QN-Mixer: A Quasi-Newton MLP-Mixer Model for Sparse-View CT Reconstruction

Supplementary Material

A. Inverse Hessian approximation

A.1. Approximation via optimization

The fundamental idea is to iteratively build a recursive approximation by utilizing curvature information along the trajectory. It is crucial to emphasize that a quadratic approximation offers a direction that can be leveraged within the iterative update scheme. This direction is defined by the equation:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \alpha_t \boldsymbol{d}_t. \tag{8}$$

In order to determine the direction $\boldsymbol{d}_t$, we can employ a quadratic approximation of the objective function. This approximation can be expressed as:

$$J(\boldsymbol{x}_t + \boldsymbol{d}) \approx m_t(\boldsymbol{d}) = J(\boldsymbol{x}_t) + \nabla J(\boldsymbol{x}_t)^\mathsf{T} \boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^\mathsf{T} \boldsymbol{B}_t \boldsymbol{d}, \tag{9}$$

where $J(\boldsymbol{x}_t)$ represents the objective function evaluated at the current point $\boldsymbol{x}_t$, $\nabla J(\boldsymbol{x}_t)$ denotes the gradient of the objective function at $\boldsymbol{x}_t$, $\boldsymbol{B}_t \in \mathbb{R}^{n\times n}$ corresponds to the approximation of the Hessian matrix. By minimizing the right-hand side of the quadratic approximation in Eq. (9), we can determine the optimal direction $\boldsymbol{d}_t$. Taking the derivative of $m_t(\boldsymbol{d})$ with respect to $\boldsymbol{d}$ and setting it to zero, we obtain:

$$\frac{\nabla m_t(\boldsymbol{d})}{\nabla \boldsymbol{d}} = \boldsymbol{d}_t \boldsymbol{B}_t + \nabla J(\boldsymbol{x}_t) \xrightarrow{\nabla m_t(\boldsymbol{d})=0} \boldsymbol{d}_t = -\boldsymbol{B}_t^{-1} \nabla J(\boldsymbol{x}_t),$$

by substituting this result in Eq. (8) we obtain:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha_t \boldsymbol{B}_t^{-1} \nabla J(\boldsymbol{x}_t).$$

The objective is to ensure that the curvature along the trajectory is consistent. In other words, at the last two iterations, m_{t+1} should match the gradient $\nabla J(\boldsymbol{x}_t)$ in the following way:

$$\nabla m_{t+1}\Big|_{\boldsymbol{d}=0} = \nabla J(\boldsymbol{x}_{t+1}),\ \nabla m_{t+1}\Big|_{\boldsymbol{d}=-\alpha_t \boldsymbol{d}_t} = \nabla J(\boldsymbol{x}_t).$$

This condition ensures that the quadratic approximation captures the correct curvature information along the trajectory, allowing for accurate optimization and convergence of the algorithm. By evaluating $\nabla m_{t+1}(\cdot)$ at the point $-\alpha_t \boldsymbol{d}_t$ we obtain:

$$\alpha_t \boldsymbol{B}_{t+1} \boldsymbol{d}_t = \nabla J(\boldsymbol{x}_{t+1}) - \nabla J(\boldsymbol{x}_t).$$

From Eq. (8) we get the secant equation:

$$\boldsymbol{B}_{t+1} \underbrace{(\boldsymbol{x}_{t+1} - \boldsymbol{x}_t)}_{\boldsymbol{s}_t} = \underbrace{\nabla J(\boldsymbol{x}_{t+1}) - \nabla J(\boldsymbol{x}_t)}_{\boldsymbol{z}_t} \rightarrow \boldsymbol{B}_{t+1} \boldsymbol{s}_t = \boldsymbol{z}_t.$$

To avoid explicitly computing the inverse matrix $\boldsymbol{B}_t^{-1}$, we can introduce an approximation $\boldsymbol{H}_t = \boldsymbol{B}_t^{-1}$ and optimize it as follows:

$$\begin{aligned} \boldsymbol{H}_{t+1} = \arg\min_{\boldsymbol{H}} \quad & \|\boldsymbol{H} - \boldsymbol{H}_t\|^2_{\boldsymbol{W}} && \triangleright \boldsymbol{H}_{t+1} \text{ close to } \boldsymbol{H}_t \\ \text{s.t.:} \quad & \boldsymbol{H} = \boldsymbol{H}^\mathsf{T} && \triangleright \text{ symmetry} \\ & \boldsymbol{H}\boldsymbol{z}_t = \boldsymbol{s}_t && \triangleright \text{secant equation} \end{aligned} \tag{10}$$

Here $\|\cdot\|^2_{\boldsymbol{W}}$ denotes the weighted Frobenius norm. This optimization problem aims to find an updated approximation $\boldsymbol{H}_{t+1}$ that is close to $\boldsymbol{H}_t$, while satisfying the constraints that $\boldsymbol{H}_{t+1}$ is symmetric and satisfies the secant equation $\boldsymbol{H}_{t+1}\boldsymbol{z}_t = \boldsymbol{s}_t$. BFGS [10, 13, 21] uses $\boldsymbol{W} = \int_0^1 \nabla^2 J(\boldsymbol{x}_t + t\alpha_t \boldsymbol{d}_t)dt$, to solve this optimization problem and obtain the iterative update of $\boldsymbol{H}$:

$$\boldsymbol{H}_{t+1} = (\boldsymbol{I} - \rho_t \boldsymbol{s}_t \boldsymbol{z}_t^\mathsf{T})\boldsymbol{H}_t(\boldsymbol{I} - \rho_t \boldsymbol{z}_t \boldsymbol{s}_t^\mathsf{T}) + \rho_t \boldsymbol{s}_t \boldsymbol{s}_t^\mathsf{T}, \tag{11}$$

