Lyon, Avril 21- 25

DLMI2025

(A short intro to) Deep learning for image reconstruction

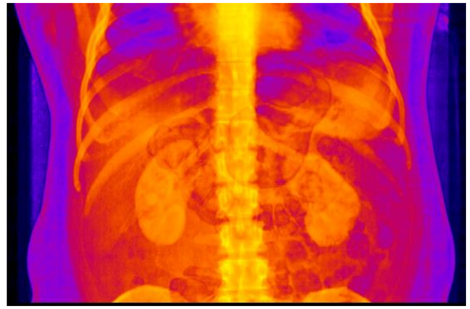
Nicolas Ducros[1, 2]

[1]CREATIS, Univ Lyon, INSA-Lyon, UCB Lyon 1,CNRS, Inserm, CREATIS UMR 5220, U1206, Lyon, France

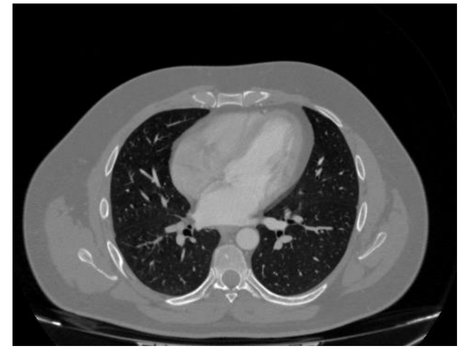
[2]IUF, Institut Universitaire de France

“Natural” vs Reconstructed Images

Nat



Rec



Rec

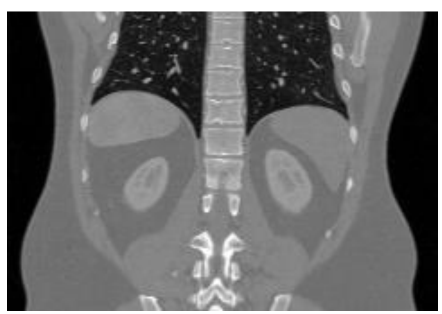
Nat

Nat

Nat

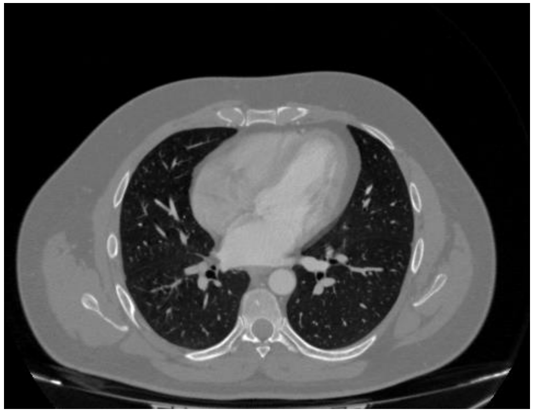
Nat

Rec



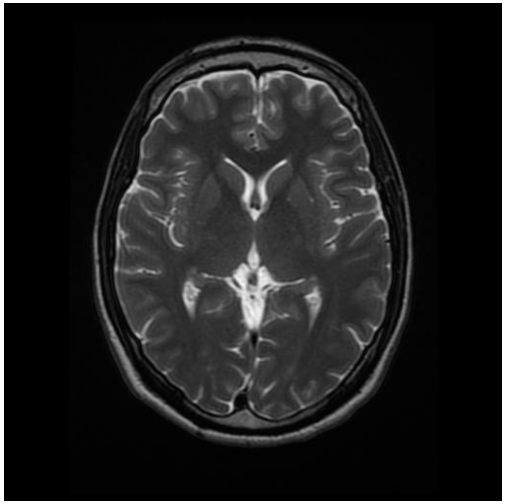
Reconstruction in Medical Imaging

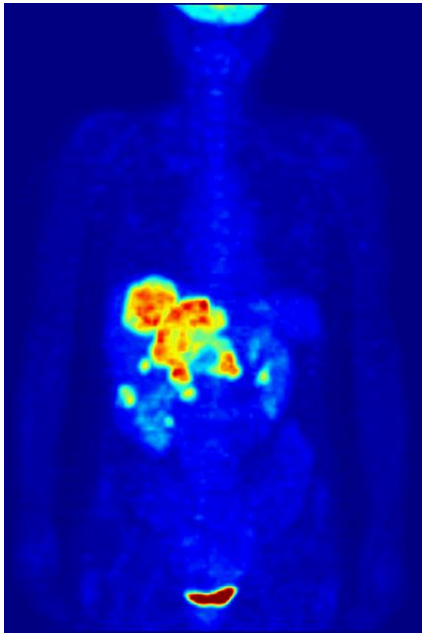
Computerized tomography (CT)



Ultrasound Imaging

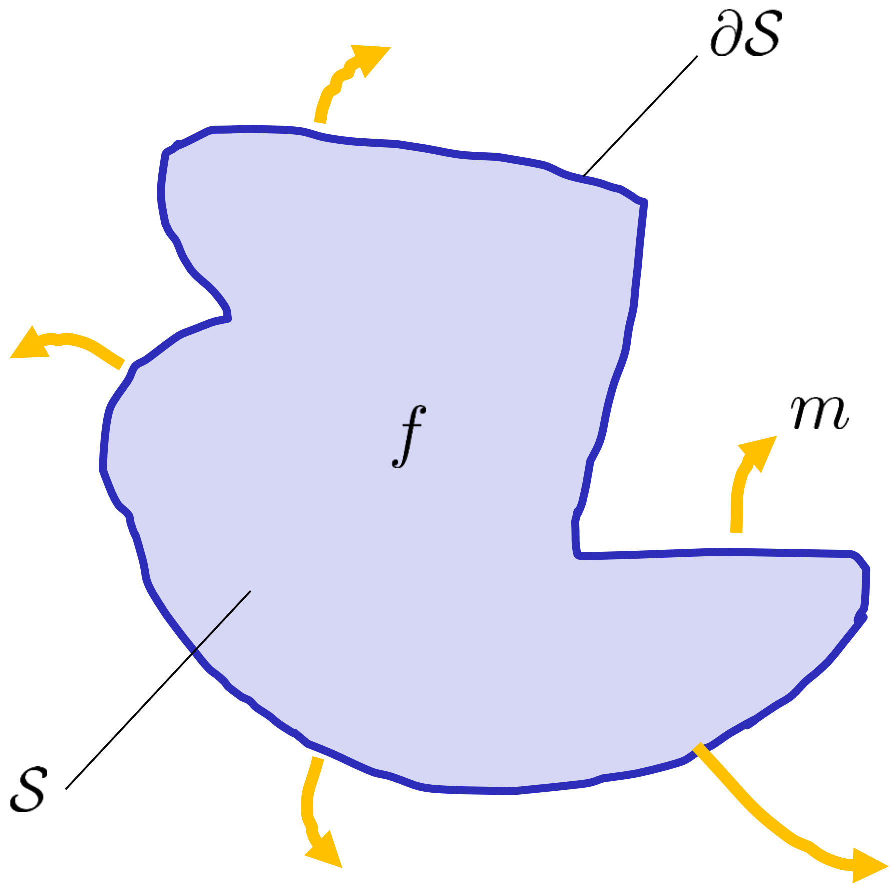
Magnetic Resonance (MRI)





Positron emission tomography (PET)

