# Application of trained Deep BCD-Net to iterative low-count PET image reconstruction

Hongki Lim, Jeffrey A. Fessler, Yuni K. Dewaraja, Il Yong Chun

University of Michigan

IEEE NSS/MIC 2018

# Formulation of emission tomography

- Measurement follows Poisson statistical model:

$$Y_i \sim \mathrm{Poisson}(\bar{y}_i(\boldsymbol{x}_{\mathrm{true}})), \quad i = 1, ..., n_d$$

where $\bar{y}_i(\boldsymbol{x}_{\mathrm{true}}) = [\boldsymbol{A}\boldsymbol{x}_{\mathrm{true}}]_i + \bar{r}_i$

- (Negative) Poisson log-likelihood function $f(\boldsymbol{x})$:

$$f(\boldsymbol{x}) \stackrel{c}{=} \sum_{i=1}^{n_d} \bar{y}_i(\boldsymbol{x}) - y_i \log(\bar{y}_i(\boldsymbol{x}))$$

- Goal of conventional emission tomography:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \geq 0$$

# SNR in positron emission tomography (PET)

- Signal to Noise Ratio (SNR) $\propto$ square root of the noise equivalent count rate (NEC)[1]:

$$\text{SNR} = c \cdot \sqrt{\text{NEC}} = c \cdot \left[ \frac{T^2}{(T + S + \gamma R)} \right]^{1/2}$$

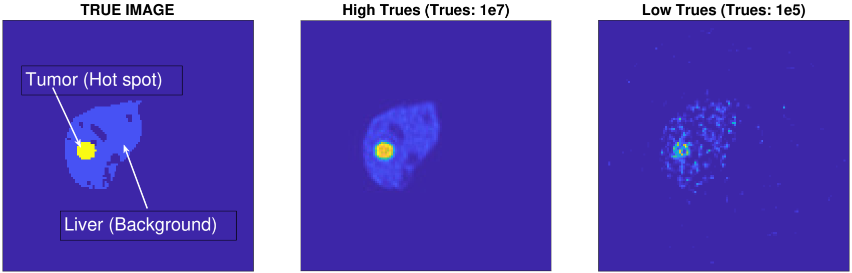
where $c$ is a constant, T is total trues, S and R are total scatters and randoms, and $\gamma$ is 1 or 2 depending on randoms estimation method.

$\rightarrow$ SNR in PET is proportional to trues and disproportionate to randoms and scatters.

[1] Strother, S. C., M. E. Casey, and E. J. Hoffman. "Measuring PET scanner sensitivity: relating count rates to image signal-to-noise ratios using noise equivalents counts." IEEE transactions on nuclear science 37.2 (1990): 783-788.

# Impact of total trues on image quality

Figure: True and estimated images with EM (100 iterations)




- Simulation with XCAT phantom
- Random fraction is high in both cases

# Approaches to low-count imaging

- Post-reconstruction filtering (used in clinic):
  Clinical choice is 3D 5-8mm FWHM Gaussian filter[2].

- Add regularization term to cost function:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \geq \boldsymbol{0}} f(\boldsymbol{x}) + \mathrm{R}(\boldsymbol{x}),$$

where $\mathrm{R}(\boldsymbol{x})$ is a regularization term.

- Two families of $\mathrm{R}(\boldsymbol{x})$:
  1. Mathematically designed regularizer
  2. Learned (trained) regularizer

[2] Carlier, Thomas, et al. "90YPET imaging: Exploring limitations and accuracy under conditions of low counts and high random fraction." Medical physics 42.7 (2015): 4295-4309.

# BCD-Net problem formulation

- BCD-Net[3] is inspired by following sparsity-based regularization with trained convolutional filters:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \min_{\boldsymbol{z}} f(\boldsymbol{x}) + \mathrm{R}(\boldsymbol{x}, \boldsymbol{z}) = \arg\min_{\boldsymbol{x}} \min_{\boldsymbol{z}} f(\boldsymbol{x}) + \beta\Big(\sum_{k=1}^{K} ||\boldsymbol{c}_k * \boldsymbol{x} - \boldsymbol{z}_k||_2^2 + \alpha_k ||\boldsymbol{z}_k||_1\Big),$$

