HHS Public Access

Author manuscript

Phys Med Biol. Author manuscript; available in PMC 2022 September 20.

Published in final edited form as:

Phys Med Biol. ; 66(19): . doi:10.1088/1361-6560/ac0f9a.

Patient-specific hyperparameter learning for optimization-based CT image reconstruction

Jingyan Xu[1], **Frederic Noo**[2]

[1] Department of Radiology, Johns Hopkins University

[2] Department of Radiology and Imaging Sciences, University of Utah

Abstract

We propose a hyperparameter learning framework that learns *patient-specific* hyperparameters for optimization-based image reconstruction problems for x-ray CT applications. The framework consists of two functional modules: (1) a hyperparameter learning module parameterized by a convolutional neural network, (2) an image reconstruction module that takes as inputs both the noisy sinogram and the hyperparameters from (1) and generates the reconstructed images. As a proof-of-concept study, in this work we focus on a subclass of optimization-based image reconstruction problems with exactly computable solutions so that the whole network can be trained end-to-end in an efficient manner. Unlike existing hyperparameter learning methods, our proposed framework generates patient-specific hyperparameters from the sinogram of the same patient. Numerical studies demonstrate the effectiveness of our proposed approach compared to bi-level optimization.

Keywords

bi-level optimization; dynamic programming; hyperparameter learning; low dose CT; sinogram smoothing

1. Introduction

Model-based image reconstruction (MBIR) problems are often formulated as an optimization problem, where an objective function consisting of a data fitting term and a regularizer is to be minimized, and the minimizer is the image to be reconstructed. The effectiveness of MBIR methods, to a certain extent, depends on a judicious choice of the hyperparameters in the objective function and/or the regularizer. Manual sweeping (grid search) of hyperparameters is sometimes employed when there are a small number (2 or 3) of them. On the other hand, the hyperparameters can be fully spatially variant. Specifying such a large number of hyperparameters by grid search is infeasible. A more automatic and efficient hyperparameter learning framework is needed.

One approach to hyperparameter learning is bi-level optimization [1,2], where a lower level optimization (*e.g.*, MBIR) finds the minimizer for any candidate set of hyperparameters;

jxu@jhmi.edu .

while an upper level optimization tries to determine the optimal hyperparameter by comparing the minimizers with the training labels. The output of bi-level optimization is one set of optimized hyperparameters to be used for all test cases. The effectiveness of bi-level optimization highly depends on the similarity between the training data and the test data.

Leveraging the feature representation power of deep neural networks (DNN) and convolutional neural networks (CNN), in this work we propose a CNN-based hyperparameter learning framework that learns a parameterization map between the sinogram, *i.e.*, CT projection data, and the desirable hyperparameters; at inference time the learned parameterization map can generate *patient-specific* hyperparameters from an individual patient's sinogram in an efficient manner.

Our learning framework consists of two functional modules (Fig. 1). The CNN (Module 1) tries to capture the intrinsic relationship between the sinogram and the hyperparameters for high quality image reconstruction; image reconstruction itself is performed by Module 2. Only Module 1 contains trainable parameters, which are trained in an end-to-end, supervised manner using ground truth labels such as noise-free or full dose images. At inference time, the two modules can be detached and work separately: Module 1 generates case-specific hyperparameters for each test case; the hyperparameters can be used by the reconstruction algorithm that runs independently, *i.e.*, outside of a deep learning (DL) library.

Deep learning with end-to-end training of an architecture like Fig. 1 requires backpropagating the loss gradient through the image reconstruction module. As a first proof of concept study, in the remainder of the paper we specialize Fig. 1 by focusing on a subclass of optimization-based image reconstruction methods whose solution can be computed exactly. This class of reconstruction methods proceeds in two steps: (1) optimization-based sinogram smoothing, and (2) filtered-backprojection (FBP) reconstruction using the smoothed sinogram. The main purpose for this specialization is to demonstrate the working principle of the proposed learning framework. Effective end-to-end training of the fully general architecture (Fig. 1) is discussed in Section 5.

2. Method

Sinogram smoothing [3, 4] is often used as a preprocessing step to reduce noise in a CT sinogram before image reconstruction. Similar to MBIR problems, optimization-based sinogram smoothing employs an objective function consisting of a data-fitting term and a regularizer that incorporates *a priori* information. The regularizer has hyperparameters that need to be specified or learned.

For sinogram smoothing problems, Module 2 of Fig. 1 specializes to the two submodules of Fig. 2: a sinogram smoothing module and an FBP module. The overall learning framework remains the same. Once the network is trained, Module 1 generates the desirable hyperparameters using a patient's own sinogram. The hyperparameters are then passed to the sinogram smoothing module to generate the smoothed sinogram, which is then reconstructed.

To enable end-to-end training, the loss gradient will need to be backpropagated through (a) the FBP module, (b) the sinogram smoothing module, (c) the CNN module, in order to update the weights in the CNN. In this work, we focus on view-by-view based sinogram smoothing problems that can be solved exactly using dynamic programming (DP); doing this using a DL library's constructs allows gradient backpropagation with the DL's automatic differentiation capability,† and the entire network can be trained in an efficient, end-to-end manner. Below we discuss in more detail each of three modules in Fig. 2 and the supervised training strategy.

