Presented at the Fully 3D Image Reconstruction in Radiology and Nuclear Medicine Conference, Xi'an, China, June 2017

Low-Dose CT Streak Artifacts Removal using Deep Residual Neural Network

Heyi Li
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794–2424
Email: heyli@cs.stonybrook.edu

Klaus Mueller
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794–2424
Email: mueller@cs.stonybrook.edu

***Abstract*—In order to effectively reduce the risk of radiation exposure, low-dose CT using fewer projection views is becoming more and more preferred recently although it suffers significantly from poor image quality. The deep learning approach has demonstrated its effectiveness in natural image field. However, its application in medical imaging is still lacking. In this paper, we present a novel deep residual network structure to remove low-dose CT streak artifacts. Experiments using real patient clinical data were conducted to prove its superior performance.**

I. Introduction

X-ray Computed Tomography (CT) has been a major diagnostic modality in the clinical field for decades. However, the potential cancer risk caused by excessive radiation exposure has become an unsustainable risk. To counteract these shortcomings, low-dose CT imaging has become an active area of research and development. One popular way of reducing the radiation dose in CT scanning is to use fewer projection views. This, however, introduces strong streak artifacts in the CT reconstructions which make clinical diagnostics difficult.

Many methods have been proposed to reduce or even remove low-dose streak artifacts. One strategy is to impose additional constraints via regularization, such as penalizing large local variations. A popular method is Total Variation Minimization (TVM) [1]. However, TVM's application in clinical practice is limited since it is an iterative optimization method and therefore exhibits extensive computational costs. In addition, TVM also introduces artifacts on its own, such as patch-like constant regions that look unnatural to clinicians. Other approaches include Non-Local Means (NLM) filtering [2]. NLM reduces low-dose noise by averaging statistically similar neighboring pixels. A major drawback of NLM filters is that the neighboring pixels selected are equally contaminated with the same low-does noise. Ha [3] have proposed a database-assisted framework to overcome these types of problems. The database contains high-quality images taken at regular dose conditions. Pixels in the aligned images and selected based on structural similarity from the database are used to update corresponding noisy pixels. The main limiting factor for this framework is the large physical storage required for maintaining the database and the potential latency caused by searching through the database.

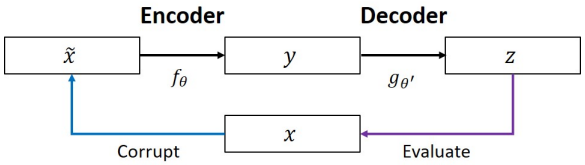

Fig. 1: Diagram for the Denoising Autoencoder

In recent years, deep learning methods have enjoyed significant success and have shown great potential in the field of natural image denoising [4], [5]. The concept of Denoising Autoencoder (DA) was first brought up by Pascal Vincent in [6]. Its goal is to reconstruct a clean image from a corrupted input by feeding it into a pre-trained neural network. As is shown in Fig.1, $\mathbf{x}$ represents a clean image and $\tilde{\mathbf{x}}$ denotes the corresponding noisy image. The corrupted input $\tilde{\mathbf{x}}$ is then encoded into a hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}})$, which in turn is decoded back to $\mathbf{z} = g_{\theta'}(\mathbf{y})$. During the training process, pairs of clean images and noisy images are required as input. The objective is to minimize a loss function which is defined as the squared error between reconstructed image $\mathbf{z}$ and clean image $\mathbf{x}$.

Despite the success of deep learning methods in natural image denoising, there have been few studies in field of medical imaging based on such an approach, especially in the area of low-dose CT. We propose a novel convolutional neural network structure to remove streak artifacts caused by an insufficient number of projection views. The next section, Section II, describes the neural network structure we have devised. The following section, Section III, presents first experimental results we have obtained with our prototype. The results confirm the great promise these types of networks have. While the training can take a large amount of time, once this has been accomplished, the streak removal is exceptionally fast and the quality acceptable. We end the paper with a discussion and pointers to future work.

II. Methods

The structure of our neural network is shown in Fig.2. Inspired by the idea of Denoising Autoencoder, our network consists of two key parts – the Encoder and the Decoder.