Inverse Problem



Inverse Problem

- **Internal unknowns from external measurements**

$$m = \mathcal{A}(f)$$

External Measurement

Forward (or direct) Model

(Unknown) quantity of Interest

- **Most medical imaging problems are**
  - Linear
  - Corrupted by noise

- **In a discrete setting**

$$\boldsymbol{m} = \boldsymbol{A}\boldsymbol{f} + \boldsymbol{\eta}$$

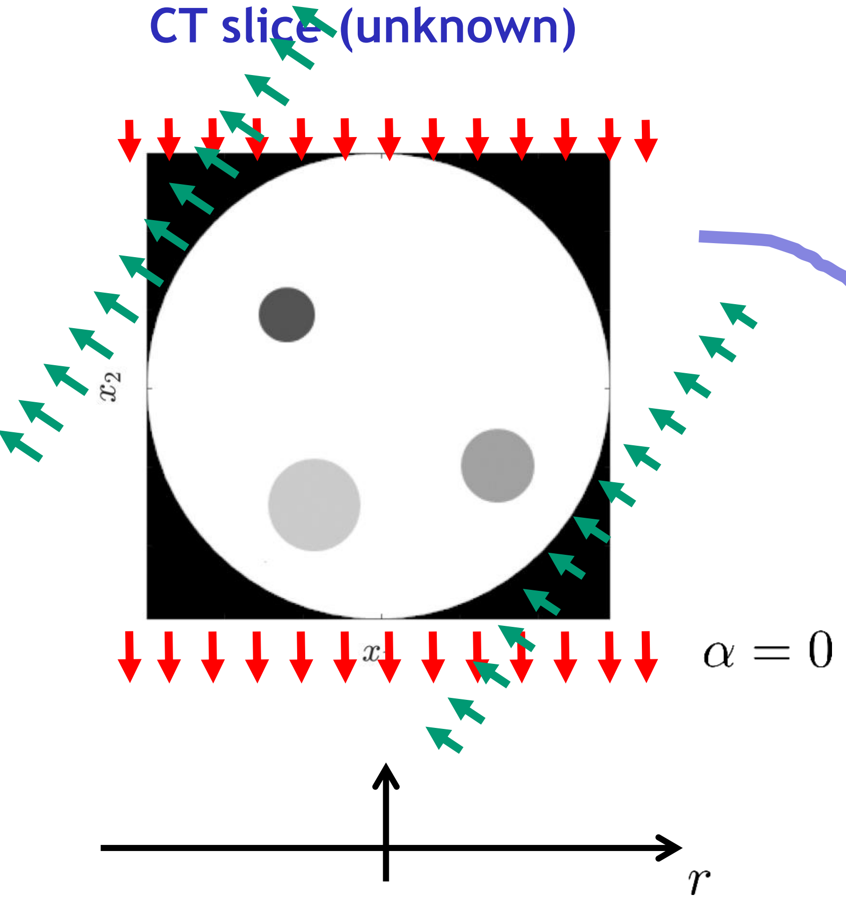
Noise

$\boldsymbol{m} \in \mathbb{R}^M$

$\boldsymbol{A} \in \mathbb{R}^{M \times N}$

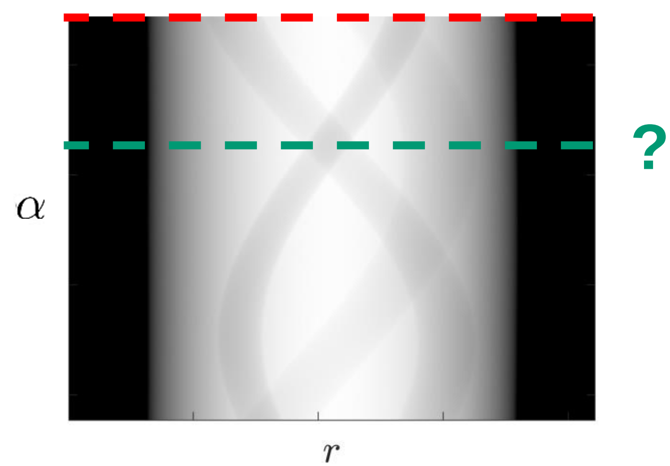
$\boldsymbol{f} \in \mathbb{R}^N$

Computed Tomography (CT)



$$m(\alpha, r) = \int_{\mathcal{L}(\alpha,r)} f(x_1, x_2)\,\mathrm{d}\ell$$

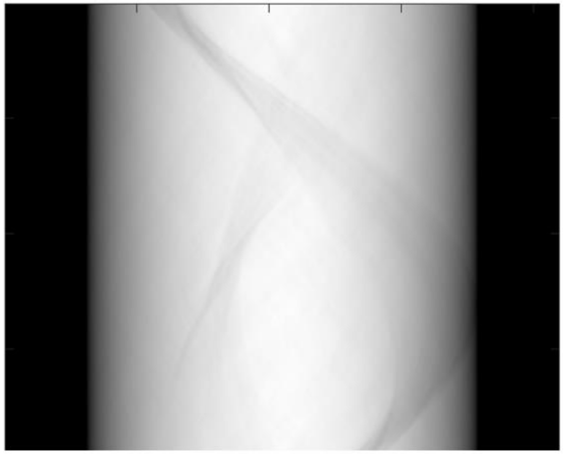
Sinogram (measured)



Computed Tomography (CT)

CT slice (unknown)

Sinogram (measured)



Different Options

- **1. Inversion "by hand"**
  - Model the forward and invert analytically

$$\texttt{derive } \mathcal{R} \texttt{ such that } f = \mathcal{R}(m)$$

- **2. Optimization of handcrafted functionals**
  - Build cost function from prior knowledge about the solution/measurements
  - Minimize the cost function

$$\texttt{find and minimize } \mathcal{C} \texttt{ such that } \mathcal{C}(f;m) \texttt{ is small}$$

- **3. "Learn" to reconstruct**
  - (Probably what you expect from this talk)

$$\texttt{learn } \mathcal{R}_\theta \texttt{ such that } f = \mathcal{R}_\theta(m)$$

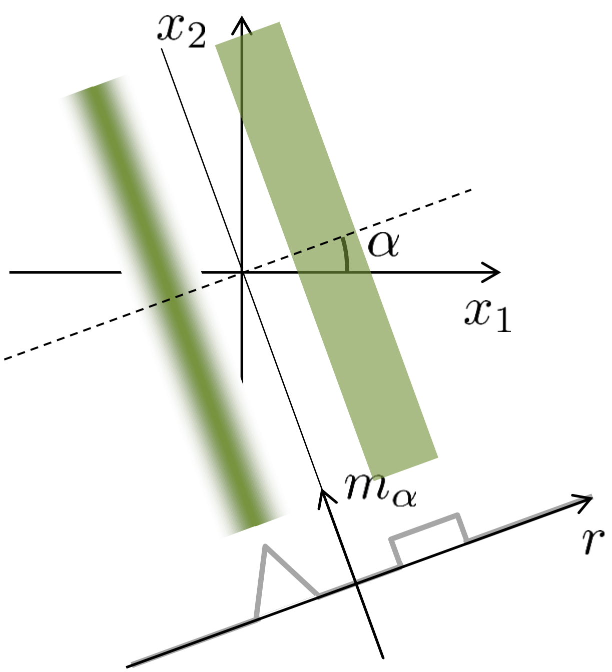
Option #1:
Analytical Methods

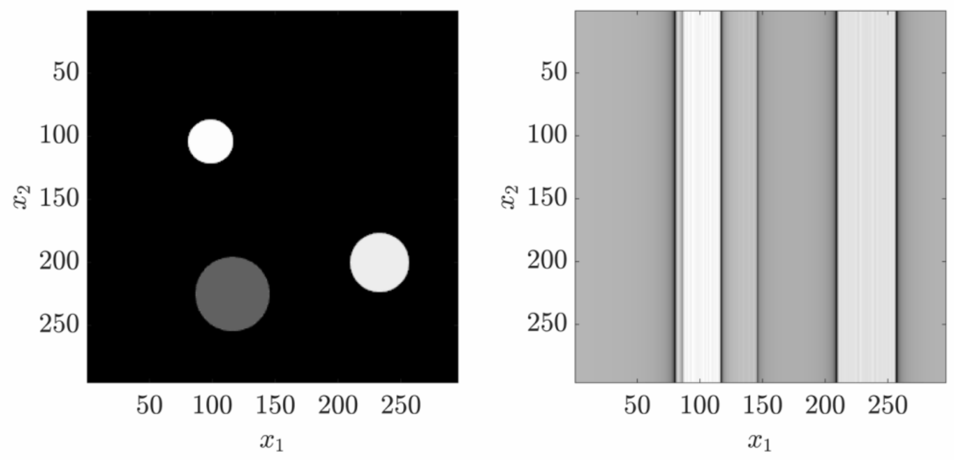
➢ Example with CT (filtered backprojection)

$$f(x_1, x_2) = \int_0^\pi m_\alpha^{\text{filt}}(x_1 \cos\alpha + x_2 \sin\alpha)\, \mathrm{d}\alpha$$

$$\hat{m}_\alpha^{\text{filt}}(\xi) = |\xi|\, \hat{m}_\alpha(\xi)$$

Ramp filter





1. Analytical Methods

- **Pros**
  - Elegant
  - Theoretical guarantees
  - Usually fast implementation
  - What if only few measurements are available?
    - For dose reduction/short scans

- **Cons**
  - Not always possible to derive a solution
  - Influence of noise?
  - What if only few measurements are available?
    - Random or pseudo-random

