Trainable Algorithms for Inverse Imaging Problems

Julien Mairal

Inria Grenoble
Collaborators

Bruno Lecouat
Jean Ponce
Part I: Lucas-Kanade reloaded: End-to-End Super-resolution from Raw Image Bursts

A 20-megapixel innocent scene
...taken at high ISO with low exposure time

Left: high-quality jpg output of the camera ISP.
...taken at high ISO with low exposure time

Left: high-quality jpg output of the camera ISP.
Right: ×4 super-resolution, after processing a burst of 30 raw images (handheld camera).
...taken at high ISO with low exposure time

Left: high-quality jpg output of the camera ISP.
Right: $\times 4$ super-resolution, after processing a burst of 30 raw images (handheld camera).
...taken at high ISO with low exposure time

Left: high-quality jpg output of the camera ISP.
Right: \( \times 4 \) super-resolution, after processing a burst of 30 raw images (handheld camera).
Our problem: multiframe super-resolution

low resolution frames

→

high-resolution image
... in contrast to single-image “super-resolution”

The problem is severely **ill-posed** and the goal is to “hallucinate” high frequencies.

*Figure: [Dahl et al., 2017]*
... in contrast to single-image “super-resolution”

The approach is data driven, and... not surprisingly...
The goal is to exploit **image misalignments** to artificially increase the number of samples from the underlying signals.

20 images are generated from the ground truth with synthetic random affine movements and average pooling downsampling.
The Camera raw processing pipeline (simplified view)

How does your camera process sensor data?

White balance.
Black subtraction.
Denoising
Demosaicking
Conversion to sRGB.
Gamma correction.

Idea: working with raw data is important, before the camera ISP produces irremediable damage!

Julien Mairal
The Camera raw processing pipeline (simplified view)

How does your camera process sensor data?

- White balance.
- Black substraction.
- Denoising
- Demosaicking
- Conversion to sRGB.
- Gamma correction.

Idea: working with raw data is important, before the camera ISP produces irremediable damage!
With raw data, we may leverage aliasing!

![Graph showing aliasing](image)

**Figure:** Example of aliasing: undersampled sinusoid causes confusion with a sinusoid with lower frequency. Picture from Wikipedia.

- Aliasing is usually mitigated with some optical / digital filters.
- If we analyze the aliasing patterns from multiple frames we can **recover high frequencies**.
Super-resolution from raw image bursts (with natural hand motion)

This is hard because it requires, simultaneously,

- accurately **aligning** images with subpixel accuracy.
- dealing with noisy data **(blind denoising)**.
- reconstructing color images from the Bayer pattern **(demosaicking)**.
Multiframe super resolution: prior work

- LR input image (1 of 4)
- Reconstruction
- Ground-truth HR image

- (Hardie et al., 1997)
- Bicubic interpolation

x4 alignment known exactly

(Baker and Kanade, 2002)
Multiframe super resolution: prior work

Handheld Multi-Frame Super-Resolution

BARTLOMIEJ WRONSKI, IGNACIO GARCIA-DORADO, MANFRED ERNST, DAMIEN KELLY, MICHAEL KRAININ, CHIA-KAI LIANG, MARC LEVOY, and PEYMAN MILANFAR, Google Research

Siggraph 2020; unknown motion, raw data.
Multiframe super resolution: prior work

Deep Burst Super-Resolution

Goutam Bhat  Martin Danelljan  Luc Van Gool  Radu Timofte
Computer Vision Lab, ETH Zurich, Switzerland
Multiframe super resolution: prior work

and, among many others:

- **interpolation-based methods**: [Hardie, 2007], [Takeda et al., 2007];
- **iterative approaches**: [Irani and Peleg, 1991], [Elad and Feuer, 1997], [Farsiu et al., 2004];
- **(deep) learning-based approaches**: [Bhat et al., 2021], [Molini et al., 2019], [Deudon et al., 2019];
- and also the literature on video super-resolution (typically not dealing with raw data).

**Interesting for us**: synthetic raw datasets from Bhat et al. [2021].
The “old” world of classical inverse problems.

Image formation model

\[ y_k = DBW_{p_k} x + \varepsilon_k. \]
The “old” world of classical inverse problems.

