Known Operator Learning - Towards Integration of Prior Knowledge into Machine Learning

Andreas Maier
Katharina Breininger

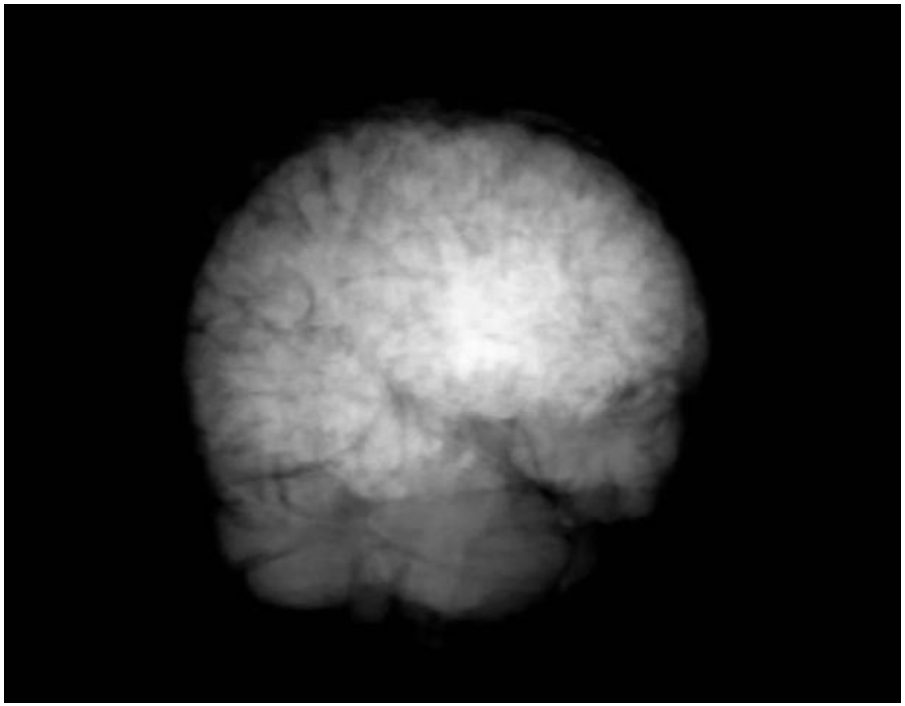
Lehrstuhl für Mustererkennung (Informatik 5),
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

European Research Council
DFG Deutsche Forschungsgemeinschaft
European Commission | Horizon 2020 European Union funding for Research & Innovation

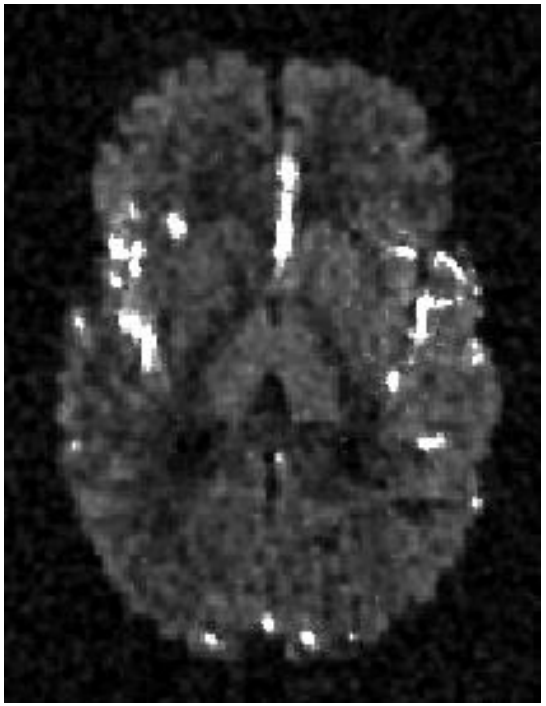
Known Operator Learning

- Motivation
- Known Operators in Deep Networks
- Examples in Computed Tomography
- Future Work

Image Reconstruction

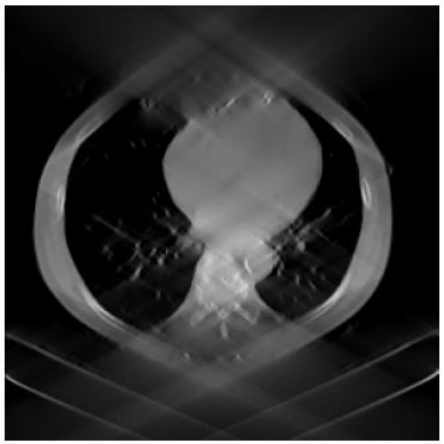


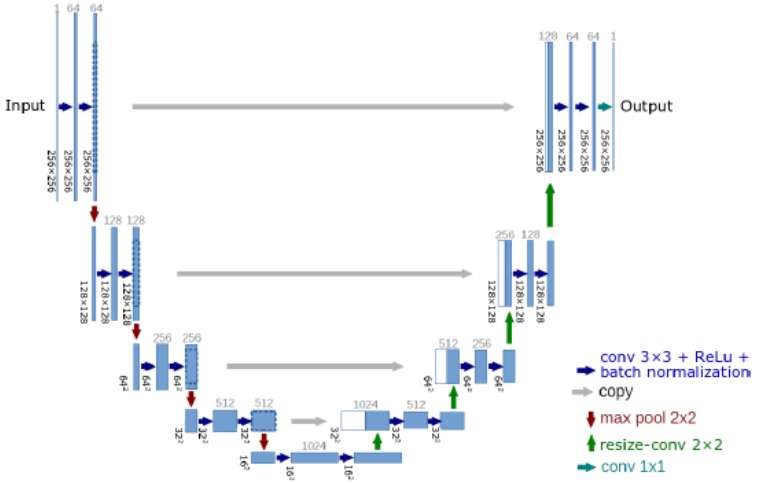
Projection Data

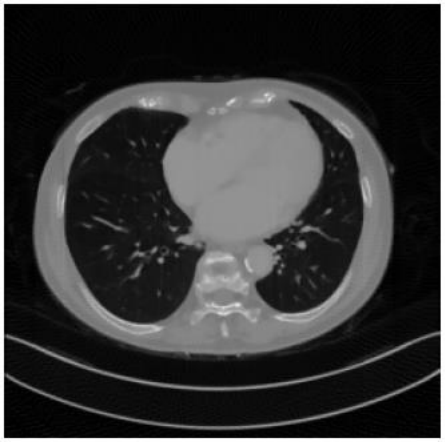


Sliced Volume

Deep Learning in Image Reconstruction?

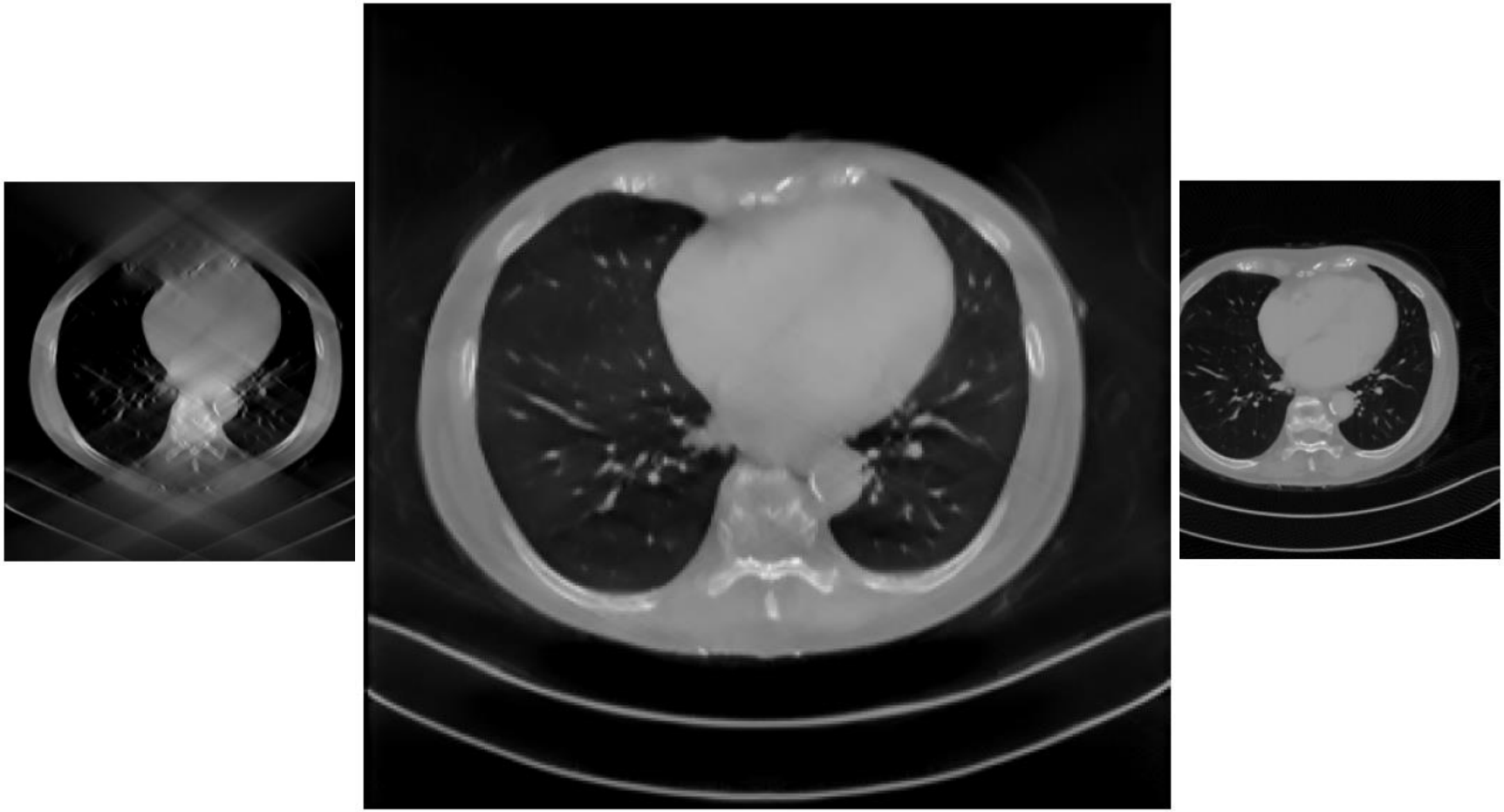






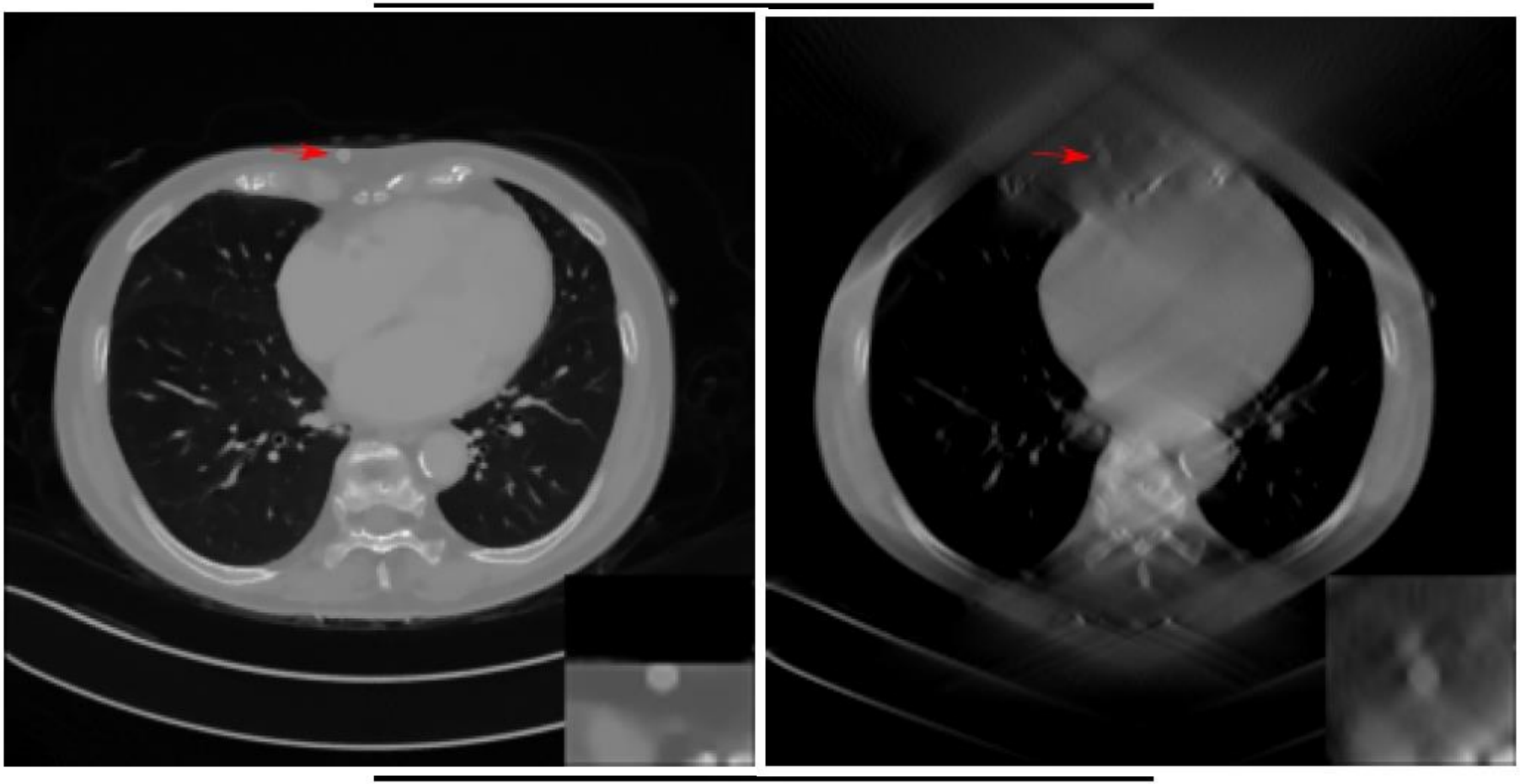
[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?



