Department of Computing

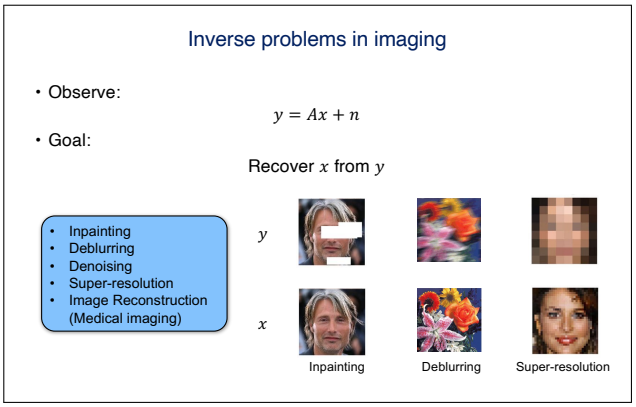
Imperial College of Science, Technology and Medicine

Image Registration

Author: Anton Zhitomirsky

Contents

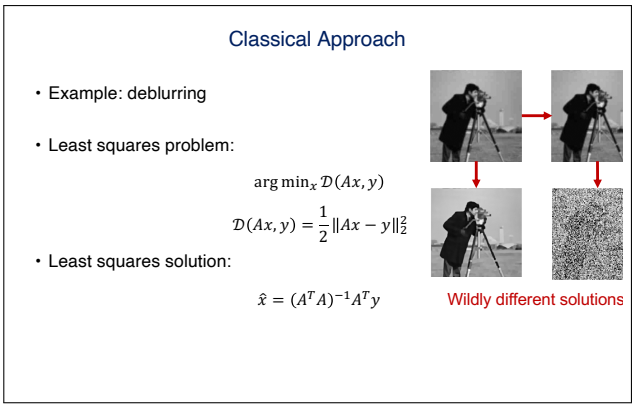
1 Inverse Problems in Imaging



The matrix A operates on some input x which generates an output with noise n. We want to try to reconstruct the original input out of the output.
For example, **Inpainting**, matrix A would act as a masking operator.

1.1 Classical Approach

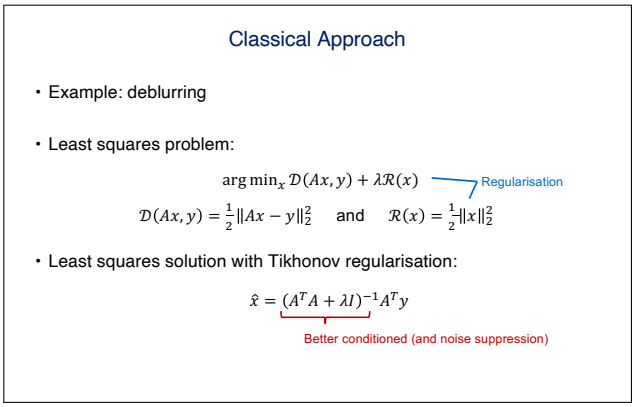
1.1.1 Problem



- In de-blurring, classically we can formulate this as a least squares problem, equation on the left. A descibes the forward map (here a linear operation). In other words, after applying a blurring operation to x we get y.
- The solution to this problem includes matrix manipulation.
- You can solve the problem without knowing what A is, but here we do.

The problem here, is that if the dat is slightly different, then you apply the same deep blurring approach, by solving the least squares solution at the bottom, you can potentially get wildly different solutions with a slightly different initial condition. This is called a **ill-posed problem** (where a slight pertubation in the data can skew results greatly).

1.1.2 Solution



Even with classical methods you can solve this by adding a regularisation term.

- In addition to minimising the distance between the observed data and the forward operation on the original data, you add a second term
- The second term in this example is a regularisation term, which is added to the solution, which modifies the behaviour of the optimisation problem.

- This gives a different solution (we effectively add an identity matrix scaled by lambda before we invert $A^\top A$). The λ defines the strength of the regularisation; if you increase λ you get smoother

solutions because you pay less and less attention to the actual distance measure which you have in the first term.

- This inversion means that $A^\top A$ becomes better conditioned, which means it's numerically more stable and you suppress noise.

1.2 General Approach

General approach for images

$$\arg\min_x \mathcal{D}(Ax, y) + \lambda\mathcal{R}(x)$$

$\mathcal{D}(Ax, y)$	Data consistency term
$\mathcal{R}(x)$	Regularisation term (encoding prior knowledge on x)
λ	Regularisation parameter

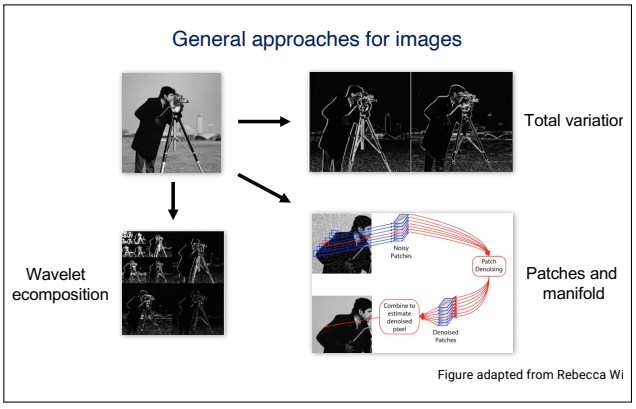
Common regularisers:

$\mathcal{R}(x) = \|\nabla x\|_2^2$	Tikhonov regularisation
$\mathcal{R}(x) = \|\nabla x\|_{2,1} = \sum_{i=0}^{n} \sqrt{\sum_d \left(\|\nabla x\|_i^{(d)}\right)^2}$	Total variation regularisation
$\mathcal{R}(x) = \|\Psi x\|_1$	Wavelet regularisation

- This regluarisation approach encodes the prior knowledge we have on our desired solution x.
- If we can encode this prior knowledge then we have a good regularisation strategy.
- The common regularisers shown to the side don't operate on image intensities, but rather on the image gradient.
- For example, the Tikhonov regularisation uses L-2 norm to regularise the gradients of the image intensities

- The second term, minimises the L-1 norm of the gradient of image intensities.
- Another common solution is to not regularise the image intensities of the gradient of image gradients, but first apply a sparsifying transformation, e.g. wavelet and then minimise that.

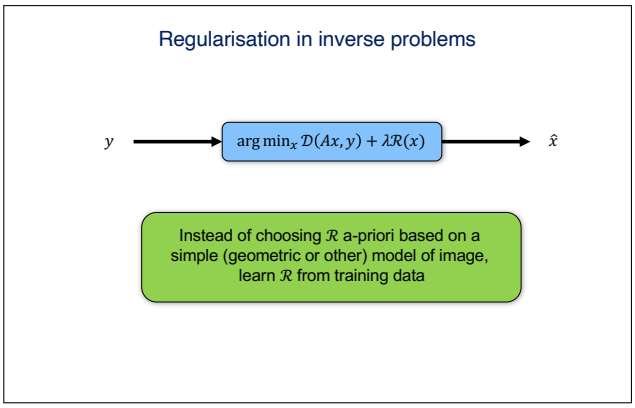
1.2.1 Why normalize gradients instead of image intensities?