Option #2:
Optimization-Based Methods

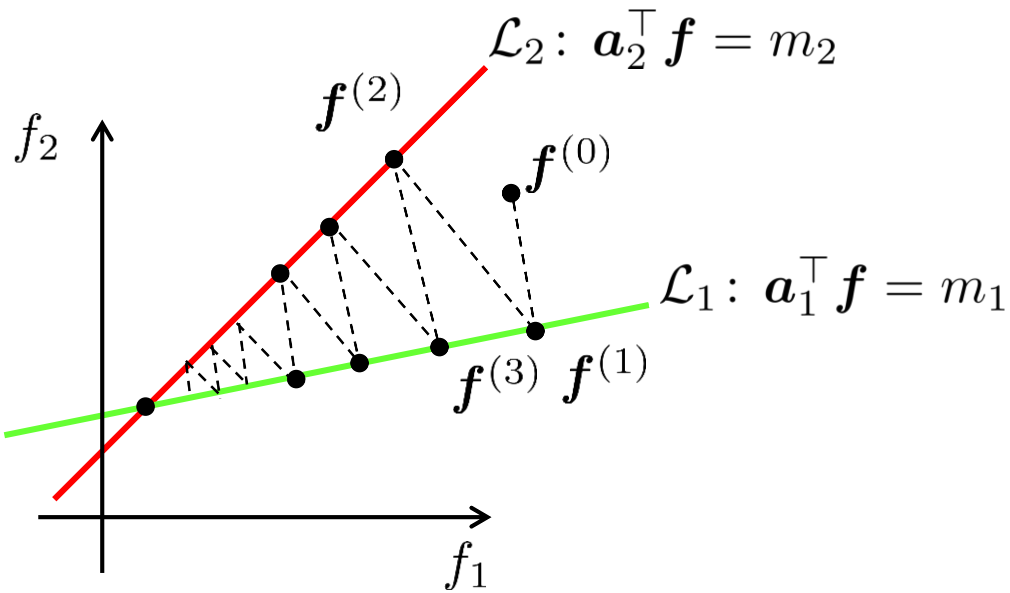
2. Optimization-Based Methods

- **Look for an image with small residuals**

$$\boldsymbol{r} = \boldsymbol{m} - \boldsymbol{A}\boldsymbol{f} \approx \boldsymbol{0}$$

- A simple example:

$$\begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$



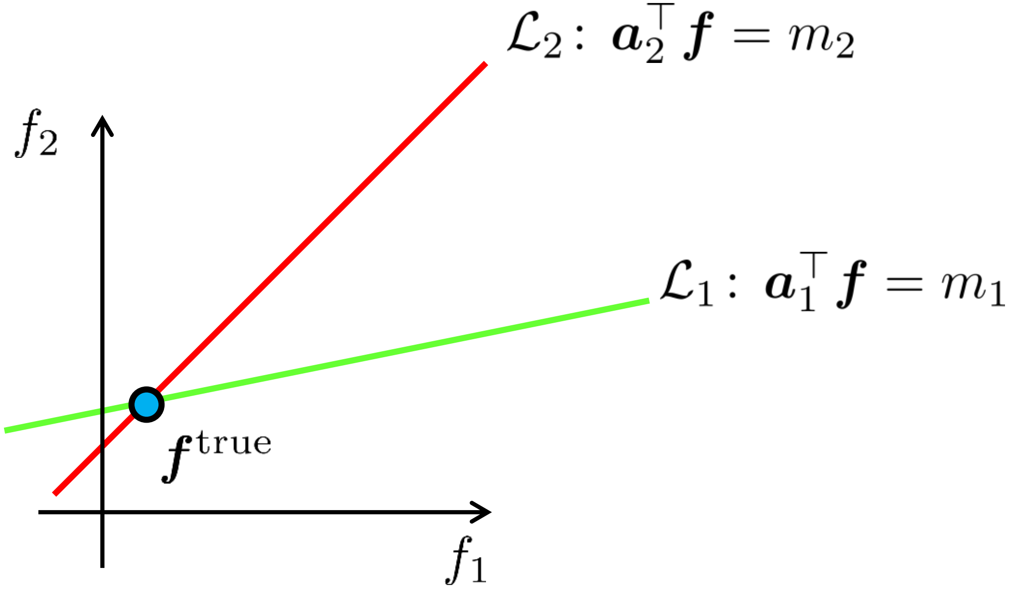
Algebraic reconstruction technique (**ART**) [Gordon R., 1970]

2. Optimization-Based Methods

- **Look for an image with small residuals**

$$\boldsymbol{r} = \boldsymbol{m} - \boldsymbol{A}\boldsymbol{f} \approx \boldsymbol{0}$$

- **Influence of noise**



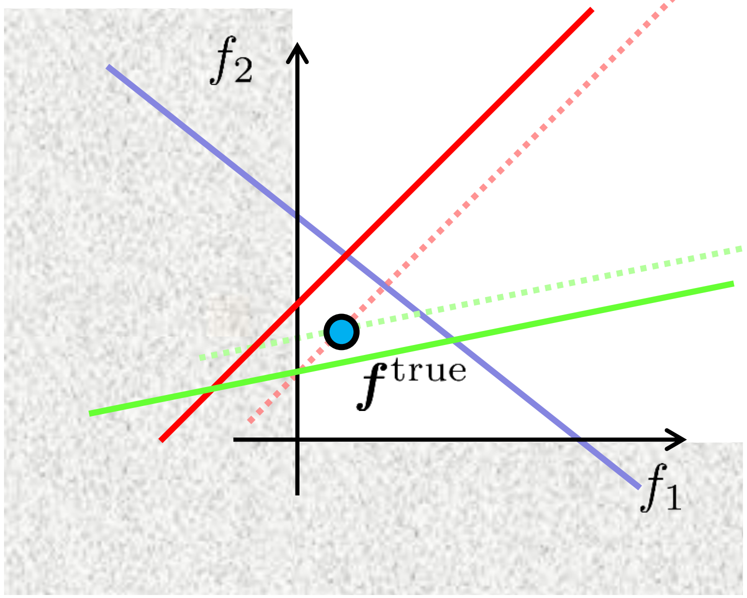
2. Optimization-Based Methods

- **Look for an image with small residuals**

$$\boldsymbol{r} = \boldsymbol{m} - \boldsymbol{A}\boldsymbol{f} \approx \boldsymbol{0}$$

- **Influence of noise**
  - More measurements (i.e., $M > N$)
  - Prior knowledge (e.g., $\boldsymbol{f} > \boldsymbol{0}$)

$$\mathcal{L}_2 : \; \boldsymbol{a}_2^\top \boldsymbol{f} = m_2 + \boxed{\eta_2}$$



$$\mathcal{L}_1 : \; \boldsymbol{a}_1^\top \boldsymbol{f} = m_1 + \boxed{\eta_1}$$

2. Optimization-Based Methods

➢ Typical cost functions

$$\min_{\boldsymbol{f}} \; \mathcal{D}(\boldsymbol{f}, \boldsymbol{m}) + \lambda \Psi(\boldsymbol{f})$$

Data fidelity

Regularization (prior)

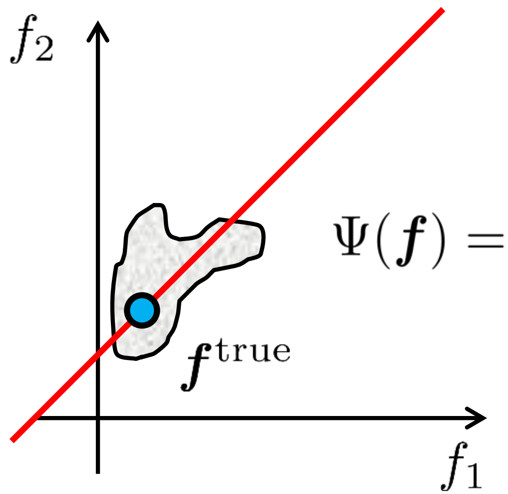
- Data fidelity is related the noise model/measurements confidence. E.g.

$$\mathcal{D}(\boldsymbol{f}; \boldsymbol{m}) = \|\boldsymbol{m} - \boldsymbol{A}\boldsymbol{f}\|_2^2$$

$$\mathcal{D}(\boldsymbol{f}; \boldsymbol{m}) = \mathrm{KL}(\boldsymbol{m}, \boldsymbol{A}\boldsymbol{f})$$

- Regularization convey prior knowledge about the solution.

$$M \ll N$$



$$\Psi(\boldsymbol{f}) = \begin{cases} 0 \text{ if } \boldsymbol{f} \in \mathcal{X} \\ \infty \text{ otherwise} \end{cases}$$