- Block Coordinate Descent (BCD) algorithm alternatively updates $\{\boldsymbol{z}_k : \boldsymbol{z}_1, ..., \boldsymbol{z}_K\}$ and $\boldsymbol{x}$ :

$$\{\boldsymbol{z}_k^{(n+1)}\} = \arg\min_{\{\boldsymbol{z}_k\}} ||\boldsymbol{c}_k * \boldsymbol{x}^{(n)} - \boldsymbol{z}_k||_2^2 + \alpha_k ||\boldsymbol{z}_k||_1 = \mathcal{T}(\boldsymbol{c}_k * \boldsymbol{x}^{(n)}, \alpha_k)$$

$$\boldsymbol{x}^{(n+1)} = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \beta\Big(\sum_{k=1}^{K} ||\boldsymbol{c}_k * \boldsymbol{x} - \boldsymbol{z}_k^{(n+1)}||_2^2\Big)$$

where $\mathcal{T}(\cdot,\cdot)$ is the element-wise soft thresholding operator:

$$[\mathcal{T}(\boldsymbol{t}, q)]_j = \begin{cases} t_j - q \cdot \mathrm{sign}(t_j), & |t_j| > q \\ 0, & |t_j| \le q \end{cases}.$$

[3] Chun & Fessler. (2018). "Deep BCD-Net Using Identical Encoding-Decoding CNN Structures for Iterative Image Recovery." 1-5. 10.1109/IVMSPW.2018.8448694.

# BCD-Net problem formulation

- Previous formulation becomes following equivalent updates with one condition:

$$\{\mathbf{z}_k^{(n+1)}\} = \mathcal{T}(\mathbf{c}_k * \mathbf{x}^{(n)}, \alpha_k)$$

$$\mathbf{x}^{(n+1)} = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \beta\Big(\sum_{k=1}^{K} ||\mathbf{c}_k * \mathbf{x} - \mathbf{z}_k^{(n+1)}||_2^2\Big)$$

$$\Updownarrow \quad \sum_{k=1}^{K} \boldsymbol{C_k}^T \boldsymbol{C_k} = \boldsymbol{I}$$

$$\mathbf{u}^{(n+1)} = \sum_{k=1}^{K} \tilde{\mathbf{c}}_k * \left(\mathcal{T}(\mathbf{c}_k * \mathbf{x}^{(n)}, \alpha_k)\right)$$

$$\mathbf{x}^{(n+1)} = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \beta||\mathbf{x} - \mathbf{u}^{(n+1)}||_2^2,$$

where $\boldsymbol{C}_k\mathbf{x} = \mathbf{c}_k * \mathbf{x}$ and $\tilde{\mathbf{c}}_k$ is flipped version of $\mathbf{c}_k$.

# BCD-Net problem formulation

- $K$ set of $\{\boldsymbol{c}_k\}, \{\alpha_k\}$ are trained at each iteration to map a noisy image into high quality (true, if possible) image:

$$\{\hat{\boldsymbol{c}}_1^{(n+1)}, \ldots, \hat{\boldsymbol{c}}_K^{(n+1)}\}, \{\hat{\alpha}_1^{(n+1)}, \ldots, \hat{\alpha}_K^{(n+1)}\} = \underset{\{\boldsymbol{c}_k\},\{\alpha_k\}}{\arg\min} \frac{1}{J_{\text{FOV}}} \left|\left| \boldsymbol{x}_{\text{true}} - \sum_{k=1}^{K} \tilde{\boldsymbol{c}}_k * \left( \mathcal{T}(\boldsymbol{c}_k * \boldsymbol{x}^{(n)}, \alpha_k) \right) \right|\right|_2^2.$$

- In this work, we train separate decoding convolutional filters $\{\boldsymbol{d}_k\}$ instead of using $\{\tilde{\boldsymbol{c}}_k\}$
- With trained convolutional filters and soft-thresholding value, each variable update will be:

$$\boldsymbol{u}^{(n+1)} = \sum_{k=1}^{K} \boldsymbol{d}_k^{(n+1)} * \left( \mathcal{T}(\boldsymbol{c}_k^{(n+1)} * \boldsymbol{x}^{(n)}, \alpha_k^{(n+1)}) \right)$$

$$\boldsymbol{x}^{(n+1)} = \underset{\boldsymbol{x}}{\arg\min}\, f(\boldsymbol{x}) + \beta ||\boldsymbol{x} - \boldsymbol{u}^{(n+1)}||_2^2.$$

