Deep Learning for Image Reconstruction

Cain Gantt[1], Yuanwei Jin[2]

[1]*Department of Mathematics, Georgia College and State University, Milledgeville, GA*
[2]*Department of Engineering and Aviation Sciences, University of Maryland Eastern Shore, Princess Anne, MD*

Abstract

Within the past twelve months, convolutional neural networks (CNNs) have been applied to enhance sparse-view reconstruction in computed tomography (CT) imaging. Due to the wide variety of parameters and network configurations possible, it remains unclear how to create and train a CNN to aid in image reconstruction. We investigate specific choices in the implementation of the CNN, including the network architecture, training parameters, and data preprocessing, to determine effects on the images produced by the network. Our implementation is modeled after an existing CNN used for sparse-view CT reconstruction, providing a baseline against which to compare the effectiveness of network outputs. By building and training a variety of networks with variations in configuration, we are able to examine the reconstruction artifacts present to guide further network design. Through the comparison different networks, we found that the most effective adjustments to parameters are those that increase the number of learnable values in the network, including increasing the total number of convolutional layers and increasing the number of feature maps produced by convolutional layers.

Methods

Training data are computed tomography (CT) scans obtained from The Cancer Imaging Archive. Preprocessing consists of calculating the Radon transformation (`radon` function in MATLAB) with 1000 views. Network inputs with undersampling rates of x5, x10, and x20 are created by computing the Inverse Radon (`iradon`) transformation with 200, 100, and 50 views, respectively, while the ground truth is created using all 1000 views. Network inputs are randomly mirrored and rotated to prevent overfitting during training.

Networks are implemented in MATLAB using the Neural Network Toolbox and trained using NVIDIA CUDA-enabled GPUs. Given a set of 1000 pairs of images (input and ground truth), networks are trained with 10 iterations (epochs) over the dataset. Learnable parameters are updated using a stochastic gradient descent algorithm with momentum of 0.99. Gradient updates are limited to 0.01 to prevent divergence during training. The initial learning rate is 0.01 and scales by 0.7943 each epoch.

The network architecture is modeled after the FBPConvNet created by McCann, *et al*. Each convolutional layer is followed by a batch normalization layer and a rectifier layer. Nested pairs of pooling and transposed convolution layers interspersed with convolutions increase and decrease the number of convolution filters in each stage. Connections that bypass pooling stages retain spatial information learned earlier in the network. We also include a residual connection to bypass all convolution layers before output.

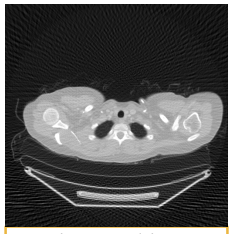

Fig 1. Network input (x10 undersampled)

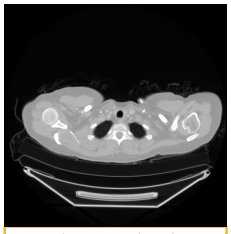

Fig 2. Ground Truth (No undersampling)

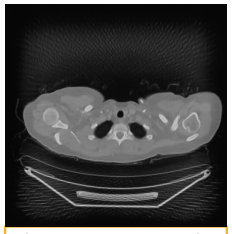

Fig 3. `imadjust` preprocessing (SNR: 13.90510)

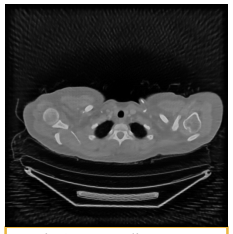

Fig 4. Two pooling stages (SNR: 14.23028)

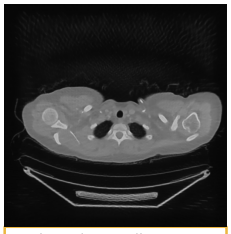

Fig 5. Three pooling stages (SNR: 19.94556)

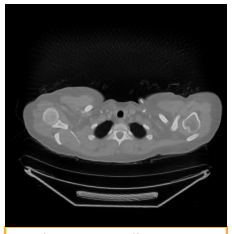

Fig 6. Four pooling stages (SNR: 19.19452)

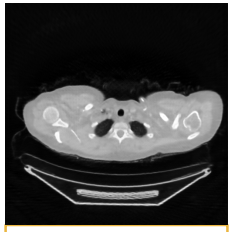

Fig 7. 32 initial filters (SNR: 21.29991)

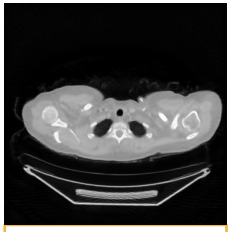

Fig 8. 128 initial filters (SNR: 22.07604)

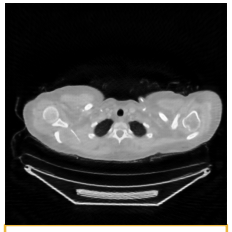

Fig 9. Batch size of 2 (SNR: 21.62261)

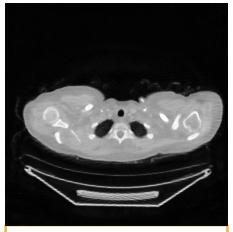

Fig 10. Batch size of 3 (SNR: 21.84305)

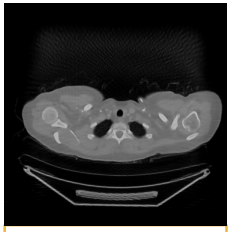

Fig 11. Five epochs (SNR: 16.92864)

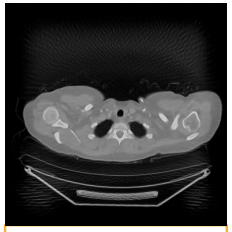

Fig 12. Twenty epochs (SNR: 15.14064)

Results

The networks examined produced output shown in the figures to the left. Figure 1 displays the reconstruction created with x10 undersampling that was used as input to the network, and Figure 2 displays the corresponding full-view reconstruction.

Figure 3 contains output from a network trained using contrast-enhanced input data. Each image is adjusted to saturate the darkest 1.5% of pixels to a value of zero (black) and the brightest 0.5% of pixels to one (white). This network has 64 filters in the first convolutional layer, and contains four pooling stages. The network was trained over 10 epochs with a batch size of one.

Figures 4-6 display output from networks with two, three, and four sets of pooling and transposed convolution layers, respectively. Each network starts with 64 convolution filters, reaching respective maxima of 256, 512, and 1024 filters in a single layer.

Figures 7 and 8 display output from networks each with two pooling stages and either 32 or 128 filters in the first convolutional layer, respectively.

Figures 9 and 10 display output from networks trained over ten epochs with batch sizes of two and three, respectively. These networks have two pooling stages, and contain 64 filters in their first convolution layers.

Figures 11 and 12 display output from networks trained over five and twenty epochs, respectively, and a batch size of one. These networks have four pooling stages, and contain 64 filters in their first convolution layers.

Restrictions imposed by software necessitate inconsistencies in the hardware utilized for training. Some networks were trained on GPU, others could run in parallel on a 12-core CPU, while others could only train in serial.

Conclusion

All networks seem to insufficiently suppress subsampling artifacts in the background region of the CT scans. This results in low signal-to-noise ratios (SNRs) for most network outputs. However, networks with a greater number of layers produce fewer artifacts and preserve finer details within the patient, where such qualities are more desirable. Further research could address the inability of the SNR to quantify this phenomenon.

Acknowledgements

This research was made possible by the Salisbury University EXERCISE REU program, funded by National Science Foundation grant CNS-1757017. Special thanks to System Administrator Richard Quackenbush.