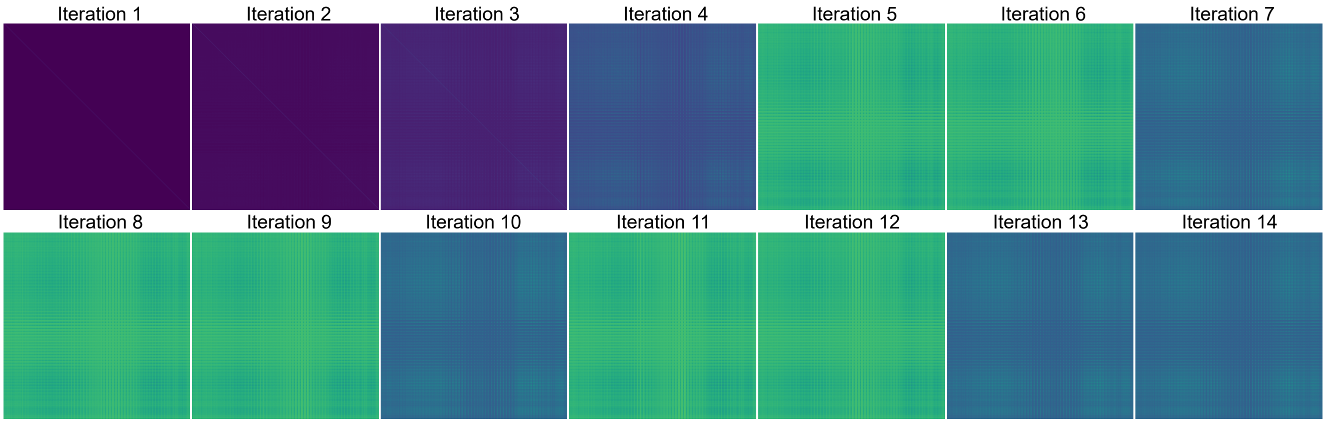


Figure 7. **Visualization of the inverse Hessian approximation across iterations**. Observe the subtle changes between each iteration, attributed to the influence of the objective function used to estimate H_t (see Eq. (10)).

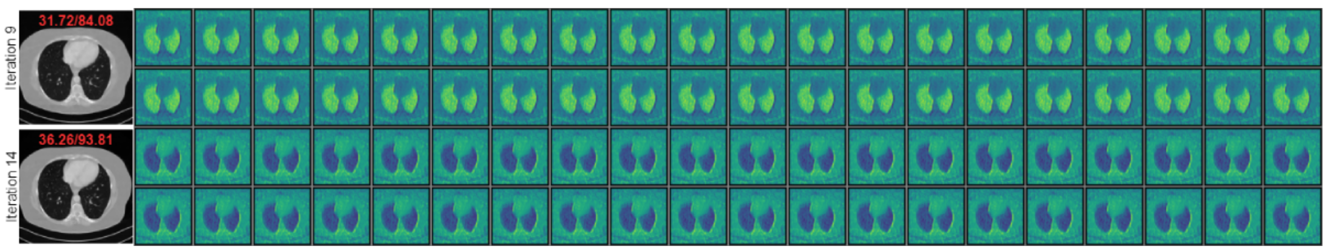


Figure 8. **Inverse Hessian approximation rows visualization**. We present the first 40 rows for the 9^{th} and 14^{th} inverse Hessian approximations on the first and second lines, respectively. Each row is of size 64^2, reshaped into a 64×64 image. The corresponding image reconstructions are shown on the left, along with PSNR (dB) and SSIM (%) values at the top.

where $\rho_t = \frac{1}{\boldsymbol{z}_t^{\mathrm{T}} \boldsymbol{s}_t}$, $\boldsymbol{s}_t = \boldsymbol{x}_{t+1} - \boldsymbol{x}_t$, and $\boldsymbol{z}_t = \nabla J(\boldsymbol{x}_{t+1}) - \nabla J(\boldsymbol{x}_t)$. This update equation allows us to iteratively refine the approximation $\boldsymbol{H}_t$ based on the current gradient information and the changes in the solution.

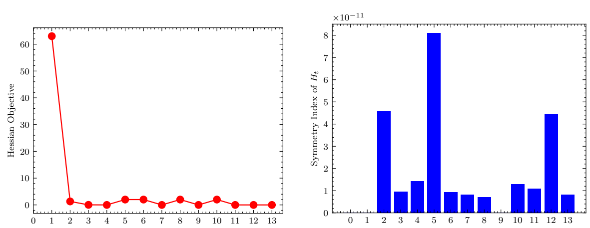


Figure 9. **Inverse Hessian matrix approximation algorithm**. Verification of requirements over the iterations. Left: Objective function value in Eq. (10); Right: Symmetry index of the inverse Hessian approximation refer to Eq. (12).

A.2. Validation of adherence to BFGS

We validate the adherence of our method to the BFGS requirements. To achieve this, we present the constraint values of the optimization algorithm given in Eq. (10) using a test set image from AAPM, as illustrated in Fig. 9. The symmetry index is defined as follows:

$$SI = \frac{1}{n \cdot (n-1)} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} |\boldsymbol{A}_{ij} - \boldsymbol{A}_{ji}|. \tag{12}$$

Our results demonstrate the effectiveness of our approach in satisfying the essential conditions required by the BFGS algorithm. Notably, the symmetry index is consistently close to zero, indicating the symmetry of the matrix $\boldsymbol{H}_t$ at each iteration, which is the first constraint of the BFGS method. Furthermore, with regard to the objective function value, it is evident that it is close to zero, except for the initial approximation. This deviation can be attributed to the use of the identity matrix as the starting point.

Inverse Hessian matrix approximation visualizations. Figure 7 depicts $\boldsymbol{H}_t$ at different iterations. These visualizations confirm the required symmetry of the matrix in each iteration. Additionally, the matrix $\boldsymbol{H}_t$ is close to the identity matrix at the second iteration, becoming more structured in the third iteration. This behavior aligns with expectations, as the matrix $\boldsymbol{H}_t$ is initialized as the identity matrix and updated based on gradient information and solution changes.

Visualization of reshaped rows. To further understand the inverse Hessian matrix approximation structure, we depict the reshaped (64×64) first 40 rows of $\boldsymbol{H}_t$ at iterations 9 and 14 in Fig. 8. These rows store gradient attention information used for updating the solution, consistent with the matrix $\boldsymbol{H}_t$ being updated based on gradient information