Notation. We denote by $\{\boldsymbol{y}^s, \bar{\boldsymbol{x}}^s\}, s = 1, \cdots, N$, a training set of N samples with paired noisy sinograms $\boldsymbol{y}^s$ and the ground truth $\bar{x}^s$ images. Each sinogram $\boldsymbol{y}^s \in R^{m,n}$ consists of m view angles and n detector bins. We sometimes omit the sample index and use $\boldsymbol{y} = \{y_{v,i}\} \in R^{m,n}$ to denote a sample sinogram, with view index v, and detector bin index i.

2.1. *Module 1:* CNN$\boldsymbol{\theta}$($\boldsymbol{y}$)

Our CNN module works on one projection view at a time. This is also the only module that contains trainable parameters ($\boldsymbol{\theta}$) in our architecture. One view of the sinogram $\boldsymbol{y}_v$ is a 1-D vector of n measurements (typically $n = 600 - 1000$). As shown in Fig. 3, we use a 1-D variant of the convolutional DenseNet [5] with dense connections for the mapping between a projection view $\boldsymbol{y}_v$ and the desirable hyperparameters $\boldsymbol{\gamma}_v$.

2.2. *Module 2a:* SinoSmooth($\boldsymbol{y}_v$, $\boldsymbol{\gamma}_v$)

The sinogram smoothing module takes as input one noisy projection view $R^n \ni \boldsymbol{y}_v \equiv \boldsymbol{y} = \{y_i\}$ $i = 1, \cdots, n$, and the hyperparameters $\boldsymbol{\gamma}_v \equiv \boldsymbol{\gamma}$ from the CNN and generates a smoothed version $\hat{y}_v = \hat{\boldsymbol{y}}$ according to:†

$$\hat{\boldsymbol{y}} = \underset{z}{\operatorname{argmin}} \{ \frac{1}{2} \sum_{i=1}^{n} w_i (y_i - z_i)^2 + f(z, \gamma) \}, \tag{1}$$

where $w_i > 0$, $i = 1, \cdots, n$, are the known statistical weights of the sinogram that models the uncertainty in the data; the regularizer f is given by:

$$f(z, \gamma) = \sum_{i=1}^{n-1} \frac{\gamma_i}{2} (z_{i+1} - z_i)^2, \qquad \gamma \triangleq \{\gamma_1, \cdots, \gamma_{n-1}\} \tag{2}$$

As specified in (2), the hyperparameter $\boldsymbol{\gamma}$ is detector dependent, which drastically enhances the modeling power of the quadratic regularizer. For example, when $\gamma_i \to 1/|z_{i+1} - z_i|$ for all i, the quadratic penalty $f(\boldsymbol{z}, \boldsymbol{\gamma})$ in (2) may emulate 1-D total variation. In this work, the hyperparameter $\boldsymbol{\gamma}$ is driven by the CNN module (1).

Problem (1) with the regularizer (2) is a quadratic optimization problem, whose solution is linear in the data $\boldsymbol{y}$. One way to derive the solution is to sequentially apply completion of

†We employ TensorFlow in this work; other deep learning libraries, *e.g.*, PyTorch, are equally capable.
†All views are processed in the same manner; here in this section we omit the view index v to avoid notational clutter.

squares, for $i = 1, \cdots, n$, to the objective function (1), each time expressing $\hat{y}_i$ in terms of the "future" $\hat{y}_{i+1}$, and repeat the process to the last data point $i = n$. This procedure will generate the following forward-backward recursions known in dynamic programming (DP):

$$q_i = \frac{(q_{i-1} + w_i)\gamma_i}{q_{i-1} + w_i + \gamma_i}, \tag{3a}$$

$$c_i = \frac{(q_{i-1}c_{i-1} + w_i y_i)}{q_{i-1} + w_i}, \tag{3b}$$

for $i = 1, \cdots, n$ as the forward recursion, and

$$\hat{y}_i = \frac{q_{i-1}c_{i-1} + w_i y_i + \gamma_i \hat{y}_{i+1}}{q_{i-1} + w_i + \gamma_i}, \tag{4}$$

for $i = n, \cdots, 1$ as the backward recursion. The initial conditions for (3) and (4) are

$$q_0 = 0,\ c_0 = 0,\ \hat{y}_{n+1} = 0$$

We implemented (3) and (4) as a custom "SinoSmooth" layer in TensorFlow (TF). Since all operations are differentiable, TF's automatic differentiation capability can backpropagate the gradient information from the output $\hat{\boldsymbol{y}}$ to the hyperparameters $\boldsymbol{\gamma}$ to enable end-to-end training.

2.3. *Module 2b:* FBP(y)

The view-by-view smoothed sinogram $\hat{y}_v$ is assembled as input to the FBP reconstruction module, which then generates the reconstructed image. We implemented the fanbeam FBP algorithm (cosine-weighting, filter, weighted-backprojection) as a second custom layer in TF. We reconstructed randomly positioned region-of-interests (ROI) of size 200 × 200 pixels within the FOV of 50 cm.† The randomly positioned ROIs may be viewed as a data augmentation mechanism similar to the (randomly extracted) patches in CNN image denoising.