Inverse problem given \( y_1, \ldots, y_K \)

\[
\min_{x,p_k} \frac{1}{K} \sum_{k=1}^{K} \| y_k - DBW_{p_k} x \|_2^2 + \lambda \phi_\theta(x).
\]

A natural strategy

- define an appropriate prior \( \phi_\theta(x) \) for natural images.
- optimize!
The “old” world of classical inverse problems.

Inverse problem given \( y_1, \ldots, y_K \)

\[
\min_{x,p_k} \frac{1}{K} \sum_{k=1}^{K} \| y_k - DBW_{p_k} x \|_2^2 + \lambda \phi_\theta(x).
\]

A natural strategy

- define an appropriate prior \( \phi_\theta(x) \) for natural images.
- optimize!

Simple relaxation with “half quadratic splitting”

\[
\min_{x,z,p_k} \frac{1}{K} \sum_{k=1}^{K} \| y_k - U_{p_k} z \|_2^2 + \frac{\mu t}{2} \| z - x \|_2^2 + \lambda \phi_\theta(x).
\]
The “old” world of classical inverse problems.

Simple relaxation with “half quadratic splitting” + block coordinate descent

\[
\min_{x,z,p_k} \frac{1}{K} \sum_{k=1}^{K} \|y_k - U_{p_k} z\|^2 + \frac{\mu_t}{2} \|z - x\|^2 + \lambda \phi_\theta(x).
\]

- minimizing with respect to \( p_k \) (parameters of an affine transformation) is performed by Gauss-Newton steps. This is the algorithm of Lucas and Kanade [1981].
- minimizing with respect to \( x \) requires computing the proximal operator of \( \phi_\theta \).
- minimizing w.r.t. \( z \) can be done by gradient descent steps.
- \( \mu_t \) increases over the iterations.
The “old” world of classical inverse problems.

Simple relaxation with “half quadratic splitting” + block coordinate descent

\[
\min_{x,z,p_k} \frac{1}{K} \sum_{k=1}^{K} \|y_k - U_{p_k}z\|^2 + \frac{\mu_t}{2} \|z - x\|^2 + \lambda \phi_\theta(x).
\]

- minimizing with respect to \( p_k \) (parameters of an affine transformation) is performed by Gauss-Newton steps. This is the algorithm of Lucas and Kanade [1981].
- minimizing with respect to \( x \) requires computing the proximal operator of \( \phi_\theta \).
- minimizing w.r.t. \( z \) can be done by gradient descent steps.
- \( \mu_t \) increases over the iterations.

Advantage: robustness and interpretability (solves what it is supposed to solve).
Drawback: designing a good image prior by hand is hard
The “new” world of deep learning models (Pic. https://xkcd.com/)

- a form of prior knowledge is encoded in the model architecture (e.g., a convolutional neural network for images).
- ability to train model parameters $\theta$ end to end.
- state-of-the-art for many tasks (once the right model/setup is found).
- requires training data.

**Advantage:** task-adaptive.

**Drawback:** tuned to specific data distribution.
Bridging the two worlds with trainable algorithms.

Idea 1: plug-and-play priors [Venkatakrishnan et al., 2013]
Replace proximal operator
\[
\arg\min_x \frac{1}{2} \|z - x\|^2 + \lambda \phi_\theta(x),
\]
by a convolutional neural network \( f_\theta(x) \).
Bridging the two worlds with trainable algorithms.

**Idea 1: plug-and-play priors [Venkatakrishnan et al., 2013]**

Replace proximal operator

\[
\arg\min_x \frac{1}{2} \| z - x \|^2 + \lambda \phi_\theta(x),
\]

by a convolutional neural network \( f_\theta(x) \).

**Idea 2: unrolled optimization [Gregor and LeCun, 2010]**

- Consider the previous optimization procedure with \( T \) steps, producing an estimate \( \hat{x}_T(Y) \), given a burst \( Y = y_1, \ldots, y_K \).
- Given a dataset of training pairs \( (x_i, Y_i)_{i=1,\ldots,n} \), minimize

\[
\min_\theta \frac{1}{n} \sum_{i=1}^n \| \hat{x}_T(Y_i) - x_i \|_1.
\]
we keep the interpretability of the classical inverse problem formulation.

we benefit from a data-driven image prior.
Do we get the best or the worse of both worlds?

Figure: Experiment with a synthetic RGB burst of 20 images with random affine motions.
Extreme $\times 16$ super-resolution.

Figure: Experiment with a synthetic RGB burst of 20 images with random affine motions.
Experiments on real raw data - Pixel 4a.