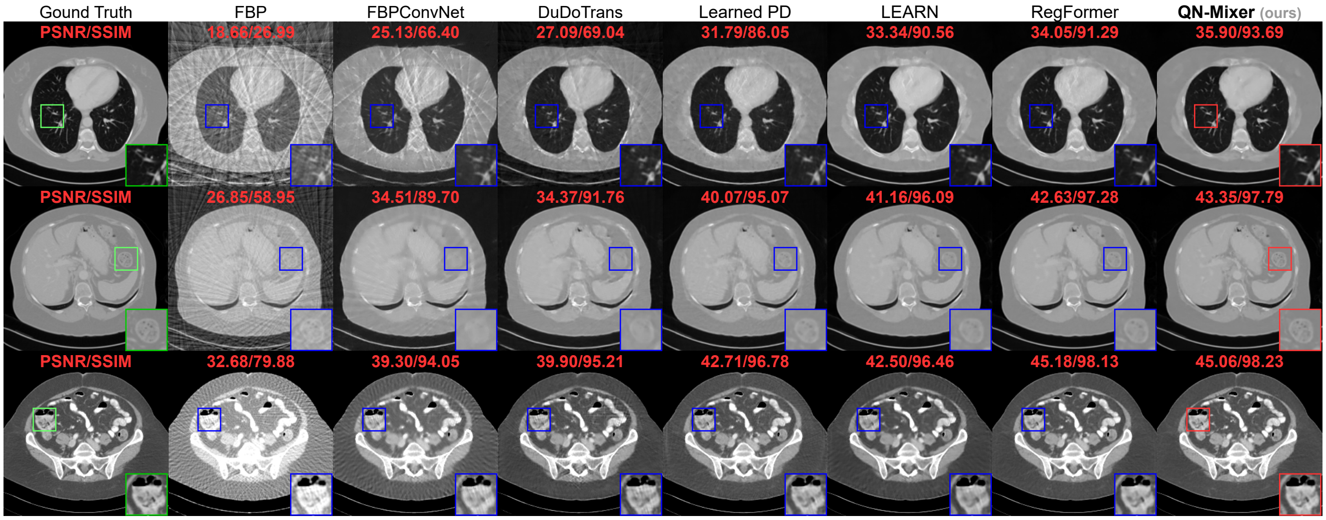


Figure 10. **Visual comparison on AAPM**. From top to bottom: the results under the following conditions: first $(n_v = 32, N_1)$, second $(n_v = 64, N_1)$, third $(n_v = 128, N_1)$. The display window is set to $[-1000, 800]$ HU for the first two rows and to $[-200, 300]$ HU for the last row.

and solution changes. In future work, we plan to explore the impact of $\boldsymbol{H}_t$ on the optimization process and its influence on reconstruction performance.

B. More Ablation Study and visualization

B.1. Ablation on Incept-Mixer

We further investigate the impact of the hyperparameters of Incept-Mixer on the reconstruction performance. We vary the patch size p and the number of stacked Mixer layers N and report the results in Tab. 7a, and Tab. 7b respectively.

p	PSNR ↑	SSIM ↑
2	38.22	95.07
4	**39.51**	**96.11**
8	38.19	95.32

(a)

N	PSNR ↑	SSIM ↑
1	37.64	94.79
2	**39.51**	**96.11**
3	38.47	95.51
4	38.17	95.40

(b)

Table 7. **Ablation of `Incept-Mixer`**. (a) p is the patch size; (b) N is the number of stacked Mixer layers. The best performance is attained using $p = 4$ and $N = 2$.

Impact of the path size. We observe that increasing the patch size p from 2 to 4 improves the performance ($+1.28$ dB and $+1.04\%$) while further increasing the patch size from 4 to 8 decreases the performance (-1.32 dB and -0.79%).

We attribute this observed pattern to the trade-off between local and global features in the reconstruction process. When the patch size is small, such as $p = 2$, the model focuses on capturing fine-grained local details, which can enhance reconstruction accuracy. As the patch size increases to $p = 4$, the network gains a broader perspective by considering larger regions, leading to an improvement in performance. However, when the patch size becomes too large, for example, $p = 8$, the model might start incorporating more global context at the expense of losing finer details. This can result in a decrease in performance as the model becomes less sensitive to localized patterns.

Impact of the number of stacked Mixer layer. We observe that increasing the number of stack N from 1 to 2 improves the performance ($+1.86$ dB and $+1.31\%$), while further increasing the patch size from 2 to 3 decreases the performance (-1.03 dB and -0.60%) and from 3 to 4 decreases the performance (-0.30 dB and -0.11%).

Similarly, when varying the number of stacked Mixer layers N, we observe a trend where an increase in N initially contributes to improved performance, as the model can capture more complex features and relationships. However, as N continues to grow, the network may encounter diminishing returns, and the benefits of additional layers diminish, potentially leading to overfitting or increased computational overhead.

Robustness to hyperparameters. Hence, there exists an optimal trade-off between the patch size p and the number of stacked Mixer layers N, but the model performs similarly for a wide range of values. In our experiments, we use $p = 4$ and $N = 2$ for all the datasets, which highlights the robustness of our method to these hyperparameters.

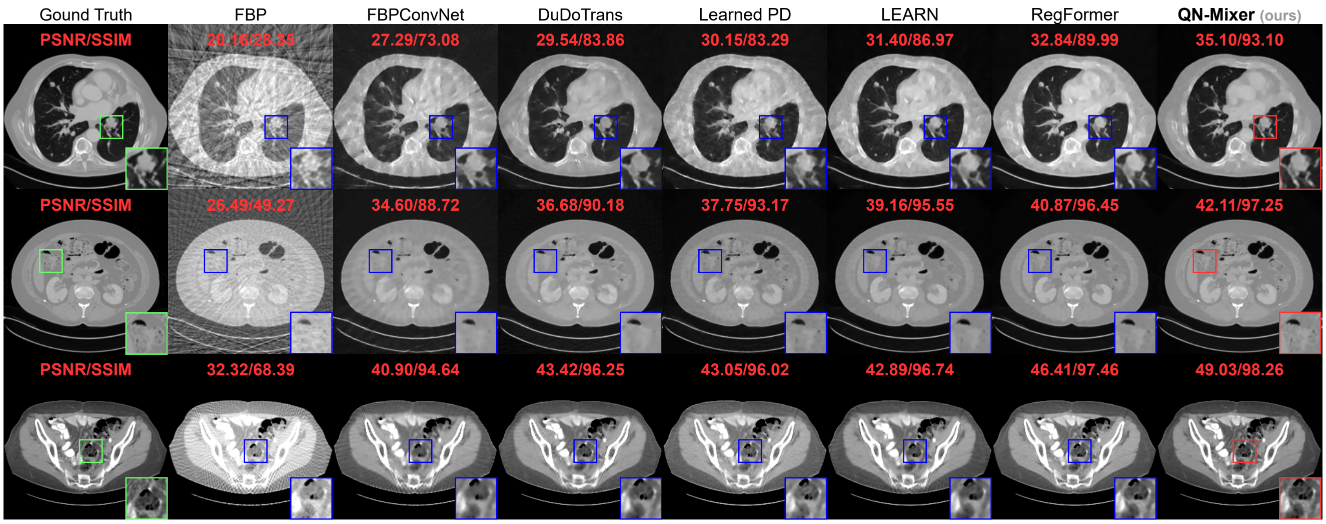


Figure 11. **Visual comparison on DeepLesion**. From top to bottom: the results under the following conditions: first ($n_v = 32, N_1$), second ($n_v = 64, N_1$), third ($n_v = 128, N_1$). The display window is set to $[-1000, 800]$ HU for the first two rows and to $[-200, 300]$ HU for the last row.