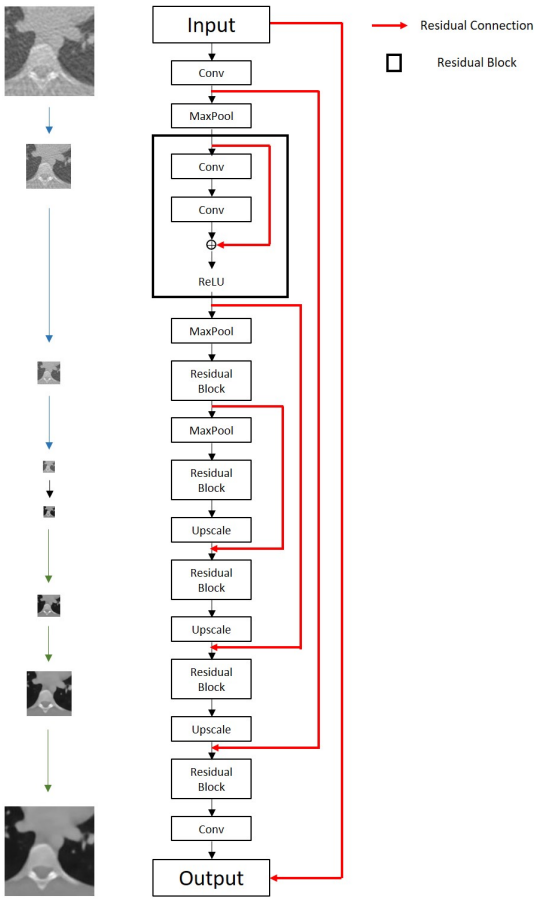

Fig. 2: Our proposed deep residual neural network

The Encoder takes low-dose images with streak artifacts as input and then delivers encoded representations to the Decoder. The Decoder, on the other hand, reconstructs images from the hidden representations and compares them with the noise-free images. The Encoder and the Decoder are structurally symmetric because of the opposite roles they play. The hidden represemtations have a smaller number of unts than the input and output. This forces them to learn a compressed representation of the input that captures the essential image structures.

A. Basic Structure

In the Encoder part, the actual image encoding is implemented by the combination of convolution layers and non-linear layers. Each convolution layer consists of multiple convolution kernels. The number of kernels within each convolution layer can be set arbitrarily. However, it is recommended to be powers of 2 for faster memory allocation considering the fact that the size of Random Access Memory (RAM) is always a power of 2. Some most widely used number of kernels in each convolution layer include 64, 128 and 256. The choice is in fact a trade-off between computational complexity and capabilies. Each convolution kernel performs convolution operations over the entire input, serving the role of one feature extractor. Therefore, more kernels will lead to more robust feature extraction and better overall performance. In the meantime, more kernels also mean more parameters to train since all the weights and the biases of each convolution layer are trainable variables.

After each convolution layer, one non-linear layer is added. Such layers are essential to the network because convolution layers are completely linear. Suppose we have two consecutive convolution layers denoted by subscript 1 and 2 respectively. Then the output y can be expressed as a function of the input x using the weights and biases of these two layers $y = W_2(W_1 * x + b_1) + b_2$. Since a convolution operation is linear, we can define a third convolution layer (W_3, b_3) with $W_3 = W_2 * W_1$ and $b_3 = W_2 * b_1 + b_2$. It is obvious that layer 3 is equivalent to layer 1 and 2 combined. Therefore, without non-linear layers in between, it is pointless to stack multiple convolution layers together. The most popular non-linear layer is called Rectified Linear Unit (ReLU) [7], which is defined as the maximum value of 0 and the input.

In the Decoder part, convolution layers and non-linear layers are also used but for another purpose. Instead of performing convolution operations over the original input, deconvolution operations are carried out over the encoded representation to reconstruct clean images. In practice, however, deconvolution operations are implemented using convolution layers. And because of the inverse operations performed by the Encoder and the Decoder, these two parts have symmetric structure.

B. Batch Normalization

Our basic network structure is already ready to be trained to remove streak artifacts. However, as we quickly realized during our experiments, this structure is far from optimal. The first modification we make is introducing the batch normalization layer [8] into our basic model.