- The problem with looking only at the gradients, is that the results are very sparse, which is a common thing in every-day images.
- If you apply L-1 norm on the wavelength decomposition, then you're effectively minimising the number of non-zero wavelength coefficients. This technique uses this wave decomposition in order to maximise sparsity or minimise the L-1.

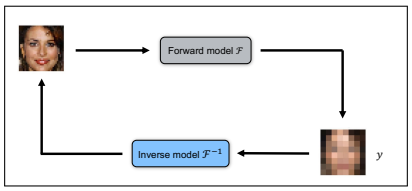
- The third approach is **Dictionary Learning** where you extract small patches from the iamge and build up a patch library. Then you encode each patch by a sparse combination of (e.g.) wavelet coefficients, denoising and combine these to reconstruct the original image.

1.3 Regularisation in inverse problems

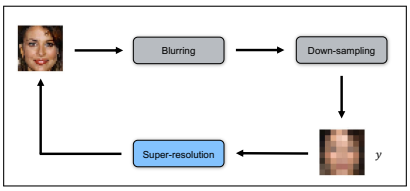


- we have an optimisation function made of two terms, the data consistency term and the regularisation.
- Can we learn a reguliser from our data rather than using these very ad hoc measures.

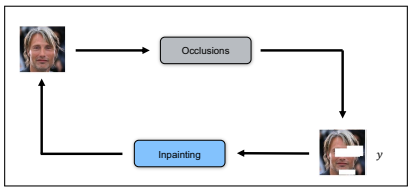
1.4 Forward model/inverse model



(a) image super resolution, the forward model is easy to implement. The inverse is harder



(b) image super resolution, the forward model can be implemented here, wheras the reverse is a black box called the super resolution algorithm.



(c) In-painting, the forward model is sipmle; just mask out image intensities. The inpainting tries to recover these masked values

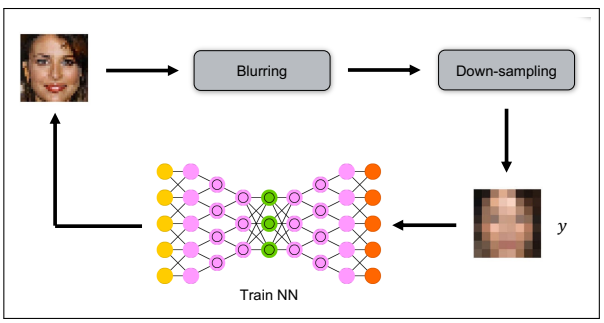
2 Solving Inverse Problems with Deep Learning

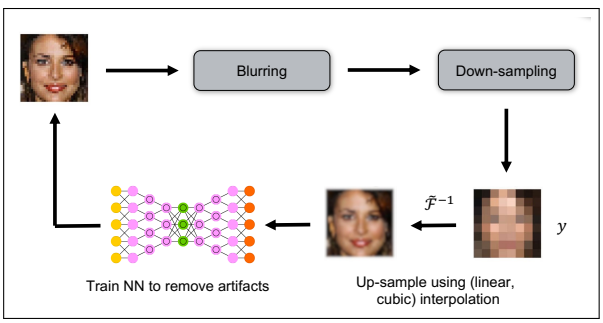
There are three classes of methods

1. Model agnostic (ignores forward model)
2. Decoupled (First learn, then reconstruct)
3. Unrolled optimisation

2.1 Model Agnostic Approach

2.1.1 Solving the inverse problems

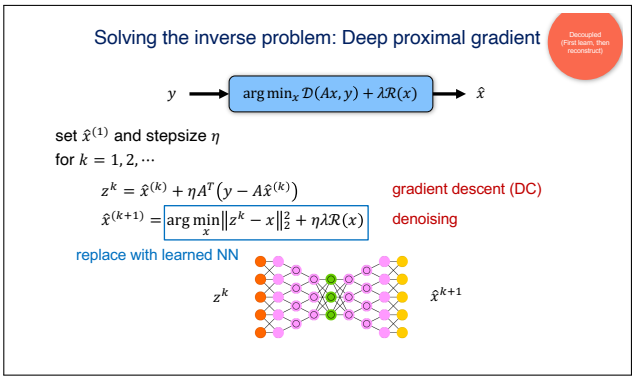




Assume that the paried data (x, y) which are easy to generate. Then take a neural network to solve the inverse problem. The issue here, is that you don't use any information about what happened in the forward operation. The model can be made more intelligent by being partly model agnostic; we can upsample using a classsical algorithm, like linear interpolation, which aren't perfect but produce something useful. Then, after this approximation, you can train a network to remove the difference between the upsampled version (which may have artifacts) and the original.

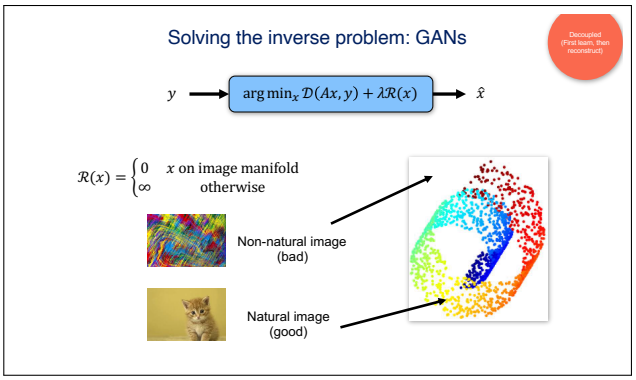
2.2 Decoupled Approach

2.2.1 Deep Proximal Gradient

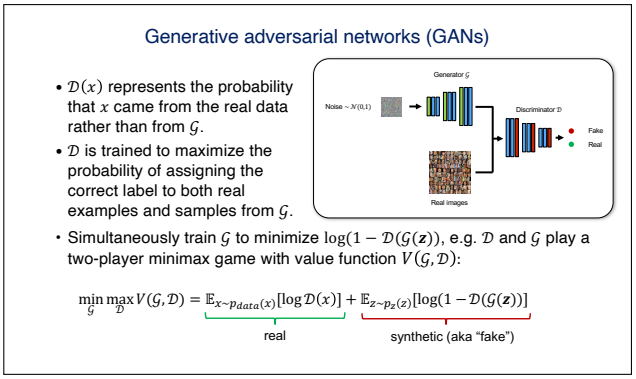


- We're still trying to solve an optimization problem, this time with an iterative optimization.
- Compute one step that minimises D and one step that minimises the image regularsiation term. We estimate using gradient descent.
- The second equation is replaced with a nerual network, which takes the estimate from the gradient descent step and produces a new estimate for the data.

