Linearized analysis of noise and resolution for DL-based image generation

Jingyan Xu, Frederic Noo

***Abstract*— Deep-learning (DL) based CT image generation methods are often evaluated using RMSE and SSIM. By contrast, conventional model-based image reconstruction (MBIR) methods are often evaluated using image properties such as resolution, noise, bias. Calculating such image properties requires time consuming Monte Carlo (MC) simulations. For MBIR, linearized analysis using first order Taylor expansion has been developed to characterize noise and resolution without MC simulations. This inspired us to investigate if linearization can be applied to DL networks to enable efficient characterization of resolution and noise. We used FBPConvNet as an example DL network and performed extensive numerical evaluations, including both computer simulations and real CT data. Our results showed that network linearization works well under normal exposure settings. For such applications, linearization can characterize image noise and resolutions without running MC simulations. We provide with this work the computational tools to implement network linearization. The efficiency and ease of implementation of network linearization can hopefully popularize the physics-related image quality measures for DL applications. Our methodology is general; it allows flexible compositions of DL nonlinear modules and linear operators such as filtered-backprojection (FBP). For the latter, we develop a generic method for computing the covariance images that is needed for network linearization.**

***Index Terms*— image resolution, image noise, (linearized) local impulse response, contrast recovery coefficient, noise covariance image, FBPConvNet**

I. Introduction

Deep-learning (DL) based CT image generation methods have proliferated over the past years. Image quality (IQ) evaluation of these DL methods often employ simple quantitative figure of merits (FOM), such as structural similarity index (SSIM), peak signal-to-noise ratio (PSNR), root mean square error (RMSE). One concern about these IQ metrics is that they are not task-specific. That is, medical images are created for some specific purpose or task, the two most common being classification and estimation [1]. The gold standard for classification task performance is observer studies with either human or model observers using receiver operating characteristic (ROC) curves and the area under the curve as FOMs. For estimation tasks, the reconstructed CT images contain quantitative information that requires both precision and accuracy [2, chapter 5]. These task-specific IQ metrics are often evaluated using image ensembles, *i.e.*, repeated experiments or Monte Carlo (MC) simulations with multiple noise realizations – such a summarizing feature from image ensembles is absent from SSIM or RMSE. Being easily computable is an advantage, *e.g.*, they can be incorporated as part of the objective function for network training. However, SSIM or RMSE may not correlate well with task performance; it may happen that SSIM or RMSE continues to improve while signal detectability does not [3], [4].

Before the arrival of DL, state-of-the art imaging performance was dominated by model-based image reconstruction (MBIR). These methods are often evaluated using physics-related image properties, such as image noise, resolution, bias. The computation associated with such ensemble properties includes (1) generating the multiple cases as inputs to MBIR and (2) running the multiple cases through MBIR. Both (1) and (2) require considerable computation. One way to obviate the extensive computation is to apply linearized analysis to the possibly nonlinear MBIR [5]–[7]. The linearization techniques apply the first order Taylor series expansion to the solution mapping of MBIR. When applicable, such linearization techniques lead to efficient characterization of noise/resolution.

Inspired by these previous studies, in this work we investigate if linearization techniques can be applied to DL networks and enable efficient characterization of image noise and resolution without running MC simulations, which, similar to MBIR, involves two aspects of computation: (1) preparing the multiple cases as DL inputs and (2) running the multiple cases through DL to accumulate the images. Depending on the type of DL networks, *i.e.*, the feed-forward type [8], [9] or the recurrent/unrolled type [10], [11], the computation savings can be substantial.

The rest of the paper is organized as follows. In Section II we explain our problem setup and the quantities that we use to characterize resolution and noise. We develop network linearization techniques and the associated computational tools in Section III; in Sections IV and V we discuss our numerical studies and compare results from network linearization with those from MC simulations. In Section VI we discuss different application scenarios for network linearization, and we conclude in Section VII with a brief outlook on possible future extensions. Two sections in the appendix provide further computational and experimental details. Additional image data can be found online at (https://tinyurl.com/4sft28vn).

J Xu is with the Department of Radiology, Johns Hopkins University. F Noo is with the Department of Radiology and Imaging Sciences, University of Utah. J Xu was partly supported by funding from The Sol Goldman Pancreatic Cancer Research Center at JHU and by funding from the NIH under R03EB030653. F Noo was partly supported by the National Institute of Biomedical Imaging and Bioengineering of the National Institutes of Health under R21EB029179. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH. E-mail: jxu@jhmi.edu.

II. Background

We consider a DL processing pipeline as a generic nonlinear mapping, $\mathcal{F} : R^m \to R^n$, from the input space $Y \in R^m$ to the output space $X \in R^n$. In the context of CT image generation, it is helpful to think of $Y \in R^m$ as the sinogram and $X \in R^n$ the CT images. $\mathcal{F}$ itself may consist of linear or nonlinear components. For example, the data processing chain of FBPConvNet [12] can be written as $\mathcal{F}(Y) = \mathcal{F}_1 \circ F_0(Y)$, where Y is the sparse-view sinogram, F_0 denotes the linear FBP reconstruction operator, $\mathcal{F}_1$ denotes the nonlinear U-Net that reduces streak artifacts, and the symbol $\circ$ denotes operator composition. As another example, dual-domain DL approaches [13], [14] can be written as $\mathcal{F}(Y) = \mathcal{F}_1 \circ F_0 \circ \mathcal{F}_2(Y)$, where a DL network $\mathcal{F}_2$ is trained to perform sinogram preprocessing, followed by FBP reconstruction F_0, and lastly post-processed by an image domain DL network $\mathcal{F}_1$. As the examples show, the nonlinear mapping $\mathcal{F}$ can be quite flexible in its structure, but we use the term "DL network" or "DNN" (for deep neural network) in a non-specific sense for $\mathcal{F}$, keeping in mind that it may include components such as F_0 that are not standard modules of a DL library.

As our characterization of resolution and noise requires image ensembles, we assume that the input $Y = [Y_1, \cdots, Y_m]' \in R^m$ is a random variable (*e.g.*, a noisy sinogram) with a known probability density function $f(y;\theta)$, where the parameter $\theta = [\theta_1, \cdots, \theta_n]' \in R^n$ is the unknown ground truth that $\mathcal{F}$ is trained to estimate. For CT image generation, it is common to assume that Y follows a multivariate normal distribution [15]–[17] with mean $\bar{Y}$ and a diagonal covariance matrix Σ_Y that are dependent on θ.

In this work, we assume all training has completed. That is, $\mathcal{F}$ is a known deterministic mapping that estimates the ground truth θ for each input sample $Y = y$ using $x = \mathcal{F}(y)$. Due to the randomness of Y, x is a sample from a random variable X with a distribution induced by Y and $\mathcal{F}$. The mean of the output X is given by

$$\mu(\theta) \triangleq E_\theta(\mathcal{F}(Y)) = \int \mathcal{F}(y) f(y;\theta)\, dy \tag{1}$$

where the expectation (E) is taken with respect to the random variable Y with parameter θ fixed.

A. Local impulse response

The local impulse response (LIR) of a nonlinear estimator $\mathcal{F}$ is defined as the gradient of the mean output [5]:

$$l_j(\theta) = \lim_{\epsilon \to 0} \frac{\mu(\theta + \epsilon\delta_j) - \mu(\theta)}{\epsilon} = \frac{\partial}{\partial \theta_j} \mu(\theta) \tag{2}$$

for each pixel of interest $j = 1, \cdots, p$, where $\delta_j \in R^n$ is the jth unit vector. The following recipe adapted from [5] can be used to generate an unbiased estimate of the LIR.

1) Generate "signal-absent" noisy sinogram samples $\{y^{(i)}\}$, $i = 1, \cdots, N$, according to the density $f(y;\theta)$;
2) Obtain DL outputs $\mathcal{F}(y^{(i)})$ for each noisy sample $\{y^{(i)}\}$;
3) Estimate the "signal-absent" mean output:

$$\hat{\mu}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{F}(y^{(i)})$$

4) Choose a set of pixels of interest, $j = 1, \cdots, p$, and a small value ϵ, generate "signal-only" noise-free sinograms s_j from the ground truth impulse image $\epsilon\delta_j$; add the signal-only sinograms s_j to the noisy "signal-absent" sinograms $y^{(i)}$ to obtain the noisy "signal-present" sinograms as $y^{(i)} + s_j$ for $i = 1, \cdots, N$ and $j = 1, \cdots, p$.
5) Apply DL to the "signal-present" sinogram samples to obtain $\mathcal{F}(y^{(i)} + s_j)$, and estimate the "signal-present" mean output for each signal location j:

$$\hat{\mu}(\theta + \epsilon\delta_j) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{F}(y^{(i)} + s_j)$$

6) Estimate the local impulse response by

$$l_j(\theta) \approx \frac{\hat{\mu}(\theta + \epsilon\delta_j) - \hat{\mu}(\theta)}{\epsilon} \tag{3}$$

A difference between the above recipe and the one in [5] is in step 4, where instead of drawing samples from the distribution $f(y;\theta+\epsilon\delta_j)$, we take advantage of the weak signal assumption (ϵ small) and assume that the weak signal only affects the mean of the sinogram but not the variance. Furthermore, we reuse the signal-absent sinograms $\{y^{(i)}\}$ without generating another set of independent noise realizations. In view of (3), this amounts to using common random numbers [18] for variance reduction in MC simulations [19, chapter 8].

The contrast recovery coefficient (CRC) at pixel j is simply the center value of the LIR, *i.e.*,

$$\mathrm{CRC}(x_j) = \delta_j^t l_j(\theta) \tag{4}$$

The CRC has been used as a scalar FOM to quantify resolution [20], [21].

B. Linearized local impulse response (LLIR)

When linearity holds, (1) can be simplified as

$$E_\theta(\mathcal{F}(Y)) \overset{\text{lin}}{=} \mathcal{F}(\bar{Y}) \overset{\triangle}{=} \bar{\mu}(\theta) \tag{5}$$

Plugging (5) into (2) leads to the *linearized* local impulse response (LLIR):

$$\ell_j(\theta) = \frac{\partial}{\partial \theta_j} \bar{\mu}(\theta) = \lim_{\epsilon \to 0} \frac{\mathcal{F}(\bar{Y}(\theta + \epsilon\delta_j)) - \mathcal{F}(\bar{Y}(\theta))}{\epsilon} \tag{6}$$

In general, *i.e.*, when $\mathcal{F}$ is nonlinear, $\ell_j(\theta)$ is at best an approximation of $l_j(\theta)$. Such a linear approximation has been applied before for nonlinear reconstruction [5], [20].

