Medical Imaging - Deep Learning Based Image Reconstruction

Institute for Biomedical Imaging, Hamburg University of Technology

- Prof. Dr.-Ing. Tobias Knopp

Table of Contents

1. Introduction

1.1 Classical Regularization

From the lecture on inverse problem we know that *prior knowledge* is important when solving inverse problems. Prior knowledge can help us to

- reduce the noise amplification caused by the inversion of the linear system
- enforce uniqueness in case of undersampling ($\rightarrow$ compressed sensing)

But how did this look like:

Tikhonov Regularization

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \underbrace{\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2}_{\text{data consistency}} + \underbrace{\lambda\|\boldsymbol{x}\|_2^2}_{\text{prior knowledge}}$$

ℓ_1 Regularization

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$$

Total-Variation Regularization

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 + \eta TV(\boldsymbol{x})$$

where $TV(\boldsymbol{x})$ is the total variation, i.e. a norm that measures edges.

Fused-Lasso Regularization

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 + \eta TV(\boldsymbol{x})$$

ℓ_1 Regularization with Sparsifying Transform

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{W}\boldsymbol{x}\|_1$$

What is the characteristic of our prior knowledge?

- In the first case we basically just want to avoid that our solution is *exploding*.
- In the second case we seek for a solution with only few non-zero entries.
- In the third case we seek for a solution with only few edges.
- In the forth case we bring additional prior knowledge in, i.e. that the wavelet coefficients for natural images are sparse.

What are we doing in all cases?

We try to characterize the solution space by penalizing certain characteristics of the solution $\boldsymbol{x}$.

> **Note**
>
> These types of regularization are very generic because we only describe the solution using certain characteristics of the function (so-called features). For instance we know that noise has high frequencies and thus many sharp edges. Therefore applying a TV penality reduces the amount of noise. But what if $\boldsymbol{x}$ itself contains sharp edges? Then our prior knowledge can be wrong.
>
> These classical approaches are said to be *feature driven* while the machine learning approaches we discuss later are *data driven*.

1.2 Time Complexity

Let us next take a look at the time complexity of classical regularization methods.

Tikhonov Regularization

The time complexity for applying Tikhonov regularization is between $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$. The later is an upper bound when using e.g. the SVD. The former is a lower bound, e.g. when applying one CGNR iteration or one *outer* Kaczmarz iteration. In practice, iterative solvers are somewhere in-between.

Advanced Regularization

Advanced regularization techniques are non-linear. They usually require an iterative algorithm, which commonly converges much slower than linear ones. Their complexity is $\mathcal{O}(PN^2)$ where P is the number of iterations. While the algorithmic complexity is the same as for Tikhonov regularization, in practice P is usually a factor of $10 - 1000$ larger. This in practice is a real problem and the reason why these methods are popular in research but often not integrated into imaging devices.

1.3 Parameter Choice

Finally, one key issue with classical regularization is the dependency on the regularization parameter(s). While there are methods for optimizing the regularization parameter (e.g. the L-curve), in practice regularization is most often done by hand. But this, again, makes it unsuitable for clinical application, where the image generation needs to be done automatically.

1.4 Summary of Challenges

We identified three major challenges for classical image reconstruction:

1. The form of prior knowledge is just a heuristic and does not describe our solution space well.
2. While Tikhonov regularization is fast, more advanced techniques like TV, Fused-Lasso or ℓ_1-Wavelet regularization are very expensive.
3. Parameter choice is a major problem.

All of these points can be tackled by machine learning as we will see later. Let's first introduce machine learning.