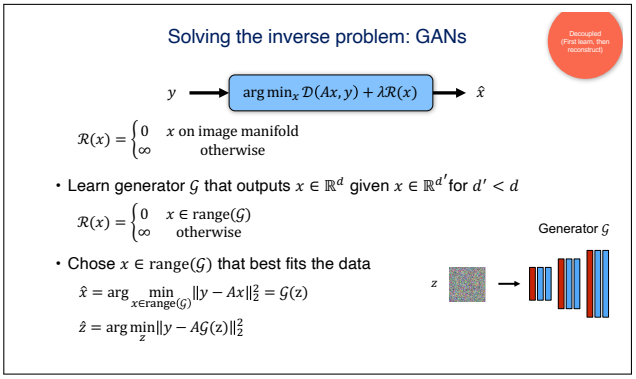
2.2.2 General Adversarial Network



- We want to learn a regularisation function which gives a value of 0 if the image looks realistic and otherwise an infinitely large value.
- If you're on the image manifold then you generate an image which looks real.



- A generator takes a vector of noise and generates images from this noise.
- The output is fed into a discriminator along with real images which tries to understand if an image is real or fake.
- Both are trained simultaneously

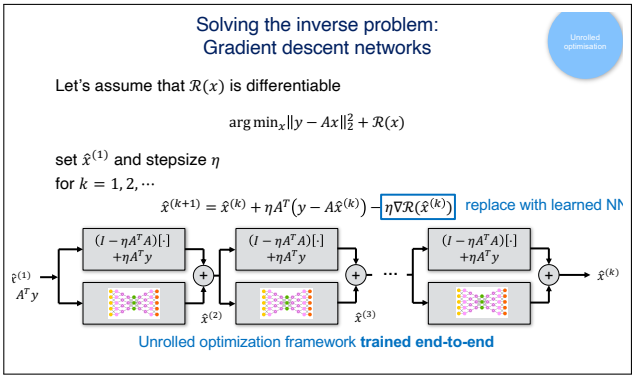


Here, we assume that we know A.

- We can now start plugging in the discriminator or generator into the inverse problem.
- We can choose only solutions of x that can be generated from the generator because the generator can only generate realistic looking images. In the optimisation, we can use the generator to generate solutions that look real.
- Note here, we have restricted the space to the image mainfold shown above using the generator.

2.3 Unrolled optimisation

2.3.1 Gradient descent networks

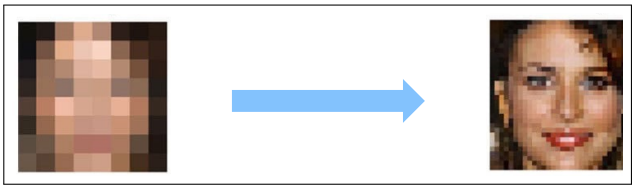


- if we assume that our regularisation term $\mathcal{R}$ is differentiable, (which is why we used the proximal gradient algorithm $\arg\min_x \mathcal{D}(ax, y) + \lambda\mathcal{R}(x)$)
- Then we can minimise the equation, and do a gradient descent step and take a step in the direction of the regularisation.
- The first term corresponds to gradient descent for data consistency, and the second for gradient descent for regularisation

- We can replace the second step with a neural network.
- The diagram: shows an iterative optimisation step, where you start with an initial estimate, you add the gradient with respect to data consistency and the gradient with respect to regularisation and make a step in the direction of the combined gradient.
- the bottom term was replaced with a nerual network. Originally, it was the $\eta\triangledown\mathcal{R}[\cdot]$
- Since we have pairs of x, y we can simulate the transformation and train the network with the transformed output.

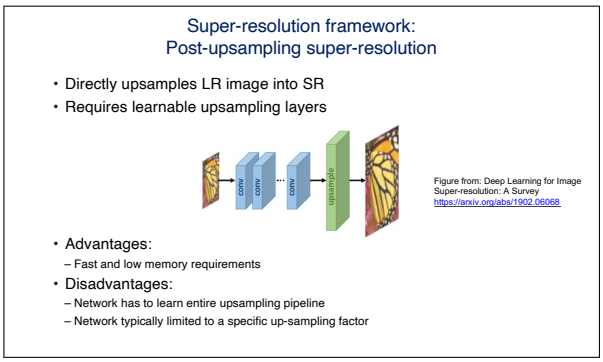
3 Deep Learning for Image Super-Resolution

3.1 Problem formulation

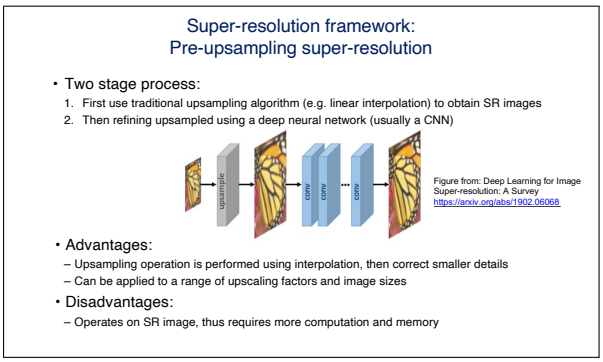


Upsample low-resolution (LR) images to high-resolution (HR or SR) image. The forward model (going from high-to-low-resolution) is straightforward and involves some image degradation followed by downsampling. The Inverse model is for example an interpolation-based model.

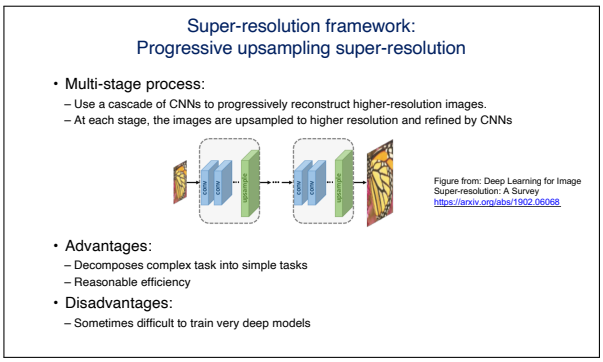
3.2 Example Frameworks



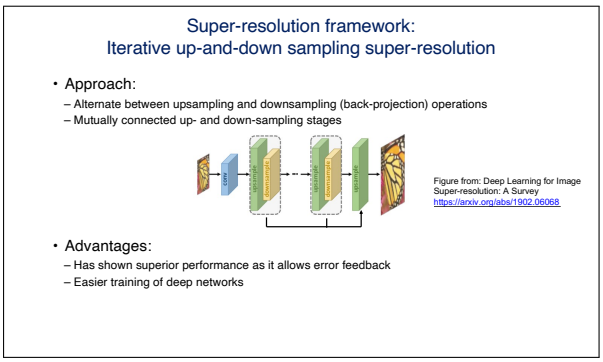
(f) We can have a network with a couple of convolutions and some upsampling



(g) Can upsample data to the desired resolution, then have layers to remove artifacts and add details.