Applying the chain rule to (6), the LLIR can be further simplified as [21], [22]

$$\ell_j(\theta) = \nabla_y \mathcal{F}(y)\big|_{y=\bar{Y}(\theta)} \nabla_x \bar{Y}(x)\big|_{x=\theta} \delta_j \tag{7}$$

where the two Jacobian matrices in (7) are defined by

$$[\nabla_y \mathcal{F}(y)]_{i,j} = \frac{\partial \mathcal{F}_i(y)}{\partial y_j}, \quad [\nabla_x \bar{Y}(x)]_{i,j} = \frac{\partial \bar{Y}_i(x)}{\partial x_j}$$

In this work, we use a linear model for CT projection generation, *i.e.*, $\bar{Y} = A\theta$, where θ is the ground truth image, and A is the projection matrix. With this assumption, the LLIR from (7) is simplified to

$$\ell_j(\theta) = \nabla_y \mathcal{F}(y)|_{y=\bar{Y}} A\delta_j \tag{8}$$

C. Noise covariance image

Given $X = \mathcal{F}(Y)$, the covariance matrix of X is given by

$$\Sigma_X = E\{(\mathcal{F}(Y) - \mu(\theta))(\mathcal{F}(Y) - \mu(\theta))^t\}$$

The matrix $\Sigma_X \in R^{n\times n}$ is enormous. It is often informative to look at the covariance between a pixel of interest j and the image itself, *i.e.*, the covariance image. This information can be extracted from the jth row/column of Σ_X as

$$\text{cov}(x_j) = \Sigma_X \delta_j \tag{9}$$

The variance of the jth-pixel is simply $\text{var}(x_j) = \delta_j^t \Sigma_X \delta_j$.

An unbiased estimate of the covariance image (9) can be obtained using sample noise realizations $x^{(i)} = \mathcal{F}(y^{(i)})$,

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)} \tag{10a}$$

$$\text{cov}(x_j) \approx \frac{1}{N-1}\sum_{i=1}^{N}(x_j^{(i)} - \hat{\mu}_j)(x^{(i)} - \hat{\mu}) \tag{10b}$$

If the mapping $\mathcal{F}$ is linear, *i.e.*, $\mathcal{F} \equiv \nabla\mathcal{F}$, the covariance matrix/images of X can be obtained from that of the input using standard noise propagation.

$$\Sigma_X = \nabla\mathcal{F}\Sigma_Y\nabla\mathcal{F}^t,\ \text{cov}(x_j) = \nabla\mathcal{F}\Sigma_Y\nabla\mathcal{F}^t\delta_j$$

III. Method

A. Network linearization

Let $\mathcal{F} : Y \to X$ be a pre-trained differentiable DNN, where $Y \in R^m$ is a random variable, and $E(Y) = \bar{Y}$. If we assume the deviation between Y and $\bar{Y}$ is small, then $\mathcal{F}(Y)$ can be approximated using the first order Taylor series expansion around $\bar{Y}$:

$$X \approx X_{(l)} = \mathcal{F}(\bar{Y}) + \nabla\mathcal{F}(\bar{Y})(Y - \bar{Y}) \tag{11}$$

where[1] $\nabla\mathcal{F}(\bar{Y}) \equiv \nabla_y\mathcal{F}(y)|_{y=\bar{Y}}$ is the Jacobian of $\mathcal{F}$ evaluated at $\bar{Y}$. From (11), the mean of the linearized output X_l is simply $\mathcal{F}(\bar{Y})$, *i.e.*, the DL output evaluated at the mean of the input. The covariance matrix and the covariance image at a pixel of interest j of the linearized output $X_{(l)}$ are

$$\Sigma_{X_{(l)}} = \nabla\mathcal{F}(\bar{Y})\Sigma_Y\nabla\mathcal{F}^t(\bar{Y}) \tag{12}$$

$$\overline{\text{cov}}(x_j) = \nabla\mathcal{F}(\bar{Y})\Sigma_Y\nabla\mathcal{F}^t(\bar{Y})\delta_j \tag{13}$$

From (8), we rewrite the LLIR at pixel j as

$$\ell_j(\theta) = \nabla\mathcal{F}(\bar{Y})A\delta_j \tag{14}$$

Our question is whether $\text{cov}(x_j)$ and $l_j(\theta)$ can be approximated by their linearized counterparts $\overline{\text{cov}}(x_j)$ and $\ell_j(\theta)$ to allow efficient evaluation. We discussed in Section II how to calculate $\text{cov}(x_j)$ and $l_j(\theta)$ using MC simulations; see also Fig. 2 for an illustration of this in the context of FBPConvNet. The rest of this section will focus on computational techniques for calculating the linearized counterparts (13) and (14).

From (13) and (14), we see that their computation requires two kinds of matrix-vector products, the first is $J^t\nu$, and the second Jv, where $J = \nabla\mathcal{F}(\bar{Y})$, and ν, v are arbitrary vectors. For the LLIR (14), we need only Jv; for the covariance image (13), we need both $J^t\nu$ and Jv performed back-to-back. Here two remarks are in order: (1) calculating the LLIR is about half the cost of calculating the linearized covariance image; (2) although our discussion needs the Jacobian matrix J and its transpose J^t, in the actual implementation we always calculate the matrix-vector product as one entity. There is no need to explicitly form the Jacobian, which makes our calculation both memory-efficient and scalable.

The autodiff capabilities in DL libraries can perform gradient *backpropagation* for network training, an operation akin to $\nabla\mathcal{F}^t\nu$. Some DL libraries have built-in interfaces for *forward* gradient computation,[2] *i.e.*, $\nabla\mathcal{F}v$. Regardless of the built-in interfaces, we can always perform forward gradient propagation using backpropagation only. The trick is well known to the autodiff community, but we have not seen it in medical imaging literature. We introduce it below, hoping that it can find more applications.

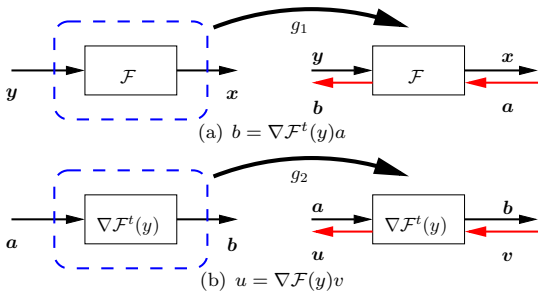


Fig. 1. (a) Tracking the forward pass $\boldsymbol{x = \mathcal{F}(y)}$ allows backpropagating gradient information, *i.e.*, $\boldsymbol{b = \nabla\mathcal{F}^t(y)a}$. (b) Now treating the gradient mapping $\boldsymbol{a \to b}$ as a forward mapping and backtrack over that, we effectively perform forward gradient mapping over the original network $\boldsymbol{\mathcal{F}}$. Here $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$ are gradient tracking mechanisms of DL. In TensorFlow, they can be implemented using GradientTape.

Fig. 1 illustrates the idea, and the accompanying pseudocode using TensorFlow is given in Algorithm 1. The trick is to track the computation during the calculation of $b = \nabla\mathcal{F}^t a$, which itself is the backward gradient pass for the forward computation $x = \mathcal{F}(y)$. Then another backward gradient pass of the linear mapping $a \to b$, evaluated at an arbitrary vector v, gives the forward gradient $\nabla\mathcal{F}v$.

Having resolved the issue of forward gradient propagation in a generic setting, we still face the issue of having nonstandard DL modules, *e.g.*, the FBP operator F_0, in the DL pipeline that are part of the linearized LIR and covariance image computation. In the next two subsections, we discuss this issue

[1]We use this simplified notation for the rest of the paper.

[2]DL libraries are constantly evolving. TensorFlow can perform forward-mode gradient computation. As of March 2022, PyTorch also has a beta version for the forward-mode gradient computation.

Algorithm 1: Forward gradient: $u = \nabla\mathcal{F}(w)v$

Input: w, v, a pretrained DNN $\mathcal{F}$
Output: $u = \nabla\mathcal{F}(w)v$

```
1  with tf.GradientTape () as g1:
   /* GradientTape tracks the gradient wrt 'watched' variable   */
2      g1.watch (w)
3      x = F(w)
   /* b = ∇_w(x · a) is available ∀a – see line 7-8 and Fig. 1(a) */
4  a = 0    /* a is an arbitrary dummy variable;                 */
   /* lines 5-8 wrap b ← a in 2nd GradientTape, see Fig. 1(b)    */
5  with tf.GradientTape () as g2:
6      g2.watch (a)
7      b = g1.gradient (x, w, \
8              output_gradients = a)
                                   /* b = ∇F^t(w)a              */
   /* mapping u ← v is the transpose of b ← a, see Fig. 1(b)    */
9  u = g2.gradient (b, a, \
10             output_gradients = v)
```

for our specific application, *i.e.*, FBPConvNet. Generalization to other structures of $\mathcal{F}$ is straightforward.

B. Calculating LLIR of FBPConvNet

Let $\mathcal{F} = \mathcal{F}_1 \circ F_0$. For convenience, we use the following shorthand notation,

$$\mathcal{G}(F_0(\bar{Y})) \triangleq \nabla_x \mathcal{F}_1(x)|_{x=F_0(\bar{Y})} \tag{15}$$

From (14), we have

$$\ell_j(\theta) = \mathcal{G}(F_0(\bar{Y})) \circ \nabla F_0(A\delta_j) \tag{16a}$$

$$\overset{(a)}{=} \mathcal{G}(F_0(\bar{Y})) \circ F_0(A\delta_j) \tag{16b}$$

where (a) is due to the linearity of FBP. That is, the forward gradient propagation through FBP $\nabla F_0 v$ is the same as applying FBP to v.

An LLIR at location j can be computed using Algorithm 1 by identifying $\mathcal{F} \equiv \mathcal{F}_1$, $w \equiv F_0(\bar{Y})$, and $v = F_0(A\delta_j)$. As such, calculating LLIR at one pixel location j amounts to forward propagating, through the U-Net, the FBP reconstruction of the sinogram from a unit-input δ_j, while the U-Net gradient is evaluated at the FBP reconstruction of the noise-free sinogram $\bar{Y}$.