2. Machine Learning

We here only roughly introduce machine learning. Let us consider a function $f : \mathbb{R}^N \to \mathbb{R}^M$, i.e.

$$f(x) = y,$$

and let us assume that we don't know f but we only know L data pairs $(x_l, y_l), l = 1, \dots, L$ with $f(x_l) = y_l$. We call this the training dataset. Then, the task that is solved by *machine learning* is to approximate f by a function $f_\theta : \mathbb{R}^N \to \mathbb{R}^M$, i.e.

$$f_\theta(x) = y,$$

where $\theta \in \mathbb{R}^Q$ are parameters that need to be optimized. To find suitable parameters one commonly solves an optimization problem that looks like

$$\sum_{l=1}^{L} D(f_\theta(x_l), y_l) \overset{\theta}{\to} \min,$$

where D is a distance measure. For instance one can use the mean squared error (MSE)

$$D_{\text{MSE}}(\tilde{y}, y) = \|\tilde{y} - y\|_2^2.$$

The aim now is that $f(x) \approx f_\theta(x)$ not only for the training data but also for unseen data.

> **Note**
>
> Machine learning is thus basically function approximation. It is related to interpolation but it is not required that $f(x) = f_\theta(x)$ for $x_l, l = 1, \dots, L$.
>
> Thus, ML is very similar to classical function approximation where a certain basis (polynomials, splines, ...) are used as a model f_θ.

2.1 Artificial Neural Networks

There are many machine learning methods. Those which got very popular in the last decade are artificial neural networks (ANN). These consists of several layers to which certain operations are applied. A simple ANN with two layers can be formulated as

$$f_{\text{ANN}}(\boldsymbol{x}) = \sigma(\boldsymbol{H}_2(\sigma(\boldsymbol{H}_1\boldsymbol{x} + \boldsymbol{c}_1)) + \boldsymbol{c}_2)$$

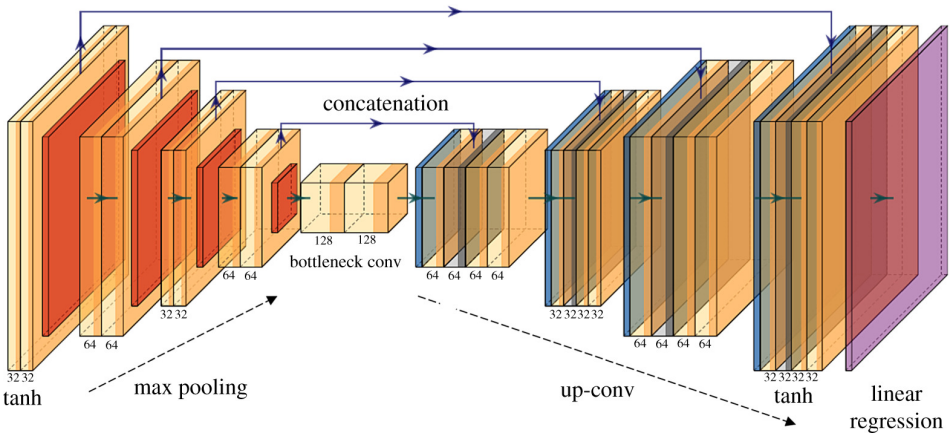
with

- $\boldsymbol{H}_1, \boldsymbol{H}_2$ being matrices transforming the input vector (linear transformations). These can be *dense matrices* (dense layers) or *convolutions* (conv layers).
- $\boldsymbol{c}_1, \boldsymbol{c}_2$ being bias vectors.
- σ being a non-linear activation function.

> **Note**
>
> An artificial neural network is said to be deep if it contains various nested layers. A network is said to be wide if it has many connections/parameters within a layer (i.e. a dense layer).

Example

The following showcases a so-called decoder/encoder ANN, which consists of several convolution layers and down/upsampling operations.



We next outline different ways to use ANNs for image reconstruction. In all cases we consider

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

to be our inverse problem and we consider that we have training samples $(\boldsymbol{x}_l, \boldsymbol{b}_l)$, $l = 1, \dots, L$.

3. Machine Learning for Image Reconstruction

In the following we sketch some methods to use machine learning for image reconstruction. The overview is not meant to be complete since the field of ML-based image reconstruction is still evolving.

