R^2-Net: Recurrent and Recursive Network for Sparse-view CT Artifacts Removal

No Author Given

No Institute Given

Abstract. We propose a novel neural network architecture to reduce streak artifacts generated in sparse-view 2D Cone Beam Computed Tomography (CBCT) image reconstruction. This architecture decomposes the streak artifacts removal into multiple stages through the recurrent mechanism, which can fully utilize information in previous stages and guide the learning of later stages. In each recurrent stage, the key components of the architecture are computed recursively. Recursive mechanism can help to save parameters and enlarge the receptive field efficiently with exponentially increased dilation of convolution. To verify its effectiveness, we conduct experiments on the AAPM CBCT dataset through 5-fold cross-validation. Measured by mainstream evaluation metrics, i.e., PSNR and SSIM, our proposed method outperforms the state-of-the-art methods quantitatively and qualitatively.

Keywords: Computed Tomography · Sparse-view Reconstruction · Convolutional Recurrent Neural Network.

1 Introduction

Cone beam computed tomography (CBCT) is first introduced into the European market in 1996 and the US market in 2001. In the past twenty years, the radiation risk issue of CBCT receives much attention. And the demand of radiation dose reduction becomes more intense. One way to reduce radiation dose and shorten acquisition time is sparse-view CBCT reconstruction, which is achieved by reducing the number of radiation angles, i.e., views. However, this process introduces some streak artifacts, which is also called "down-sampling streak artifacts" since reducing the number of radiation angles can be regarded as down-sampling.

Great efforts have been devoted to improve sparse-view CBCT reconstruction's quality. Existing approaches to address the artifacts can be mainly divided into two categories: (1) Classical methods: ASD-POCS [7] and PICCS [8] are based on compressed sensing theory, AwTV [9] and TVS-POCS [10] are total variation based methods, ASDL [11] and AS-LNLM [12] develop from dictionary learning and so on. (2) Deep learning methods: almost all state-of-the-art (SOTA) deep learning methods are based on U-Net [13] scheme, such as Tight Frame U-Net(TF U-Net) [5], cascade of U-Nets [14], etc. In addition to these neural networks, some methods [15, 16] are based on Generative Adversarial Networks (GAN), whose generators are still following the U-Net framework. In terms of PSNR and SSIM, deep learning methods outperform traditional methods.

In most mainstream deep learning algorithms, there are always existing two weaknesses. (1)They propose that a sparse-view CT image I_s can be decomposed into a background-like dense-view CT image I_d and a down-sampling streak artifacts image A. These components can be combined linearly as below:

$$I_s = I_d + A. \tag{1}$$

However, if we observe the sparse-view CT images more carefully, we will find that the streak artifacts in different positions, under different luminance or overlapped with different organs have different characteristics. So it is more proper to decompose the original CT image into a background image and several different streak artifacts, which can be formulated as:

$$I_s = I_d + \sum_{i=1}^{n} \alpha_i A_i, \tag{2}$$

where A_i represents different streak artifacts images with their own characteristics, and α_i denotes the intensity of a certain kind of streak artifacts layer.

(2)U-Net fuses the low-level and high-level features through "contracting path" and "expansive path", which cost plenty of extra memory. The SOTA deep learning methods are always based on U-Net, so they also suffer from this waekness.

Motivated by the above two issues, we propose an architecture, so called R^2-Net. On the one hand, we think it is difficult to remove all down-sampling streak artifacts in only one stage. Therefore, it is more reasonable to decompose the down-sampling streak artifacts removal into multiple stages. While most U-Net-like models are based on the assumption in Eq. (1), which limits their capacity, we propose a novel network following Eq. (2) to incorporate the **recurrent** mechanism for removing streak artifacts effectively.

On the other hand, our proposed network also utilizes the low-level and high-level features, but need not extra branch to contract and expand like U-Net. This scheme is achieved by the **recursive** mechanism.

Because our network contains both recurrent and recursive mechanism, we call it R^2-Net.