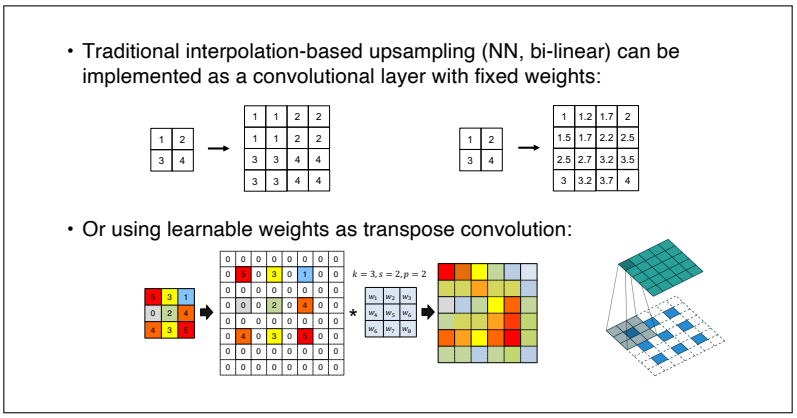


(h) A cascade by upsample by increasing factors (by a bit)

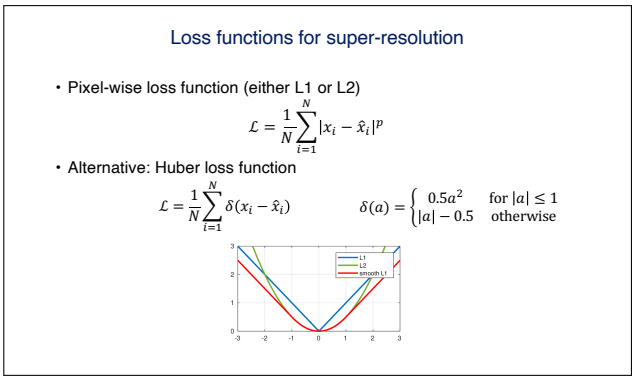


(i) upsample then downsample. Compute the error between the two, and use it in future iterations to improve performance.

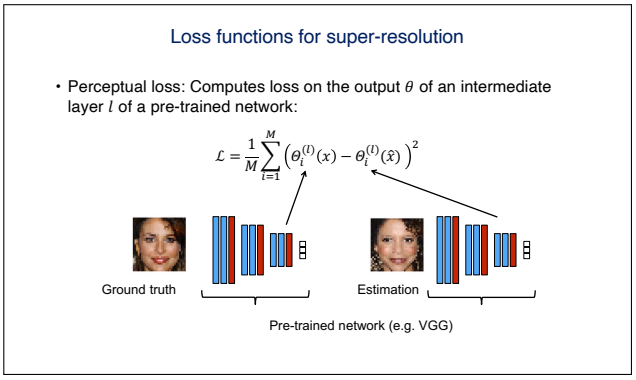
3.3 How to implement upsampling?



3.4 Loss functions for super-resolution

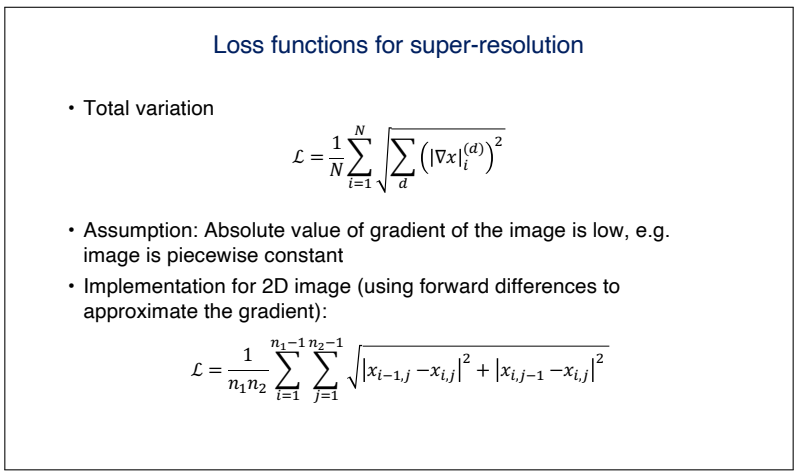


- either take the square or absolute difference between intensities depending on the norm chosen.
- Or a huber loss function (a mix between L1 and L2)
- L1 is not differentiable which causes numerical problems, so people use something that behaves like L1 but is easier to work with. This is the Huber Loss function.



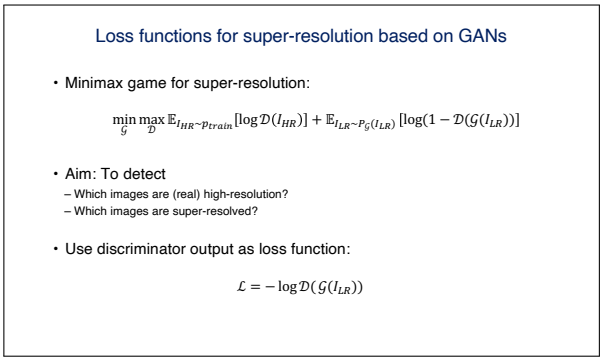
- Don't compare L1 and L2 losses between pixel values. Instead, compute activations in a pre-trained network, like a VGG or ResNet, then compute it for the original data and for the upsampled data.
- Then compute the loss not in terms of pixel intensities but in terms of activations in the network.

- The network has been pre-trained on a task to classify objects for instance, then the activations describe the image in more perceptual terms and that might be a better way of computing values.



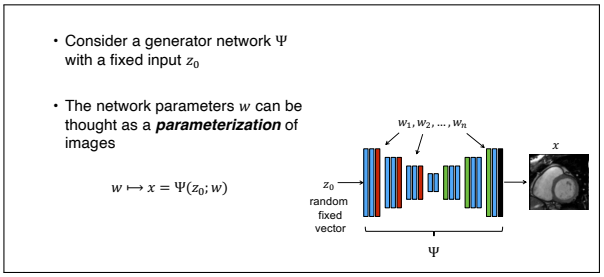
3.5 GANs for Loss

Recall, that the generator tries to generate realistic looking images, and the discriminator tries to identify which images are real and which ones are fake. You can the generative adversarial networks as a loss function.

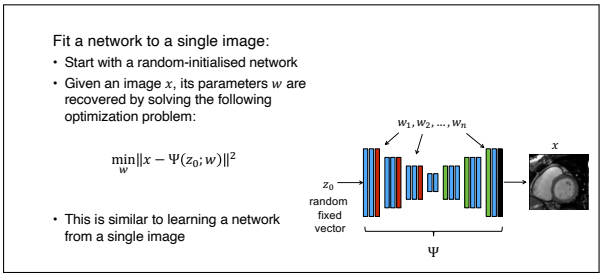


- Generate with a neural network a realistic super-resolved Image
- Use the discriminator to identify how realistic the image is looking.
- If the discriminator tells me the image is realistic, we have a low cost function, otherwise, we get a high value.

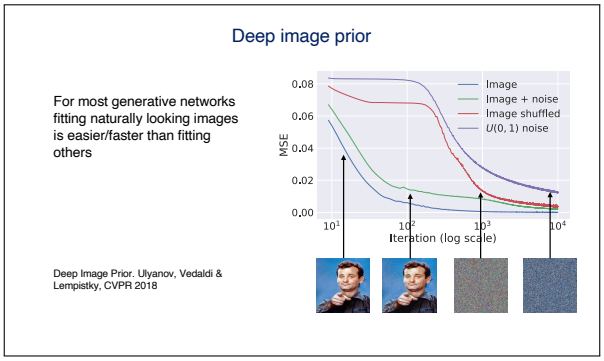
3.6 Deep Image Prior — Parameterizing images via generative networks



(j) Deep image Prior: introduce the idea to repersent an image not by its pixel values, but by the weights of a vector. We take a random fixed vector z_0 as input, and optimise the parameters to obtain an iamge x as output.