# Details on $x$-update

- To solve following update image step:

$$\boldsymbol{x}^{(n+1)} = \underset{\boldsymbol{x} \geq \boldsymbol{0}}{\arg\min}\, f(\boldsymbol{x}) + \beta ||\boldsymbol{x} - \boldsymbol{u}^{(n+1)}||_2^2,$$

- we use EM-surrogate for $f(\boldsymbol{x})$:

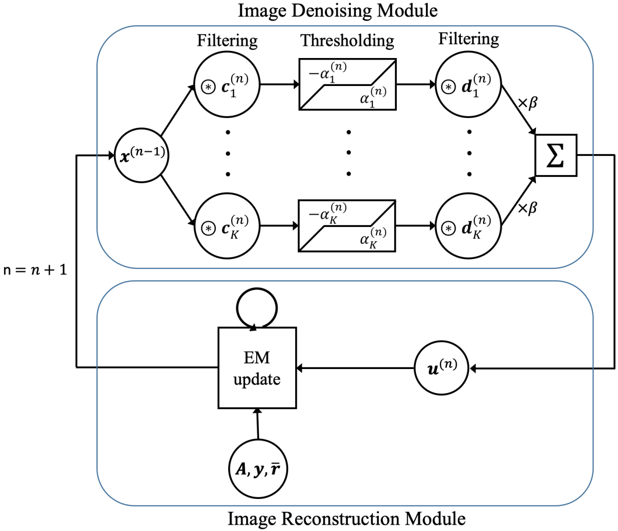
$$\begin{aligned} f(\boldsymbol{x}) + \beta ||\boldsymbol{x} - \boldsymbol{u}^{(n+1)}||_2^2 &= \sum_i [\boldsymbol{A}\boldsymbol{x}]_i + \bar{r}_i - y_i \log([\boldsymbol{A}\boldsymbol{x}]_i + \bar{r}_i) + \beta \sum_j (x_j - u_j^{(n+1)})^2 \\ &\leq \sum_j a_j x_j - e_j(\boldsymbol{x}^{(n')})(x_j^{(n')}) \log(x_j) + \beta (x_j - u_j^{(n+1)})^2 \\ &= \sum_j Q_j(x_j), \end{aligned}$$

where $e_j(\boldsymbol{x}^{(n')}) = \sum_i a_{ij} \frac{y_i}{\bar{y}_i(\boldsymbol{x}^{(n')})}$ and $n'$ is $n'$th iteration in $x$-update.

- Equating $\frac{\partial Q_j(x_j)}{\partial x_j}$ to zero is equivalent to finding the root of the following formula:

$$2\beta x_j^2 + (a_j - 2\beta z_j^{(n+1)}) x_j - e_j(\boldsymbol{x}^{(n')})(x_j^{(n')}) = 0.$$

# Architecture of BCD-Net



# Conventional non-trained regularizers

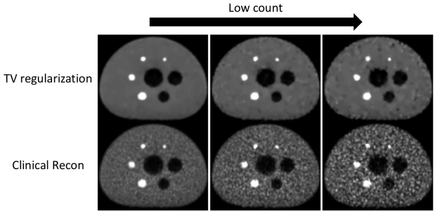
- Total-variation (TV) regularization
- Non-local means (NLM) regularization

# TV regularization

- TV regularization solves following formulation:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \geq \boldsymbol{0}}{\arg\min}\, f(\boldsymbol{x}) + \mathrm{R}(\boldsymbol{x}) = \underset{\boldsymbol{x} \geq \boldsymbol{0}}{\arg\min}\, f(\boldsymbol{x}) + \beta \|\boldsymbol{C}\boldsymbol{x}\|_1 .$$

- Recent work[4] demonstrated that TV regularized reconstruction gives better qualitative & quantitative result than clinical reconstruction for low-count data sets (w.r.t. contrast recovery, variability)




[4] Zhang, Zheng, et al. "Optimization-Based Image Reconstruction from Low-Count, List-Mode TOF-PET Data." IEEE Transactions on Biomedical Engineering (2018).

