Basics of AI

And Some X-Ray or CT Examples

Marc Kachelrieß

German Cancer Research Center (DKFZ)

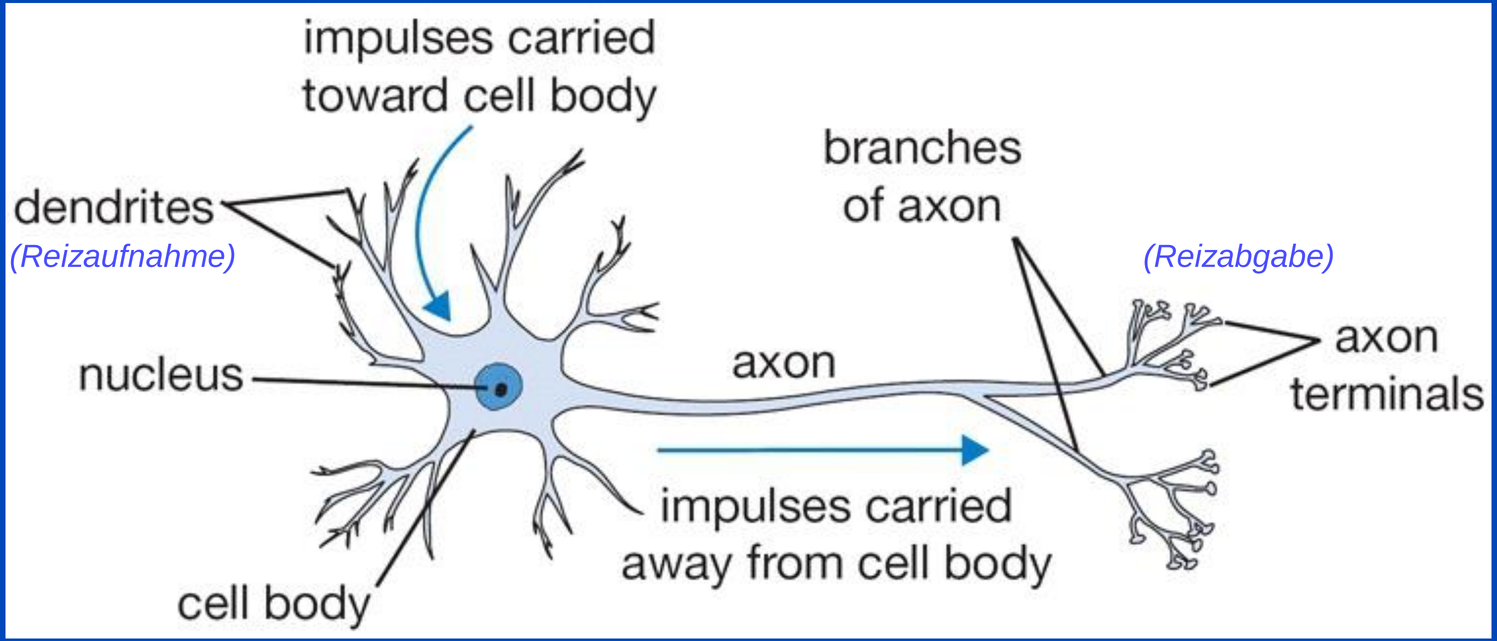
Heidelberg, Germany

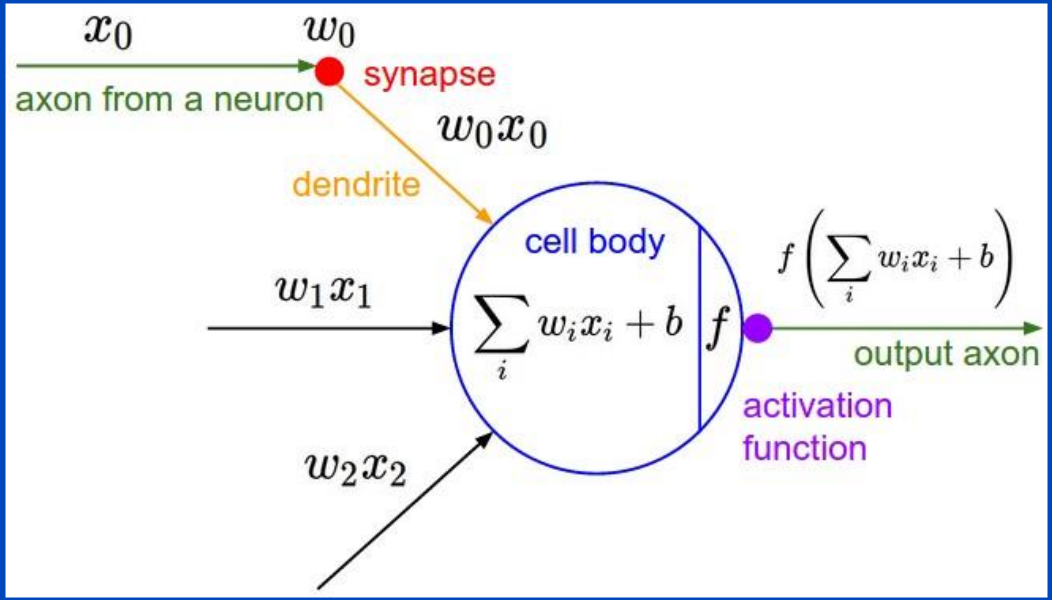
www.dkfz.de/ct

DEUTSCHES KREBSFORSCHUNGSZENTRUM IN DER HELMHOLTZ-GEMEINSCHAFT

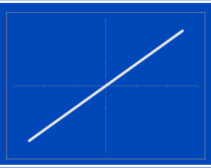
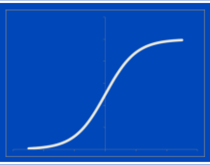
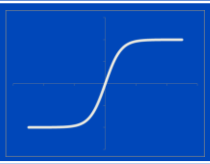
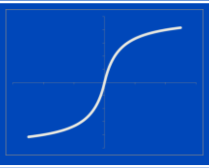
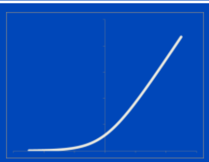
Nomenclature

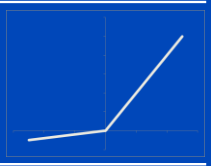
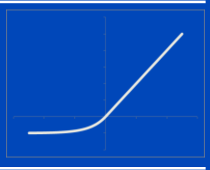
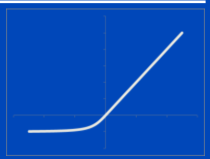
- Iteration = Epoch
- Batch = Subset (randomly changing for each epoch)
- Loss function = Cost function
- Learning rate = η


impulses carried
toward cell body
dendrites
(Reizaufnahme)
branches
of axon
(Reizabgabe)
nucleus
axon
axon
terminals
impulses carried
away from cell body
cell body


x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$f\left(\sum_i w_i x_i + b\right)$
$w_1 x_1$
$\sum_i w_i x_i + b$
f
output axon
activation
function
$w_2 x_2$

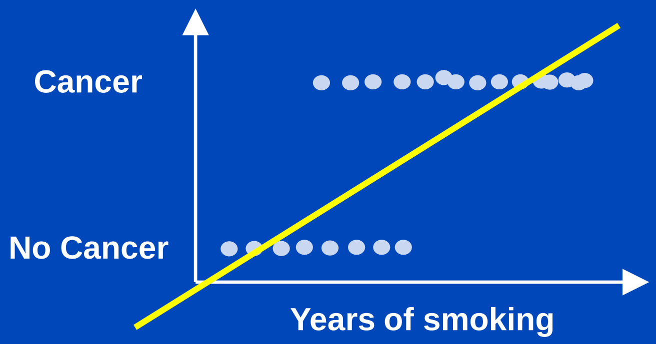
Activation Functions

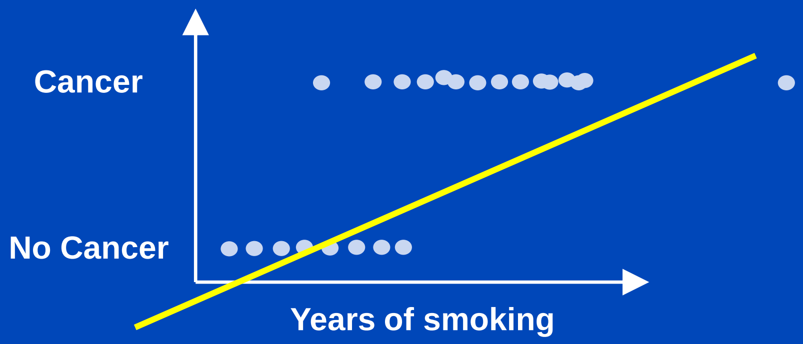
Function	Equation	Plot
Identity	$f(x) = x$	
Sigmoid	$f(x) = \dfrac{1}{1+e^{-x}}$	
Hard sigmoid	$f(x) = \begin{cases} 0 & \text{for } x < -\alpha \\ \frac{\alpha + x}{2\alpha} & \text{for } -\alpha \leq x < \alpha \\ 1 & \text{for } x \geq \alpha \end{cases}$	
Tanh	$f(x) = \dfrac{2}{1+e^{-2x}} - 1$	
Softsign	$f(x) = \dfrac{x}{1+\lvert x \rvert}$	
Softplus	$f(x) = \log(1 + \exp x)$	

Function	Equation	Plot
ReLU	$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	
Leaky ReLU	$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	
ELU	$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	
Inverse square root LU	$f(x) = \begin{cases} \frac{x}{\sqrt{1+\alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	
…	…	…

Binary Logistic Regression

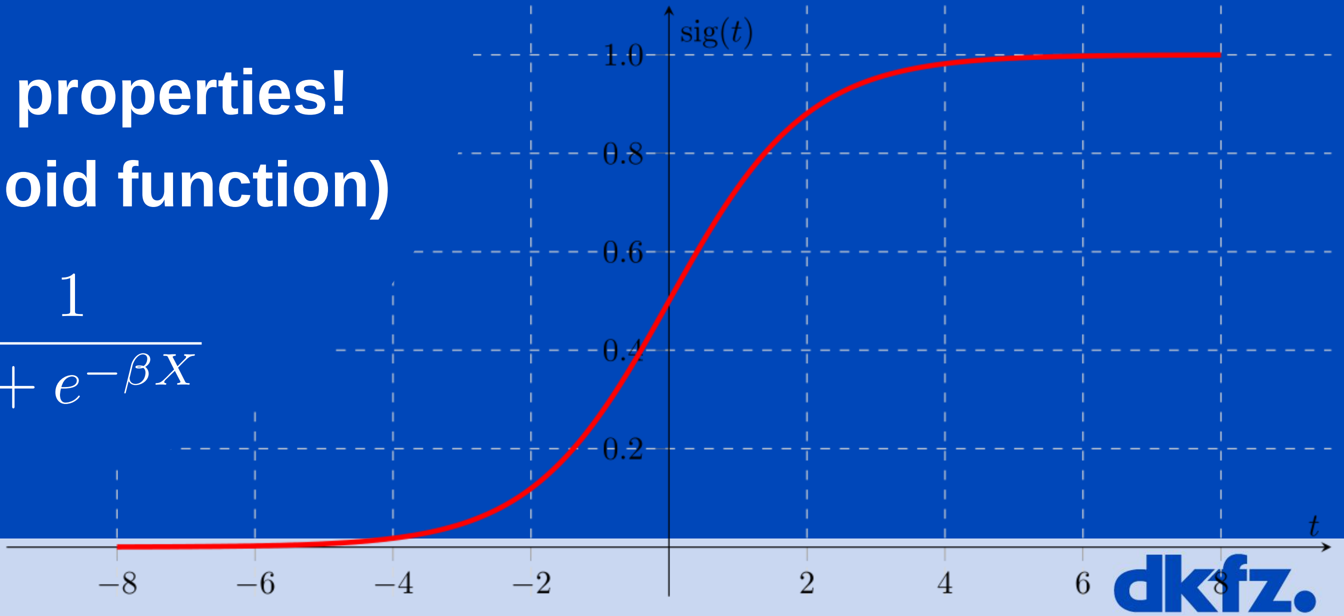
- What if *y* is categoric, e.g. $y \in \{0, 1\}$?





- Linear regression has undesired properties!
- Use logistic model instead (sigmoid function)

$$p(x) = p(t(x)) = \frac{1}{1 + e^{-t(x)}} = \frac{1}{1 + e^{-\beta X}}$$



Loss Function

- The neural network coefficients (weights and biases) *c* are chosen by minimizing a loss function (cost function)

$$\boldsymbol{c} = \arg\min_{\boldsymbol{c}} \sum_{n=1}^{N} L(\boldsymbol{c}, \boldsymbol{x}_n, \boldsymbol{y}_n)$$

  with x_n being the training data input, $y(c, x_n)$ being the network output, and y_n being the so-called labels, i.e. the training target, and N being the number of training samples.
- An example for such a loss function is the MSE loss

$$L(\boldsymbol{c}, \boldsymbol{x}_n, \boldsymbol{y}_n) = \big(\boldsymbol{y}(\boldsymbol{c}, \boldsymbol{x}_n) - \boldsymbol{y}_n\big)^2$$

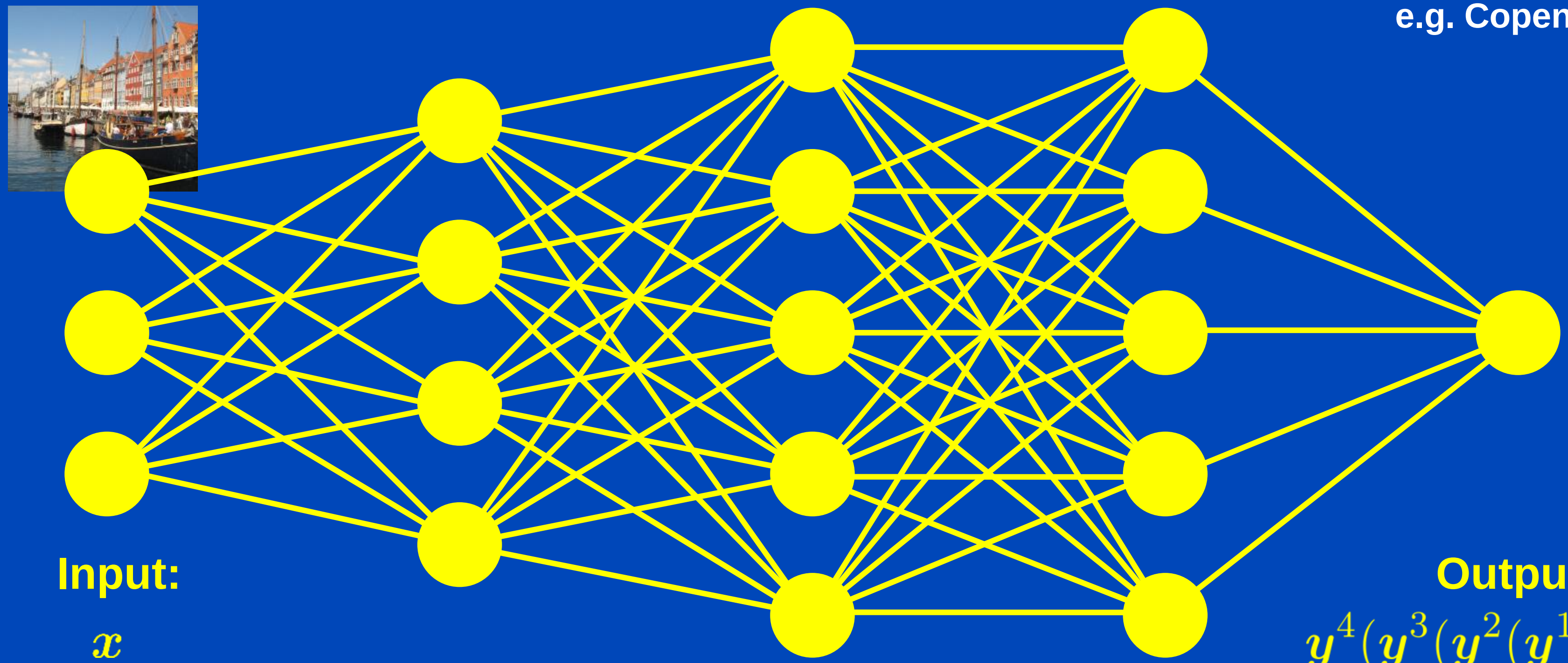
Fully-Connected Neural Network

- Each layer fully connects to previous layer
- Difficult to train (many parameters in *W* and *b*)
- Spatial relations not necessarily preserved

Input | Hidden | Hidden | Hidden | Output

e.g. 512×512×3 pixels
e.g.

e.g. 1 label
e.g. Copenhagen



Input:
x

Output:
$y^4(y^3(y^2(y^1(x))))$

$$\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{W}\cdot\boldsymbol{x}+\boldsymbol{b}) \text{ with } \boldsymbol{f}(\boldsymbol{x}) = \big(f(x_1), f(x_2), \ldots\big) \text{ point-wise scalar, e.g. } f(x) = x \vee 0 = \mathrm{ReLU}$$

Universal Approximation Theorem[1]

- **A fully-connected network with at least three layers is also called a multi layer perceptron (MLP).**

If σ is continuous, bounded and nonconstant, then $\Re_M(\sigma)$ is dense in the subset C of continuous and compact functions of $\mathbb{R}^M$.

