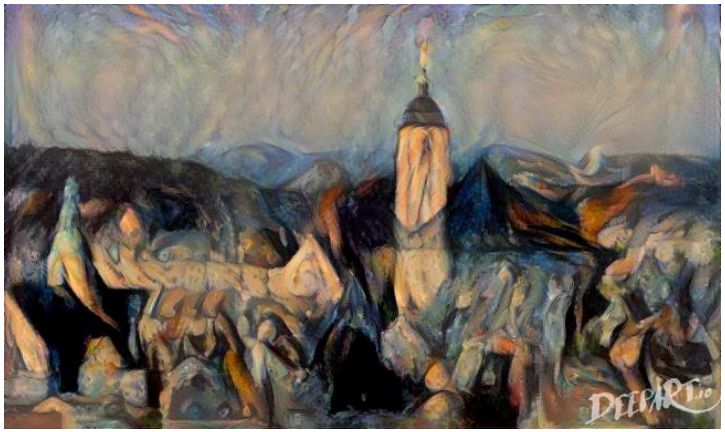
Fusing learning and model-based reconstruction techniques

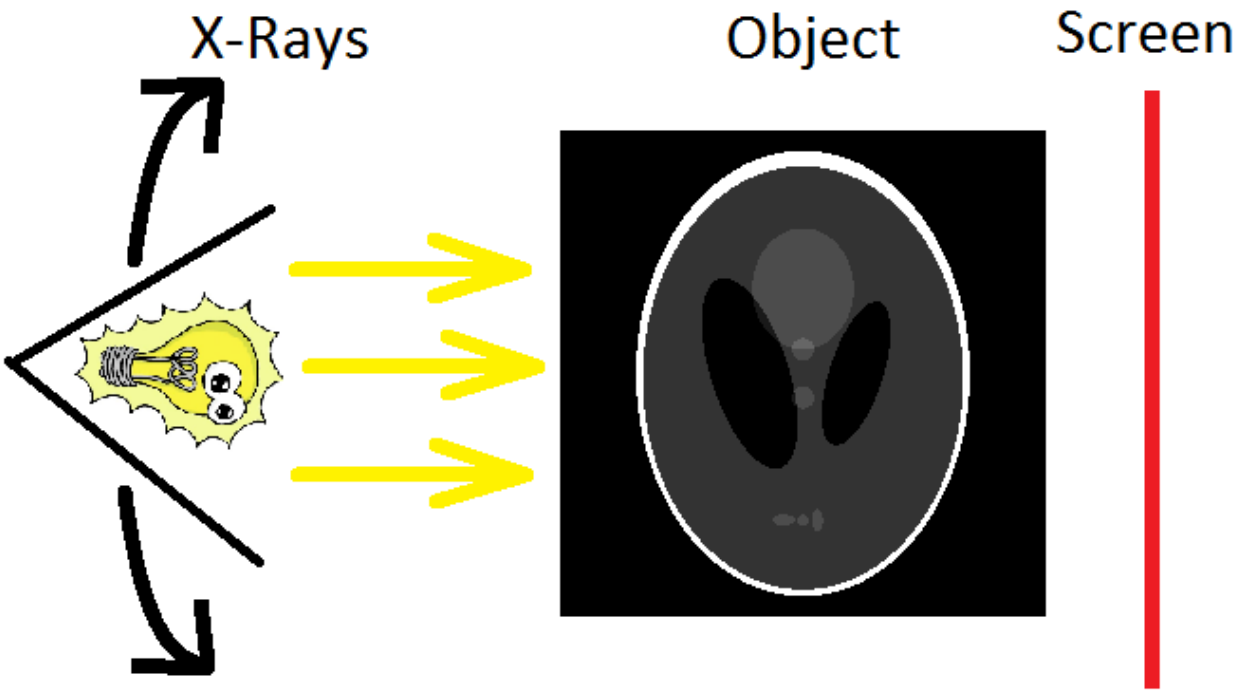
Michael Möller – michael.moeller@uni-siegen.de



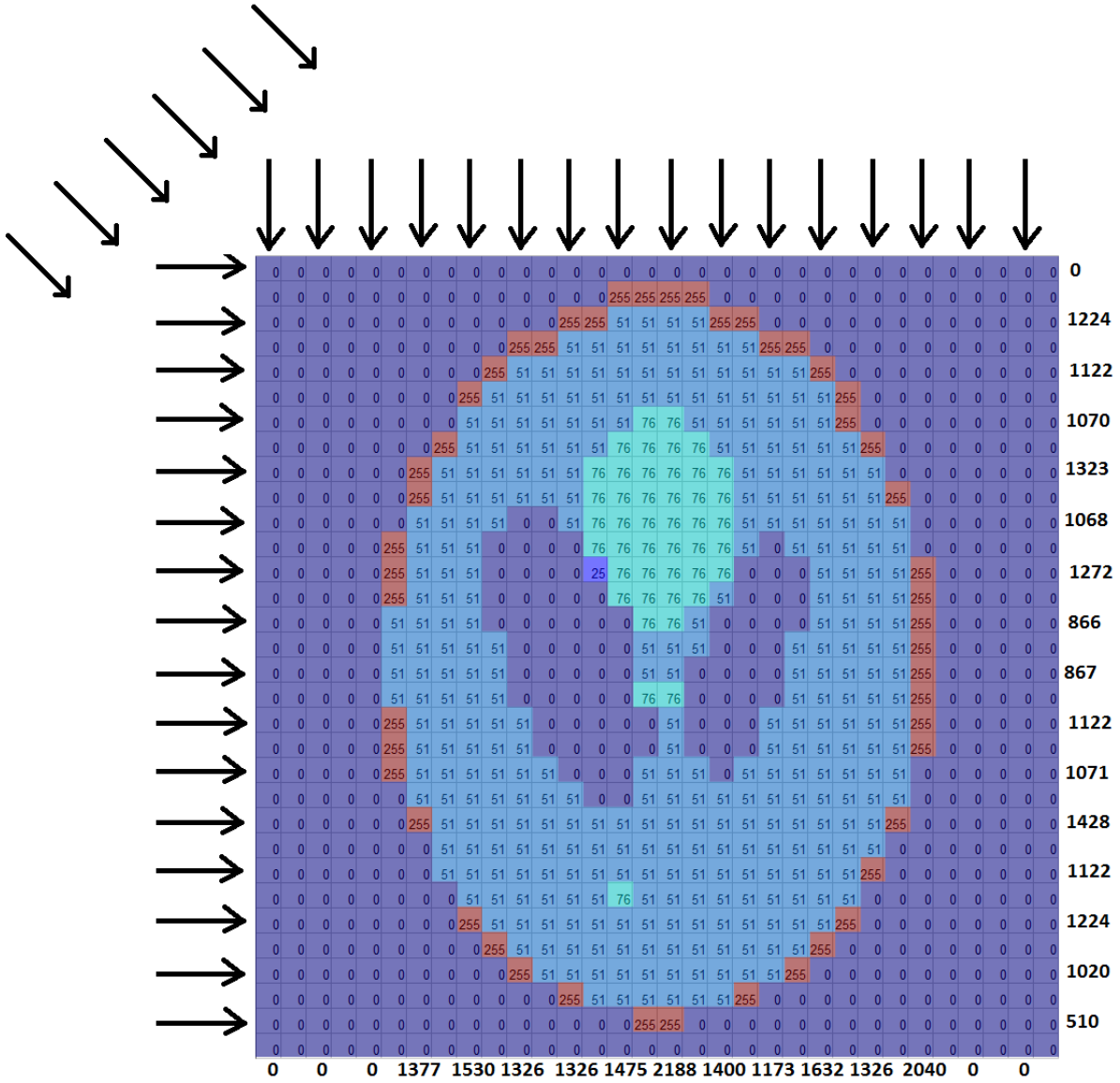
Inverse problems

In many applications the desired quantity cannot be observed directly, but we have a thorough understanding of what exactly the relation is.

Example: CT reconstruction



Inverse problems



Desired quantity

Data ***Noise***

$$f = A(u) + n$$

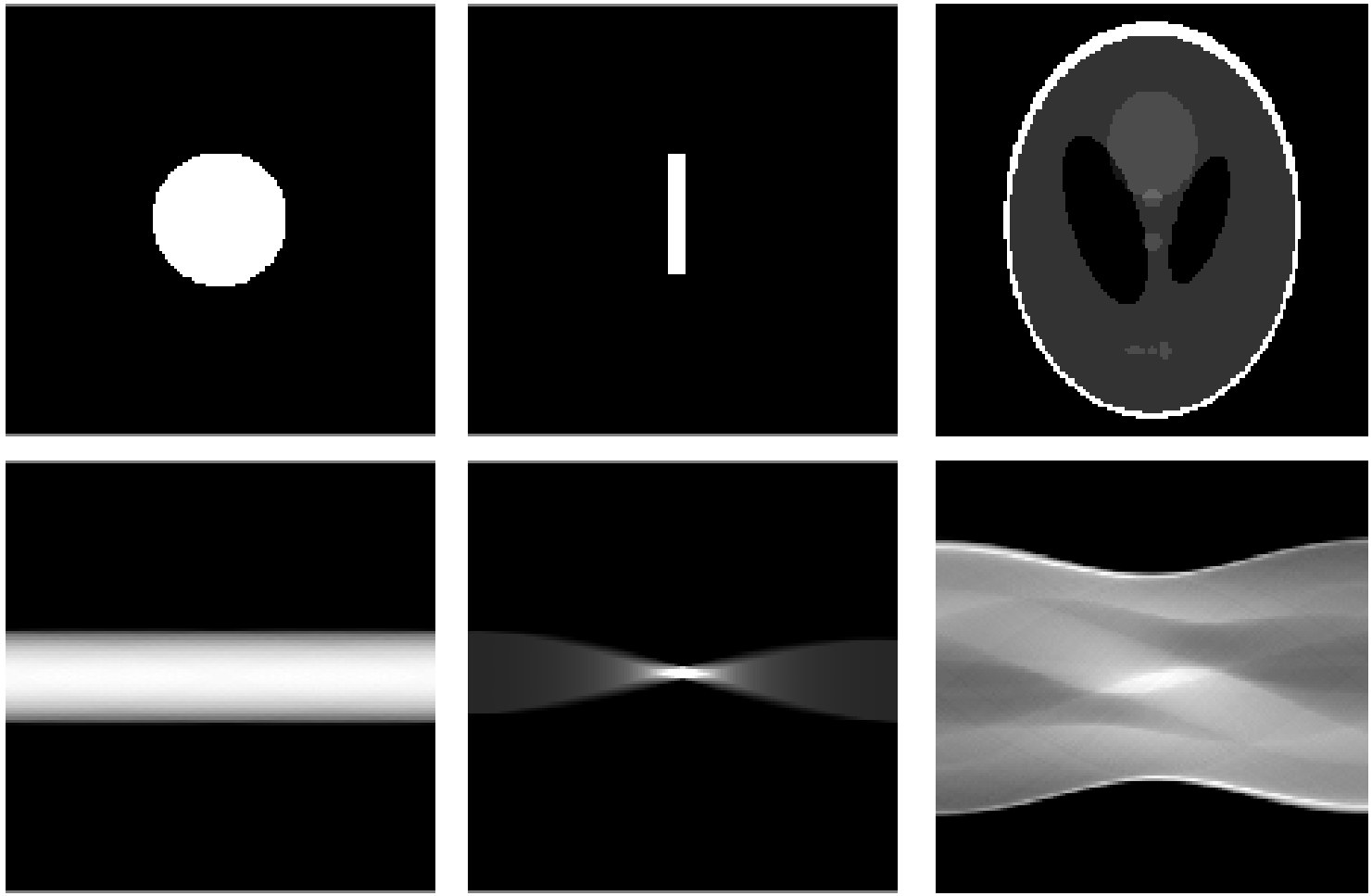
Linear operator that takes line integrals