Typical regularizers

- Quadratic/Tikhonov regularization

$$\Psi(\boldsymbol{f}) = \|\boldsymbol{f}\|_2^2$$

… leads to

$$\boldsymbol{f}^* = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top + \lambda \boldsymbol{I}_M)^{-1}\boldsymbol{m}$$

- Sparsity-promoting

$$\Psi(\boldsymbol{f}) = \|\boldsymbol{\Phi}\boldsymbol{f}\|_1$$

… requires iterative algorithms

$$\boldsymbol{z}^{(k)} = \boldsymbol{f}^{(k-1)} - \eta \boxed{\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{f}^{(k-1)} - \boldsymbol{m})}$$

$$\boldsymbol{f}^{(k)} = \boxed{\mathtt{prox}_{\lambda\Psi}}(\boldsymbol{z}^{(k)})$$

gradient of data fidelity

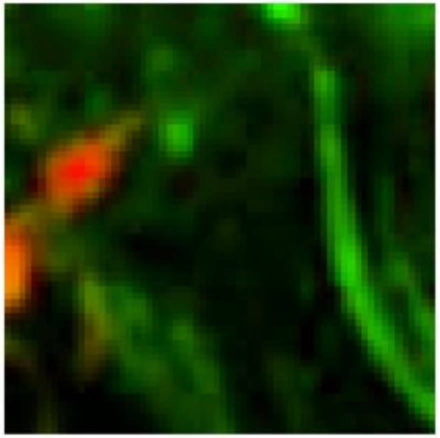
proximal operator of regularizer

2. Optimization-Based Methods

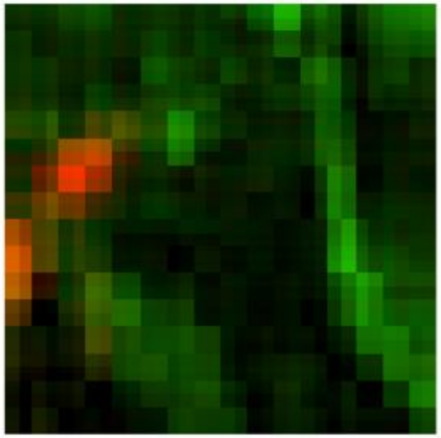
➢ Illustrative results

$N = 64 \times 64$ image
$M = 333$ measurements
$N / M \approx 8\%$

Ground-Truth



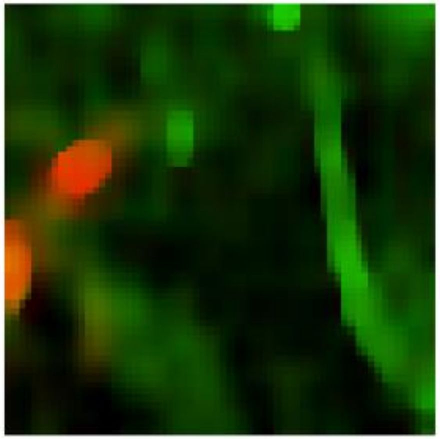
$\Psi(\boldsymbol{f}) = \|\boldsymbol{f}\|_2^2$



++ Analytical solution (fast computation)

- - Image quality

$\Psi(\boldsymbol{f}) = \|\nabla \boldsymbol{f}\|_1$



- - Iterative algorithms (time consuming)

++ Image quality

Option #3:
Learning-Based Methods

3. Learning-Based Methods

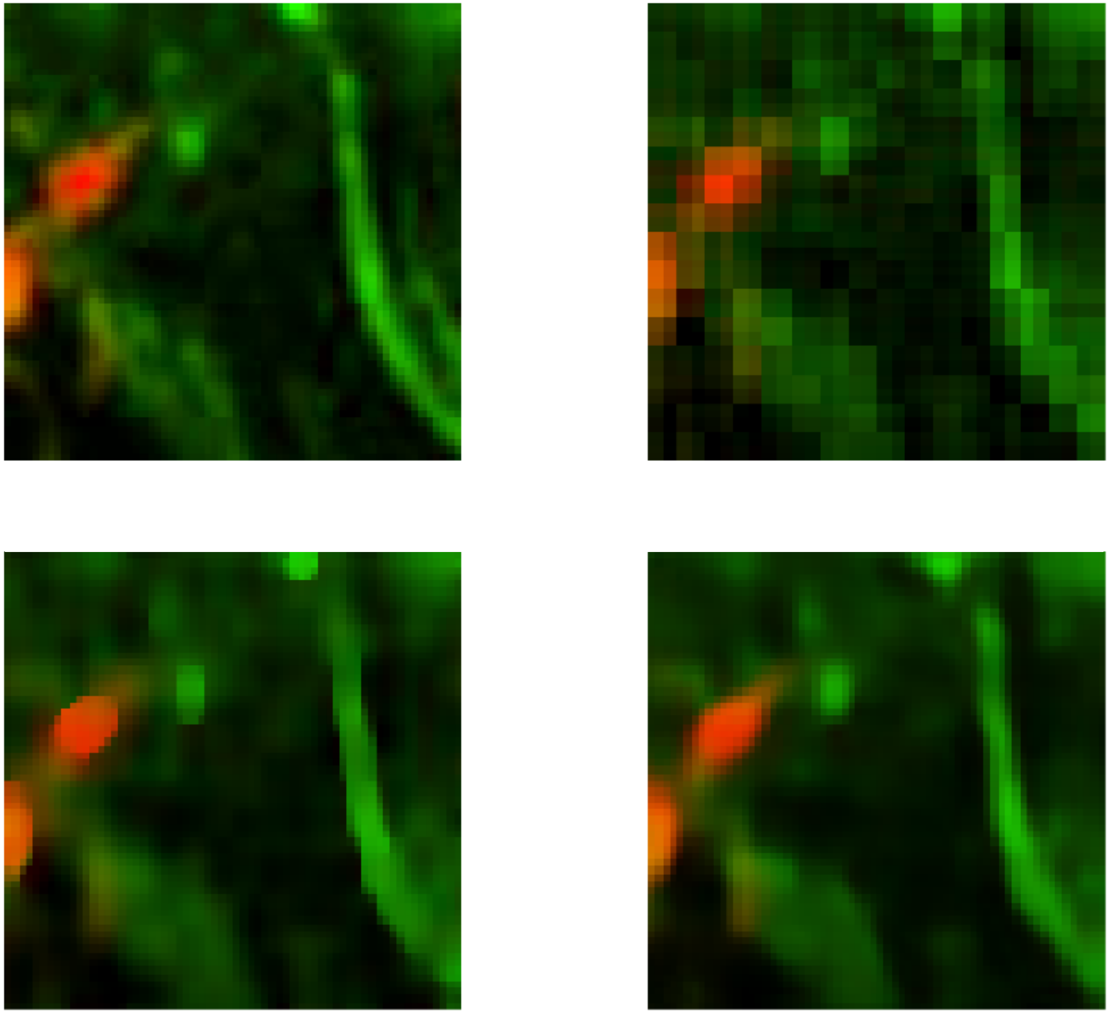
➢ Optimization- vs learning-based methods

N = 64 × 64 image
M = 333 measurements
N / M ≈ 8%

Ground-Truth

$\Psi(\boldsymbol{f}) = \|\boldsymbol{f}\|_2^2$

(pinv)



$\Psi(\boldsymbol{f}) = \|\nabla \boldsymbol{f}\|_1$

+2.5 dB (wrt pinv)

CNN

+3.5 dB (wrt pinv)

[N. Ducros et al., IEEE ISBI, 2019]

3. Learning-Based Methods

- **Our dream is to find**

$$\mathcal{R}^* : \mathbb{R}^M \mapsto \mathbb{R}^N \text{ such that } \mathcal{R}^*(\boldsymbol{m}) = \boldsymbol{f}^{\text{true}}$$

… able to reconstruct well any image, i.e., something like

$$\mathcal{R}^* \in \arg\min_{\mathcal{R}} \frac{1}{L} \sum_{\ell} \|\mathcal{R}(\boldsymbol{m}^\ell) - \boldsymbol{f}^\ell\|_2^2$$