$$\Re_M(\sigma) = \bigcup_N \Re_M^{(N)}(\sigma)$$

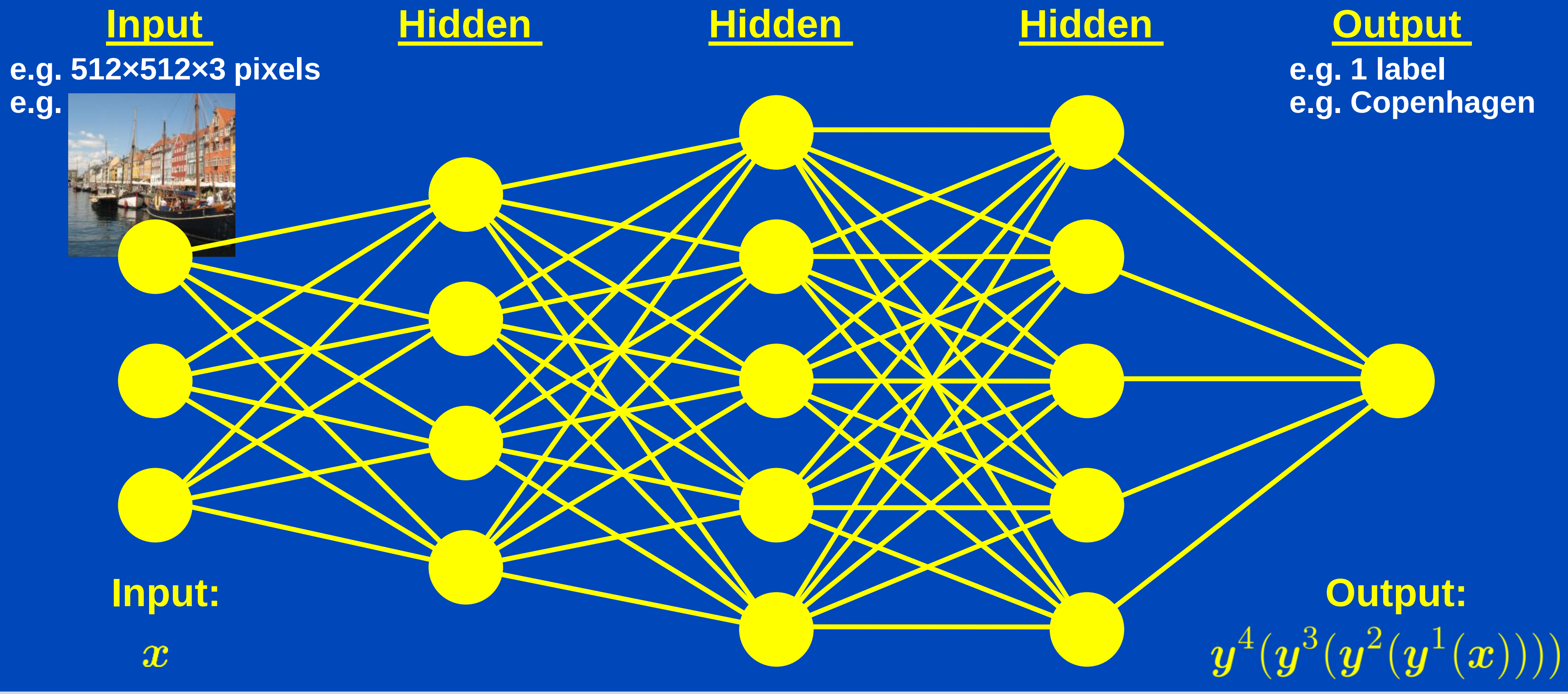
$$= \bigcup_N \left\{ g : \mathbb{R}^M \to \mathbb{R} \;\middle|\; \overbrace{g(x) = \sum_{j=0}^{N} w_{k,j}^{(3)}\, \sigma\left(\textstyle\sum_{i=0}^{M} w_{j,i}^{(2)} x_i\right)}^{\text{3-layer MLP with final activation linear and } N \text{ hidden neurons}} \right\}$$

- **For any function $h(x) \in C \in \mathbb{R}^M$ we can find a function $g(x) \in \mathcal{R}_M(\sigma)$ for which $\|h(x) - g(x)\|_p < \epsilon$.**
- **Any 3-layer MLP with appropriately chosen layer sizes and activation function, e.g. the sigmoid function, is a universal function approximator.**
- **This theorem does not provide any insight into how to find the unknowns!**

[1]Hornik, Kurt; Stinchcombe, Maxwell; White, Halbert (1989). Multilayer Feedforward Networks are Universal Approximators. Neural Networks. Vol. 2. Pergamon Press. pp. 359–366; Theorem 2

Fully-Connected Neural Network

- Each layer fully connects to previous layer
- Difficult to train (many parameters in *W* and *b*)
- Spatial relations not necessarily preserved



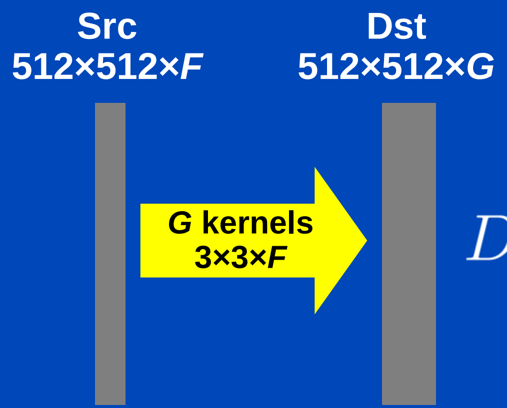
$\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{W}\cdot\boldsymbol{x}+\boldsymbol{b})$ with $\boldsymbol{f}(\boldsymbol{x}) = \big(f(x_1), f(x_2), \ldots\big)$ point-wise scalar, e.g. $f(x) = x \vee 0 = \mathrm{ReLU}$

Convolutional Neural Network (CNN)

- Replace dense *W* in $y(x) = f(W \cdot x + b)$ by a sparse matrix *W* with sparsity being of convolutional type (band diagonal of Toeplitz type).
- CNNs consist (mainly) of convolutional layers.
- Convolutional layers are not fully connected.
- Convolutional layers are small, say 3×3, convolution kernels whose entries need to be found by training.
- CNNs preserve spatial relations to some extent.

$$\boldsymbol{W}_{1\mathrm{D}} = \begin{pmatrix} 1 & 8 & 0 & 0 & 0 & 0 \\ 2 & 1 & 8 & 0 & 0 & 0 \\ 0 & 2 & 1 & 8 & 0 & 0 \\ 0 & 0 & 2 & 1 & 8 & 0 \\ 0 & 0 & 0 & 2 & 1 & 8 \\ 0 & 0 & 0 & 0 & 2 & 1 \end{pmatrix}$$

Only three unknowns!



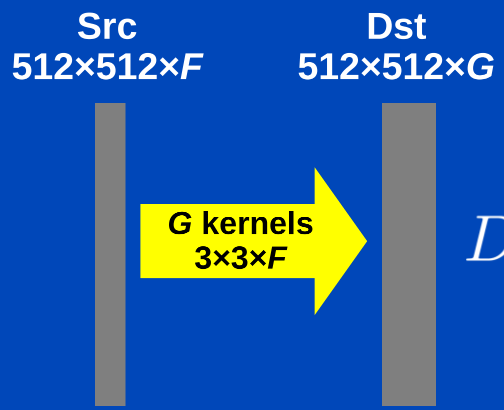
$$D_{i,j,g} = \sum_f S_{i,j,f} * K^g_{i,j,f} = \sum_{a,b,f} S_{i-a,j-b,f} K^g_{a,b,f}$$

Attention: No convolution in depth direction!

Here, a 2D example is shown. Conv layers also exist in 3D and higher dimensions.

Convolution Layers

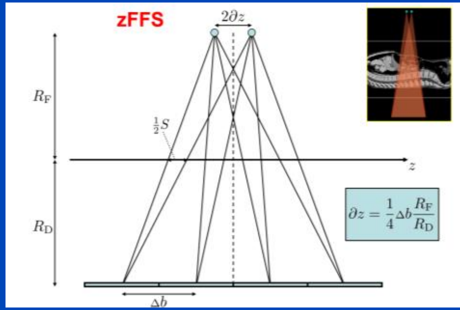
- **Input layer *S***
  - vector of size *I* with *F* features: *I*×*F*
  - image of size *I* by *J* with *F* features: *I*×*J*×*F*
  - volume of size *I* by *J* by *K* with *F* features: *I*×*J*×*K*×*F*
  - …
- **Convolution kernel *K***
  - G kernels of size (2*A*+1)×(2*B*+1)×*F* with or without padding*
- **Output layer *D***
  - same spatial dimensions as input layer*
  - *G* features (depth *G*)



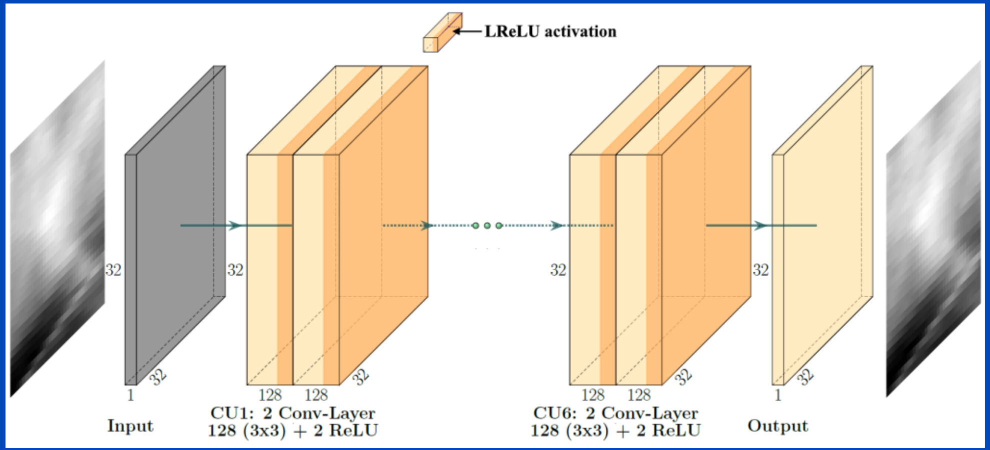
$$D_{i,j,g} = \sum_f S_{i,j,f} * K^g_{i,j,f} = \sum_{a,b,f} S_{i-a,j-b,f} K^g_{a,b,f}$$

Attention: No convolution in depth direction!

*Convolution may include a stride (step size) > 1. Similar to convolution with stride 1 follwed by pooling.



Row Interpolation to Mimick zFFS



Ground truth with zFFS | **Without zFFS (LI)** | **RIDL trained with synthetic data; single input**

C = 0 HU, *W* = 200 HU

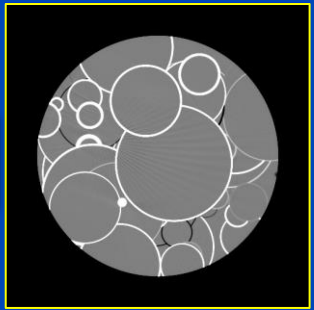
C = 60 HU, *W* = 360 HU

C = -400 HU, *W* = 1500 HU

C = 60 HU, *W* = 400 HU

Example projection

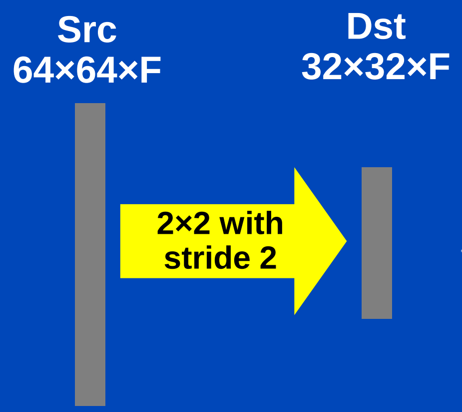
Random synthetic phantom



C = 0 HU, *W* = 400 HU

Pooling
Downsampling

- **Input layer *S***
  - image of size *I* by *J* with *F* features: *I×J×F*
  - …
- **Pooling kernel**
  - pooling function, e.g. max, mean, stochastic, …
  - size and strides
- **Output layer *D***
  - reduced spatial size
  - same depth



$$D_{i,j,f} = \max_{b,d} S_{ai+b,cj+d,f}$$

1	1	1	3	2	3	1	2
2	3	0	3	1	9	6	9
1	8	0	4	0	8	9	9
1	1	2	3	9	2	3	1
0	5	1	3	2	1	1	3
1	1	1	1	0	0	1	1
2	5	0	7	1	9	7	9
2	0	0	8	2	4	0	1

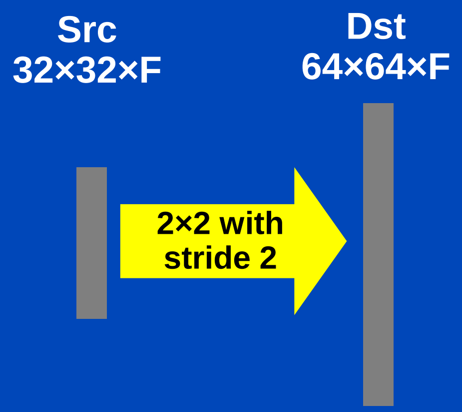
2×2 stride 2×2 max pool

3	3	9	9
8	4	9	9
5	3	2	3
5	8	9	9

Unpooling

Upsampling

- **Input layer *S***
  - image of size *I* by *J* with *F* features: *I*×*J*×*F*
  - …
- **Unpooling kernel**
  - pooling function, e.g. max, mean, stochastic, …
  - size and strides
- **Output layer *D***
  - increased spatial size
  - same depth



0	0	0	3	0	0	0	0
0	3	0	3	0	9	0	9
0	8	0	4	0	0	9	9
0	0	0	0	9	0	0	0
0	5	0	3	2	0	0	3
0	0	0	0	0	0	0	0
0	5	0	0	0	9	0	9
0	0	0	8	0	0	0	0

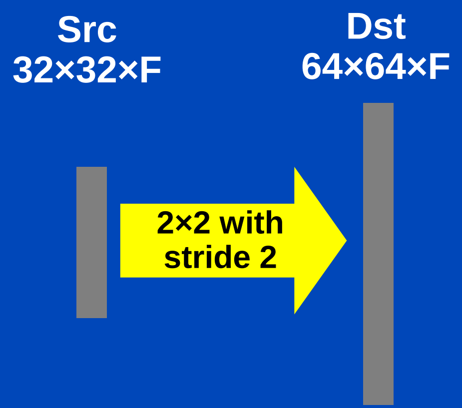
← **2×2 stride 2×2 max unpool**

3	3	9	9
8	4	9	9
5	3	2	3
5	8	9	9

Max values at max positions that were originally found during pooling. Zeroes at non-max positions.

Unpooling
Upsampling

- **Input layer *S***
  - image of size *I* by *J* with *F* features: *I*×*J*×*F*
  - …
- **Unpooling kernel**
  - pooling function, e.g. max, mean, stochastic, …
  - size and strides
- **Output layer *D***
  - increased spatial size
  - same depth



3	3	3	3	9	9	9	9
3	3	3	3	9	9	9	9
8	8	4	4	9	9	9	9
8	8	4	4	9	9	9	9
5	5	3	3	2	2	3	3
5	5	3	3	2	2	3	3
5	5	8	8	9	9	9	9
5	5	8	8	9	9	9	9

2×2 stride 2×2
max unpool

3	3	9	9
8	4	9	9
5	3	2	3
5	8	9	9

Max values at all positions.

Dilated Convolutions

- Convolution

$$D_{i,j,g} = \sum_f S_{i,j,f} * K^g_{i,j,f} = \sum_{a,b,f} S_{i-a,j-b,f} K^g_{a,b,f}$$

- 8-dilated convolution

$$D_{i,j,g} = \sum_f S_{i,j,f} *_8 K^g_{i,j,f} = \sum_{a,b,f} S_{i-8a,j-8b,f} K^g_{a,b,f}$$

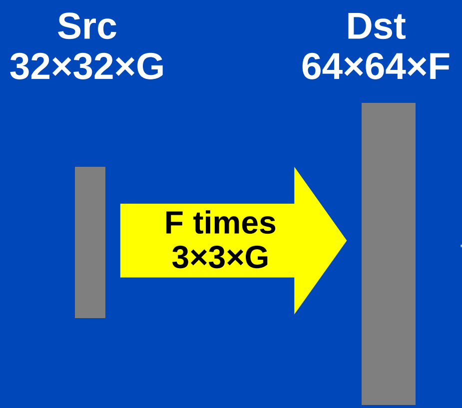
- Dilation helps to increase the receptive field of the kernel without increasing the number of unknowns in the kernel.
- Similar effect as pooling followed by convolution.

