FAST DE-STREAKING METHOD USING PLAIN NEURAL NETWORK

Yuxiang Li

Ecole polytechnique, Palaiseau, France

Bo Zhang, Raoul Florent

Philips Research - Medisys, Suresnes, France

ABSTRACT

Radon transform fundamentally underlies reconstructions from computed tomography. Radon transform results in a sinogram where each coefficient represents the integral of the image on one line from one angle. From a full Radon sinogram, one can rebuild the initial image with Filtered Back Projection (FBP) algorithm. However, when information from some angles are missing, streaks appear on the output image using such method. In this work, we propose a method that works exclusively on the image domain: a de-streaker based on plain neural network. The objective is not to replace reconstruction methods, but to offer a fast post-processing that reduces artifacts from the output image, particularly when sinogram is unavailable.

***Index Terms*—** Neural network, Radon transform, Streak reduction

1. INTRODUCTION

Image reconstruction from Radon sinogram is traditionally solved with Filtered Back Projection (FBP) algorithm. When dealing with partial projections with a few angles, it is well known that the method results in streak artifacts. As a consequence, a range of algorithms have been proposed using the compressed sensing framework [1], which considerably reduce the artifacts [2][3]. These methods alternatively optimize the solution between sinogram and image domain, under certain sparsity regularization.

However, as sinogram can be unavailable, we practically may need to seek post-processing approaches for streak reduction. Under this scope, we are inspired by neural network-based machine learning methods for image restoration. For instance, Burger et al. [5] apply a plain neural network and show a competitive performance against BM3D, the state-of-the-art denoiser [4]. Later, Eigen et al. use a convolutional neural network to remove dirt or rain from photo [6]. Based on these successes, we are wondering if the learning framework may also be used to remove streak artifacts.

Contributions: Instead of using an iterative optimization framework, we propose a neural network-based de-streaker which can be used as a post-processing method for traditional FBP algorithm. We show that our method is both simple to implement and fast to compute in practice. Given proper datasets, our method could also be extended to improve other reconstruction approaches that may suffer from corrupted outputs.

2. OUR METHOD

We describe in this section a neural network-based post-processing method for FBP. The processing is applied on image patches and the de-streaked patches are reassembled to produce the output. The pipeline of our method is shown on Figure 1. Details of each part are given in the following sections.

2.1. Database creation

We build our database on a set of natural images from ImageNet. We start with images of size 256×256 and compute their full sinogram with 180 projection angles (from 0 to 179). The sinogram is then downsampled by keeping only 36 angles (20% of projections) regularly located between 0 and 179 degrees. Both images reconstructed by FBP from full and partial sinograms are stored. The following datasets are used for training and testing:

- *Training set:* 50,000 image pairs using images from ImageNet validation set 2010 [8] cropped to 256×256 and converted to grayscale.
- *Test set:* 12 grayscale image pairs of 256×256 pixels consisting of 11 images used in [4] plus Shepp-Logan phantom [9].

2.2. Neural network structure

The model we use is a plain neural network with 4 hidden layers of 2047 neurons. The input and the supervised output are 2 image patches of size 17×17, respectively coming from partial and full sinogram reconstructed images. Each patch is flattened to a vector of dimension 289. After each hidden layer, we insert an activation layer with PReLU [10] as our activation function. The PReLU function has an extra parameter per neuron that makes it highly adaptive. Our network is implemented using Keras, a neural network framework. We use a GTX860m GPU to run our model.