C. Calculating the covariance image of FBPConvNet

Using the shorthand notation (15), the linearized covariance image of the FBPConvNet can be written as

$$\overline{\mathrm{cov}}(x_j) = \mathcal{G}(F_0(\bar{Y}))F_0(\bar{Y})\Sigma_Y F_0^t(\bar{Y})\mathcal{G}^t(F_0(\bar{Y}))\delta_j \tag{17}$$

The calculation of (17) can be divided into three steps:

$$a \triangleq \mathcal{G}^t(F_0(\bar{Y}))\delta_j, \tag{18a}$$

$$b \triangleq F_0(\bar{Y})\Sigma_Y F_0^t(\bar{Y})a \tag{18b}$$

$$\overline{\mathrm{cov}}(x_j) = \mathcal{G}(F_0(\bar{Y}))b \tag{18c}$$

where (18a) and (18c) involves gradient backward and forward propagation and both can be carried out using autodiff (cf. Section III-A and Algorithm 1); the computations in (18b) involves (i) backpropagation through FBP, *i.e.*, the transpose operator of FBP (F_0^t); (ii) multiplication with the (diagonal) covariance matrix of the sinogram Σ_Y, and (iii) then forward propagation through FBP F_0. Step (iii) is the same as doing one FBP as we have discussed, and (ii) is also trivial as Σ_Y is diagonal, so the remaining issue is (i).

Regarding (i), one may write FBP as a layer using a DL library's native constructs, then autodiff can take care of gradient backpropagation (F_0^t). The difficulty of this approach is that it increases the memory footprint of the whole processing pipeline, sometimes to such an extent that a full field-of-view reconstruction is not possible [23]; this may be the reason why some dual-domain approaches are trained separately [24], not end-to-end. To calculate the linearized covariance image, we do need end-to-end gradient propagation in both directions even though the training may be done in a different manner.

In Appendix II, we provide Algorithm 3 for calculating the matrix-vector product in the form of $F_0\Sigma_Y F_0^t v$. This algorithm can be plugged into (18b) directly for the covariance image using FBPConvNet; it also includes a modular component for $F_0^t v$, *i.e.*, backpropagating v through FBP which may be useful for other structures of $\mathcal{F}$.

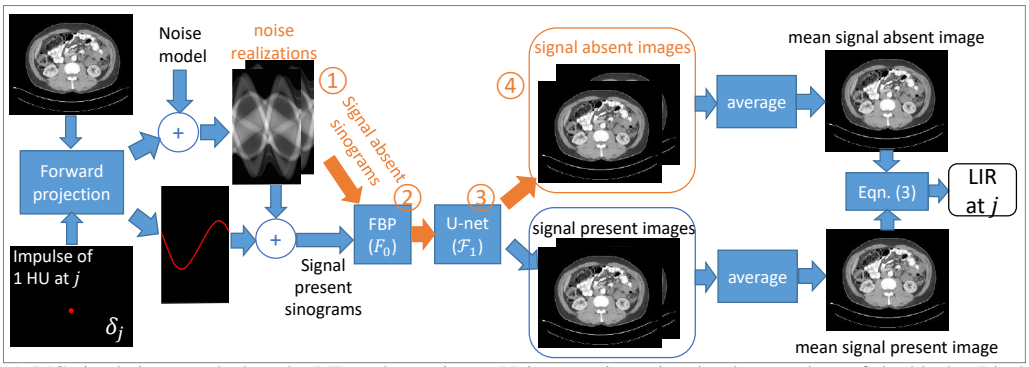
Algorithm 2: Calculate FBPConvNet covariance (17).

Input: Σ_Y, $\bar{Y}$, δ_j, FBPConvNet $\mathcal{F} = \mathcal{F}_1 \circ F_0$
Output: $\overline{\mathrm{cov}}(x_j)$

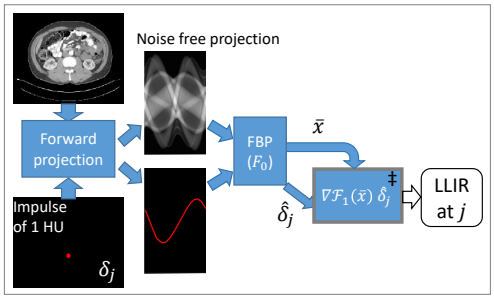
```
   /* FBP reconstruction of Ȳ to obtain input to F1             */
1  w = F0(Ȳ)
   /* Gradient backtracking wrt watched variable                */
2  with tf.GradientTape () as g1:
3      g1.watch (w)
4      x = F1(w)
   /* ∇_w(x · d) is available, ∀d – applied to δ_j on lines 7-8 */
   /* lines 5-8 calculate (18a) and prepare for (18c)           */
5  with tf.GradientTape () as g2:
6      g2.watch (δ_j)
7      a = g1.gradient (x, w, \
8              output_gradients = δ_j)
                                   /* a = G^t δ_j               */
   /* line 9 calculates (18b)                                   */
9  b = F0 Σ_Y F0^t a  using Alg 3 in Appx II
   /* compute (18c) with gradient (matrix) transpose, cf Alg 1  */
10 cov(x_j) = g2.gradient (a, δ_j, \
11         output_gradients = b)
                                   /* cov(x_j) = G b            */
```

With the help of Algorithm 3, the pseudocode for calculating (17) is provided in Algorithm 2, where lines 1 - 8, line 9, and lines 10-11 correspond to the three steps in (18), respectively. Note that step 3 (18c) can be achieved by plugging in Algorithm 1 with appropriate variable substitutions. In Algorithm 2, this is combined with step 1 (18a) to reuse computations.

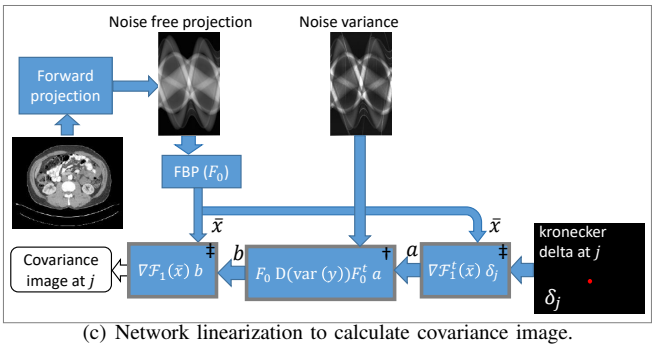
We end this section by reiterating the following impor-



(a) MC simulation to calculate the LIR and covariance. Noise covariance is related to a subset of the blocks. Blocks ①–④ yield samples of the DNN output from which covariance images are estimated.



(b) Network linearization to calculate the LLIR.



(c) Network linearization to calculate covariance image.

Fig. 2. Diagrams of the computational procedure for (a) nonlinear evaluation of LIR and noise covariance using MC of multiple noise realizations, and for calculating the LLIR (b) and noise covariance image (c) using network linearization. See text and Table I for the computational cost of the blocks marked with †, ‡. Multiple $\boldsymbol{\delta}_j$ can be used to compute LLIR and noise covariance at multiple locations simultaneously, provided that these locations are well separated and do not interfere with each other.

tant points: (1) forward gradient propagation can be done using gradient backpropagation only; and (2) calculating the linearized LIR and covariance requires end-to-end gradient propagation in both the reverse and the forward directions.

D. Computational cost

The computational procedures for 1) calculating the (nonlinear) LIR and noise covariance image using MC, and 2) calculating LLIR and noise covariance image using network linearization, for FBPConvNet are illustrated in Fig. 2.

There are two major differences in terms of computation between MC and network linearization. First, linear characterization of either the LLIR or the noise covariance image does not involve multiple noise realizations; we only need to know the noise properties (mean, variance, covariance) of the U-Net input, which for FBPConvNet is the FBP reconstructed image. This makes data preparation almost trivial for network linearization compared to MC.

Second, MC simulation involves running many forward passes through the DNN (block ③); in network linearization these forward passes are replaced by a few gradient passes (those blocks marked with †, ‡). It is well known that the complexity of gradient evaluation is proportional to that of function evaluation, with a small (4-5) constant factor, see, *e.g.*, [25, chapter 3]. Assuming a factor of 5, Table I lists the breakdown of computations in both MC and DL linearization. It can be seen that the computation cost of linearization is similar to running a MC simulation with 10 noise realizations, which is much lower than what is typically needed for MC.

TABLE I
COMPUTATIONAL COST FOR MC WITH N NOISE REALIZATIONS AND DL LINEARIZATION IN FBPCONVNET.

	MC	linearization	
		LLIR	covariance image
# FBP	N	1	$1 + 2^{\dagger}$
# DL eval.	N	5	$2 \times 5^{\ddagger}$

† Covariance matrix-vector product costs about 2 FBP.
‡ Gradient amounts to at most 5 function evaluations.

IV. SIMULATION STUDIES

A. Data generation

We generated fanbeam CT data using the Pancreas-CT data set [26], which consisted of ∼18,000 CT images from 82 patients. We used a distance-driven projector [27] and calculated the noise-free line integrals $\bar{y}_{v,i} = [\bar{A}\theta]_{v,i}$ through the CT image θ for each view $v = 0, \cdots, 1159$ and channel $i = 0, \cdots, 671$. The noisy transmission data $z_{v,i}$ followed the Poisson distribution with mean $\bar{z}_{v,i} = I_0 B_i e^{-\bar{y}_{v,i}}$, where I_0 represents the exposure level, B_i denotes the body bowtie filter profile, the noisy line integrals were obtained as $y_{v,i} = \log(I_0 B_i / z_{v,i})$.

To evaluate the extent to which network linearization is applicable, we simulated three exposure levels by adjusting

I_0. For exposure level 1, the combined effect of $I_0 B_i$ varied between 2×10^7 for the center channels to 4×10^6 for the edge channels; for the next two lower exposure levels, we decreased I_0 tenfold each to obtain $I_1 = I_0/10$ and $I_2 = I_0/100$. We discuss in Appendix I how the exposure value I_0 was determined to match the normal clinical settings for abdominal CT imaging. Reconstructed images (of size 512 $\times$ 512) using the FBP algorithm [28] from the full-view noise-free line integrals $\{\bar{y}_{v,i}\}, v = 0, \cdots, 1159$, and the sparse-view noisy line integrals $\{y_{8v',i}\}, v' = 0, \cdots, 144$, *i.e.*, with a subsampling factor of 8, were used to form paired labels and inputs for DL training. Sample region-of-interest (ROI) inputs and the noise-free training labels can be found in Figs. 5 and 9 for the different exposure levels. Additional image samples are provided online.

Our implementation was based on TensorFlow version 2.1 with an Nvidia Quadro P5000 GPU with 16 GB memory. We employed the U-Net as in the original FBPConvNet [12] (with the same network hyperparameters, *i.e.*, the convolution kernel size, the number of channels, etc) for artifacts removal from sparse-view FBP images. Among the 82 patients in Pancreas-CT, the first 72 were used for network training, and the remaining 10 for testing. We separately trained three U-Nets, one for each exposure level, using the mean-squared error (MSE) loss. The exposure levels only affected the U-Net input, the training labels for the three U-Nets were the same FBP reconstruction of noise-free full view line integrals.