Transposed Convolution

- **Sometimes also called fractionally-strided convolution layer or deconvolution layer**
- **Deconvolution layer is a very unfortunate name and should rather be called a transposed convolutional layer.**

$$D_{i,j,g} = \sum_{f} S_{2i,2j,f} * K^{g}_{i,j,f} = \sum_{a,b,f} S_{2i-a,2j-b,f} K^{g}_{a,b,f}$$

Convolution with stride 2

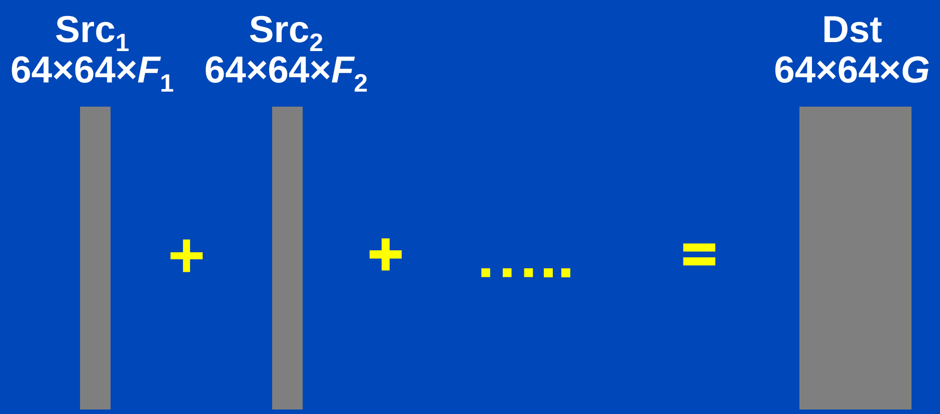


Transposed convolution with stride 2

$$D_{i,j,f} = \sum_{a,b,g} S_{i/2+a,j/2+b,g} K^{f}_{a,b,g}$$

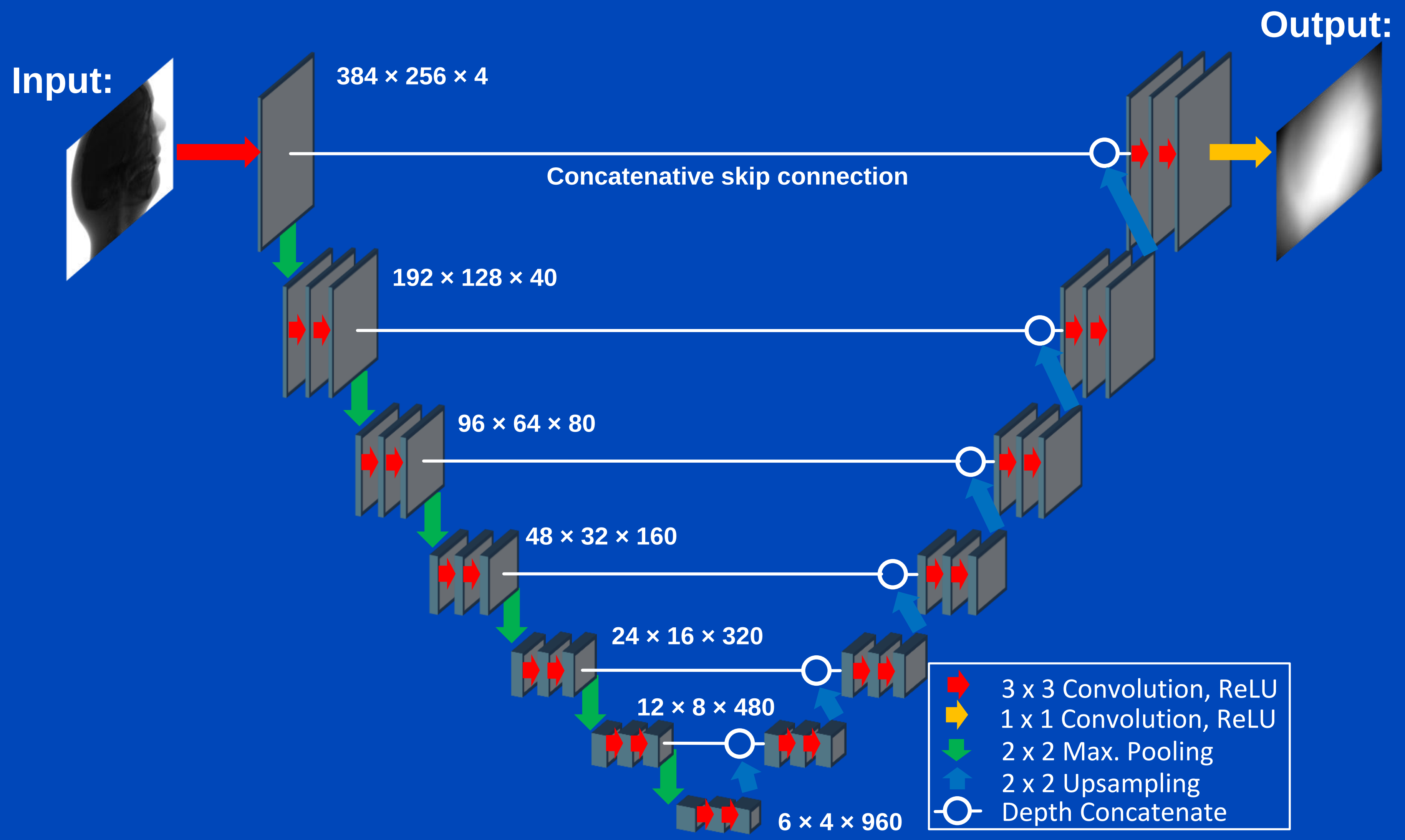
Depth Concatenation

- N input layers S_n
  - vector of size I with F_n features: $I \times F_n$
  - image of size I by J with F_n features: $I \times J \times F_n$
  - volume of size I by J by K with F_n features: $I \times J \times K \times F_n$
  - …
- Output layer D
  - same spatial dimensions as input layer
  - $G = F_1 + F_2 + \ldots + F_N$ features



$$G = \sum_n F_n$$

U-Net[1]

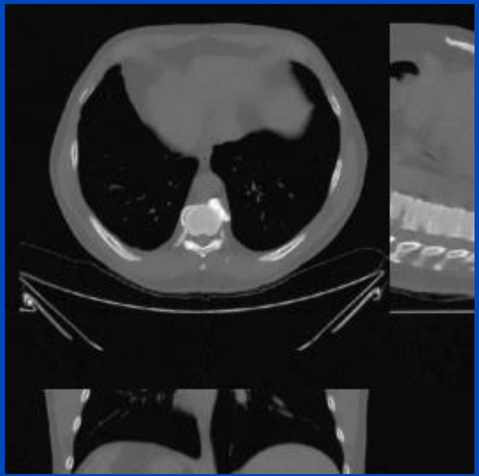


[1]O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. Proc. MICCAI:234-241, 2015.

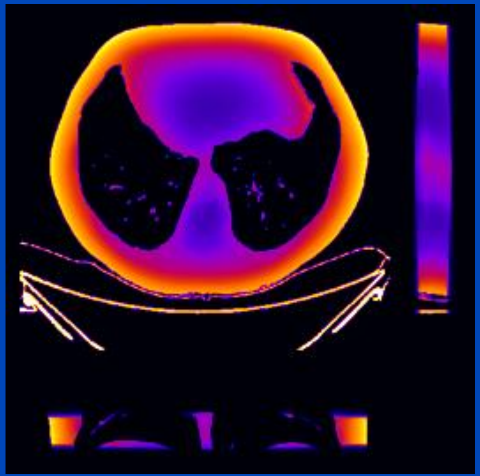
U-Net Example: The Deep Dose Estimation (DDE)

Thorax, tube A, 120 kV, no bowtie

CT image



First order dose

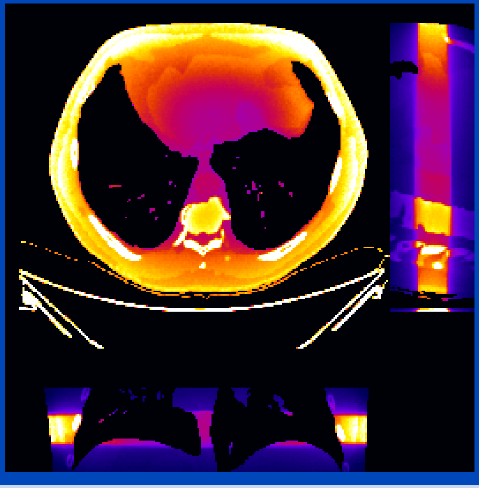


	MC	DDE
48 slices	1 h	0.25 s
whole body	20 h	5 s

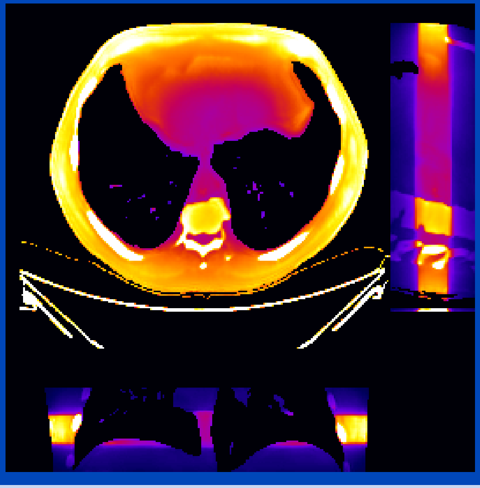
MC uses 16 CPU kernels
DDE uses one Nvidia Quadro P600 GPU

DDE training took 74 h for 300 epochs, 1440 samples, 48 slices per sample

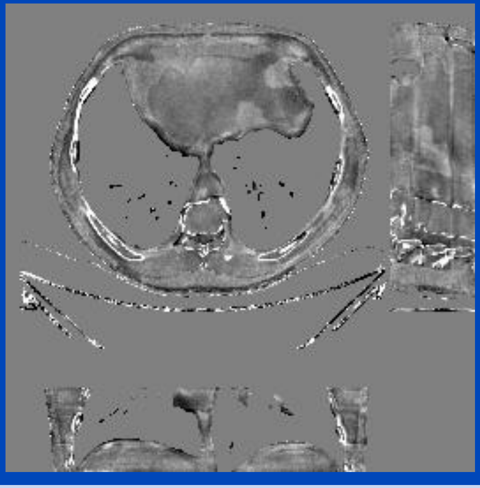
MC ground truth



DDE



Relative error



C = 0%
W = 40%

J. Maier, L. Klein, E. Eulig, S. Sawall, and M. Kachelrieß. Real-time estimation of patient-specific dose distributions for medical CT using the deep dose estimation. Med. Phys. 49(4):2259-2269, April 2022. *Best Paper within Machine Learning at ECR 2019!*

Monte Carlo (180 min)

Compute times as of 2021

Deep Dose Estimation (2 s)

Percentage Error

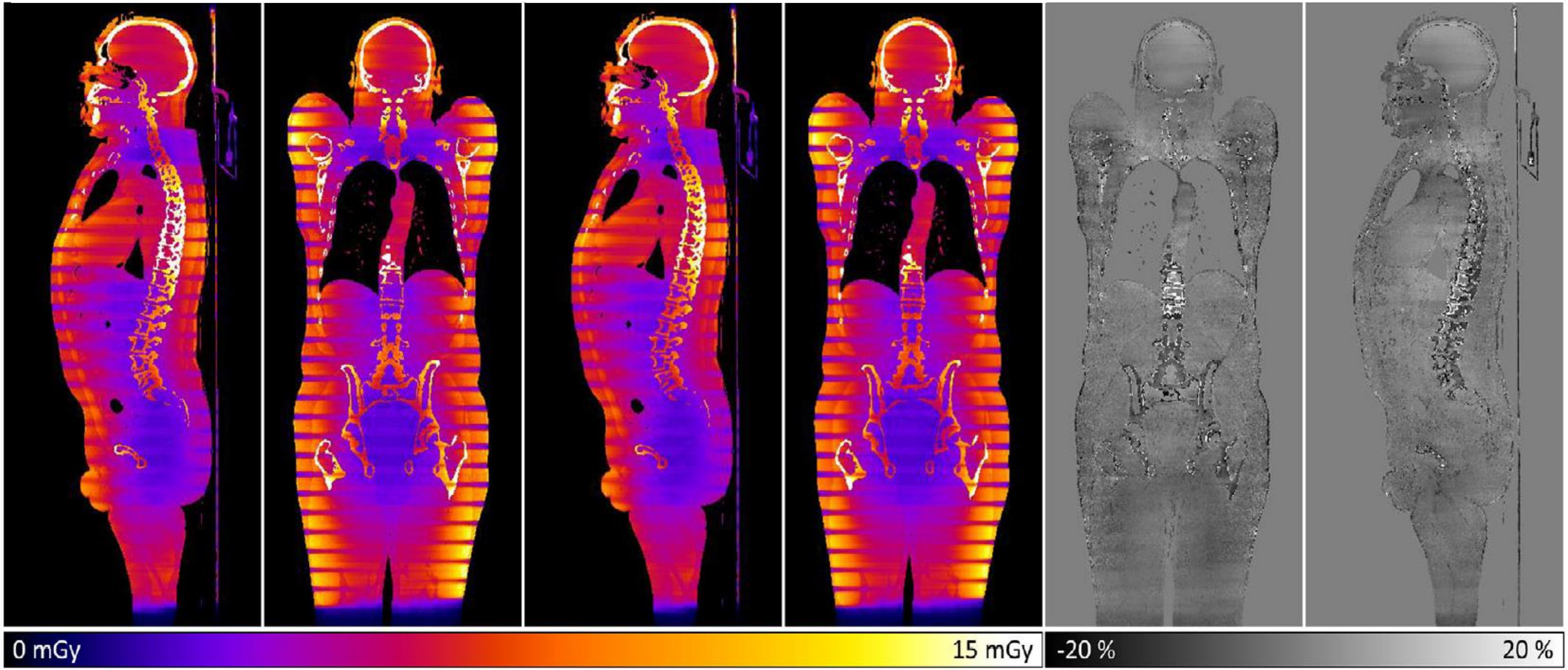
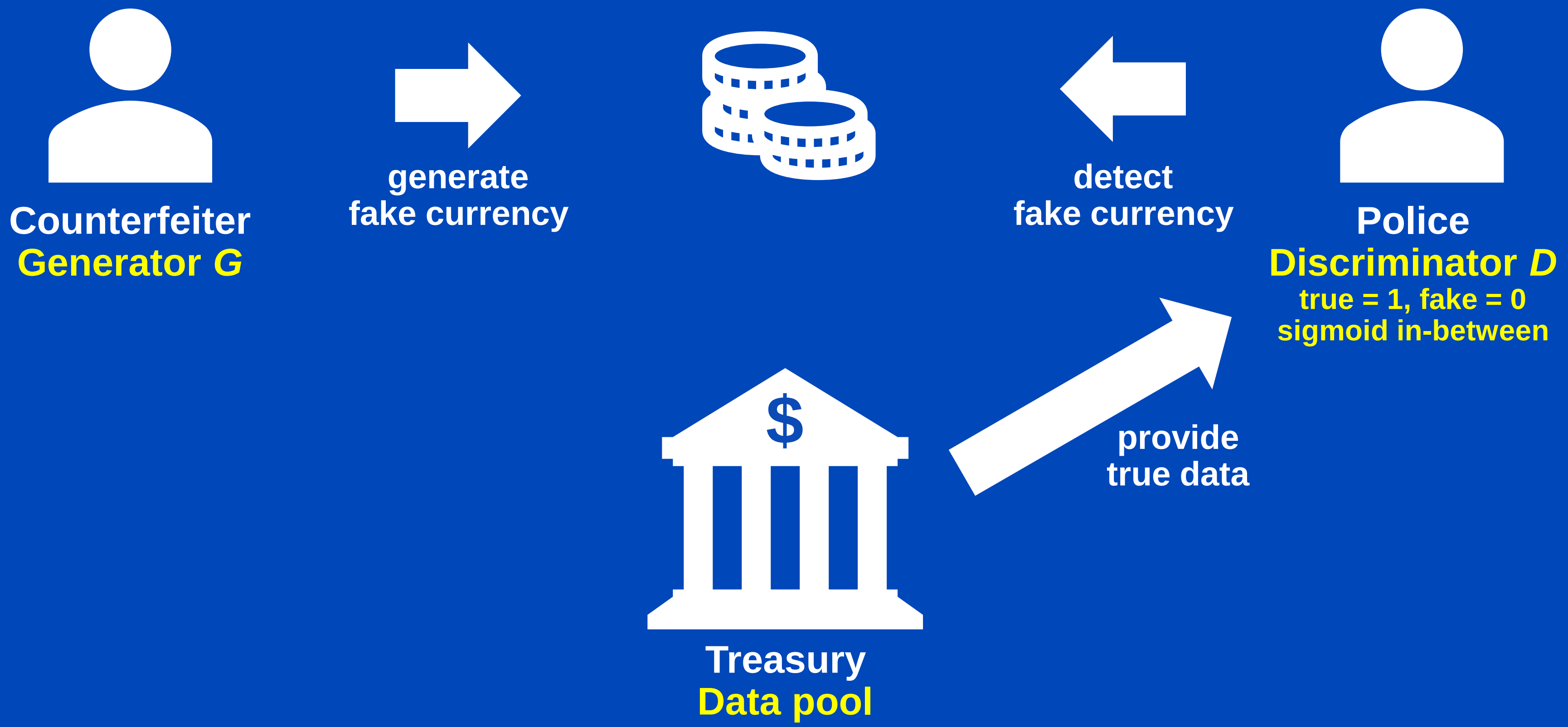


FIGURE 5 Sagittal and coronal view of the dose distribution of a 100 kV whole-body spiral computed tomography (CT) scan including a bowtie filter and an angular tube current modulation. Here, the two left columns show the ground truth, the middle columns show the deep dose estimation (DDE) prediction and the two right columns the corresponding percentage error. Note that dose to air is neglected for computational reasons, and therefore, displayed as zero

Generative Adversarial Network[1] (GAN)

- Useful, if no direct ground truth (GT) is available, the training data are unpaired, unsupervised learning

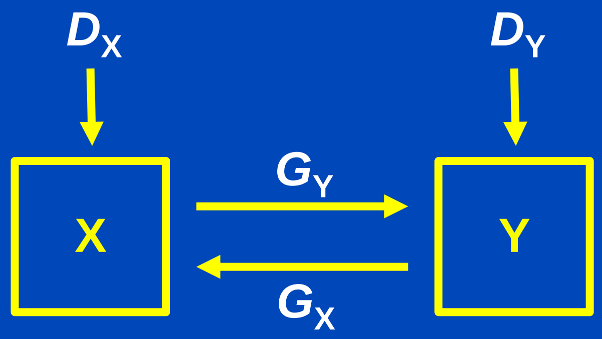


Generative Adversarial Network (GAN)