2.4. Supervised training

Training is carried out in a supervised manner using paired noisy sinograms $\boldsymbol{y}^s$ as input and noise-free images $\bar{\boldsymbol{x}}^s$ as training labels. More precisely, the loss function for training can be written as $\min_\theta \frac{1}{N}\sum_i \left\|\bar{\boldsymbol{x}}^s - \hat{\boldsymbol{x}}^s\right\|^2$, where $\bar{\boldsymbol{x}}_s$ is the sth noise-free image (label), and $\hat{\boldsymbol{x}}_s$ is the network output calculated as the following:

†The limited memory of our GPU card makes it impossible to do a full reconstruction of size 512 × 512. To improve memory efficiency, it may be necessary to provide the custom gradient for the backprojection operation in FBP, similar to [6].

$$\text{Module}(1)\gamma_v^s = \text{CNN}_\theta\left(\boldsymbol{y}_v^s\right)$$

$$\text{Module }(2a)\ \hat{\boldsymbol{y}}_v^s = \text{SinoSmooth}\left(\boldsymbol{y}_v^s, \gamma_v^s\right)$$

$$\text{Module }(2b)\ \hat{\boldsymbol{x}}^s = \text{FBP}\left(\hat{\boldsymbol{y}}_{\{v\}}^s\right)$$

where $s = 1, \cdots, N$ $v = 1, \cdots, m$. Both Module (1) and Module (2a) treat a m-view sinogram as 1-D signals of a batch size m. That is, the view dimension is taken as the batch dimension, and all views necessary for image reconstruction are processed in parallel.

3. Numerical studies

3.1. Data generation

We used the pancreas-CT dataset (total 18135 slices from 82 control patients) from the Cancer Imaging Archive [7] for data generation. The 2D CT images were first upsampled to double the matrix size ($512^2 \rightarrow 1024^2$), then forward projected with a distance-driven projector [8] to generate noise-free fanbeam data using a Siemens scanner (Sensation 64) geometry (2π acquisition of size $m = 1160$, $n = 672$). The noise-free sinograms were then reconstructed using FBP algorithm without apodization to generate the training labels $\bar{\boldsymbol{x}}^s$ of size 512^2. Noisy sinograms $\{\boldsymbol{y}^s\} = \left\{y_{v,i}^s\right\}$ were created by first converting the noise-free line integrals $\bar{y}_{v,i}^s$ to transmissions $I_i e^{-\bar{y}_{v,i}^s}$, where I_i, the air-scan photon counts, was adjusted by a body bowtie such that I_i varied from $2e5$ near the central to $4e4$ near the edge channels. The noisy transmission data followed the Poisson distribution with mean $I_i e^{-\bar{y}_{v,i}^s}$, and was log-converted to the noisy line integrals $\left\{y_{v,i}^s\right\} = \boldsymbol{y}^s$. The noisy sinograms were used as input to Module 1 and 2a of Fig. 2.

3.2. Network training

Of the 82 patients from the pancreas-CT datasets, 72 (~15,500 slices) were used for network training, and the rest (~2,600 slices) was used for network testing.† A random sample of 10 test cases is shown in Fig. 4. It can be seen that there is a rich variation in patient size and anatomy.

We implemented our network architecture in TensorFlow tf2.2 on an Nvidia Quadro P5000 GPU with 16 GB memory. For CNN training, the batch size for the 1-D DenseNet was 1160, the same as the total view angles over 2π acquisition. More details of the CNN module are

†This work focuses on a feasibility study, not on choosing the best CNN model for hyperparameter generation. Therefore we performed a 2-part train-test split, rather than the 3-part train-validation-test split which is common for model selection [9, page 222].

provided in Table 1, together with a tabulation of the layer parameters. Note that the # of parameters include both the convolution kernels, bias, and batch normalization. The large number of channels in layers B and C is due to the dense connections.

The CNN parameters were initialized with random weights and estimated using the Adam optimizer, with learning rate 1e-4 and learning rate decay following a polynomial schedule. Loss curves for training and testing are shown in Fig. 5.

3.3. Methods for comparison

We compared the proposed approach with two alternatives: (1) FBP, and (2) bi-level optimization. For FBP, we used the Hanning apodization and optimized the cutoff frequency individually for each test case presented. Bi-level optimization generates one set of the optimal hyperparameters from the training data, which is then applied to all test cases. Using the diagram of Fig. 2, it is easy to see that we obtain bilevel optimization by removing the CNN parameterization module. In this sense, our proposed approach amounts to a small but fundamental change to the bi-level method to enable patient-specific hyperparameter learning.

Our implementation of bi-level optimization is based on the same TF implementation of Module 2a and 2b; the only difference is that the backpropagated gradient through 2a and 2b was applied directly to the hyperparameters (not to the weights in CNN).

We used root mean squared error (RMSE) and structural similarity index (SSIM) to quantify the performance of the different methods.

4. Numerical results

The RMSE and SSIM of the 10 test cases (Fig. 4) for the different methods are plotted in Fig. 6. Among the three methods, bi-level optimization outperformed FBP in most cases, while the proposed hyperparameter learning performed the best in terms of the two figure-of-merits.

Fig. 7 shows our reconstruction results of two test case (case 4 and 1). The four images in Fig. 7(a) are: (1) the ground truth image, (2) the cut-off optimized FBP image, (3) the bi-level image, and (4) the reconstruction result using the proposed hyperparameter learning. The error images with respect to the ground truth are shown in Fig. 7(b); (c) and (d) are similar to (a) and (b) but for case 1. Overall, the errors of the proposed approach have the lowest magnitude, which agrees with Fig. 6.