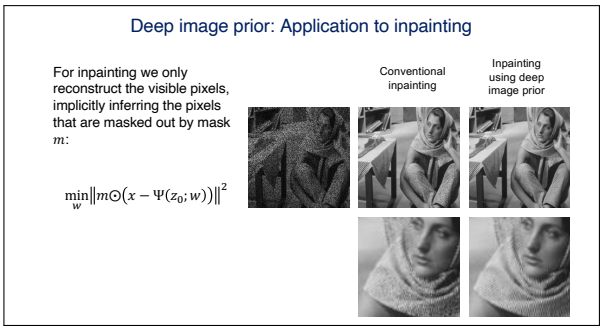


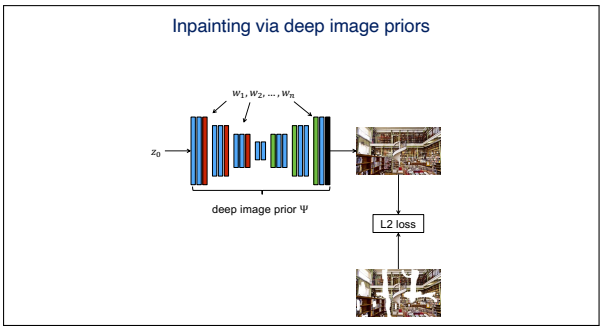
(k) If you learn the weights in this manner, this works well for natural images.



(l) Here we see that generating noise is harder, but real images converge quicker. We can use this to solve image-in-painting. We compare the observed image with the parameterisation through the neural network.

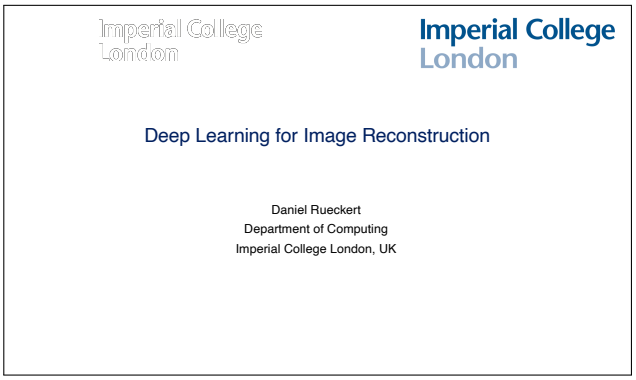
3.6.1 Applciation to inpainting

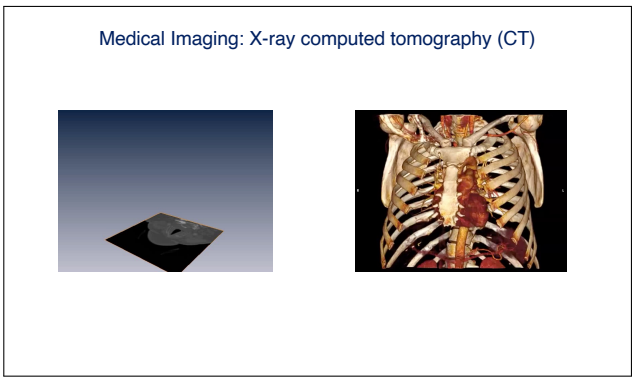


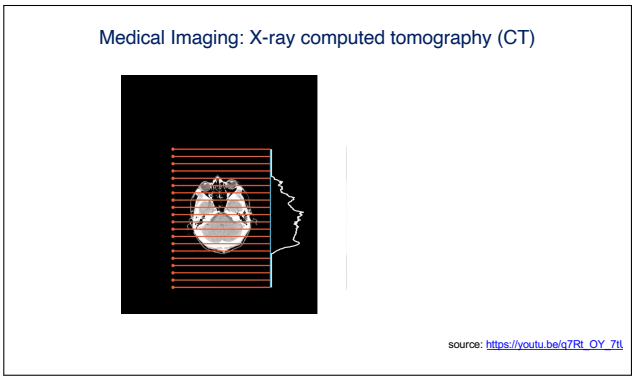


Then, the pixels which are masked out by m and optimise the function over all network weights and learn how to do in-painting. We can then reconstruct the iamge by fitting a noise vector into the image, and reconstruct the image. We achive the following by optimising weights and minimising L2 loss between image generated and the observed image minus the occlusions.

4 Deep Learning for Image Reconstruction

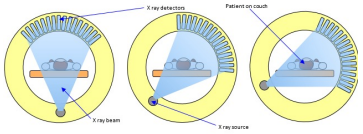




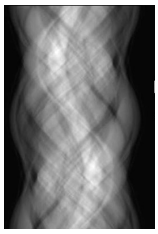


Medical Imaging: X-ray computed tomography (CT)

- CT imaging
  - high contrast
  - high spatial resolution
  - fast acquisition
  - but ionising radiation



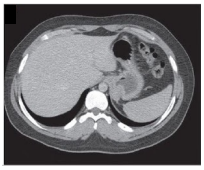
Medical Imaging: X-ray computed tomography (CT)



Forward problem
(Radon transform)

Inverse problem:

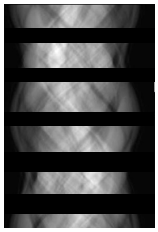
Fully sampled
reconstruction



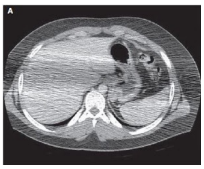
Sinogram

CT image

Medical Imaging: X-ray computed tomography (CT)



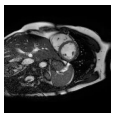
Under-sampled
reconstruction



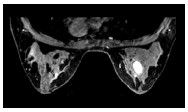
Sinogram

CT image

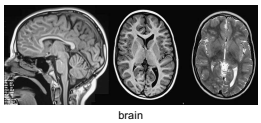
Medical Imaging: Magnetic Resonance (MR)



heart



cancer



brain

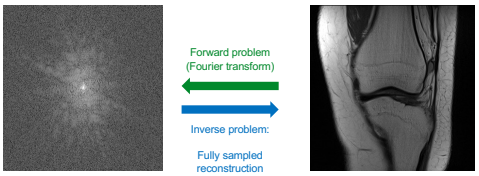
lung

Whole body

Medical Imaging: Magnetic Resonance (MR)

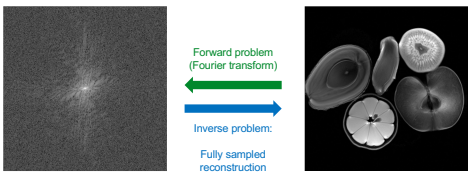
- MR imaging:
  - high contrast
  - high spatial resolution
  - no ionising radiation
  - but slow acquisition process
- Slow acquisition is
  - ok for static objects (e.g. brain, bones, etc)
  - problematic for moving objects (e.g. heart, liver, fetus)
- Options for MR acquisition:
  - real-time MR: fast, but 2D and relatively poor image quality
  - gated MRI fine for period motion, e.g. respiration or cardiac motion but requires gating (ECG or navigators) leading to long acquisition times (30-90 min).