We used randomly cropped patches from FBP images for training, and found that gradually increasing the patch size helped removing the sparse view artifacts. Our results were obtained by training with patch size 64 $\times$ 64 for 600 epochs, with 128 $\times$ 128 for 200 epochs, and finally with 256 $\times$ 256 for 200 epochs. The trained U-Nets together with the code for network linearization will be made publicly accessible.

The basic operations in the U-Net includes 2-D convolution, batch normalization (BN), and ReLU activation; these operations are repeated in a multi-scale manner through max-pooling (to increase scale) and upsampling (to decrease scale). As such, several components contribute to the nonlinearity of the U-Net, including BN, ReLU, and maxpooling.

B. Characterization of noise covariance and resolution

One test case (case 2) among the 10 test cases is shown in Fig. 3. We calculate the LIR and the covariance image at the center of each of the four ROIs (size 64 $\times$ 64), which differ substantially in their background anatomies. Additional data from a different test case (case 10) are provided online.

1) Nonlinear evaluation: Monte Carlo (MC) simulations with multiple noise realizations were used to estimate the noise and resolution properties, see Fig. 2(a). For noise covariance characterization, the simulated noisy fanbeam projection data $\{y^{(s)}_{8v',i}\}$, where s denotes the noise realization index, were reconstructed by FBP, the results of which were passed through the trained U-Net, and the U-Net output were saved for noise covariance calculation as detailed in Section II-C for up to 10,000 noise realizations. This process was repeated for the three trained U-Nets at the three exposure levels.

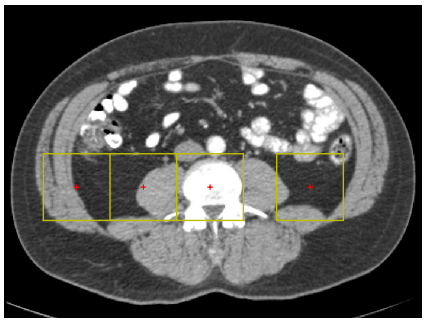


Fig. 3. The four ROIs of size 64 $\times$ 64 in one test case (case 2). The ROIs are overlaid on the ground truth label. For each ROI, we calculate the (linearized) LIR at the center pixel (red '+') of the ROI, and noise covariance between the center pixel and the ROI. Display (C, W) = (26, 400) HU.

For resolution characterization, we followed the recipe outlined in Section II-A. An impulse $\epsilon\delta_j$ of intensity ϵ = 1 HU at the center of each ROI was forward projected using the same fanbeam geometry, the sparse-view delta projection was added to the noisy projection of the patient, the noisy projection with and without the delta insertion were reconstructed by FBP and then passed through the trained U-Net. The LIR at the center of each ROI was estimated using eqn. (3) with up to 1000 noise realizations. This process was again repeated for the three U-Nets at the three exposure levels.

2) Linear approximation: The LLIR was calculated according to Section III-B, where the FBP reconstruction of the unit-input projection was forward propagated through the U-Net, with the U-Net gradient evaluated at the $F_0(\bar{Y})$ using the sparse-view data ($\bar{Y} = \{\bar{y}_{8v',i}\}$) (cf. (15)).

The linearized noise covariance image was calculated according to Algorithm 2 in Section III-C, where the gradient $\nabla_x \mathcal{F}_1(x)$ was again evaluated at $x = F_0(\bar{Y})$.

V. Experimental Results

The goal of our experimental studies was to investigate the validity, accuracy, and robustness of network linearization. We tested on FBPConvNet using both a small scale and a large scale data set under a normal exposure setting. DL linearization accuracy was also evaluated under two lower exposure settings. The robustness of network linearization was evaluated by applying the pre-trained U-Net to real CT scans where the required statistical properties were estimated using a plug-in approach. Finally, we studied effect on DL linearization using an alternative loss function for U-Net training.

A. Image denoising and artifacts removal performance

The performance of the trained U-Nets were evaluated using RMSE and SSIM [29], with the FBP reconstruction of noise-free full-view data as the reference. These FOM values for all 10 test cases at three exposure levels are shown in Fig. 4. It is clear that the trained U-Nets are effective in reducing noise and streak artifacts; there is a marked improvement in terms of RMSE and SSIM for all three exposure levels. Comparing the different exposures, the U-Net performance at the lower exposures is inferior to that at the higher exposures

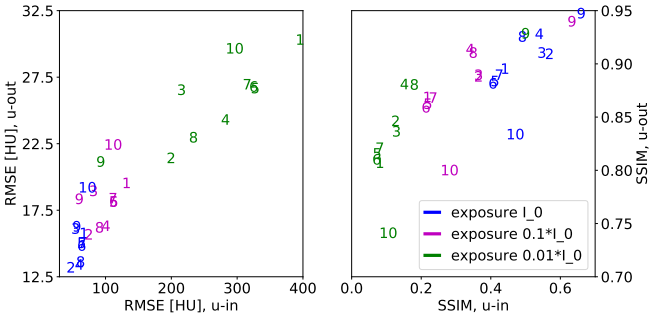


Fig. 4. Denoising performance. The test case indices (1, · · · , 10) are placed at the corresponding locations of U-Net input/output RMSE (or SSIM) values. Different colors represent different exposures.

in terms of both FOMs, and the loss of details in the images[3] is more pronounced at lower exposures. This is reasonable as the FBP inputs to the U-Net at lower exposures were severely contaminated by high noise and artifacts.

B. Network linearization at the exposure I_0 – small scale

Figs. 5 – 7 show the results at the exposure level I_0 for test case 2. The subfigures of Fig. 5 are: (a) ROIs showing the 3-tuple of (the ground truth, the input, and the output of U-Net), and (b) the LIR using MC and the LLIR using network linearization. Similar to the arrangement of Fig. 5(b), Fig. 6 shows the noise covariance image using MC and network linearization. In both Fig. 5(b) and Fig. 6, the row headings are the number of noise realizations used in the MC or 'lin' meaning network linearization. The footings denote either the CRC for the (L)LIR or the noise standard deviation in HU at the center pixel. As the (L)LIR has a limited spatial extent, for better visualization Fig. 5(b) focuses on the central 32 $\times$ 32 ROI outlined by the yellow boxes. For a quantitative comparison, the horizontal and vertical profiles through the center pixel for the bottom 2 rows of Fig. 5(b) and Fig. 6 are shown in Fig. 7.

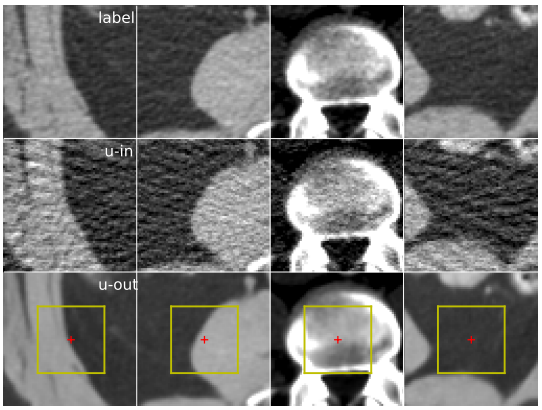
For the MC simulations in Fig. 5, the (unit-less) CRC, calculated using (4), is a random variable and is subject to noise fluctuations. As the number of MC noise realizations increased, the CRC values stabilized. We considered a maximum number of 1000 noise realizations for the LIR as changes were negligible beyond this number. Similar considerations also apply to the noise covariance images in Fig. 6.

At the exposure level I_0, the LLIR well captures the shape and the spatially variant nature of the LIR for a range of anatomical backgrounds; the CRCs also closely match those from MC simulations (Fig. 5(b)). Similarly, the covariance images calculated using network linearization also closely approximates those from MC simulations (Fig. 6). The central profiles in Fig. 7 confirm these statements.

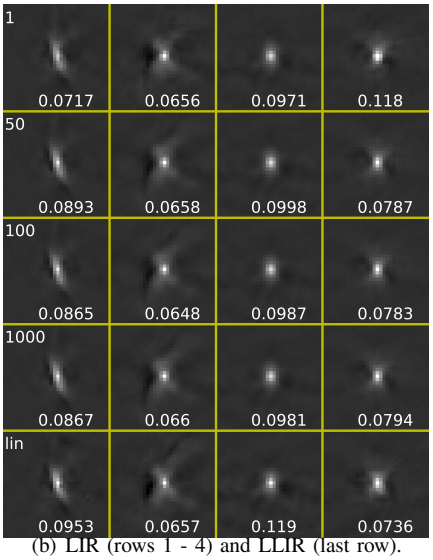
C. Network linearization at the exposure I_0 – large scale

We then performed a large scale test to further validate the performance of network linearization. We randomly selected

[3]The image data from which the FOMs in Fig. 4 are summarized are available from J Xu's github page (https://tinyurl.com/4sft28vn).



(a) Image samples. Different columns are different ROIs.



(b) LIR (rows 1 - 4) and LLIR (last row).

Fig. 5. Comparison of network linearization with MC simulations at exposure I_0, test case 2. The four columns are for each of the four ROIs. (a) FBP reconstruction using full view (1160) noise-free data (row 1, ground truth); using sparse view (145) noisy data (row 2, U-Net input); and the U-Net output (row 3). The '+' sign marks the locations where the (L)LIR and noise covariance images are calculated. Same display window as in Fig. 3 except for the 3rd column, where the window is centered on 226 HU for better visualization. (b) The different rows are: the LIR calculated using MC simulation of 1, 50, 100, and 1000 noise realizations, and LLIR calculated using network linearization. The text annotations are the CRC at the '+' pixel. The yellow boundaries correspond to the yellow box outline in (a).

50 patient images among all test cases, and for each patient image we applied a bounding box of size 256 $\times$ 256 to cover the bulk part of the patient anatomy (Fig. 8). The bounding box was then divided into 16 (= 4 $\times$ 4) non-overlapping ROIs each of size 64 $\times$ 64, for which we calculated the covariance images at the center using both linearization and MC with up to 10k (MC-10k) noise realizations. One example among the 50 cases is shown in Fig. 8(a), together with the calculated covariance images using linearization and MC-10k.

Using the covariance image from MC-10k as the reference, we calculated the PSNR of the covariance image using linearization, and MC with 500, 1000, and 2000 noise realizations (MC-500, MC-1000, MC-2000, respectively). The 800 PSNR values (50 patients $\times$ 16 ROIs) for each of the 4 methods are shown as pairwise histograms in Fig. 8(b).

