Supervised Deep Sparse Coding Networks

Xiaoxia Sun†, Nasser M. Nasrabadi‡ and Trac D. Tran†

† The Johns Hopkins University
‡ West Virginia University

October 7, 2018
Motivations

- Motivated by convolutional neural networks.
- One-layer sparse coding model does not work well on large dataset.
- Can sparse coding be efficiently extend to deep architecture?
Introduce SparseNet

Deep Sparse Coding Networks (SparseNet)

- Clean and neat framework based on sparse coding.
- Less tweaking on network architecture.
- Competitive performance on image classification using small model.
- Better interpretation of deep networks.
Overwhelmingly outperforms previous sparse coding-based model.

- **CIFAR-10**: 94.19% accuracy compared to 81.40%.  
- **CIFAR-100**: 80.07% accuracy compared to 60.80%.
- **STL-10**: 83.11% accuracy compared to 67.90%.
- **MNIST**: 0.36% error rate compared to 0.54%.

SparseNet on Image Classification

Exhibits competitive performance compared to deep neural networks (DNN).

- **CIFAR-10**: 94.19\% accuracy compared to 96.42\%. ²
- **CIFAR-100**: 80.07\% accuracy compared to 82.69\%.
- **STL-10**: 83.11\% accuracy compared to 76.29\%.
- **MNIST**: 0.36\% error rate compared to 0.21\%.

² Best reported result of deep neural network until 2017 under fair comparison.
Easily reproducible

Coded based on third-party deep learning toolbox (MatConvNet).

- Code available on GitHub: https://github.com/XiaoxiaSun/supervised-deep-sparse-coding-networks

Easy integration with deep learning schemes.

- Batch normalization
- Shortcut connection
- Dropout, Swapout and more...
Difference from previous approaches

- Dictionary is trained using end-to-end supervised learning based on error backpropagation.
- Nonlinear dimension reduction is employed to reduce the redundancy of sparse codes.
- Regularization parameters are adaptive to the given task.
- Render state-of-the-art performance.

Figure: We extend sparse coding to a 14-layer deep architecture.
Inference with nonnegative sparse coding

Enforce nonnegative constraint on sparse codes.

\[ \alpha^* = \arg \min_{\alpha > 0} \frac{1}{2} \| x - D\alpha \|_2^2 + \lambda_1 \| \alpha \|_1 + \frac{\lambda_2}{2} \| \alpha \|_2^2, \]  

(1)

Advantages of nonnegative sparse coding.

- Known clustering effect, similar to semi-nonnegative matrix factorization (semi-NMF).
Explosion of feature dimension

One-layer sparse coding cannot be naturally extended to multilayer architecture

- Employ wide dictionary each layer is computationally infeasible.
- Needs to reduce the dimensionality of sparse codes before passing to the deeper layer.
Bottleneck module

Reduce the dimensionality of sparse codes.

\[
\begin{align*}
\mathbf{x}^{(1)} &= \mathbf{D}^{(1)} \\
\mathbf{a}^{(1)} &= \arg\min_{\mathbf{a}^{(1)} > 0} \frac{1}{2} \| \mathbf{y} - \mathbf{D}^{(1)} \mathbf{a}^{(1)} \|_2^2 + \lambda_1 \| \mathbf{a}^{(1)} \|_1 + \frac{\lambda_2}{2} \| \mathbf{a}^{(1)} \|_2^2 \\
\mathbf{a}^{(2)*} &= \arg\min_{\mathbf{a}^{(2)} > 0} \frac{1}{2} \| \mathbf{a}^{(1)} - \mathbf{D}^{(2)} \mathbf{a}^{(2)} \|_2^2 + \lambda_1 \| \mathbf{a}^{(2)} \|_1 + \frac{\lambda_2}{2} \| \mathbf{a}^{(2)} \|_2^2
\end{align*}
\]
The deep SparseNet is constructed by repeatedly stacking multiple bottleneck modules.

Bottleneck module consists of one expansion layer and one reduction layer.

Contains 14 sparse coding layers.
Dimension reduction leads to clustering effect

SparseNet

CNN
Dimension reduction leads to clustering effect

A closer look:
Supervised learning for SparseNet

Formulation with multilevel optimization:

\[
\begin{align*}
\min_{\theta} & \quad \frac{1}{S} \sum_{s=1}^{S} L(y_s, f(A_s^{(h)}, w)) + \frac{\mu}{2} R(\theta), \\
\text{s.t.} & \quad \alpha_s^{(H)^*} = \arg \min_{\alpha_s^{(H)} \geq 0} F(D^{(H)}, \lambda^{(H)}, x_s^{(H)}), \\
& \quad s.t. \quad \alpha_s^{(1)^*} = \arg \min_{\alpha_s^{(1)} \geq 0} F(D^{(1)}, \lambda^{(1)}, x_s^{(1)}), \\
& \quad s.t. \quad \lambda^{(h)} > 0, \quad x_s^{(h)} = \psi(\alpha_s^{(h-1)^*}), \quad \forall h = 1, \ldots, H, \\
\end{align*}
\]

(2)

where \( \theta = \{D^{(h)}, \lambda^{(h)}\}_{h=1}^{H} \).
CIFAR-10 and CIFAR-100