B.2. More visualization results

Fig. 10 displays supplementary visualizations of our approach on AAPM. Our method consistently produces high-quality reconstructions across all views. Notably, among state-of-the-art techniques, QN-Mixer excels in reconstructing fine-grained details. For instance, it accurately captures small vessels in the first row, delicate soft tissue structures in the second row, and sharp boundaries in the third row.

In Fig. 11, we showcase additional visualizations of our method applied to DeepLesion. QN-Mixer demonstrates superior performance across all views, yielding high-quality reconstructions. This is particularly evident in the challenging scenario of 32 views, where our method outperforms others in capturing fine-grained details, such as small vessels and lesions. Importantly, these results are achieved with fewer iterations compared to alternative unrolling networks like RegFormer.

B.3. Iterative results visualization

In order to demonstrate the effectiveness of QN-Mixer, we present a series of intermediate reconstruction results in Fig. 12. These results illustrate the progression of the reconstruction process at different iterations of our method. By examining the reconstructed outputs at each iteration, our goal is to offer insights into the evolution of image quality. Notably, we observe that the improvement in quality, as quantified by the PSNR and SSIM values of each iteration, does not consistently increase with each iteration (see Iteration 10 in Fig. 12). We suspect that the observed unexpected behavior may arise from the variation of the objective function (i.e. Eq. (2)) around the point t in the unrolled network, which is dependent on a learnable gradient regularization term Incept-Mixer.

B.4. Reconstruction error visualization

We present the reconstruction error of QN-Mixer in comparison to LEARN and RegFormer in Fig. 13. The images are organized from left to right based on the SSIM value. As illustrated, our method consistently produces high-quality reconstructions. In the most challenging scenario ($n_v = 32$) with no added noise, and for the least favorable image, our method achieves a reconstruction with an SSIM of 93.53%, maintaining notably satisfactory performance compared to RegFormer with an SSIM of 91.30%. For the best reconstruction across all methods, our method achieves an SSIM of 97.72%, while RegFormer achieves an SSIM of 97.48%. These results demonstrate the robustness of our method when dealing with challenging scenarios.

Method	$n_v = 32$		$n_v = 64$		$n_v = 128$	
	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑
FBP	72.88	18.97	83.42	22.13	<u>91.75</u>	24.85
FBPConvNet [19]	63.91	20.94	73.02	24.12	80.60	25.74
DuDoTrans [49]	60.51	19.09	79.75	25.00	85.65	27.23
Learned PD [1]	67.99	21.92	83.79	25.51	85.42	25.86
LEARN [7]	<u>79.70</u>	<u>24.46</u>	<u>84.44</u>	<u>26.74</u>	88.16	26.20
RegFormer [54]	72.45	23.69	77.33	25.46	84.99	<u>28.22</u>
QN-Mixer (ours)	**86.17**	**25.95**	**94.56**	**30.95**	**97.04**	**33.98**

Table 8. **Quantitative results of the reconstruction of the cropped OOD circle. Bold**: Best, <u>under</u>: second best.

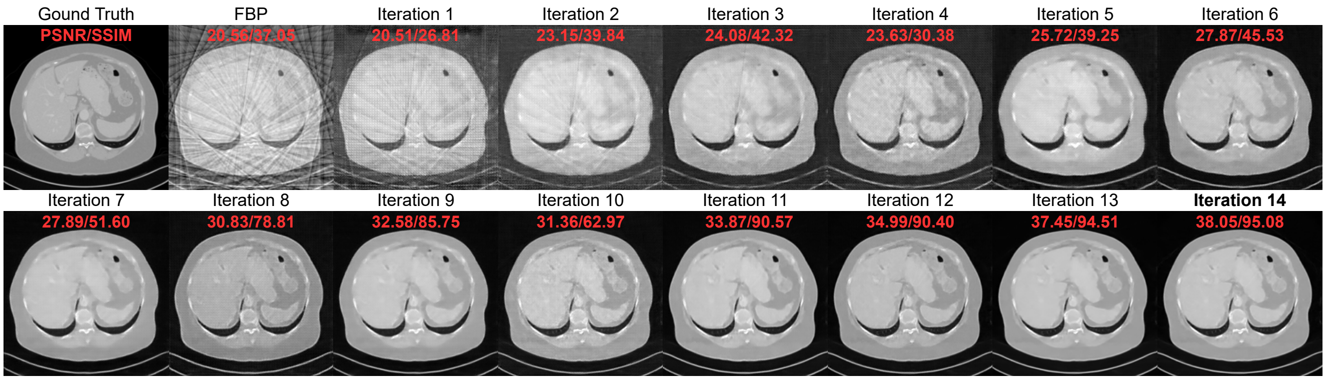
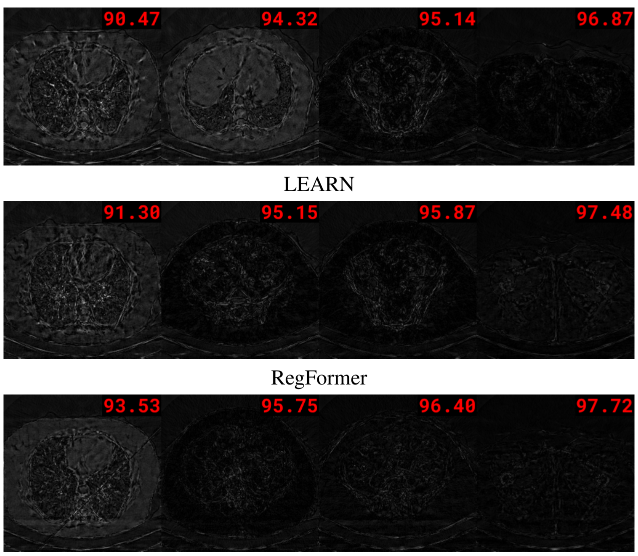


Figure 12. **QN-Mixer's intermediate reconstructions** using AAPM with $(n_v = 32, N_1)$. Display window is set to $[-1000, 800]$ HU.



