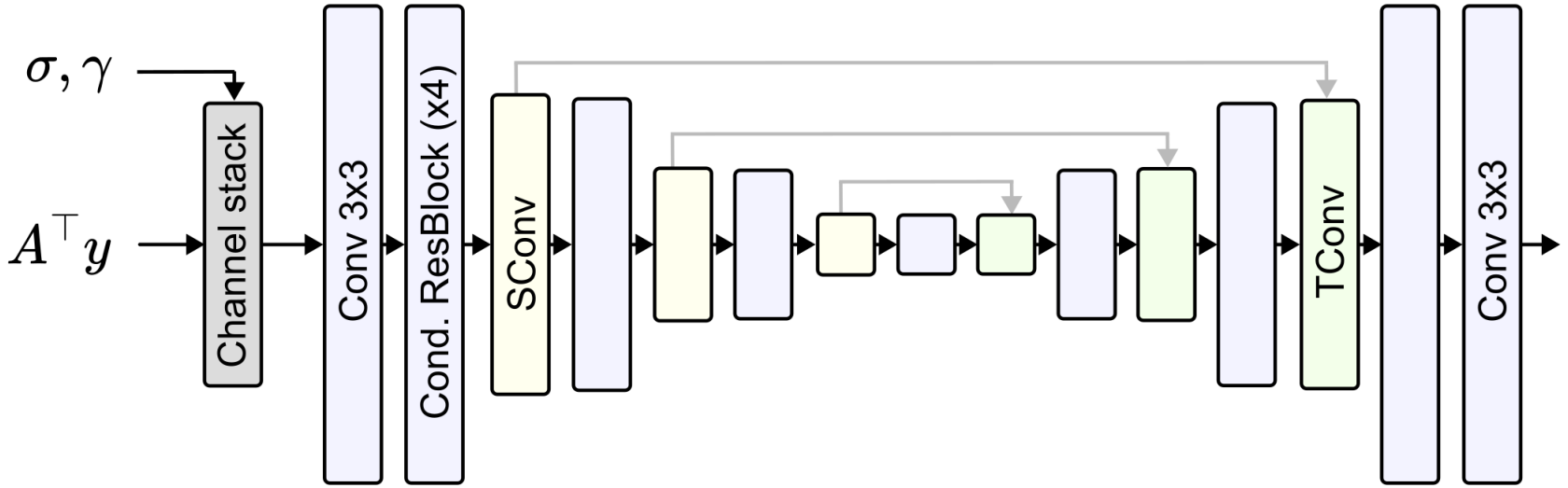
How can we come up with a better end-to-end model?

- **Hypothesis 1**: A standard U-Net architecture should be enough
- **Hypothesis 2**: Backpropagating through $A^\top A$ boosts performance
- **Hypothesis 3**: The network should generalize to multiple A if trained on a family [Gossard & Weiss, 2022]

Towards a new model

- **Hypothesis 1**: A standard U-Net architecture should be enough

- State-of-the-art architecture used for denoising, super-resolution, segmentation, etc.
- We take the **DRUNet architecture** (Zhang et al., 2021) as backbone
- We can start with pretrained denoising weights



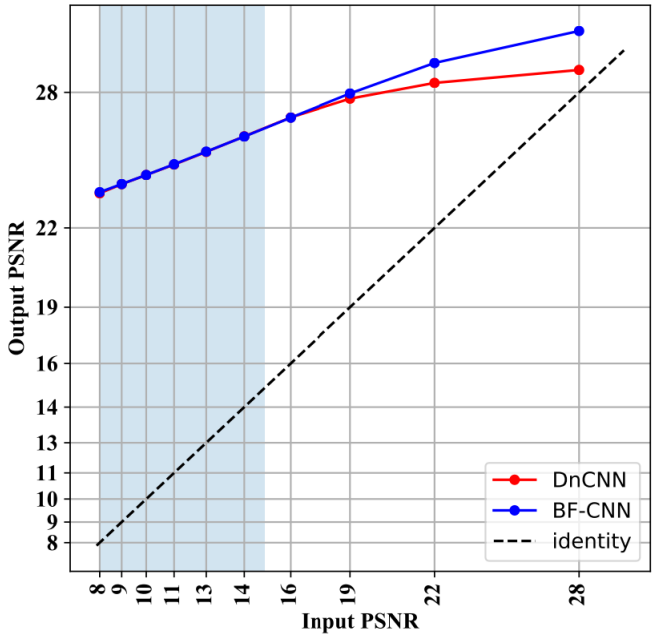
Towards a new model

- **Hypothesis 1**: A standard U-Net architecture should be enough
- The network is **scale equivariant** (Mohan et al., 2020), (Herbreteau et al., 2023)
- **Poisson-Gaussian** noise model: $\boldsymbol{y} = \gamma \mathcal{P}\left(\frac{A\boldsymbol{x}}{\gamma}\right) + \sigma\boldsymbol{\epsilon}$
- Scaling by $\alpha > 0$ gives $\alpha\boldsymbol{y} = \alpha\gamma \mathcal{P}\left(\frac{A\alpha\boldsymbol{x}}{\alpha\gamma}\right) + \alpha\sigma\boldsymbol{\epsilon}$