- **Typical loss function and minimax game:**

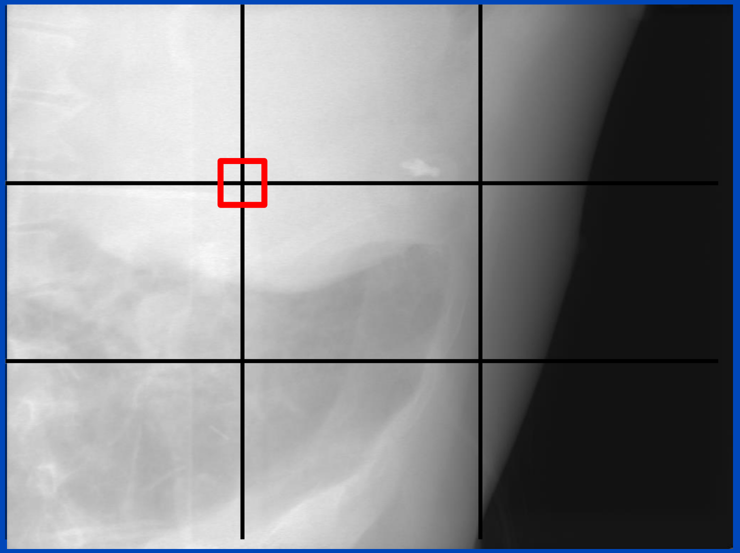
$$\min_G \max_D L(D,G) := \mathrm{E}_x \ln \big(1 - D(G(x))\big) + \mathrm{E}_y \ln D(y)$$

- **Conditional GAN[1]**
  - **Conditional GANs sample the generator input *x* not from a uniform distribution but from a conditional distribution, e.g. noisy CT images.**
  - **Need some measure to ensure similarity to input distribution (e.g. pixelwise loss added to the minimax loss function)**
- **Cycle GAN[2]**
  - **Two GANs ($X \rightarrow Y$ and $Y \rightarrow X$)**
  - **Demand cyclic consistency, i.e. $x = G_X(G_Y(x))$ and $y = G_Y(G_X(x))$**

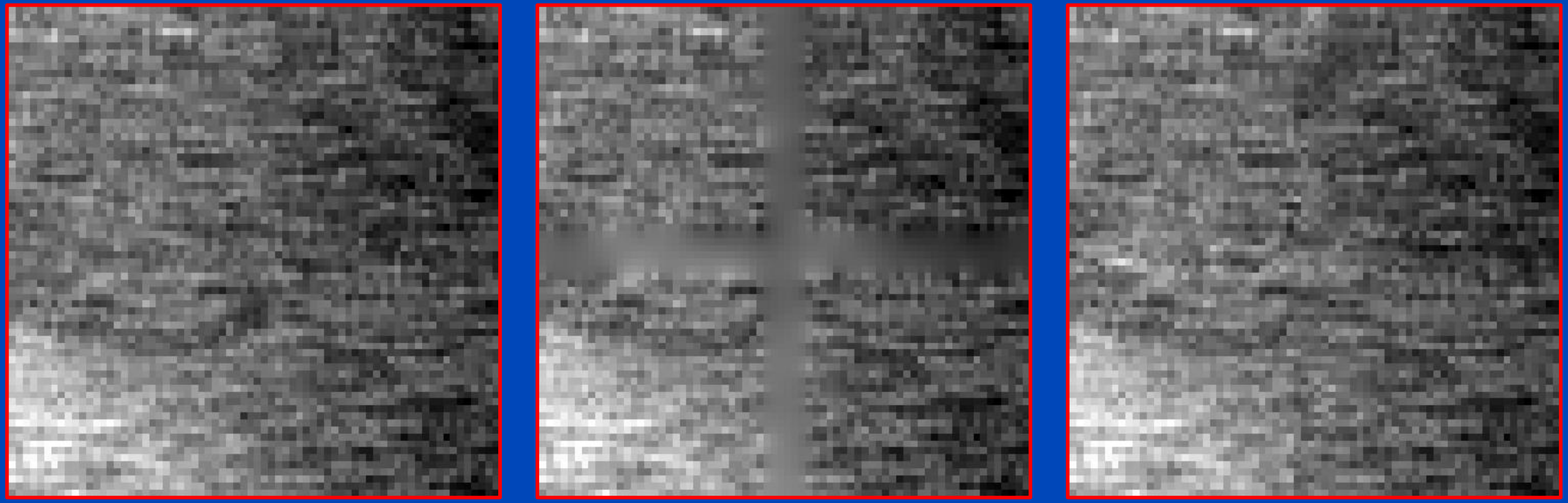


[1]Isola et al. 2017
[2]Zhu et al., 2017

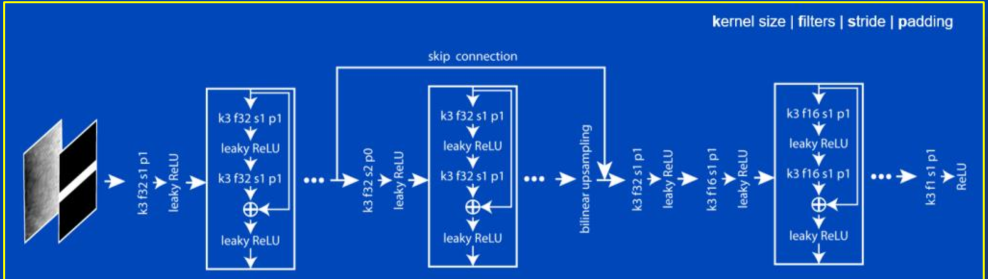
Example: Inpainting



Original | CNN with MSE only | CNN with GAN



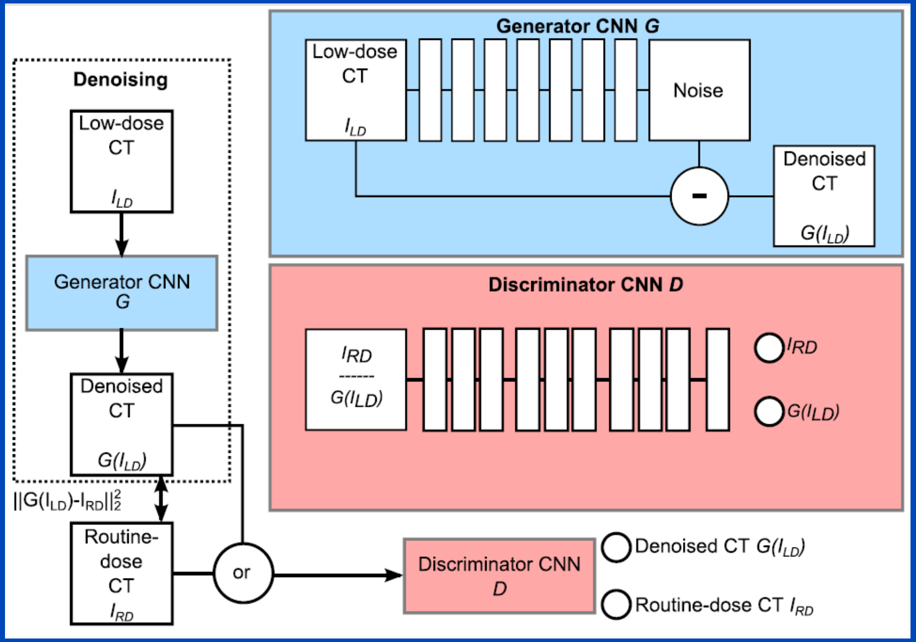
$$G(x) =$$



- **Receives patch and mask as input**
- **Fully convolutional**
- **Leaky ReLUs as nonlinearities to further training stability**
- **Consists of several residual blocks**
- **One downsampling with skip connection to upsampled image to increase the receptive field.**

GAN Example: Noise Removal with GAN

- Task: Reduce noise from low dose CT images.
- A conditional generative adversarial networks (GAN) is used
- Generator *G*:
  - 3D CNN that operates on small cardiac CT sub volumes
  - Seven 3×3×3 convolutional layers yielding a receptive field of 15×15×15 voxels for each destination voxel
  - Depths (features) from 32 to 128
  - Batch norm only in the hidden layers
  - Subtracting skip connection
- Discriminator *D*:
  - Sees either routine dose image or a generator-denoised low dose image
  - Two 3×3×3 layers followed by several 3×3 layers with varying strides
  - Feedback from *D* prevents smoothing.
- Training data:
  - 120 kV
  - Unenhanced (why?) patient data acquired with Philips Briliance iCT 256.
  - Two scans (why?) per patient, one with 0.2 mSv and one with 0.9 mSv effective dose.



J. Wolterink, T. Leiner, M. Viergever, and I. Išgum. Generative Adversarial Networks for Noise Reduction in Low-Dose CT. IEEE TMI 36(12):2536-2544, Dec. 2017.

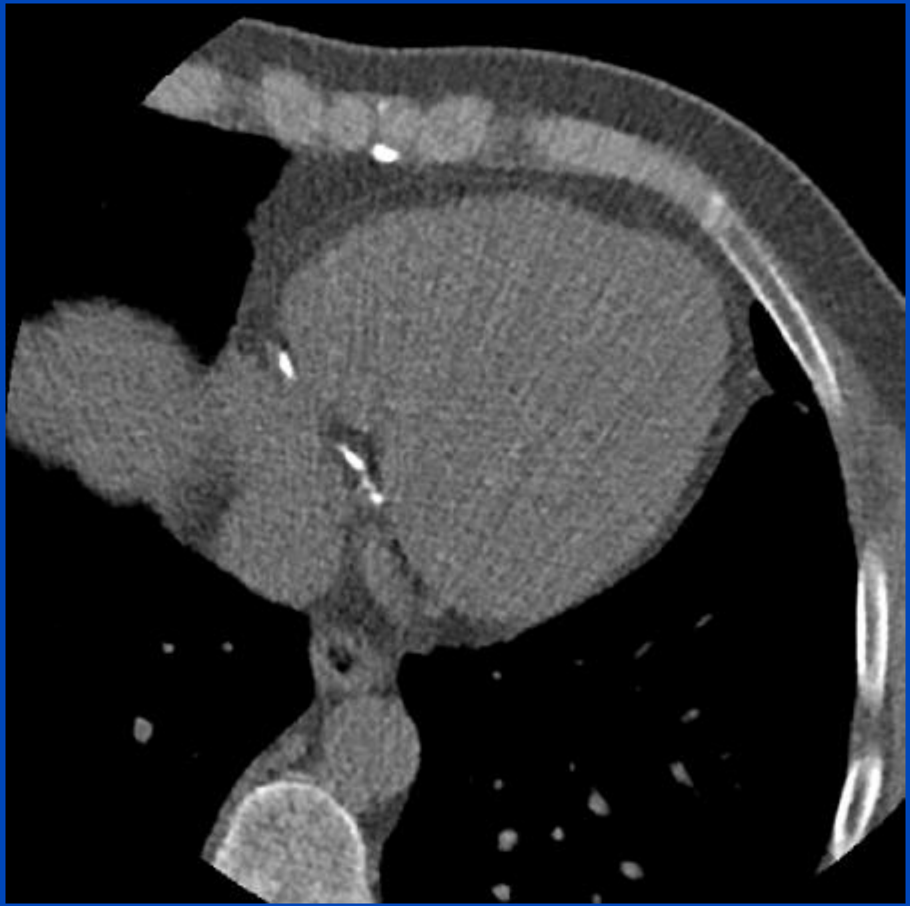
dkfz.

GAN Example: Noise Removal with GAN

Low dose image (0.2 mSv)

J. Wolterink, T. Leiner, M. Viergever, and I. Išgum. Generative Adversarial Networks for Noise Reduction in Low-Dose CT. IEEE TMI 36(12):2536-2544, Dec. 2017.

GAN Example: Noise Removal with GAN



iDose level 3 reconstruction (0.2 mSv)

GAN Example: Noise Removal with GAN

Denoised low dose image (0.2 mSv)

J. Wolterink, T. Leiner, M. Viergever, and I. Išgum. Generative Adversarial Networks for Noise Reduction in Low-Dose CT. IEEE TMI 36(12):2536-2544, Dec. 2017.

GAN Example: Noise Removal with GAN

Normal dose image (0.9 mSv)

J. Wolterink, T. Leiner, M. Viergever, and I. Išgum. Generative Adversarial Networks for Noise Reduction in Low-Dose CT. IEEE TMI 36(12):2536-2544, Dec. 2017.

Loss Function

- The neural network coefficients (weights and biases) *c* are chosen by minimizing a loss function (cost function)

$$\boldsymbol{c} = \arg\min_{\boldsymbol{c}} \sum_{n=1}^{N} L(\boldsymbol{c}, \boldsymbol{x}_n, \boldsymbol{y}_n)$$

  with x_n being the training data input, $y(c, x_n)$ being the network output, and y_n being the so-called labels, i.e. the training target, and *N* being the number of training samples.
- An example for such a loss function is the MSE loss

$$L(\boldsymbol{c}, \boldsymbol{x}_n, \boldsymbol{y}_n) = \big(\boldsymbol{y}(\boldsymbol{c}, \boldsymbol{x}_n) - \boldsymbol{y}_n\big)^2$$

Gradient Descent – A Method to find *L*‘s Minimum

- Walk along the direction of the negative gradient
- Steepest descent
- Learning rate η

$$\boldsymbol{c}^{\text{new}} = \boldsymbol{c}^{\text{old}} - \eta\, \boldsymbol{\nabla}_{\boldsymbol{c}} L(\boldsymbol{c}, \boldsymbol{x}_n, \boldsymbol{y}_n)$$

- Easy to understand, but not optimal
- Methods in use
  - Batch gradient descent
  - Sochastic gradient descent
  - Mini-batch gradient descent
  - Conjugate gradient descent
  - Quasi Newton methods
  - Momentum methods
  - …

Toy Example

Nested scalar functions *f*(*c*, *x*) with unknown coefficients *c*

Loss Function

last layer output, 2nd layer output, 1st layer output, input, training target

$$L(c_3, c_2, c_1, x) = \left(f_3(c_3, f_2(c_2, f_1(c_1, x))) - y\right)^2$$

Intermediate Values

$$L_3 = \frac{dL}{df_3} = 2(f_3 - y)$$

$$L_2 = \frac{dL}{df_2} = \frac{dL}{df_3}\frac{df_3}{df_2} = L_3 \frac{df_3}{df_2}$$

$$\dots$$

$$L_n = \frac{dL}{df_n} = L_{n+1}\frac{df_{n+1}}{df_n}$$

Desired Gradients

$$\boxed{\frac{dL}{dc_3}} = \frac{dL}{df_3}\frac{df_3}{dc_3} = L_3 \frac{df_3}{dc_3}$$

$$\frac{dL}{dc_2} = \frac{dL}{df_3}\frac{df_3}{df_2}\frac{df_2}{dc_2} = L_2 \frac{df_2}{dc_2}$$

$$\dots$$

$$\frac{dL}{dc_n} = L_n \frac{df_n}{dc_n}$$

dkfz.

Toy Example

Nested scalar functions $f(c, x)$ with unknown coefficients c

last layer output
2nd layer output
1st layer output
input
training target

Loss Function

$$L(c_3, c_2, c_1, x) = \left(f_3(c_3, f_2(c_2, f_1(c_1, x))) - y\right)^2$$

Intermediate Values

$$L_3 = \frac{dL}{df_3} = 2(f_3 - y)$$

$$L_2 = \frac{dL}{df_2} = \frac{dL}{df_3}\frac{df_3}{df_2} = L_3 \frac{df_3}{df_2}$$

$$\cdots$$

$$L_n = \frac{dL}{df_n} = L_{n+1}\frac{df_{n+1}}{df_n}$$

Backpropagation

Desired Gradients

$$\boxed{\frac{dL}{dc_3}} = \frac{dL}{df_3}\frac{df_3}{dc_3} = L_3 \frac{df_3}{dc_3}$$

$$\boxed{\frac{dL}{dc_2}} = \frac{dL}{df_3}\frac{df_3}{df_2}\frac{df_2}{dc_2} = L_2 \frac{df_2}{dc_2}$$

$$\cdots$$

$$\frac{dL}{dc_n} = L_n \frac{df_n}{dc_n}$$

Toy Example 2

Nested vector-valued functions ***f***(***c***, ***x***) with unknown coefficients vectors ***c***

Loss Function

$$L(\boldsymbol{c}_3,\boldsymbol{c}_2,\boldsymbol{c}_1,\boldsymbol{x}) = \Big(\underset{n}{\overset{\text{last layer output}}{\boldsymbol{f}_3}}(\underset{\nu}{\boldsymbol{c}_3},\underset{m}{\overset{\text{2nd layer output}}{\boldsymbol{f}_2}}(\underset{\mu}{\boldsymbol{c}_2},\underset{l}{\overset{\text{1st layer output}}{\boldsymbol{f}_1}}(\underset{\lambda}{\boldsymbol{c}_1},\overset{\text{input}}{\boldsymbol{x}}))) - \overset{\text{training target}}{\boldsymbol{y}}\Big)^2$$

Intermediate Values

$$L_{3n} = \frac{dL}{df_{3n}} = 2(\boldsymbol{f}_3 - \boldsymbol{y})_n$$

$$L_{2m} = \frac{dL}{df_{2m}} = \sum_n \frac{dL}{df_{3n}}\frac{df_{3n}}{df_{2m}} = \sum_n L_{3n}\frac{df_{3n}}{df_{2m}}$$

$$L_{1l} = \frac{dL}{df_{1l}} = \ldots = \sum_m L_{2m}\frac{df_{2m}}{df_{1l}}$$

Desired Gradients

$$\frac{dL}{dc_{3\nu}} = \sum_n \frac{dL}{df_{3n}}\frac{df_{3n}}{dc_{3\nu}} = \sum_n L_{3n}\frac{df_{3n}}{dc_{3\nu}} = 0$$

$$\frac{dL}{dc_{2\mu}} = \sum_{nm} \frac{dL}{df_{3n}}\frac{df_{3n}}{df_{2m}}\frac{df_{2m}}{dc_{2\mu}} = \sum_m L_{2m}\frac{df_{2m}}{dc_{2\mu}} = 0$$

$$\frac{dL}{dc_{1\lambda}} = \ldots = \sum_l L_{1l}\frac{df_{1l}}{dc_{1\lambda}}$$

Toy Example 3

Backprojection of convolved CNN-preprocessed rawdata with unknown $\boldsymbol{c}_1$

Loss Function

$$L(\boldsymbol{c}_3,\boldsymbol{c}_2,\boldsymbol{c}_1,\boldsymbol{x}) = \Big(\underbrace{\boldsymbol{f}_3}_{n}\big(\underbrace{\boldsymbol{c}_3}_{\nu},\underbrace{\boldsymbol{f}_2}_{m}(\underbrace{\boldsymbol{c}_2}_{\mu},\underbrace{\boldsymbol{f}_1}_{l}(\underbrace{\boldsymbol{c}_1}_{\lambda},\boldsymbol{x}))\big) - \boldsymbol{y}\Big)^2$$

backprojection, convolution, CNN, rawdata, target image

Intermediate Values

$$L_{3n} = \frac{dL}{df_{3n}}$$ = image = 2 · reconstructed minus target image

$$L_{2m} = \frac{dL}{df_{2m}} = \sum_n \frac{dL}{df_{3n}}\frac{df_{3n}}{df_{2m}} = \sum_n L_{3n}\frac{df_{3n}}{df_{2m}}$$ = forward projection of image L_3

backprojection derived wrt the *m*-th sinogram value, i.e. image where pixel *n* holds the intersection of ray *m* with pixel *n*

$$L_{1l} = \frac{dL}{df_{1l}} = \ldots = \sum_m L_{2m}\frac{df_{2m}}{df_{1l}}$$ = convolution of L_2 with RamLak kernel

convolution derived wrt the *l*-th sinogram value, i.e. sinogram where pixel *m* holds the convolution kernel entry *m-l*

Desired Gradients

$$\frac{dL}{dc_{3\nu}} = \sum_n \frac{dL}{df_{3n}}\frac{df_{3n}}{dc_{3\nu}} = \sum_n L_{3n}\frac{df_{3n}}{dc_{3\nu}} = 0$$

$$\frac{dL}{dc_{2\mu}} = \sum_{nm} \frac{dL}{df_{3n}}\frac{df_{3n}}{df_{2m}}\frac{df_{2m}}{dc_{2\mu}} = \sum_m L_{2m}\frac{df_{2m}}{dc_{2\mu}} = 0$$

$$\frac{dL}{dc_{1\lambda}} = \ldots = \sum_l L_{1l}\frac{df_{1l}}{dc_{1\lambda}}$$

dkfz.