Traditional convolutional neural networks tend to suffer from saturation and slow convergence caused by changes in the nput distribution during training. This is known as *internal covariate shift* . Let us assume we have a network with two succeeding layers 1 and 2. The loss function is defined as $loss = L_2(L_1(x, \Theta_1), \Theta_2)$, where x is the network input and Θ represents the parameters for each layer. The output of layer 1 is viewed as the input for layer 2. Then given a learning rate α, based on a simple gradient descend update method, Θ_2 will be updated using the following equation for each step.

$$\Theta_2 = \Theta_2 - \frac{\alpha}{N} \sum_{i=1}^{N} \frac{\partial L_2(L_1(x_i, \Theta_1), \Theta_2)}{\partial \Theta_2}$$

As we can see, the changes of parameters in previous layers have a direct impact on updating the parameters in the following layers. Since we assume that the training data and the test data follow the same distribution, the input distribution properties should be kept throughout the network. But in practice, it is difficult to maintain especially with non-linear layers. Therefore, Θ_2 has to readjust to compensate for the change in the input distribution slowing down convergence.

The batch normalization layer normalizes each batch to have zero mean and unit variance. By adding a batch normalization layer after each convolution layer we force the input of each layer to follow the same distribution. As a result, the training process of our network can be dramatically accelerated.

C. Residual Learning

As is discussed in [9], the depth of the network plays a vital role in achieving superior performance. However, as the network goes deeper, it is increasingly more prone to the notorious problem of degradation [10]. Such a problem is not caused by overfitting, but is rather due to the phenomenon that optimization of the loss function tends to get harder as more layers are stacked to the network. Therefore, instead of obtaining better performance as expected, the degradation problem causes deeper networks to return unsatisfying results simply because the objective function is not properly optimized during the training process. This prevents us from further exploiting the power of deep neural network.

In order to deal with this problem, we add residual blocks (see Fig.2) on top of the convolution layers in our basic model. We use x to represent the input and $F(x)$ to denote the original function of the network. By adding a direct bypass from the input to the output we recast the original mapping F to a new mapping $H : H(x) = F(x)+x$. The network is then trained to approximate a residual function $H(x)-x$, explicitly changing the function of the network to $F(x)+x$. This modification does not introduce any new trainable parameters to the network. Thus it does not increase its overall computation complexity. However, in practice, it has been shown that the new residual function is much easier to optimize than the original function leading to better performance for deeper networks.

D. Multi-scale vs. Single scale

One major drawback of the traditional Denoising Autoencoder is that its effective noise removal ability is strictly limited to Gaussian noise or salt-and-pepper noise which exhibit more local distribution patterns. Suppose we have one convolution layer and the size of its convolution kernel is 3×3. Then the receptive field of the output is simply 3×3 meaning that each pixel in the output comes from 9 pixels within a 3×3 region in the input. Comparing this with the size of the input image which is usually 512×512, this receptive field is quite small. This is why the Denoising Autoencoder has been rather successful in removing locally distributed noise.

In our case, however, low-dose CT streak artifacts exhibit a strong global pattern as well as a local one. As a result, extracting features on a single scale is not sufficient. In order to achieve the multi-scale learning needed to resolve the streak artifacts, we introduce pooling layers to our network. A pooling layer compresses the input image based on a given criterion and ratio. A 2×2 maxpooling layer is the most preferred choice. It only picks the pixel with the maximum value from a 2×2 region. We can then use subsequent convolution layers to extract features on the pooled images, which are on another scale. In this way, the receptive field can be doubled after each 2×2 pooling layer. Thus, by adding multiple maxpooling layers and corresponding residual connections to the network, we are able to achieve a multi-scale learning scheme.

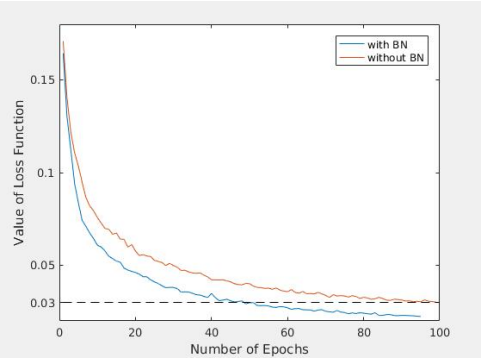

Fig. 3: Convergence rate comparison between networks with BN and without BN

III. Experiment

A. Dataset Preparation

Thanks to The Cancer Imaging Archive (TCIA) [11], we were able to download normal-dose abdomen CT images with size of 512×512 from 20 patients. We manually selected 30 images from each patient to form 20 groups of images. Fom each such image, we cropped out a region of size 128×128 that shared a similar structure for our experiments. One group was randomly chosen to be the test subject while images from other patients were used to train the network.