Thus we want:

$$f(\alpha\boldsymbol{y}, \alpha\sigma, \alpha\gamma) = \alpha f(\boldsymbol{y}, \sigma, \gamma)$$

which is automatically verified by the DRUNet noise conditioning

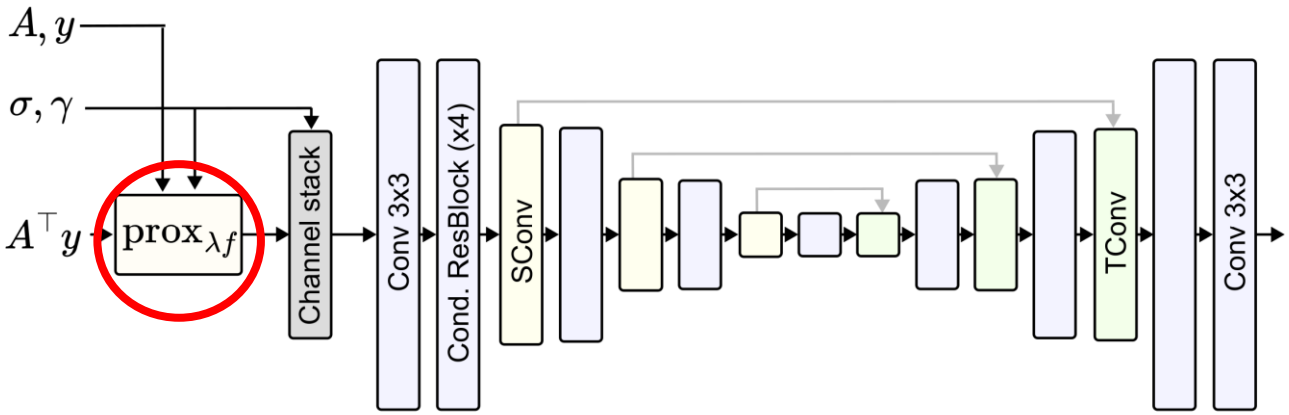


Towards a new model

- **Hypothesis 1**: A standard U-Net architecture should be enough
- We map measurements y to an input of DRUNet as:

$$\text{input} = \ \text{argmin}\ ||\boldsymbol{y} - A\boldsymbol{x}||^2 + \gamma||\boldsymbol{x} - A^\top \boldsymbol{y}||^2$$

where $\gamma > 0$ is a (learnable) parameter which interpolates between $A^\dagger \boldsymbol{y}$ (variance) and $A^\top \boldsymbol{y}$ (bias)



Towards a new model

- **Hypothesis 1**: A standard U-Net architecture should be enough

- We validate empirically that a prox input improves performance, with *almost no change in the number of learnable parameters*

Configuration	PSNR (dB)	#Params
base (DRUNet backbone)	25.83	32.6M
base + prox	26.64	32.6M

Towards a new model

- **Hypothesis 2**: Backpropagating through $A^\top A$ boosts performance

Unrolled networks incorporate A in two ways

1) *Gradient step*: $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \eta\, A^\top A\, \boldsymbol{x}^k + A^\top \boldsymbol{y}$

2) *Proximal step*: $\boldsymbol{x}^{k+1} = \left(\eta\, A^\top A + I\right)^{-1} (\eta\, A^\top \boldsymbol{y} + \boldsymbol{x}^k)$

Prox. is typically computed via conjugate gradient, a Krylov subspace method,

$$\text{where } x^{k+1} = \text{span}\left\{\left(\eta\, A^\top A + I\right)^i \left(\eta\, A^\top \boldsymbol{y} + \boldsymbol{x}^k\right)\right\}_{i=1:n}$$

$$= \sum_i^N \alpha_i \left(A^\top A\right)^i \boldsymbol{x}^k + \beta_i \left(A^\top A\right)^i A^\top \boldsymbol{y}$$