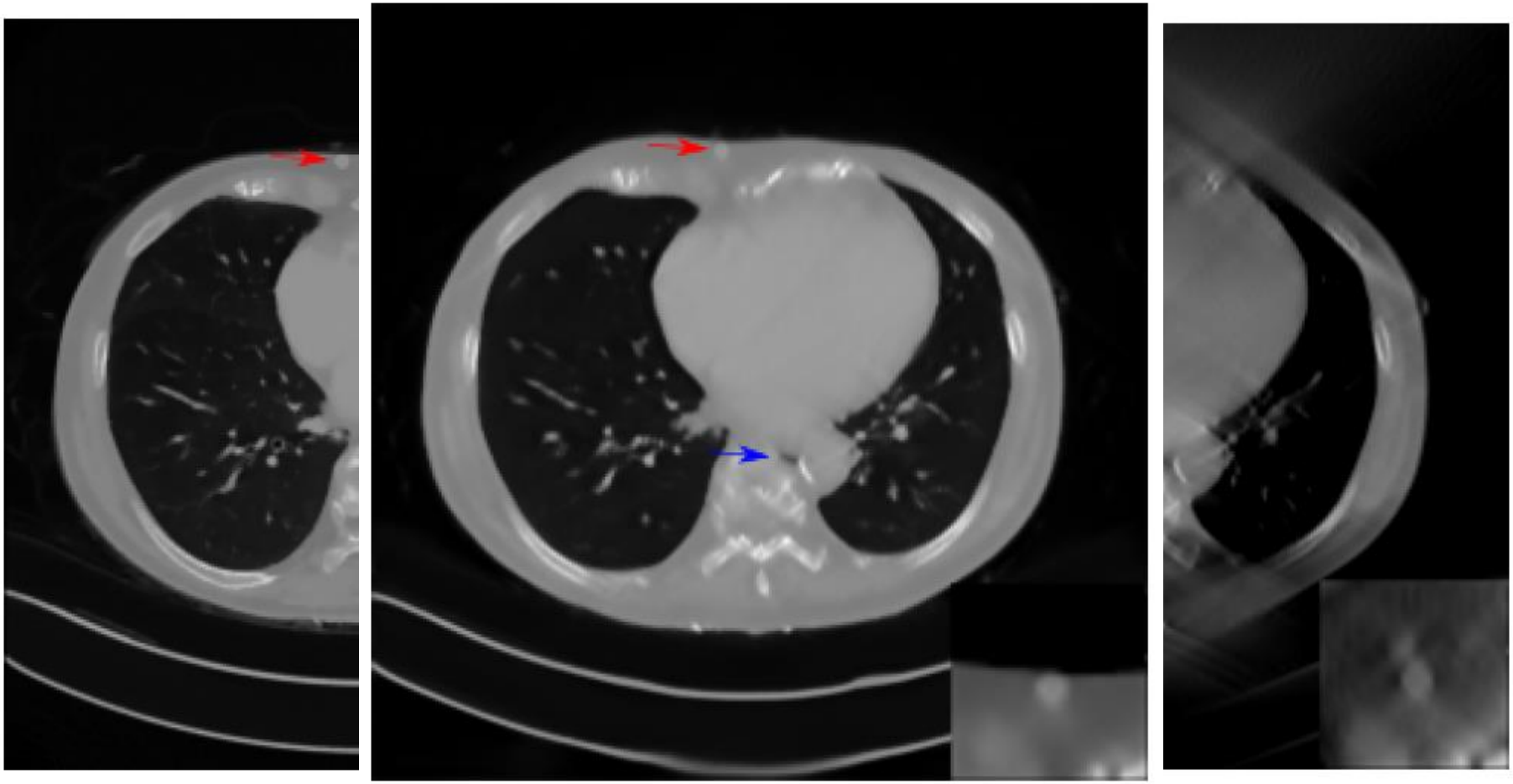
[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?



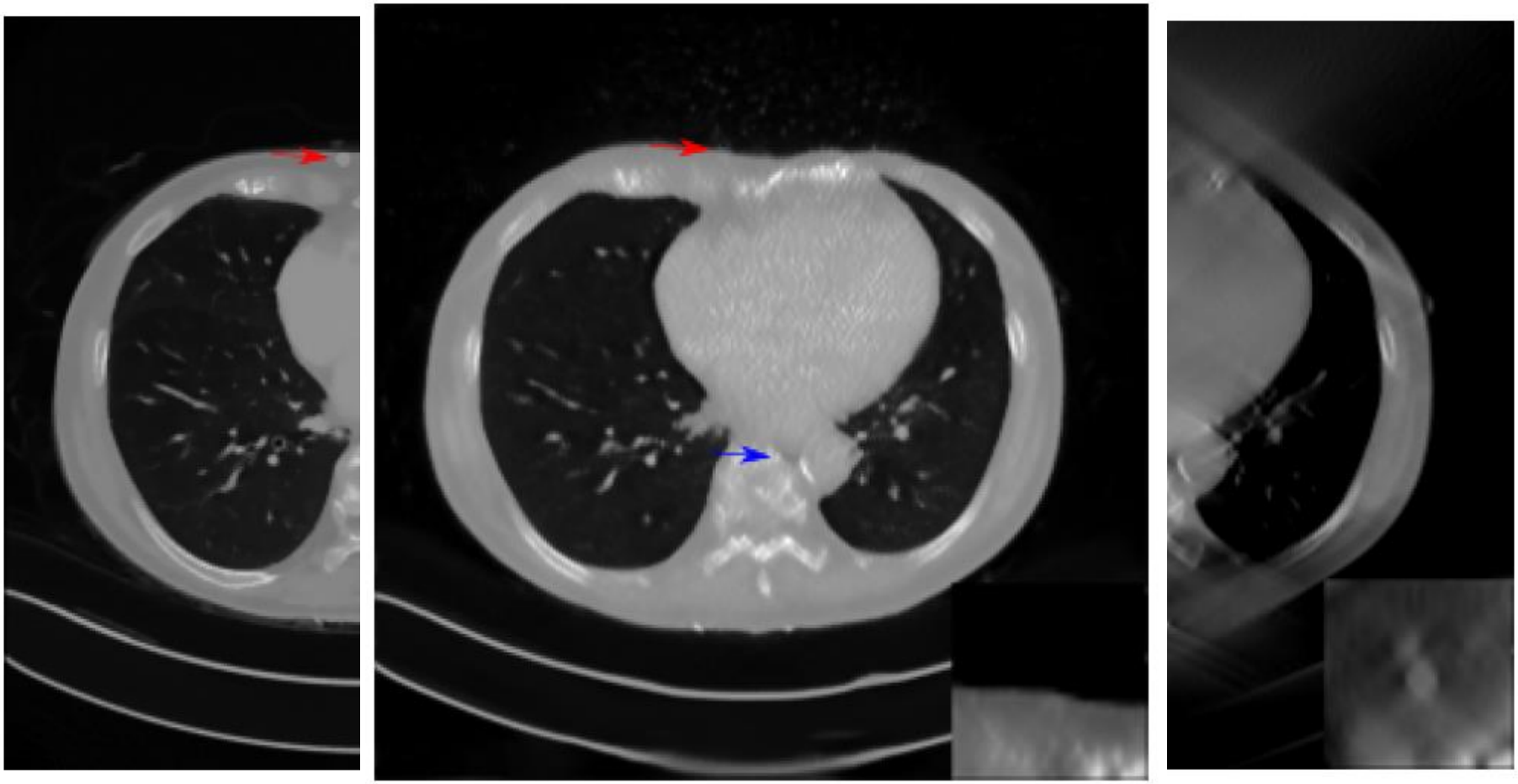
[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?



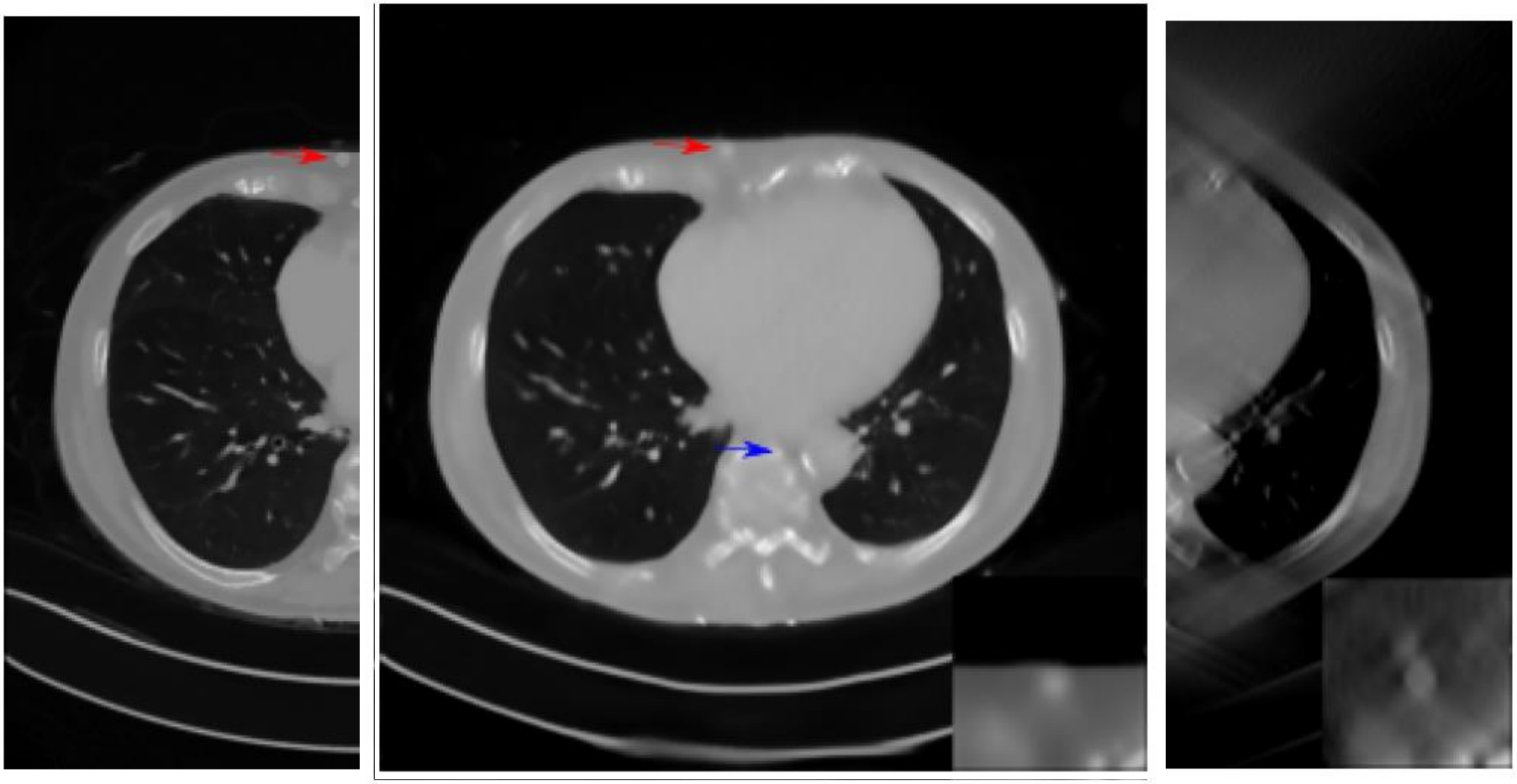
[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?