- 50,000 training images, 10,000 testing images.
- Evenly split into 10 (CIFAR-10) or 100 (CIFAR-100) classes.
## Classification on CIFAR-10 and CIFAR-100

**Table:** Classification Error (%) on CIFAR-10 and CIFAR-100.

<table>
<thead>
<tr>
<th>Method</th>
<th># Params</th>
<th># Layers</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCKN [1]</td>
<td>10.50M</td>
<td>10</td>
<td>10.20</td>
<td>-</td>
</tr>
<tr>
<td>OMP [2]</td>
<td>0.70M</td>
<td>2</td>
<td>18.50</td>
<td>-</td>
</tr>
<tr>
<td>PCANet [3]</td>
<td>0.28B</td>
<td>3</td>
<td>21.33</td>
<td>-</td>
</tr>
<tr>
<td>NOMP [4]</td>
<td>1.09B</td>
<td>4</td>
<td>18.60</td>
<td>39.92</td>
</tr>
<tr>
<td>NiN [5]</td>
<td>-</td>
<td>-</td>
<td>8.81</td>
<td>35.68</td>
</tr>
<tr>
<td>DSN [6]</td>
<td>1.34M</td>
<td>7</td>
<td>7.97</td>
<td>36.54</td>
</tr>
<tr>
<td>WRN [7]</td>
<td>36.5M</td>
<td>28</td>
<td>4.00</td>
<td>19.25</td>
</tr>
<tr>
<td>ResNet-110 [8]</td>
<td>0.85M</td>
<td>110</td>
<td>6.41</td>
<td>27.22</td>
</tr>
<tr>
<td>ResNet-1001 v2 [9]</td>
<td>10.2M</td>
<td>1001</td>
<td>4.92</td>
<td>27.21</td>
</tr>
<tr>
<td><strong>ResNext-29 [10]</strong></td>
<td><strong>68.10M</strong></td>
<td><strong>29</strong></td>
<td><strong>3.58</strong></td>
<td><strong>17.31</strong></td>
</tr>
<tr>
<td>SCN-1</td>
<td>0.17M</td>
<td>15</td>
<td>8.86</td>
<td>25.08</td>
</tr>
<tr>
<td>SCN-2</td>
<td>0.35M</td>
<td>15</td>
<td>7.18</td>
<td>22.17</td>
</tr>
<tr>
<td><strong>SCN-4</strong></td>
<td><strong>0.69M</strong></td>
<td><strong>15</strong></td>
<td><strong>5.81</strong></td>
<td><strong>19.93</strong></td>
</tr>
</tbody>
</table>
STL-10

- 5,000 labeled training images, 8,000 testing images.
- Evenly split into 10 classes.
Classification on STL-10

Table: Classification Accuracy (%) on STL-10.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Params</th>
<th>#Layers</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWWAE [12]</td>
<td>10.50M</td>
<td>10</td>
<td>74.33</td>
</tr>
<tr>
<td>Deep-TEN [13]</td>
<td>25.60M</td>
<td>50</td>
<td>76.29</td>
</tr>
<tr>
<td>SCN-4</td>
<td>0.69M</td>
<td>15</td>
<td>83.11</td>
</tr>
</tbody>
</table>

- Follow the supervised training protocol of [13].
- Training takes about 25 hours on a server with 4 Nvidia Tesla P40 GPUs.
- State-of-the-art performance with few labeled samples.
Classification on MNIST

Table: Classification Error (%) on MNIST.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Params</th>
<th>#Layers</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKN [14]</td>
<td>-</td>
<td>2</td>
<td>0.39</td>
</tr>
<tr>
<td>ScatNet [15]</td>
<td>-</td>
<td>3</td>
<td>0.43</td>
</tr>
<tr>
<td>PCANet [3]</td>
<td>-</td>
<td>3</td>
<td>0.62</td>
</tr>
<tr>
<td>S-SC [16]</td>
<td>-</td>
<td>1</td>
<td>0.84</td>
</tr>
<tr>
<td>TDDL [17]</td>
<td>-</td>
<td>1</td>
<td>0.54</td>
</tr>
<tr>
<td>SCN-4</td>
<td>0.69M</td>
<td>15</td>
<td>0.36</td>
</tr>
</tbody>
</table>

- Train with 25 epochs.
- Training takes about 4 hours on a server with 4 Nvidia Tesla P40 GPUs.
- Highest accuracy among sparse coding-based models.
Future works - Simplify backpropagation rule

- Dictionary update require matrix inversion:

\[
\frac{\partial L}{\partial d_{jk}} = \left( \frac{\partial L}{\partial \alpha} \right)_{\Lambda}^\top \cdot (D_\Lambda^\top D_\Lambda + \lambda_2 I_{|\Lambda|})^{-1} \left( \frac{\partial D_\Lambda^\top x}{\partial d_{jk}} - \frac{\partial D_\Lambda^\top D_\Lambda}{\partial d_{jk}} \alpha_\Lambda \right).
\] (3)

- Around 80% computation time are spent for matrix inversion.
- Find possible ways to avoid it.
Conclusion

- Dictionary learning can efficiently adapt features to the given dataset.
- Extending sparse coding to multilayer architecture is able to substantially improve the performance.
- Computational complexity is much higher than deep neural network during backpropagation.
- Large potentials of improving performance of SparseNet.


Conclusion