Figure: Full scene - camera ISP - Our $\times 4$ results.
Experiments on real raw data - Pixel 4a.

**Figure:** Full scene - camera ISP - Our $\times 4$ results.
Experiments on real raw data - Samsung S7.

Figure: Full scene - camera ISP - Our $\times 4$ results.
Experiments on real raw data - Panasonic GX9.

Figure: Full scene - camera ISP - Our $\times 4$ results.
Quantitative experiments.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR (db)</th>
<th>Geom (pix)</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scores on public validation set</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETH Bhat et al. [2021]</td>
<td>39.09</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours (refine)</td>
<td>41.45</td>
<td>-</td>
<td>0.95</td>
</tr>
<tr>
<td><strong>Scores on our own validation set to conduct the ablation study</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiframe + TV</td>
<td>34.48</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Single Image</td>
<td>36.80</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours (no refinements)</td>
<td>40.38</td>
<td>0.55</td>
<td>0.958</td>
</tr>
<tr>
<td>Ours (refinements)</td>
<td>41.30</td>
<td>0.32</td>
<td>0.963</td>
</tr>
<tr>
<td>Ours (known motion)</td>
<td>42.41</td>
<td>0.00</td>
<td>0.971</td>
</tr>
</tbody>
</table>

Table: **Results with synthetic raw image bursts** of 14 images generated from the Zurich raw to RGB dataset with synthetic affine motions. Results in average PSNR and geometrical registration error in pixels for our models.
Current issues with moving objects

Figure: Misalignements artefacts due to moving objects in the scene. Our current implementation does not handle fast moving objects and then generates visual artefacts.
Conclusion

Take-home messages

- 40-years old computer vision algorithms are useful.
- aliasing is good.
- “classical” approaches are robust and interpretable and greatly benefit from deep learning principles (differentiable programming).

Future work

- microscopy and astronomical imaging where we want to recover “true” signals.
- high-quality and high-dynamic range panoramas.
- going beyond static scenes.
Part II: End-to-End Sparse Coding Models

Image denoising: classical image models

\[ y = x_0 + \varepsilon. \]

Energy minimization problem - MAP estimation

\[ E(x) = \frac{1}{2} \| y - x \|_2^2 + \psi(x). \]

Some classical priors

- Smoothness \( \lambda \| \mathcal{L} x \|_2^2 \);
- total variation \( \lambda \| \nabla x \|_1 \) [Rudin et al., 1992];
- Markov random fields [Zhu and Mumford, 1997];
- wavelet sparsity \( \lambda \| D^T x \|_1 \).
Image denoising

The method of Elad and Aharon [2006]

Given a fixed dictionary $D$, a patch $y_i$ (e.g., $8 \times 8$) is denoised as follows:

1. center $y_i$,

   $y_i^c \triangleq y_i - \mu_i 1_m$ with $\mu_i \triangleq \frac{1}{n} 1_m^T y_i$;

2. find a sparse linear combination of dictionary elements that approximates $y_i^c$ up to the noise level:

   $$\min_{\alpha_i \in \mathbb{R}^p} \|\alpha_i\|_0 \text{ s.t. } \|y_i^c - D\alpha_i\|_2^2 \leq \varepsilon,$$  

   where $\varepsilon$ is proportional to the noise variance $\sigma^2$;

3. add back the mean component to obtain the clean estimate $\hat{x}_i$:

   $$\hat{x}_i \triangleq D\alpha_i^* + \mu_i 1_m,$$
Image denoising
The method of Elad and Aharon [2006]

An \textit{adaptive} approach

1. extract all overlapping $\sqrt{m} \times \sqrt{m}$ patches $y_i$.
2. \textbf{dictionary learning}: learn $D$ on the set of centered noisy patches $[y_1^c, \ldots, y_n^c]$.
3. \textbf{final reconstruction}: find an estimate $\hat{x}_i$ for every patch using the approach of the previous slide;
4. \textbf{patch averaging}: $\hat{x} = \frac{1}{m} \sum_{i=1}^{n} R_i^\top \hat{x}_i$. 
Parenthesis on sparsity-inducing penalties

\[ \|\alpha\|_1 \leq 1 \]