2 Method

2.1 Overview

Our proposed streak artifacts removal network consists of several components, including an encoder network **E** to transform a 2D CBCT image to feature maps,

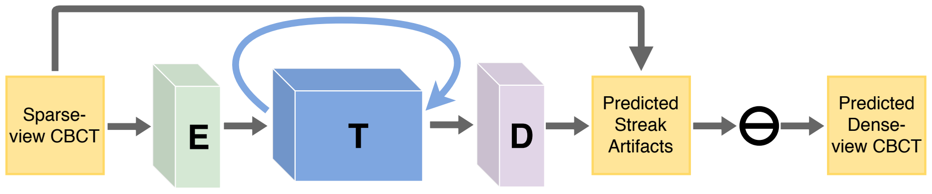

Fig. 1. The architecture of our proposed method in one recurrent stage.

a recursive transformer **T** in feature space, a decoder **D** to generates the residual image between input and ground truth.

For the overall framework, it contains several recurrent stages, and each recurrent stage has several recursive stages. In one recurrent stage, firstly, a sparse-view CBCT image $\mathbf{I}_s$ is used as input of the encoder network **E**. Then the recursive transformer **T** extracts feature maps recursively and sends these feature maps of all recursive stages to the decoder **D**. Later on, the decoder **D** aggregates refined feature maps from different recursive stages of **T** and estimates the residual image between input, i.e., sparse-view CBCT, and ground truth, i.e., dense-view CBCT. In the next recurrent stage, the output of previous recurrent stage is used as input to predict the residual more precisely.

In the following, we first describe the architecture of the base model, i.e., the model in one recurrent stage. Then we propose the recurrent structure, which lifts the model's capacity by iteratively decomposing down-sampling streak artifacts.

2.2 Base Model

The base model, i.e., the model in one recurrent stage, of our proposed method can be regarded as an image-to-image model illustrated in Fig. 1, where a forward network without recurrence transforms $\mathbf{I}_s$ to an ideally artifacts-free image that looks like $\mathbf{I}_d$.

Recurrent Unit In our proposed model, we choose Convolutional GRU as the key component of our recurrent unit, which is shown in Fig. 2(a). Convolutional GRU includes two convolutional kernels: one on the input vector and the other one on the hidden state vector. The input vector comes from previous layer. The hidden state vector provides information from previous recurrent stages, which can be considered as the recurrent nature of the unit. This design intends to adapt to our recurrent mechanism.

Besides, we regard each channel of recurrent unit's output as the embedding of several kinds of streak artifacts. So we extend each basic convolution layer with SE [1] block to explicitly compute alpha-value(α_i) for every channel of each feature map. Through multiplying alpha-values, feature maps are re-weighted to better fit the distribution of streak artifacts with different characteristics.

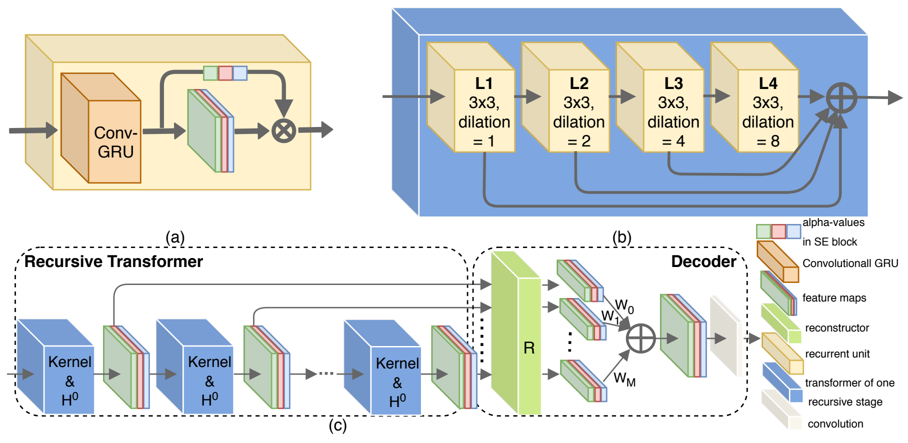

Fig. 2. (a): recurrent unit (b) recursive transformer in one recursive stage, i.e., a convolutional GRU group with exponentially increasing dilation and sum of all layers' output (c) unrolling recursive transformer, reconstructor and weighted sum of feature maps, i.e., **T** and **D** in Fig. 1. And the number of recursive stages is M in the figure.

Encoder E The encoder (sky blue tube marked with **E** in Fig. 1) is comprised of a recurrent unit and designed to transform an image to feature maps.