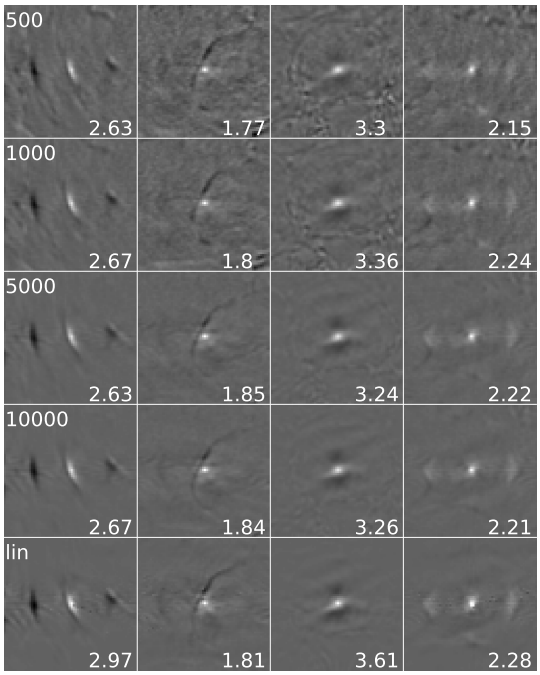


Fig. 6. Comparison of covariance image using network linearization with MC simulations at exposure $\boldsymbol{I_0}$, test case 2. The columns are for different ROIs. The row headings are the number of noise realizations in MC simulations, *i.e.*, 500, 1000, 5000, 10000 noise realizations, or "lin" which means network linearization. The text annotations are the noise standard deviation in HU at the '+' pixel of Fig. 5(a).

Compared to MC-500 or MC-1000, the PSNR histogram of linearization is shifted to the right and is well separated from that of MC-500 and MC-1000, indicating higher accuracy for network linearization. On the other hand, the histograms of linearization and MC-2000 highly overlap, meaning that the accuracy of linearization is comparable to MC with 2000 noise realizations. The summary statistics in Table II show that network linearization has significantly higher PSNR compared to MC with 500, 1000, and 2000 noise realizations.

TABLE II
PSNR COMPARISON BETWEEN NETWORK LINEARIZATION AND MC WITH DIFFERENT NOISE REALIZATIONS.

method 1	method 2	diff. mean PSNR	joint 90% CI*
MC-500	network lin.	7.618	[7.368, 7.868]
MC-1000	network lin.	4.389	[4.139, 4.638]
MC-2000	network lin.	0.966	[0.713, 1.219]

*Confidence interval (CI) adjusted for multiple comparisons using Bonferroni correction.

D. Network linearization at the lower exposures

Fig. 9 shows the change of the LIR and the covariance image for ROI 2 of test case 2 as the exposure level is vastly reduced. The text annotations and their meanings are similar to those in Fig. 5 and 6. An immediate observation is that as the exposure level decreases, network linearization eventually loses its approximation accuracy, similar to what has been observed in linearization techniques applied to MBIR [7]. More specifically, at reduced exposures, the LIRs from MC have a larger spatial extent,[4] and correspondingly a lower CRC at the center pixel. This can be attributed to the more nonlocal, nonlinear behavior of the U-Net to achieve noise and artifacts reduction. Network linearization does not capture the larger spatial extent of LIR, and over-estimates the CRC. Similarly, network linearization under-estimates the correlation range and over-estimates the noise standard deviation at the vastly reduced exposure levels; the extent of under- (over-) estimation also correlates with the exposure level.

In Fig. 9, the linearized covariance images at the lower exposure level (lower right corner) appear to have spike noise artifacts. These artifacts are caused by the degraded approximation accuracy of linearization. Fig. 10 shows both the zeroth order and the first order (linear) approximation together with the network input and the nonlinear network output. The nonlinear behavior of the trained U-Net is able to produce artifacts-free output (Fig. 10(b)), but the linearized output (Fig. 10(d)) cannot fully capture the nonlinear U-Net behavior, containing residual sparse-view and noise artifacts. The linearized covariance images are the covariance of the linearized output. The residual noise artifacts in the linearized output can be averaged out from multiple noise realizations, but the residual sparse-view artifacts may not, causing artifacts in the linearized covariance.

E. Plug-in evaluation

The previous results confirmed the validity and accuracy of network linearization under normal exposure settings, so that efficient noise/resolution characterization of DL images is feasible. But as shown in Fig. 2(b) and (c), network linearization requires statistical properties such as the mean for the linearization point in the Taylor expansion. In practice, these statistical properties are not available and need to be estimated from the real data.

A similar situation also occurs in MBIR resolution and noise covariance estimation. There the required statistical properties can be approximated using a plug-in approach [6], *e.g.*, replacing the signal mean by one noise realization. Here we evaluate if a similar plug-in approach can be applied to DNN and obtain informative results.

1) Data acquisition: In our previous studies [30], [31], we acquired projection data from a QRM phantom on a Siemens Sensation 64 CT scanner. The scan was repeated 186 times using 2 $\times$ 1 mm collimation in the same fan-beam geometry as we used in our simulations. The size of the projection data was 672 $\times$ 1160. The x-ray tube setting was 25 mAs and 120 kVp. Other details about data acquisition can be found in the original publications [30], [31].

2) Data preprocessing: The 2-slice CT data were averaged to obtain a single-slice fanbeam projection. The projection data were view-wise subsampled by a factor of 8 to obtain sparse-view data of size 672 $\times$ 144, and then reconstructed using our in-house FBP algorithm to obtain 186 noise realizations.

The QRM phantom data were acquired at a low dose setting. To avoid retraining the U-Net at the normal exposure, we

[4]Unlike in Fig. 5 where we focus on the central 32 $\times$ 32 pixels for better visualization, in Fig. 9 the (L)LIRs are shown in the full 64 $\times$ 64 size.

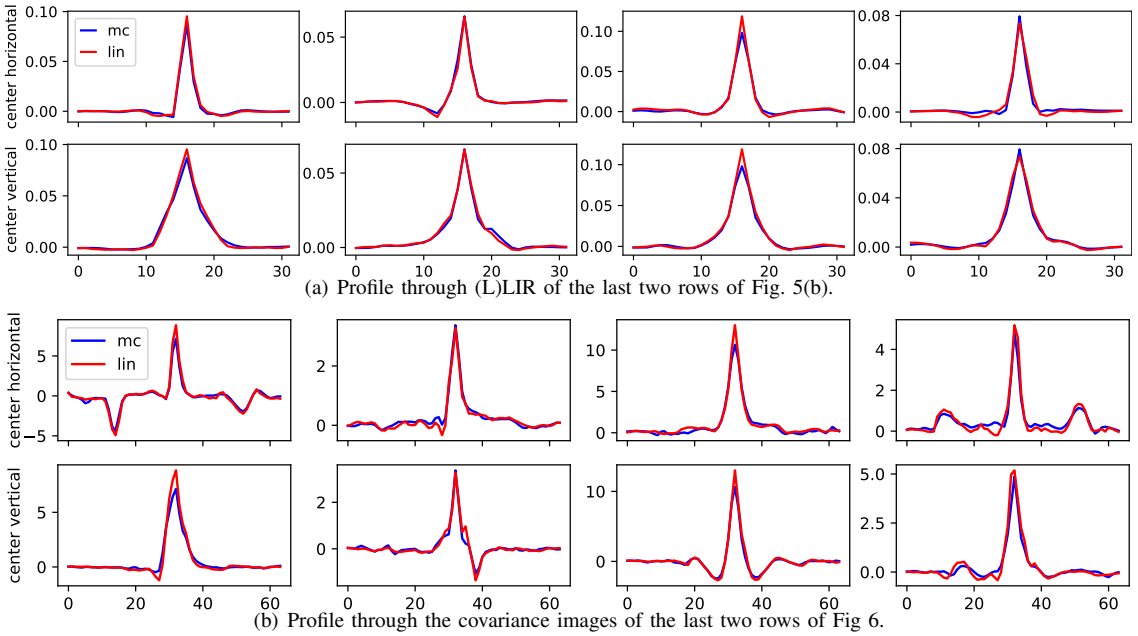


Fig. 7. Patient case 2, exposure $\boldsymbol{I_0}$. The columns correspond to the ROIs of Fig. 5(a). (a) Center horizontal and vertical profiles through the LIR using MC (1000 noise realizations) and network linearization. The peak values are contrast recovery coefficients annotated in Fig. 5(b). (b) Center horizontal and vertical profiles through the noise covariance images using MC (10000 noise realizations) and network linearization. The peak values are the noise variances, *i.e.*, the square of the standard deviation values annotated in Fig. 6. The horizontal axis is the pixel index.