# Algorithm for TV regularization

- Primal-Dual Hybrid Gradient (PDHG)[5] solves TV-constrained problem by reformulating the primal problem as the saddle point optimization problem. Then PDHG approaches to the saddle point by using proximal mapping of each function:

$$\min_{x}\{F(\boldsymbol{Kx}) + G(\boldsymbol{x})\}$$

$$\Updownarrow \boldsymbol{y} = \boldsymbol{Kx}$$

$$\min_{x}\max_{y}\{\langle \boldsymbol{Kx}, \boldsymbol{y}\rangle + G(\boldsymbol{x}) - F^*(\boldsymbol{y})\}$$

$$\Downarrow \text{solve via proximal-point method}$$

$$\boldsymbol{y}^{(n+1)} = prox_{\sigma}[F^*](\boldsymbol{y}^{(n)} + \sigma \boldsymbol{K}\bar{\boldsymbol{x}}^{(n)})$$

$$\boldsymbol{x}^{(n+1)} = prox_{\tau}[G](\boldsymbol{x}^{(n)} - \tau \boldsymbol{K}^T \boldsymbol{y}^{(n+1)})$$

$$\bar{\boldsymbol{x}}^{(n+1)} = \boldsymbol{x}^{(n+1)} + \theta(\boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{(n)})$$

[5] Chambolle, Antonin, and Thomas Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging." Journal of mathematical imaging and vision 40.1 (2011): 120-145.

# Conventional non-trained regularizers

- Total-variation (TV) regularization
- Non-local means (NLM) regularization

# NLM regularization

- NLM[6] regularization[7] solves following formulation:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \geq \boldsymbol{0}}{\arg\min}\, f(\boldsymbol{x}) + \mathrm{R}(\boldsymbol{x}) = \underset{\boldsymbol{x} \geq \boldsymbol{0}}{\arg\min}\, f(\boldsymbol{x}) + \beta \sum_{i,j \in S_i} p(||\boldsymbol{N}_i \boldsymbol{x} - \boldsymbol{N}_j \boldsymbol{x}||_2^2).$$

where $p(t)$ is a potential function of a scalar variable $t$,
$S_i$ is the search neighborhood around the $i$th voxel, and
$\boldsymbol{N}_i \boldsymbol{x}$ is a vector of image intensities of all voxels within a fixed distance from the $i$th voxel.

[6] Buades, A., Coll, B., Morel, J. M. (2005). A review of image denoising algorithms, with a new one. Multiscale Modeling Simulation, 4(2), 490-530.

[7] Lou, Yifei, et al. "Image recovery via nonlocal operators." Journal of Scientific Computing 42.2 (2010): 185-197.

## Algorithm for NLM regularization

- For the acceleration of convergence, we use variable splitting method and solve the problem via ADMM[8] with adaptive penalty parameter ($\rho$) selection method[9]:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\geq \boldsymbol{0}} f(\boldsymbol{x}) + \mathrm{R}(\boldsymbol{x})$$

$$\Updownarrow$$

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\geq \boldsymbol{0}} \min_{\boldsymbol{z}} \mathbb{1}^T(\boldsymbol{A}\boldsymbol{z} + \bar{\boldsymbol{r}}) - \boldsymbol{y}^T \log(\boldsymbol{A}\boldsymbol{z} + \bar{\boldsymbol{r}}) + \mathrm{R}(\boldsymbol{x}), \quad \text{s.t. } \boldsymbol{z} = \boldsymbol{x}$$

$$\Downarrow \text{solve via ADMM}$$

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\geq \boldsymbol{0}} \min_{\boldsymbol{z}} \max_{\boldsymbol{u}} \mathbb{1}^T(\boldsymbol{A}\boldsymbol{z} + \bar{\boldsymbol{r}}) - \boldsymbol{y}^T \log(\boldsymbol{A}\boldsymbol{z} + \bar{\boldsymbol{r}}) + \mathrm{R}(\boldsymbol{x}) + \frac{\rho}{2}||\boldsymbol{x} - \boldsymbol{z} + \boldsymbol{u}||_2^2 - \frac{\rho}{2}||\boldsymbol{u}||_2^2$$

[8] Chun, Se Young et al. "ADMM for tomography with nonlocal regularizers." IEEE TMI 33.10 (2014): 1960-1968.