Michael Möller – michael.moeller@uni-siegen.de

Inverse problems

Image u

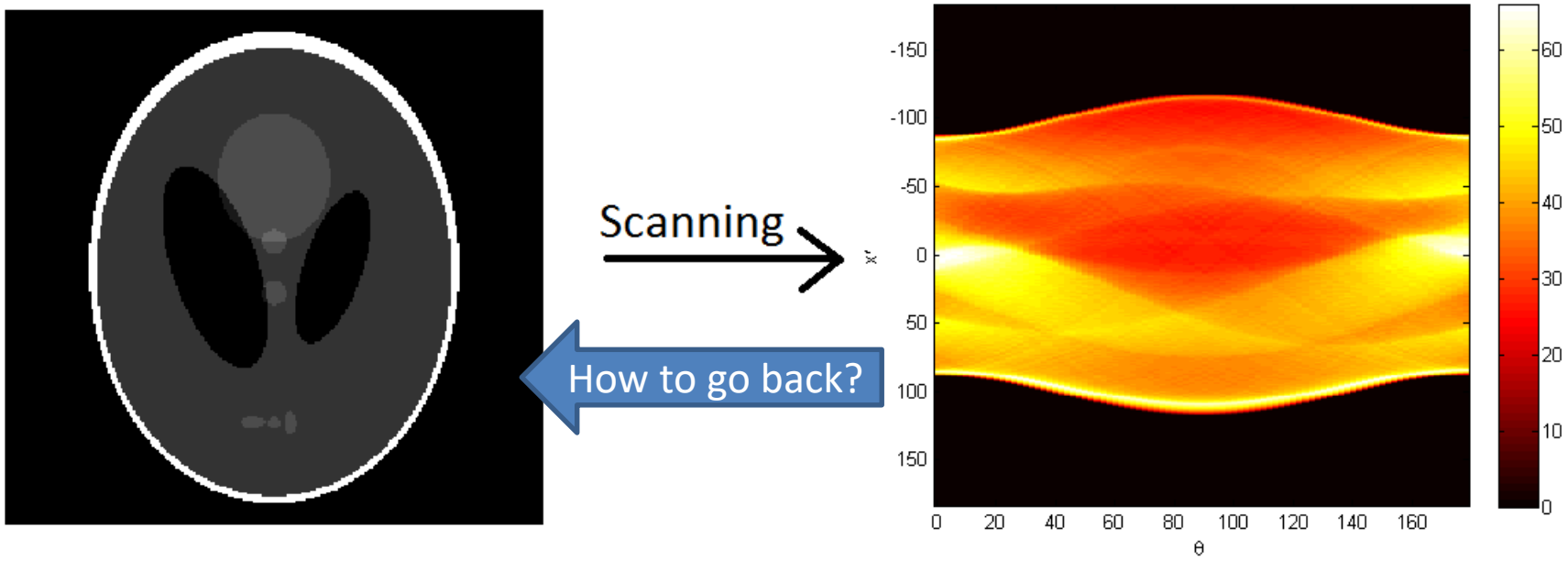
Sinogram f



Inverse problems

In many applications the desired quantity cannot be observed directly, but we have a thorough understanding of what exactly the relation is.

Example: CT reconstruction

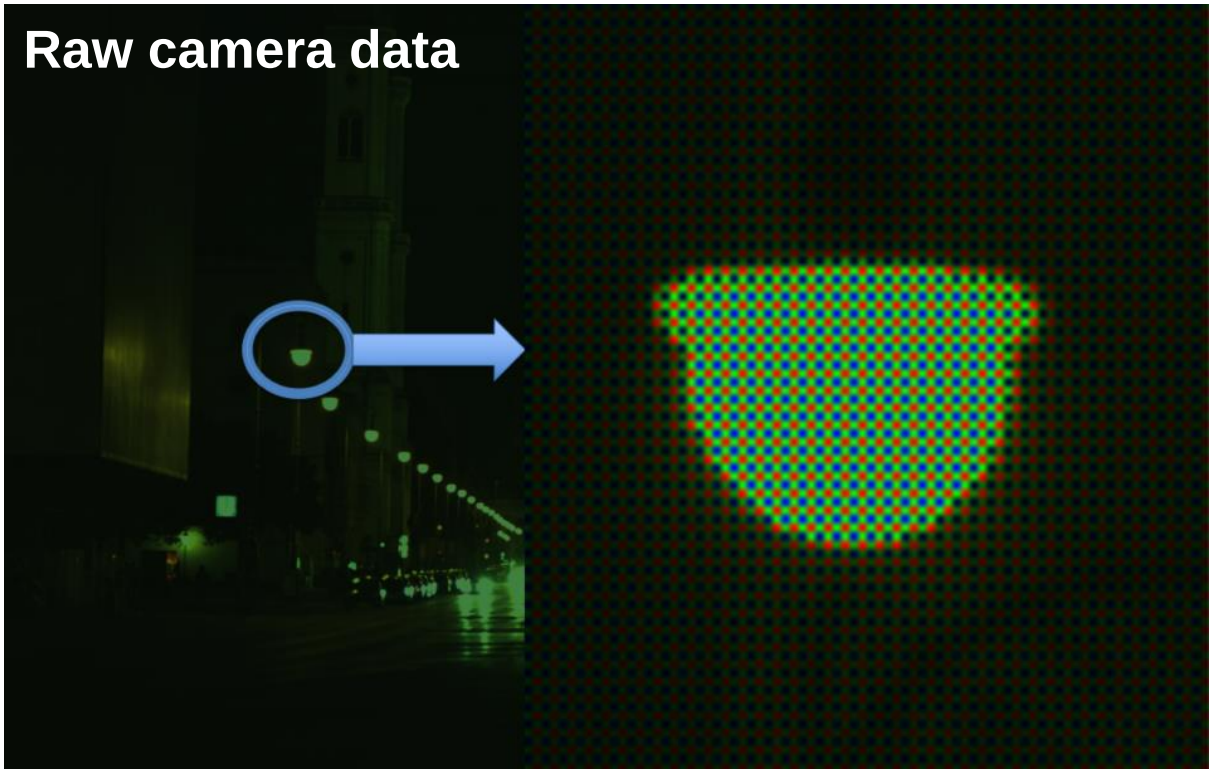


Many other (similar) problems arise e.g. in MRI, PET, EIT, Ultrasound

Inverse problems

In many applications the desired quantity cannot be observed directly, but we have a thorough understanding of what exactly the relation is.

Example: Demosaicking



Desired quantity

Data ***Noise***

$$f = A(u) + n$$

Linear operator that switches off two out of three color values at each pixel

Inverse problems

In many applications the desired quantity cannot be observed directly, but we have a thorough understanding of what exactly the relation is.

Example: Decompression

Desired quantity

Data ***Noise***

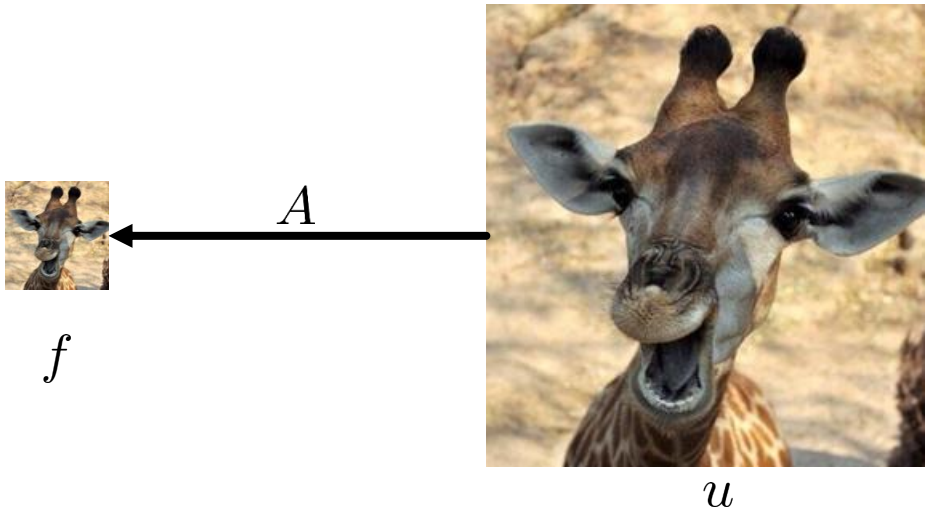
$$f = A(u) + n$$

Operator that computes a patchwise DCT and quantizes the resulting values

Inverse problems

In many applications the desired quantity cannot be observed directly, but we have a thorough understanding of what exactly the relation is.