Gradient Descent

$B = 1$: stochastic gradient descent
$B = N$: gradient descent
else : batch gradient descent

For each epoch:

Shuffle the *N* data samples

For each batch (batch size *B*):

For each data sample x_b of the current batch:

Calculate the loss function's gradient

$$\frac{dL}{d\boldsymbol{c}} = \boldsymbol{\nabla}_{\boldsymbol{c}} L(\boldsymbol{c}, \boldsymbol{x}_b)$$

wrt the unknowns *c* by backpropagation.

Now, update the network parameters (weights, biases, etc.):

$$\boldsymbol{c}^{\text{new}} = \boldsymbol{c}^{\text{old}} - \frac{\eta}{B} \sum_{b=1}^{B} \boldsymbol{\nabla}_{\boldsymbol{c}} L(\boldsymbol{c}, \boldsymbol{x}_b)$$

Optimization Algorithms

SGD with Momentum: Add fraction of past update vector to current update vector

$$v' = \gamma v + \eta \cdot \nabla_\theta C(\theta)$$
$$\theta' = \theta - v'$$

Nesterov Accelerated Gradient (NAG): Compute gradients with respect to the approximate future position of parameters

$$v' = \gamma v + \eta \cdot \nabla_\theta C(\theta - \gamma v)$$
$$\theta' = \theta - v'$$

Adagrad: Adapt updates to each individual parameter

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

$$g_{t,i} = \nabla_\theta C(\theta_{t,i})$$

$$G_t = \begin{pmatrix} \sum_{t'<t} g_{t',1}^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sum_{t'<t} g_{t',N}^2 \end{pmatrix}$$

Optimization Algorithms

Adadelta: Improve on Adagrad by restricting the window of past time steps to some fixed window → No vanishing gradients

Additionally, perform normalization of gradients → No learning rate needed

RMSprop: Identical to Adadelta, without normalization. Adapt learning rate with exponentially decaying average of past squared gradients.

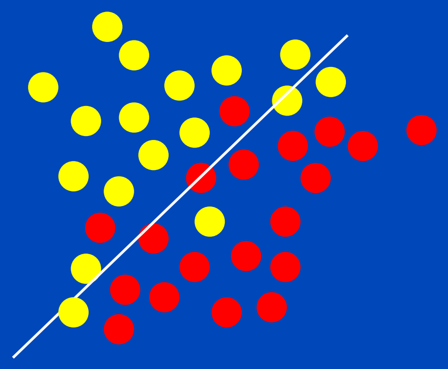
Adaptive Moment Estimation (Adam): Momentum for first, and second order moment (mean and variance) of gradients

$$m_t^{(1)} = \beta_1 m_{t-1}^{(1)} + (1-\beta_1) g_t$$
$$m_t^{(2)} = \beta_2 m_{t-1}^{(2)} + (1-\beta_2) g_t$$

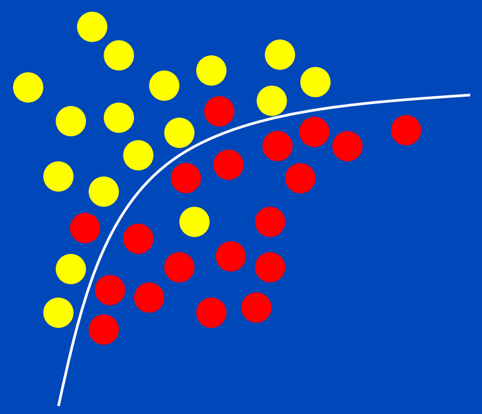
$$\hat{m}_t^{(1)} = \frac{m_t^{(1)}}{1-\beta_1} \qquad \hat{m}_t^{(2)} = \frac{m_t^{(2)}}{1-\beta_2}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{m}_t^{(2)} + \epsilon}} \odot \hat{m}_t^{(1)}$$

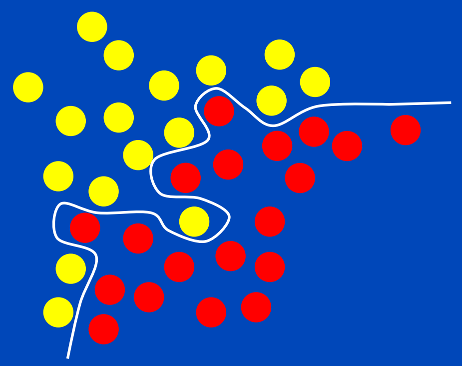
Underfitting



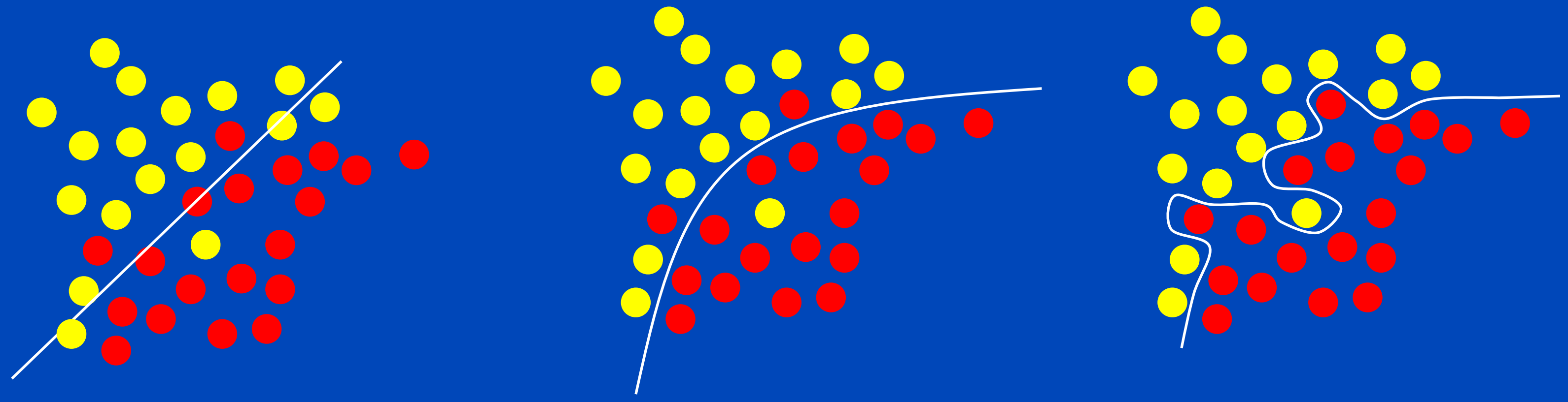
Fitting



Overfitting



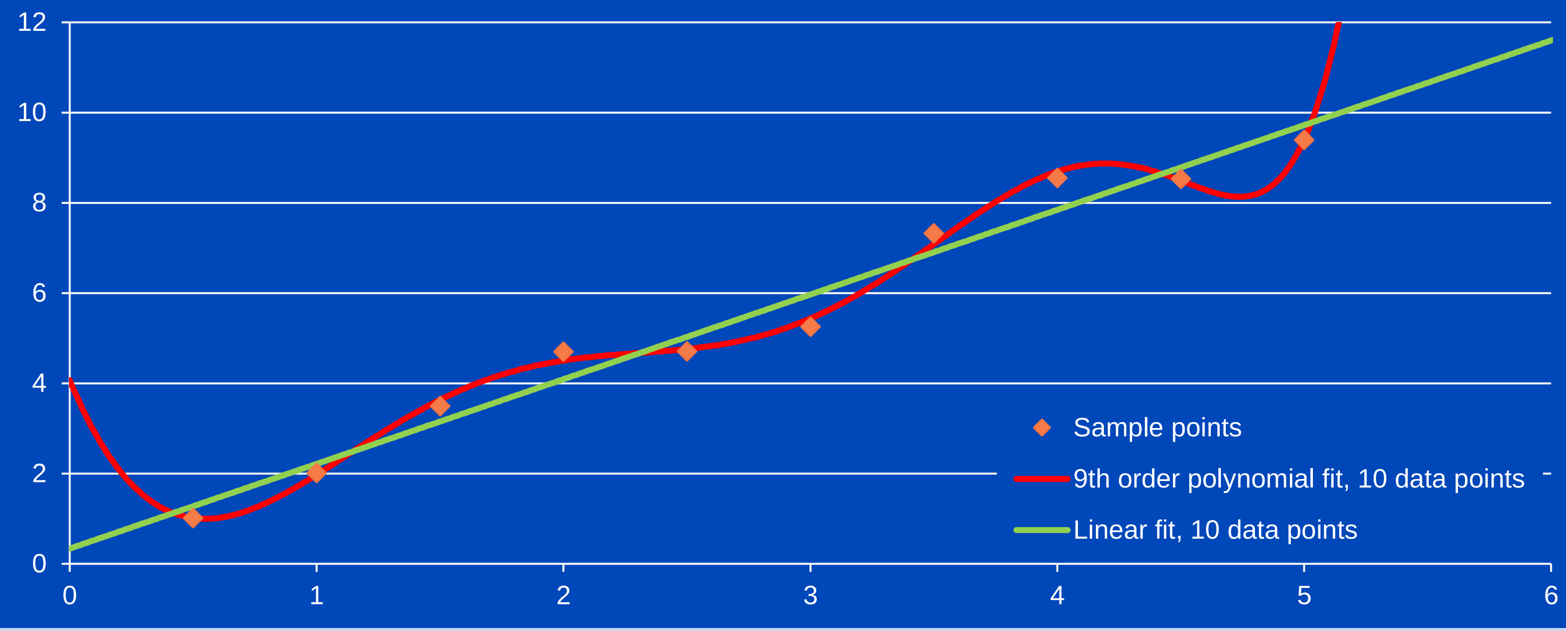
Fitting



underfit reasonable overfit

Overfitting

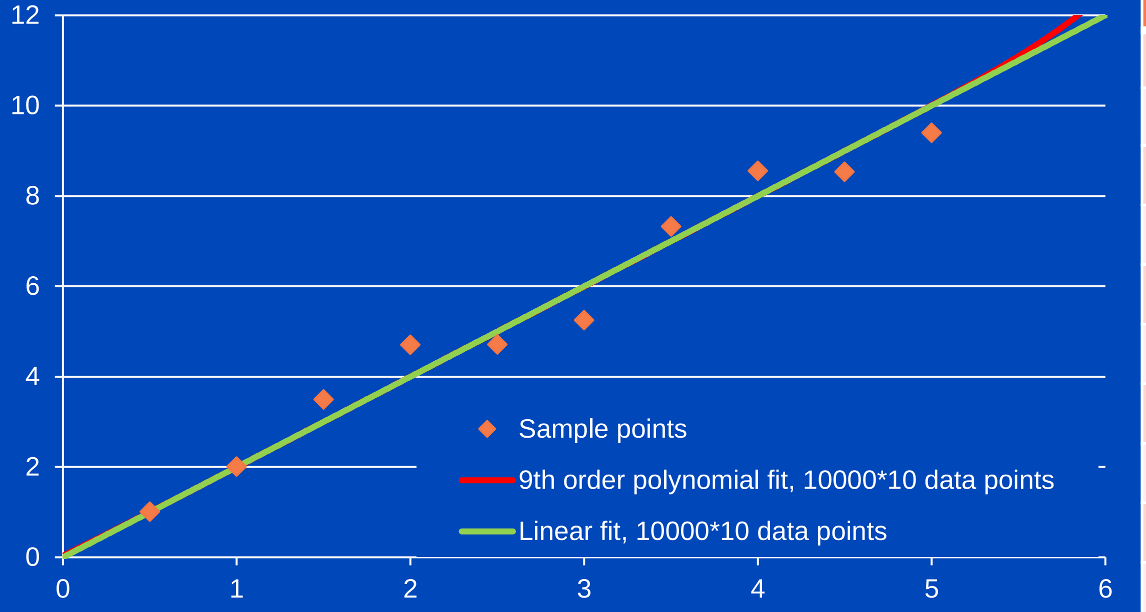
- **Assume our training data result from sampling the function f(x) = 2x at a given number of points.**
- **Since the sampling might include some random noise, the samples slightly deviate from the function f(x) = 2x.**
- **A 9th order polynomial perfectly fits the training data, but fails to appropriately predict test data such as x = 0.25 for instance.**



Regularization

Increase of training data

- The increase of the amount of training data makes the network more robust against single deviations.
- The training data can also be increased artificially.
- Similar results can be observed if the polynomial is fitted to 100000 samples.



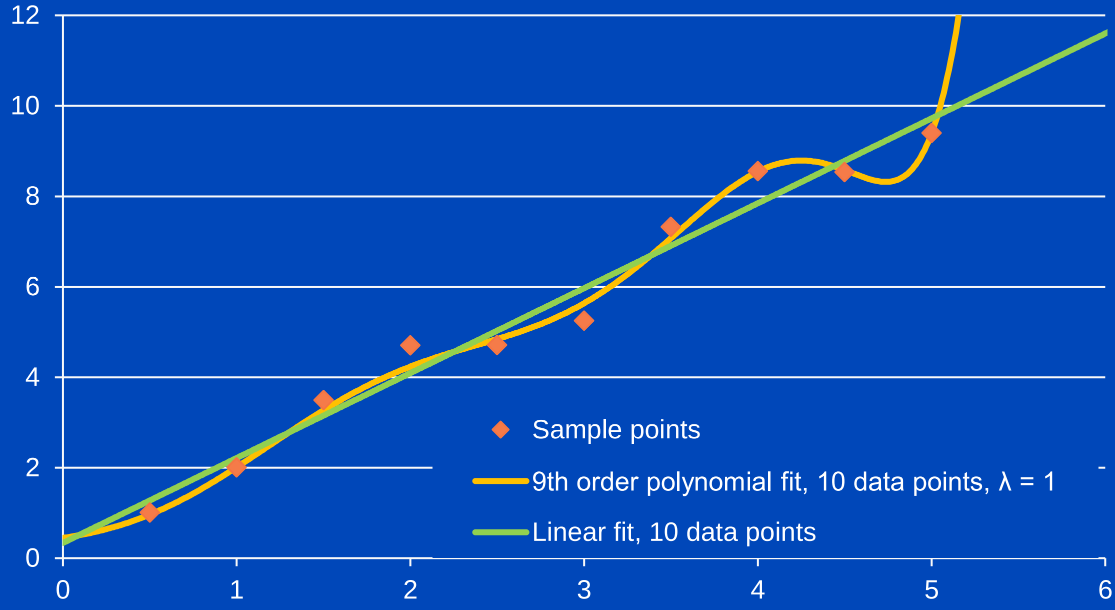
Coefficients	Linear	9th order
c_0	-0.00295	0.03343
c_1	2.000325	1.904762
c_2		0.079125
c_3		-0.02262
c_4		0.000435
c_5		-4.96E-05
c_6		0.000339
c_7		-4.25E-05
c_8		-9.19E-06
c_9		1.43E-06

dkfz.

Regularization

Penalizing large weights

- Modification of the cost function to penalize large weight (i.e. quadratic penalty): $C = C_0 + \lambda \sum_w w^2$
- If a certain weight is large, the output strongly depends on the input of that weight.



Coefficients	Linear	9th order
c_0	-0.00295	0.447558
c_1	2.000325	0.575279
c_2		0.665781
c_3		0.562606
c_4		0.049884
c_5		-0.45894
c_6		0.186099
c_7		-0.01496
c_8		-0.00342
c_9		0.000471

Regularization: Dropout

- Dropout is intended to reduce overfitting.
- Dropout randomly (before each mini bach) zeroes activations with probability 1-*p* during training.
- Let *D* be a Bernoulli random variate with *P(D = 1) = p* and regard the following toy example:

x_1, x_2

$$y_1 = Df(w_{11}x_1 + w_{12}x_2 + b_1)$$
$$y_2 = Df(w_{21}x_1 + w_{22}x_2 + b_2)$$
$$y_3 = Df(w_{31}x_1 + w_{32}x_2 + b_3)$$

$$z = f(w_1y_1 + w_2y_2 + w_3y_3 + b)$$

- After training, on inference, the outputs y_1, y_2, and y_3 need to be multiplied by *p* to be equivalent to the training situation, on expectation (alternatively divide by *p* during training).
- Dropout can be realized using dropout layers.