[9] Boyd, Stephen, et al. "Distributed optimization and statistical learning via the ADMM." Foundations and Trends in Machine learning 3.1 (2011): 1-122.

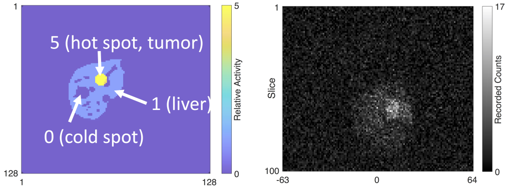
# Experimental setting

- Simulating Y-90 Radioembolization

| | Patient A | Patient B |
|---|---|---|
| Y-90 Injection (GBq) | 3.9 | 0.9 |
| True Prompts | 675K | 97K |
| Random Prompts | 3.3M | 1.7M |
| Total Prompts | 3.9M | 1.8M |
| Random Fraction (%) | 83 | 95 |

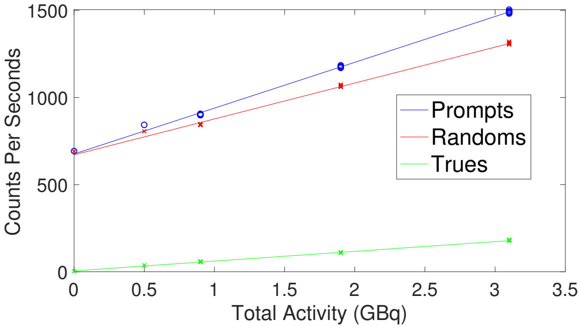
* Random Fraction = (Random prompts / Total prompts) $\times$ 100

- Digital phantom and projection



# Particulars about Y-90 imaging

- Y-90: Radioisotope for radioembolization
  - ▷ Almost pure beta emitter
  - ▷ Very low probability of positron emission : $3.2 \times 10^{-5}$
  - ▷ $^{176}$Lu and Bremsstrahlung photons contribute to random coincidences
  - $\rightarrow$ Low true coincidence counts & very high random fraction



# BCD-Net training details

- $\mathbf{x}_{\text{Train}}$: 6 set of 3-D XCAT (128×128×100) with a voxel size 4×4×4 (mm$^3$),
  1:5 activity ratio between healthy liver and lesion.
- $\mathbf{x}^{(0)}$: An image estimated with EM algorithm using 20 iterations
- 10 layers ($n = 10$)
- 3D Filters and soft-thresholding values are trained using back-propagation algorithm:
  - Pytorch deep-learning library
  - ADAM optimization with epoch number: 200, mini-batch size: 1 (128×128×100×1)
  - Learning rate: Encoding(1e-3), Decoding(1e-3), Thresholding(1e-1)
  - Learning rate decay method is used (lr = lr × 0.9 every 10 epoch)
  - Initialization of thresholding value needs to be carefully chosen
    → ($n_p \times 0.1$)-th largest value of sorted $x^{(0)}$, where $n_p$ is number of voxels in image $x$
  - Filter size: 3×3×3, Filter number (K): 200
- Testing 1 set of 3-D XCAT (128×128×100): changed location of lesion and cold spot.

## Evaluation metrics

- Activity Recovery (AR):

$$\text{AR} = \frac{\text{Estimated } C_{VOI}}{\text{True } C_{VOI}} \times 100(\%),$$

where $C_{VOI}$ is mean counts in the volume of interest (VOI)

- Contrast to Noise Ratio (CNR):

$$\text{CNR} = \frac{C_{hotspot} - C_{bkg}}{STD_{bkg}}.$$

- Root Mean Square Error (RMSE):

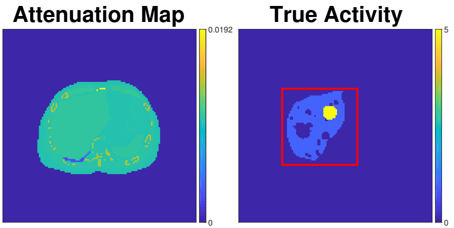
$$\text{RMSE} = \sqrt{\frac{\sum_j (\boldsymbol{x}_{true}[j] - \hat{\boldsymbol{x}}[j])^2}{J_{FOV}}} \times 100(\%),$$