The relationship between the learned hyperparameters and the noise-free sinogram is shown in Fig. 8. The 10 panels correspond to the 10 test cases of Fig 4, arranged in the same order. Each panel is a 2-D histogram of the learned sinogram-domain hyperparameters γ (the vertical axis) and the neighboring differences of the noise-free projection data (the horizontal axis). For each test case and each neighboring difference, the centroid γ value is shown as the thick black line that is overlaid on the histogram.

We observe from Fig. 8 that the learned hyperparameters are patient specific. The overall γ values are higher for the larger-sized patient (*e.g.*, case 1, 5), and lower for the smaller patients (*e.g.*, case 2, 9). Among the 10 test cases, case 9 has the smallest body size and the lung image of this patient is mostly air. Correspondingly, the sinogram of case 9 has the lowest noise level (due to our simulation of a common exposure). The learned hyperparameter values for this patient are also the smallest, as there is not much noise in the data to begin with.

For each test case, the centroid values of the learned hyperparameters have a unimodal shape: the hyperparameters are larger for smaller (noise-free) neighboring difference and lower for larger neighboring difference.‡ This observation agrees with our intuition that if we have the *a priori* knowledge of the underlying true sinogram, then we should apply higher penalty to neighbors that are known to be similar, and apply lower penalty to neighbors that are known to be different. This intuition has been used as an empirical guide for setting hyperparameters if a higher quality image is available, see, *e.g.*, [10]. Our learned hyperparameters exhibit a similar trend, and apply penalties to where they are needed.

5. Discussion

The contribution of the paper is twofold. First and foremost, we proposed a conceptual framework that exploits the synergy between DL and optimization-based reconstruction methods to improve image quality, *i.e.*, using DL to generate the hyperparameters for optimization based image reconstruction. Second, as a way of demonstrating its practical value, we applied the conceptual framework to generate hyperparameters for optimization-based sinogram smoothing.

Concept-wise, DL-based hyperparameter tuning for iterative reconstruction algorithms has been previously investigated, see, *e.g.*, the parameter tuning policy network (PTPN) [11] and the tuning-free plug-and-play (TFPP) [12]. These existing DL-based hyperparameter learning methods [11, 12] learn a hyperparameter *tuning* strategy based on feedback from intermediate image reconstruction results. They are not efficient at inference time as they require either (1) running multiple iterations of an image reconstruction algorithm [12], or (2) running multiple loops, each of which involves running an iterative algorithm till convergence [11]. Our approach is different in that once the network is trained, the patient-specific hyperparameters are generated in a feedforward manner, independent of the particular MBIR algorithm to be used. The inference-time efficiency of our hyperparameter generation is therefore much improved over the existing methods.

On the application side, we have shown that our DL-based patient-specific hyperparameter learning approach outperformed bilevel learning, which is not patient specific. Alternatively, DL can also be used to directly map a corrupted sinogram to an improved version as a way of sinogram preprocessing. The application areas of the existing DL-based sinogram preprocessing can be grouped as (1) sparse view CT [13,14] to improve sampling and reduce sparse-view artifacts, (2) metal artifacts [15] reduction, and (3) sinogram smoothing [16].

‡Here the larger (smaller) differences refer to the differences measured in absolute value.

The output sinogram of these approaches is directly used for image reconstruction. In our case, we do not use DL to enhance the sinogram itself; we use the DL network to learn hyperparameters that are subsequently used in a well-defined sinogram domain optimization problem. We do not claim this approach is the best DL-based sinogram smoothing approach. In fact, we believe low-dose sinogram smoothing would benefit from working directly with the pre-log data [17], not the post-log data that we used in this paper.

Coming back to PTPN, we point out that PTPN is an image domain approach: the hyperparameters reside in the image space, and the training objective is calculated using the reconstructed images. On the other hand, our application example of hyperparameter learning is a hybrid approach: the hyperparameters reside in the sinogram, but the training objective is calculated using the reconstructed images. It is not possible to apply PTPN to our setup without completely modifying it to a sinogram smoothing plus image reconstruction setting. Such a modification is beyond the scope of the paper and precludes at this stage a direct comparison.

In a broader context, this work, the two alternative hyperparameter tuning methods [11, 12] and the recent papers [18, 19], can all be viewed as ways of seeking a synergistic combination between DL and compressed sensing. As these works have evidently demonstrated, such a synergy is important and beneficial to improve image reconstruction.

Future work may proceed in several directions. We focused on view-by-view sinogram smoothing with a spatially variant quadratic penalty since the resulting quadratic problem has an easy-to-implement DP solution that can be ported within a DL library. Indeed, the objective function in (1) can accommodate other penalties, *e.g.*, the 1-D total variation [20], yet still allow efficient DP solutions. We previously developed a DP algorithm for solving (1) with the Huber penalty function [21]. Unlike the quadratic or the TV penalty, the Huber function itself contains a hyperparameter that should be learned from the data together with the penalty weights. The framework of Fig. 2 can be applied to such joint learning tasks.