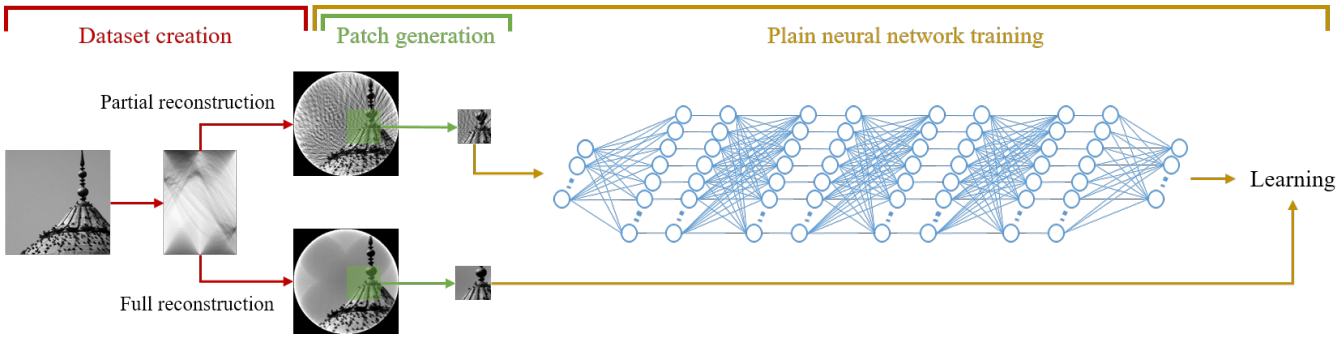

Fig. 1: Training pipeline of our neural network-based de-streaker. The result of dataset creation, e.g. reconstructed images, is stored locally. Patches are then generated on live during the training procedure.

2.3. Training phase

We use Mean Squared Error (MSE) as our loss function in order to minimize the distance between corrupted input (with streaks) and clean output (without streaks). We use RMSProp [11] as solver with an initial training rate of 0.0005. To accelarate the training, we train our network with batches of 100 image patches. Under these settings, the training process works as follow:

1. Pick randomly a pair of images from the *Training set*
2. Choose a random position and generate a pair of corrupted and clean patches
3. Rescale the patches within -2.5 and 2.5 [5]
4. Append the pair to the batch or feed the batch to the network

2.4. Testing phase

Our neural network works on a small patch. We need to de-streak all possible patches on the image in order the get a full-sized output. For this, we use the framing window technique that sums up overlapping patches with a Gaussian kernel. In practice, we set the stride between windows to 3 and the standard deviation of Gaussian kernel to 3. Do notice that the shape of the kernel can have an impact on the final outcome and it depends on the size of the patch we use.

3. RESULTS

We test our method on the *Test set* containing 12 grayscale images. We evaluate our result with both PSNR and visual perception. As we can see on Table 1, our de-streaker improves the PSNR of FBP reconstruction from partial projections by about 5 dB. However, de-streaked images tend to be slightly blurred. Figure 2 shows the result on "Cameraman" and "Phantom". Artifacts are almost completely removed from the output in expense of some local blurring. While blurring effect is less visible in images with lots of details, it is dominant in simple images such as "Phantom".

Image	FBP	De-streaked	Improvement
Barbara	23.15	28.42	+5.27
Boat	21.90	26.95	+5.05
Cameraman	20.17	26.33	+6.16
Couple	21.68	27.05	+5.37
Fingerprint	16.40	21.02	+4.62
Hill	23.86	28.69	+4.83
House	24.73	30.88	+6.15
Lena	23.26	28.94	+5.68
Man	22.93	27.96	+5.03
Peppers	22.98	27.15	+4.17
Pantagon	23.32	28.77	+5.45
Phantom	26.81	33.79	+6.98

Table 1: PSNRs on 12 images from *Test set*

4. DISCUSSIONS

In order to understand the de-streaking mechanism from a neural-network perspective, we visualize the input patterns and the output patterns of our network. The first weight matrix contains 2047 input patterns of size 17×17, they decide what information is useful for our model. In our case of reconstruction, we notice that many patterns look like Gabor filter which are used to detect the presence of streak. The output patterns are given by the last weight matrix. They represent image words that the network uses to build the output. Instead of showing edge detectors, we have more Dirac-liked patterns and background textures at the output, which is intuitive because we want to eliminate streaks (edges).