Recursive Transformer T This module (blue tube marked with **T** in Fig. 1) is recursively used in the base model. The unrolling version of **T** in shown in Fig. 2(c). Because this module is recursively utilized in one recurrent stage's forward propagation, so the convolution parameters are fixed and the computing process can be regarded as same. If we define the computing process of **T** in different recursive stages as function F, H_i as the hidden state vector of different recursive stages, x as the input of **T**, y as the output, and the recursive number is M, then the module can be formulated as:

$$y = F(H_{M-1}, F(H_{M-2}, F(\dots F(H_0, x)))). \tag{3}$$

Through this recursive mechanism, T reuses the copy of convolution parameters. So we can enlarge the receptive field without introducing more convolution parameters. In detail, if the recursive number is M and the original receptive field size is S^2, the final receptive field size will be $(\mathrm{MS})^2$.

As for inner structure of **T**, due to larger receptive field is very helpful to acquire more contextual information, dilation scheme is adopted in our recursive transformer. As shown in Fig. 2(b), for layers L1 to L4, the dilation increases from $1(=2^{1-1})$ to $8(=2^{4-1})$ exponentially, which leads to the exponential growth in receptive field of every element in high-level feature maps. Moreover, we use 3×3 convolution for all layers in the transformer. So the receptive field size, i.e., S, of L4's element is 31^2 in one recursive stage. The above two strategies lead to the fastest growth in receptive field size without missing a pixel in any element's receptive field. In each recursive stage, the outputs of each recurrent unit are added up to create new feature maps as input of next recursive stage. This design allows the low-level context information to be used directly together with the high-level context information, which is helpful to the next recursive stage.

Decoder D The recursive module has not only pros, but also cons. While the recursive module is simple and powerful, we find that training a deeply-recursive module is quite difficult. There are mainly two reasons accounting for this phenomenon, including the gradients vanishing and exploding [2, 3]. To solve the above issues, we supervise all recursions in order to alleviate the effect of gradients vanishing/exploding by a reconstructor (green tube marked with **R** in Fig. 2(c)). As shown in Fig. 2(c), "Supervise all recursions" means that the **R** is shared for predictions of all recursive stages from **T** in one recurrent stage. If the recursive number is N, all N groups of feature maps are sent to **D** and simultaneously supervised during training.

As for structure, **D** is comprised of **R** and two convolutions. First, **R** is built on a recurrent unit. Then all N predictions of **R**, as shown in Fig. 2(c) are summed by optimal weights which are automatically learned during training. The next part of **D** is comprised of a 3×3 convolution and a 1×1 convolution. After **R** transforms information from feature space to residual image space, we adopt the two convolutions, i.e., a 3×3 convolution and a 1×1 convolution, to recover gray channel for residual image, i.e., down-sampling streak artifacts.

Finally, the predicted residual image should be subtracted from sparse-view CT image to create the prediction of dense-view CT image.

2.3 Recurrent Model

As there are many different characteristics of streak artifacts in different positions, under different luminance or overlapped with different organs in a sparse-view CT image. So it is not easy to remove all streak artifacts in only one stage. Therefore, we incorporate recurrent mechanism into advanced model to decompose the streak artifacts removal task into multiple stages. In order to fully investigate the recurrent connections between different stages, we incorporate recurrent units with memory unit to better utilize the information from previous stages and guide learning in later stages. The recurrent model is shown in Fig. 3.

In each recurrent stage, our proposed model predicts the whole residual, i.e., streak artifacts. Our scheme can be formulated as:

$$I_s^1 = I_{ori}, \tag{4}$$

$$\widehat{Res^i} = F_i(I_s^i, H^{i-1}),\ 1 \leq i \leq N, \tag{5}$$

$$\widehat{I_d^i} = I_{ori} - \widehat{Res^i}, \tag{6}$$

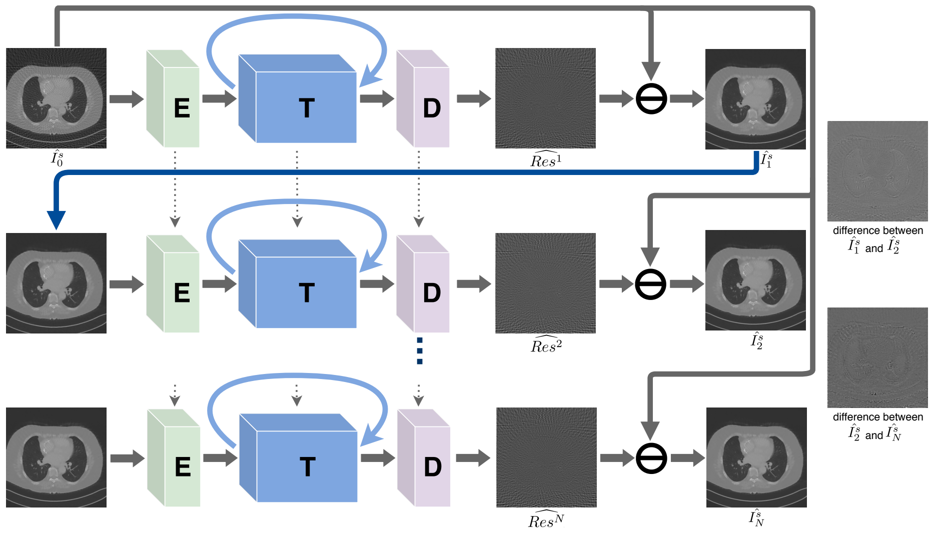

Fig. 3. The unrolling architecture of our proposed method.

$$I_s^{i+1} = \widehat{I_d^i}, \tag{7}$$

where I_{ori} indicates the original sparse-view CT image, I_s^i represents the input of the i-th recurrent stage, N is the number of recurrent stages, F_i indicates the computing process of the i-th recurrent stage, H^{i-1} represents previous states, $\widehat{Res^i}$ indicates the output of the i-th recurrent stage, and $\widehat{I_d^i}$ is the predicted dense-view CT image as well as intermediate artifacts-free image after the i-th recurrent stage.

The overall loss function is defined as the sum of all recurrent stages' residual loss, which is formulated as:

$$L(\Theta) = \sum_{i=1}^{N} \left\| \widehat{Res^i} - Res \right\|_F^2, \tag{8}$$

where Res is the residual between original sparse-view CT image and dense-view CT image, and Θ represents the network's parameters.

3 Experiments

3.1 Dataset

We evaluate our proposed network architecture on a dataset from AAPM (American Association of Physicsis in Medicine), which consists of 2378 CBCT images from 10 patients. The 3D volumes contain from 128 up to 343 slices per patient.

The performance measure is reported on 5-fold cross-validation, which average the values computed in the whole 5 loops. In each loop, among the 10 patients, 8 patients' data are used for training and the other 2 patients' data are used for testing.

For the training set, we use the 2D FBP reconstruction images form 60, 120, 180, 240 projection views as input, and the residual images between the dense view (720 views) reconstructions and the sparse view reconstructions are used as label. On account of the accuracy needed in medicine and the fact that down-sampling streak artifacts are approximately centrosymmetric, we use the original 512×512 images without cropping as input. What's more, in order to avoid the influence of outliers, we normalize the dataset according to the upper and lower 25 points of all pixels' value.

3.2 Experimental Setup

For fair comparison, we do not use any data augmentation. All architectures are trained by ADAM. As for learning rate, we choose the best decrease range and strategy for each method, such as 10^{-3} to 10^{-5} for R^2-Net, 5×10^{-4} to 10^{-5} for DD-Net [4] and 10^{-3} to 10^{-6} for TF U-Net [5]. The decrease range and strategy are chosen based on experiments. For evaluation metrics, we adopt PSNR and SSIM.

3.3 Experimental Results

Comparison between R^2-Net and other SOTA models In Table 1, we present the average PSNR and SSIM values of FBP, other SOTA methods and R^2-Net. All methods are significantly better than FBP. Our R^2-Net achieves much better results than other two SOTA methods.

As for the memory cost, we calculate total memory of the three methods, which is illustrated in Fig. 4. R^2-Net costs only 59.25% memory of DD-Net and 17.95% memory of TF U-Net.

SSIM/PSNR	30 views	60 views	120 views	180 views	240 views
FBP	0.459/19.83	0.661/25.54	0.904/33.56	0.976/39.88	0.992/44.83
DD-Net	0.945/33.74	0.963/37.13	0.983/40.54	0.993/43.66	0.995/47.11
TF U-Net	0.948/33.87	0.957/36.42	0.979/40.31	0.992/44.16	0.997/48.66
R^2-Net	0.951/34.31	0.971/38.07	0.991/43.97	0.996/47.19	0.998/49.68

Table 1. Quantitative comparison between R^2-Net and other SOTA methods on AAPM's CBCT dataset. The result is averaged by 5-fold cross-validation.