Medical Imaging: Magnetic Resonance (MR)



K-Space MR image

Medical Imaging: Magnetic Resonance (MR)

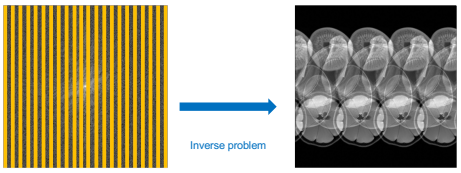


K-Space MR image

Recover an image $x \in \mathbb{K}^{N_x}$ from a set of observations $y \in \mathbb{K}^{N_y}$ which are corrupted by noise $n \in \mathbb{K}^{N_y}$, and A: $\mathbb{K}^{N_x} \to \mathbb{K}^{N_y}$ is a linear operator

$$y = Ax + n$$

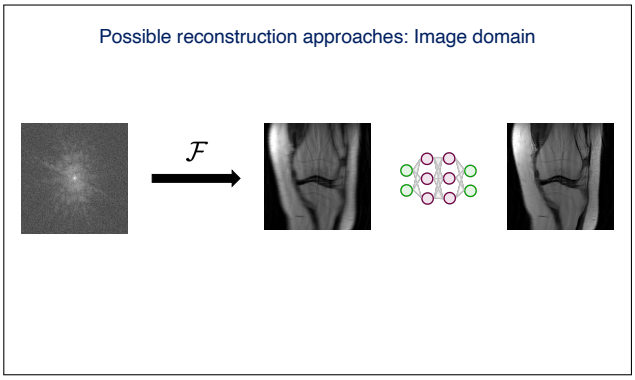
Medical Imaging: Magnetic Resonance (MR)

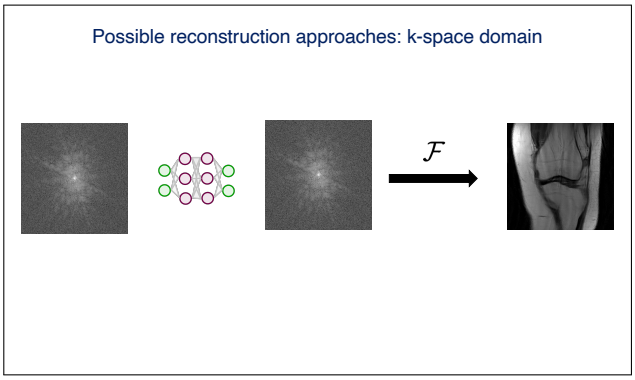


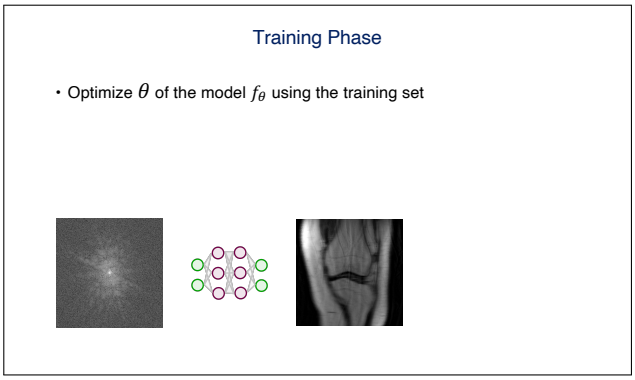
Undersampled K-Space Aliased MR image

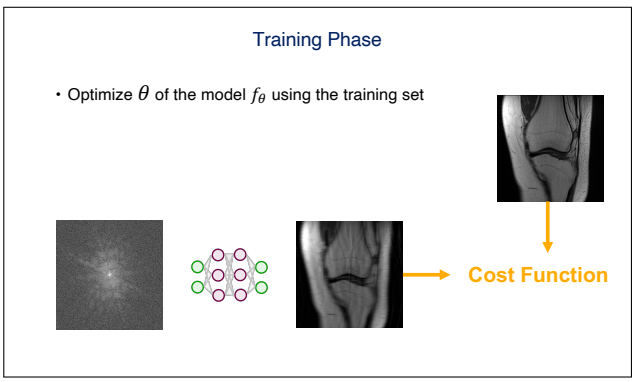
Recover an image $x \in \mathbb{K}^{N_x}$ from a set of observations $y \in \mathbb{K}^{N_y}$ which are corrupted by noise $n \in \mathbb{K}^{N_y}$, and A: $\mathbb{K}^{N_x} \to \mathbb{K}^{N_y}$ is a linear operator