Batch Normalization

Batch normalization

- normalizes each activation to have zero expectation and unit variance within the batch
- introduces trainable scale and offset for each activation (or for each feature map) to, potentially, denormalize again
- is part of the model architecture
- reduces the need for dropout
- reduces internal covariate shift and thus accelerates training
- fixes the means and variances of layer inputs
- improves gradient flow through the network
- allows for higher learning rates without the risk of divergence
- prevents the net from getting trapped in saturated modes
- makes it possible to use saturating nonlinearities

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

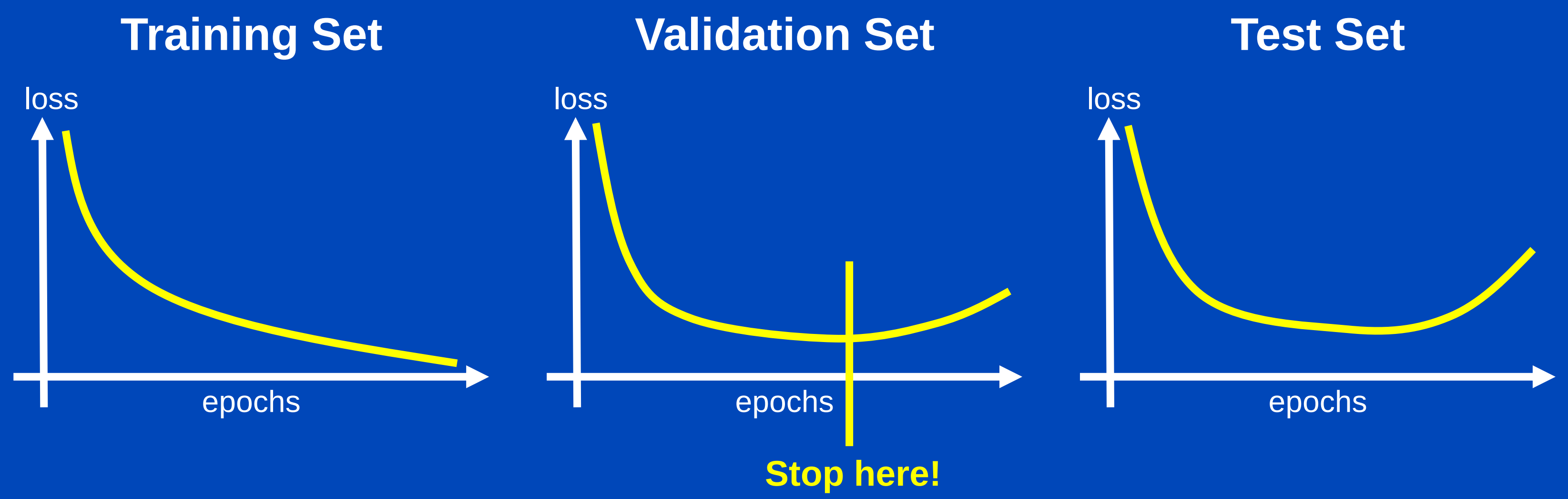
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015.

Further Means to Avoid Overfitting

- Choose adequate network architecture
- Preprocess data
  - Normalize data (mean, var, …)
  - Add prior knowledge (e.g. exp(-x))
- Data augmentation
  - Random transformations (mirror, affine, deformable, …)
  - Gray value distribution
  - Change spatial resolution
  - Add noise
  - …
- Penalize loss function
  - Enforce small weights
  - Enforce sparse weights
  - …

Learning Curve



- Training and validation set are part of the training
- Do not use test set for training
- Early stopping (at minimum validation loss)
- Training : Validation : Test ≈ 70 : 20 : 10

Weight Initialization

- Weights in neural networks should be initialized such that the neurons are not saturated (since saturation often decreases the learning rate).
- Assume we have a fully-connected network with 1000 input neurons.
- Let us further assume that half of the input equals 1 and the other half equals 0.
- If the weights and the bias are initialized with Gaussian random numbers with zero mean and a standard deviation of 1, the weighted sum $z = \sum w_j x_j + b$ to the first hidden neuron is zero mean Gaussian with standard deviation $\sigma = \sqrt{501} \approx 22.4$.
- Thus, it is very likely that $z \ll 0$ or $z \gg 0$ and the neuron saturates.
- Therefore, if we have n_{in} inputs, an initialization with Gaussian random numbers with zero mean and a standard deviation of $1/\sqrt{n_{\text{in}}}$ would be a better choice.

What was not Discussed Here

- Attention mechanism
- Transformer networks
- Diffusion networks
- …

Thank You!

This presentation will soon be available at www.dkfz.de/ct.

Job opportunities through DKFZ’s international PhD or Postdoctoral Fellowship programs (marc.kachelriess@dkfz.de).

Parts of the reconstruction software were provided by RayConStruct® GmbH, Nürnberg, Germany.

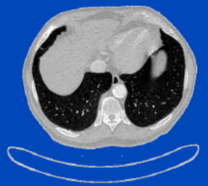
Denoising benchmark with surprising results

PART 10:
IS NEWER ALWAYS BETTER?

Low Dose CT Image Denoising

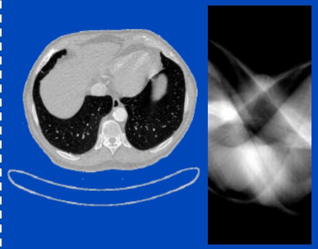
Domain

Image **(I)**



Sinogram **(S)**

Dual **(D)**



Algorithms

Classical

- **Non-local means [1]** (I/S/D)
- **BM3D [2]** (I/S)
- **Total variation [3]** (I/S/D)
- …

Deep Learning

Supervised

- **RED-CNN [4]** (I)
- **WGAN-VGG [5]** (I)
- **Sinogram Inner-Structure Transformer [6]** (D)
- **DP-ResNet [7]** (D)
- …

Unsupervised / Self-supervised

- **Half2Half [8]** (I/S)
- **Adjacent views [9]** (S)
- **Inter-slice congruence [10]** (I)
- **Deep Image Prior [11]** (I)
- …

[1] Yuan Y, Zhang YB, Yu HY (2018) "Adaptive nonlocal means method for denoising basis material images […]". J Comput Assist Tomogr 42:972–981.
[2] Feruglio, P Fumene, C Vinegoni, J Gros, A Sbarbati, and R Weissleder (2010) "Block Matching 3D Random Noise Filtering for Absorption Optical Projection Tomography." *Physics in Medicine and Biology* 55 (18): 5401–15.
[3] Jia L, Zhang Q, Shang Y, et al. (2018) "Denoising for low-dose CT image by discriminative weighted nuclear norm minimization". IEEE Access
[4] Chen, Hu, Yi Zhang, Mannudeep K. Kalra, Feng Lin, Yang Chen, Peixi Liao, Jiliu Zhou, and Ge Wang. 2017. "Low-Dose CT with a Residual Encoder-Decoder Convolutional Neural Network." *IEEE Transactions on Medical Imaging* 36 (12): 2524–35.
[5] Yang, Qingsong, Pingkun Yan, Yanbo Zhang, et al.. 2018. "Low-Dose CT Image Denoising Using a Generative Adversarial Network with Wasserstein Distance and Perceptual Loss." *IEEE Transactions on Medical Imaging* 37 (6): 1348–57.
[6] L. Yang, Z. Li, R. Ge, J. Zhao, H. Si and D. Zhang, "Low-Dose CT Denoising via Sinogram Inner-Structure Transformer," in IEEE Transactions on Medical Imaging, vol. 42, no. 4, pp. 910-921, 2023.
[7] Yin, Xiangrui, Qianlong Zhao, Jin Liu, Wei Yang, Jian Yang, Guotao Quan, Yang Chen, Huazhong Shu, Limin Luo, and Jean-Louis Coatrieux. 2019. "Domain Progressive 3D Residual Convolution Network to Improve Low-Dose CT Imaging." *IEEE TMI* 38 (12).
[8] Yuan, N., Zhou, J., & Qi, J. (2020). Half2Half: deep neural network based CT image denoising without independent reference data. *Physics in Medicine & Biology*, *65*(21), 215020.
[9] Hong, Zixuan, Dong Zeng, Xi Tao, and Jianhua Ma. 2023. "Learning CT Projection Denoising from Adjacent Views." *Medical Physics* 50 (3): 1367–77.
[10] Bera, Sutanu, and Prabir Kumar Biswas. 2023. "Self Supervised Low Dose Computed Tomography Image Denoising Using Invertible Network Exploiting Inter Slice Congruence." In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 5603–12.
[11] Baguer, Daniel Otero, Johannes Leuschner, and Maximilian Schmidt. 2020. "Computed Tomography Reconstruction Using Deep Image Prior and Learned Reconstruction Methods." *Inverse Problems* 36 (9.

Low Dose CT Image Denoising

Domain

Image **(I)**

Sinogram **(S)**

Dual **(D)**

Algorithms

Classical

Non-local means [1]
(I/S/D)

BM3D [2]
(I/S)

Total variation [3]
(I/S/D)

…

Deep Learning

Supervised

RED-CNN [4]
(I)

WGAN-VGG [5]
(I)

Sinogram Inner-Structure Transformer [6]
(D)

DP-ResNet [7]
(D)

…

Unsupervised / Self-supervised

Half2Half [8]
(I/S)

Adjacent views [9]
(S)

Inter-slice congruence [10]
(I)

Deep Image Prior [11]
(I)

…

[1] Yuan Y, Zhang YB, Yu HY (2018) “Adaptive nonlocal means method for denoising basis material images […]”. J Comput Assist Tomogr 42:972–981.
[2] Feruglio, P Fumene, C Vinegoni, J Gros, A Sbarbati, and R Weissleder (2010) “Block Matching 3D Random Noise Filtering for Absorption Optical Projection Tomography.” *Physics in Medicine and Biology* 55 (18): 5401–15.
[3] Jia L, Zhang Q, Shang Y, et al. (2018) “Denoising for low-dose CT image by discriminative weighted nuclear norm minimization”. IEEE Access
[4] Chen, Hu, Yi Zhang, Mannudeep K. Kalra, Feng Lin, Yang Chen, Peixi Liao, Jiliu Zhou, and Ge Wang. 2017. “Low-Dose CT with a Residual Encoder-Decoder Convolutional Neural Network.” *IEEE Transactions on Medical Imaging* 36 (12): 2524–35.
[5] Yang, Qingsong, Pingkun Yan, Yanbo Zhang, et al.. 2018. “Low-Dose CT Image Denoising Using a Generative Adversarial Network with Wasserstein Distance and Perceptual Loss.” *IEEE Transactions on Medical Imaging* 37 (6): 1348–57.
[6] L. Yang, Z. Li, R. Ge, J. Zhao, H. Si and D. Zhang, "Low-Dose CT Denoising via Sinogram Inner-Structure Transformer," in IEEE Transactions on Medical Imaging, vol. 42, no. 4, pp. 910-921, 2023.
[7] Yin, Xiangrui, Qianlong Zhao, Jin Liu, Wei Yang, Jian Yang, Guotao Quan, Yang Chen, Huazhong Shu, Limin Luo, and Jean-Louis Coatrieux. 2019. “Domain Progressive 3D Residual Convolution Network to Improve Low-Dose CT Imaging.” *IEEE TMI* 38 (12).
[8] Yuan, N., Zhou, J., & Qi, J. (2020). Half2Half: deep neural network based CT image denoising without independent reference data. *Physics in Medicine & Biology*, *65*(21), 215020.
[9] Hong, Zixuan, Dong Zeng, Xi Tao, and Jianhua Ma. 2023. “Learning CT Projection Denoising from Adjacent Views.” *Medical Physics* 50 (3): 1367–77.
[10] Bera, Sutanu, and Prabir Kumar Biswas. 2023. “Self Supervised Low Dose Computed Tomography Image Denoising Using Invertible Network Exploiting Inter Slice Congruence.” In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 5603–12.
[11] Baguer, Daniel Otero, Johannes Leuschner, and Maximilian Schmidt. 2020. “Computed Tomography Reconstruction Using Deep Image Prior and Learned Reconstruction Methods.” *Inverse Problems* 36 (9.

Low Dose CT Image Denoising

Algorithms

Classical

- **Non-local means [1]** (I/S/D)
- **BM3D [2]** (I/S)
- **Total variation [3]** (I/S/D)
- …

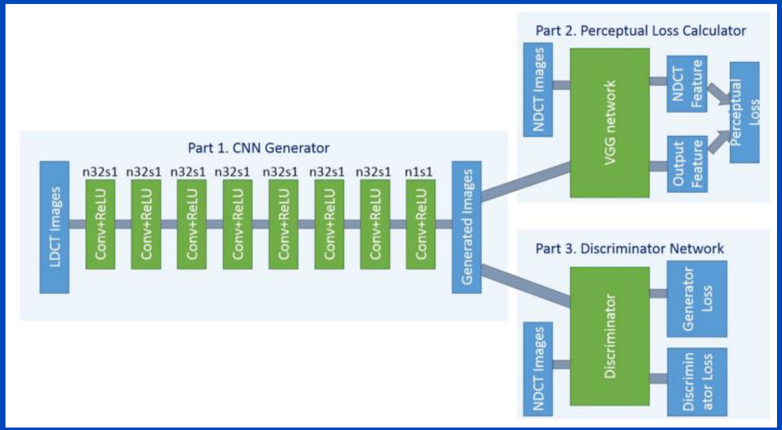
Deep Learning

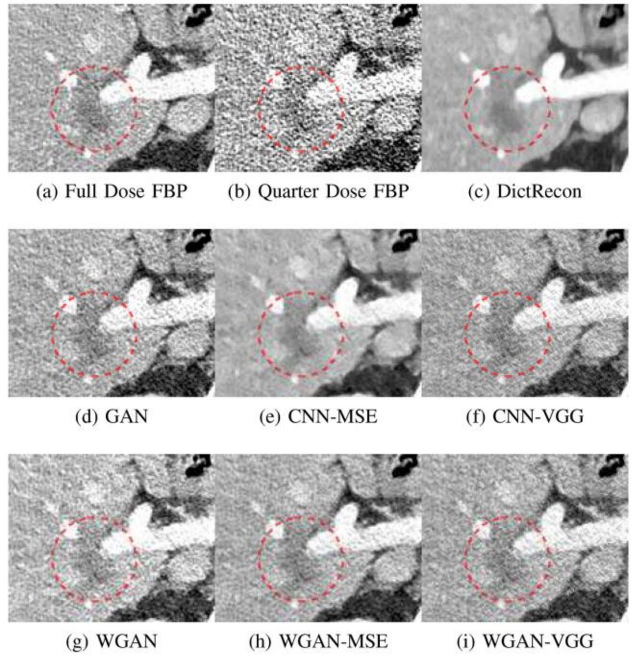
Supervised

- **RED-CNN [4]** (I)
- **WGAN-VGG [5]** (I)
- **Sinogram Inner-Structure Transformer [6]** (D)
- **DP-ResNet [7]** (D)
- …

Unsupervised / Self-supervised

- **Half2Half [8]** (I/S)
- **Adjacent views [9]** (S)
- **Inter-slice congruence [10]** (I)
- **Deep Image Prior [11]** (I)
- …





Low Dose CT Image Denoising

Algorithms

Deep Learning

Classical

Non-local means [1]
(I/S/D)

BM3D [2]
(I/S)

Total variation [3]
(I/S/D)

…

Supervised

RED-CNN [4]
(I)

WGAN-VGG [5]
(I)

Sinogram Inner-Structure Transformer [6]
(D)

DP-ResNet [7]
(D)

…

Unsupervised / Self-supervised

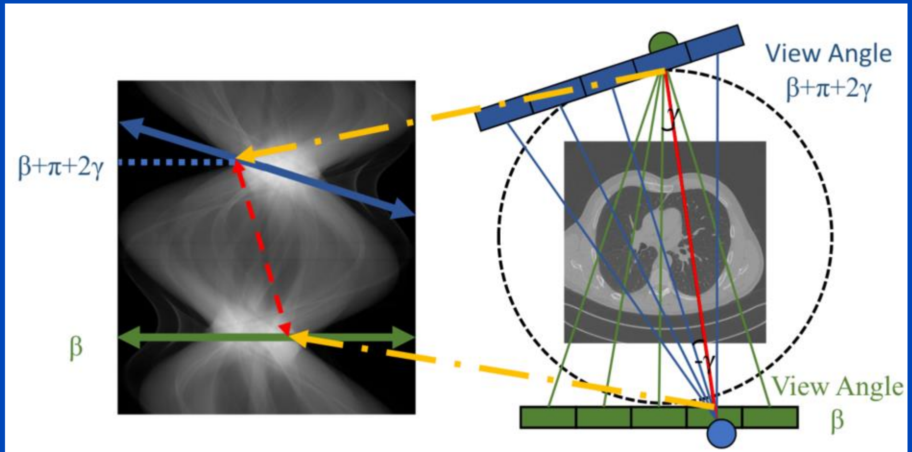
Half2Half [8]
(I/S)