Example: Single Image Superresolution



Desired quantity

Data ***Noise***

$$f = A(u) + n$$

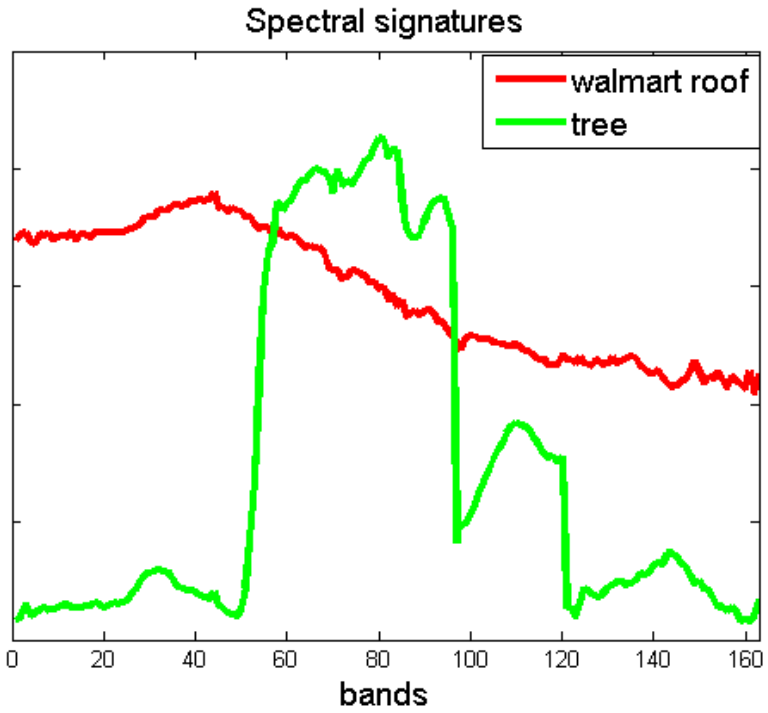
Linear operator that downscales a high resolution image

Inverse problems

In many applications the desired quantity cannot be observed directly, but we have a thorough understanding of what exactly the relation is.

Example: Hyperspectral Unmixing

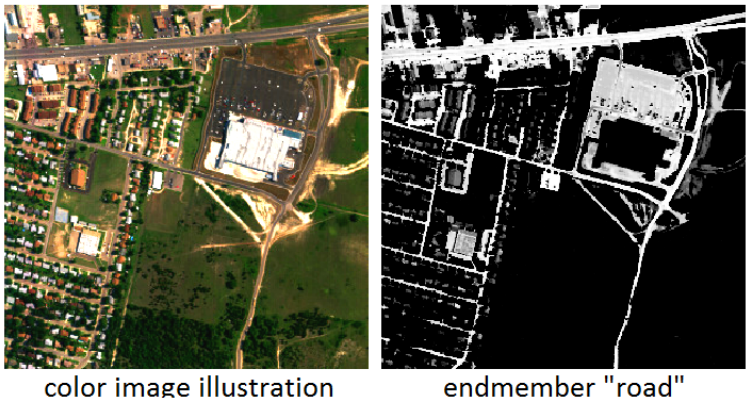
Hyperspectral cube with 163 bands



Inverse problems

In many applications the desired quantity cannot be observed directly, but we have a thorough understanding of what exactly the relation is.

Example: Hyperspectral Unmixing



color image illustration | endmember "road"

endmember "roof" | endmember "trees"

Desired quantity = abundance of material

Data at each pixel

Noise

$$f = A(u) + n$$

Linear operator that mixes the spectral signatures of raw materials

Energy minimization methods

What is a common way to solve such problems?

Solve *the inverse problem* as an optimization problem!

This is known under the name *of energy minimization methods*!

Least-squares problem:

$$\min_u \|A(u) - f\|_2^2$$

Least-squares problem with certain regularity:

$$\min_u \|A(u) - f\|_2^2 + R(u)$$

Energy minimization problem with arbitrary data fidelity term:

$$\min_u H(A(u), f) + R(u)$$

Why not just use least-squares estimates?

Inverse Operator

Blur Operator

Add tiny amount of noise

Inverse Operator

Extreme differences!!

Variational Methods

Energy minimization methods

candidate image

$$\hat{u} = \arg\min_u E(u)$$

Small if u is has desirable properties

Large, otherwise

Data fidelity term

Regularization

$$= \arg\min_u H(A(u), f) + R(u)$$

Known from an explicit model

(as illustrated in the examples)

Often difficult! Common energy minimization approaches for instance use $TV(u) = \|\nabla u\|_1$ the *total variation regularization*

Learning vs. Modelling

Energy minimization methods

Data fidelity term *Regularization*

$$\hat{u} = \arg\min_u \boxed{H_f(u)} + \boxed{R(u)}$$

Describe how the data is formed

Describe 'natural' or 'typical' images

Pro: Active control over how the data is formed and what type of noise to expect

Con: Good regularizers are difficult to design.

Machine learning methods

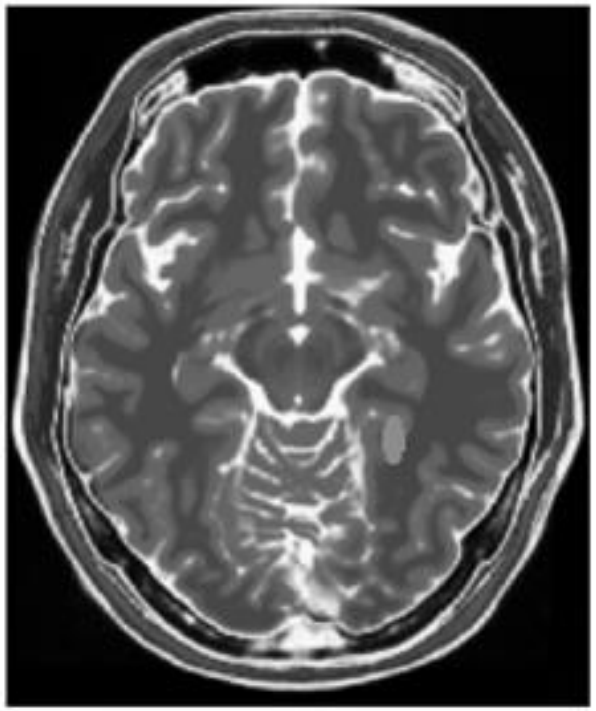
- Simulate 100,000 pairs of u and f with the data formation process and expected noise level
- Learn how to map f to u

Pro: Powerful!

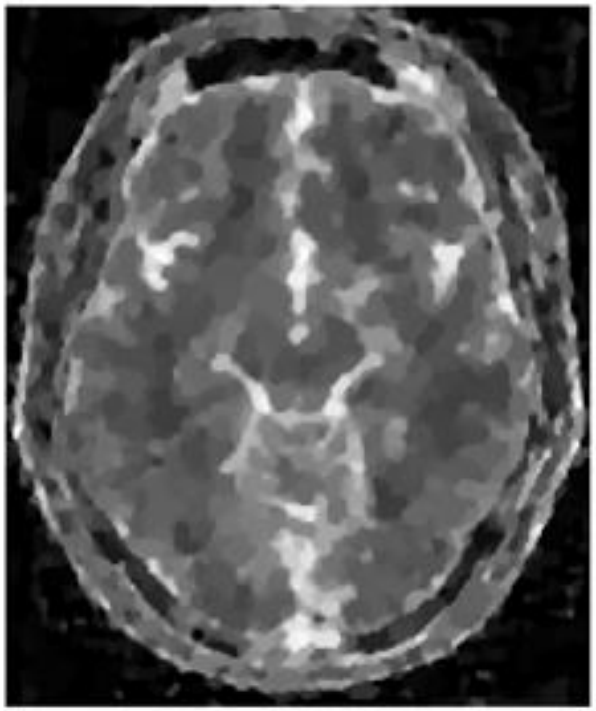
Con: Costly training. Does not explicitly use prior knowledge about data formation. Dangerous?

Understanding Deep Learning

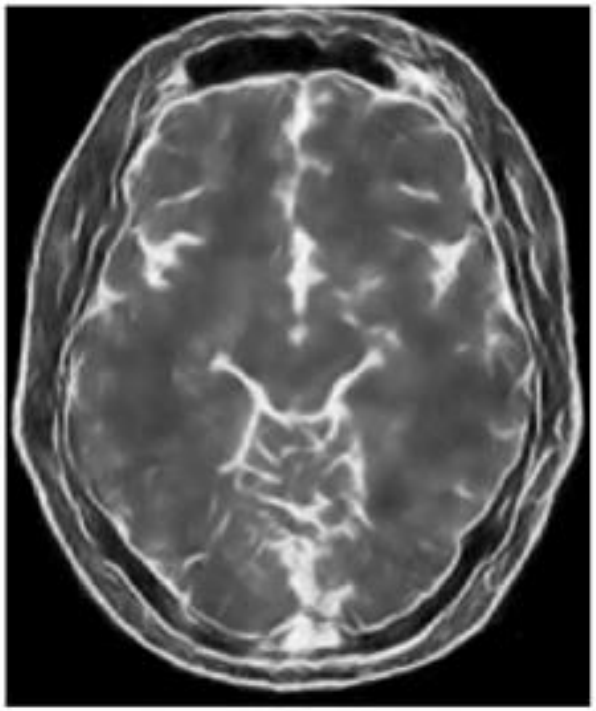
Modelling approaches, such as energy minimization methods are thoroughly understood! Deep learning often remains too much of an uncontrolled black-box.