A practical issue for implementing the fully generic framework of Fig. 1 is how to implement the module for optimization-based image reconstruction, *i.e.*, MBIR, as such problems usually do not have closed form solutions. One approach is to implement an approximate solution by “unrolling” an iterative algorithm [22–24] for a fixed number of iterations; porting such an unrolled solution within a DL library may allow end-to-end training using Fig. 1. To maximize GPU memory efficiency for such unrolling type implementations, it may be necessary to implement C++ based custom layers and gradient layers [6] rather than relying on autodiff.

6. Conclusions

We presented a generic framework for hyperparameter learning in the context of optimization-based CT image reconstruction. As a proof-of-concept study, in this work we specialized the generic framework and focused on a special two-step reconstruction problem, *i.e.*, optimization-based sinogram smoothing plus FBP, where the sinogram smoothing problem has exactly computable solutions. Our numerical experiments

demonstrate improved performance, in terms of RMSE and SSIM, by using the proposed hyperparameter learning framework compared to bi-level optimization.

The conceptual difference between the proposed approach and bi-level optimization is that bi-level optimization learns a set of parameters from the training data, whereas the proposed approach learns a parameterization map. The learned parameterization (or functional mapping) between the sinogram and the hyperparameters can be used to generate patient-specific hyperparameters in an efficient manner. The personalized hyperparameters are expected to improve the performance of optimization-based image reconstruction for CT applications.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

References

[1]. Bengio Y, “Gradient-based optimization of hyperparameters,” Neural computation, vol. 12, no. 8, pp. 1889–1900, 2000. [PubMed: 10953243]

[2]. Okuno T and Takeda A, “Bilevel optimization of regularization hyperparameters in machine learning,” in Bilevel Optimization, pp. 169–194, Springer, 2020.

[3]. La Rivière PJ, Bian J, and Vargas PA, “Penalized-likelihood sinogram restoration for computed tomography,” IEEE transactions on medical imaging, vol. 25, no. 8, pp. 1022–1036, 2006. [PubMed: 16894995]

[4]. Yu L, Manduca A, Trzasko JD, Khaylova N, Kofler JM, McCollough CM, and Fletcher JG, “Sinogram smoothing with bilateral filtering for low-dose CT,” in Medical Imaging 2008: Physics of Medical Imaging, vol. 6913, p. 691329, International Society for Optics and Photonics, 2008.

[5]. Huang G, Liu Z, Pleiss G, Van Der Maaten L, and Weinberger K, “Convolutional networks with dense connectivity,” IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2019.

[6]. Syben C, Michen M, Stimpel B, Seitz S, Ploner S, and Maier AK, “Pyro-nn: Python reconstruction operators in neural networks,” Medical physics, vol. 46, no. 11, pp. 5110–5115, 2019. [PubMed: 31389023]

[7]. Clark K, Vendt B, Smith K, Freymann J, Kirby J, Koppel P, Moore S, Phillips S, Maffitt D, Pringle M, Tarbox L, and Prior F, “The Cancer Imaging Archive (TCIA): Maintaining and Operating a Public Information Repository,” Journal of Digital Imaging, vol. 26, pp. 1045–1057, 12. 2013. [PubMed: 23884657]

[8]. De Man B and Basu S, “Distance-driven projection and backprojection in three dimensions,” Physics in Medicine & Biology, vol. 49, no. 11, p. 2463, 2004. [PubMed: 15248590]

[9]. Hastie T, Tibshirani R, and Friedman J, The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media, 2009.

[10]. Kim K, El Fakhri G, and Li Q, “Low-dose CT reconstruction using spatially encoded nonlocal penalty,” Medical physics, vol. 44, no. 10, pp. e376–e390, 2017. [PubMed: 29027240]

[11]. Shen C, Gonzalez Y, Chen L, Jiang SB, and Jia X, “Intelligent parameter tuning in optimization-based iterative CT reconstruction via deep reinforcement learning,” IEEE transactions on medical imaging, vol. 37, no. 6, pp. 1430–1439, 2018. [PubMed: 29870371]

[12]. Wei K, Aviles-Rivero A, Liang J, Fu Y, Schönlieb C-B, and Huang H, “Tuning-free plug-and-play proximal algorithm for inverse imaging problems,” in International Conference on Machine Learning, pp. 10158–10169, PMLR, 2020.

[13]. Lee H, Lee J, Kim H, Cho B, and Cho S, “Deep-neural-network-based sinogram synthesis for sparse-view CT image reconstruction,” IEEE Transactions on Radiation and Plasma Medical Sciences, vol. 3, no. 2, pp. 109–119, 2018.

[14]. Tan J, Gao Y, Huo Y, Li L, and Liang Z, “Sharpness preserved sinogram synthesis using convolutional neural network for sparse-view CT imaging,” in Medical Imaging 2019: Image Processing, vol. 10949, p. 109490E, International Society for Optics and Photonics, 2019.

[15]. Ghani MU and Karl WC, “Deep learning based sinogram correction for metal artifact reduction,” Electronic Imaging, vol. 2018, no. 15, pp. 472–1, 2018.

[16]. Ghani MU and Karl WC, “CNN based sinogram denoising for low-dose CT,” in Mathematics in Imaging, pp. MM2D–5, Optical Society of America, 2018.