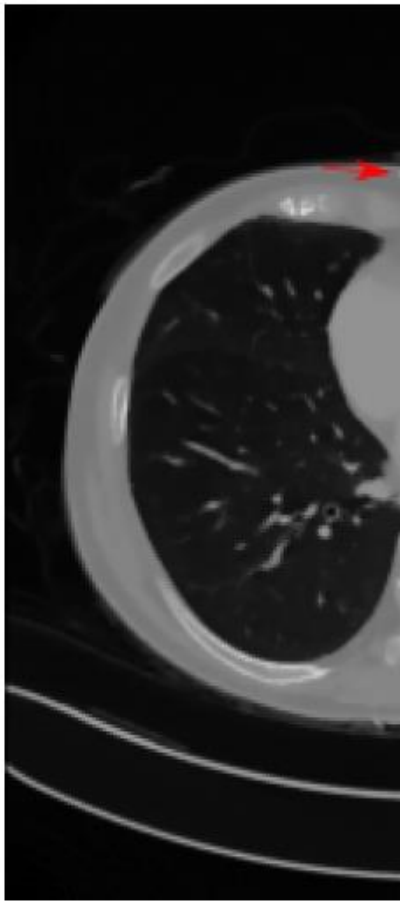
[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

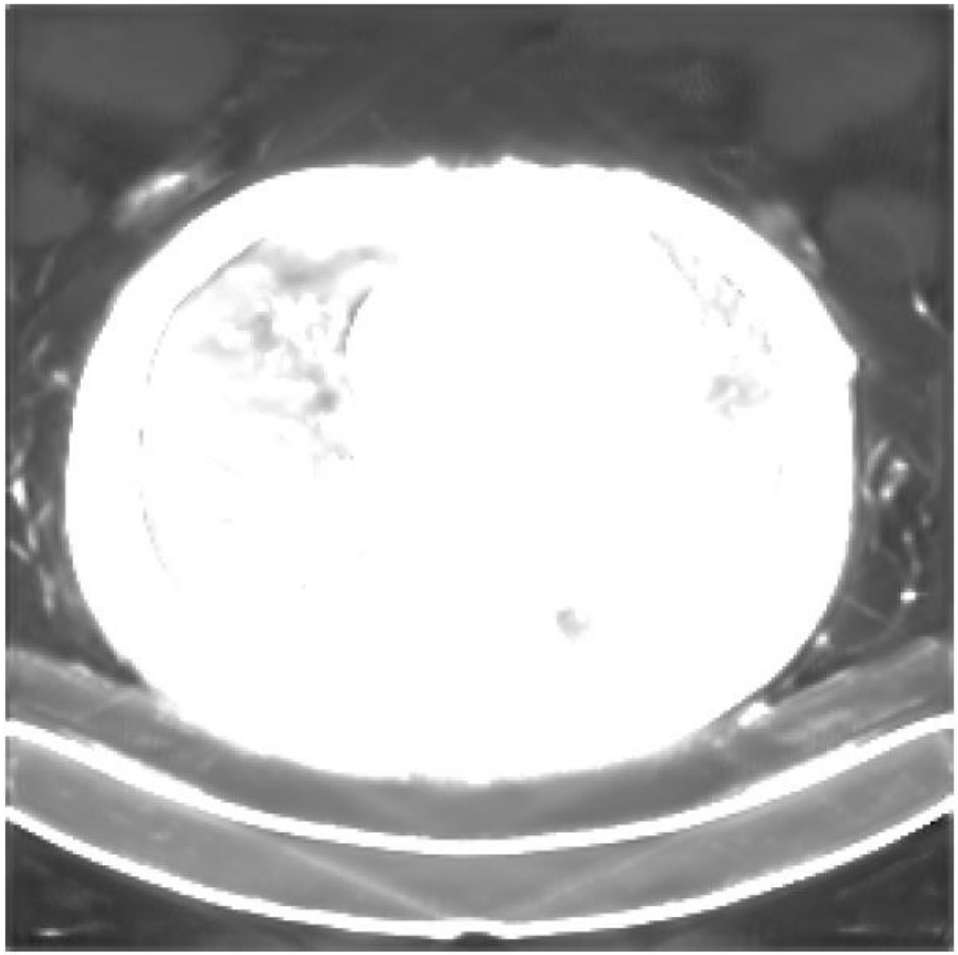
Deep Learning in Image Reconstruction?

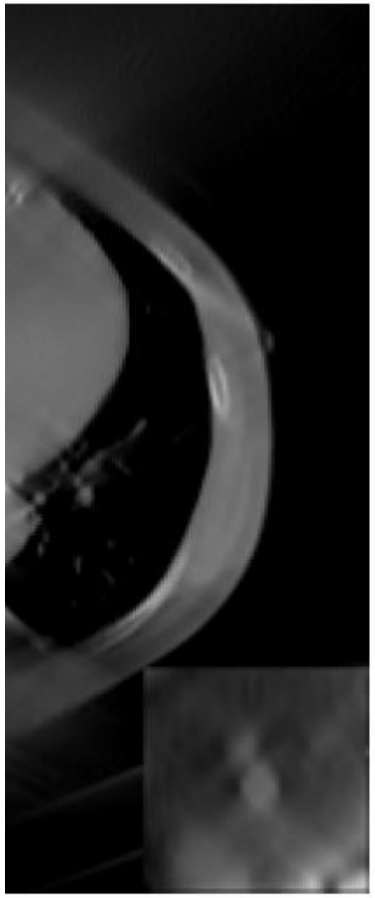


[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Deep Learning in Image Reconstruction?







[4] Yixing Huang et al. Some Investigations on Robustness of Deep Learning in Limited Angle Tomography. MICCAI 2018.

Known Operators in Neural Networks

"Let’s not reinvent the wheel…"

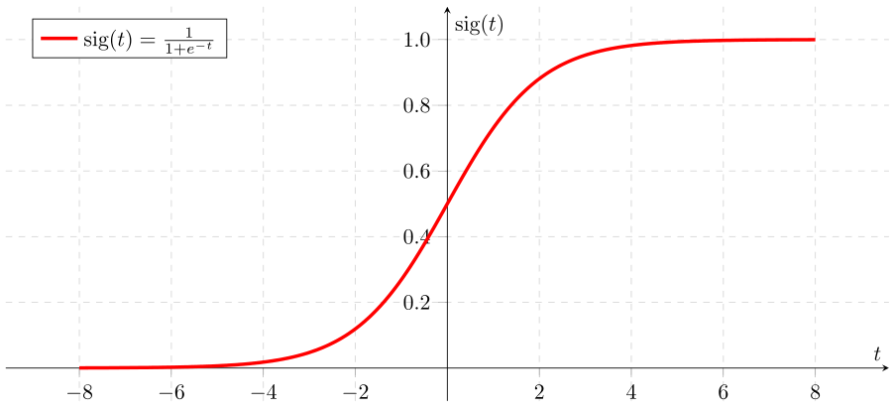
Universal Approximation Theorem

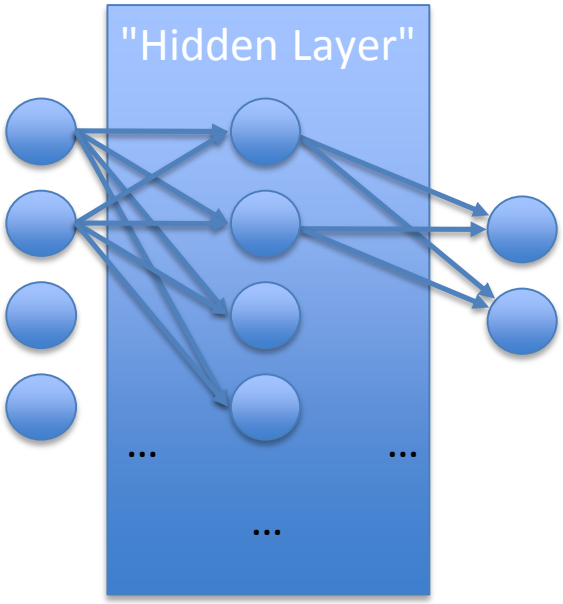
- **Any continuous function can be approximated by Neural Net**

$$u(\mathbf{x}) \approx U(\mathbf{x}) = \sum_i u_i s(\mathbf{w}_i^\top \mathbf{x} + w_{j,0})$$

- **The error is bound by**

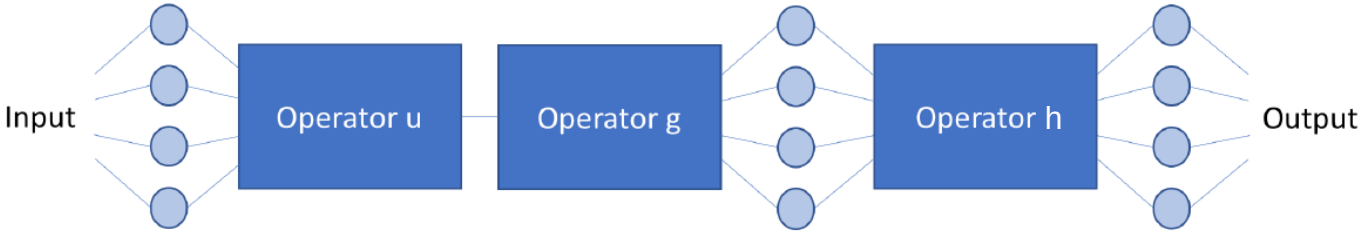
$$|U(\mathbf{x}) - u(\mathbf{x})| \leq \epsilon_u$$





Known Operators – Precision Learning

- **Consider the case of using known operators in the net**



- **Specifically consider the use of two operators in sequence**

$$f(\mathbf{x}) = g(\mathbf{u}(\mathbf{x}))$$

[5] Andreas Maier et al. Precision Learning: Towards use of known operators in neural networks. ICPR 2018.

Approximation Sequences

- **Sequential operations**

$$f(\mathbf{x}) = g(\mathbf{u}(\mathbf{x}))$$

- **Can be approximated:**

$$\begin{aligned} F_u(\mathbf{x}) &= g(\mathbf{U}(\mathbf{x})) &&= f(\mathbf{x}) - e_u \\ F_g(\mathbf{x}) &= G(\mathbf{u}(\mathbf{x})) &&= f(\mathbf{x}) - e_g \\ F(\mathbf{x}) &= G(\mathbf{U}(\mathbf{x})) &&= f(\mathbf{x}) - e_f \end{aligned}$$

Error of Approximation Sequences

- **Approximation introduces error**

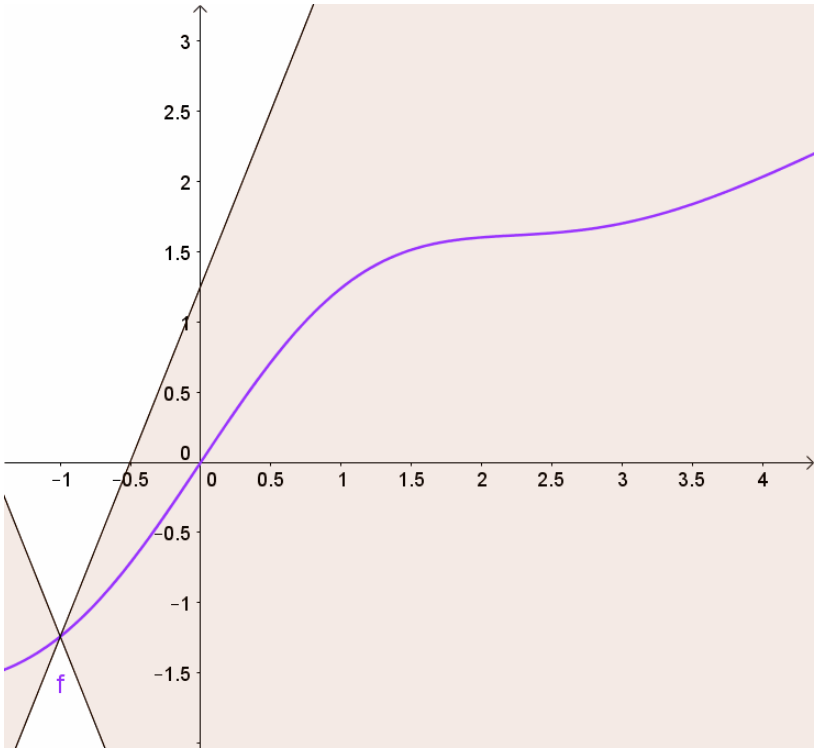
$$
\begin{aligned}
f(\mathbf{x}) &= g(\mathbf{u}(\mathbf{x})) = G(\mathbf{u}(\mathbf{x})) + e_g \\
&= \sum_j g_j s\left(u_j(\mathbf{x})\right) + g_0 + e_g \\
&= \sum_j g_j s\left(U_j(\mathbf{x}) + e_{u_j}\right) + g_0 + e_g
\end{aligned}
$$

- **Can we find bounds on this error?**

Bounds for Sigmoid Functions

- **Sigmoid function satisfies the following upper bound:**

$$s(x+e) \leq s(x) + l_s \cdot |e|$$



https://en.wikipedia.org/wiki/Lipschitz_continuity

Bounds for Sigmoid Functions

- **Sigmoid function satisfies the following upper bound:**

$$s(x+e) \leq s(x) + l_s \cdot |e|$$

- **However this does not hold for linear combinations**

- **Alternate formulation:**

$$g_j s(x+e) \leq g_j s(x) + |g_j| \cdot l_s \cdot |e|$$

Error of Approximation Sequences (2)

- **Recall**

$$f(\mathbf{x}) = \sum_j g_j s\left(U_j(\mathbf{x}) + e_{u_j}\right) + g_0 + e_g \leq \underbrace{\sum_j g_j s(U_j(\mathbf{x})) + g_0}_{F(\mathbf{x})} + \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + e_g$$

- **Subtraction of F(x) yields**

$$\underbrace{f(\mathbf{x}) - F(\mathbf{x})}_{e_f} \leq \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + e_g$$

- **Which is bounded by**

$$e_f \leq \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + \epsilon_g$$

Error of Approximation Sequences (3)

- **Same idea can also be used for lower bound**

$$e_f \geq -\sum_j |g_j| \cdot l_s \cdot |e_{u_j}| - \epsilon_g$$

- **Thus a general bound is**

$$|e_f| \leq \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + \epsilon_g$$

Observations on Bounds

- **Bound on Error:**

$$|e_f| \leq \sum_j |g_j| \cdot l_s \cdot |e_{u_j}| + \epsilon_g$$

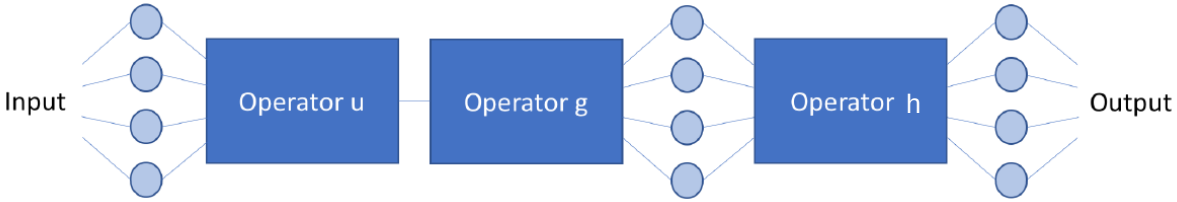
Error U(x)

Error G(x)

- **Observations**
  - **Error in U(x) and G(x) additive**
  - **Error of U(x) amplified by g(x)**
  - **Interpretation as Feature Extractor => Importance of Features**
  - **Requires Lipschitz continuity**

Observations on Bounds

- **Extension to Deep Networks:**



- **Proof by Recursion:**

nature machine intelligence
ARTICLES
https://doi.org/10.1038/s42256-019-0077-5

Learning with known operators reduces maximum error bounds

Andreas K. Maier[1*], Christopher Syben[1], Bernhard Stimpel[1], Tobias Würfl[1], Mathis Hoffmann[1], Frank Schebesch[1], Weilin Fu[1], Leonid Mill[1], Lasse Kling[2] and Silke Christiansen[2,3]

We describe an approach for incorporating prior knowledge into machine learning algorithms. We aim at applications in physics and signal processing in which we know that certain operations must be embedded into the algorithm. Any operation that allows computation of a gradient or sub-gradient towards its inputs is suited for our framework. We derive a maximal error bound for deep nets that demonstrates that inclusion of prior knowledge results in its reduction. Furthermore, we show experimentally that known operators reduce the number of free parameters. We apply this approach to various tasks ranging from computed tomography image reconstruction over vessel segmentation to the derivation of previously unknown imaging algorithms. As such, the concept is widely applicable for many researchers in physics, imaging and signal processing. We assume that our analysis will support further investigation of known operators in other fields of physics, imaging and signal processing.