3.1 Learn the Inverse Directly

The first idea is to directly learn the the inverse. This means we formulate

$$f_\theta^{\text{inverse}}(\boldsymbol{b}) = \boldsymbol{x}$$

and optimize the parameters $\boldsymbol{\theta}$.

Pros

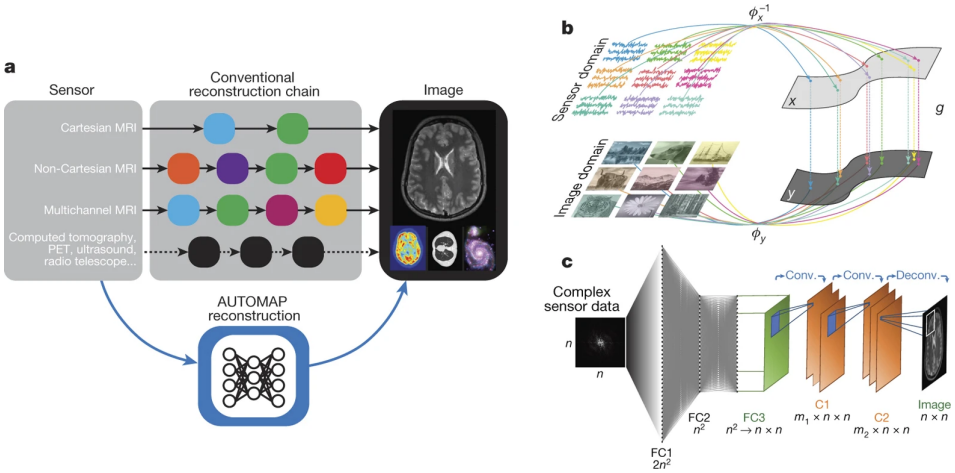
- We can take all imperfections into account since we don't assume any (simplified) physical model.

Cons

- We don't use information of the imaging operator, i.e. we learn things that we already know. In the case of MRI the ANN $f_\theta^{\text{inverse}}$ needs to learn the Fourier transform.
- The ANN needs to be *expressive* enough to learn the (inverse) imaging operator. This requires dense layers, which makes training much more complicated (both from the accuracy and the training time/memory point of view).
- In order to train such an ANN it needs a very large training dataset.

Example AUTOMAP

See: Zhu, B., Liu, J., Cauley, S. et al. Image reconstruction by domain-transform manifold learning. Nature 555, 487–492 (2018)



3.2 Postprocessing Networks

Learning the inverse directly is not really the best option. We next consider models that use a traditional image reconstruction in its core and afterwards apply the ANN in a post-processing step.

So let's assume $f^{\text{classic reco}}(\boldsymbol{b})$ is a classical image reconstruction method. For instance the FFT in the case of MRI and the FBP in the case of CT. Then a postprocessing network would be formulated as

$$f_\theta^{\text{postprocessing}}(f^{\text{classic reco}}(\boldsymbol{b})) = \boldsymbol{x}$$

Looks very similar but the important difference is that $f_\theta^{\text{postprocessing}}$ is an ANN acting in image space, which is a well known task in the field of image processing. Hence, well known network architectures can be used here.

Pros

- Less complex model needed. CNNs are sufficient.
- Less training data needed.
- Physics is taken into account.

Cons

- Not as generic/powerful than learning the inverse.

3.3 Use Neural Networks with Iterative Solver

Advanced regularization techniques usually use iterative solvers like FISTA, ADMM, CG(NR), or the Kaczmarz method. Let us for the moment take a much more basic approach: the Landweber iteration. Similar to the other methods it aims at solving the least squares problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \text{ where } f(\boldsymbol{x}) := \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2.$$

Since we want to minimize $\boldsymbol{f}$ we can iterate over $\boldsymbol{x}$ and follow the gradient in order to minimize $\boldsymbol{f}$, i.e. we use gradient descent:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \omega \nabla f(\boldsymbol{x}_k)$$

Here, ω is a relaxation factor which needs to be appropriately chosen (not discussed at this point).