[17]. Fu L, Lee T-C, Kim SM, Alessio AM, Kinahan PE, Chang Z, Sauer K, Kalra MK, and De Man B, “Comparison between pre-log and post-log statistical models in ultra-low-dose CT reconstruction,” IEEE transactions on medical imaging, vol. 36, no. 3, pp. 707–720, 2016. [PubMed: 28113926]

[18]. Wu W, Hu D, Cong W, Shan H, Wang S, Niu C, Yan P, Yu H, Vardhanabhuti V, and Wang G, “Stabilizing deep tomographic reconstruction networks,” arXiv preprint arXiv:2008.01846, 2020.

[19]. Hayes JW, Montoya J, Budde A, Zhang C, Li Y, Lia K, Hsieh J, and Chen G-H, “High pitch helical CT reconstruction,” IEEE Transactions on Medical Imaging, vol. on line, early access, pp. 1–1, 2021.

[20]. Condat L, “A Direct Algorithm for 1-D Total Variation Denoising,” IEEE Signal Processing Letters, vol. 20, pp. 1054–1057, 11. 2013.

[21]. Xu J, Noo F, and Tsui BM, “A direct algorithm for optimization problems with the Huber penalty,” IEEE transactions on medical imaging, vol. 37, no. 1, pp. 162–172, 2017. [PubMed: 28981412]

[22]. Hammernik K, Klatzer T, Kobler E, Recht MP, Sodickson DK, Pock T, and Knoll F, “Learning a variational network for reconstruction of accelerated MRI data,” Magnetic resonance in medicine, vol. 79, no. 6, pp. 3055–3071, 2018. [PubMed: 29115689]

[23]. Adler J and Öktem O, “Learned primal-dual reconstruction,” IEEE transactions on medical imaging, vol. 37, no. 6, pp. 1322–1332, 2018. [PubMed: 29870362]

[24]. Yang Y, Sun J, Li H, and Xu Z, “ADMM-CSNet: A deep learning approach for image compressive sensing,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 3, pp. 521–538, 2020. [PubMed: 30507495]

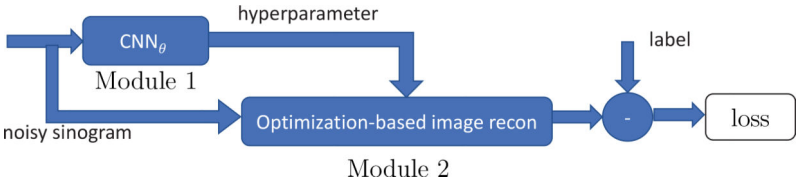

Figure 1:

The proposed hyperparameter learning framework consists of two functional modules: (1) the hyperparameter learning module, (2) optimization-based image reconstruction (*e.g.*, an MBIR module) that generates the reconstructed image. Only Module 1 contains trainable parameters, which are trained end-to-end in a supervised manner.

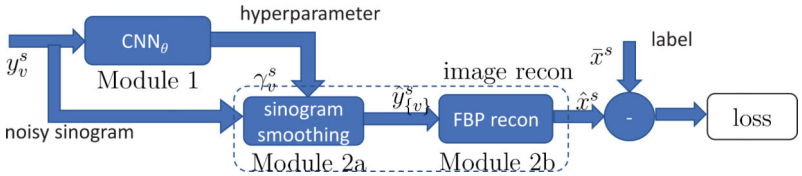

Figure 2:

The network training architecture for learning a parametric mapping between the sinogram and the hyperparameters.

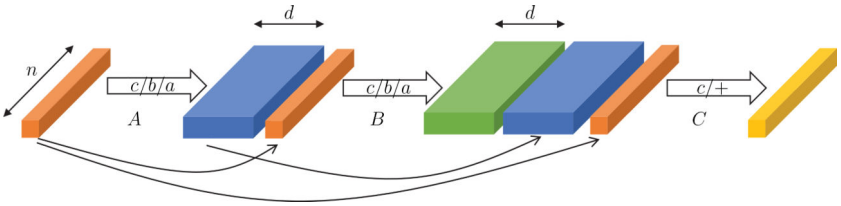

Figure 3:

Module 1 architecture. In layers A, B, the symbol c/b/a stands for convolution, batch normalization, and leaky ReLU activation. In layer C, *c*/+ stands for convolution and ReLU. Leaky ReLU was used in layers A and B to allow gradient backpropagation for all values of inputs to the activation, thereby avoiding the dying ReLU problem. ReLU was used in layer C to ensure that hyperparameters are nonnegative. The arrows → indicate block copy in dense connections. When the batch size is 1, the input is a 1-D vector of length *n* (detector elements) representing one projection view; *d* is the number of 1-D convolution filters.

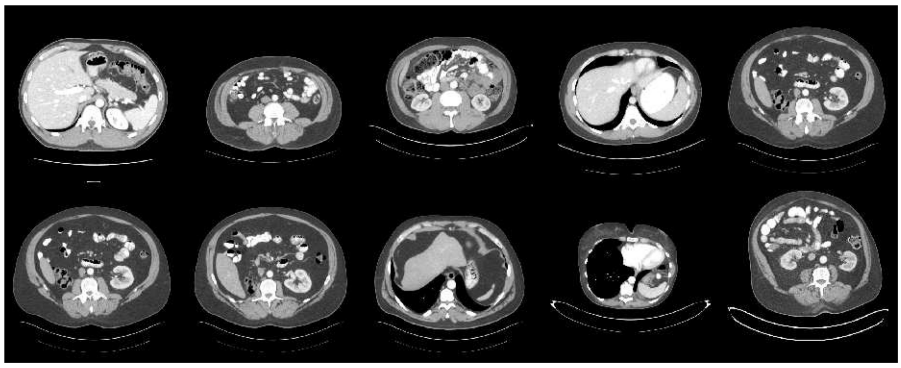

Figure 4:
A random sample of 10 test cases from TCIA used for quantitative analysis. Display window (C, W) = (1034, 400) HU.