Minimum mean square error (MMSE) estimator

… Often intractable

- **We have to reduce the dimension of the solution space**
  - E.g.,

$$\mathcal{R}(\boldsymbol{m}) = \boldsymbol{W}\boldsymbol{m} + \boldsymbol{b},$$

Linear MMSE estimator

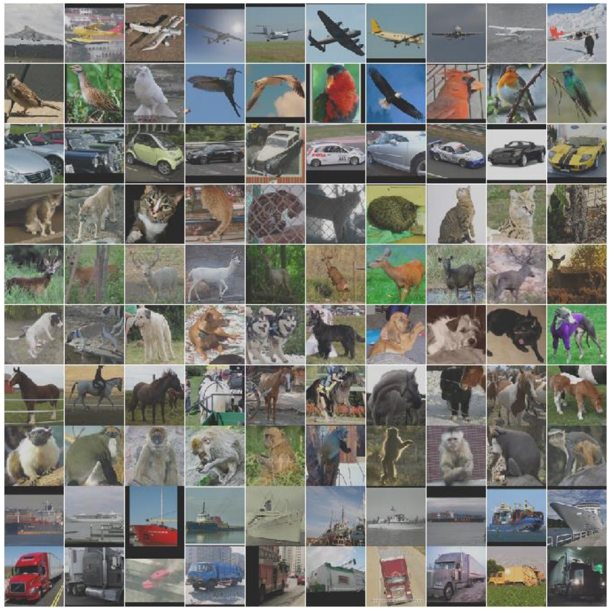
- **Linear MMSE**

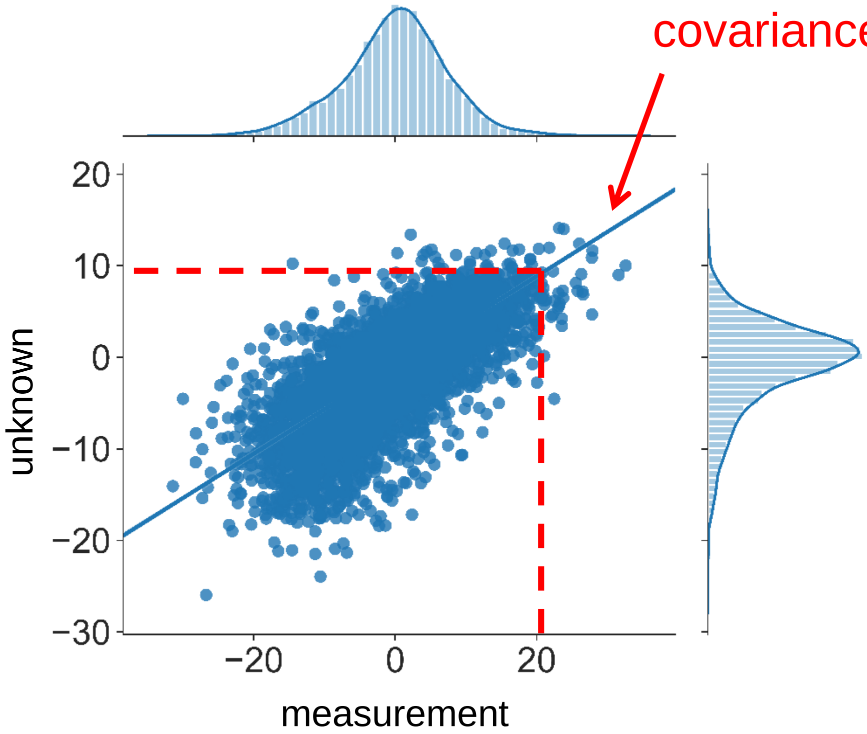
$$\boldsymbol{f}^* = \bar{\boldsymbol{f}} + \boldsymbol{\Sigma}_{f,m}\boldsymbol{\Sigma}_m^{-1}(\boldsymbol{m} - \bar{\boldsymbol{m}})$$

Covariance between measurements and unknowns

Covariance of measurements

slope → covariance





3. Learning-Based Methods

- **Learning approaches *only* reduce the dimension of the solution space to a family of *non linear* mappings**

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \frac{1}{L} \sum_{\ell} \|\mathcal{R}(\boldsymbol{\theta}; \boldsymbol{m}^{\ell}) - \boldsymbol{f}^{\ell}\|_2^2$$

- Training phase
  - Image-measurement pairs $\{\boldsymbol{f}^{(\ell)}; \boldsymbol{m}^{(\ell)})\}_{1\leq\ell\leq L}$
  - Loss (e.g., MSE)
  - Optimization machinery (i.e., PyTorch or TensorFlow)

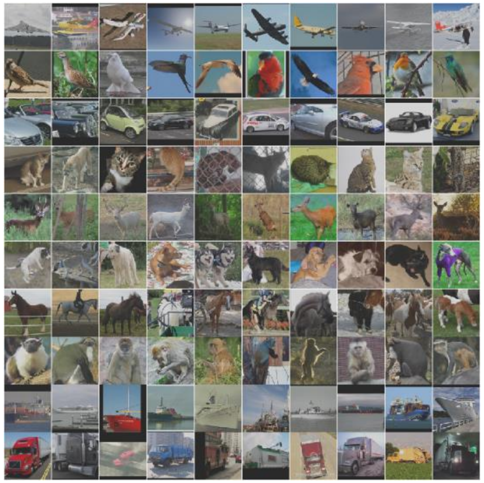
D.P. Kingma and J.L Ba, ICRL, 2015 (> 215k citations)

A. Paszke *et al.*, NEURIPS, 2019 (> 22k citations)

- Reconstruction phase

$$\boldsymbol{f}^* = \mathcal{R}_{\boldsymbol{\theta}^*}(\boldsymbol{m})$$

STL-10 dataset

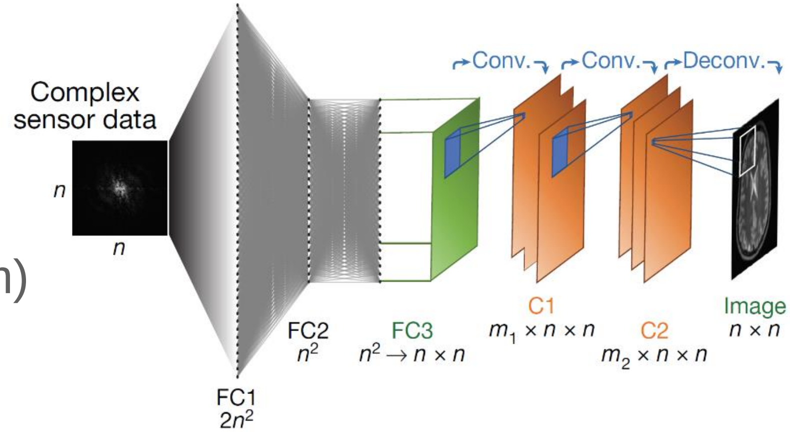


3. Learning-Based Methods

- **Pros**
  - Reconstruction performance
    - Empirically excellent (i.e., almost always outperform optimization-based approaches)
  - Computation times
    - Training phase is slow, i.e., several hours or days
    - Inference is fast, i.e., tens or hundreds of milliseconds

- **Cons**
  - No reconstruction guarantees (*mathematicians don't like it*)
  - Black box (*radiologists don't like it*)
  - Practical issue
    - How to choose the model?

"Automap"
(> 6.10^9 param)



[B. Zhu et al., Nature Letters, 2018] (> 1.5k citations)