Ground truth



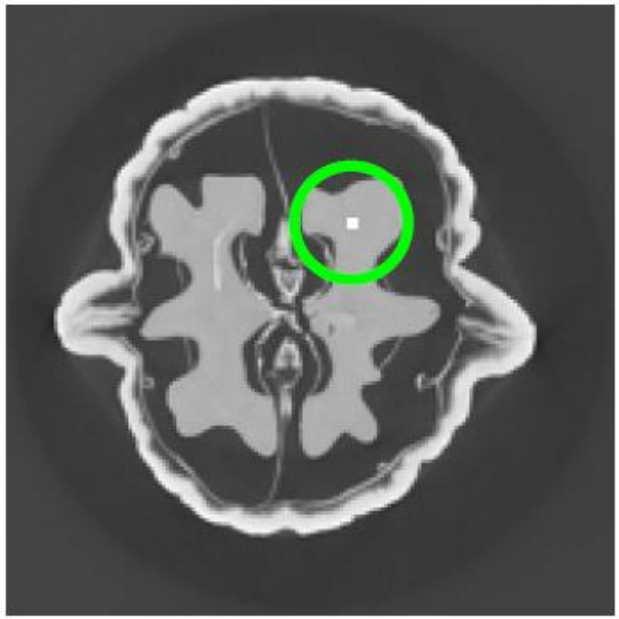
Total variation regularized



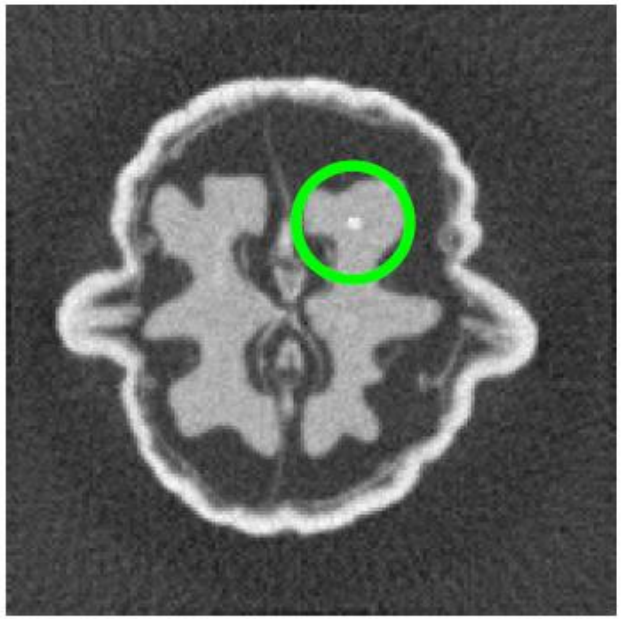
Deep learning approach

Understanding Deep Learning

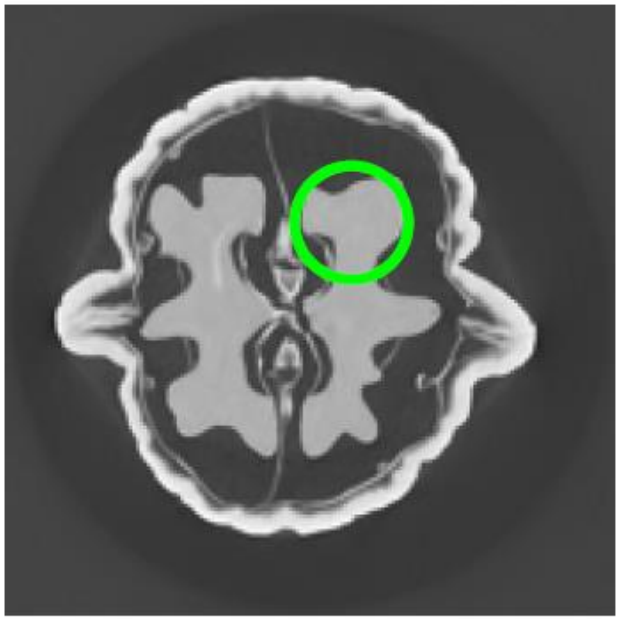
Modelling approaches, such as energy minimization methods are thoroughly understood! Deep learning often remains too much of an uncontrolled black-box.



ground truth



gradient descent, PSNR 27.5



learned, PSNR 40.2

Models + Deep Learning?

How can we combine model- and learning-based techniques?

Let us focus on one very simple example:

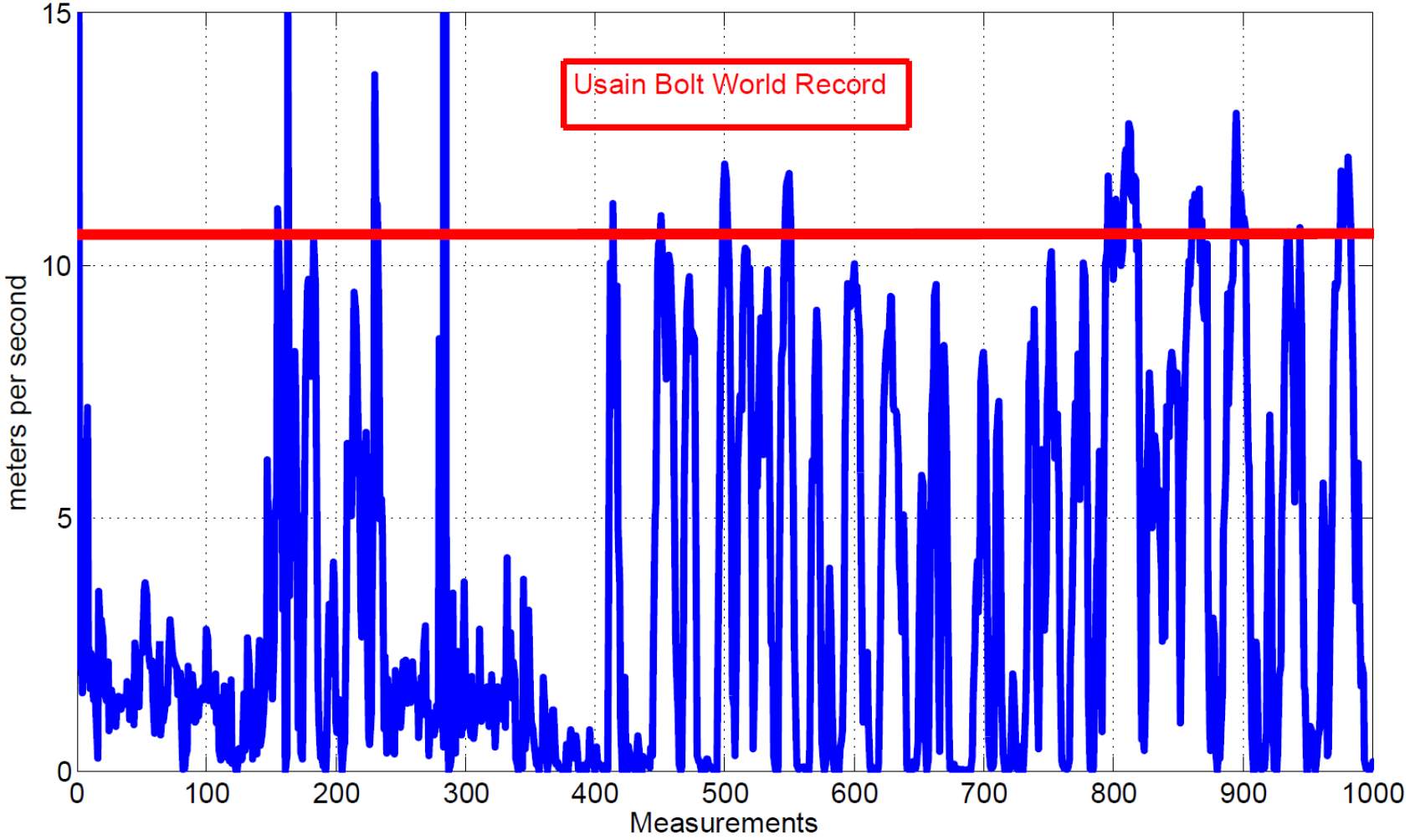
Data from: *Microsoft Research GeoLife GPS Trajectories*

How fast did this person go?

Time	'12:44:12'	'12:44:13'	'12:44:15'
Latitude	39.974408918	39.974397078	39.973982524
Longitude	116.30352210	116.30352693	116.30362184

Models + Deep Learning?

Speed is travelled distance in meters per time needed…



Models + Deep Learning?

Speed is travelled distance in meters per time needed…

Position at time t_i

Estimated speed at time t_i

$$u_i = \frac{f_i - f_{i-1}}{t_i - t_{i-1}} = \frac{1}{\Delta t}(f_i - f_{i-1})$$

If measurements are equidistant

Phrased as an inverse problem

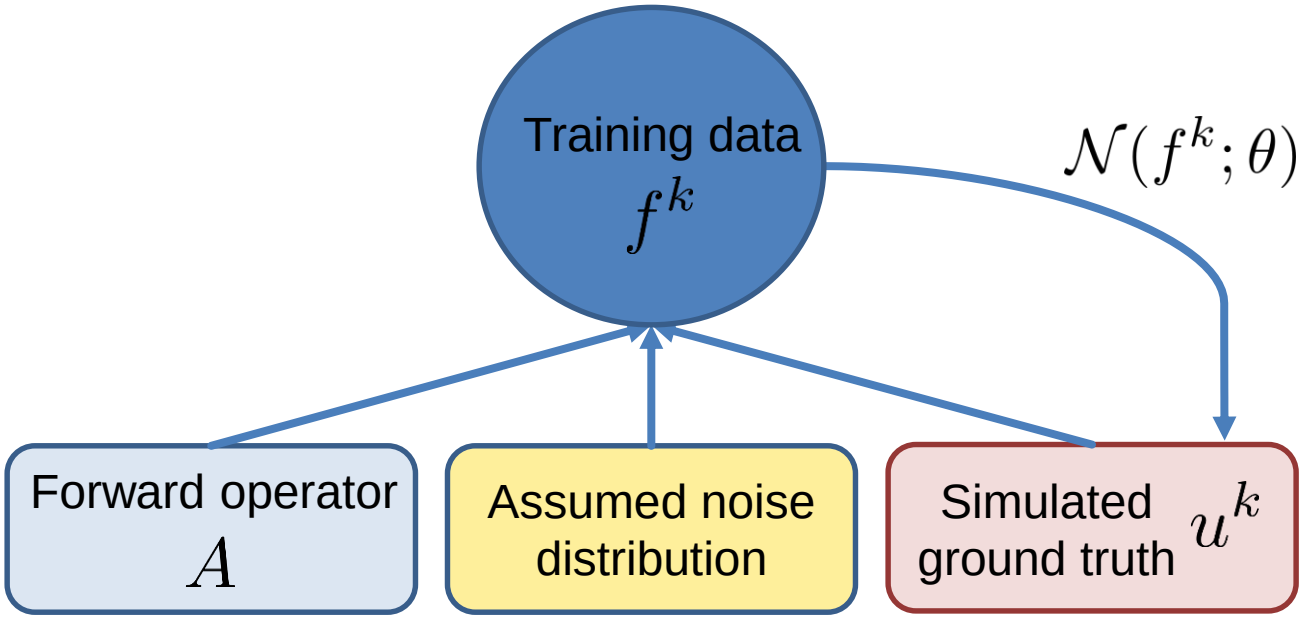
$$\begin{pmatrix} f_1 \\ . \\ . \\ . \\ f_n \end{pmatrix} = \underbrace{\Delta t \cdot \begin{pmatrix} 1 & 0 & . & . & . & 0 \\ 1 & 1 & 0 & . & . & 0 \\ 1 & . & 1 & . & . & 0 \\ 1 & . & . & . & 1 & 0 \\ 1 & 1 & . & . & 1 & 1 \end{pmatrix}}_{= A} \begin{pmatrix} u_1 \\ u_2 \\ . \\ . \\ . \\ u_n \end{pmatrix}$$

-> See Matlab

Models + Deep Learning?