Visualizing the input and output patterns tells us how our network removes streaks from the image. It also shows why our model introduces a blurring effect: there are few oriented structures in the output weights for edge preservation. To further improve the performance, one may need to increase the

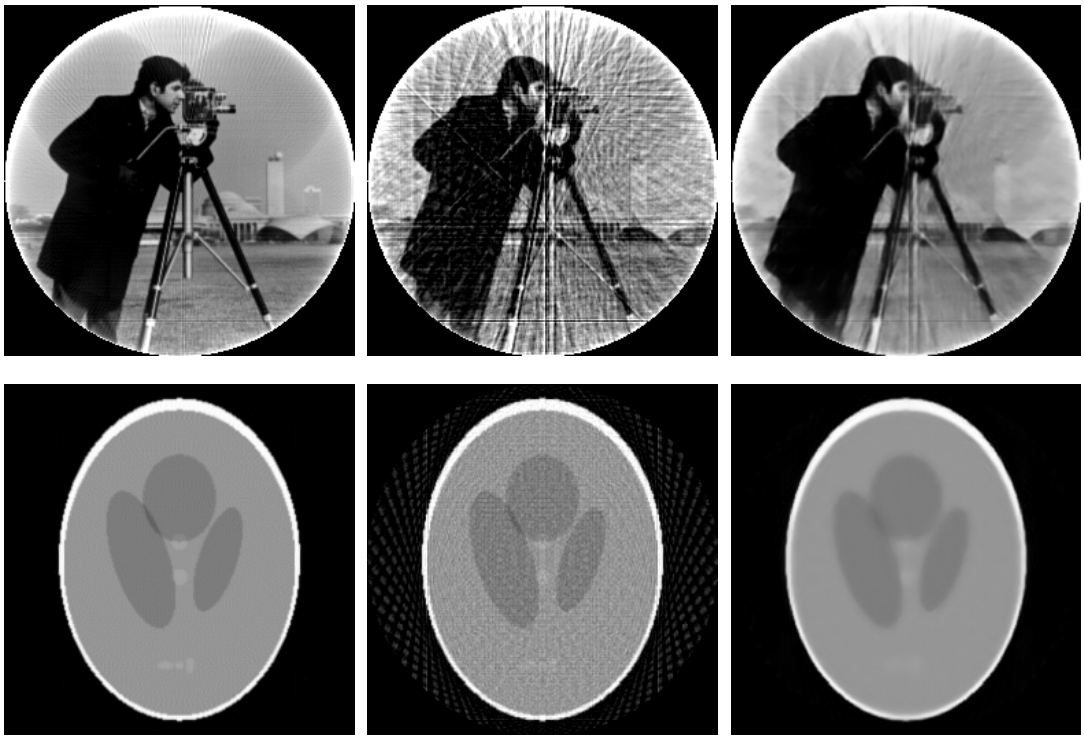

Fig. 2: Full reconstruction, partial reconstruction (with 20% of projections) and output of our de-streaking method for "Cameraman" and "Phantom"

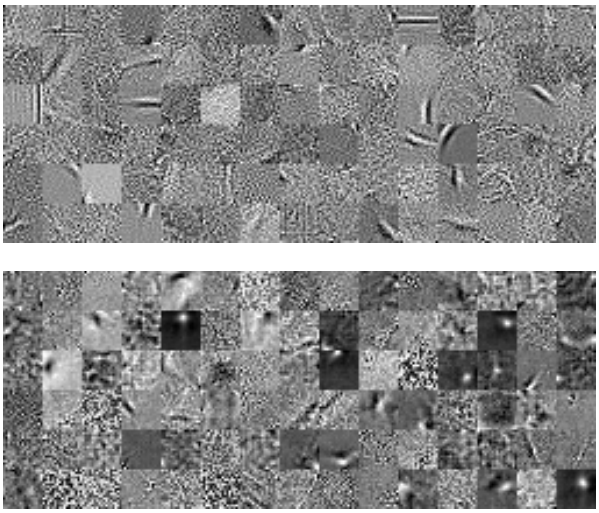

Fig. 3: Randomly selected input features (top) and output features (bottom) of our network

patch size at a cost of training time.

