Compressed Sensing and Computed Tomography with Deep Neural Networks

David Boublil, Michael Zibulevsky and Michael Elad

Technion, Computer Science Department
Neural Networks
Neural Networks

\[ u = w^T x \]

\[ y(u) = \begin{bmatrix} y(u_1) \\ \vdots \\ y(u_M) \end{bmatrix} \]

\[ y = \begin{pmatrix} 1 \\ \vdots \\ d \end{pmatrix} \psi \left( \sqrt{y} (w^T x + b) + c \right) \]
\[ y = f(x; \ w, b, v, c, d) = d^T y \left( \sqrt{v} y (w^T x + b) + c \right) \]

Training examples \( \{ x^{(k)}, y^{(k)} \}_{k=1}^{K} \)

\[ \min_{w, b, v, c, d} \left\{ E = \sum_{k=1}^{K} \left( f(x^{(k)}; w, b, v, c, d) - y^{(k)} \right)^2 \right\} \]
Photon detection in positron emission tomography using neural network [12]
Image denoising

Pattern recognition

Noisy image → NN → Clean image

20 → NN → +1
4 → NN → -1
Nonlinear compression
Linear compression (Compressed sensing)
Linear compression (Compressed sensing)

A New Single-Pixel Camera, http://dsp.rice.edu/cscamera
CS using Neural Network: Training Set

- Some image examples from the training set:
**CS using Neural Network – Results**

- Hidden layers: 12
- Activation: Tanh
- Redundancy factor: 2
- Block size: 16x16
- Training examples: 500k
CS using Neural Network – Results

Compression Rate = 0.2, PSNR = 25.1005dB

Compression Rate = 0.3, PSNR = 26.2258dB
CS using Neural Network – Results

Compression Rate = 0.2, PSNR = 34.7304dB
Compression Rate = 0.3, PSNR = 37.5494dB
Classical Compressed sensing

Measurements \( y \) = Sensing Matrix \( \Phi \) \( \rightarrow \) Signal \( x \)

Recovery: \[
\min_{x \in X} \| \Phi x - y \|_2^2
\]

Set \( X \) reflects prior knowledge about the signal
Classical Compressed sensing

- Recovery: \( \min_{x \in X} \| \Phi x - y \|_2^2 \)

- Set \( X \) reflects prior knowledge about the signal:
  1. Limited Total Variation (TV): \( \int \| \nabla x \| \, dx \leq a \)
  2. Sparse representability: \( x = \sum_i c_i \psi_i \), \( \| c \|_0 \leq b \)

Constraints may be put as a penalty term
L. Gan, J. E. Fowler and others...

- Transforms: DCT, Wavelet (DWT), Contourlet (CT), Dual tree wavelet (DDWT)

- Initialization of each block: \( y_i = \Phi_B x_i \)

- The algorithm:

  ```
  function x^{(i+1)} = SPL(x^{(i)}, y, \Phi_B, \Psi, \lambda)
  \hat{x}^{(i)} = \text{Wiener}(x^{(i)})
  for each block \( j \)
  \hat{x}_j^{(i)} = \hat{x}_j^{(i)} + \Phi_B^T (y - \Phi_B \hat{x}_j^{(i)})
  \tilde{x}^{(i)} = \Psi \hat{x}^{(i)}
  \tilde{x}^{(i)} = \text{Threshold}(\tilde{x}^{(i)}, \lambda)
  \tilde{x}^{(i)} = \Psi^{-1} \tilde{x}^{(i)}
  for each block \( j \)
  x_j^{(i+1)} = \tilde{x}_j^{(i)} + \Phi_B^T (y - \Phi_B \tilde{x}_j^{(i)})
  ```

- Further improvement by multi-scale (MS-BCS-SPL) and multi-hypothesis (MH-BCS-SPL) extensions.
Comparison with other Block-CS algorithms