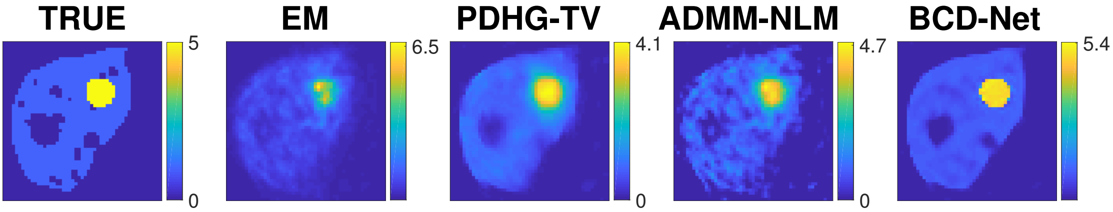
where $J_{FOV}$ is the total number of voxels in field of view (FOV).

# Choosing regularization parameter for each method

- $\beta$ value test range:
  - ▷ TV regularization: $2^{-4}$ - $2^{4}$
  - ▷ NLM regularization: $2^{-20}$ - $2^{-7}$
  - ▷ BCD-Net regularization: $2^{-4}$ - $2^{4}$
- Parameter value to obtain the highest contrast to noise ratio (CNR) and lowest RMSE:
  - ▷ $\beta_{\text{TV}} = 2^{-2}$
  - ▷ $\beta_{\text{NLM}} = 2^{-10}$
  - ▷ $\beta_{\text{BCD-Net}} = 2^{2}$

# Reconstructed images



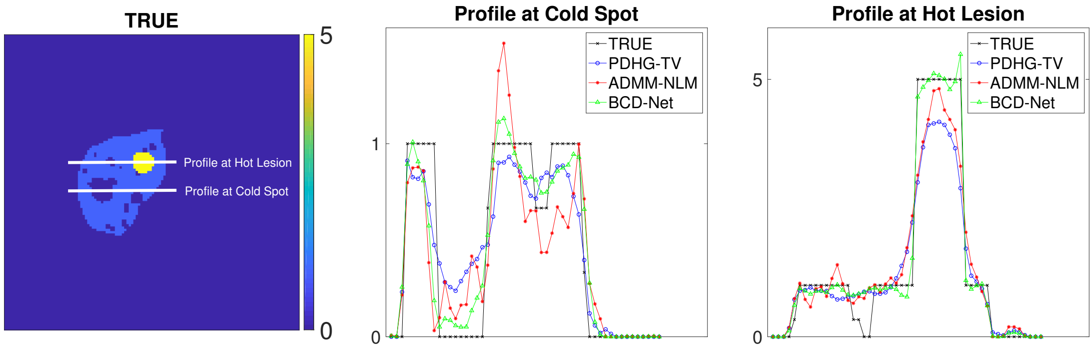



# Quantitative evaluation results

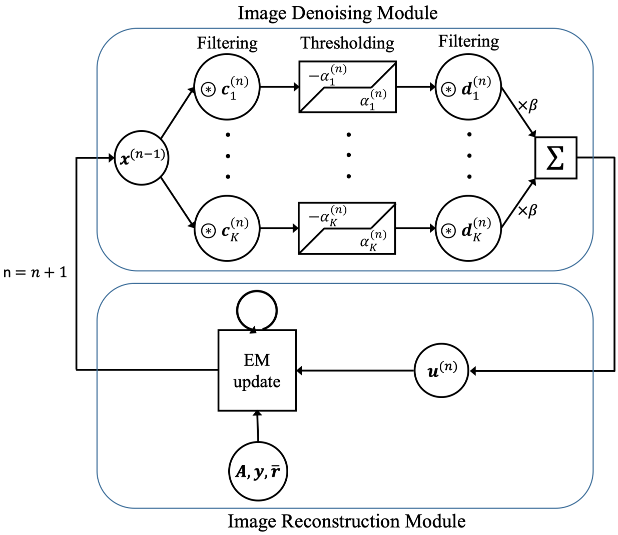
Table: Evaluation results on regularizers with $\beta$ value obtaining highest CNR and lowest RMSE

| | Iteration # | Time (Sec) | AR-Hot Lesion | AR-Liver | CNR | RMSE |
|---|---|---|---|---|---|---|
| EM | 50 | 46 | 89.4 | 86.9 | 5.0 | 12.9 |
| PDHG-TV | 200 | 209 | 68.2 | 86.0 | 8.9 | 7.7 |
| ADMM-NLM | 200 | 2,907 | 71.0 | 84.7 | 7.0 | 9.2 |
| BCD-NET | 200 (20 $\times$ 10) | 233 | **96.5** | **88.8** | **17.5** | **5.9** |