QN-Mixer

Figure 13. **Reconstruction errors** with LEARN, RegFormer, and QN-Mixer using $(n_v = 32, N_0)$. Images are ordered left to right by SSIM, with the first column showing the worst reconstruction among 214 patient images. The second and third columns represent the $1/3$ and $2/3$ percentiles, respectively, and the last column corresponds to the best reconstruction with the highest SSIM.

C. More experiments

C.1. Out-of-Distribution

OOD circle performance across entire image In the main text, we assess the robustness of methods to a simple out-of-distribution scenario, where an unseen during training, white circle is inserted, and computing SSIM and PSNR metrics for the *entire image*. We achieved the best performance across all views in this setting (Tab. 3). We note, however that this setting mixes the inherent performance on clean data and its ability to handle unseen patterns.

OOD circle performance in anomalous region To further isolate the capacity of models to reconstruct unseen OOD patterns (i.e., white circle), we extend our evaluation. Instead of evaluating the whole image, we compute the reconstruction performance of a crop *region containing the white circle*, thus isolating the reconstruction performance exclusively to the circle region.

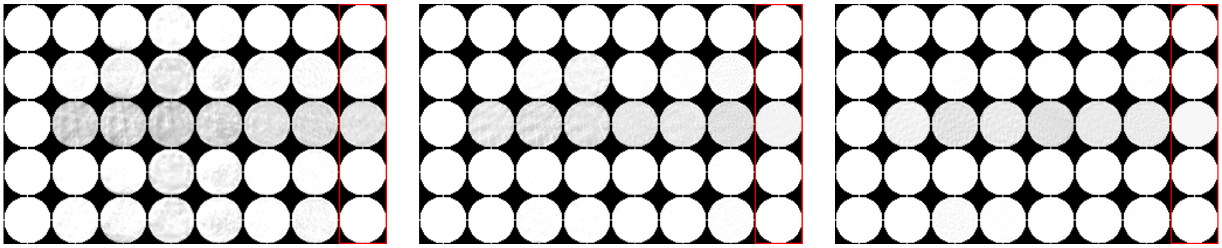
For that, we randomly selected 5 samples from the AAPM test set depicted. The evaluated data is depicted in Fig. 14, and the overall performance across the complete set of 214 patient images is summarized in Tab. 8 for 3 views.

Our method significantly outperforms the second-best across all views in both SSIM and PSNR. For the most challenging case of 32 views, we surpass the second best by $+6.47\%$ and $+1.49$ dB. With 64 views, our performance exceeds the second best by $+10.12\%$ and $+4.21$ dB. In the easiest case of 128 views, we outperform the second best by $+5.29\%$ and $+5.76$ dB. As anticipated, all methods exhibit degraded performance when focusing on the circle region, and the gap between our method and the second-best widens compared to the complete image. Moreover, the numerical results in Tab. 8 align with the visualizations in Fig. 14.

Train	AAPM n_v=32		AAPM n_v=128			
Test	DeepLesion n_v=32		AAPM n_v=32		AAPM n_v=64	
Method	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
DuDoTrans	28.69	73.77	25.60	51.01	32.38	82.35
LEARN	37.21	93.46	28.20	66.10	34.40	87.21
RegFormer	<u>37.97</u>	<u>94.01</u>	<u>29.83</u>	<u>72.53</u>	**37.78**	**92.64**
QN-Mixer (ours)	**38.28**	**94.66**	**30.43**	**80.03**	<u>37.16</u>	<u>91.81</u>

Table 9. **Quantitative results of the reconstruction performance of OOD cases: anatomy and geometry Bold**: Best, <u>under</u>: second best.

Anatomy and Geometry OOD. We explore more complex out-of-distribution (OOD) scenarios, particularly focusing on changes in anatomy and geometry. For the geometry aspect, we train our methods using AAPM data and evaluate them on DeepLesion, maintaining a fixed geometry parameter of n_v=32. Concerning anatomy, we test under the most



(a) $n_v = 32, N_0$ (b) $n_v = 64, N_0$ (c) $n_v = 128, N_0$

Figure 14. **Visualization of 5 samples of the OOD circle texture reconstruction.** From each figure and from left to right, we show the ground truth, FBP, FBPConvNet, DuDoTrans, Learned PD, LEARN, RegFormer and QN-Mixer.

challenging scenario, which involves training on AAPM data with n_v=128 and subsequently testing on AAPM data with two lower geometries: n_v=32 and n_v=64. The outcomes of this experiment are detailed in Tab. 9.

- **anatomy**: we outperform RegFormer by $+0.31$ dB and $+0.65\%$.
- **geometry** n_v=32: we outperform Regformer by $+0.60$ dB and $+7.50\%$ in most challenging setting.
- **geometry** n_v=64: we lag behind Regformer by -0.62 dB and -0.83% in the easiest setting.

C.2. Results when A is the $90°$ limited view CT

To further evaluate the robustness of our method, we conducted experiments on a more challenging CT inverse problem, specifically involving a $90°$ limited-angle CT setup. In this scenario, the system matrix A comprises a restricted number of views, resulting in significant artifacts and distortions in the reconstructed images. We compared the performance of our proposed QN-Mixer, with that of RegFormer on this demanding task. Our method demonstrated superior performance, achieving promising results with a PSNR of 30.17 dB and a SSIM of 90.01%. This outperformed RegFormer by $+1.03$ dB in PSNR and $+0.45\%$ in SSIM. Visual results illustrating the effectiveness of our approach are provided in Fig. 15.

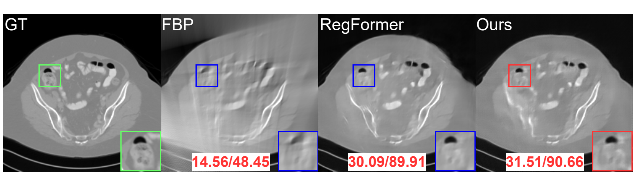


Figure 15. **Visual comparaison on AAPM with Limited-angle CT**. The system matrix A is a $90°$ limited view. The display window is set to $[-1000, 800]$ HU.

C.3. MBIR evaluation

To demonstrate the superiority of our method, we conducted an additional comparison with the state-of-the-art model-based iterative reconstruction (MBIR) technique, which is widely employed in clinical settings. Our evaluation involved testing our approach against MBIR using datasets with varying numbers of views, specifically n_v=32, 64, and 128, with low noise added to sinograms. The corresponding PSNR values obtained with MBIR were 22.94, 28.80, and 34.33 dB, with SSIM values of 67.70%, 72.60%, and 80.68%, respectively.