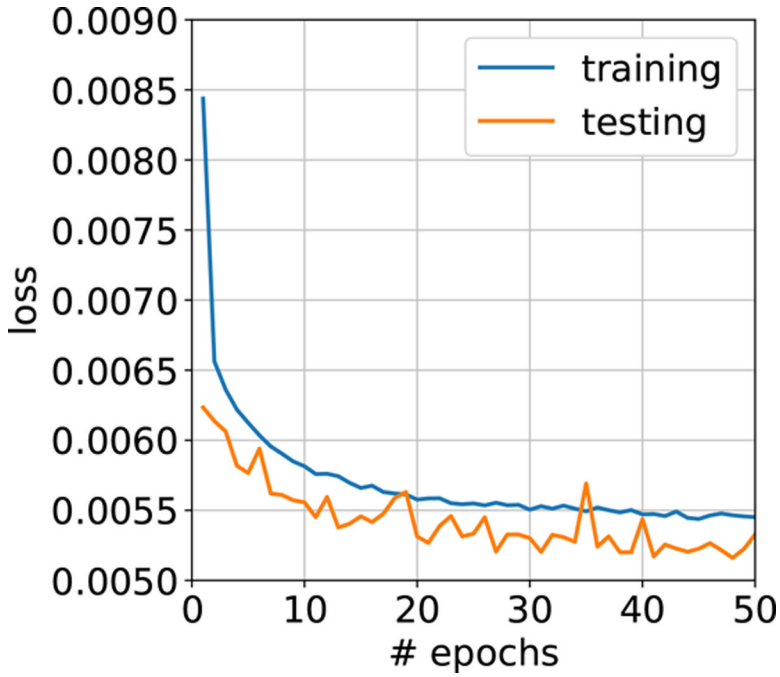

Figure 5:
Loss curves for training and testing.

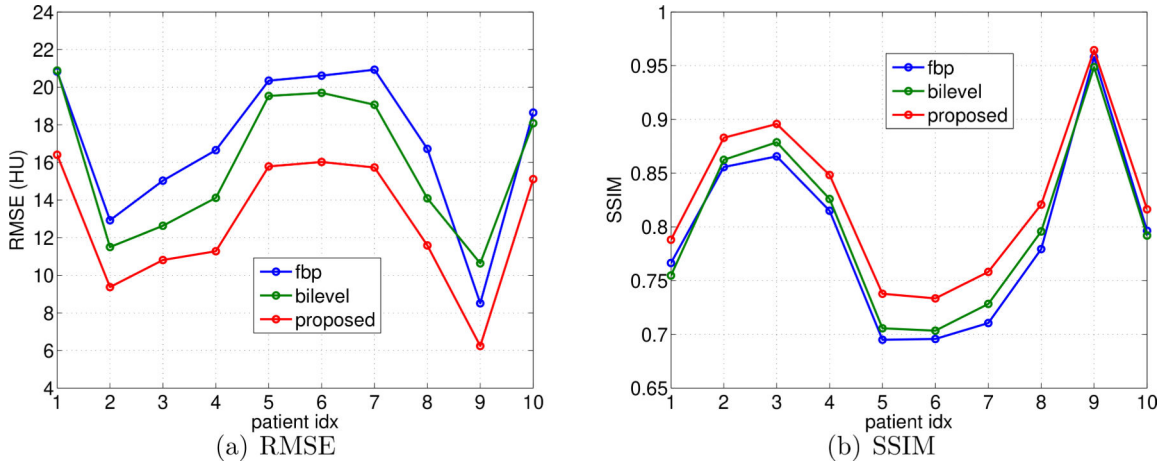

Figure 6:

(a) RMSE and (b) SSIM of the 10 test cases in Fig. 4. The FBP images are individually optimized in terms of the cutoff frequencies. The proposed method outperformed FBP and bi-level optimization.