Computed Tomography

- Efficient solution via filtered back-projection:

$$f(x,y) = \int_0^{\pi} p(s,\theta) * h(s) \Big|_{s=x\cos\theta + y\sin\theta} d\theta$$

- Three steps:
  - Convolution along s
  - Back-projection along θ
  - Suppress negative values

Computed Tomography

- Efficient solution via filtered back-projection (FBP):

$$f(x,y) = \int_0^{\pi} p(s,\theta) * h(s)\Big|_{s=x\cos\theta + y\sin\theta} d\theta$$

In the continuous case: $h(s)$ in frequency domain = „ramp“

- Can also be derived in matrix notation:

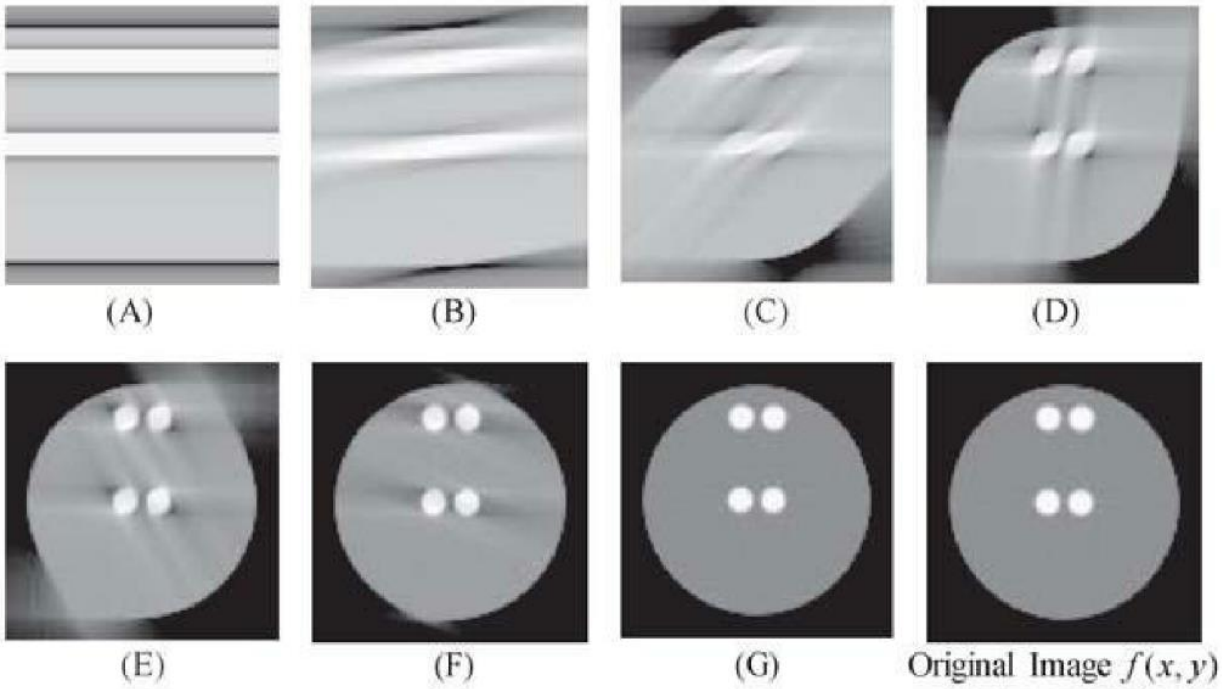
$$\mathbf{A}\mathbf{x} = \mathbf{p}$$

$$\mathbf{x} = \mathbf{A}^{+}\mathbf{p} = \mathbf{A}^{\mathbf{T}} \underbrace{\left(\mathbf{A}\mathbf{A}^{\mathbf{T}}\right)^{+}}_{\textbf{Filter}} \mathbf{p}$$

Computed Tomography

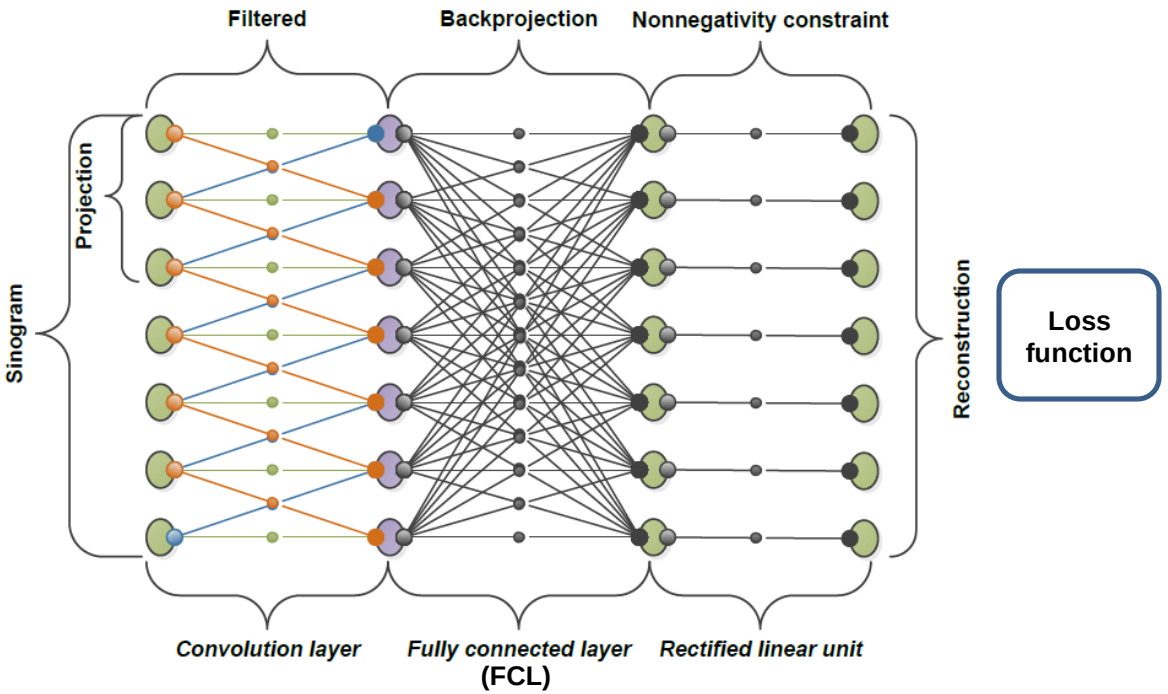
- Efficient solution via filtered back-projection (FBP):

$$f(x,y) = \int_0^{\pi} p(s,\theta) * h(s)\Big|_{s=x\cos\theta+y\sin\theta} d\theta$$



Computed Tomography using Neural Networks

- All three steps can be modeled as neural network:



- Interesting: All weights are known from FBP

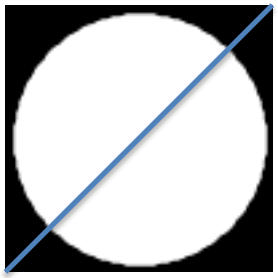
Discretization (2)

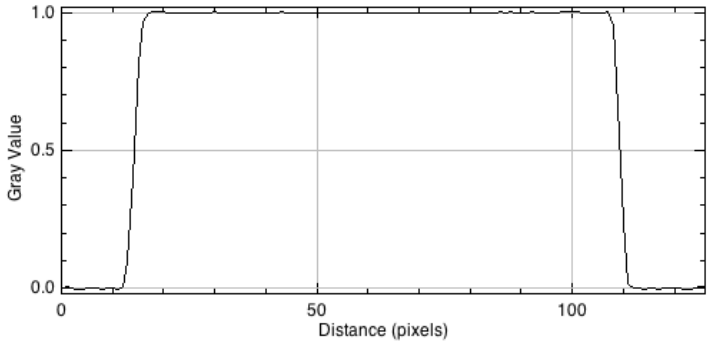
- **Implementation by the book:**

[26] Christopher Syben et al. A Deep Learning Approach for Reconstruction Filter Kernel Discretization. CT Meeting 2018. Accepted

Discretization (2)

- **Implementation by the book:**

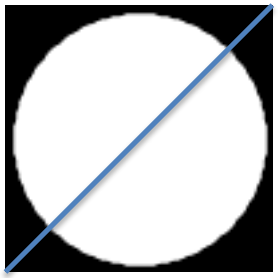


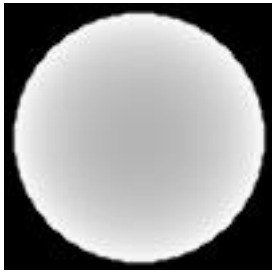


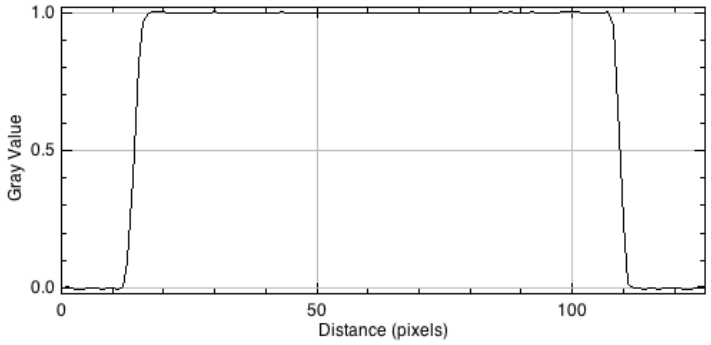
[26] Christopher Syben et al. A Deep Learning Approach for Reconstruction Filter Kernel Discretization. CT Meeting 2018. Accepted

Discretization (2)

- **Implementation by the book:**



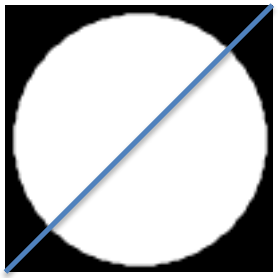


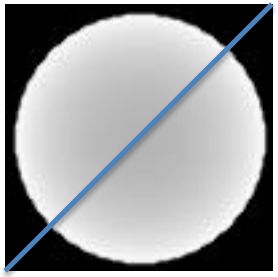


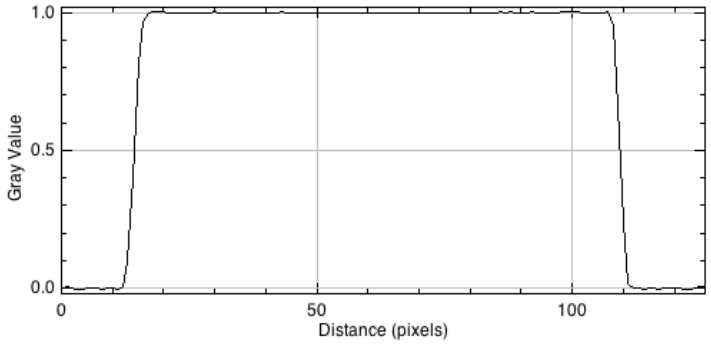
[26] Christopher Syben et al. A Deep Learning Approach for Reconstruction Filter Kernel Discretization. CT Meeting 2018. Accepted

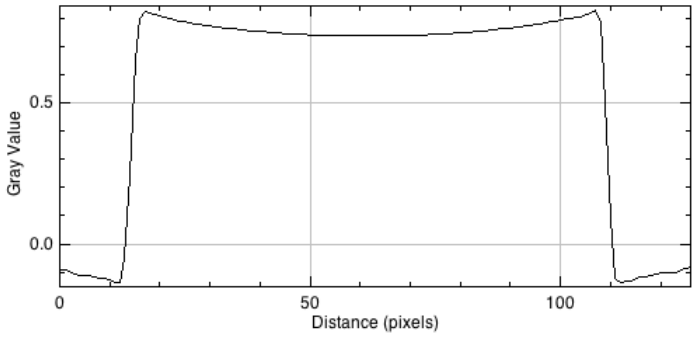
Discretization (2)