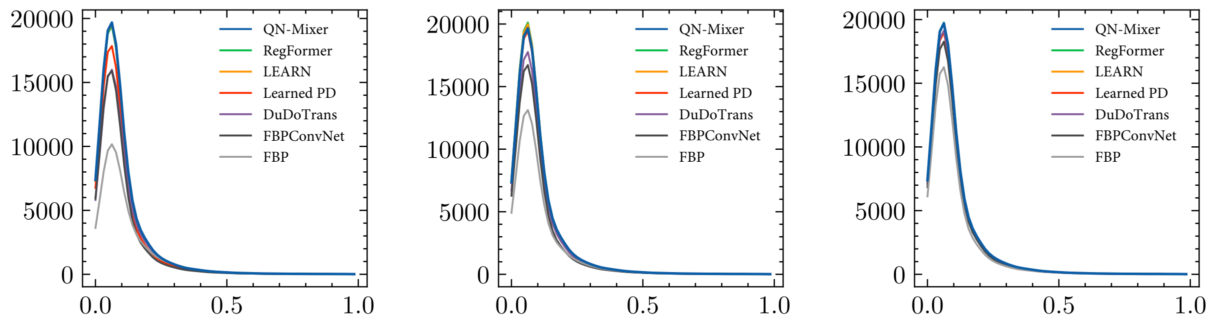
Our method, QN-Mixer, consistently outperformed MBIR across all views, showcasing an average improvement of $+12.72$ dB in PSNR and $+23.25\%$ in SSIM. These results underscore the robustness and effectiveness of our approach compared to MBIR, a benchmark technique widely utilized in clinical practice.

C.4. Noise Power Spectrum analysis

We conducted a comprehensive examination of the noise characteristics in our reconstructed images through noise power spectrum (NPS) analysis. NPS serves as a metric, quantifying the magnitude and spatial correlation of noise properties, or textures, within an image. It is derived from the Fourier transform of the spatial autocorrelation function of a zero-mean noise image.

NPS analysis was performed on a configuration of Regions of Interest (ROIs) as depicted in Fig. 17. This process was applied to all 214 images from the AAPM test set and for three different views (32, 64, and 128). The average 1D curves were generated by radially averaging the 2D NPS maps, and the results are presented in Fig. 16.

The area under the NPS curve is equal to the square of the noise magnitude. Importantly, the ordering of methods based on noise magnitude corresponds to the ranking observed in our quantitative experiments for PSNR and SSIM in the main text. For example, FBP, which exhibits the lowest noise magnitude, also performs the poorest in terms of PSNR and SSIM. Conversely, our method, with the highest noise magnitude, stands out as the top performer in both PSNR and SSIM metrics. Furthermore, the mean and peak frequencies serve as key indicators of noise texture or "noise grain size", where higher frequencies denote finer texture. Remarkably, our method showcases superior mean and peak frequencies compared to other methods, suggest-



(a) $n_v = 32, N_1$ (b) $n_v = 64, N_1$ (c) $n_v = 128, N_1$

Figure 16. **Noise Power Spectrum (NPS) Analysis** in comparison to state-of-the-art methods. The x-axis represents normalized frequency in cycles per pixel (px^{-1}), and the y-axis represents noise power spectrum ($\mathrm{HU}^2\mathrm{px}^2$). Display windows are configured as $[-1000, 800]$ HU. Mean and peak frequencies are intricately linked to noise texture, with finer textures correlating to higher mean and peak frequencies in the NPS. Our method exhibits the highest peak frequencies, indicating that our reconstructed images feature the most refined noise texture among all compared methods.

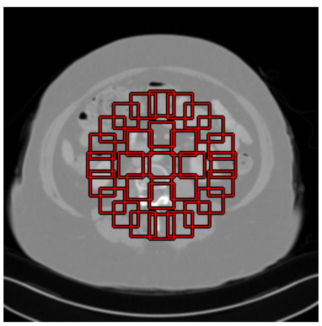


Figure 17. **ROIs for NPS Analysis:** Red squares denote 20×20 pixel ROIs distributed evenly across two circular regions. The first circle (radius 25) holds 8 ROIs, and the second circle (radius 50) has 20 ROIs. Both circles, centered at the image center, include a total of 29 ROIs per image. This standard positioning in the CT community underscores the clinical diagnostic importance of the image center.

ing a finer noise texture or smaller grain size.

This alignment with good clinical practice standards reinforces the robust performance of our method in capturing and preserving image details, as supported by both quantitative metrics and noise analysis.

D. Reproducibility

All our experiments are fully reproducible. While the complete algorithm is already provided in the main paper (see Algorithm 2), we additionally present a PyTorch pseudo-code for enhanced reproducibility in Appendix D.1. We furnish comprehensive references to all external libraries used in Appendix D.2. Detailed information regarding the initialization of our model can be found in Appendix D.3. The precise parameters of our regularizer, Incept-Mixer architecture, are available in Appendix D.4.

We outline the exact data splits utilized across the paper for

Algorithm 3: Minimal QN-Mixer pseudo-code

```
class GradientFunction(nn.Module):
  def __init__(self, regularizer):
    self.regularizer = regularizer
    self.lambda = nn.Parameter(torch.zeros(1))

  def forward(self, physics, y, x):
    y_t = physics.forward_operator(x)
    # Compute the regularization term
    reg_x = self.regularizer(x)
    # Compute the data fidelity term
    y_dft = y_t - y
    # Compute the backprojection
    x_dft = physics.backward_operator(y_dft)
    g = self.lambda * x_dft + reg_x
    return g

class QN_Iteration(nn.Module):
  def __init__(self, gradient_function):
    self.gradient = gradient_function

  def latent_bfgs(self, h, s_t, z_t):
    I = torch.eye(len(s_t))
    rho_t = 1. / torch.dot(z_t, s_t)
    u_t = I - torch.outer(s_t, z_t) * rho_t
    d_t = I - torch.outer(z_t, s_t) * rho_t
    return (torch.matmul(u_t, torch.matmul(h, d_t))
            + (torch.outer(s_t, s_t) * rho_t))

  def forward(self, physics, encoder, decoder,
              y, x, h, r, is_last):
    # Compute latent direction s_t
    s_t = -torch.matmul(h, r)
    d = decoder(s_t)

    # Update the reconstruction
    x = x + d
    # Return x if it is the last iteration
    if is_last:
        return x, h, r
    else:
        r_p = encoder(self.gradient(physics, y, x))
        z_t = r_p - r
        with torch.no_grad():
            h = self.latent_bfgs(h, s_t, z_t)
        return x, h, r_p
```

the AAPM dataset in Appendix D.5. Lastly, to facilitate the reproduction of our out-of-distribution (OOD) protocol, we provide the pseudo-code in Appendix D.6.