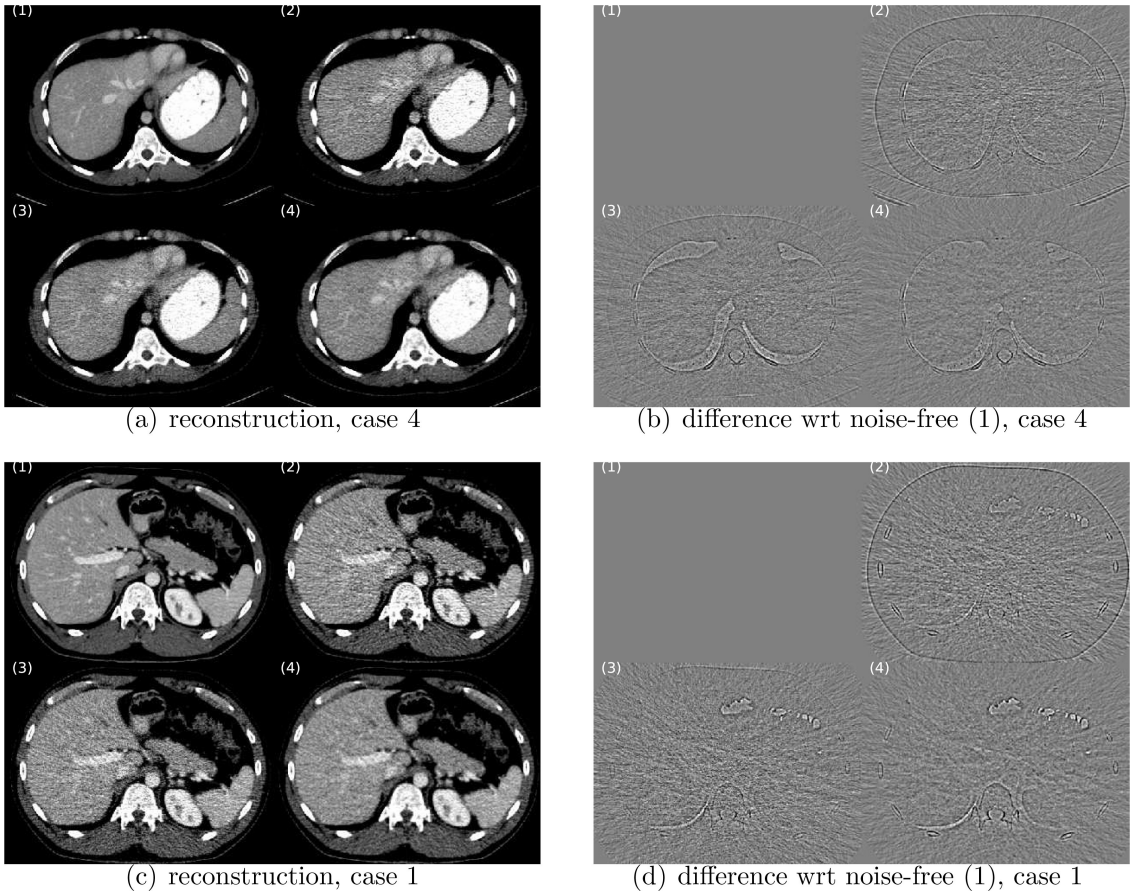

Figure 7:

Result of two test cases (case 1 and 4). (a) The four images are: (1) the ground truth, (2) cut-off optimized FBP reconstruction, (3) bi-level optimization, (4) result using the proposed CNN hyperparameter learning. (b) The difference images with respect to the noise-free ground truth. (c) and (d) are similar to (a) and (b) but for test case 1. Display window: (C, W) = (1170, 300) HU for (a), (c), (C, W) = (0, 300) HU for (b), (d).

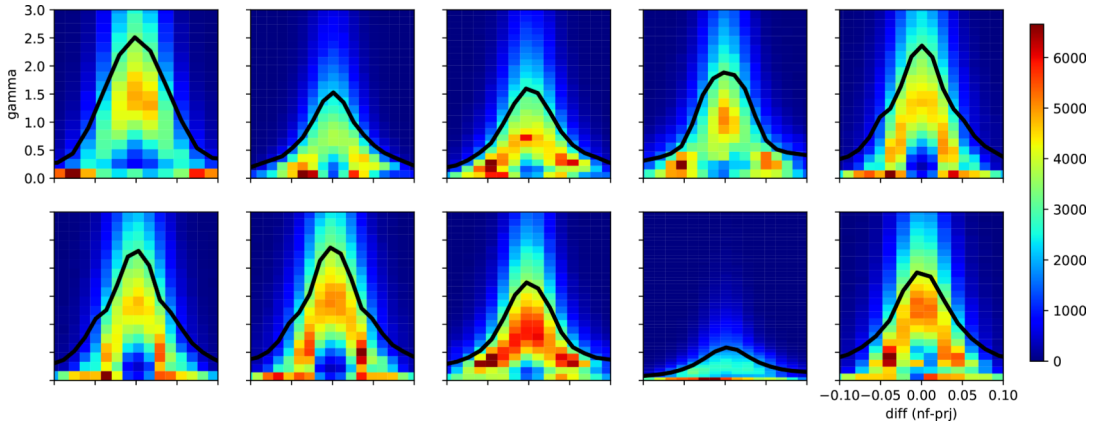

Figure 8:

Relatioship between the learned hyperparameter (γ) and the neighboring difference of the noise-free line-integrals $\bar{y}_i$, shown as 2-D histograms. The thick black lines are the centroid γ_i value calculated as a function of the neighboring differences $\bar{y}_{i+1} - \bar{y}_i$. The 10 panels correspond to the 10 test cases, arranged in the same order as Fig. 4. All plots have the same axis scale, given in the first (for the vertical axis) and the last (for the horizontal axis) panels.

Table 1:

Parameters in CNN, module (1), with 1-D convolution kernel size 3, and $d = 128$. The shape of the input and output follows the convention of batch size × width × channels.

layers	input	output	# params
A	1160 × 672 × 1	1160 × 672 × 128	1024
B	1160 × 672 × 129	1160 × 672 × 128	49664
C	1160 × 672 × 257	1160 × 672 × 1	772