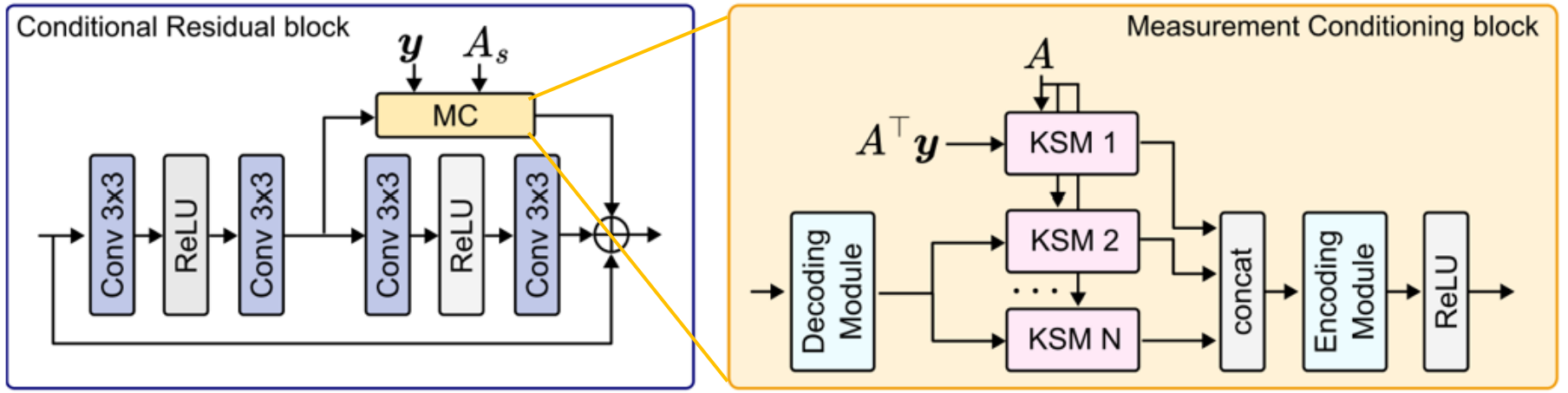
for some scalars $\{(\alpha_i, \beta_i)\}_{i=1:n}$

Towards a new model

- **Hypothesis 2**: Backpropagating through $A^\top A$ boosts performance

We generalize the grad and prox steps via a Krylov embedding layer

of order N, written as $\phi(\text{concat}\left[\left(A^\top A\right)^i \boldsymbol{x}^k, \left(A^\top A\right)^i A^\top \boldsymbol{y}\right]_{i=1:N})$ with ϕ being some conv layers.



Towards a new model

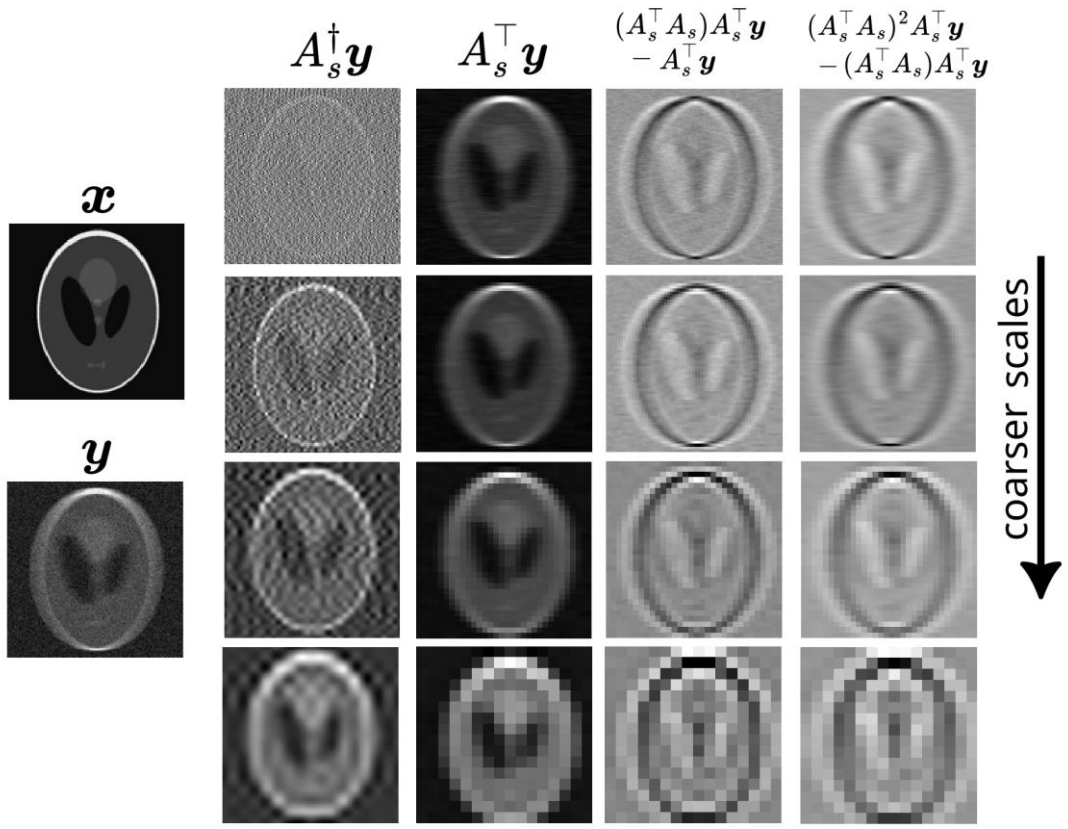
- **Hypothesis 2**: Backpropagating through $A^\top A$ boosts performance