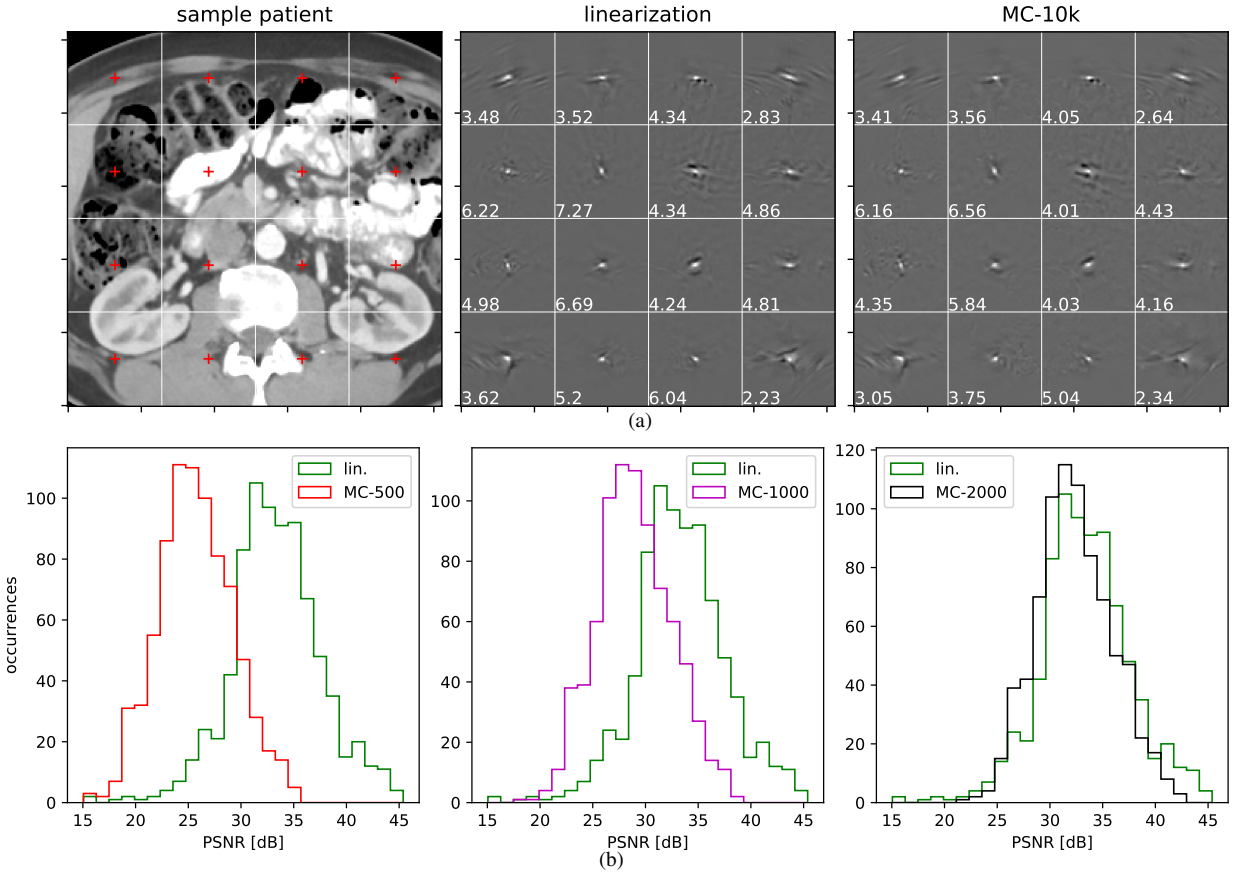


Fig. 8. (a) Left: one sample patient image (among 50 samples) with 16 ROIs each of size 64 $\times$ 64 pixels covering the patient body; the red crosses mark the center of each ROI. The middle and right panels are the covariance images at the center location calculated using linearization and MC with 10k noise realizations (MC-10k) respectively. (b) Using the covariance image from MC-10k as the reference, the PSNR was calculated for MC of noise realizations 500, 1000, 2000, and linearization, for all 16 $\times$ 50 (= 800) ROIs. These PSNRs are shown as histograms comparing linearization and MC with different numbers of noise realizations.

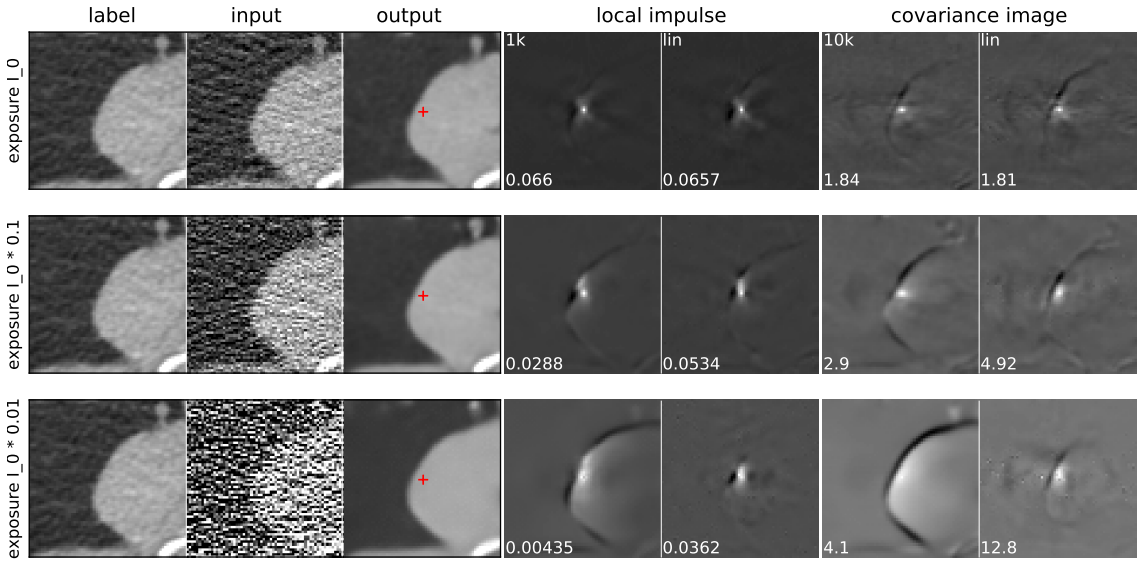


Fig. 9. Test case 2 at different exposures. As the exposure level is vastly reduced, the LIR becomes more spatially extended and the noise correlation range becomes larger. Network linearization under-estimates the spatial extent of the LIR and the covariance image, and over-estimate the CRC and the noise standard deviation.

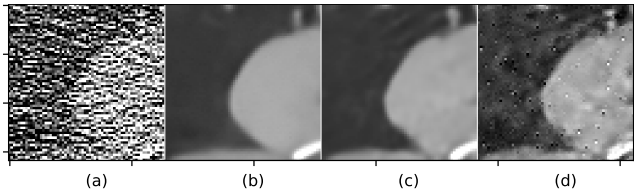


Fig. 10. Artifacts in the linearized covariance image originate from the approximate nature of linearization. (a) U-Net input, $\boldsymbol{X} \triangleq \boldsymbol{F_0}(\boldsymbol{Y})$, *i.e.*, FBP reconstructed noisy sparse-view data; (b) the nonlinear U-Net output $\boldsymbol{\mathcal{F}_1}(\boldsymbol{X})$ is free of artifacts; (c) the zeroth order approximation $\boldsymbol{\mathcal{F}_1}(\bar{\boldsymbol{X}})$; (d) the first order approximation $\boldsymbol{\mathcal{F}_1}(\bar{\boldsymbol{X}}) + \boldsymbol{\nabla \mathcal{F}_1}(\bar{\boldsymbol{X}})(\boldsymbol{X} - \bar{\boldsymbol{X}})$ contains residual artifacts. Display (C, W) = (26, 400) HU.

averaged every 5 noise realizations in order to match the noise level of the input to the pre-trained U-Net. Therefore in total we had 37 independent noise realizations for the real data. One such noise realization before and after U-Net processing is shown in Fig. 11. It can be seen that the U-Net removed most of the sparse-view artifacts, despite the fact that the U-Net had been trained using only Pancreas-CT data.

We applied network linearization using one noise realization; and as a reference for comparison, we also calculated their nonlinear counterparts according to Fig. 2(a) using the remaining 36 noise realizations. The details are provided below for (L)LIR and noise covariance separately.

3) LLIR estimation using plug-in: To apply the plug-in estimation for LLIR, the noise-free projection in Fig. 2(b) was replaced by one noise realization; the resulting plug-in LLIR is shown in Fig. 12(c), together with the ROI images and the estimated nonlinear LIR using the 36 remaining noise realizations. The plug-in LLIR agrees well with the nonlinear LIR, both qualitatively in terms of the shape of the impulse response, and quantitatively in terms of the CRC.

4) Covariance image estimation using plug-in: As shown in Fig. 2(c), linearized noise covariance estimation requires both the mean and the variance of the projection data. We substituted one noisy projection data for the mean projection, and estimated the projection noise variance using the remaining 36 noise realizations.[5]

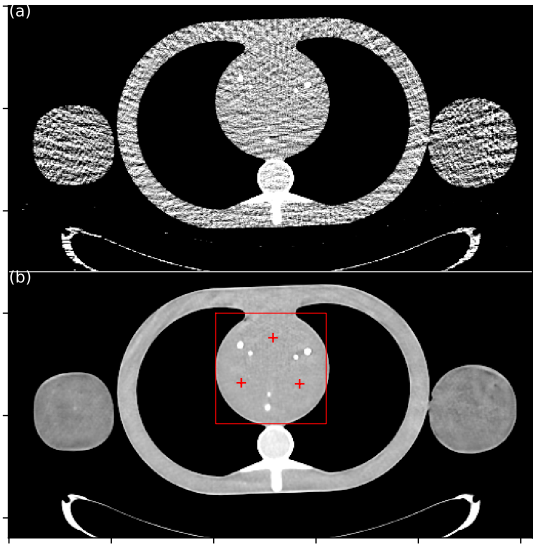


Fig. 11. (a) FBP reconstructed image from 144-view QRM phantom data, used as input to the pre-trained U-Net. (b) U-Net output. The red box is the ROI (size 108 × 108 pixels) where covariance image quantification was carried out; the "+" signs mark the three locations for LIR quantification. Display (C, W) = (10, 300) HU.

To create a meaningful MC-based reference following Fig. 2(a), the 36 noise realizations, however, were not sufficient. To effectively increase noise realizations, we subdivided the 108 × 108 ROI shown in Fig. 11(b) into a 9×9 grid of sub-ROIs each of size 12 × 12. Among the 81 sub-ROIs, those with background non-uniformities (red indices in Fig. 13(a), total 36 of them) were unselected; while the remaining 45 uniform sub-ROIs each with 36 noise realizations were treated

[5]More on variance estimation shortly.

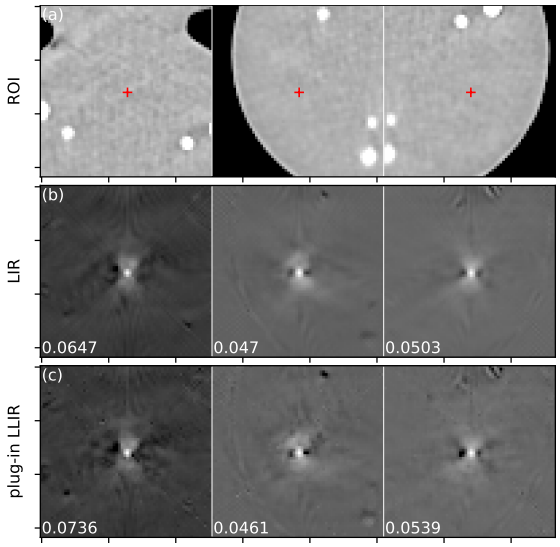


Fig. 12. (a) The ROIs of size 64 × 64 centered on the "+", the same as those in Fig. 11(b). Display (C, W) = (10, 300) HU. (b) The LIR from 36 noise realizations. (c) The plug-in LLIR calculated using 1 noise realization. In both (b) and (c), the text annotations are the CRC at the ROI center "+".