How can we combine model- and learning-based techniques for finding a realistic u such that $Au \approx f$?

1. Pure supervised learning (using the forward model to train on simulated data)

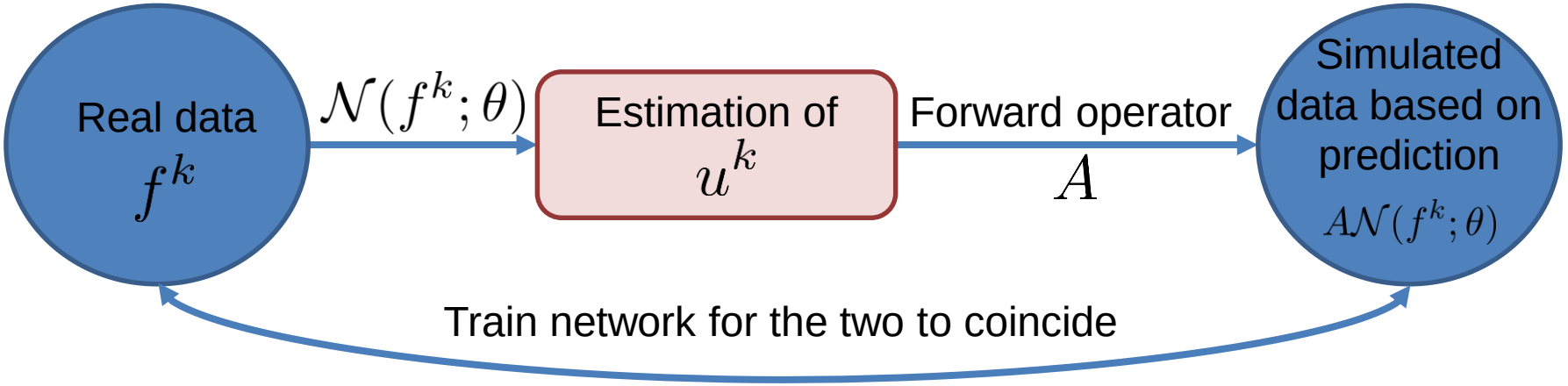


e.g. by training via $\min_\theta \sum_{\text{training examples } k} \|\mathcal{N}(f^k;\theta) - u^k\|^2$

Models + Deep Learning?

How can we combine model- and learning-based techniques for finding a realistic u such that $Au \approx f$?

2. Model-based autoencoder (see Volker Blanz's talk)



e.g. by training via $\min_\theta \sum_{\text{data examples } k} \|A\mathcal{N}(f^k;\theta) - f^k\|^2$

Models + Deep Learning?

How can we combine model- and learning-based techniques for finding a realistic u such that $Au \approx f$?

3. Letting the network architecture be based on energy minimization methods
(to be combined with any of the previous approaches)

Step 1: Get a reasonable energy minimization approach, e.g.

$$\hat{u} = \arg\min_u \ \frac{1}{2}\|Au - f\|_2^2 \ + \ R(u)$$

with a suitable regularizer.

See Matlab code for extremely simple choices

$$R(u) = \frac{\alpha}{2}\|u\|_2^2$$

$$R(u) = \frac{\alpha}{2}\|Du\|_2^2$$ for D taking finite differences

Models + Deep Learning?

$$\hat{u} = \arg\min_u \frac{1}{2}\|Au - f\|_2^2 + R(u)$$

For more complicated choices of R, the solution of the above minimization cannot be stated in closed form anymore! One needs an algorithm to determine the solution!

Simple possibility: Gradient descent

Iterate $u(k+1) = u(k) - \tau \left(A^T(Au(k) - f) + \nabla R(u(k)) \right)$

New iterate — $u(k+1)$

Old iterate — $u(k)$

Step size — τ

Gradient of $\frac{1}{2}\|Au - f\|_2^2$

Gradient of $R(u)$

Replace by some kind of parameterized (learnable) operation!

Models + Deep Learning?

3.1 Rolled-out algorithms

Interpret

$$u(k+1) = u(k) - \tau \left(A^T(Au(k) - f) + \mathcal{N}(u(k);\theta) \right)$$

for a fixed number of iterations as a (recurrent) network.

Example for three iterations

$$\tilde{\mathcal{N}}(f;\theta) = u(2) - \tau \left(A^T(Au(2) - f) + \mathcal{N}(u(2);\theta) \right)$$

where $u(2) = u(1) - \tau \left(A^T(Au(1) - f) + \mathcal{N}(u(1);\theta) \right)$

where $u(1) = \tau A^T f$

Then use $\tilde{\mathcal{N}}(f;\theta)$ as a network in your favorite training proceedure.

Models + Deep Learning?

3.1 Rolled-out algorithms

Interpret

$$u(k+1) = u(k) - \tau \left(A^T(Au(k) - f) + \mathcal{N}(u(k);\theta) \right)$$

for a fixed number of iterations as a (recurrent) network.

Specific choice for $\mathcal{N}(u;\theta)$

$$\mathcal{N}(u;\theta) = \nabla_u R(u;\theta)$$

For example

$$R(u;\theta) = \frac{1}{2}\sum_{i=1}^{n} \max((\theta u)_i, 0)^2 \quad \Rightarrow \quad \nabla_u R(u;f) = \theta^T \max(\theta u, 0)$$

In this case the algorithm can still be interpreted as an (aproximate) energy minimization method with learned regularizer.

Models + Deep Learning?

3.1 Rolled-out algorithms

More flexible version: Consider

$$u(k+1) = u(k) - \tau \left(A^T(Au(k) - f) + \mathcal{N}^k(u(k); \theta^k) \right)$$

for a fixed number of iterations as a network.

Now the architecture is very flexible, still motivated by an energy minimization method but there is no energy such a network minimizes, not even approximately.

Exemplary References

Learning a regularization

- Fields of experts model by Roth and Black (2005)
- Analysis operator learning by Chen, Ranftl, and Pock (2014)

Learning an algorithm

- Shrinkage Fields by Schmidt and Roth (2014)
- From a PDE perspective: On learning optimized reaction diffusion processes for effective image restoration by Chen, Yu, and Pock (2015)

Models + Deep Learning?

3.2 Task-independent algorithmic schemes

Let us return to

$$u(k+1) = u(k) - \tau \left(A^T(Au(k) - f) + \nabla R(u(k)) \right)$$

and rewrite as

$$u(k+1) = \frac{1}{2} z_1 + \frac{1}{2} z_2,$$

with $z_1 = u(k) - 2\tau A^T(Au(k) - f),$

Takes the current iterate towards more fidelity

and ~~$z_2 = u(k) - 2\tau \nabla R(u(k))$~~

Takes the current iterate towards more regularity

$z_2 = \mathcal{N}(u(k); \theta)$ Network trained on denoising

Advantage: One can change the data fidelity term while keeping the same network!

Replacing the regularizer

Image with Salt-and-Pepper noise

Replacing the regularizer

Denoising Network trained on Gaussian Noise

Replacing the regularizer

Exemplary References

Decoupling learned regularity from the data formation process

- Learning proximal operators: Using denoising networks for regularizing inverse imaging problems by Meinhardt, Moeller, Hazirbas, Cremers (2017).
- One network to solve them all – solving linear inverse problems using deep projection models by Chang et al. (2017).
- Learning deep CNN denoiser prior for image restoration by Zhang, Zuo, Gu, Zang (2017).

Models + Deep Learning?

4. Predicting Descent Directions

The previous methods have one drawback: They cannot guarantee the forward model to be respected!

Structurally different approach: Predict a direction that allows to provably reduce the data fidelity costs!