- **Implementation by the book:**









[26] Christopher Syben et al. A Deep Learning Approach for Reconstruction Filter Kernel Discretization. CT Meeting 2018. Accepted

Discretization (3)

- **Problem: „Ramp“-filter assumes infinite projections**
  → discretization error

- **Idea: Neural Networks are discrete end-to-end**

  → Network should be able to learn correct solution

- **Correct discretization is an intrinsic property of the net**

$$\mathbf{x} = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{p}$$
$$\mathbf{x} = \mathbf{A}^\top\mathbf{F}^\mathbf{H}\mathbf{K}\mathbf{F}\mathbf{p}$$

Discretization (3)

- **Problem: „Ramp“-filter assumes infinite projections**
  → discretization error

- **Idea: Neural Networks are discrete end-to-end**

  → Network should be able to learn correct solution

- **Correct discretization is an intrinsic property of the net**

$$\mathbf{x} = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{p}$$

$$\mathbf{x} = \mathbf{A}^\top\mathbf{F}^\mathbf{H}\mathbf{K}\mathbf{F}\mathbf{p}$$

Net

Discretization (3)

- **Problem: „Ramp“-filter assumes infinite projections**
  → discretization error

- **Idea: Neural Networks are discrete end-to-end**
  → Network should be able to learn correct solution

- **Correct discretization is an intrinsic property of the net**

$$\mathbf{x} = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{p}$$
$$\mathbf{x} = \mathbf{A}^\top\mathbf{F}^\mathbf{H}\mathbf{K}\mathbf{F}\mathbf{p}$$

- **Associated optimization problem:**

$$f(\mathbf{K}) = \frac{1}{2}||\mathbf{A}^\top\mathbf{F}^\mathbf{H}\mathbf{K}\mathbf{F}\mathbf{p} - \mathbf{x}||_\mathbf{2}^\mathbf{2}$$

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2}||\mathbf{A}^\top \mathbf{F}^{\mathbf{H}} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}||_2^2$$

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2}||\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}||_\mathbf{2}^\mathbf{2}$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \mathbf{F}\mathbf{A}(\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x})(\mathbf{F}\mathbf{p})^\top$$

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2}||\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}||_\mathbf{2}^\mathbf{2}$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \mathbf{F}\mathbf{A}(\underbrace{\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p}}_{\text{Recon}} - \mathbf{x})(\mathbf{F}\mathbf{p})^\top$$

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2}||\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}||_\mathbf{2}^\mathbf{2}$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \mathbf{F}\mathbf{A}(\underbrace{\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}}_{\text{Error}})(\mathbf{F}\mathbf{p})^\top$$

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2}||\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}||_\mathbf{2}^\mathbf{2}$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \mathbf{F}\mathbf{A}\left(\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}\right)(\mathbf{F}\mathbf{p})^\top$$

Error

Backpropagation

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2}||\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}||_\mathbf{2}^\mathbf{2}$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \mathbf{F}\mathbf{A}(\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x})(\mathbf{F}\mathbf{p})^\top$$

Backpropagation

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2}||\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}||_\mathbf{2}^\mathbf{2}$$

- **Gradient:**

$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \underbrace{\mathbf{F}\mathbf{A}(\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x})}_{\text{Backpropagation}} \underbrace{(\mathbf{F}\mathbf{p})^\top}_{\text{„l-1“}}$$

Discretization (3)

- **Objective function:**

$$f(\mathbf{K}) = \frac{1}{2}||\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x}||_\mathbf{2}^\mathbf{2}$$

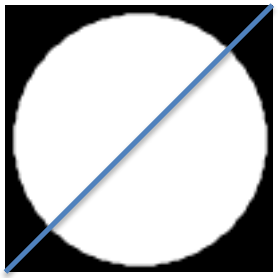
- **Gradient:**

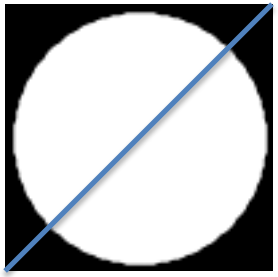
$$\frac{\partial f(\mathbf{K})}{\partial \mathbf{K}} = \underbrace{\mathbf{F}\mathbf{A}(\mathbf{A}^\top \mathbf{F}^\mathbf{H} \mathbf{K} \mathbf{F} \mathbf{p} - \mathbf{x})}_{\text{Backpropagation}} \underbrace{(\mathbf{F}\mathbf{p})^\top}_{\text{„l-1“}}$$

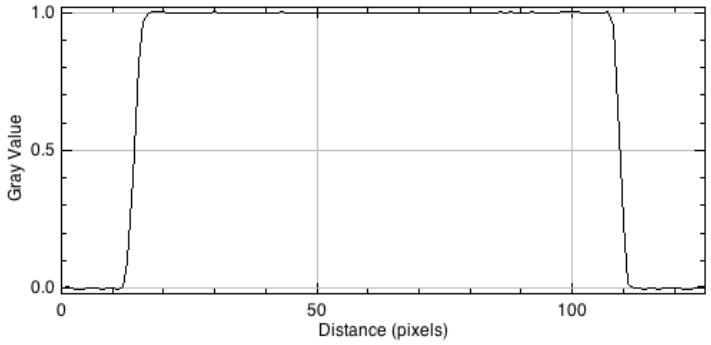
$$\Delta w_{ij}^{(l)} = \eta \, \delta_j^{(l)} \, y_i^{(l-1)}$$

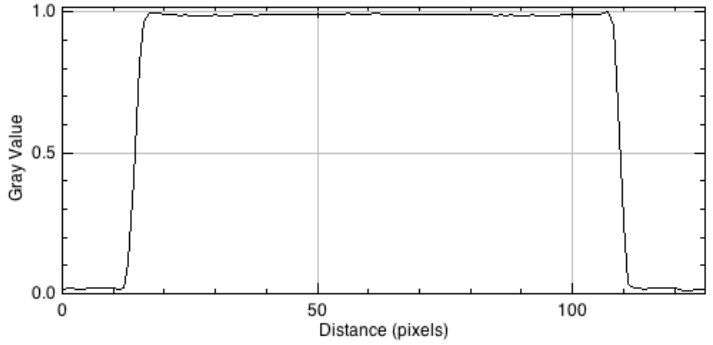
Discretization (3)

- **Filter after "Learning":**





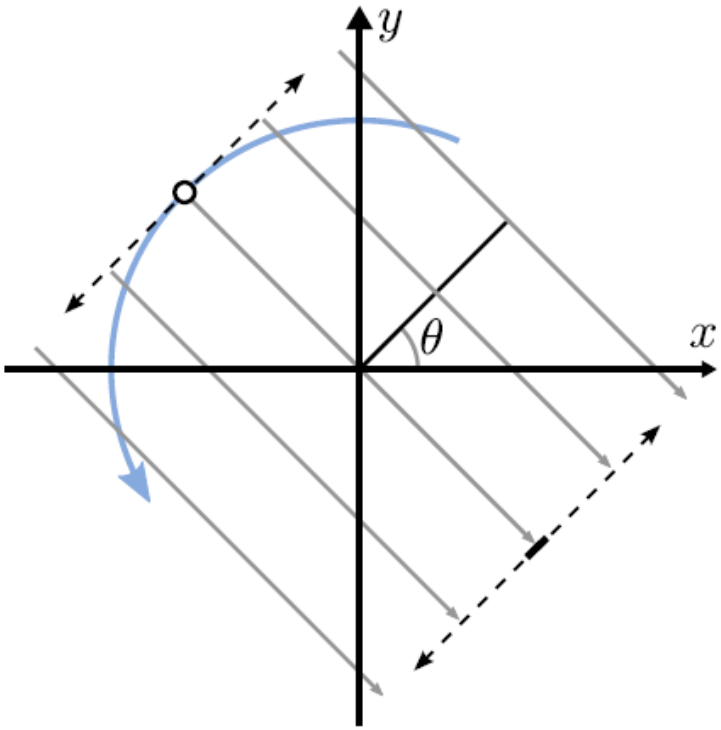


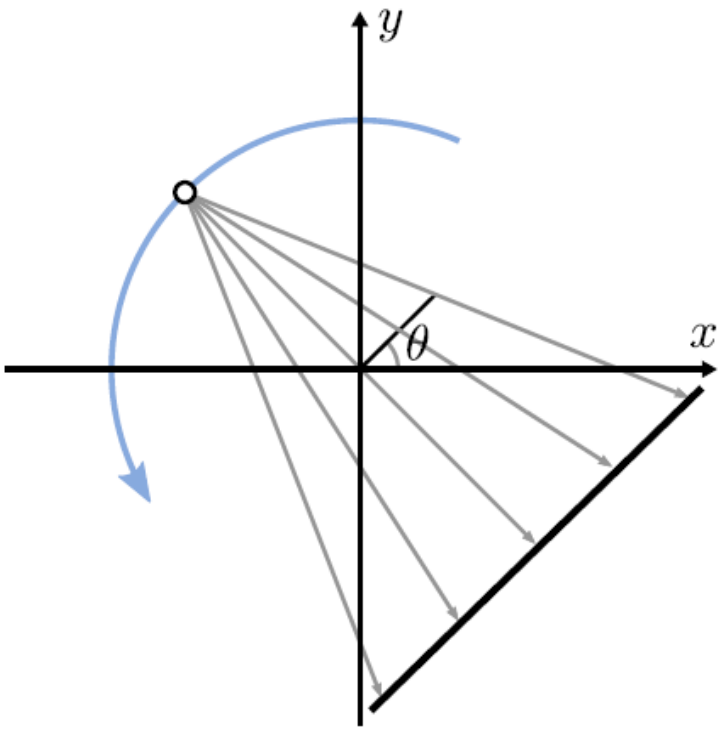


[26] Christopher Syben et al. A Deep Learning Approach for Reconstruction Filter Kernel Discretization. CT Meeting 2018. Accepted

Computed Tomography using Neural Networks

- Parallel vs. fan-beam reconstruction:

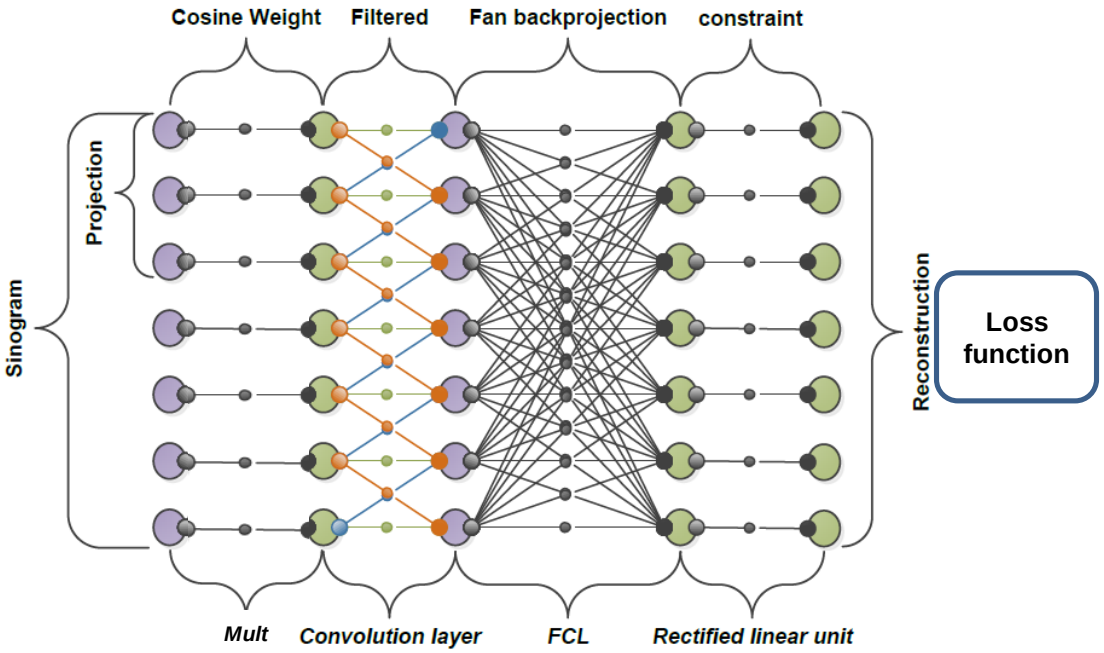




Computed Tomography using Neural Networks

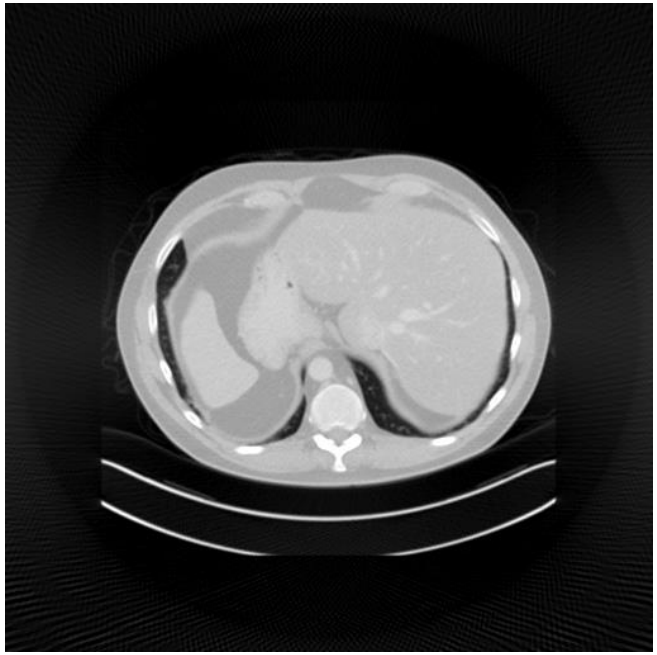
- Fan-beam reconstruction formula:

$$x = \mathbf{A}^\top \mathbf{C}\mathbf{W}\mathbf{p}$$



Computed Tomography using Neural Networks

- **Application to incomplete scans [2]**

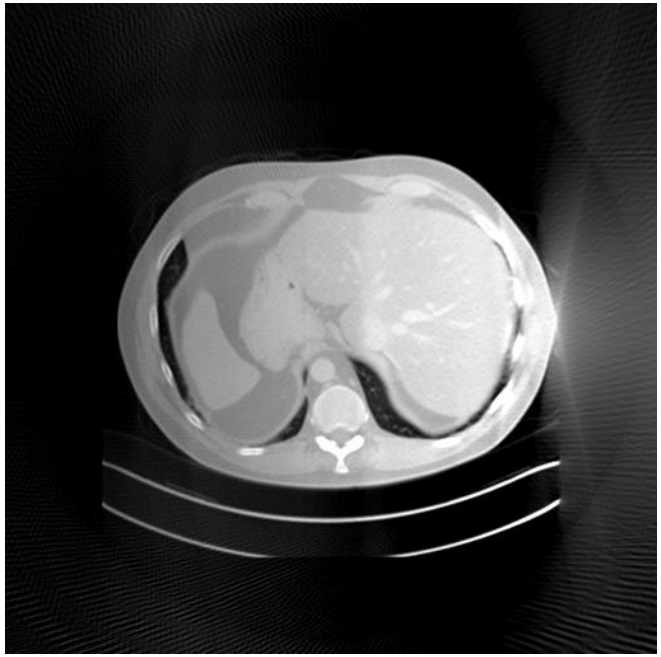


Reconstruction with 360 deg

[6] Tobias Würfl, Florin Ghesu, Vincent Christlein, Andreas Maier. Deep Learning Computed Tomography. MICCAI 2016.

Computed Tomography using Neural Networks

- **Application to incomplete scans [2]**

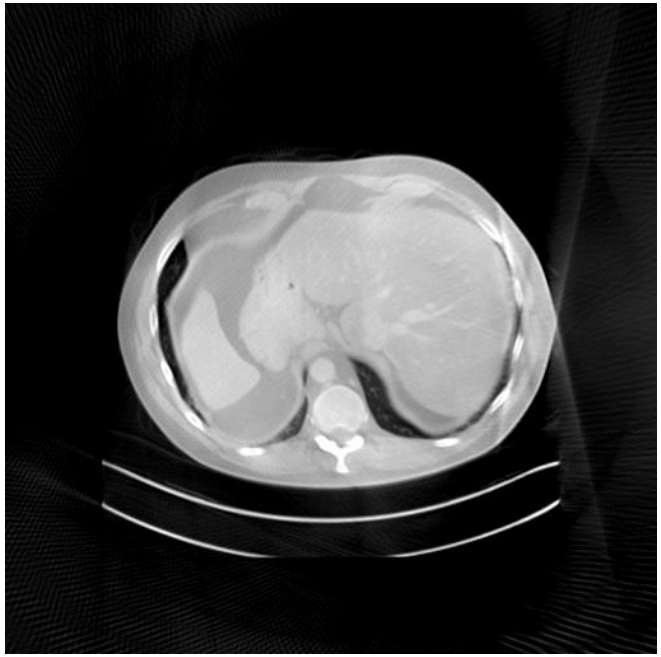


Reconstruction with 180 deg (FBP)

[6] Tobias Würfl, Florin Ghesu, Vincent Christlein, Andreas Maier. Deep Learning Computed Tomography. MICCAI 2016.

Computed Tomography using Neural Networks

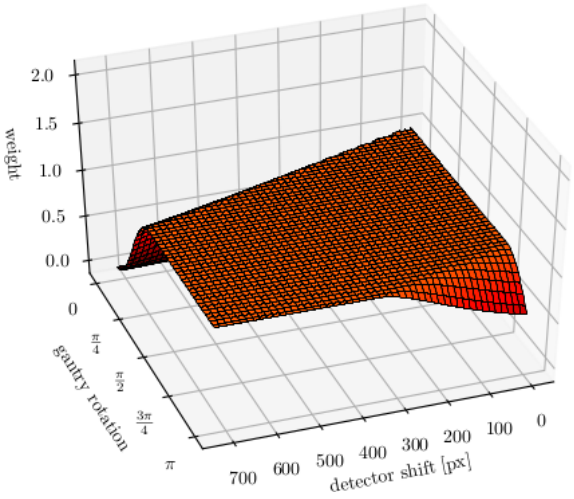
- **Application to incomplete scans [2]**



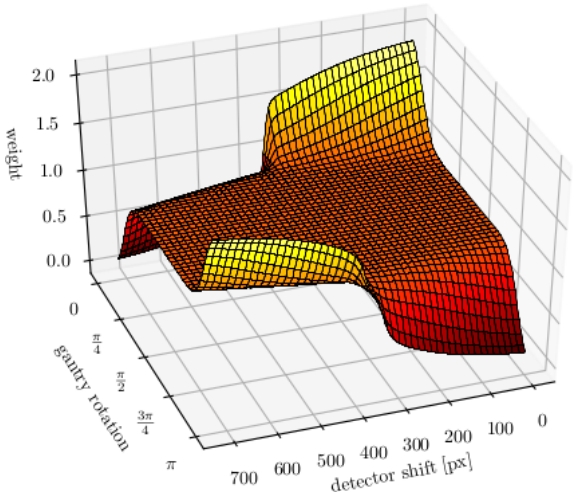
Reconstruction with 180 deg (NN)

[6] Tobias Würfl, Florin Ghesu, Vincent Christlein, Andreas Maier. Deep Learning Computed Tomography. MICCAI 2016.

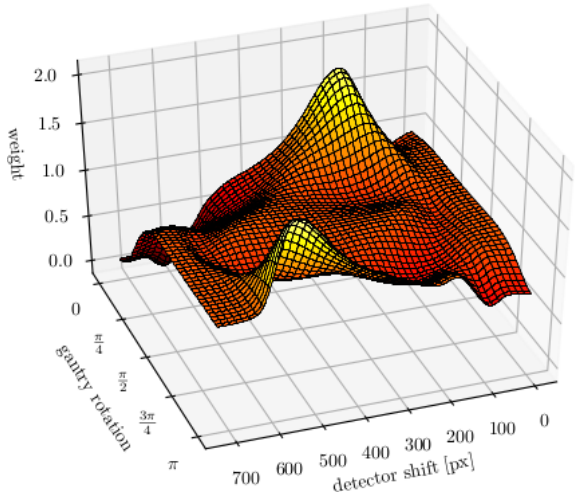
Learned Weights



Parker et al.
1982
„works well“



Schäfer et al.
2017
„works better“



Würfl et al.
2016
data optimal

[6] Tobias Würfl, Florin Ghesu, Vincent Christlein, Andreas Maier. Deep Learning Computed Tomography. MICCAI 2016.

Further Extensions

- **Add non-linear de-streaking and de-noising step:**

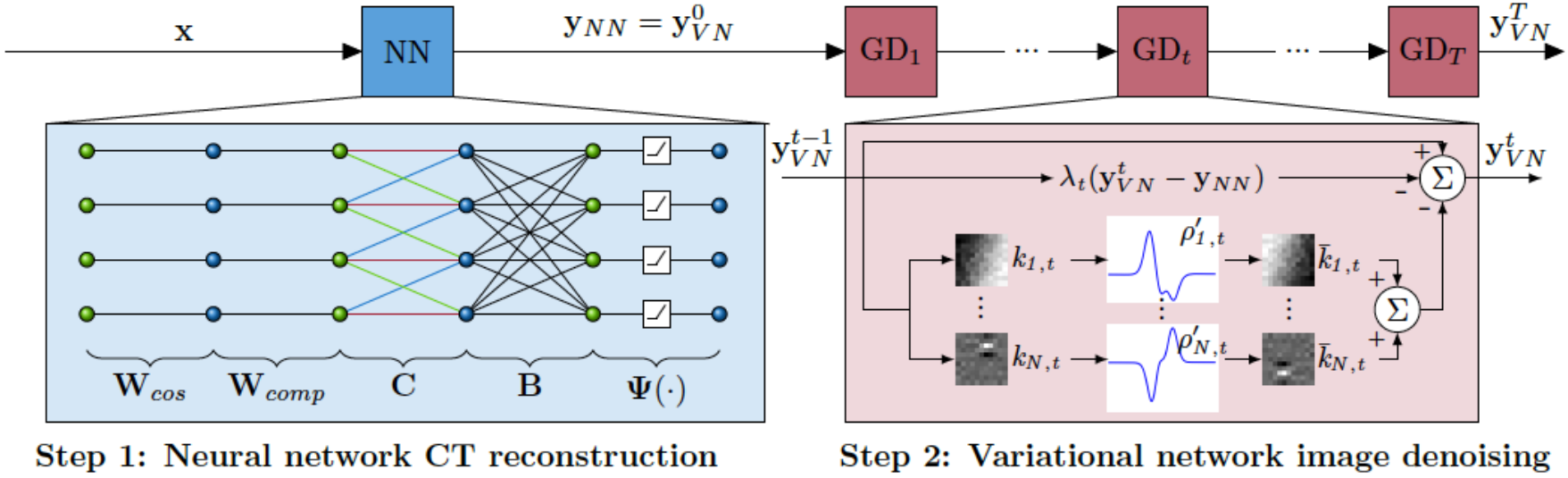
$$E(\boldsymbol{y}) = \frac{\lambda}{2}\|\boldsymbol{y}_{VN} - \boldsymbol{y}_{NN}\|_2^2 + \sum_{i=1}^{N_k} \rho_i(\mathbf{K}_i \boldsymbol{y}_{VN})$$

$$\boldsymbol{y}_{VN}^t = \boldsymbol{y}_{VN}^{t-1} - \sum_{i=1}^{N_k} \mathbf{K}_{i,t}^T \rho_{i,t}'(\mathbf{K}_{i,t}\boldsymbol{y}_{VN}^{t-1}) - \lambda_t(\boldsymbol{y}_{VN}^{t-1} - \boldsymbol{y}_{NN})$$

[7] Hammernik, Kerstin, et al. "A deep learning architecture for limited-angle computed tomography reconstruction." *Bildverarbeitung für die Medizin 2017*. Springer Vieweg, Berlin, Heidelberg, 2017. 92-97.

Further Extensions

- **Add non-linear de-streaking and de-noising step:**

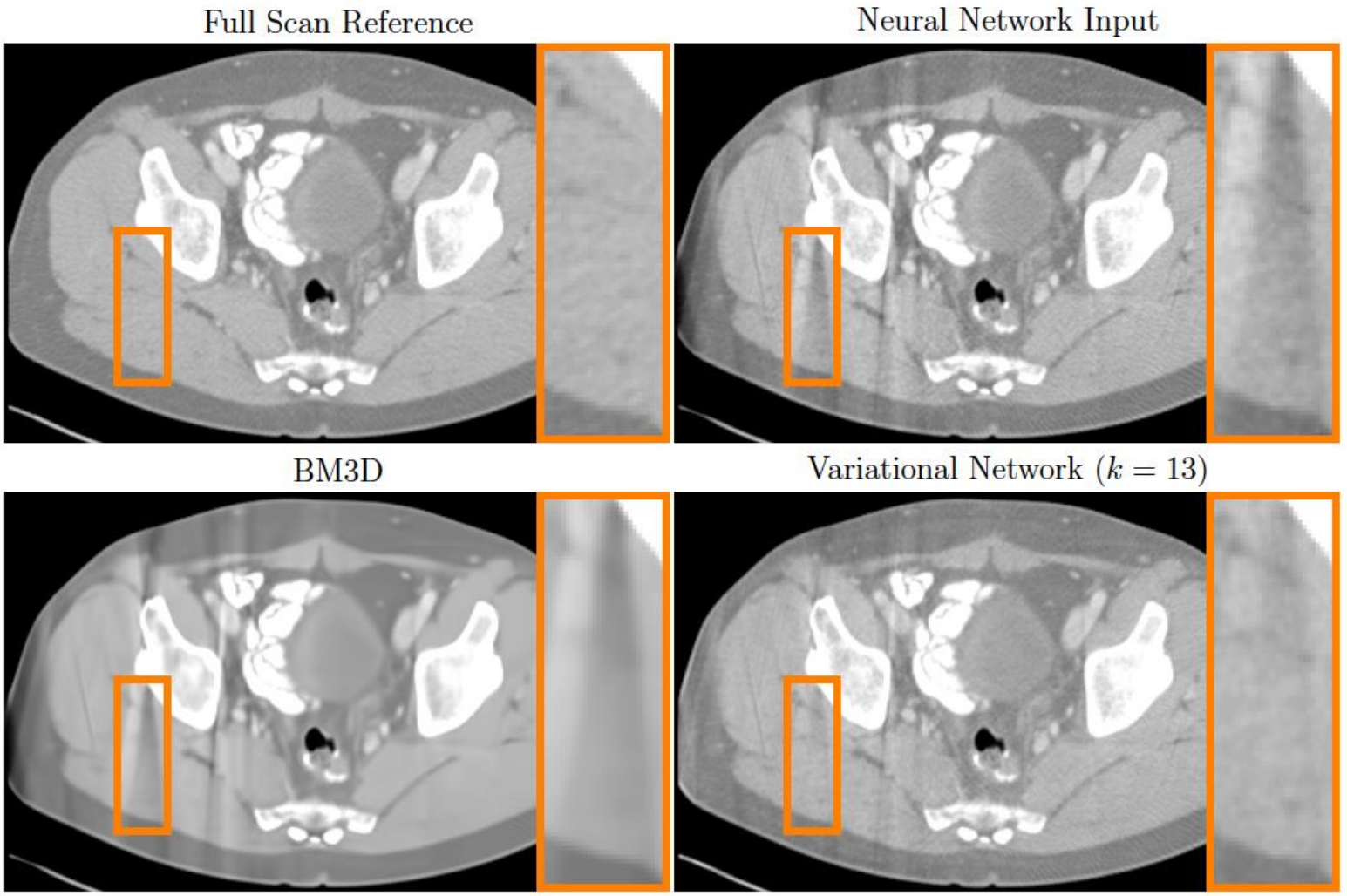


Step 1: Neural network CT reconstruction

Step 2: Variational network image denoising

[7] Hammernik, Kerstin, et al. "A deep learning architecture for limited-angle computed tomography reconstruction." *Bildverarbeitung für die Medizin 2017*. Springer Vieweg, Berlin, Heidelberg, 2017. 92-97.