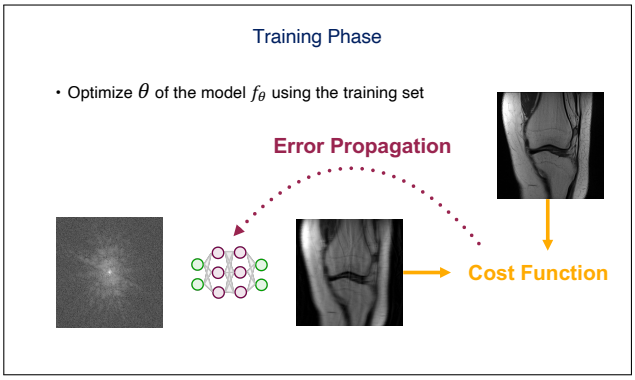
$$y = Ax + n$$

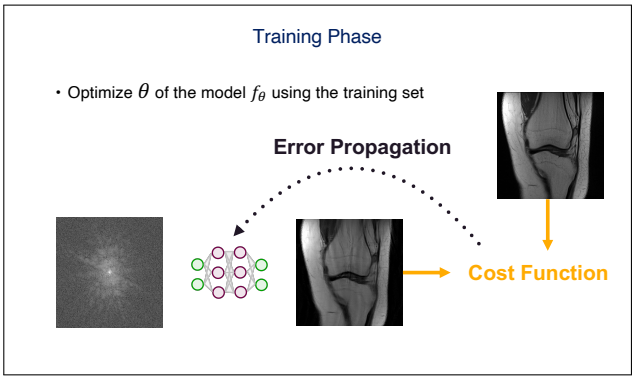

Possible reconstruction approaches: Image domain
$\mathcal{F}$

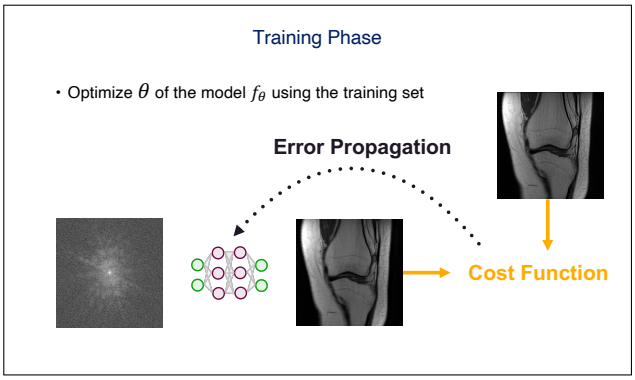

Possible reconstruction approaches: k-space domain
$\mathcal{F}$

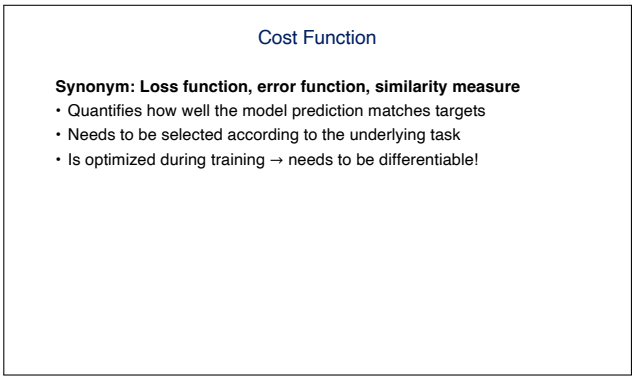

Training Phase
• Optimize θ of the model f_θ using the training set


Training Phase
• Optimize θ of the model f_θ using the training set
Cost Function


Training Phase
• Optimize θ of the model f_θ using the training set
Error Propagation
Cost Function


Training Phase
• Optimize θ of the model f_θ using the training set
Error Propagation
Cost Function


Training Phase
• Optimize θ of the model f_θ using the training set
Error Propagation
Cost Function


Cost Function
Synonym: Loss function, error function, similarity measure
• Quantifies how well the model prediction matches targets
• Needs to be selected according to the underlying task
• Is optimized during training → needs to be differentiable!

Cost Function

Synonym: Loss function, error function, similarity measure

- Quantifies how well the model prediction matches targets
- Needs to be selected according to the underlying task
- Is optimized during training → needs to be differentiable!

Example for image reconstruction: Mean Squared Error

$$MSE(y, \hat{y}) = \frac{1}{N_S} \sum_{n=1}^{N_S} \| y_n - \hat{y}_n \|_2^2$$

Training Phase

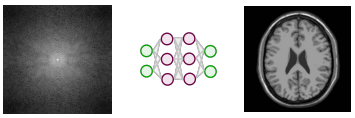
Learning model: $\hat{y}_n = f_\theta(x_n)$

- The parameters θ of the mapping function f_θ are optimized under a cost function
- The cost function quantifies how well $\hat{y}_n = f_\theta(x_n)$ is predicted given x_n
- The parameters θ by minimizing the cost function $\mathcal{L}$ with learning rate τ

$$\theta^{(k+1)} = \theta^{(k)} - \tau \frac{\partial \mathcal{L}(\theta, x)}{\partial \theta}|_{\theta=\theta^{(k)}}$$

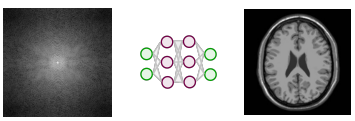
Testing Phase (Inference)

- Apply f_θ using the optimized θ to the test set

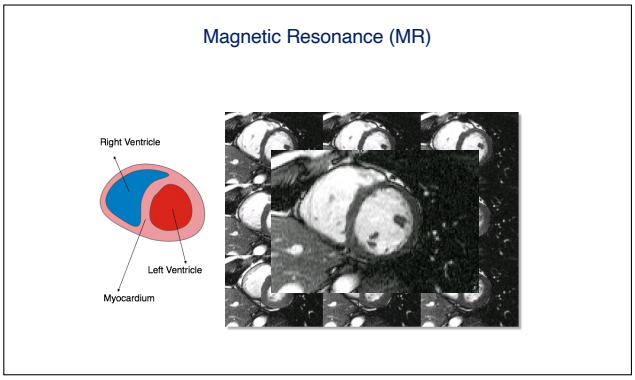


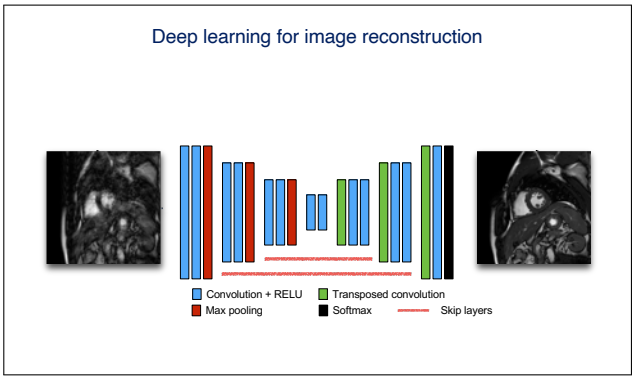
Testing Phase (Inference)

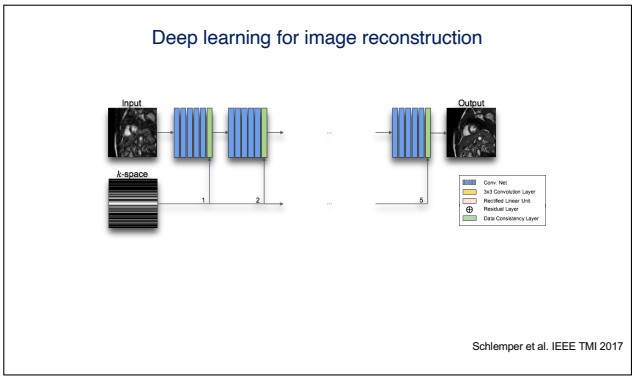
- Apply f_θ using the optimized θ to the test set

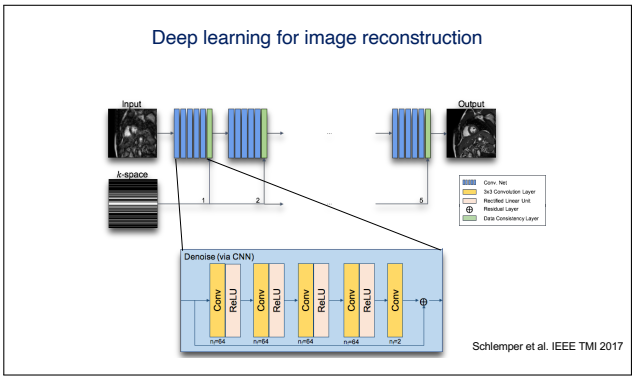


Generalization: Ability to correctly predict unseen examples

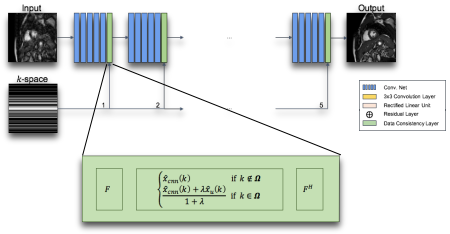

Magnetic Resonance (MR)
Right Ventricle
Left Ventricle
Myocardium


Deep learning for image reconstruction
Convolution + RELU
Max pooling
Transposed convolution
Softmax
Skip layers


Deep learning for image reconstruction
Input
k-space
Output
Schlemper et al. IEEE TMI 2017