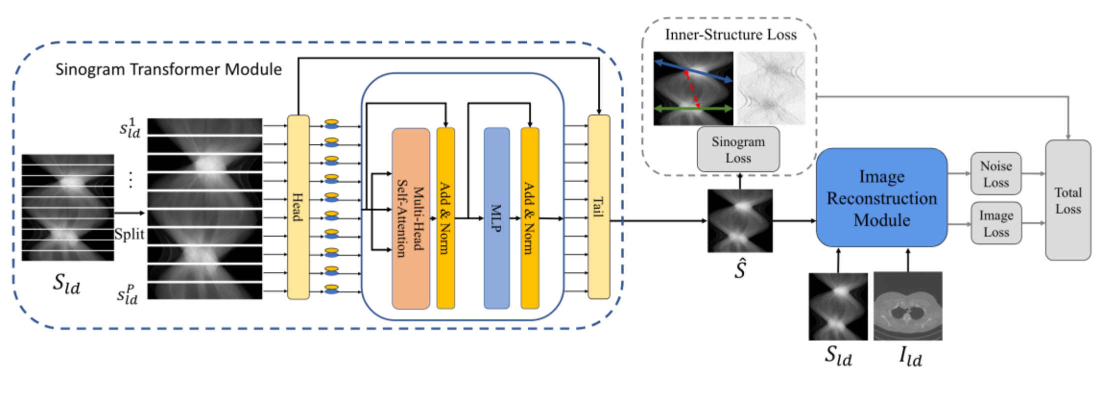
Adjacent views [9]
(S)

Inter-slice congruence [10]
(I)

Deep Image Prior [11]
(I)

…





Low Dose CT Image Denoising

Algorithms

Deep Learning

Classical

Non-local means [1]
(I/S/D)

BM3D [2]
(I/S)

Total variation [3]
(I/S/D)

…

Supervised

RED-CNN [4]
(I)

WGAN-VGG [5]
(I)

Sinogram Inner-Structure Transformer [6]
(D)

DP-ResNet [7]
(D)

…

Unsupervised / Self-supervised

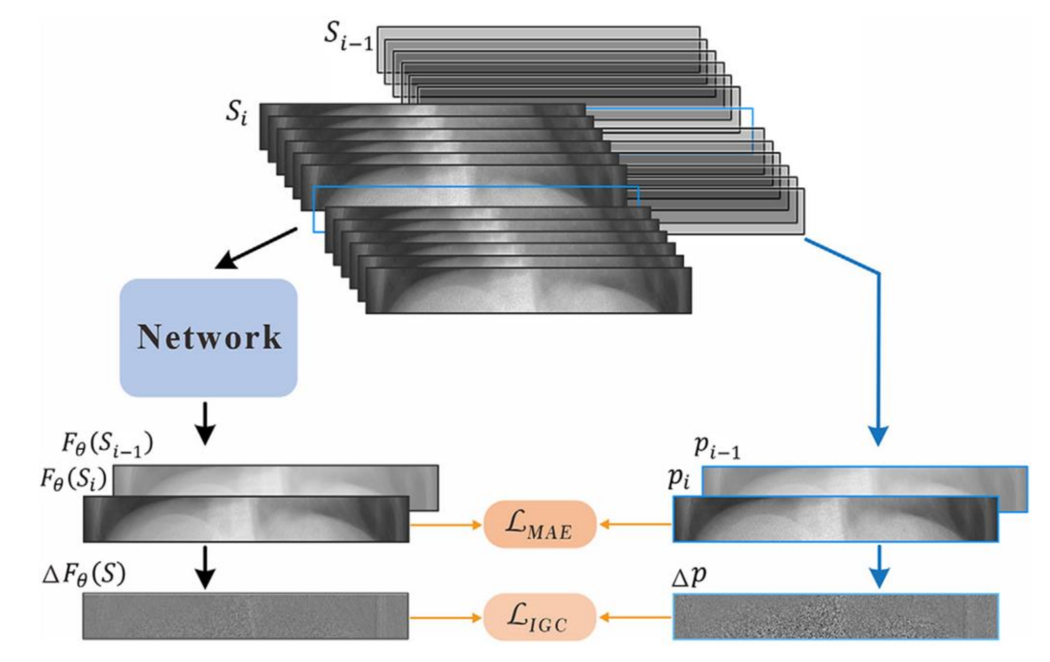
Half2Half [8]
(I/S)

Adjacent views [9]
(S)

Inter-slice congruence [10]
(I)

Deep Image Prior [11]
(I)

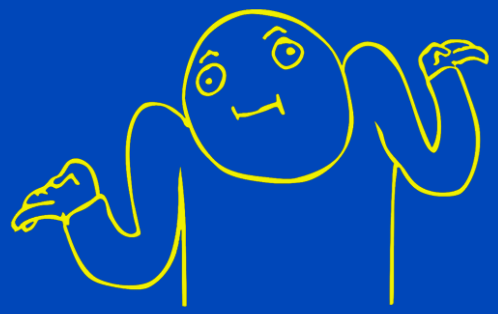
…



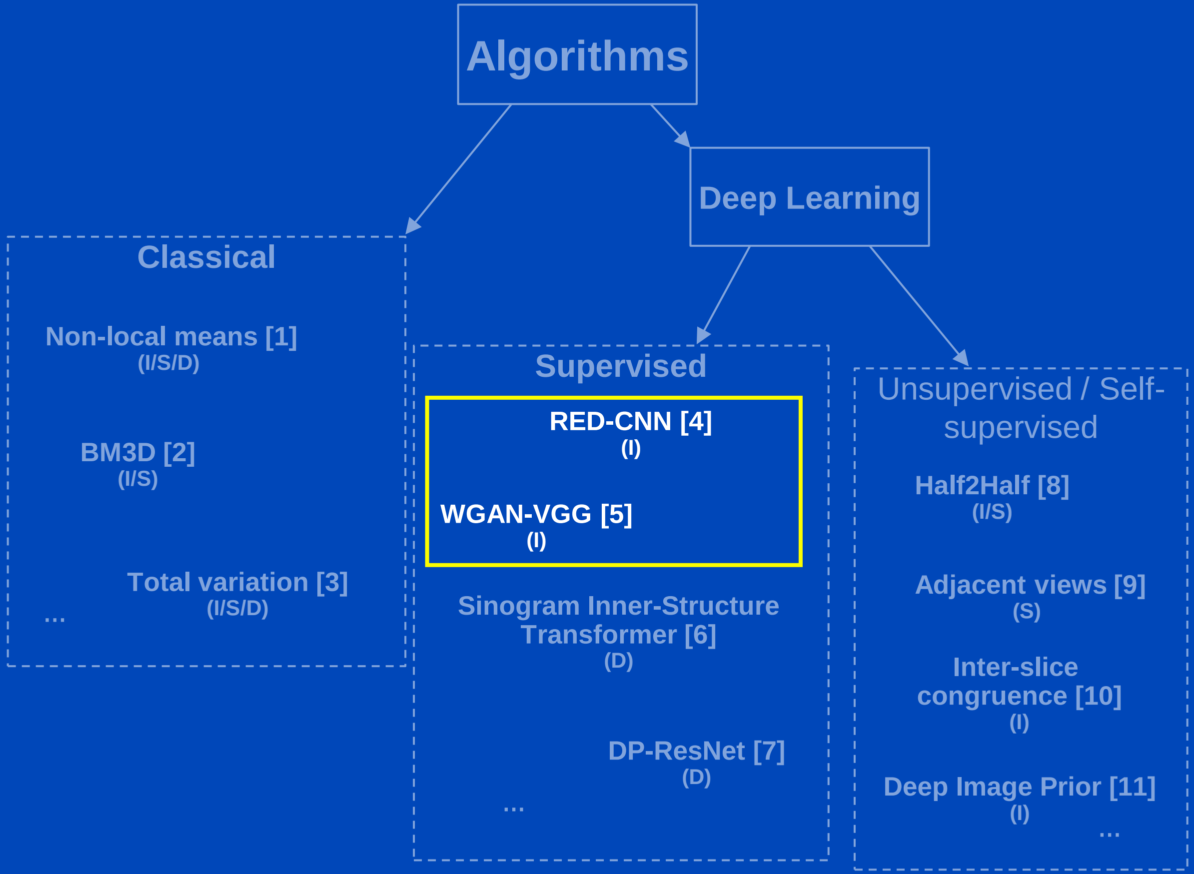
Low Dose CT Image Denoising

Q: Which algorithm performs best?

A:



Low Dose CT Image Denoising



$$\theta^* = \arg\min_{\theta} \mathbb{E}_{x,y\sim\mathcal{D}^{\text{train}}} ||f_\theta(x) - y||$$

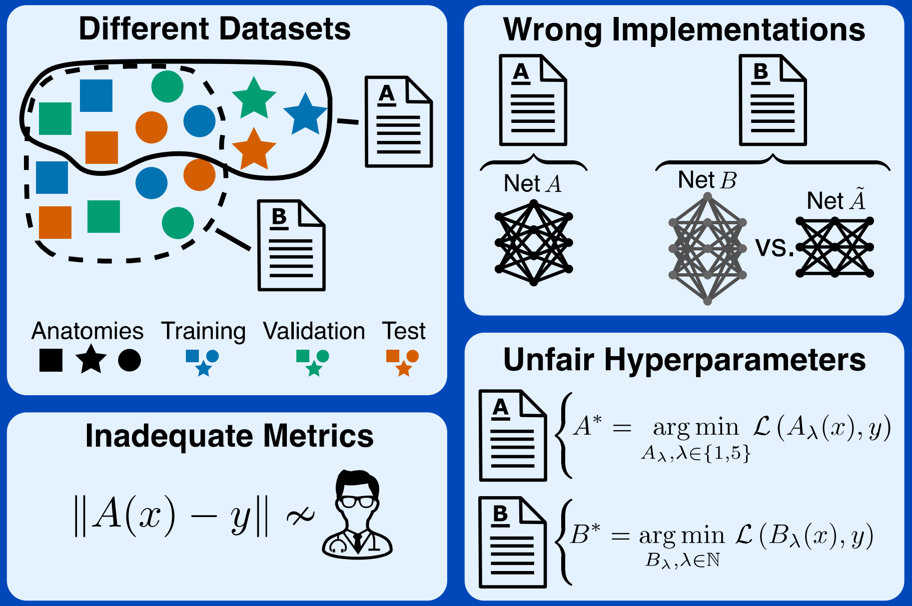
Flaws of Current Evaluation Protocols

No open-source implementations: Increases chance of (un)intentional errors

> *Meanwhile, to keep the reasonable model complexity, we reduced 96 filters to 32 filters in each layer.* [1]

Inadequate metrics: Standard IQA metrics (MSE, SSIM, …) do not correlate well with human reader ratings [2]

Unfair hyperparameters: Either no HP optimization or limited to subset of parameters / methods. Often authors use HPs reported in reference publications
→ Problematic since no consensus dataset exists



[1] Fan, Fenglei, Hongming Shan, Mannudeep K. Kalra, Ramandeep Singh, Guhan Qian, Matthew Getzin, Yueyang Teng, Juergen Hahn, and Ge Wang. 2020. “Quadratic Autoencoder (Q-AE) for Low-Dose CT Denoising.” *IEEE Transactions on Medical Imaging.*
[2] K. Ohashi, Y. Nagatani, M. Yoshigoe, K. Iwai, K. Tsuchiya, A. Hino, Y. Kida, A. Yamazaki, and T. Ishida, “Applicability evaluation of fullreference image quality assessment methods for computed tomography images,” Journal of Imaging Informatics in Medicine, vol. 36, no. 6, pp. 2623–2634, Dec. 2023.

Benchmark Setup

Dataset

Open-source *LDCT and Projection Dataset* [1] with 150 CT scans of abdomen, head, and chest at routine dose levels. Low dose images were simulated at 25% dose for abdomen/head and 10% dose for chest

All tested methods use the same train/validation set and were evaluated on the same test set

[1] McCollough, Cynthia, Baiyu Chen, David R Holmes III, Xinhui Duan, Zhicong Yu, Lifeng Yu, Shuai Leng, and Joel Fletcher. 2020. “Low Dose CT Image and Projection Data.” The Cancer Imaging Archive. https://doi.org/10.7937/9NPB-2637.

Benchmark Setup

LDCT Denoising Algorithms

- CNN-10 (2017)
- RED-CNN (2017)
- ResNet (2018)
- WGAN-VGG (2017)
- QAE (2019)
- DU-GAN (2021)
- TransCT (2021)
- Bilateral (2022)

Standard CNNs trained with pixelwise losses

CNNs trained with adversarial losses

Specialized architectures trained with pixelwise losses

Benchmark Setup

- Hyperparameter Optimization
  - Rigorous HP optimization including the weighting factors of loss function terms
  - 50 iterations of sequential model-based optimization (SMBO) using Gaussian processes and expected improvement as acquisition function
  - As metric to optimize we use SSIM on validation dataset
- Retrain each method 10 times with
  - optimal HPs and
  - different random seeds

	Parameter	Prior
All algorithms	Learning rate	$\log\mathcal{U}(1\times10^{-5}, 0.01)$
	Maximum iterations	$\mathcal{U}(1\times10^{3}, 1\times10^{5})$
	Mini-batch size	$\mathcal{U}(2, 128)$
CNN-10 (2017)	Patchsize	$\mathcal{U}(32, 128)$
RED-CNN (2017)	Patchsize	$\mathcal{U}(32, 128)$
WGAN-VGG (2017)	β_1 of Adam	$\mathcal{U}(0.3, 0.9)$
	Loss weight: $\lambda_{\text{perceptual}}$	$\mathcal{U}(0, 1)$
	Critic updates	$\mathcal{U}(1, 5)$
	Patchsize	$\mathcal{U}(32, 128)$
ResNet (2018)	Patchsize	$\mathcal{U}(32, 128)$
QAE (2019)	Patchsize	$\mathcal{U}(32, 128)$
DU-GAN (2021)	β_1 of Adam	$\mathcal{U}(0.3, 0.9)$
	Cutmix warmup	$\mathcal{U}(0, 1\times10^{4})$
	Loss weight: λ_{adv}	$\mathcal{U}(0, 1)$
	Loss weight: λ_{CM}	$\mathcal{U}(0, 10)$
	Loss weight: $\lambda_{\text{px,grad}}$	$\mathcal{U}(0, 40)$
	Critic updates	$\mathcal{U}(1, 5)$
	Patchsize	$\mathcal{U}(32, 128)$
TransCT (2021)	-	-
Bilateral (2022)	Learning rate for σ_r	$\log\mathcal{U}(1\times10^{-5}, 0.01)$
	Patchsize	$\mathcal{U}(32, 128)$
	Initalization for σ_r	$\mathcal{U}(0, 1)$
	Initalization for $\sigma_{x,y}$	$\mathcal{U}(0, 1)$

$\mathcal{U}$: Uniform distribution; $\log\mathcal{U}$: Log-uniform distribution

Benchmark Setup

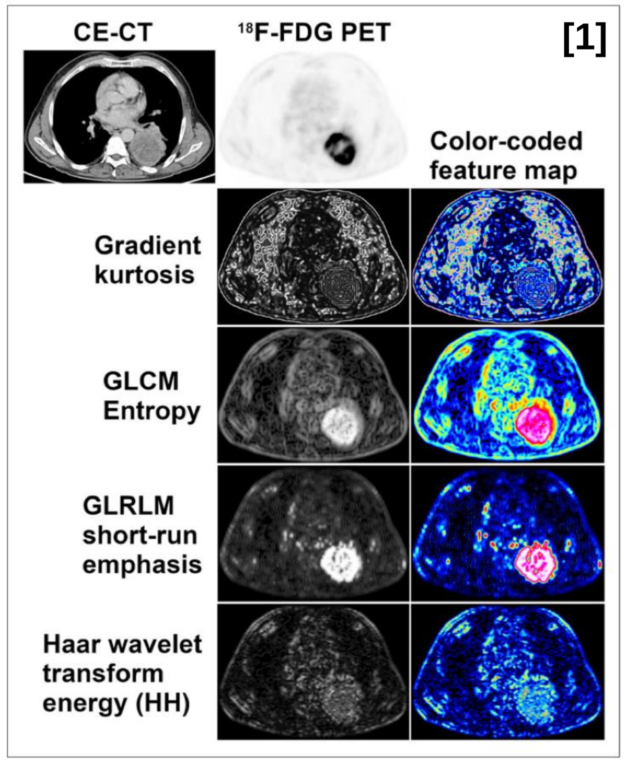
Metrics

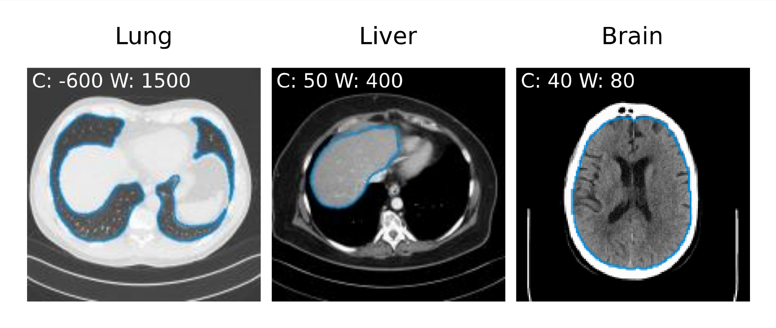
- **Standard metrics: SSIM, PSNR, Visual information fidelity (VIF)**
- **Clinically relevant image properties: Radiomic feature similarity (RFS)**
  1. **Automatically segment organs in high-dose scan (s)**
  2. **Compare features on high-dose scan with those on denoised scans**

$$\mathrm{RFS}_i^{(s)} = \cos\left(r_i^{(s)}, r_{\mathrm{GT}}^{(s)}\right), \quad r_i^{(s)} = \underbrace{\left(R_{i,1}^{(s)}, \ldots, R_{i,J}^{(s)}\right)}_{\text{Normalized radiomic features for some algorithm } i \text{ on scan } s}$$