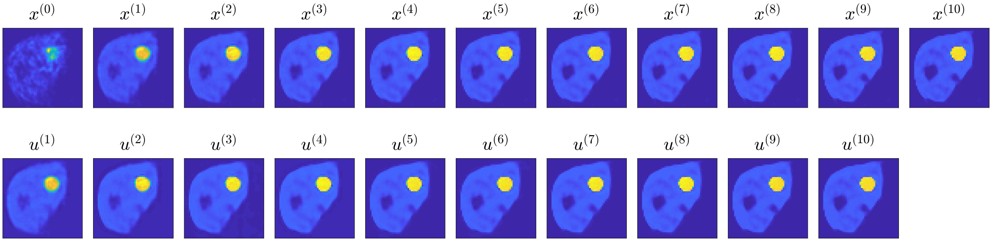
# Profile comparison

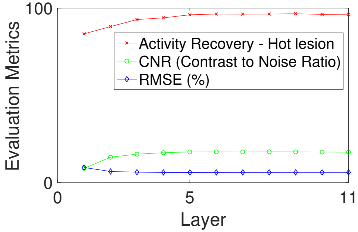


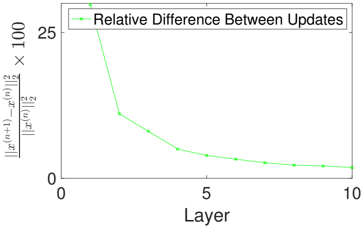
# BCD-Net notation reminder



# BCD-Net: images and evaluation results at each layer



## BCD-Net: comparison between single image denoising network

- Many related works are based on the framework composed of single image denoising (deep) network (e.g., U-Net) as a post-reconstruction processing, however, this single denoising framework has a potential not to fully recover the image (+ risk of over-fitting).

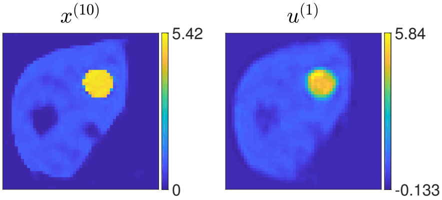
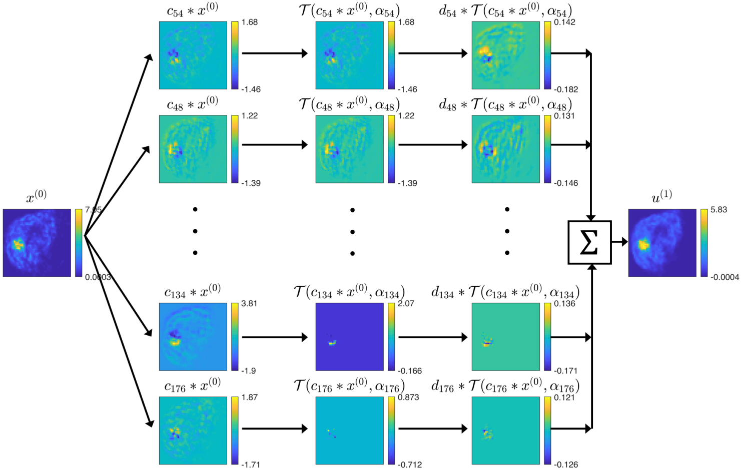



Table: Evaluation results on $\boldsymbol{x}^{(10)}$ and $\boldsymbol{u}^{(1)}$

| | AR-Hot Lesion | AR-Liver | CNR | RMSE |
|---|---|---|---|---|
| $\boldsymbol{x}^{(10)}$ | **96.5** | **88.8** | **17.5** | **5.9** |
| $\boldsymbol{u}^{(1)}$ | 89.4 | 86.3 | 15.0 | 6.3 |