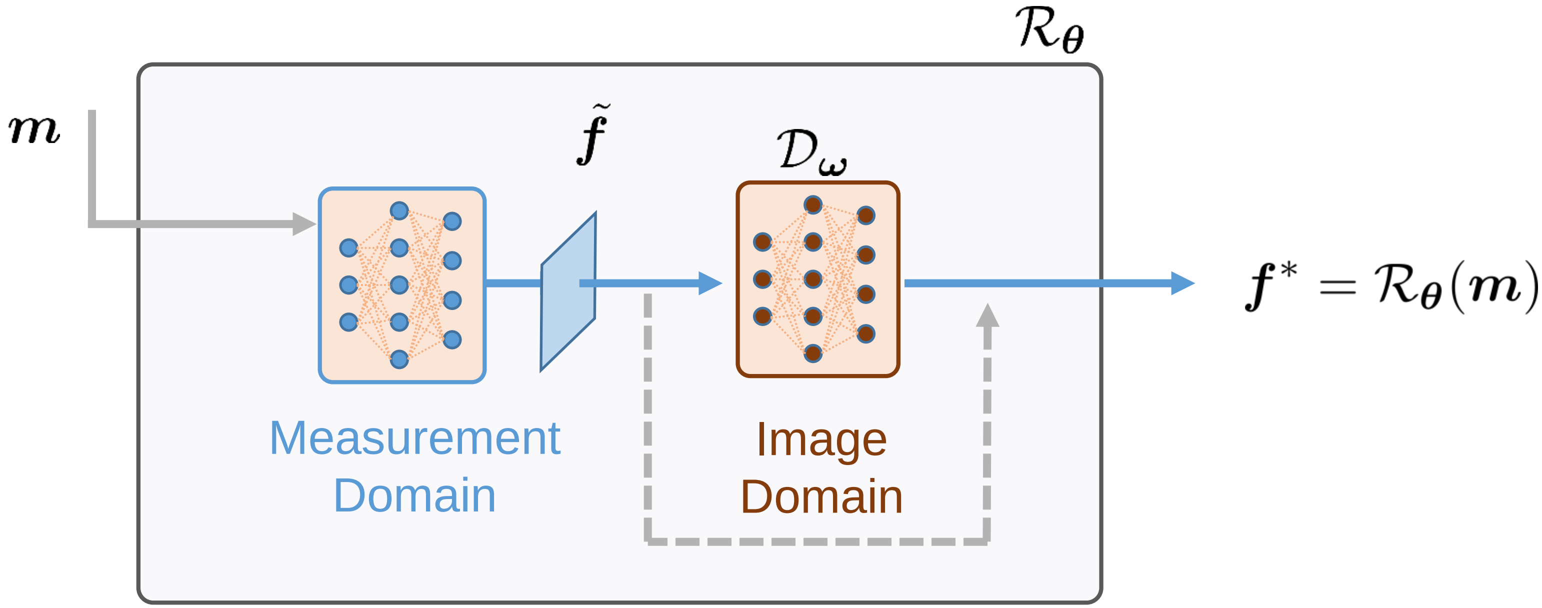
3.1. Post-Processing

- **Two-step methods**

$$\tilde{\boldsymbol{f}} = \tilde{\boldsymbol{A}}^{-1}\boldsymbol{m}$$

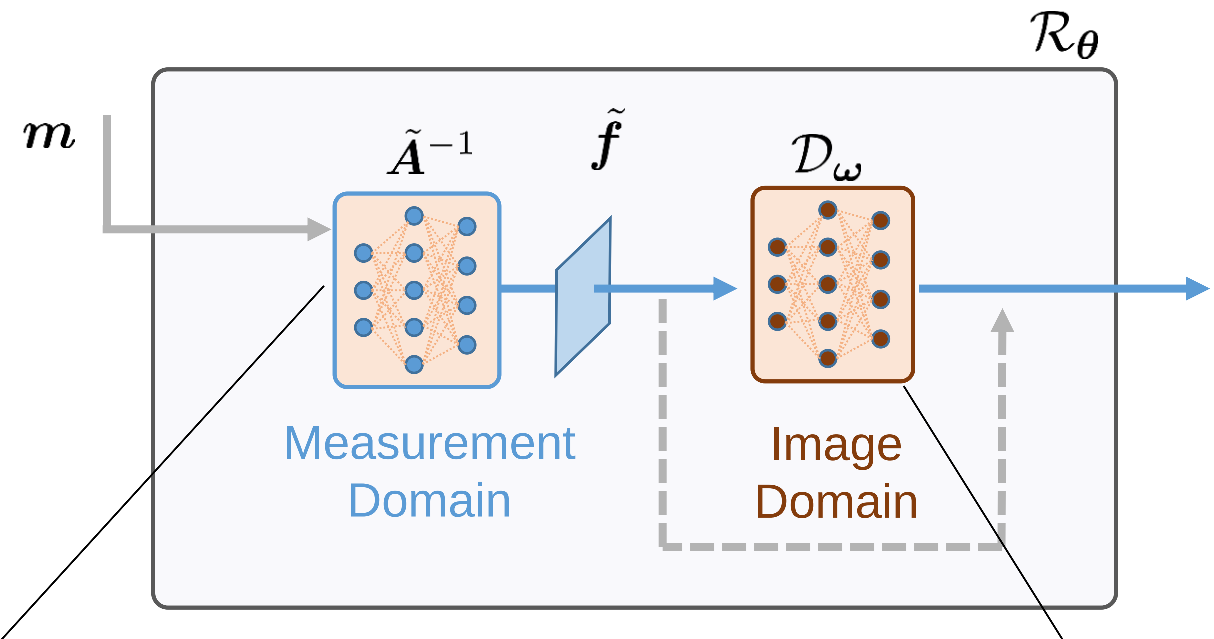
$$\boldsymbol{f}^* = \mathcal{D}_{\boldsymbol{\omega}}(\tilde{\boldsymbol{f}}) + \tilde{\boldsymbol{f}}$$

where $\tilde{\boldsymbol{A}^{-1}}$ is an approximate inverse of the forward, i.e., $\tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}\boldsymbol{f} \approx \boldsymbol{f}$



3.1. Post-Processing

➢ **Equivalent to a neural network with a frozen layer**



network parameters

$$\boldsymbol{f}^* = \mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{m})$$

E.g., $\boldsymbol{\theta} = \boldsymbol{\omega}$.

```
Atilde = nn.Linear(..., bias=False, ...)
Atilde.weight.requires_grad = False
```

$$\tilde{\boldsymbol{A}}^{-1} = \boldsymbol{A}^\top$$

$$\tilde{\boldsymbol{A}}^{-1} = \boldsymbol{A}^\dagger \text{ (E.g., } \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1} \text{ for full row rank } A)$$

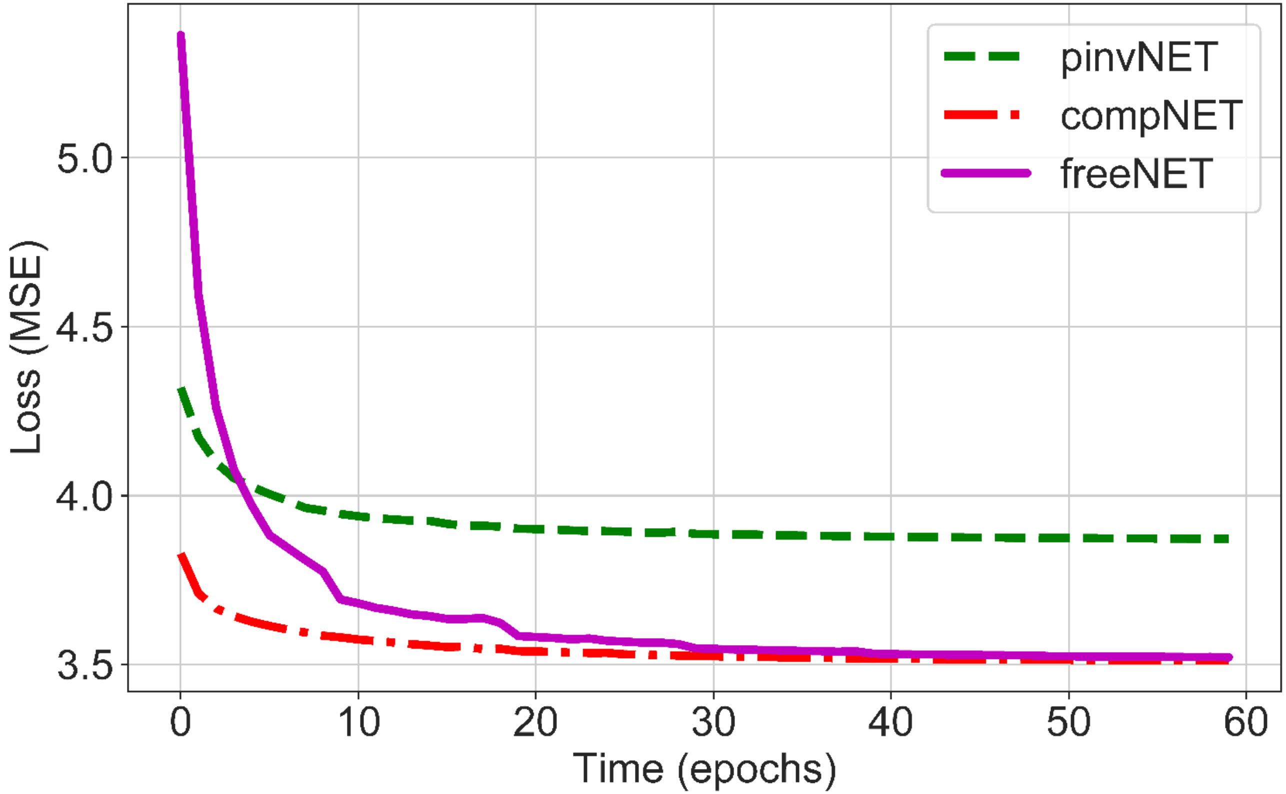
$$\tilde{\boldsymbol{A}}^{-1} = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top + \boldsymbol{I})^{-1}$$

```
D = nn.Module(...)
requires_grad = True
```