Further Extensions



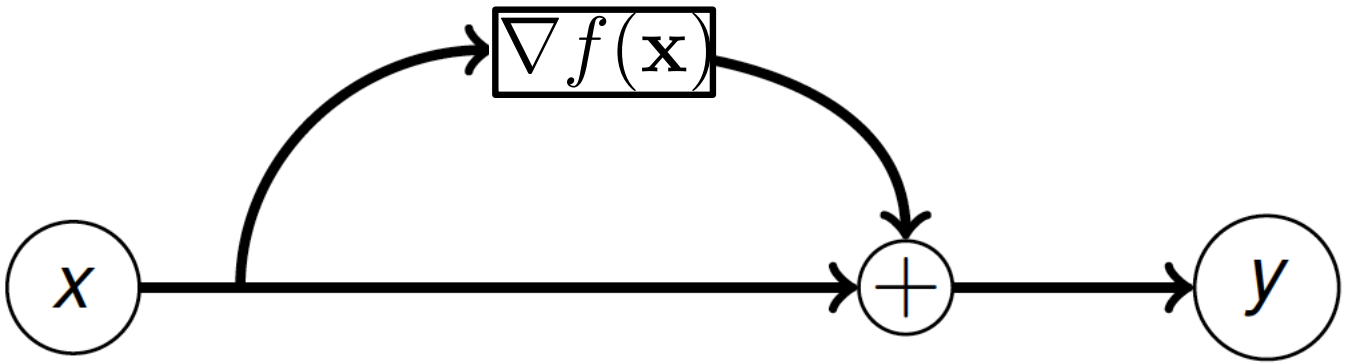
ResNets Revisited

General Function Optimization: Find maxima of

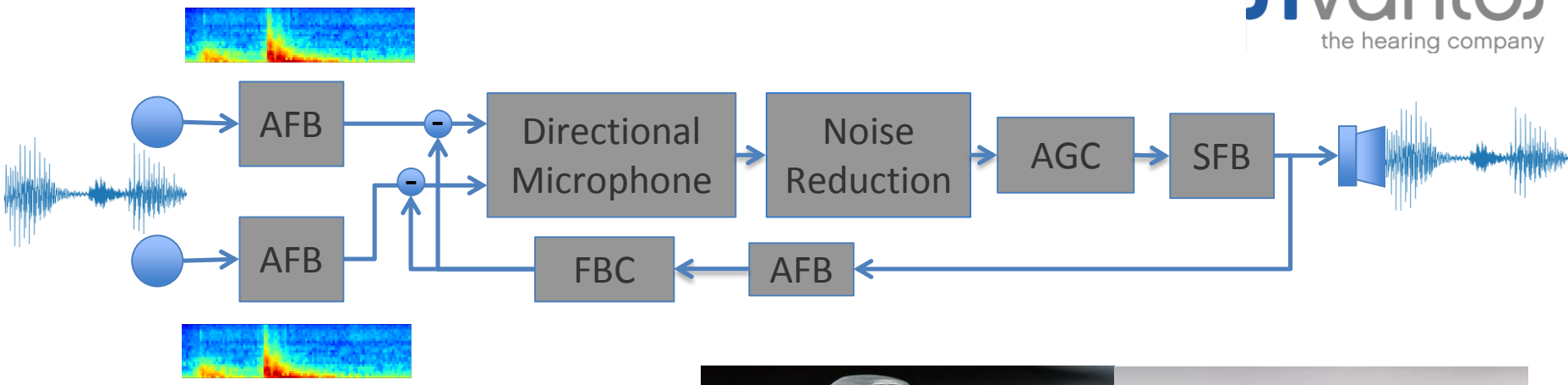
$$f(\mathbf{x})$$

Idea: follow gradient direction

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \nabla f(\mathbf{x}_n)$$



Simplified Modern Hearing Instrument Pipeline

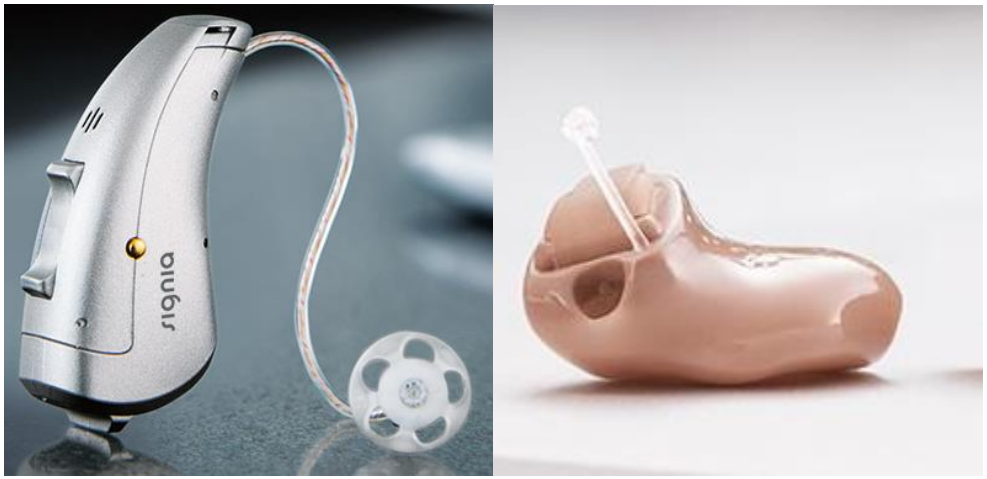


AFB = Analysis Filterbank

AGC = Automatic Gain Control

SFB = Synthesis Filterbank

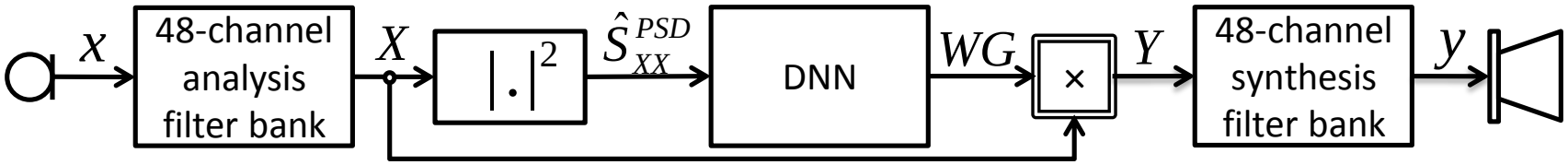
FBC = Feedback Canceler



Images from Carat&Insio (cropped)

[8] Aubreville, Marc, et al. "Deep Denoising for Hearing Aid Applications." 2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC). IEEE, 2018.

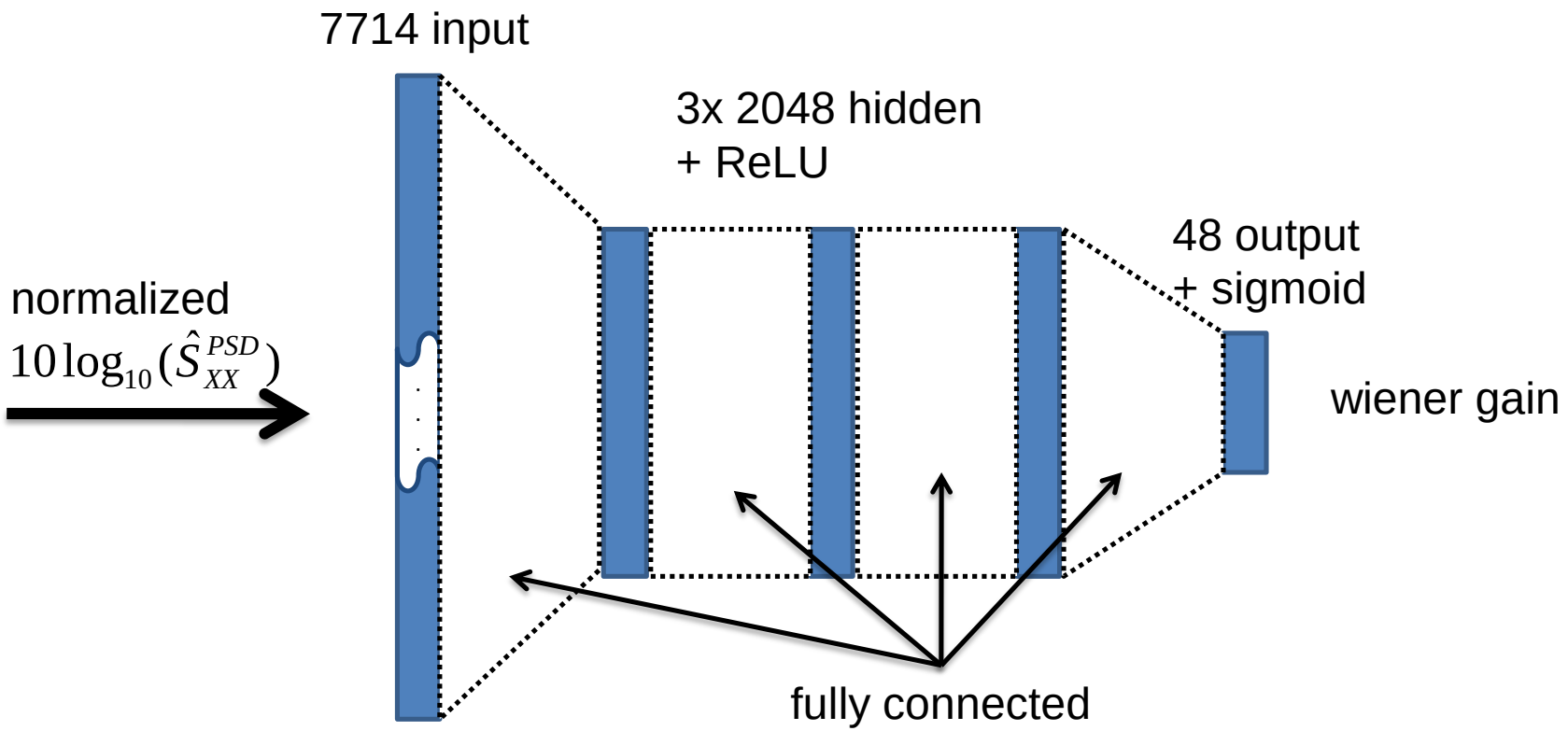
DL Hearing Instrument Pipeline



Dense Neural Network (DNN)