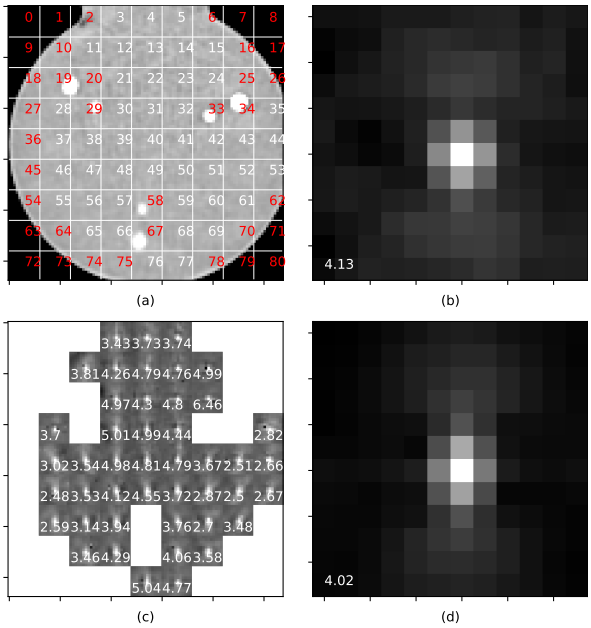


Fig. 13. (a) The 108 × 108 ROI of Fig. 11(b) was divided into 81 sub-ROIs each of 12 × 12, to effectively increase the number of noise realizations. Sub-ROIs of uniform background (white indices, total 45) were included. (b) The covariance image, size 12×12, estimated using 36×45 noise realizations. (c) The plug-in covariance image at the selected sub-ROIs. (d) The average of the 45 plug-in covariance images. The text annotations in (b)-(d) are the standard deviation in HU at the center pixel.

as 36 × 45 noise realizations of one image, from which the noise covariance image in Fig. 13(b) was estimated.

We then calculated the linearized covariance images, according to Fig. 2(c), for each of the 45 uniform sub-ROIs as shown in Fig. 13(c). The average of the 45 linearized covariance images is given in Fig. 13(d).

From Fig. 13, we observe again general agreement in terms of both the shape and the standard deviation at the center pixel in the covariance image between linearization and MC.

Ideally, to apply the plug-in approach to the linearized covariance, one should not estimate the variance of projection data using multiple noise realizations. If the air scan were available, then the variance of the projection can be estimated from a single noise realization, based on the transmission domain Poisson statistics (cf. Section IV-A). As our QRM phantom data were acquired for a different study, air scan was not available. This limitation necessitated the alternative we used, which was estimating the projection variance using multiple scans. Nevertheless, our results are encouraging and suggest that DL linearization can be applied to real data in a reliable manner.

F. An alternative training loss

The original FBPConvNet was trained using the MSE loss. We also investigated if an alternative, nonsmooth, training objective would affect DL linearization. For this purpose, we retrained the same U-Net using the L_1 loss. We then ran the same test case in Fig. 8(a) according to Fig. 2 through the retrained U-Net. Shown in Fig. 14 are the linearized covariance images for the 16 ROIs, and the corresponding nonlinear covariance images estimated using 10000 noise realizations (MC-10k). Using MC-10k as the reference, we calculated the PSNR of linearization and various numbers of MC noise realizations. These PSNR values are summarized in Table III by the mean and the standard deviation based on the 16 ROIs.

TABLE III
PSNR COMPARISON FOR L_1 TRAINED U-NET.

MC-500	MC-1000	MC-2000	linearization
26.77 ± 2.10	30.01 ± 2.08	33.64 ± 2.04	31.92 ± 3.40

Both the covariance images and the summarized PSNRs demonstrate that DL linearization also works well for the L_1 trained U-Net. From Fig. 14, there is agreement in both the shape of the covariance image and the standard deviation at the center pixel. Regarding Table III, statistical tests comparing linearization versus (a) MC-500, (b) MC-1000, and (c) MC-2000 showed that, after Bonferroni correction for multiple comparisons, the PSNR differences between linearization and MC-500 or MC-1000 were significant ($p < 0.1$), while with MC-2000 it was not.

We can also compare Fig. 14 with Fig. 8(a) as both were calculated for the same test case. Some similarities in the general shape of covariance images can be observed, which can be attributed to the underlying anatomy. The differences between them come solely from the different U-Net parameters determined during the training process, as both U-Nets have the same network structure. More quantitative statements as to the effect of the different loss functions on resolution and noise would require more in-depth investigations.

VI. DISCUSSION

It should not be surprising that the performance of network linearization is related to the noise/exposure level. As our network linearization technique applies the first order Taylor

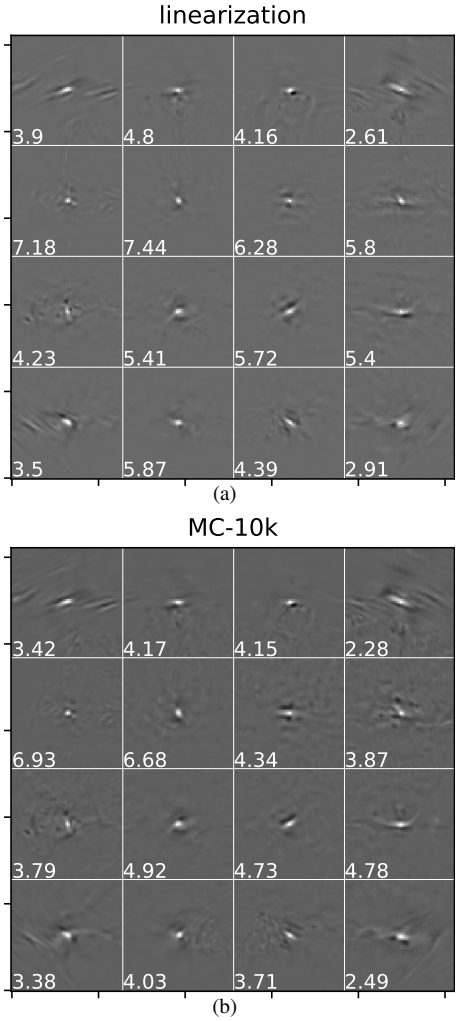


Fig. 14. Effect of $\boldsymbol{L_1}$ loss function for U-Net training on network linearization. The linearized covariance image (a) and those from MC with 10000 noise realizations (b) of the same example as in Fig. 8(a).

expansion at the mean of the network input (cf. (11)), as long as the noise deviation is small, network linearization is guaranteed to work well. This last statement has been confirmed by numerous experiments in this work.

Our study opens many opportunities for network linearization to characterize and quantify DL performance. Many DL applications work with normal exposure data, including those for low dose applications. For example in sparse view reconstruction, dose reduction is achieved with reduced data acquisition. At the exposure level I_0 in our set up, network linearization works very well with a dose reduction factor of 8 from sparse-view sampling, which is a major dose saving.

We used FBPConvNet as an example of a DL network $\mathcal{F}$ to develop network linearization and demonstrate its performance. The nonlinear modules in FBPConvNet such as ReLU activation, maxpooling, and batch normalization are common components in many DNNs. Therefore network linearization can be easily applied to other DNNs or even more versatile architectures. Our development is the first *scalable* and *generalizable* approach for efficient characterization of image noise and resolution.

In terms of implementation, the only requirement of our method is that gradient propagation through the DNN can be performed *end-to-end* in *both* directions using either autodiff or analytic calculation. The required forward gradient propagation through standard DL modules can be performed using reverse-mode autodiff. As such, network linearization is no more difficult than gradient calculation for network training.

VII. Conclusions

We applied the first order Taylor series expansion to DNNs, which we call network linearization, to evaluate the noise and resolution properties of DL image generation. At normal exposure level, network linearization can obtain accurate and efficient characterization of image resolution and noise covariance properties without running MC simulations. As the exposure level is reduced, network linearization eventually loses its accuracy due to its first order approximation nature.

This work is a feasibility study on DL linearization to characterize image noise and resolution. Future investigations may study how these image properties are affected by different network architectures, different applications (*e.g.*, super-resolution, metal artifacts reduction), or different loss functions [32], [33]. Another direction of research is to use the resolution/noise properties as criteria to choose the hyperparameters in the loss function for network training. Finally, it would also be interesting to see if the efficiently computable resolution/noise metrics can be incorporated directly into the loss function for network fine-tuning so as to provide more prospective resolution and noise control.

Appendix I
Determination of exposure level I_0

We used [16] as a reference to determine a value for exposure per view that is representative of a normal dose abdominal CT scan. Specifically, I_0 was such that the noise level for test case 10, measured in a central circular ROI of FBP reconstruction (no apodization) of the noisy, full-view projection data, matched the noise level of 10 HU at the liver/diaphragm level reported in [16]. The noise levels for all 10 cases varied between 2.7 HU (case 9) and 14.0 HU (case 1). For test case 2 for which more experimental results were reported, the noise level at I_0 with full-view projection was 6.0 HU.

Appendix II
Calculation of $F_0\Sigma_Y F_0^t a$

Let F_0 be an FBP reconstruction algorithm, and Σ_Y the covariance matrix of the sinogram Y, then the covariance matrix of the FBP reconstructed image is given by $F_0\Sigma_Y F_0^t$. On line 9 of Algorithm 2, we need to compute the product of this matrix with an arbitrary vector a, *i.e.*, $b = F_0\Sigma_Y F_0^t a$. This calculation can be divided into three blocks: (1) apply the adjoint of FBP: $a_1 = F_0^t a$, (2) multiply with the covariance matrix of the sinogram data: $a_2 = \Sigma_Y a_1$; as Σ_Y is diagonal, this step amounts to an element-wise multiplication; and (3) do another FBP reconstruction, *i.e.*, $b = F_0 a_2$. Block (1), the adjoint of FBP, can be implemented as a simple adaptation of

Algorithm 3: Direct calculation of the product of the covariance matrix of FBP and an arbitrary vector.

Input: Vector $\boldsymbol{a} = \boldsymbol{a}[x, y]$ in the form of an image, the variance of projection $\mathrm{Var}[v, \gamma]$

Output: $\boldsymbol{b} = F_0 \Sigma_Y F_0^t \boldsymbol{a}$ in the form of an image, where F_0 is the FBP algorithm, and Σ_Y is the diagonal covariance.