3.3. Training

- **STL-10 (training: ~100k images; test: 8k images)**

$$\sum_{\ell \in \mathcal{I}_{\text{test}}} \|\boldsymbol{f}^{(\ell)} - \mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{m}^{(\ell)})\|^2$$



$$\tilde{\boldsymbol{A}}^{-1} = \boldsymbol{A}^{\dagger}$$
$$\tilde{\boldsymbol{A}}^{-1} = \boldsymbol{A}^{\top}(\boldsymbol{A}\boldsymbol{A}^{\top} + \boldsymbol{\Sigma})^{-1}$$
$$\tilde{\boldsymbol{A}}^{-1} = \texttt{Linear}$$

1.3M vs 8k trainable parameters

[N. Ducros et al., IEEE ISBI, 2019]

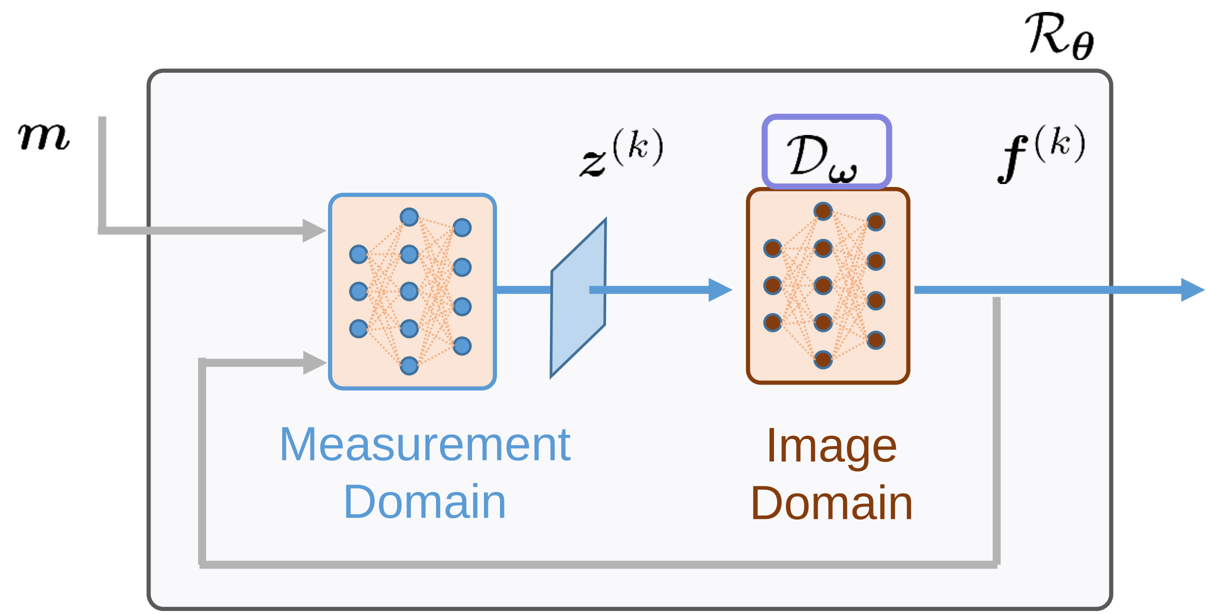
3.2. Unrolling (a.k.a. Unfolding)

➢ **Iterative methods**

$$\boldsymbol{z}^{(k)} = \boldsymbol{f}^{(k-1)} - \eta \boxed{\boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{f}^{(k-1)} - \boldsymbol{m})}$$
$$\boldsymbol{f}^{(k)} = \boxed{\texttt{prox}_{\lambda\Psi}}(\boldsymbol{z}^{(k)})$$

data fidelity

proximal operator



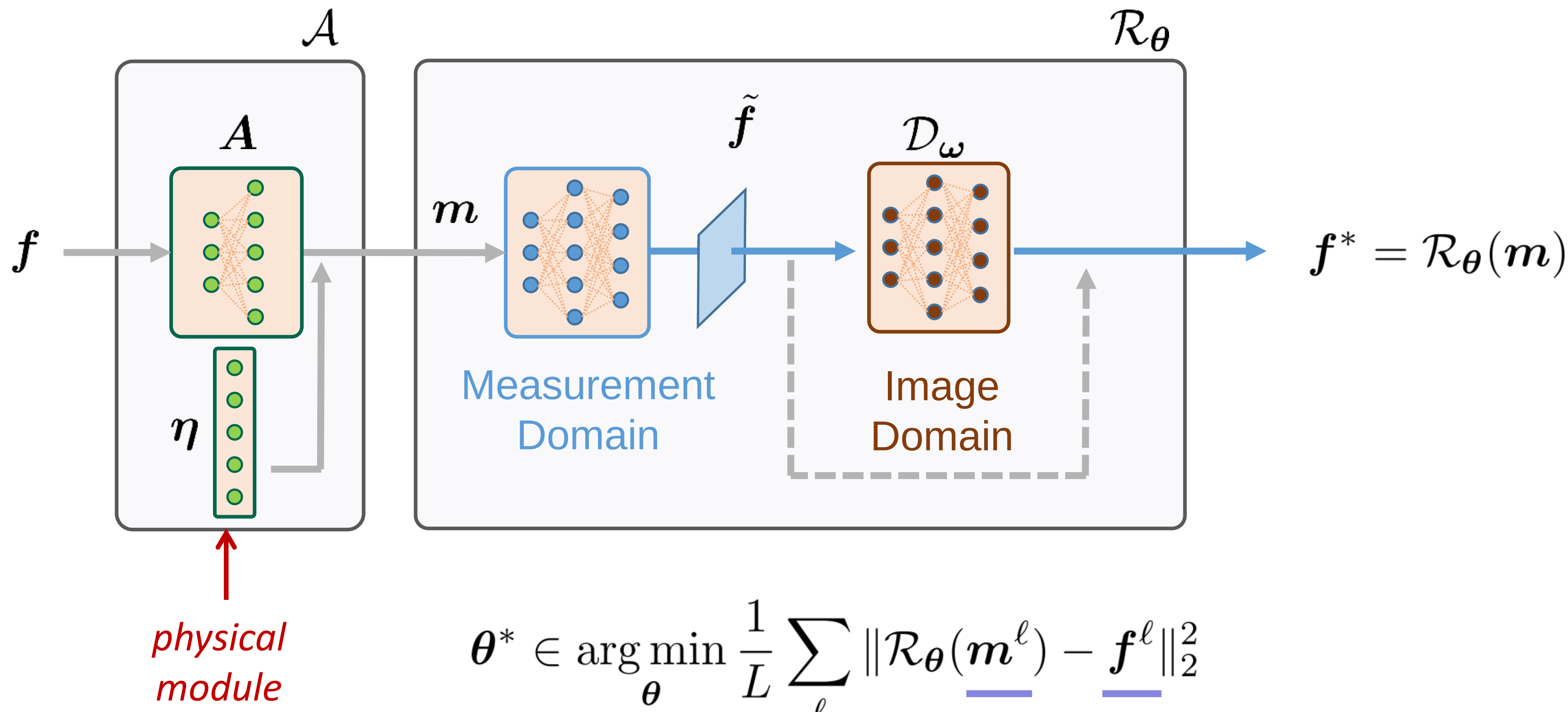
network parameters

$$\boldsymbol{f}^* = \mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{m})$$
$$= \boldsymbol{f}^{(K)}$$

Parameters can be <u>shared</u> across iterations <u>or not</u>

E.g., $\boldsymbol{\theta} = \boldsymbol{\omega}$,
or $\boldsymbol{\theta} = [\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(K)}]$.