Make network predictions $d^k := \mathcal{N}(u^k, \nabla E(u^k), f; \theta)$

and make sure that it provably holds that

$$\langle d^k, -\nabla E(u^k) \rangle \geq \zeta \|\nabla E(u^k)\|$$

Then the next iterate $u^{k+1} = u^k + \tau_k d^k$ provably satisfies

$$E(u^{k+1}) \leq E(u^k)$$

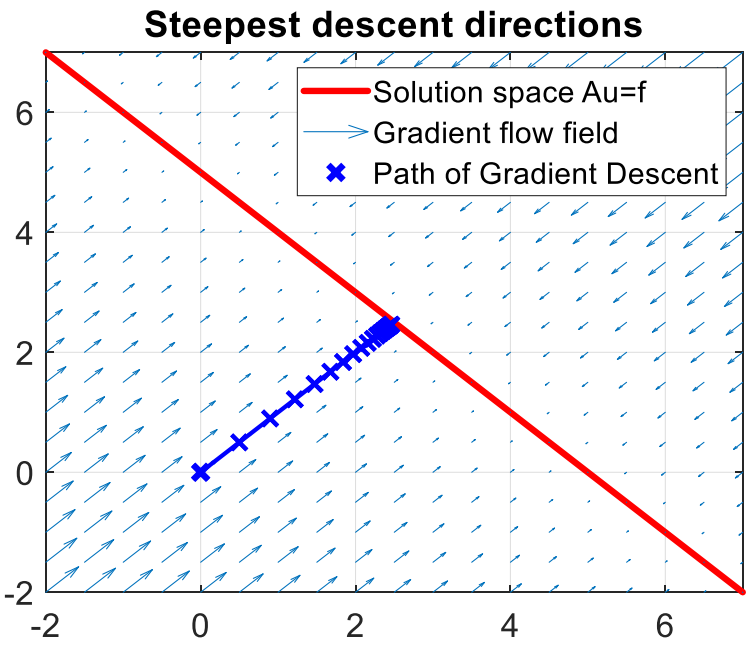
Energy dissipating networks

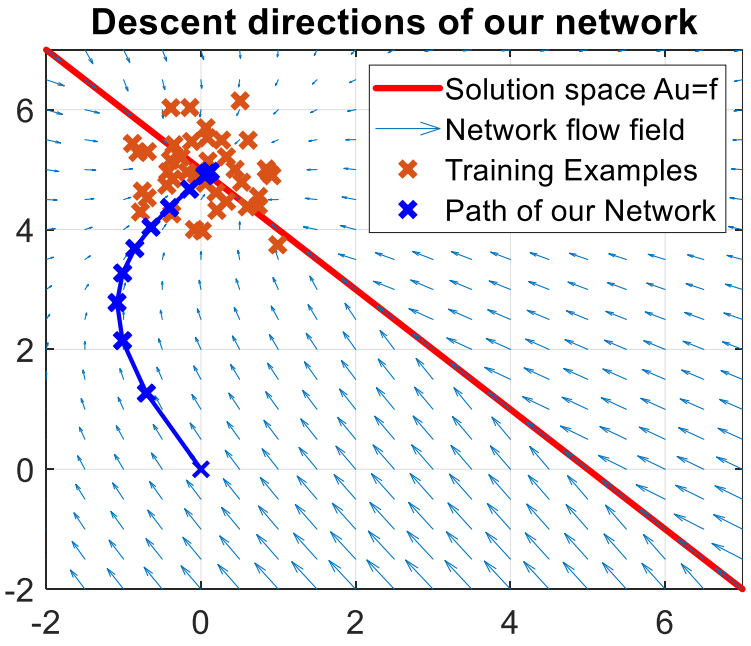
Proposition: Under mild regularity assumptions and with an appropriate step size rule the iteration

$$u^{k+1} = u^k + \tau_k d^k\,, \qquad d^k := \mathcal{N}(u^k, \nabla E(u^k), f; \theta)$$

Meets $\lim_{k\to\infty} \|\nabla E(u^k)\| = 0$. If the minimizer u^* of E is unique, then $\lim_{k\to\infty} u^k = u^*$.

Why not just minimize E then? Reason 1: Data-driven selection of minimizers!

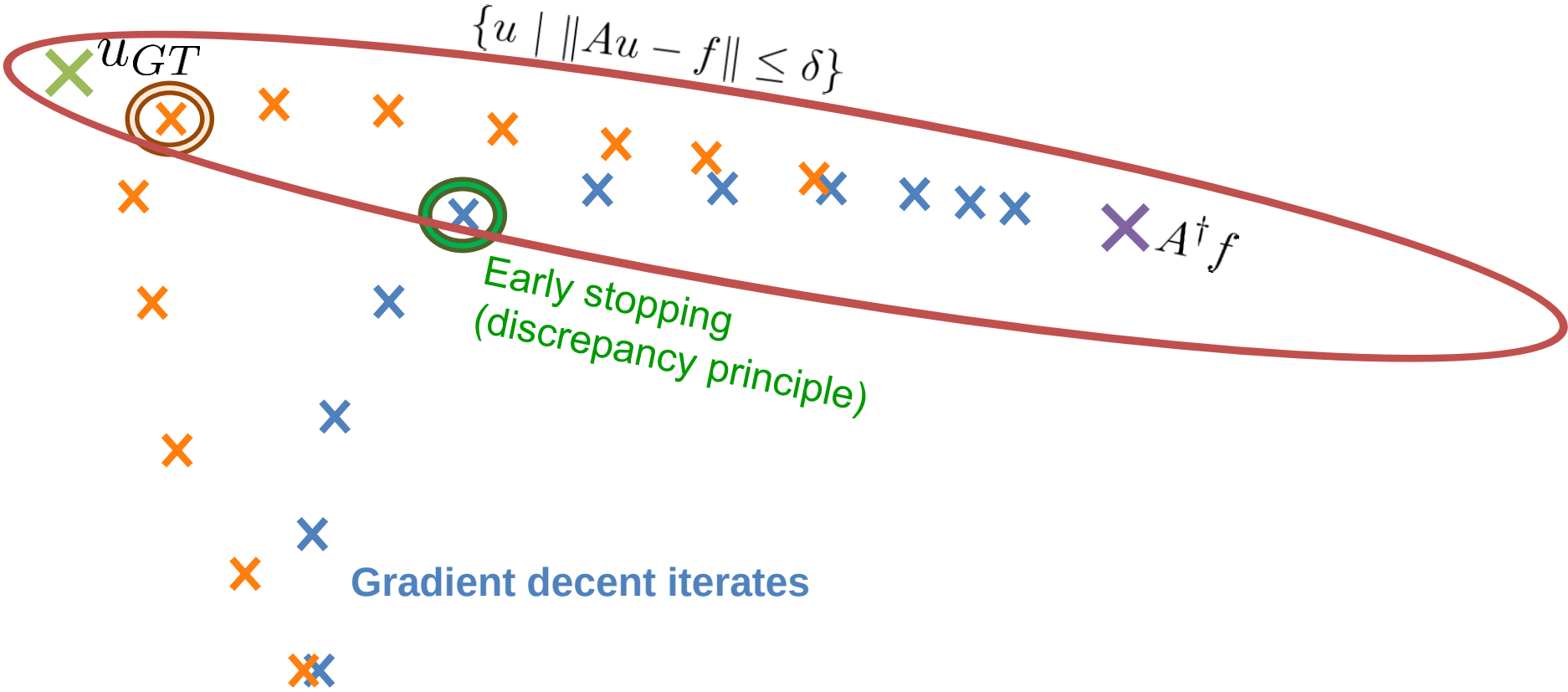




Energy dissipating networks

Why not just minimize E then? Reason 2: Taking a better path!