To enable a fair comparison, the original CT images were re-generated from 720 projections over 360° using a fan-beam geometry. Images reconstructed using 720 projections were then set as ground truth. We found that 720 projections were sufficient for simulating good quality normal-dose CT images. Our criteria were RMS error as well as manual examination. Finaly, another group of images were generated using 90 views to simulate low-dose CT images with strong streak artifacts.

B. Implementation and Training Details

The proposed neural network was implemented using the Theano framework [12] in a Python2 environment under Ubuntu 14.04 LTS. All experiments were performed using a NVIDIA M2070 Tesla graphics card with 6 GB RAM.

The number of kernels in each convolution layer was set to 64 with a size of 3×3. All weights were randomly initialized. We used the adaptive gradient algorithm (AdaGrad) [13] to update the parameters during the training process for its faster convergence compared with the traditional Stochastic Gradient Descent (SGD) algorithm. The initial learning rate was set to 0.01 and gradually reduced to 0.001 over 100 epochs. The training time for each epoch was about 43.5 seconds for a total time of 70 minutes. After the training was finished, the reconstruction time for each input image was around 13 ms.

(a) Ground Truth (b) Input (c) Total Variation (d) Single Scale (e) Proposed

Fig. 4: Reconstructed results of a slice of abdomen CT images by TV based denoising method and our proposed method

TABLE I: Quantitative measurements for the images in Fig.5

	Low-dose	TV	Single scale	Proposed
RMSE	41.77	11.70	8.52	**7.43**
PSNR	15.71	26.77	29.53	**30.71**
SSIM	0.59	0.64	0.86	**0.89**

C. Results and Comparison

In order to demonstrate the effectiveness of batch normalization layers, we used two networks as comparison, one with batch normalization layers and another without. The other structural parameters and training settings were all kept the same. The values of the loss functions for each epoch are plotted in Fig.3. As we can see, the network with batch normalization layers enjoys a convergence rate twice as fast.

We also constructed a network without pooling layers to demonstrate the advantage of a multi-scale structure over the single scale structure of such a network. Our proposed network model was evaluated by comparing the reconstructed results with the TV-based denoising method. Fig. 4 compares the streak-removal performance of the single-scale and multi-scale network, respectively, with that of a TVM implementation. We observe that the multi-scale processed image (labeled 'proposed') has fewer artifacts and more detail than any of images processed with the other methods. Likewise, Table I shows that its RMS error is lowest, while its PSNR and SSIM (Structural Similarity with the ground truth) is highest.

IV. Conclusion

We presented a novel deep residual network structure purposed to apply deep learning techniques for low-dose CT streak artifacts removal. We started with a basic model motivated by a standard Denoising Autoencoder and continued to improve its performance by modifying the structure and adding further layers. We then used clinical data for conducting preliminary experiments to examine the effectiveness of our framework. Given the encouraging results we believe that our approach has great potential for low-dose CT artifact removal.

In future work, we would like to apply our network to deal with more challenging tasks, such as the removal of metal artifact, beam hardening, and so on. Further, although we presented our method with fan-beam data only, we believe that our overall framework will naturally extend to data from cone-beam CT, helical CT, and others. The most significant requirement will be the availability of a larger multi-GPU server architecture. Experiments that use such a computational infrastructure are underway. Finally, we would also like to integrate the deep learning restoration into an iterative reconstruction pipeline to mitigate possible image feature suppression imposed by the network.

Acknowledgment

We thank NSF grant IIS 1527200 for partial support.

References

[1] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.

[2] A. Buades, B. Coll, and J.-M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 490–530, 2005.

[3] S. Ha and K. Mueller, "Low dose ct image restoration using a database of image patches," *Physics in medicine and biology*, vol. 60, no. 2, p. 869, 2015.

[4] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[5] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 341–349.

[6] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[7] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[11] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle *et al.*, "The cancer imaging archive (tcia): maintaining and operating a public information repository," *Journal of digital imaging*, vol. 26, no. 6, pp. 1045–1057, 2013.

[12] J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. Warde-Farley, I. Goodfellow, A. Bergeron *et al.*, "Theano: Deep learning on gpus with python," in *NIPS 2011, BigLearning Workshop, Granada, Spain*. Citeseer, 2011.

[13] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.