- We can incorporate the operator at every scale of the UNet, as

$A_s = AU_s$ where U_s is an upsampling operator

- The operator becomes better conditioned at coarse scales

- We can often compute $A_s^\top A_s$ efficiently without passing through the fine scale



Towards a new model

- **Hypothesis 2**: Backpropagating through $A^\mathsf{T}A$ boosts performance

- Again, we see that incorporating the operator improves performance, at additional little parameter cost:

Configuration	PSNR (dB)	#Params
base (DRUNet backbone)	25.83	32.6M
base + prox	26.64	32.6M
base + prox + embed y	26.71	33.4M
base + prox + embed y + Krylov (RAM)	27.61	35.6M

Towards a new model

- **Hypothesis 3**: The network should generalize to multiple A if trained on a family

- We evaluate this claim by training on a single or various tasks:

  training on all the tasks performs the best or very close to.

Training tasks	**Inpainting**	**Deblurring**	**SR ($\times$2)**
Inpainting only	30.84	23.03	26.17
Deblurring only	14.06	25.62	28.02
SR$\times$2, $\sigma = 0$ only	n/a	21.25	28.60
All three tasks	30.73	25.58	29.79

Towards a new model

- **Hypothesis 3**: The network should generalize to multiple A if trained on a family

- We train the final model on a family of inverse problems:
  - *Deblurring: random motion, confocal, diffraction, gaussian blurs*
  - *Super-resolution: bicubic, bilinear, sinc*
  - *Tomography: random angles*
  - *Single-coil MRI: random masks (uniform, gaussian, etc)*
  - *Inpainting: random masks (missing lines, Bernoulli)*
  - *Demosaicing: Bayer*
  - *Denoising*

- All tasks are corrupted with Gaussian, Poisson and Poisson-Gaussian noise with random noise parameters

Section Navigation

- Basics
- Optimization
- Plug-and-Play
- Sampling
- Unfolded
- Patch Priors
- Self-Supervised Learning
- Adversarial Learning
- Advanced

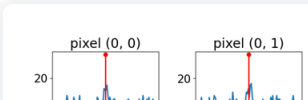
Examples

All the examples have a download link at the end. You can load the example's notebook on Google Colab and run them by adding the line

```
pip install git+https://github.com/deepinv/deepinv.git#egg=deepinv
```

to the top of the notebook (e.g., as in here).

Basics



Deep Inverse

cnrs Inria THE UNIVERSITY OF EDINBURGH 1794 ENS | PSL ens ENS DE LYON EPFL VIS

Towards a new model

- **Hypothesis 3**: The network should generalize to multiple A if trained on a family

- We consider **multi-channel data:**
  - Single channel images – LIDC-IDRI 1k samples
  - Complex images – FastMRI 2k samples
  - RGB images – LSDIR 90k samples

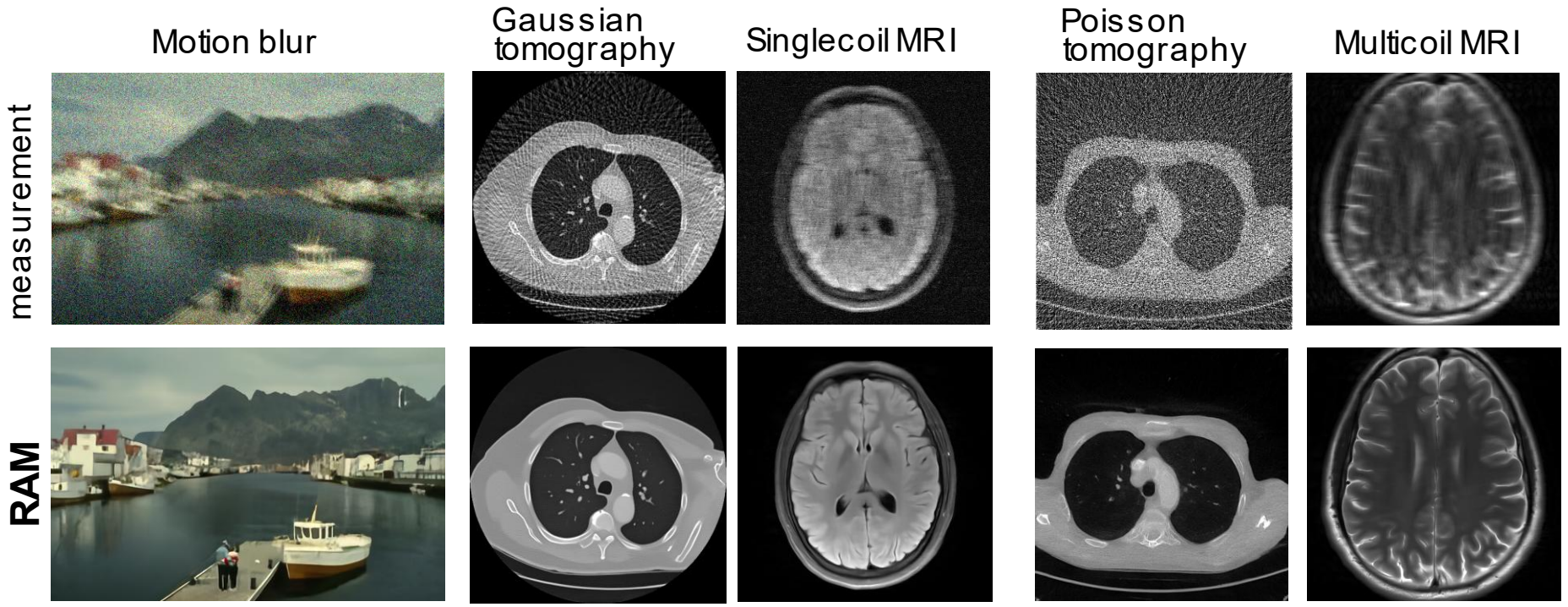
The resulting supervised loss is:

$$\underset{f}{\operatorname{argmin}} \sum_{i=1}^{N} \mathbb{E}_A \mathbb{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i,A}\, w_i ||\boldsymbol{x}_i - f(\boldsymbol{y}_i, A)||_1 \quad \text{with } w_i \propto \frac{||\boldsymbol{x}_i||}{||A^\top \boldsymbol{y}_i - \boldsymbol{x}_i||}$$

- The network is trained for 200k gradient steps with the Adam optimizer.

Reconstruct Anything Model

- The model obtains very good results on in-distribution tasks, and other out-of-distribution tasks which are close to the training ones.



Zero-shot performance

Reconstruct Anything Model

- We perform equally or better than PnP, diffusion and unrolled models, *all of them with many more parameters, and inference time*

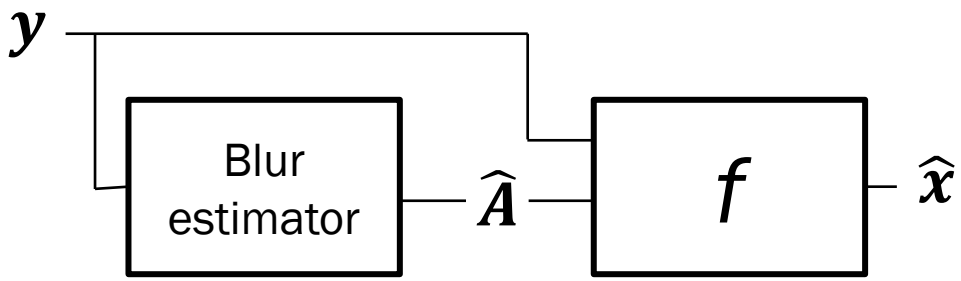
	Method	Params	CBSD68 Easy	CBSD68 Med.	CBSD68 Hard	Urban100 Easy	Urban100 Med.	Urban100 Hard	Div2K Easy	Div2K Med.	Div2K Hard
Iterative	DPIR	32M	33.45	26.53	24.42	**34.39**	28.12	24.38	**36.32**	29.97	27.16
Iterative	DiffPIR	552M	31.83	25.98	23.80	32.93	27.03	23.69			
Iterative	DDRM*	552M	33.1(								
End-to-end	PDNet	456K	30.5⁴								
End-to-end	Restormer	26M	30.9(								
End-to-end	uDPIR/t	32M	33.78								
End-to-end	uDPIR/u	256M	33.6⁴								
End-to-end	RAM	36M	34.0⁴								