D.1. QN-Mixer pseudo-code

Our QN-Mixer algorithm is introduced in Algorithm 2, and for improved reproducibility, we present a PyTorch pseudo-code in Algorithm 3. The fundamental concept underlying unrolling networks lies in having a modular gradient function, denoted as $\nabla J(\boldsymbol{x}_t)$, which can be easily adapted to incorporate various regularization terms. Subsequently, the core element is the unrolling iteration block responsible for updating both the solution $\boldsymbol{x}_t$ and the inverse Hessian approximation $\boldsymbol{H}_t$. The update of the inverse Hessian approximation is executed through the latent BFGS algorithm. Notably, each iteration call takes the physics operator as input, tasked with computing the forward and pseudo-inverse operators for the CT reconstruction problem, along with the gradient encoder and direction decoder, which are shared across all iterations. For a more in-depth understanding, refer to Algorithm 3. Note the employment of `torch.no_grad()` to inhibit the computation of gradients for the inverse Hessian approximation. Since there is no necessity to compute gradients for this variable, given that it is updated through the latent BFGS algorithm.

Within these two modules, second-order quasi-Newton methods can be seamlessly incorporated by simply modifying the latent BFGS algorithm or the regularization term, offering flexibility to the user.

D.2. External libraries used

We utilized the following external libraries to implement our framework and conduct our experiments:

- Operator Discretization Library (ODL): `https://github.com/odlgroup/odl`
- High-Performance GPU Tomography Toolbox (ASTRA): `https://www.astra-toolbox.com/`
- Medical Imaging Python Library (Pydicom): `https://pydicom.github.io/`

D.3. QN-Mixer's parameters initialization

To enhance reproducibility, we provide the parameters initialization of QN-Mixer. *First*, for the gradient function, we initialize the CNNs of Incept-Mixer using the Xavier uniform initialization. The multi-layer perceptron of the MLP-Mixer is initialized with values drawn from a truncated normal distribution with a standard deviation of 0.02. The λ_t values are initialized to zero, and the inverse Hessian approximation $\boldsymbol{H}_0$ is initialized with the identity matrix $\boldsymbol{I}$. *Second*, for the latent BFGS, both the encoder and decoder CNNs are initialized with the Xavier uniform initialization.

D.4. Incept-Mixer's architecture

For enhanced reproducibility, we present the architecture of Incept-Mixer in Tab. 11. The Incept-Mixer architecture consists of a sequence of Inception blocks, followed by Mixer blocks. Each Mixer block comprises a channel-mixing MLP and a spatial-mixing MLP. The MLPs are constructed with a fully-connected layer, a GELU activation function, and another fully-connected layer. Ultimately, the regularization value is projected to the same dimension as the input image through a patch expansion layer, which is composed of a fully-connected layer and a CNN layer.

Patient ID	L067	L109	L143	L192	L286	L291	L096	L506	L333	L310
#slices	224	128	234	240	210	343	330	211	244	214
Training	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗
Validation	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Testing	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Table 10. **AAPM dataset split specification**. The validation set comprises images from patient L333, and testing utilizes images from patient L310. The images from the remaining patients have been designated for training purposes.

Stage	Layers	#Param (k)	Output size
Input	-	-	$1 \times 256 \times 256$
InceptionBlock-1	convblock1: conv1-1: K1C16S1P0 prelu1-1 convblock2: conv2-1: K1C16S1P0 prelu2-1 conv2-2: K3C32S1P1 prelu2-2 convblock3: conv3-1: K1C16S1P0 prelu3-1 conv3-2: K5C32S1P2 prelu3-2 convblock4: maxpool4-1: K3S1P1 conv4-1: K1C16S1P0 prelu4-1	17.6	$96 \times 256 \times 256$
PatchEmbed-2	conv2-1: K4C96S4P0 rearrange2-1: bchw $\rightarrow$ bhwc	145.5	$96 \times 64 \times 64$
MixerLayer-3 $\times(N=2)$	layernorm3-1: D96 rearrange3-1: bhwc $\rightarrow$ bcwh heightmlp3-1: linear3-1: D64O256 gelu3-1 linear3-2: D256O64 rearrange3-2: bcwh $\rightarrow$ bchw widthmlp3-1: linear3-3: D64O256 gelu3-2 linear3-4: D256O64 rearrange3-3: bchw $\rightarrow$ bhwc layernorm3-2: D96 channelmlp3-1: linear3-5: D96O384 gelu3-3 linear3-6: D384O96	140.8×2	$96 \times 64 \times 64$
PatchExpand-4	linear4-1: D96O1536 layernorm4-1: D96 conv4-1: K1C1S1P0	147.7	$1 \times 256 \times 256$

Table 11. **Incept-Mixer architecture**. `K-C-S-P` represents the kernel, channel, stride, and padding configuration of CNNs, while `D-O` indicates the input and output dimensions of linear layers.

D.5. AAPM dataset splits

In our experiments, we use the AAPM 2016 Clinic Low Dose CT Grand Challenge public dataset [35], which holds substantial recognition as it was formally established and authorized by the esteemed Mayo Clinic. To ensure the integrity of our evaluation process, we followed the precedent set by [7, 54] and created the training set using data from eight patients, while reserving a separate patient for the testing and validation sets. This approach guarantees that no identity information is leaked during test time. Our specification is presented in Tab. 10.

D.6. Robustness eval. protocol for OOD scenarios

Algorithm 4: `add_circle_ood` pseudo-code.

```
def add_circle_ood(img, value=1):
  h, w = img.shape[::-1][:2]
  radius = np.random.randint(5, 20)
  c_x = np.random.randint(radius, w-radius)
  c_y = np.random.randint(radius, h-radius)
  center = (c_x, c_y)

  Y, X = np.ogrid[:h, :w]
  dist_x = (X - center[0])**2
  dist_y = (Y - center[1])**2
  dist_from_center = np.sqrt(dist_x + dist_y)
  mask = dist_from_center <= radius
  img[0, mask] = value
  return img
```

In medical imaging, it's crucial to develop methods that generalize to scans with lesions or anomalies, and assessing the model's capability to reconstruct abnormal data holds significant relevance, as test patient data may deviate from the training data in clinical applications. To this end, we design a simple protocol specifically crafted for evaluating the effectiveness of methods when handling abnormal data. In this case, a white circle mimicking an out-of-distribution texture, which was never seen during training, is forged into CT images with noise-free sinograms. The pseudo-code to realize this is provided in Algorithm 4. Then, performance can be evaluated on the entire image as shown in Tab. 3, or on a cropped region within the circle as detailed in Tab. 8.