Original

BCS-SPL-DDWT: PSNR = 26.6569dB

MS-BCS-SPL-DC: PSNR = 28.7494dB

MH-BCS-SPL: PSNR = 26.344dB

MH-MS-BCS-SPL: PSNR = 28.814dB

NN: PSNR = 29.3217dB
### Comparison with other Block-CS algorithms

<table>
<thead>
<tr>
<th>Methods</th>
<th>Lena 8%</th>
<th>Lena 10%</th>
<th>Lena 20%</th>
<th>Lena 30%</th>
<th>Lena 40%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCS-SPL-DDWT</td>
<td>26.45</td>
<td>27.94</td>
<td>30.94</td>
<td>33.39</td>
<td>35.38</td>
</tr>
<tr>
<td>MS-BCS-SPL</td>
<td>31.01</td>
<td>31.83</td>
<td>35.19</td>
<td>37.58</td>
<td>39.08</td>
</tr>
<tr>
<td>MH-BCS-SPL</td>
<td>28.04</td>
<td>28.73</td>
<td>33.05</td>
<td>35.44</td>
<td>37.14</td>
</tr>
<tr>
<td>MH-MS-BCS-SPL</td>
<td>30.87</td>
<td>31.99</td>
<td>35.68</td>
<td>37.86</td>
<td>40.06</td>
</tr>
<tr>
<td>BCS with NN</td>
<td>30.88</td>
<td>30.93</td>
<td>34.73</td>
<td>37.54</td>
<td>40.27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Goldhill 8%</th>
<th>Goldhill 10%</th>
<th>Goldhill 20%</th>
<th>Goldhill 30%</th>
<th>Goldhill 40%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCS-SPL-DDWT</td>
<td>26.65</td>
<td>27.12</td>
<td>29</td>
<td>30.73</td>
<td>32.04</td>
</tr>
<tr>
<td>MS-BCS-SPL</td>
<td>28.74</td>
<td>29.25</td>
<td>31.42</td>
<td>33.32</td>
<td>34.34</td>
</tr>
<tr>
<td>MH-BCS-SPL</td>
<td>26.34</td>
<td>27.1</td>
<td>30.12</td>
<td>32.41</td>
<td>34.09</td>
</tr>
<tr>
<td>MH-MS-BCS-SPL</td>
<td>28.81</td>
<td>29.3</td>
<td>31.75</td>
<td>33.63</td>
<td>35.57</td>
</tr>
<tr>
<td>BCS with NN</td>
<td>29.32</td>
<td>29.82</td>
<td>32.5</td>
<td>34.58</td>
<td>37.12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Mandrill 8%</th>
<th>Mandrill 10%</th>
<th>Mandrill 20%</th>
<th>Mandrill 30%</th>
<th>Mandrill 40%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCS-SPL-DDWT</td>
<td>20.2</td>
<td>20.63</td>
<td>21.68</td>
<td>22.7</td>
<td>23.93</td>
</tr>
<tr>
<td>MS-BCS-SPL</td>
<td>21.2</td>
<td>21.37</td>
<td>22.9</td>
<td>24.43</td>
<td>25.36</td>
</tr>
<tr>
<td>MH-BCS-SPL</td>
<td>19.86</td>
<td>20.01</td>
<td>21.93</td>
<td>23.57</td>
<td>25.02</td>
</tr>
<tr>
<td>BCS with NN</td>
<td>21.65</td>
<td>21.89</td>
<td>23.7</td>
<td>25.21</td>
<td>26.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Boat 8%</th>
<th>Boat 10%</th>
<th>Boat 20%</th>
<th>Boat 30%</th>
<th>Boat 40%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCS-SPL-DDWT</td>
<td>24.24</td>
<td>25.3</td>
<td>27.96</td>
<td>30</td>
<td>31.57</td>
</tr>
<tr>
<td>MS-BCS-SPL</td>
<td>26.91</td>
<td>27.53</td>
<td>30.47</td>
<td>32.52</td>
<td>33.9</td>
</tr>
<tr>
<td>MH-BCS-SPL</td>
<td>24.25</td>
<td>25.53</td>
<td>29.11</td>
<td>31.72</td>
<td>33.45</td>
</tr>
<tr>
<td>MH-MS-BCS-SPL</td>
<td>26.89</td>
<td>27.57</td>
<td>30.69</td>
<td>32.73</td>
<td>34.88</td>
</tr>
<tr>
<td>BCS with NN</td>
<td>27.6</td>
<td>27.8</td>
<td>31</td>
<td>33.23</td>
<td>35.46</td>
</tr>
</tbody>
</table>
Comparison with other Block-CS algorithms