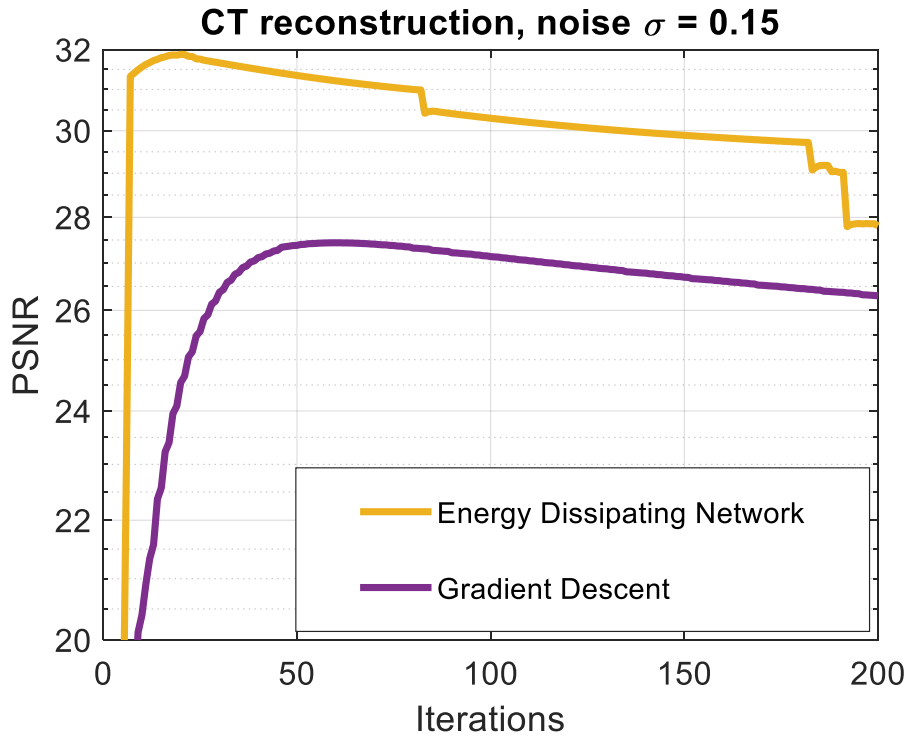
Let $E(u) = \frac{1}{2}\|Au - f\|^2$

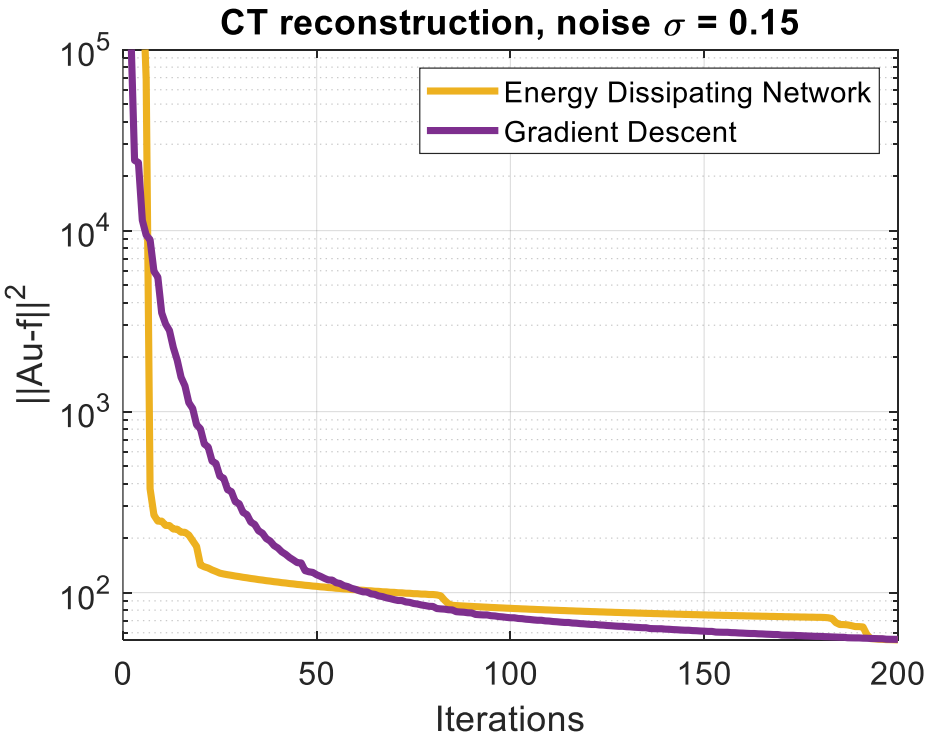


Energy dissipating networks

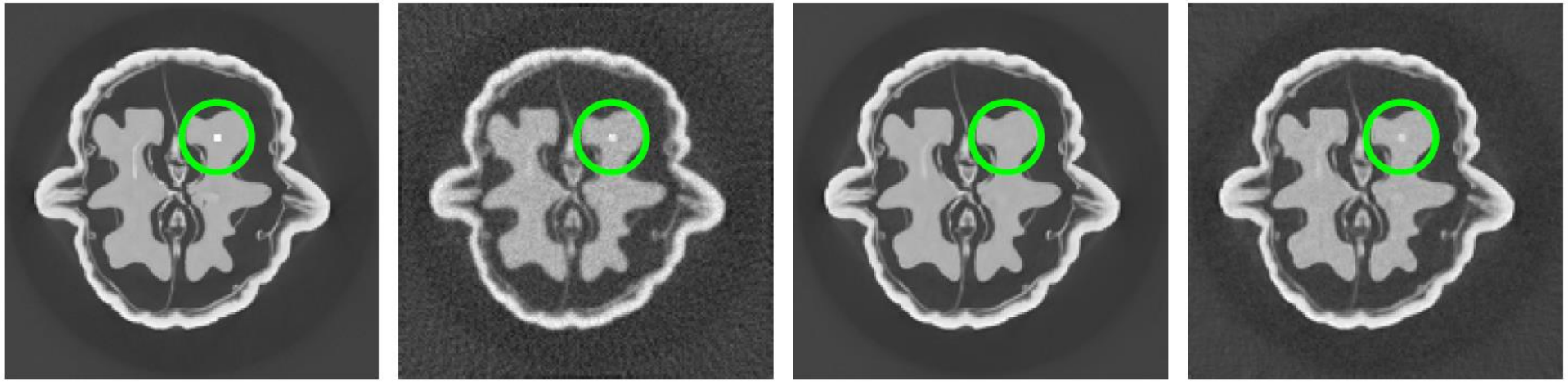
Exemplary results for CT reconstruction

$$E(u) = \frac{1}{2}\|Au - f\|^2$$





Energy dissipating networks



ground truth | gradient descent, PSNR 27.5 | learned, PSNR 40.2 | energy dissipating, PSNR 33.0

The idea of energy dissipating networks is quite generic and provides a tool to marry constraints with learning based approaches.

Reference

Michael Moeller, Thomas Möllenhoff, Daniel Cremers. *Controlling Neural Networks via Energy Dissipation,* 2019, preprint at https://arxiv.org/abs/1904.03081

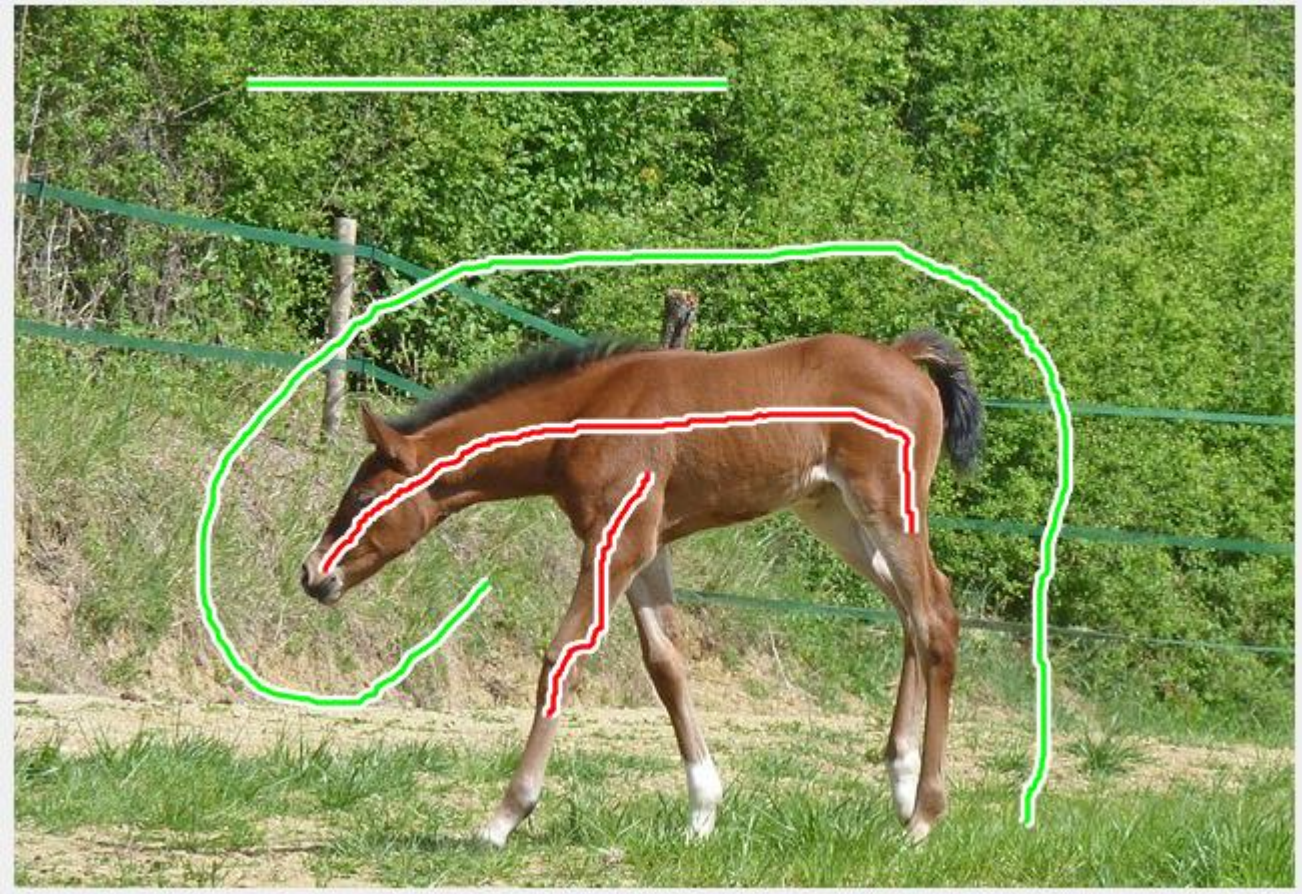
Intermediate conclusions

- There is a great benefit in the combination of model and learning-based approaches, particularly to control the latter with the former
- Replacing an optimization substep on the regularizer with a neural network is an interesting versatile approach with many properties yet to be understood
- Constraining networks by forcing them to yield descent directions w.r.t. suitable energies is a promising approach for many different applications.

Up next: Some further ideas on mimicking variational approaches even for non-model-based problems

Segmentation

Example where the data fidelity term is not clearly known



Segmentation

Based on the scribbles some likelyhood for foreground an background needs to be created.

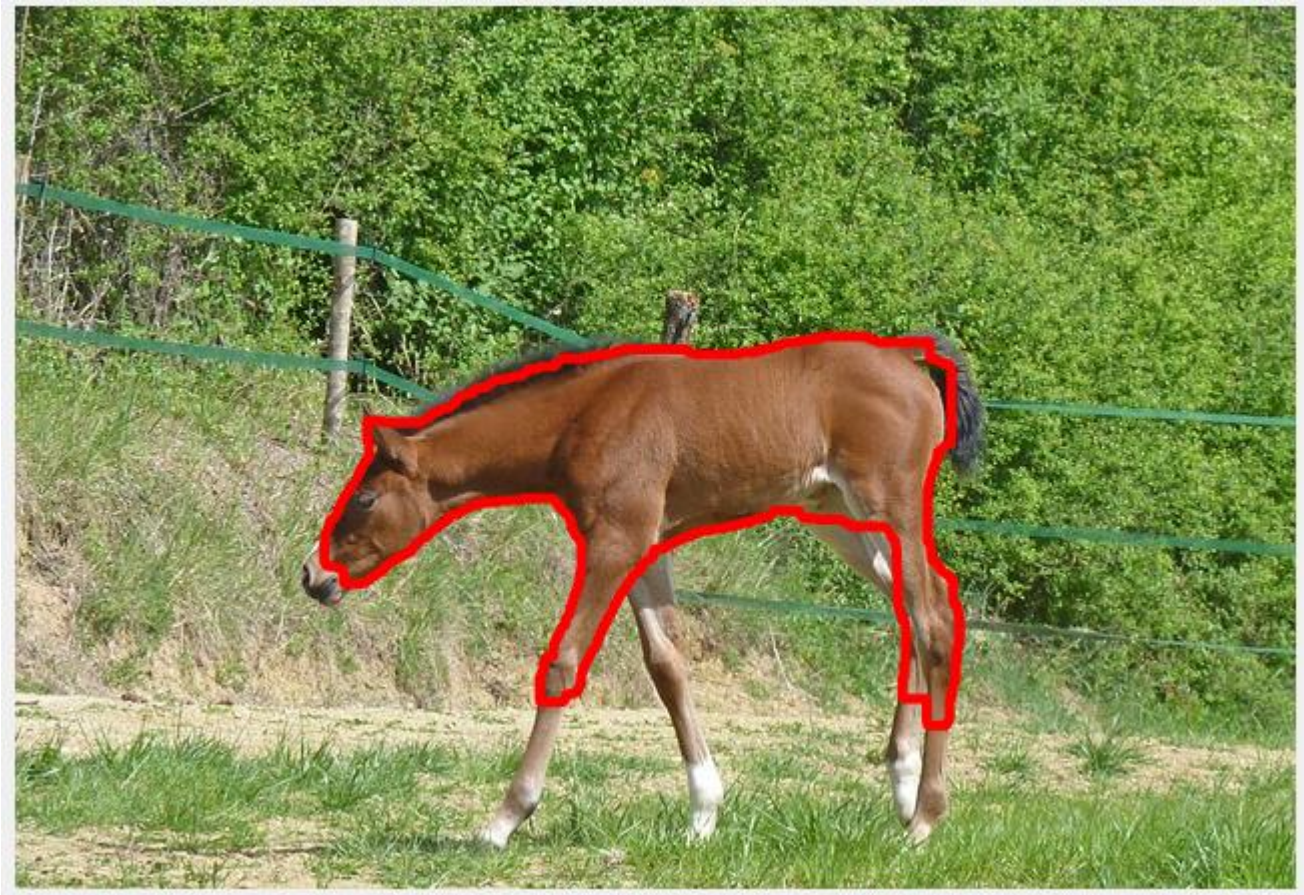
Segmentation

But such an estimate is often not sufficient

Segmentation

There is no clear model of the scribbles to the segmentation, but depending on the application, you can make a lot of reasonable assumptions!