> **Note**
>
> Side note: The conjugate gradient method follows the same approach but does not use the steepest decent but follows conjugate directions, i.e. the history of descents is taken into account, which accelerates convergence.

For our specific $\boldsymbol{f}$ we can calculate the gradient explicitly

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \omega \boldsymbol{A}^H(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b}).$$

We can more generally say that this form of iteration is written as

$$\boldsymbol{x}_{k+1} = \mathrm{DC}(\boldsymbol{x}_k)$$

where DC is the function that ensures *data consistency*, i.e. the minimization of the residual. With this more general formulation, DC could for instance also be one *outer* Kaczmarz iteration, i.e. one sweep over all rows.

With this knowledge we can integrate machine learning into iterative algorithms. Two different variants that are discussed next.

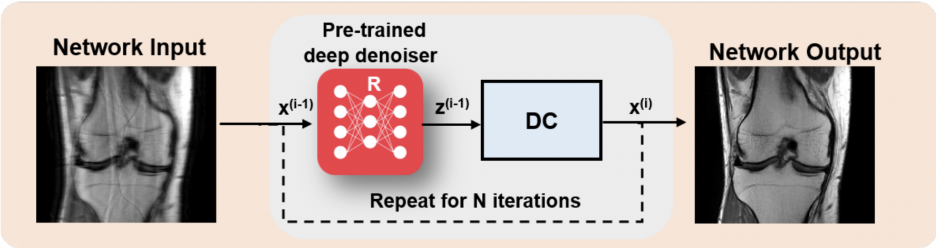
3.3.1 Plug-and-Play Approach

The easiest and most robust way is to train a neural network for a specific task like denoising, super-resolution or in general *image enhancement*. Then we apply the enhancement neural network after each DC step:

$$\boldsymbol{x}_{k+1} = f_\theta^{\text{enhance}}(\text{DC}(\boldsymbol{x}_k)).$$

We call this *plug and play* since the neural network is not trained for the actual image reconstruction but outside the reconstruction pipeline.

The following picture is taken from this publication and graphically showcases the plug-and-play approach:



3.3.2 Unrolled Iteration

The plug-and-play approach can be refined by training the network within the reconstruction problem. In order to do so, one needs to use a fixed number of iterations and *unroll* the network.

For instance for $P = 3$ iterations we can do this by:

$$\boldsymbol{x}_{\text{final}} = f_{\theta_3}^{\text{enhance}}(\text{DC}(f_{\theta_2}^{\text{enhance}}(\text{DC}(f_{\theta_1}^{\text{enhance}}(\text{DC}(\boldsymbol{x}_0))))).$$

Now we can train $\boldsymbol{\theta_1}$, $\boldsymbol{\theta_2}$, and $\boldsymbol{\theta_3}$ in an end-to-end fashion. Here, there are two variants:

- One can use different NN parameters in each iteration.
- One can share the weights.

The former is more powerful but more difficult to train because it has more parameters.

Note

What might not be obvious is that the unrolled iteration can also significantly speed up the reconstruction process. The reason is that the neural networks are not only able to do image enhancement in the classical sense (i.e. denoising) but they can also generate short paths. This is because the number of iterations is fixed and known during training such that the algorithm is forced to converge in the predefined number of iterations. In practice, often just about $P = 10$ iterations are needed.

Note

All ML-based reconstruction methods we outlined were parameter-free. They need to be trained on different noise classes and then usually can automatically adapted to a certain noise class. This adresses a further challenge of the feature-based reconstruction methods.

4. Summary

In this short lecture we have discussed the challenges of classical (feature-based) image reconstruction and regularization techniques. We then outlined some ways how machine learning can be used to provide a tailored domain specific form of prior knowledge. You will need some first experience in machine learning and the respective frameworks (Python: PyTorch, TensorFlow; Julia: Flux.jl, Lux.jl) before you can start implementing the methods discussed in this lecture.