- **Network input:** $10\log_{10}(\hat{S}_{XX}^{PSD})$
  - 200 frames from past
  - current frame
  - 2 frames from future → delay of 2 ms + filter bank delay
  - normalized to zero-mean & unit variance frequency-bin-wise
- 3 hidden layer /w 2048 hidden nodes + ReLU
- Network output:
  - current frame (=48 nodes) + Sigmoid
  - prediction of wiener gain

$$WG = \frac{\hat{S}_{SS}^{PSD}}{\hat{S}_{SS}^{PSD} + \hat{S}_{NN}^{PSD}}$$



Used with permission from Kai Ehrensperger and Marc Aubreville

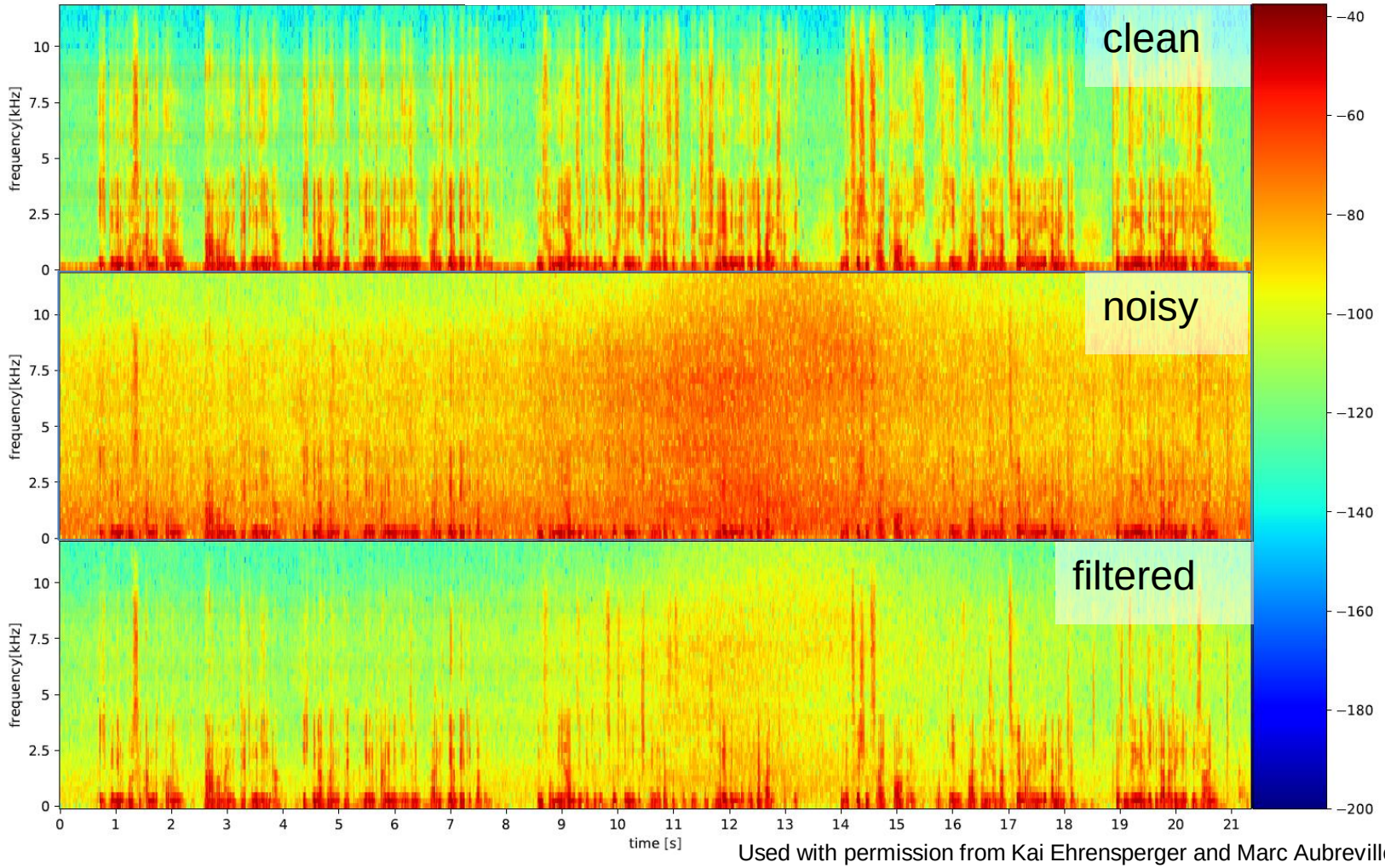
Dataset

Database (fs=24kHz):

- 259 clean speech signals:
  - expected duration: 21 s
- 48 non-stationary noise signals
  - expected duration: 118 s

Results

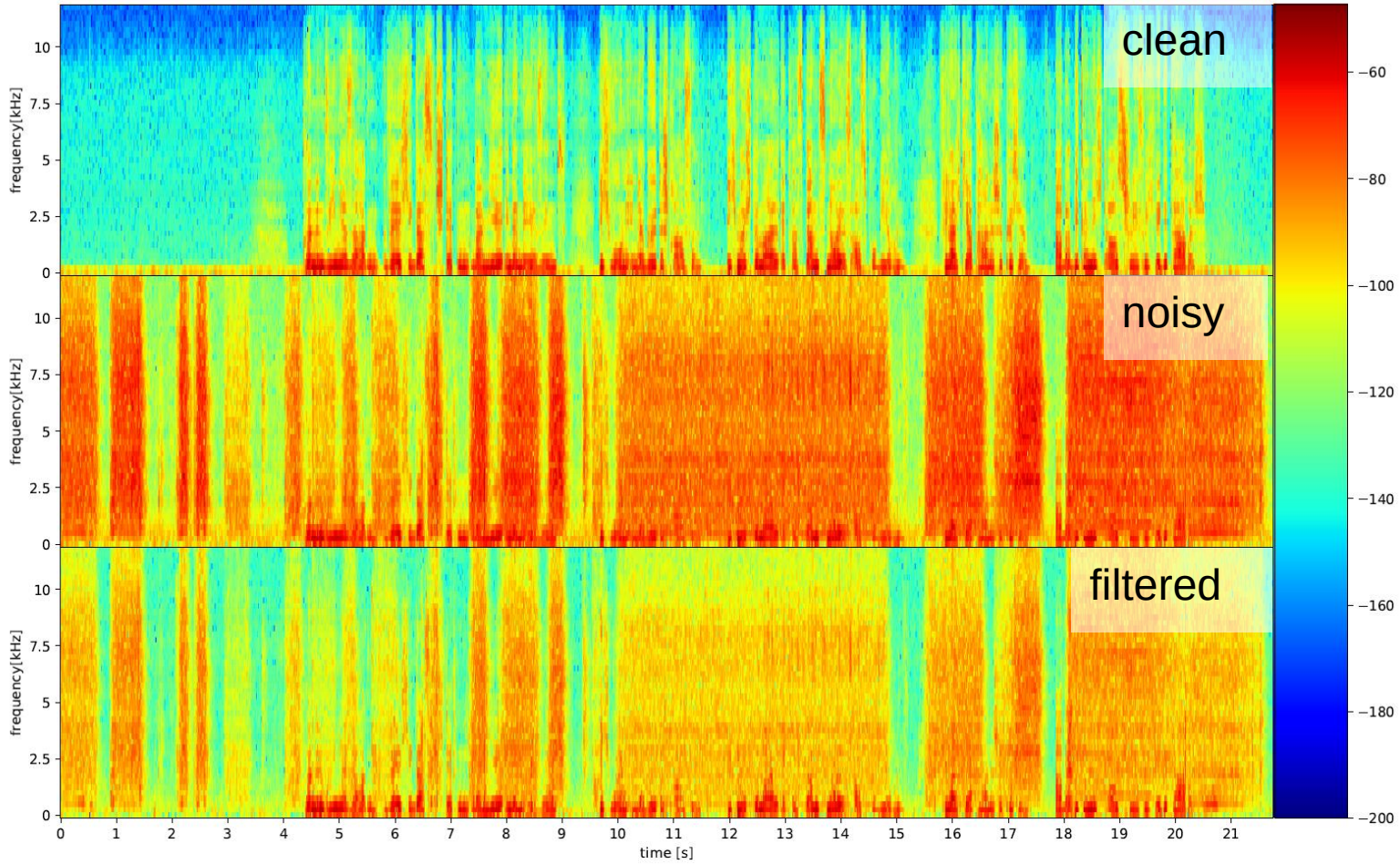
SNR: 10 → 13 dB



Used with permission from Kai Ehrensperger and Marc Aubreville

Results

SNR: -5 → 6.7 dB



[8] Aubreville, Marc, et al. "Deep Denoising for Hearing Aid Applications." 2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC). IEEE, 2018.

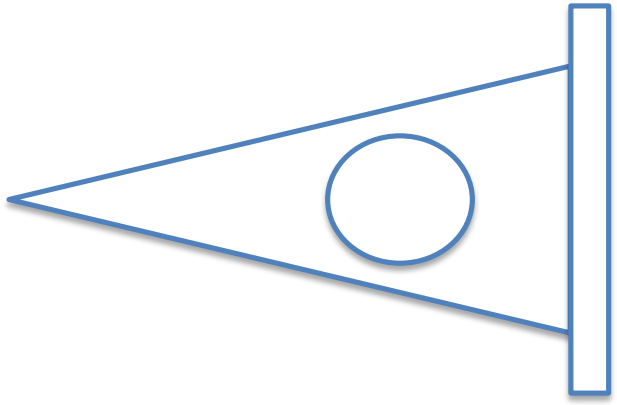
Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. https://arxiv.org/abs/1807.03057

Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$



Cone-beam acquisition

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. https://arxiv.org/abs/1807.03057

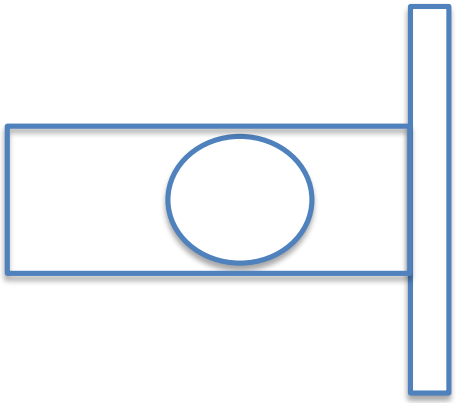
Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$
$$\mathbf{A}_{PB}\mathbf{x} = \mathbf{p}_{PB}$$

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. https://arxiv.org/abs/1807.03057

Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$
$$\mathbf{A}_{PB}\mathbf{x} = \mathbf{p}_{PB}$$

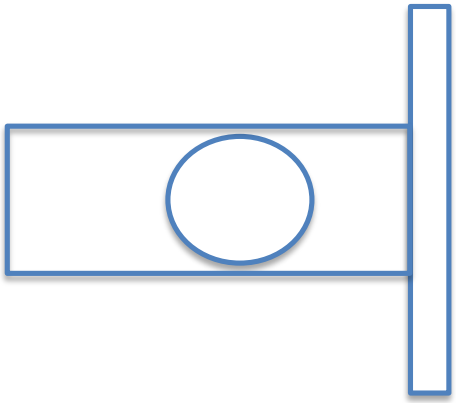


Parallel projection

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. https://arxiv.org/abs/1807.03057

Can we „derive“ networks?

$$\mathbf{A}_{CB}\mathbf{x} = \mathbf{p}_{CB}$$
$$\mathbf{A}_{PB}\mathbf{x} = \mathbf{p}_{PB}$$



Parallel projection
Can’t be measured!

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. https://arxiv.org/abs/1807.03057

Can we „derive“ networks?

$$
\begin{aligned}
\mathbf{A}_{CB}\mathbf{x} &= \mathbf{p}_{CB} \\
\mathbf{A}_{PB}\mathbf{x} &= \mathbf{p}_{PB} \\
x &= \mathbf{A}_{CB}^{\top}(\mathbf{A}_{CB}\mathbf{A}_{CB}^{\top})^{-1}\mathbf{p}_{CB}
\end{aligned}
$$

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. https://arxiv.org/abs/1807.03057

Can we „derive“ networks?

$$
\begin{aligned}
\mathbf{A}_{CB}\mathbf{x} &= \mathbf{p}_{CB} \\
\mathbf{A}_{PB}\mathbf{x} &= \mathbf{p}_{PB} \\
x &= \mathbf{A}_{CB}^{\top}(\mathbf{A}_{CB}\mathbf{A}_{CB}^{\top})^{-1}\mathbf{p}_{CB} \\
\mathbf{p}_{PB} &= \mathbf{A}_{PB}\mathbf{A}_{CB}^{\top}(\mathbf{A}_{CB}\mathbf{A}_{CB}^{\top})^{-1}\mathbf{p}_{CB}
\end{aligned}
$$

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. https://arxiv.org/abs/1807.03057

Can we „derive“ networks?

$$
\begin{aligned}
\mathbf{A}_{CB}\mathbf{x} &= \mathbf{p}_{CB} \\
\mathbf{A}_{PB}\mathbf{x} &= \mathbf{p}_{PB} \\
x &= \mathbf{A}_{CB}^{\top}(\mathbf{A}_{CB}\mathbf{A}_{CB}^{\top})^{-1}\mathbf{p}_{CB} \\
\mathbf{p}_{PB} &= \mathbf{A}_{PB}\mathbf{A}_{CB}^{\top}(\mathbf{A}_{CB}\mathbf{A}_{CB}^{\top})^{-1}\mathbf{p}_{CB} \\
\mathbf{p}_{PB} &= \mathbf{A}_{PB}\mathbf{A}_{CB}^{\top}F^{H}KF\mathbf{p}_{CB}
\end{aligned}
$$

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. https://arxiv.org/abs/1807.03057

Can we „derive“ networks?

$$
\begin{aligned}
\mathbf{A}_{CB}\mathbf{x} &= \mathbf{p}_{CB} \\
\mathbf{A}_{PB}\mathbf{x} &= \mathbf{p}_{PB} \\
x &= \mathbf{A}_{CB}^{\top}(\mathbf{A}_{CB}\mathbf{A}_{CB}^{\top})^{-1}\mathbf{p}_{CB} \\
\mathbf{p}_{PB} &= \mathbf{A}_{PB}\mathbf{A}_{CB}^{\top}(\mathbf{A}_{CB}\mathbf{A}_{CB}^{\top})^{-1}\mathbf{p}_{CB} \\
\mathbf{p}_{PB} &= \mathbf{A}_{PB}\mathbf{A}_{CB}^{\top}F^{H}KF\mathbf{p}_{CB}
\end{aligned}
$$

New Net Topology ?

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. https://arxiv.org/abs/1807.03057

[9] Christopher Syben, Bernhard Stimpel, Jonathan Lommen, Tobias Würfl, Arnd Dörfler, Andreas Maier. Deriving Neural Network Architectures using Precision Learning: Parallel-to-fan beam Conversion. GCPR 2018. https://arxiv.org/abs/1807.03057

Known Operator Learning

- Many traditional approaches mathematically equivalent to neural networks and vice versa
- Learned algorithms are again traditional algorithms

- Side effect: Learned parameters can be interpreted
- Many state-of-the art methods can be integrated
- Methods are efficient and interpretable

NEXT TIME

ON DEEP LEARNING