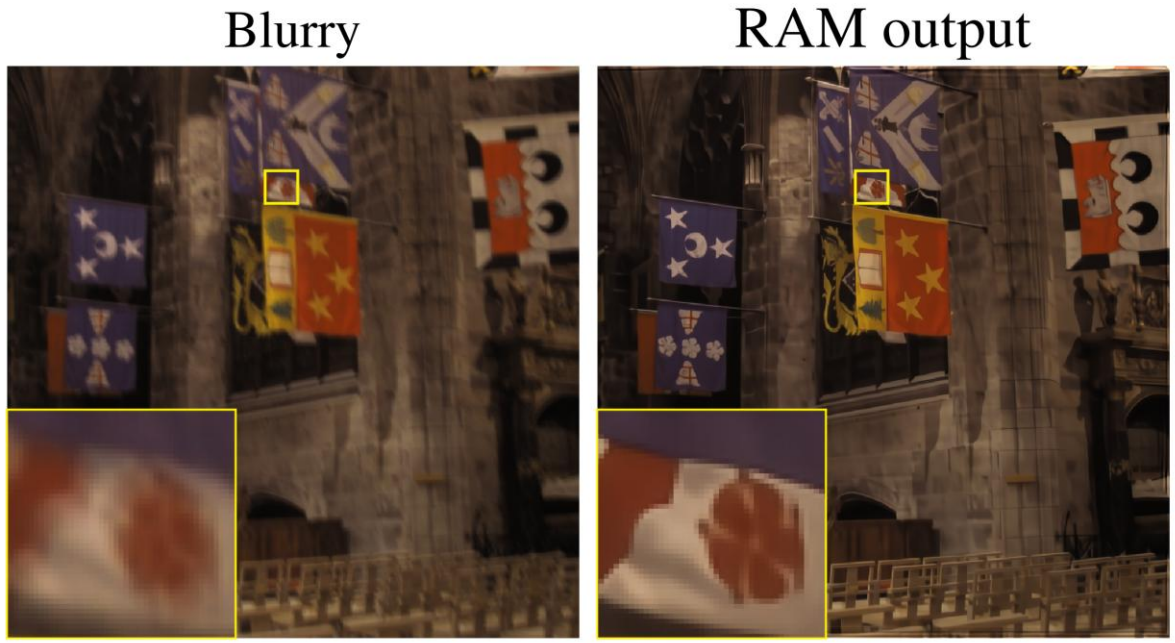
	CBSD68 Easy	CBSD68 Med.	CBSD68 Hard	Urban100 Easy	Urban100 Med.	Urban100 Hard	Div2K Easy	Div2K Med.	Div2K Hard
DPIR	32.10	25.70	22.73	**32.53**	23.78	20.38	**35.23**	28.14	24.71
DiffPIR	30.93	24.96	22.51	31.17	23.63	20.32	-	-	-
DDRM*							34.46	28.25	24.89
PDNet							32.01	27.29	24.03
Restormer							30.01	27.20	25.00
uDPIR/t							34.63	28.45	25.20
uDPIR/u							34.47	28.37	25.23
RAM							35.23	28.56	25.16

	PDNet	uDPIR-untied	uDPIR-tied	RAM
GFLOPS	60	2234	2234	360
Test memory (BS=1)	52	1213	298	354
Train memory (BS=8)	5815	37288	36374	9670

Non-linear problems

- The model can be extended to do phase retrieval and blind deblurring *without any retraining*
- For blind deblurring, we first estimate a space-varying blur (Carbajal et al., 2023), and then use RAM



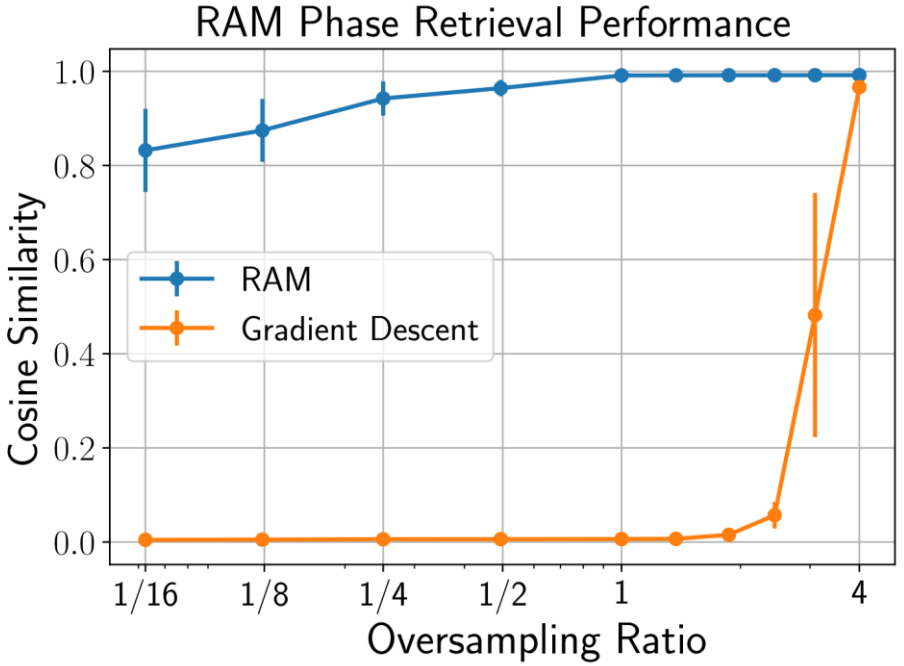


Non-linear problems

- For phase retrieval $y = |Ax|^2$, we use RAM inside the Gerchberg Saxton algorithm

for $k = 1, \dots, K$

- $\boldsymbol{\phi}^k = \text{phase}(A\boldsymbol{x}^k)$
- $\boldsymbol{x}^{k+1} = f(\boldsymbol{y} \circ \mathrm{e}^{i\boldsymbol{\phi}^k}, A)$