Ablation Study about Recurrent and Recursive mechanism In this part, we conduct experiments to compare the effect of different settings of recurrent and recursive numbers. This study may help us to understand how important roles are these two mechanisms play in the architecture. In Table 2, we report the ablation study's results. In the table, M indicates the number of recursive number, N indicates the number of recurrent number. It's obvious that both recurrent and recursive mechanism are beneficial to the performance. In detail, the recurrent mechanism contributes more than recursive mechanism.

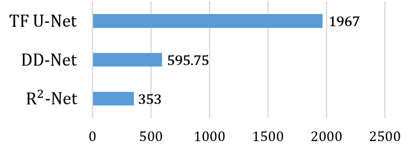

Fig. 4. Quantitative comparison among TF U-Net, DD-Net and R^2-Net.

(M, N)	(1, 1)	(3, 1)	(1, 3)	(3, 3)
PSNR	42.66	43.03	43.71	43.97
SSIM	0.9885	0.9889	0.9908	0.9912

Table 2. Quantitative comparison among different settings of R^2-Net in 120 views.

4 Conclusion

In this work, we propose a novel neural network architecture based on recurrent and recursive mechanism. Based on a more elaborate streak artifacts removal assumption, we incorporate recurrent mechanism to suppress artifacts stage by stage. What's more, in order to reduce the parameters cost, we introduce recursive mechanism into our architecture. Thanks to these two mechanisms, R^2-Net outperforms SOTA methods in the field.

References

1. Hu Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
2. Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. Neural Networks, IEEE Transactions on, 5(2), 1994. 1, 4
3. R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In ICML, 2013. 4
4. Zhang, Zhicheng, et al. "A sparse-view CT reconstruction method based on combination of DenseNet and deconvolution." IEEE transactions on medical imaging 37.6 (2018): 1407-1417.
5. Han, Yoseob, and Jong Chul Ye. "Framing U-Net via deep convolutional framelets: Application to sparse-view CT." IEEE transactions on medical imaging 37.6 (2018): 1418-1429.
6. Kak, Avinash C., and Malcolm Slaney. "Principles of computerized tomographic imaging." The Institute of Electrical and Electronics Engineers, Inc (1988).
7. Sidky, Emil Y., and Xiaochuan Pan. "Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization." Physics in Medicine & Biology 53.17 (2008): 4777.
8. Chen, Guang-Hong, Jie Tang, and Shuai Leng. "Prior image constrained compressed sensing (PICCS)." Photons Plus Ultrasound: Imaging and Sensing 2008: The Ninth Conference on Biomedical Thermoacoustics, Optoacoustics, and Acousto-optics. Vol. 6856. International Society for Optics and Photonics, 2008.
9. Liu, Yan, et al. "Adaptive-weighted total variation minimization for sparse data toward low-dose x-ray computed tomography image reconstruction." Physics in Medicine & Biology 57.23 (2012): 7923.

10. Liu, Yan, et al. "Total variation-stokes strategy for sparse-view X-ray CT image reconstruction." IEEE transactions on medical imaging 33.3 (2014): 749-763.
11. Chen, Yang, et al. "Artifact suppressed dictionary learning for low-dose CT image processing." IEEE transactions on medical imaging 33.12 (2014): 2271-2292.
12. Chen, Yang, et al. "Thoracic low-dose CT image processing using an artifact suppressed large-scale nonlocal means." Physics in medicine & biology 57.9 (2012): 2667.
13. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.
14. Kofler, Andreas, et al. "A U-Nets Cascade for Sparse View Computed Tomography." International Workshop on Machine Learning for Medical Image Reconstruction. Springer, Cham, 2018.
15. Thaler, Franz, et al. "Sparse-View CT Reconstruction Using Wasserstein GANs." International Workshop on Machine Learning for Medical Image Reconstruction. Springer, Cham, 2018.
16. Liao, Haofu, et al. "Adversarial Sparse-View CBCT Artifact Reduction." International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham, 2018.
17. Xie, Shipeng, et al. "Artifact removal using improved GoogLeNet for sparse-view CT reconstruction." Scientific reports 8 (2018).