One may notice that existing reconstruction methods work mainly on both sinogram and image domains, whereas our method works exclusively on the image with one single shot. Our method de-streak an image of 256×256 pixels can be done in just a couple of milliseconds, which is insignificant compared to FBP algorithm. Moreover, by not depending on sinograms, one may alter the training set so that the post-processing works with other reconstruction methods.

5. CONCLUSION

In this paper, we apply neural network based learning algorithm for image de-streaking when reconstructed with partial projections. We demonstrate that this method, which is simple to implement, may considerably improve FBP performance with minimum computatinal cost. In the future, we may extend the same framework for overcoming various other image restoration artifacts.

6. REFERENCES

[1] Emmanuel J Candes, Justin K Romberg, and Terence Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on pure and applied mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[2] Curtis R Vogel and Mary E Oman, "Fast, robust total variation-based reconstruction of noisy, blurred im-

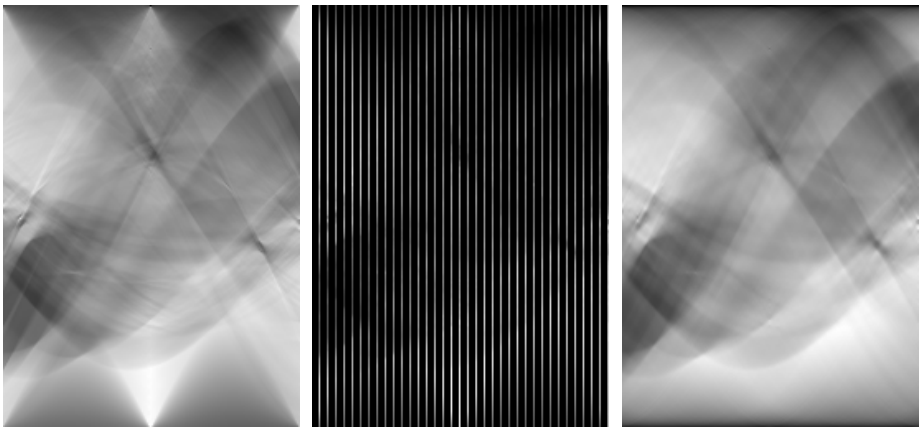

Fig. 4: "Cameraman": full sinogram with all 180 angles, partial sinogram with 36 angles (20% of projections), and the full sinogram of the output image

ages," *IEEE Transactions on Image Processing*, vol. 7, no. 6, pp. 813–824, 1998.

[3] Bin Dong, Jia Li, and Zuowei Shen, "X-ray ct image reconstruction via wavelet frame based regularization and radon domain inpainting," *Journal of Scientific Computing*, vol. 54, no. 2-3, pp. 333–349, 2013.

[4] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *Image Processing, IEEE Transactions on*, vol. 16, no. 8, pp. 2080–2095, 2007.

[5] Harold C Burger, Christian J Schuler, and Stefan Harmeling, "Image denoising: Can plain neural networks compete with bm3d?," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2392–2399.

[6] David Eigen, Dilip Krishnan, and Rob Fergus, "Restoring an image taken through a window covered with dirt or rain," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 633–640.

[7] Reza Nekovei and Ying Sun, "Back-propagation network and its configuration for blood vessel detection in angiograms," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 64–72, 1995.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[9] Lawrence A Shepp and Benjamin F Logan, "The fourier reconstruction of a head section," *Nuclear Science, IEEE Transactions on*, vol. 21, no. 3, pp. 21–43, 1974.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[11] Tijmen Tieleman and Geoffrey Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, pp. 2, 2012.