We strongly advocate for future research endeavors to embrace and employ this protocol as a standard for evaluating the robustness of reconstruction methods.

E. Limitations

Our approach inherits similar limitations from prior methods [7, 54]. First, our method entails a prolonged optimization time, stemming from the utilization of unrolling reconstruction networks [7, 54], in contrast to post-processing-based denoising methods [19, 49]. While our method represents the fastest unrolling network, there is still a need to address the existing gap. Integrating Limited-memory BFGS into our QN-Mixer framework is an interesting research direction for accelerating training. Second, while we have assessed our method using the well known AAPM low-dose and DeepLesion datasets and compared it with several state-of-the-art methods, the evaluation is conducted on images representing specific anatomical regions (thoracic and abdominal images). The generalizability of our method to a broader range of datasets, which may exhibit diverse characteristics or variations, remains unclear. Third, the acquisition of paired data has always been an important concern in clinic. Combining our approach with unsupervised training framework to overcome this limitation can be an exciting research direction. Finally, the incorporation of actual patient data into our training datasets raises valid privacy concerns. Although the datasets we utilized underwent thorough anonymization and are publicly accessible, exploring a solution that can effectively operate with synthetic data emerges as an intriguing avenue to address this challenge.

F. Notations

We offer a reference lookup table, available in Table 12, containing notations and their corresponding shapes as discussed in this paper.

Notation	Shape	Value(s)	Description
n_v	$\mathbb{N}^*$	$\{32, 64, 128\}$	The number of projection views
n_d	$\mathbb{N}^*$	512	The number of projection detectors
h	$\mathbb{N}^*$	256	Height of the image
w	$\mathbb{N}^*$	256	Width of the image
c	$\mathbb{N}^*$	1	Channels of the image
l_h	$\mathbb{N}^*$	64	Latent height
l_w	$\mathbb{N}^*$	64	Latent width
$m = n_v \times n_d$	$\mathbb{N}^*$	$n_v \times 512$	Data (sinogram) size
$n = h \times w$	$\mathbb{N}^*$	256×256	Image size
$(l_h \cdot l_w) \times (l_h \cdot l_w)$	$\mathbb{R}$	$(64 \cdot 64) \times (64 \cdot 64)$	Size of the latent BFGS optimization variable i.e. $\boldsymbol{H}$
T	$\mathbb{N}^*$	14	Number of iterations of our method
t	$\mathbb{N}$	-	Iteration of the loop in the algorithm
$\boldsymbol{y}$	$n_v \times n_d$	-	Sparse sinogram
$\boldsymbol{A}$	$\mathbb{R}^{n \times m}$	-	The forward model (i.e. discrete Radon transform)
$\boldsymbol{A}^\dagger$	$\mathbb{R}^{m \times n}$	-	The pseudo-inverse of $\boldsymbol{A}$
$\boldsymbol{x}_0$	$\mathbb{R}^{h \times w \times c}$	$\boldsymbol{A}^\dagger \boldsymbol{y}$	Initial reconstruction
λ_t	$\mathbb{R}$	-	Regularization weight at step t
α_t	$\mathbb{R}$	-	Step size (i.e. search step)
$\boldsymbol{x}_t$	$\mathbb{R}^{h \times w \times c}$	-	Reconstructed image at iteration t
$\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_t)$	$\mathbb{R}^{h \times w \times c}$	-	Gradient value at iteration t
$\boldsymbol{H}_t$	$\mathbb{R}^{(l_h \cdot l_w) \times (l_h \cdot l_w)}$	-	Approximation of the inverse Hessian matrix at iteration t
$\boldsymbol{I}^{n \times n}$	$\mathbb{N}^{n \times n}$	-	Identity matrix of size $n \times n$
$\boldsymbol{f}_t$	$\mathbb{R}^{h \times w \times d}$	-	Feature map after the Inception block at iteration t
$\boldsymbol{e}_t$	$\mathbb{R}^{\frac{h}{p} \times \frac{w}{p} \times d}$	-	MLP-Mixer embeddings
d	$\mathbb{R}$	96	Depth of features
p	$\mathbb{N}^*$	4	Stride and kernel size in the patchification Conv 2D net
N	$\mathbb{N}^*$	2	Number of stacked Mixer layers
$\mathcal{G}(\cdot)$	-	-	Learned gradient of the regularization term (i.e. the Incept-Mixer model)
$\mathcal{G}(\boldsymbol{x}_t)$	$\mathbb{R}^{h \times w \times c}$	-	Regularization term at step t
$\mathcal{E}(\cdot)$	-	-	The gradient encoder
$\mathcal{D}(\cdot)$	-	-	The direction decoder
k	$\mathbb{N}^*$	$\{2, 3, 4, 5\}$	Number of Downsampling stacks in the encoder
$\boldsymbol{f}_{\mathcal{E}} = 2^k$	$\mathbb{N}^*$	$\{4, 8, 16, 32\}$	Downsampling factor of the gradient in the encoder
$w_l = w / \boldsymbol{f}_{\mathcal{E}}$	$\mathbb{N}^*$	$\{64, 32, 16, 8\}$	Number of columns of the down-sampled gradient
$h_l = h / \boldsymbol{f}_{\mathcal{E}}$	$\mathbb{N}^*$	$\{64, 32, 16, 8\}$	Number of rows of the down-sampled gradient
$\boldsymbol{r}_t = \mathcal{E}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_t))$	$\mathbb{R}^{l_h \cdot l_w}$	-	Latent representation of the gradient
$s_t = -\boldsymbol{H}_t \boldsymbol{r}_t$	$\mathbb{R}^{l_h \cdot l_w}$	-	Direction in the latent space
$\rho_t = (\boldsymbol{z}_t^\top \boldsymbol{s}_t)^{-1}$	$\mathbb{R}^{l_h \cdot l_w}$	-	BFGS divider variable
N_0	-	-	Zero noise added to the sinogram
N_1	-	-	5% Gaussian noise, 1×10^6 intensity Poisson noise
N_2	-	-	5% Gaussian noise, 5×10^5 intensity Poisson noise

Table 12. **Lookup table of notations and hyperparameters** used in the paper.