Self-Supervised Finetuning

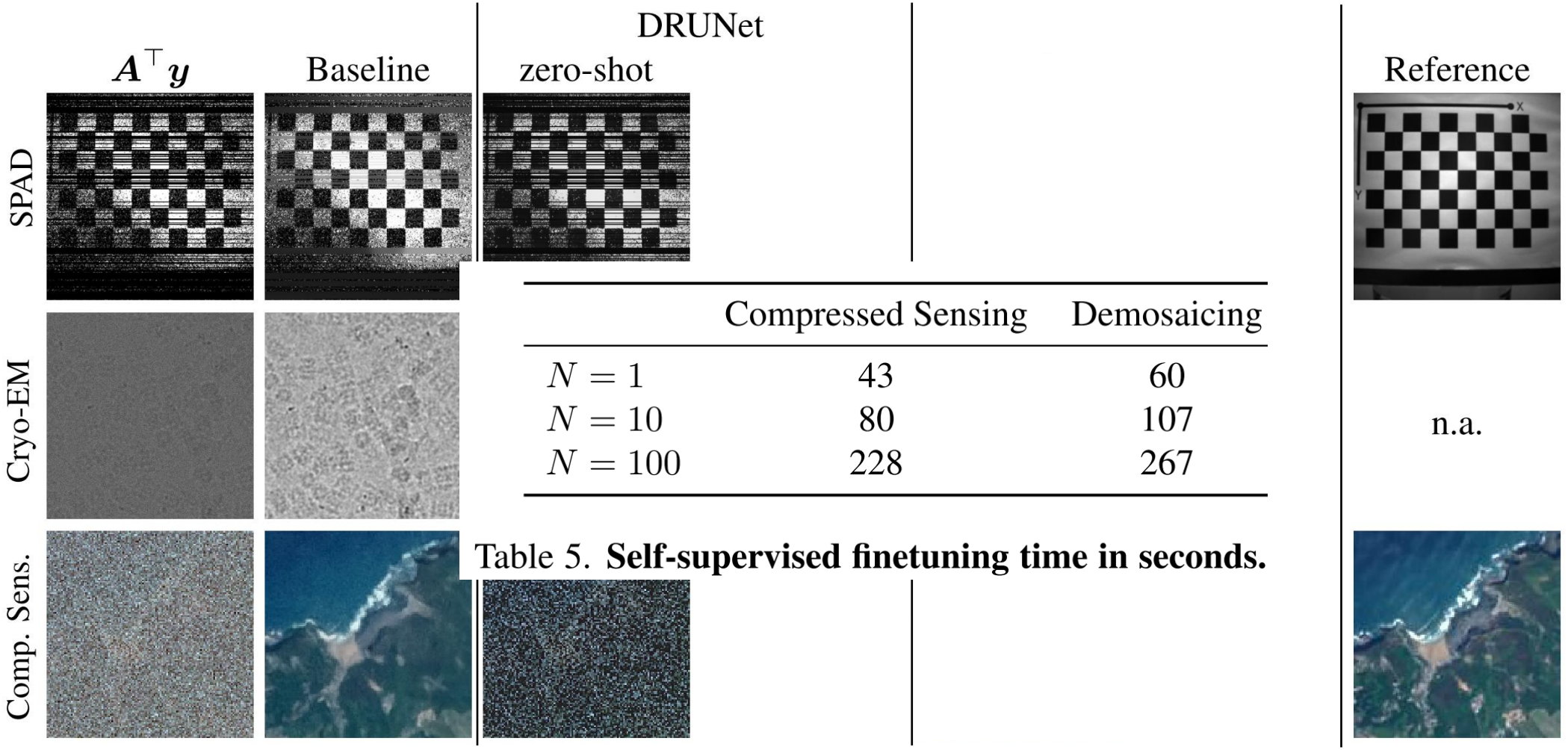
- If the inverse problem is ‘far’ from the training distribution, the performance may degrade
- To alleviate this, we can finetune RAM with **measurement data only** by minimizing a self-supervised loss, such as SURE for gaussian noise:

$$\underset{f}{\operatorname{argmin}} \sum_i \left|\left|\boldsymbol{y}_i - Af(\boldsymbol{y}_i)\right|\right|^2 + 2\sigma^2 \operatorname{div}(A \circ f)(\boldsymbol{y}_i)$$

- SURE can be replaced by UNSURE, Recorrupted2Recorrupted, etc., according to knowledge of noise
- if A is incomplete/ill-posed, we add the EI, MOI or splitting loss
- See Tachella and Davies “Learning from Noisy and Incomplete Data”, Foundations & Trends in SP, 2026

Self-Supervised Finetuning

- The model can be finetuned with self-supervised losses on up to a single $\boldsymbol{y}$ $(N = 1)$ **in minutes**