\( \ell_1 \)-ball

\[ \alpha[1] \]

\[ \alpha[2] \]
Parenthesis on sparsity-inducing penalties

\[ \| \alpha \|_2^2 \leq 1 \]
Parenthesis on sparsity-inducing penalties

\[(1 - \gamma)\|\alpha\|_1 + \gamma\|\alpha\|_2^2 \leq 1\]
Parenthesis on sparsity-inducing penalties

\[ \| \alpha \|_1 \leq 1 \]

\( \ell_1 \)-ball

\( \alpha[1] \)

\( \alpha[2] \)
Parenthesis on sparsity-inducing penalties

$$\|\alpha\|_q \leq 1 \text{ with } q < 1$$
Other patch modeling approaches

Non-local means and non-parametric approaches

Image pixels are well explained by a Nadaraya-Watson estimator:

\[
\hat{x}[i] = \frac{\sum_{j=1}^{n} K_h(y_i - y_j) y[j]}{\sum_{l=1}^{n} K_h(y_i - y_l) y[j]},
\]

(2)

with successful application to

- texture synthesis: [Efros and Leung, 1999]
- image denoising (Non-local means): [Buades et al., 2005]
- image demosaicking: [Buades et al., 2009].
Other patch modeling approaches

**BM3D**

a state-of-the-art image denoising approach [Dabov et al., 2007]:

- **block matching**: for each patch, find similar ones in the image;
- **3D wavelet filtering**: denoise blocks of patches with 3D-DCT;
- **patch averaging**: average estimates of overlapping patches;
- **second step with Wiener filtering**: use the first estimate to perform again and improve the previous steps.

Further refined by Dabov et al. [2009] with shape-adaptive patches and PCA filtering.
Other patch modeling approaches

Non-local sparse models [Mairal et al., 2009]

Combine the non-local means principle with dictionary learning.

The main idea is that **similar patches should admit similar decompositions** by using group sparsity:

The approach uses group sparse coding and patch averaging.
How to derive a trainable algorithm from sparse coding principles?

Consider the Lasso problem

\[
\min_{\alpha \in \mathbb{R}^p} \frac{1}{2} \| x - D\alpha \|^2 + \lambda \| \alpha \|_1
\]

A classical algorithm to solve the optimization problem is the proximal gradient descent method

\[
\alpha_t \leftarrow \text{prox}_{\lambda \| \cdot \|_1} \left[ \alpha_{t-1} - \eta D^\top (D\alpha_{t-1} - x) \right]
\]

This motivates the LISTA approach consisting of unrolling \( T \) steps of

\[
\alpha_t \leftarrow \text{prox}_{\lambda \| \cdot \|_1} \left[ \alpha_{t-1} + C^\top (D\alpha_{t-1} - x) \right],
\]

and see the resulting iterate \( \alpha_T(x) \) as a parametric function of \( D \) and \( C \).
How to derive a trainable algorithm from sparse coding principles?

Consider the Lasso problem

$$\min_{\alpha \in \mathbb{R}^p} \frac{1}{2} \| x - D\alpha \|^2 + \lambda \| \alpha \|_1$$

A classical algorithm to solve the optimization problem is the proximal gradient descent method

$$\alpha_t \leftarrow \text{prox}_{\lambda \| \cdot \|_1} \left[ \alpha_{t-1} - \eta D^\top (D\alpha_{t-1} - x) \right]$$

This motivates the LISTA approach consisting of unrolling $T$ steps of

$$\alpha_t \leftarrow \text{prox}_{\lambda \| \cdot \|_1} \left[ \alpha_{t-1} + C^\top (D\alpha_{t-1} - x) \right],$$

and see the resulting iterate $\alpha_T(x)$ as a parametric function of $D$ and $C$.

**Remark:** One step performs an affine transformation of $\alpha_{t-1} + \text{prox}$. 

How to derive a trainable algorithm from sparse coding principles?

Consider the Lasso problem

$$\min_{\alpha \in \mathbb{R}^p} \frac{1}{2} \|x - D\alpha\|^2 + \lambda \|\alpha\|_1$$

A classical algorithm to solve the optimization problem is the proximal gradient descent method

$$\alpha_t \leftarrow \text{prox}_{\lambda \|\cdot\|_1} \left[ \alpha_{t-1} - \eta D^\top (D\alpha_{t-1} - x) \right]$$

This motivates the LISTA approach consisting of unrolling $T$ steps of

$$\alpha_t \leftarrow \text{prox}_{\lambda \|\cdot\|_1} \left[ \alpha_{t-1} + C^\top (D\alpha_{t-1} - x) \right],$$

and see the resulting iterate $\alpha_T(x)$ as a parametric function of $D$ and $C$.