$j \in \{1, 2, \ldots, J\}$ **: Radiomic features**

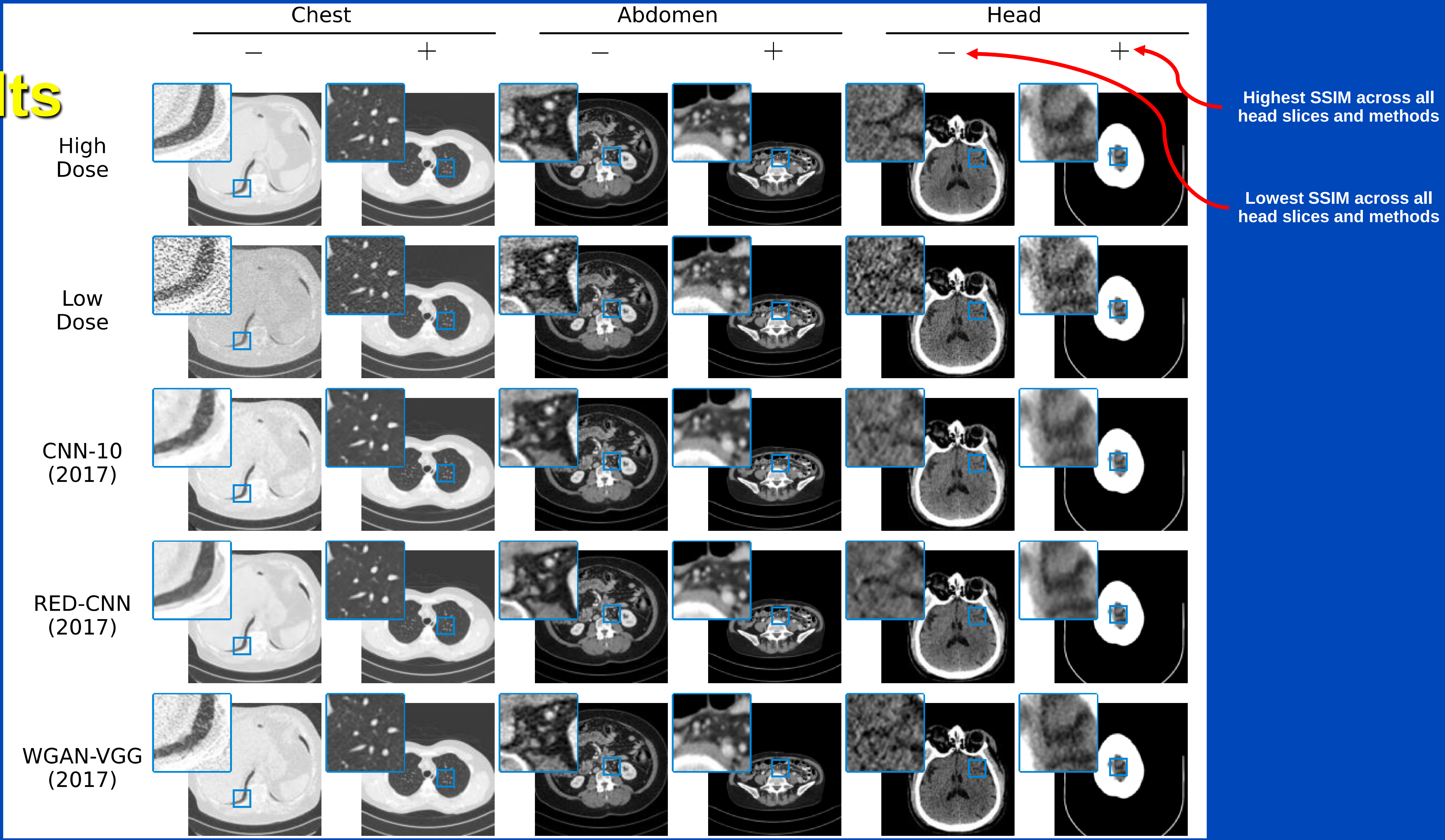
$i \in \{1, 2, \ldots, N\}$ **: Algorithms**



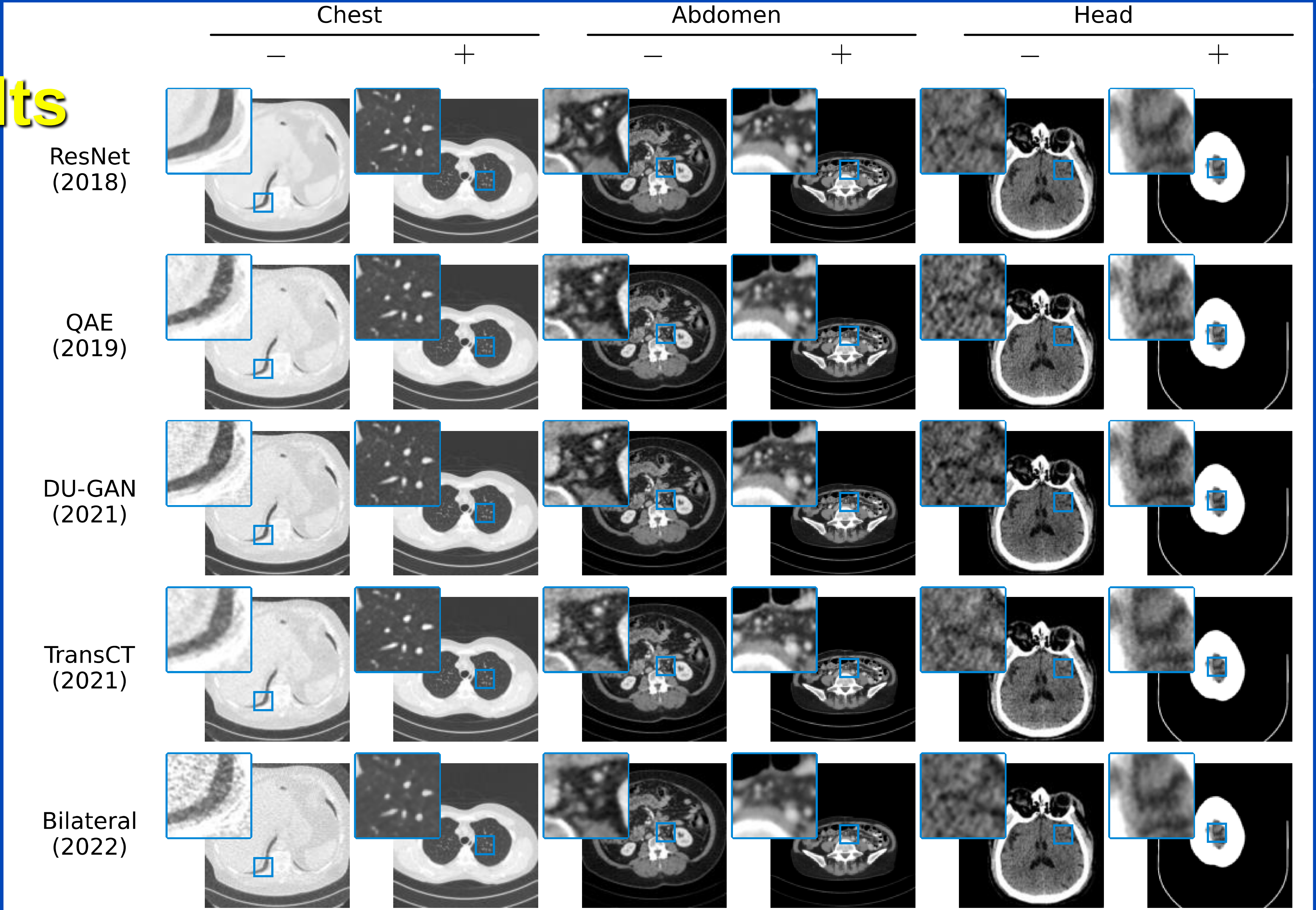


[1] Mayerhoefer, M. E., Materka, A., Langs, G., Häggström, I., Szczypiński, P., Gibbs, P., & Cook, G. (2020). Introduction to Radiomics. Journal of nuclear medicine : official publication, Society of Nuclear Medicine, 61(4), 488–495.

Results


Chest
Abdomen
Head
–
+
–
+
–
+
High Dose
Low Dose
CNN-10 (2017)
RED-CNN (2017)
WGAN-VGG (2017)
Highest SSIM across all head slices and methods
Lowest SSIM across all head slices and methods

Results



Chest
Abdomen
Head
−
+
−
+
−
+
ResNet (2018)
QAE (2019)
DU-GAN (2021)
TransCT (2021)
Bilateral (2022)

Results

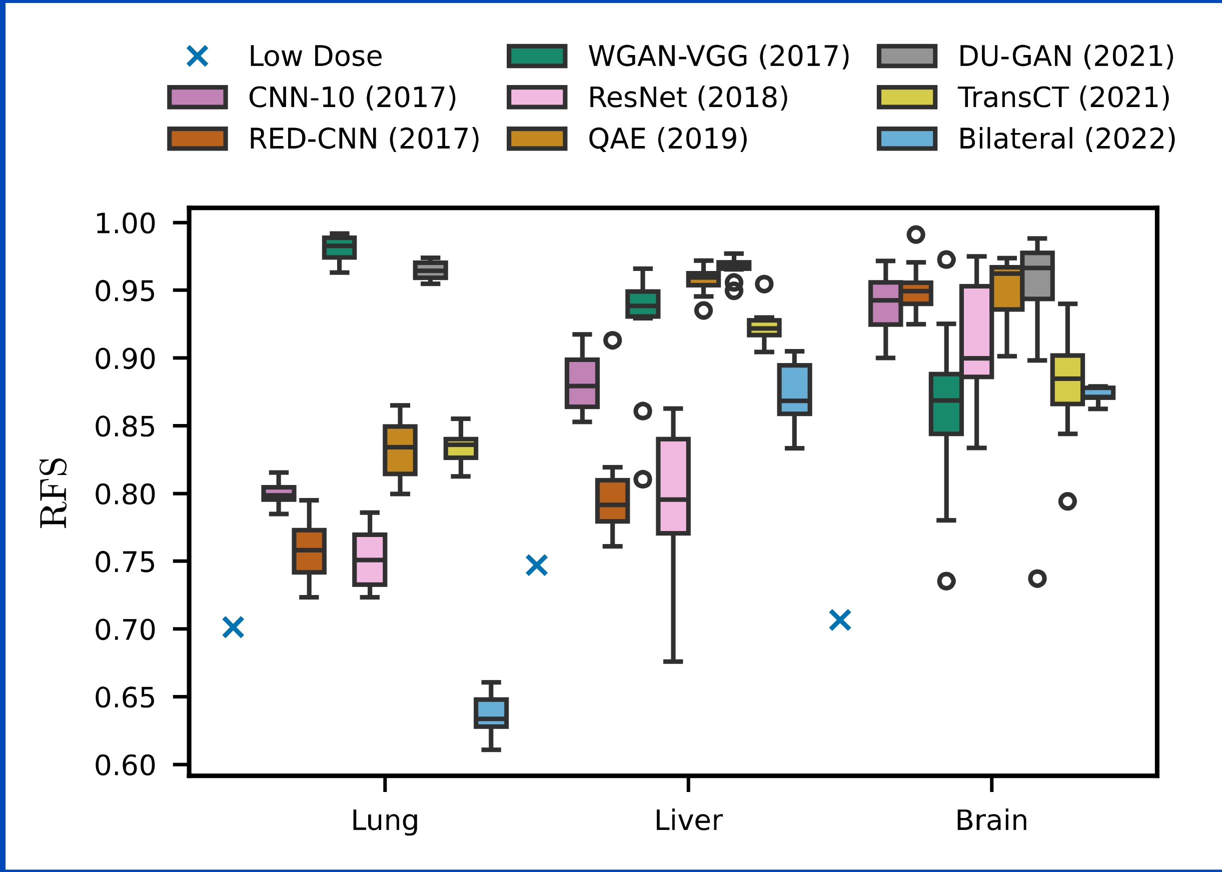
SSIM, PSNR, VIF

	Chest (10% dose)			Abdomen (25% dose)			Head (25% dose)		
	SSIM	PSNR (dB)	VIF	SSIM	PSNR (dB)	VIF	SSIM	PSNR (dB)	VIF
LD	0.34	18.77	0.09	0.84	28.67	0.34	0.88	26.4	0.55
CNN-10 (2017)	**0.5867 ± 0.0006**	**27.71 ± 0.02**	**0.1915 ± 0.0008**	**0.896 ± 0.001**	**32.4 ± 0.1**	**0.449 ± 0.003**	**0.896 ± 0.004**	**28.9 ± 0.6**	**0.620 ± 0.006**
RED-CNN (2017)	**0.609 ± 0.002**	**28.36 ± 0.03**	**0.221 ± 0.003**	**0.9028 ± 0.0007**	**33.22 ± 0.07**	**0.491 ± 0.008**	**0.904 ± 0.001**	**30.4 ± 0.2**	**0.69 ± 0.01**
WGAN-VGG (2017)	*0.51 ± 0.03*	*25.5 ± 0.2*	*0.148 ± 0.004*	*0.882 ± 0.002*	*30.5 ± 0.9*	*0.38 ± 0.01*	*0.88 ± 0.02*	*25 ± 3*	*0.53 ± 0.02*
ResNet (2018)	**0.610 ± 0.001**	**28.42 ± 0.03**	**0.224 ± 0.002**	*0.901 ± 0.002*	*33.15 ± 0.08*	*0.487 ± 0.006*	0.901 ± 0.005	*29.6 ± 0.8*	*0.67 ± 0.02*
QAE (2019)	*0.584 ± 0.003*	*27.62 ± 0.09*	*0.186 ± 0.003*	*0.894 ± 0.002*	*32.0 ± 0.2*	*0.418 ± 0.007*	*0.899 ± 0.001*	*28.5 ± 0.3*	*0.594 ± 0.008*
DU-GAN (2021)	*0.565 ± 0.004*	*26.7 ± 0.1*	*0.168 ± 0.002*	*0.894 ± 0.002*	*32.1 ± 0.3*	*0.427 ± 0.005*	0.903 ± 0.003	*29 ± 1*	*0.622 ± 0.005*
TransCT (2021)	*0.563 ± 0.002*	*26.99 ± 0.05*	*0.167 ± 0.002*	*0.877 ± 0.003*	*30.5 ± 0.2*	*0.372 ± 0.007*	*0.849 ± 0.005*	*24.7 ± 0.4*	*0.44 ± 0.01*
Bilateral (2022)	*0.555 ± 0.001*	*25.59 ± 0.04*	*0.159 ± 0.002*	*0.859 ± 0.003*	*27.1 ± 0.1*	*0.361 ± 0.003*	*0.873 ± 0.002*	*26.6 ± 0.1*	*0.500 ± 0.004*

Bold: Significantly better than previously best method
***Italics*: Significantly worse than previously best method**

Results

Radiomic Feature Similarity (RFS)



	Lung	Liver	Brain
LD	0.7	0.75	0.71
CNN-10 (2017)	**0.800 ± 0.009**	**0.88 ± 0.02**	**0.94 ± 0.02**
RED-CNN (2017)	*0.76 ± 0.02*	*0.80 ± 0.04*	0.95 ± 0.02
WGAN-VGG (2017)	**0.98 ± 0.01**	**0.92 ± 0.05**	*0.86 ± 0.07*
ResNet (2018)	*0.75 ± 0.02*	*0.79 ± 0.06*	0.91 ± 0.05
QAE (2019)	*0.83 ± 0.02*	**0.96 ± 0.01**	0.95 ± 0.02
DU-GAN (2021)	*0.965 ± 0.007*	**0.967 ± 0.008**	0.94 ± 0.08
TransCT (2021)	*0.83 ± 0.01*	*0.92 ± 0.01*	*0.88 ± 0.04*
Bilateral (2022)	*0.64 ± 0.01*	*0.87 ± 0.02*	*0.873 ± 0.006*

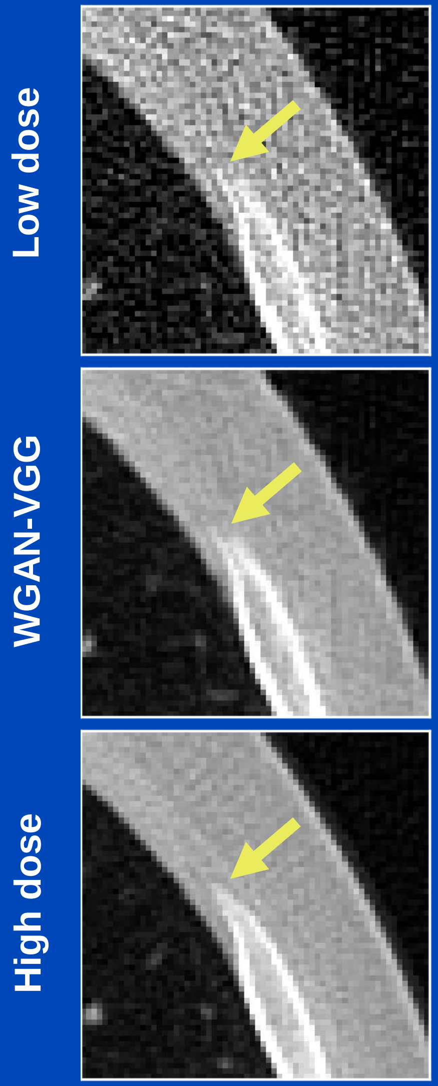
Bold: Significantly better than previously best method
***Italics*: Significantly worse than previously best method**

Summary & Conclusions

- Revisited some of the deep learning-based methods for low dose CT image denoising
- Newer algorithms do not consistently outperform earlier ones both in terms of standard IQA metrics and the proposed radiomic feature similarity

→ Highlights the need for more rigorous and fair evaluation of novel deep learning based denoising methods for LDCT image denoising*

Important research direction: Develop metrics that capture the robustness of algorithms wrt anatomical details



*Similar to reality checks in related fields [1, 2]

[1] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," in *ICLR*, 2018.
[2] K. Musgrave, S. Belongie, and S.-N. Lim, "A metric learning reality check," in *ECCV,* 2020.