	Compressed Sensing	Demosaicing
$N = 1$	43	60
$N = 10$	80	107
$N = 100$	228	267

Table 5. **Self-supervised finetuning time in seconds.**

Uncertainty Quantification

- Since the model is lightweight, we can inference it multiple times to quantify uncertainty

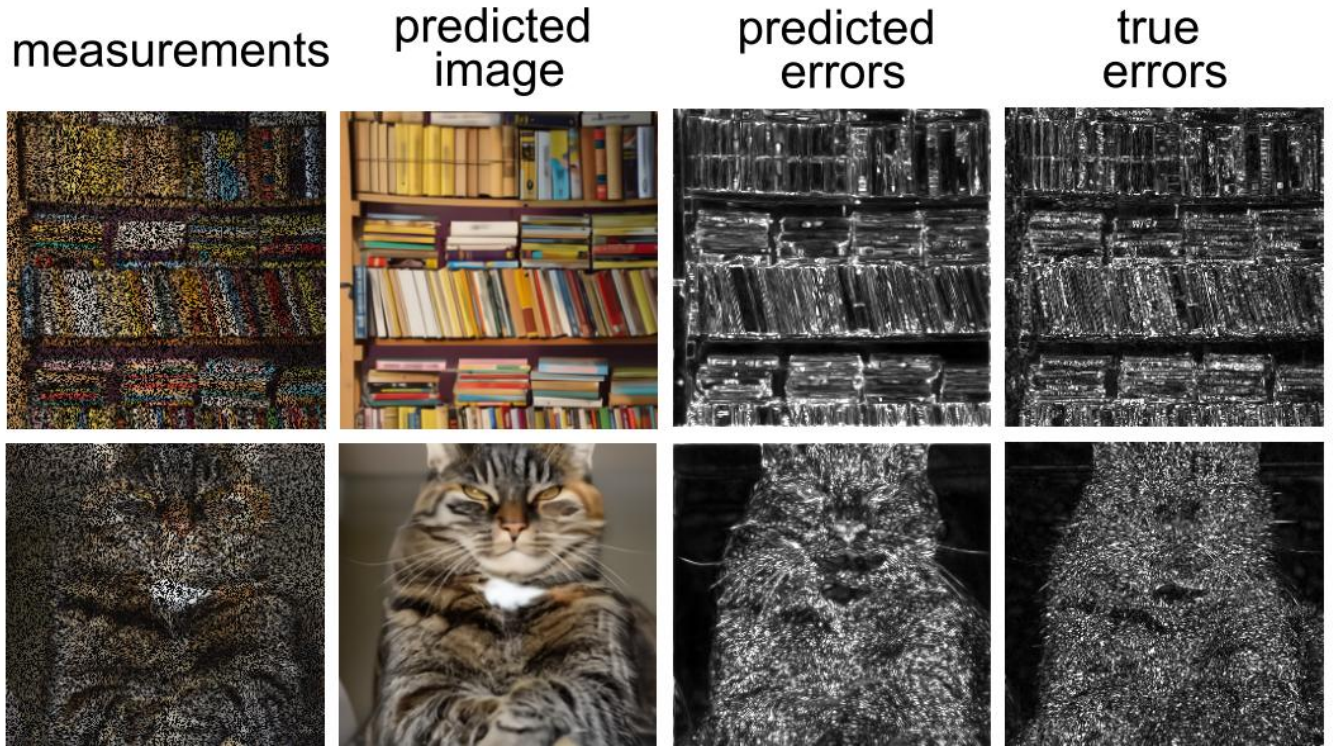
- *Equivariant bootstrap* (Tachella & Pereyra, 2024):
  Using $\hat{\boldsymbol{x}} = f(\boldsymbol{y})$ as 'ground-truth':

For $j = 1, \ldots, r$

Sample transform g and noise $\boldsymbol{\epsilon}_j$

Bootstrap $\boldsymbol{x}^j = T_g^{-1} f(A T_g \hat{\boldsymbol{x}} + \boldsymbol{\epsilon}_j)$

Error estimates: $\boldsymbol{e}^j = \hat{\boldsymbol{x}} - \boldsymbol{x}^j$



Conclusions

- **Disadvantages**:
  - ~~Trained for a specific task, doesn't generalize to different tasks~~
  - ~~Requires a large supervised dataset specific to application~~
  - ~~It is a black-box, no notion of uncertainty~~

$p(A)$ is low-dimensional

If you want to try out the model:

```
import deepinv as dinv
x = dinv.utils.load_example("butterfly.png")
physics = dinv.physics.Inpainting(...)
y = physics(x)
model = dinv.models.RAM()
x_hat = model(y, physics) # run model
```

Open questions

- Can we iterate it for better perceptual performance (Delbraccio, 2023)?
- What is the **least** number of A evaluations necessary?
- Can we go further by **scaling up** the model?
- Can we express our uncertainty about A?
- What about 3D or 3D+time?

Stay tuned, RAMv2 coming soon!