<table>
<thead>
<tr>
<th>Methods</th>
<th>Compression rate</th>
<th>Average on 10 images</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8%</td>
<td>10%</td>
</tr>
<tr>
<td>BCS-SPL-DDWT</td>
<td>24.01</td>
<td>24.95</td>
</tr>
<tr>
<td>MS-BCS-SPL</td>
<td>26.75</td>
<td>27.35</td>
</tr>
<tr>
<td>MH-BCS-SPL</td>
<td>24.56</td>
<td>25.9</td>
</tr>
<tr>
<td>MH-MS-BCS-SPL</td>
<td>26.95</td>
<td>27.69</td>
</tr>
<tr>
<td>BCS with NN</td>
<td><strong>27.27</strong></td>
<td>27.43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Run Time (sec) (Compression rate 10%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCS-SPL-DDWT</td>
<td>15.59</td>
</tr>
<tr>
<td>MS-BCS-SPL</td>
<td>8.26</td>
</tr>
<tr>
<td>MH-BCS-SPL</td>
<td>49.93</td>
</tr>
<tr>
<td>MH-MS-BCS-SPL</td>
<td>24.49</td>
</tr>
<tr>
<td>BCS with NN</td>
<td><strong>0.34</strong></td>
</tr>
</tbody>
</table>
- Comparison with Romberg Algorithm [11]

- It is generally applied on the whole image, for this comparison we have used it on each block to get a fair comparison.

- Compression Rate = 5%

![Image showing original and compressed images with PSNR values]
Influence of Network depth

- Compression rate = 10%
- Number of examples = 500k
- Redundancy factor = 2
- Block size = 8x8
Influence of Network depth: Peppers

![Graph showing the relationship between Number of hidden layers and PSNR (dB)]
Influence of redundancy factor (layer width): Lena

- Compression rate = 10%
- Number of examples = 500k
- Depth: 8 hidden layers
- Block size = 8x8
Influence of block size: Boat

- Compression rate = 10%
- Number of examples = 500k
- Depth: 8 hidden layers
- Redundancy factor = 2
Influence of number of training examples

- Compression rate = 10%
- Block size = 8x8
- Depth: 8 hidden layers
- Redundancy factor = 2
Future Directions

- Use large networks and training sets to achieve patch size of 32x32 and bigger
- Global sensing / reconstruction using multiresolution neural networks
- Get back to nonlinear compression
Computed Tomography

Reconstruction Algorithm (FBP, PWLS, ...)

Output image

Sinogram values

\[ g_k = -\log\left(\frac{y_k}{\lambda_0}\right) \]

X-ray source

photons

Detectors

line \( k \)

photon counts \( y_k \)
Reconstruction Algorithms

- **Filtered Back-Projection (FBP)** => Linear Operator
  \[ T_{FBP} = \mathbf{R}^* \mathbf{F}_{\text{low Ram-Lak}} \]
  - Adjoint of Radon Transform (Back-projection)
  - LPF – prevents noise amplification at high frequencies
  - 1D convolution filter - Applied to each projection

- **Penalized Weighted Least Square (PWLS)** => Iterative algorithm
  \[ \hat{f} = \arg \min_f \| \log(y) - Af \|_D + \beta R(f) \]
  - Measured counts – projections
  - Radon transform approximation - Models scan process
  - Prior on clean CT image
Main Themes of Our Work [13]

Reducing Radiation Dose By Learning to Fuse Several Output Images
The Algorithm:
- Sweep the variance-resolution tradeoff
- Extract pixel neighborhood from each version
- Build a decision rule to perform the local fusion

Fusion Rule:
Use a regression to build one automatically, with an Artificial Neural Network (ANN)
FBP algorithm: sweep the cut-off frequency of the low-pass sinogram filter, and collect few images with different resolution-variance trade-off.

PWLS algorithm: perform the regular reconstruction while collecting versions along the iterations or sweeping different penalized weights $\beta$. 

Fusion: Which Images to Use?

- Initial image: Created with FBP
- Converged image: Standard PWLS result

versions from the iterations
Simple fully connected Neural Network

Activation function is an hyperbolic tangent

Caffe software was used to train the network

Training data (CT images) is taken from Visible Human Project
Empirical results - ANN FBP

Original

FBP- filter 1: 19.11 dB

FBP- filter 2: 26.27 dB

FBP- filter 3: 23.93 dB

Neural Network

Large NN

FBP- ANN: 29.4 dB
Empirical results - ANN PWLS

Original

PWLS - iter 60: 24.51 dB

PWLS - iter 120: 26.40 dB

Neural Network

PWLS - iter 400: 27.4 dB

Large NN

PWLS- ANN: 30.03 dB
Thank You !!!
Bibliography (1)


Bibliography (2)