**Remark:** One step performs an affine transformation of $\alpha_{t-1} + \text{prox}.$

**Remark 2:** $\text{prox}_{\lambda \|\cdot\|_1}[u] = \text{sign}(u) \ast \text{RELU}(|u| - \lambda)$
End-to-end sparse coding models

Denoising with the $\ell_1$-norm

Then, we can “unroll” the patch-based sparse coding denoising approach and learn the matrices $D$ and $C$ in a supervised fashion, given training pairs of noisy/clean images, as done by Simon and Elad [2019].

Denoising with non-local sparse models (our work)

- deal with the **proximal operator of the Group Lasso penalty**.
- **take into account self-similarities** via similarity matrix $\Sigma$.
- end-to-end learning with unrolled optimization.
End-to-end sparse coding models

**Algorithm 1** Pseudo code for the inference model of GroupSC.

1: Extract patches $Y = [y_1, \ldots, y_n]$ and center them;
2: Initialize the codes $\alpha_i$ to 0;
3: Initialize image estimate $\hat{x}$ to the noisy input $y$;
4: Initialize pairwise similarities $\Sigma$ between patches of $\hat{x}$;
5: for $k = 1, 2, \ldots K$ do
6: Compute pairwise patch similarities $\hat{\Sigma}$ on $\hat{x}$;
7: Update $\Sigma \leftarrow (1 - \nu)\Sigma + \nu\hat{\Sigma}$;
8: for $i = 1, 2, \ldots, N$ in parallel do
9: $\alpha_i \leftarrow \text{prox}_{\Sigma, \Lambda_k} \left[ \alpha_i + C^\top (y^c_i - D\alpha_i) \right]$;
10: end for
11: Update the denoised image $\hat{x}$ by averaging;
12: end for
For the young generation

\[
\begin{align*}
&y_{H \times W \times C} \\
&y_{c_{H \times W \times N}} \\
&\text{unfolding (im2col)} \\
&\text{Proximal iteration} \\
&\text{Self-similarities} \\
&\Sigma^{(i)}_{(H \times W) \times (H \times W)} \\
&\text{prox group lasso} \\
&\text{conv2d } 1 \times 1 \text{ (C)} \\
&B \\
&\text{Conv2d } 1 \times 1 \text{ (D)} \\
&A_{H \times W \times p} \\
&\text{convtranspose2d } s \times s \text{ (W)} \\
&\tilde{x}_{H \times W \times C} \\
&\text{Non-local self-similarity module} \\
&A^{(i)}_{H \times W \times p} \\
&\Sigma^{(i)}_{(H \times W) \times (H \times W)} \\
&\text{convtranspose2d } s \times s \text{ (W)} \\
&\tilde{x}_{H \times W \times C} \\
&\text{unfolding} \\
&\text{patch pairwise distance} \\
&1 - v_{H \times W \times N} \\
&\Sigma^{(i+1)}_{(H \times W) \times (H \times W)} \\
&v_{H \times W \times N} \\
&\Sigma^{(i+1)}_{(H \times W) \times (H \times W)}
\end{align*}
\]
Demosaicking. Sparse coding vs. non-local sparse coding
Denoising. Sparse coding vs. non-local sparse coding
JPEG Deblocking. Sparse coding vs. non-local sparse coding
Interesting conclusion: parameter-efficient models

Table: Demosaicking. Training on CBSD400 unless a larger dataset is specified between parenthesis. Performance is measured in terms of average PSNR. SSIMs are reported in the appendix.

<table>
<thead>
<tr>
<th>Method</th>
<th>Trainable</th>
<th>Params</th>
<th>Kodak24</th>
<th>BSD68</th>
<th>Urban100</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSSC</td>
<td>X</td>
<td>-</td>
<td>41.39</td>
<td>40.44</td>
<td>36.63</td>
</tr>
<tr>
<td>IRCNN (BSD400+Waterloo)</td>
<td></td>
<td>-</td>
<td>40.54</td>
<td>39.9</td>
<td>36.64</td>
</tr>
<tr>
<td>Kokinos (MIT dataset)</td>
<td>380k</td>
<td>41.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MMNet (MIT dataset)</td>
<td>380k</td>
<td>42.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RNAN</td>
<td>8.96M</td>
<td>42.86</td>
<td>42.61</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SC (ours)</td>
<td>119k</td>
<td>42.34</td>
<td>41.88</td>
<td>37.50</td>
<td></td>
</tr>
<tr>
<td>CSR (ours)</td>
<td>119k</td>
<td>42.25</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>GroupSC (ours)</td>
<td>119k</td>
<td>42.71</td>
<td>42.91</td>
<td>38.21</td>
<td></td>
</tr>
</tbody>
</table>
Interesting conclusion: parameter-efficient models