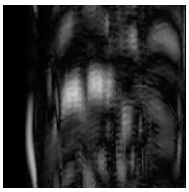

Deep learning for image reconstruction
Input
k-space
Output
Denoise (via CNN)
Conv
ReLU
Schlemper et al. IEEE TMI 2017

Deep learning for image reconstruction



Schlemper et al. IEEE TMI 2017

Magnitude reconstruction (6-fold)



(a) 6x Undersampled

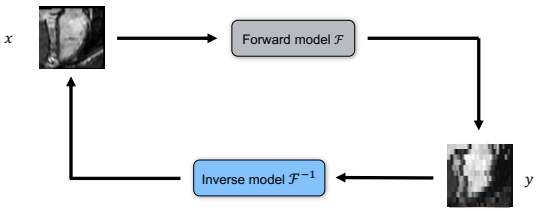
Schlemper et al. IEEE TMI 2017

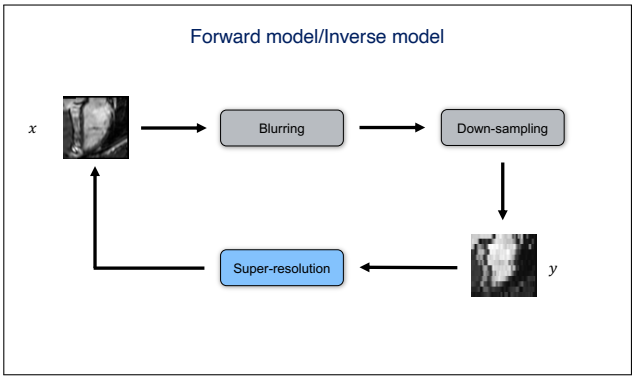
Magnitude reconstruction (11-fold)

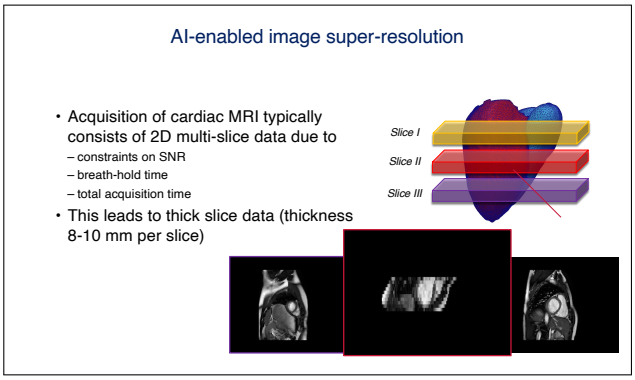
(a) 11x Undersampled

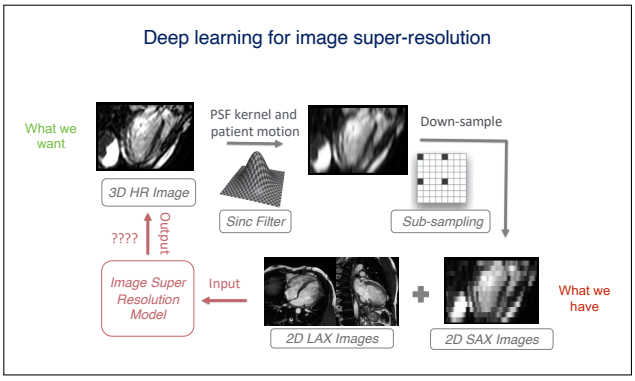
Schlemper et al. IEEE TMI 2017

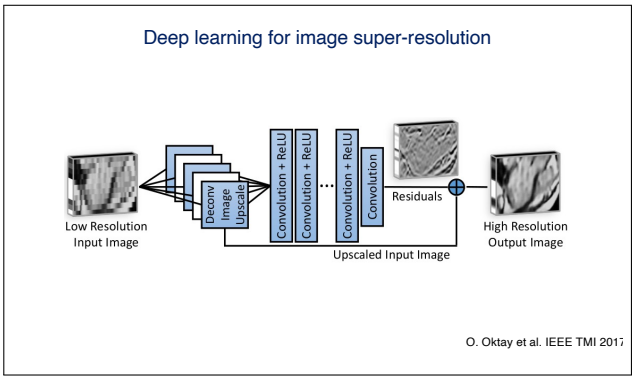
Forward model/Inverse model

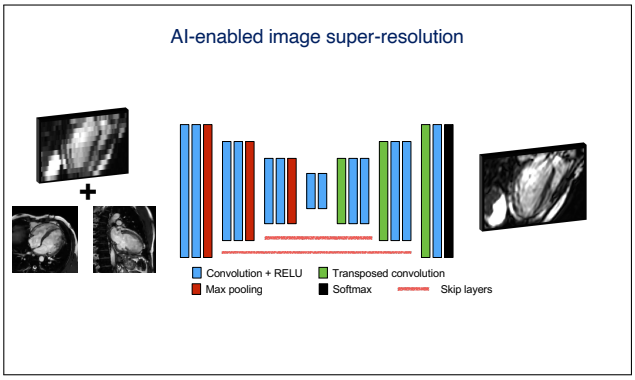


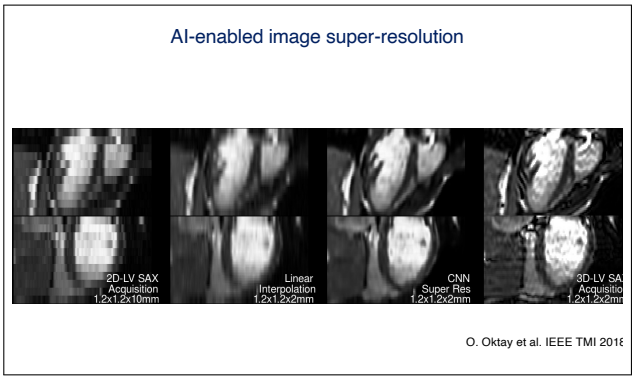

Forward model/Inverse model
x
Blurring
Down-sampling
Super-resolution
y


AI-enabled image super-resolution
• Acquisition of cardiac MRI typically consists of 2D multi-slice data due to
– constraints on SNR
– breath-hold time
– total acquisition time
• This leads to thick slice data (thickness 8-10 mm per slice)
Slice I
Slice II
Slice III


Deep learning for image super-resolution
What we want
3D HR Image
PSF kernel and patient motion
Sinc Filter
Down-sample
Sub-sampling
????
Output
Image Super Resolution Model
Input
2D LAX Images
2D SAX Images
What we have


Deep learning for image super-resolution
Low Resolution Input Image
Deconv Image Upscale
Convolution + ReLU
Convolution + ReLU
Convolution + ReLU
Convolution
Residuals
Upscaled Input Image
High Resolution Output Image
O. Oktay et al. IEEE TMI 2017


AI-enabled image super-resolution
Convolution + RELU
Transposed convolution
Max pooling
Softmax
Skip layers


AI-enabled image super-resolution
2D-LV SAX
Acquisition
1.2x1.2x10mm
Linear
Interpolation
1.2x1.2x2mm
CNN
Super Res
1.2x1.2x2mm
O. Oktay et al. IEEE TMI 201


That's all for now…