/* adjoint of FBP, see [28] for more details on notation. */

1 **for** view $v = 1, \cdots, N_v$ **do**

2 **for** row $y = 1, \cdots, N_y$ **do**

3 **for** col $x = 1, \cdots, N_x$ **do**

4 $\gamma^* \leftarrow$ ray index $[x, y, v]$; $\gamma = \lfloor \gamma^* \rfloor$; $\alpha = \gamma^* - \gamma$

5 $w_{\mathrm{dist}} = \|s(v) - \underline{x}\|$ /* distance between x-ray source $s(v)$ and $\underline{x} = [x, y]$ */

6 $P_0[\gamma] \mathrel{+}= (1 - \alpha)\, \boldsymbol{a}[x, y] / w_{\mathrm{dist}}^2$

7 $P_0[\gamma + 1] \mathrel{+}= \alpha\, \boldsymbol{a}[x, y] / w_{\mathrm{dist}}^2$ /* adjoint of distance-square weighted backprojection */

8 $P_{0,f}[\gamma'] = \sum_\gamma h_f(\gamma - \gamma') P_0[\gamma],\ \forall \gamma'$ /* adjoint of ramp filter $h_f(\gamma)$ */

9 **for** channel $\gamma = 1, \cdots, N_c$ **do**

10 $P_1[v, \gamma] = w_{\cos}[\gamma]\, m[v, \gamma]\, P_{0,f}[\gamma]$ /* adjoint of cosine $w_{\cos}$ and redundancy weighting $m[v, \gamma]$ */

/* weighting by projection variance */

11 **for** $v = 1, \cdots, N_v$ **do**

12 **for** $\gamma = 1, \cdots, N_c$ **do**

13 $P[v, \gamma] = \mathrm{Var}[v, \gamma] * P_1[v, \gamma]$

/* do another FBP, see [28] for more details on notation. */

14 **for** $v = 1, \cdots, N_v$ **do**

15 **for** $\gamma = 1, \cdots, N_c$ **do**

16 $P_w[\gamma] = w_{\cos}[\gamma]\, m[v, \gamma]\, P[v, \gamma]$ /* cosine weighting $w_{\cos}$, redundancy weighting $m[v, \gamma]$ */

17 $P_f[\gamma] = \sum_{\gamma'} h_f(\gamma - \gamma') P_w[\gamma'],\ \forall \gamma$ /* ramp filter $h_f(\gamma)$ */

18 **for** $y = 1, \cdots, N_y$ **do**

19 **for** $x = 1, \cdots, N_x$ **do**

20 $\gamma^* \leftarrow$ ray index $[x, y, v]$; $\gamma = \lfloor \gamma^* \rfloor$; $\alpha = \gamma^* - \gamma$

21 $w_{\mathrm{dist}} = \|s(v) - \underline{x}\|$

22 $\boldsymbol{b}[x, y] \mathrel{+}= ((1 - \alpha) P_f[\gamma] + \alpha P_f[\gamma + 1]) / w_{\mathrm{dist}}^2$ /* distance-square weighted backprojection */

block (3) by reversing the orders of operations and swapping the left-hand side and the right-hand side for all calculations. The pseudo-code for this calculation using the variant of an FBP algorithm [28] is given in Algorithm 3. One point worth noting is on line 8, where we compute the adjoint of the ramp filtering operation. This step is the same as applying the ramp filter itself since the ramp kernel is symmetric. Covariance calculation for other versions of FBP, *e.g.*, with parallel-beam rebinnings or for 3D algorithms such as FDK, may be derived similarly.

Algorithm 3 is an *exact* algorithm for calculating the matrix-vector product; more specifically, block (1) of Algorithm 3, line 1-10, is the exact adjoint operator of FBP.

For the special case where $a = \delta_j$ is a unit-vector input, the output of Algorithm 3 is the covariance image at pixel j of FBP. Special-purpose algorithms for calculating such covariance images have been derived for specific variants of FBPs [28], [34]. On the other hand, Algorithm 3 (for the special case $a = \delta_j$) can be regarded as a *universal* approach to calculating the covariance image of a *generic* FBP algorithm. In other words, the adjoint operations can be applied easily to another variant of FBP, to easily instantiate an exact algorithm for covariance image calculation.

REFERENCES

[1] H. H. Barrett, "Objective assessment of image quality: effects of quantum noise and object variability," *JOSA A*, vol. 7, no. 7, pp. 1266–1278, 1990.

[2] J. Hsieh, *Computed Tomography: Principles, Design, Artifacts, and Recent Advances*. SPIE PRESS, 2015.

[3] K. Li, W. Zhou, H. Li, and M. A. Anastasio, "Assessing the impact of deep neural network-based image denoising on binary signal detection tasks," *IEEE Transactions on Medical Imaging*, vol. 40, no. 9, pp. 2295–2305, 2021.

[4] Z. Yu, M. A. Rahman, T. Schindler, R. Gropler, R. Laforest, R. Wahl, and A. Jha, "AI-based methods for nuclear-medicine imaging: Need for objective task-specific evaluation," *Journal of Nuclear Medicine*, vol. 61, no. 1, pp. 575–575, 2020.

[5] J. A. Fessler and W. L. Rogers, "Spatial resolution properties of penalized-likelihood image reconstruction: space-invariant tomographs," *IEEE Transactions on Image processing*, vol. 5, no. 9, pp. 1346–1358, 1996.

[6] J. Qi and R. M. Leahy, "Resolution and noise properties of MAP reconstruction for fully 3-D PET," *IEEE transactions on medical imaging*, vol. 19, no. 5, pp. 493–506, 2000.

[7] S. Ahn and R. M. Leahy, "Analysis of resolution and noise properties of nonquadratically regularized image reconstruction methods for PET," *IEEE transactions on medical imaging*, vol. 27, no. 3, pp. 413–424, 2008.

[8] H. Chen, Y. Zhang, M. K. Kalra, F. Lin, Y. Chen, P. Liao, J. Zhou, and G. Wang, "Low-dose CT with a residual encoder-decoder convolutional neural network," *IEEE transactions on medical imaging*, vol. 36, no. 12, pp. 2524–2535, 2017.

[9] Y. Li, K. Li, C. Zhang, J. Montoya, and G.-H. Chen, "Learning to reconstruct computed tomography images directly from sinogram data

under a variety of data acquisition conditions," *IEEE transactions on medical imaging*, vol. 38, no. 10, pp. 2469–2481, 2019.

[10] H. K. Aggarwal, M. P. Mani, and M. Jacob, "Modl: Model-based deep learning architecture for inverse problems," *IEEE transactions on medical imaging*, vol. 38, no. 2, pp. 394–405, 2018.

[11] J. Adler and O. Öktem, "Learned primal-dual reconstruction," *IEEE transactions on medical imaging*, vol. 37, no. 6, pp. 1322–1332, 2018.

[12] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, 2017.

[13] Q. Zhang, Z. Hu, C. Jiang, H. Zheng, Y. Ge, and D. Liang, "Artifact removal using a hybrid-domain convolutional neural network for limited-angle computed tomography imaging," *Physics in Medicine & Biology*, vol. 65, no. 15, p. 155010, 2020.

[14] A. Zheng, H. Gao, L. Zhang, and Y. Xing, "A dual-domain deep learning-based reconstruction method for fully 3D sparse data helical CT," *Physics in Medicine & Biology*, vol. 65, no. 24, p. 245030, 2020.

[15] T. Li, X. Li, J. Wang, J. Wen, H. Lu, J. Hsieh, and Z. Liang, "Nonlinear sinogram smoothing for low-dose X-ray CT," *IEEE Transactions on Nuclear Science*, vol. 51, no. 5, pp. 2505–2513, 2004.

[16] L. Yu, M. Shiung, D. Jondal, and C. H. McCollough, "Development and validation of a practical lower-dose-simulation tool for optimizing computed tomography scan protocols," *Journal of computer assisted tomography*, vol. 36, no. 4, pp. 477–487, 2012.

[17] J. Nuyts, B. De Man, J. A. Fessler, W. Zbijewski, and F. J. Beekman, "Modelling the physics in the iterative reconstruction for transmission computed tomography," *Physics in Medicine & Biology*, vol. 58, no. 12, p. R63, 2013.

[18] H. Schondube and F. Noo, "Statistically-efficient estimation of Hotelling observer performance with unknown means," in *Proceedings of the 4th International Conference on Image Formation in X-ray Computed Tomography*, pp. 379–382, 2016.

[19] A. B. Owen, *Monte Carlo theory, methods and examples*. 2013.

[20] J. Qi and R. M. Leahy, "A theoretical study of the contrast recovery and variance of MAP reconstructions from PET data," *IEEE transactions on medical imaging*, vol. 18, no. 4, pp. 293–305, 1999.

[21] J. H. Cho and J. A. Fessler, "Regularization designs for uniform spatial resolution and noise properties in statistical image reconstruction for 3-D X-ray CT," *IEEE transactions on medical imaging*, vol. 34, no. 2, pp. 678–689, 2014.

[22] H. R. Shi and J. A. Fessler, "Quadratic regularization design for 2-D CT," *IEEE transactions on medical imaging*, vol. 28, no. 5, pp. 645–656, 2008.

[23] J. Xu and F. Noo, "Patient-specific hyperparameter learning for optimization-based CT image reconstruction," *Physics in Medicine & Biology*, vol. 66, no. 19, p. 19NT01, 2021.

[24] X. Yin, Q. Zhao, J. Liu, W. Yang, J. Yang, G. Quan, Y. Chen, H. Shu, L. Luo, and J.-L. Coatrieux, "Domain progressive 3D residual convolution network to improve low-dose CT imaging," *IEEE transactions on medical imaging*, vol. 38, no. 12, pp. 2903–2913, 2019.

[25] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[26] H. Roth, A. Farag, E. B. Turkbey, L. Lu, J. Liu, and R. M. Summers, "Data From Pancreas-CT," 2016. Version Number: 2 Type: dataset.

[27] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," *Physics in Medicine & Biology*, vol. 49, no. 11, p. 2463, 2004.

[28] A. Wunderlich and F. Noo, "Image covariance and lesion detectability in direct fan-beam x-ray computed tomography," *Physics in Medicine & Biology*, vol. 53, no. 10, p. 2471, 2008.

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[30] A. Wunderlich and F. Noo, "Confidence intervals for performance assessment of linear observers," *Medical Physics*, vol. 38, no. S1, pp. S57–S68, 2011.

[31] A. Wunderlich, F. Noo, B. D. Gallas, and M. E. Heilbrun, "Exact confidence intervals for channelized Hotelling observer performance in image quality studies," *IEEE Transactions on Medical Imaging*, vol. 34, no. 2, pp. 453–464, 2015.

[32] Q. Yang, P. Yan, Y. Zhang, H. Yu, Y. Shi, X. Mou, M. K. Kalra, Y. Zhang, L. Sun, and G. Wang, "Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss," *IEEE transactions on medical imaging*, vol. 37, no. 6, pp. 1348–1357, 2018.

[33] J. M. Wolterink, T. Leiner, M. A. Viergever, and I. Išgum, "Generative adversarial networks for noise reduction in low-dose CT," *IEEE transactions on medical imaging*, vol. 36, no. 12, pp. 2536–2545, 2017.

[34] A. Wunderlich and F. Noo, "Exact and efficient computation of noise covariance for fan-beam fbp reconstructions that use rebinning to parallel-beam geometry," in *Medical Imaging 2012: Physics of Medical Imaging*, vol. 8313, pp. 617–625, SPIE, 2012.