Table: **Grayscale Denoising** on BSD68, training on BSD400 for all methods. Performance is measured in terms of average PSNR. SSIMs are reported in the appendix.

<table>
<thead>
<tr>
<th>Method</th>
<th>Trainable</th>
<th>Trainable Params</th>
<th>Noise Level (σ)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>BM3D</td>
<td>X</td>
<td>-</td>
<td>37.57</td>
</tr>
<tr>
<td>LSSC</td>
<td>X</td>
<td>-</td>
<td>37.70</td>
</tr>
<tr>
<td>BM3D PCA</td>
<td>X</td>
<td>-</td>
<td>37.77</td>
</tr>
<tr>
<td>TNRD</td>
<td></td>
<td>-</td>
<td>31.42</td>
</tr>
<tr>
<td>CSCnet</td>
<td></td>
<td>62k</td>
<td>37.84</td>
</tr>
<tr>
<td>LKSVD</td>
<td></td>
<td>45k</td>
<td>-</td>
</tr>
<tr>
<td>FFDNet</td>
<td></td>
<td>486k</td>
<td>-</td>
</tr>
<tr>
<td>DnCNN</td>
<td></td>
<td>556k</td>
<td>37.68</td>
</tr>
<tr>
<td>NLRN</td>
<td></td>
<td>330k</td>
<td>37.92</td>
</tr>
<tr>
<td>SC (baseline)</td>
<td></td>
<td>68k</td>
<td>37.84</td>
</tr>
<tr>
<td>GroupSC (ours)</td>
<td></td>
<td>68k</td>
<td><strong>37.95</strong></td>
</tr>
</tbody>
</table>
Interesting conclusion: leveraging interpretability

Table: **Blind denoising** on CBSD68, training on CBSD400. Performance is measured in terms of average PSNR. SSIMs are in the appendix. Best is in bold, second is underlined.

<table>
<thead>
<tr>
<th>Noise level</th>
<th>CBM3D</th>
<th>CDnCNN-B</th>
<th>CUNet</th>
<th>CUNLnet</th>
<th>SC (ours)</th>
<th>GroupSC (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>40.24</td>
<td>40.11</td>
<td>40.31</td>
<td>40.39</td>
<td>40.30</td>
<td>40.43</td>
</tr>
<tr>
<td>10</td>
<td>35.88</td>
<td>36.11</td>
<td>36.08</td>
<td>36.20</td>
<td>36.07</td>
<td><strong>36.29</strong></td>
</tr>
<tr>
<td>15</td>
<td>33.49</td>
<td>33.88</td>
<td>33.78</td>
<td>33.90</td>
<td>33.72</td>
<td><strong>34.01</strong></td>
</tr>
<tr>
<td>20</td>
<td>31.88</td>
<td>32.36</td>
<td>32.21</td>
<td>32.34</td>
<td>32.11</td>
<td><strong>32.41</strong></td>
</tr>
<tr>
<td>25</td>
<td>30.68</td>
<td><strong>31.22</strong></td>
<td>31.03</td>
<td>31.17</td>
<td>30.91</td>
<td><strong>31.25</strong></td>
</tr>
</tbody>
</table>

- learn common $D, C$ parameters for different noise levels.
- learn noise-specific regularization parameters $\lambda$

$$\min_{\alpha \in \mathbb{R}^p} \frac{1}{2} \| x - D\alpha \|^2 + \lambda \| \alpha \|_1.$$
Conclusion

On trainable algorithms
- this is a hybrid point of view between deep learning black boxes and classical inverse problem formulations.
- This is a natural way to encode a priori knowledge in the model and obtain smaller models.
- Interpretability is useful!

Caveats
- unrolled optimization is often unstable and requires heuristics for training.
- not much theory (leading to exciting new challenges).
References I


References II