- **With a "physical" module: no need for meas/image pairs**



$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \frac{1}{L} \sum_{\ell} \|\mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{m}^{\ell}) - \boldsymbol{f}^{\ell}\|_2^2$$

$$\mathcal{A}(\boldsymbol{f}) = \boldsymbol{A}\boldsymbol{f} + \boldsymbol{\eta}$$

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \frac{1}{L} \sum_{\ell} \|(\mathcal{R}_{\boldsymbol{\theta}} \circ \mathcal{A})(\boldsymbol{f}^{\ell}) - \boldsymbol{f}^{\ell}\|_2^2$$

3.4. Plug-and-Play

- **Idea: use denoisers (e.g., BM3D) in place of proximal operators**

$$\boldsymbol{z}^{(k)} = \boldsymbol{f}^{(k-1)} - \eta \boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{f}^{(k-1)} - \boldsymbol{m})$$

$$\boldsymbol{f}^{(k)} = \underbrace{\boxed{\mathtt{prox}_{\lambda\Psi}}}_{\text{proximal operator}}(\boldsymbol{z}^{(k)}) \quad \Longrightarrow \quad \boldsymbol{f}^{(k)} = \underbrace{\boxed{\mathtt{Denoi}}}_{\text{Denoiser}}(\boldsymbol{z}^{(k)}; \sigma)$$

Noise level

- **The denoiser can be data-driven. E.g.** $\mathtt{Denoi} = \mathtt{CNN}_{\boldsymbol{\theta}^*}$ **with**

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \frac{1}{L} \sum_{\ell} \|\mathtt{CNN}_{\boldsymbol{\theta}}(\boldsymbol{f}^\ell + \underline{\sigma\boldsymbol{\epsilon}}; \sigma) - \boldsymbol{f}^\ell\|_2^2$$

Gaussian noise with variance σ^2 $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(0,1)$

3.4. Plug-and-Play

- **Idea: use denoisers in place of proximal operators**

$$
\begin{aligned}
\boldsymbol{z}^{(k)} &= \boldsymbol{f}^{(k-1)} - \eta \boldsymbol{A}^{\top}(\boldsymbol{A}\boldsymbol{f}^{(k-1)} - \boldsymbol{m}) \\
\boldsymbol{f}^{(k)} &= \boxed{\texttt{Denoi}}(\boldsymbol{z}^{(k)}; \sigma)
\end{aligned}
$$

- **Pros**
  - Training is independent of the forward model
    - Flexibility
    - Applies to any inverse problem
  - Adapt to varying noise levels via hyperparameter
  - Convergence can be guaranteed

- **Cons**
  - Manual tuning of hyperparameter
  - Many iterations required compared to supervised methods (E.g., K = 100—1,000 vs K = 1—10)
    - Longer reconstruction times
    - Higher memory requirement
  - Convergence is not always guaranteed (e.g., Lipschitz constant of denoiser)
  - Underlying prior not always known

3.5. Untrained/unsupervised

- **Deep Generative Models**

$$f = \mathcal{R}_{\theta}(z)$$

Random vector

$$z^* \leftarrow \min_{z} \|A\mathcal{R}_{\theta}(z) - m\|_2^2$$

$$f^* = \mathcal{R}_{\theta}(z^*)$$

- **Pros**
  - Only requires measurements from a single acquisition
  - Theoretical guarantees (based on compressed sensing, e.g., considering Gaussian random matrices)
- **Cons**
  - Long and challenging reconstruction
  - Training of DGM is challenging (lots of data/long times)
  - Stability issues (arbitrary forward models, out-of-distribution images, etc.)

3.5. Untrained/unsupervised

- **Deep Image Priors**

$$f = \mathcal{R}_{\theta}(z)$$

Fixed random vector

$$\theta^* \leftarrow \min_{\theta} \|A\mathcal{R}_{\theta}(z) - m\|_2^2$$
$$f^* = \mathcal{R}_{\theta^*}(z)$$

Note: Minimization must be stopped before convergence (tends to noise otherwise)

- **Pros**
  - Only requires measurements from a single acquisition
  - The reconstruction quality is surprisingly good

- **Cons**
  - Long reconstruction times
  - No guarantees

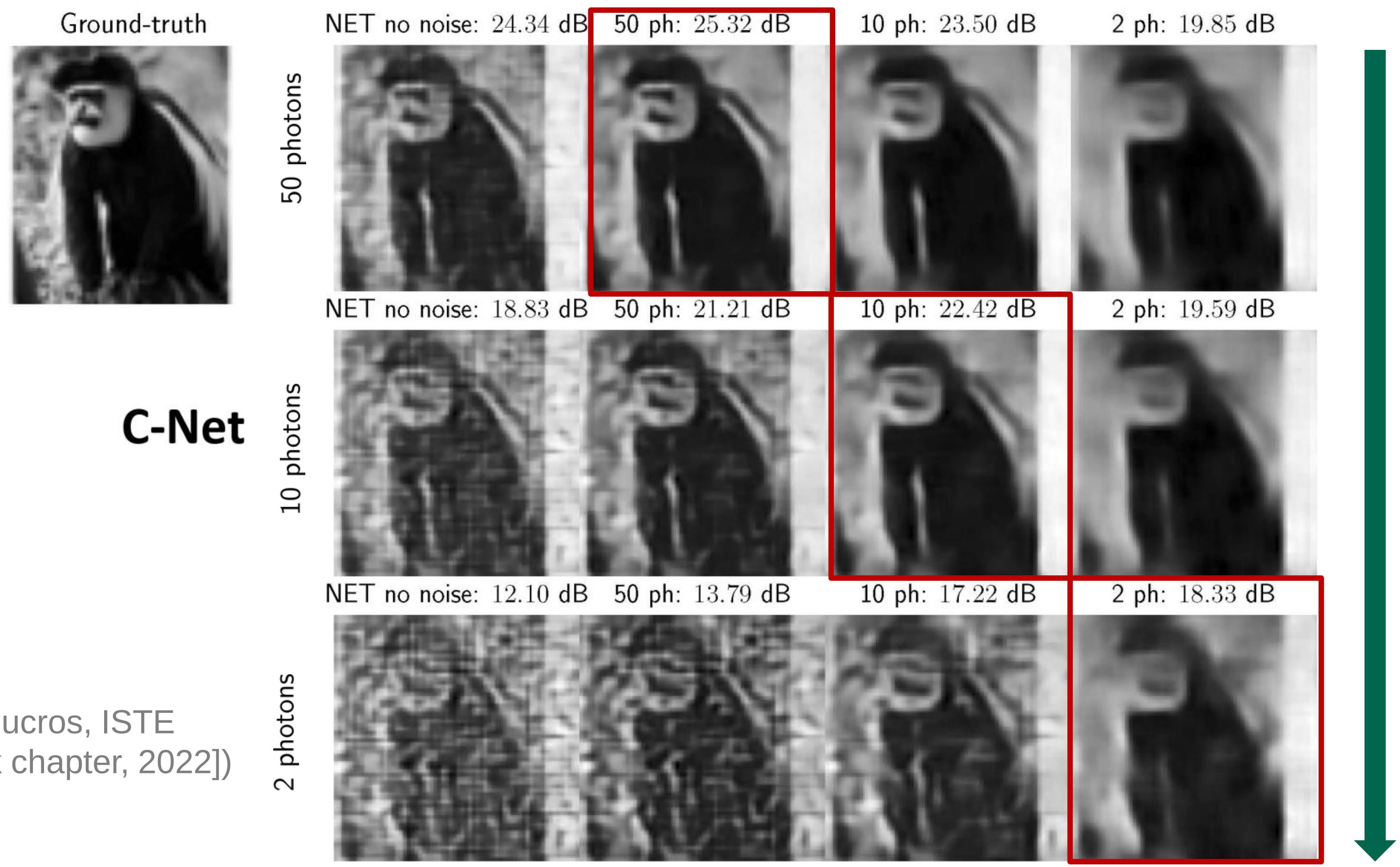
Conclusions

Benchmark

	Memory Requirement	Recon-struction Time (inference)	Training	Hyperparam/ Comment
Supervised	Low to intermediate	1—10	$\boldsymbol{A} \quad \{\boldsymbol{f}^\ell\}$	No adaptation (forward, noise)
PnP	Intermediate to high	100—1,000	$\{\boldsymbol{f}^\ell\}$	Noise level
Untrained	Usually low	> 1,000	—	Number of iterations

Noise robustness

Increasing training noise



Increasing test noise

[N. Ducros, ISTE Book chapter, 2022])

Conclusions

- **Data-driven DL-based approaches for image reconstruction are**
  - Powerful!
  - No longer black boxes
  - Supervised, PnP, based on generative models, untrained, etc.

- **Supervised vs PnP methods**
  - Supervised methods usually require fewer parameters
  - Supervised methods performs very well
  - PnP methods adapts to different
    - Imaging modalities (i.e., forward models)
    - Noise levels

- **Warning**
  - Handling noise is still an issue.
    - Evaluate the robustness to noise level deviations
    - Train with noise (supervised)
    - Tune hyperparameters (PnP)

→ Hands-on session on Friday at 2 pm! ←