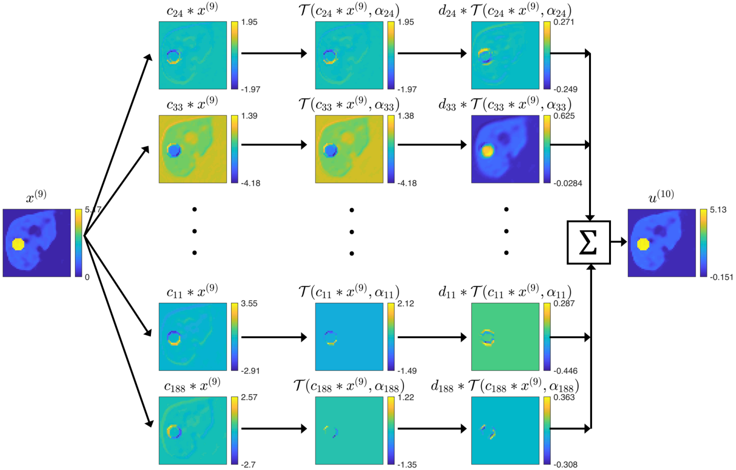
# BCD-Net: convolutional filters and thresholding analysis

- Visualization of steps in $\boldsymbol{d}_k^{(1)} * \mathcal{T}(\boldsymbol{c}_k^{(1)} * \boldsymbol{x}^{(0)}, \alpha_k^{(1)})$ with ascending order of thresholdings



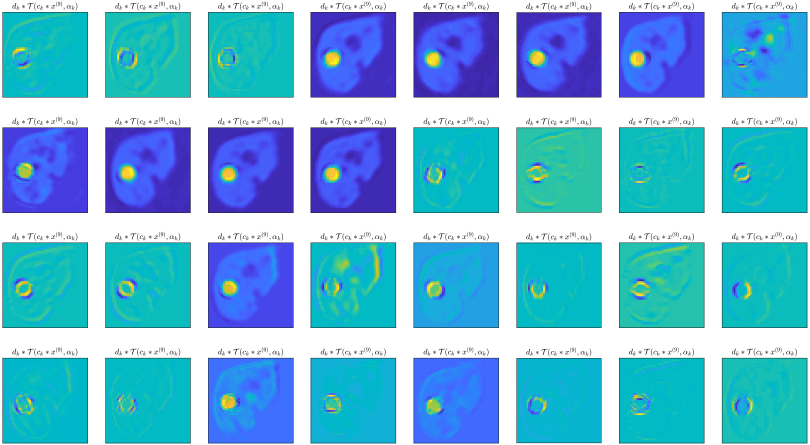
# BCD-Net: convolutional filters and thresholding analysis

- Visualization of steps in $\boldsymbol{d}_k^{(10)} * \mathcal{T}(\boldsymbol{c}_k^{(10)} * \boldsymbol{x}^{(9)}, \alpha_k^{(10)})$ with ascending order of thresholdings



# BCD-Net: convolutional filters and thresholding analysis

- Visualization of $\boldsymbol{d}_k^{(10)} * \mathcal{T}(\boldsymbol{c}_k^{(10)} * \boldsymbol{x}^{(9)}, \alpha_k^{(10)})$ with ascending order of thresholdings



## Summary, Future work & Acknowledgement

- Non-trained regularizers had a trade-off between noise and recovery accuracy, whereas BCD-Net improved activity recovery for a hot sphere and reduced noise at the same time.
- BCD-Net improved CNR and activity recovery by 96.6% (150.0%) and 41.5% (35.9%) compared to PDHG-TV (ADMM-NLM) regularized reconstruction.

- Robustness to noise-level (count-level).
- Adaptive regularization parameter selection.
- Train and test on more diverse data including measured data.
- Thorough comparison between single denoising network framework.

- We acknowledge Se Young Chun (UNIST) for providing NLM regularization codes.
- We acknowledge Zhengyu Huang (UMich) for providing Pytorch codes to train the image denoising network.
- This work is supported by NIH-NIBIB grant R01EB022075

# Thank You

Slides will be available at anytime here:
https://limhongki.github.io