1) Penalize the boundary length of the segmentation:



Segmentation

There is no clear model of the scribbles to the segmentation, but depending on the application, you can make a lot of reasonable assumptions!

2) Penalize moments such as area, centroid, variance, etc.

User input

Segmentation based on appearance only

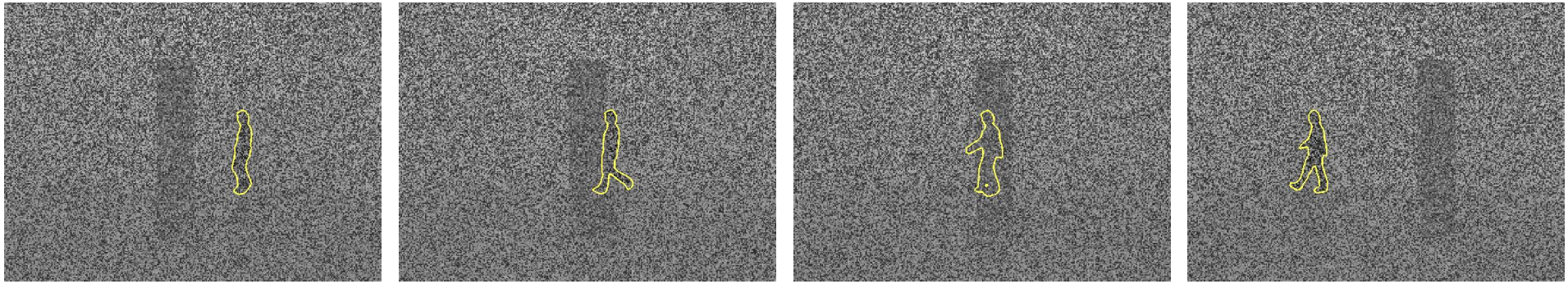
Segmentation with moment constraints

Images from: M. Klodt, F. Steinbrücker, and D. Cremers. Moment constraints in convex optimization for segmentation and tracking. In *Advanced Topics in Computer Vision*. 2013.

Segmentation

There is no clear model of the scribbles to the segmentation, but depending on the application, you can make a lot of reasonable assumptions!

3) Complex shape priors



Images from: D. Cremers. Dynamical statistical shape priors for level set-based tracking. *IEEE transactions on pattern analysis and machine intelligence*, 28(8):1262–1273, 2006.

This gives rise to a whole new class of prior information: Temporal consistency!

Segmentation

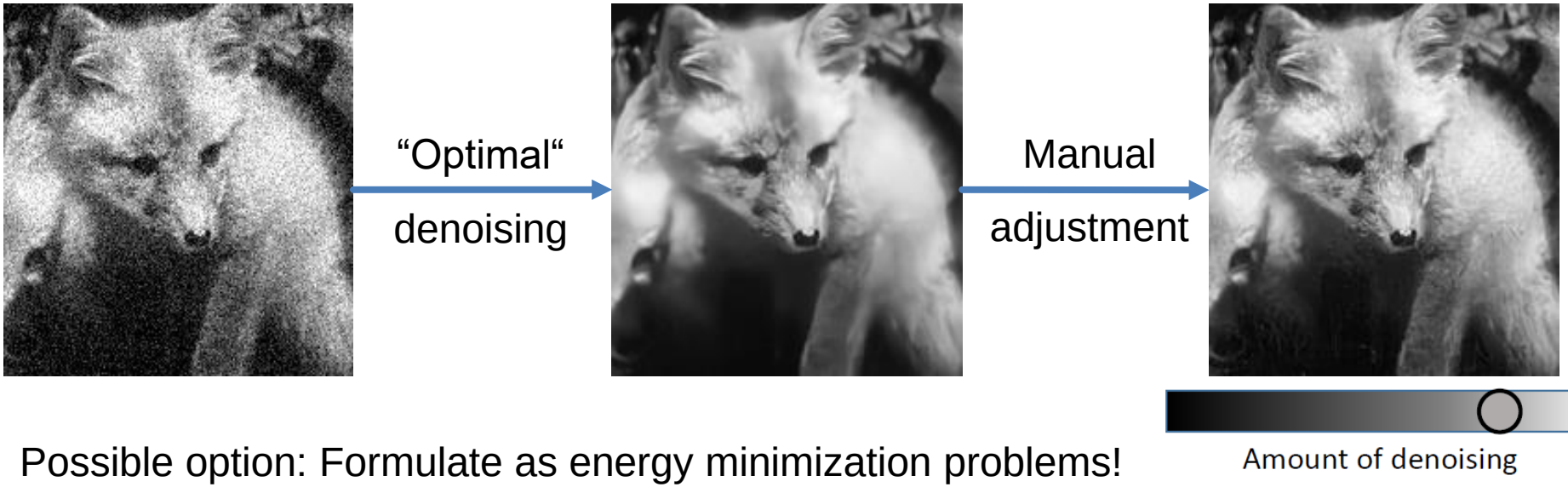
Courtesy of
Komal Vendidandi

Prior information in ML?

All examples motivate one question:

How do we incorporate prior knowledge in machine learning approaches?

For example, rough guesses of
a) the centroid of a segmentation
b) the area of a segmentation
c) deformable shape models of the object we're looking for
d) the amount of noise to be removed from a noisy image

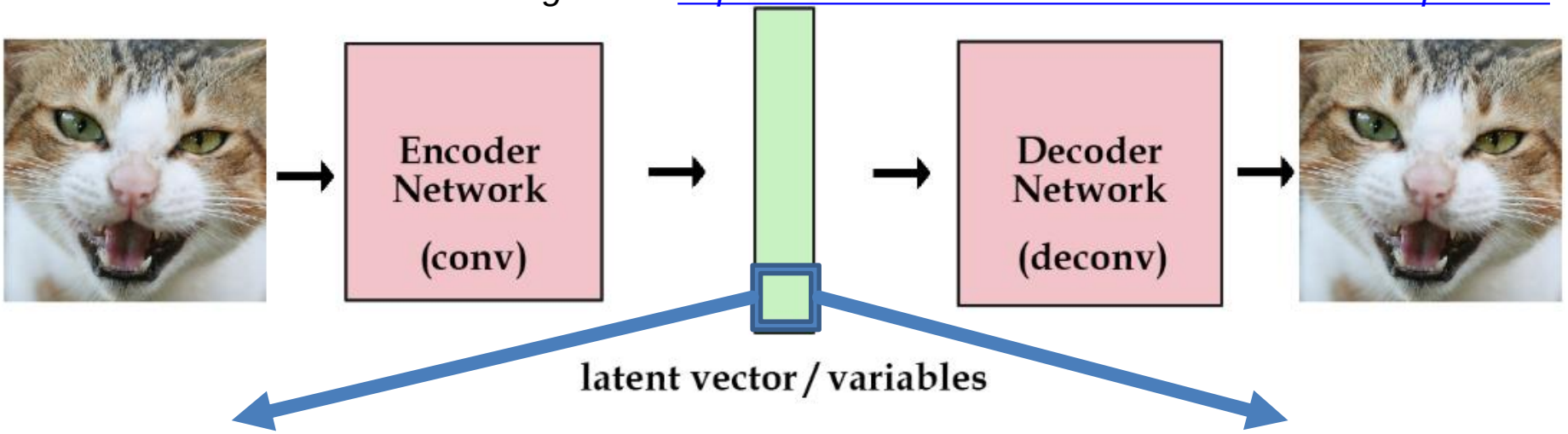


Possible option: Formulate as energy minimization problems!

Prior information in ML?

Possible fun try for a miniproject: Train an autoencoder or a variational autoencoder

Image from http://kvfrans.com/variational-autoencoders-explained/



Train/hardcode a desired piece of information on the image here, e.g. mean values for R, G, and B.

Explicitly encourage the decoder network to use this information somehow.

During inference, encode an image, manipulate the information and decode again – does it work?

Summary

For a given forward model, we discussed

1. Pure supervised learning (using the forward model to train on simulated data)

2. Model-based autoencoder

3.1 Rolled-out algorithms

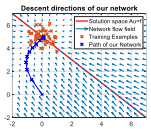
$$u(k+1) = u(k) - \tau \left(A^T(Au(k) - f) + \mathcal{N}^k(u(k); \theta^k) \right)$$

3.2 Task-independent algorithmic schemes

$$u(k+1) = \frac{1}{2}z_1 + \frac{1}{2}z_2, \qquad z_1 = u(k) - 2\tau A^T(Au(k) - f)$$

$$z_2 = \mathcal{N}(u(k); \theta) \quad \text{Denoising Network}$$

4. Predicting Descent Directions



Finally, we found that incorporating prior knowledge into neural networks is important far beyond a known forward model in inverse problems.

The quest for the best way to incorporate such knowledge continues!