An empirical comparison of CNNs and other methods for classification of protein subcellular localization with microscopy images

Mengli Xiao, Wei Pan

Division of Biostatistics
University of Minnesota

June 5, 2018
Outline

1 Background
   - Protein subcellular localization
   - Data description
   - Implementation

2 A Convolutional Neural Network: DeepYeast
   - CNN model structure
   - Result

3 Residual Neural Network
   - ResNet model structures
   - Result

4 Feature extraction and transfer learning
   - Definition
   - Result

5 Summary
   - Comparison of different methods
   - Discussion
Background

1. Protein subcellular localization
2. Data description
3. Implementation

A Convolutional Neural Network: DeepYeast

- CNN model structure
- Result

Residual Neural Network

- ResNet model structures
- Result

Feature extraction and transfer learning

- Definition
- Result

Summary

- Comparison of different methods
- Discussion
Protein subcellular localization

- A protein’s subcellular localization \(\iff\) function
- Spatial temporal variation of a protein’s location results from genetic and environmental perturbations
- High-throughput imaging: image classification
  - proteins are fluorescently labeled to track their locations within a cell;
- Why automating?
- Large, but not so large, amounts of data: deep learning? others?
Data description

- Data: Pärnamaa and Parts (2017);
- Each image contains a single cell.

**Figure 1:** DeepYeast dataset overview with 4 images per category (Pärnamaa and Parts, 2017)
# Data description

<table>
<thead>
<tr>
<th>Subcellular categories</th>
<th>training</th>
<th>validation</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell periphery</td>
<td>6924</td>
<td>961</td>
<td>1569</td>
</tr>
<tr>
<td>Cytoplasm</td>
<td>6935</td>
<td>1223</td>
<td>1276</td>
</tr>
<tr>
<td>Endosome</td>
<td>2692</td>
<td>697</td>
<td>689</td>
</tr>
<tr>
<td>ER</td>
<td>6195</td>
<td>1393</td>
<td>1755</td>
</tr>
<tr>
<td>Golgi</td>
<td>2770</td>
<td>208</td>
<td>382</td>
</tr>
<tr>
<td>Mitochondria</td>
<td>6547</td>
<td>1560</td>
<td>1243</td>
</tr>
<tr>
<td>Nuclear Periphery</td>
<td>6661</td>
<td>1252</td>
<td>1164</td>
</tr>
<tr>
<td>Nucleolus</td>
<td>7014</td>
<td>1147</td>
<td>1263</td>
</tr>
<tr>
<td>Nuclei</td>
<td>6440</td>
<td>1312</td>
<td>1627</td>
</tr>
<tr>
<td>Peroxisome</td>
<td>1683</td>
<td>297</td>
<td>164</td>
</tr>
<tr>
<td>Spindle</td>
<td>4713</td>
<td>1517</td>
<td>781</td>
</tr>
<tr>
<td>Vacuole</td>
<td>6426</td>
<td>936</td>
<td>587</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>65000</td>
<td>12500</td>
<td>12500</td>
</tr>
</tbody>
</table>
Implementation

- Keras in Tensorflow - CNNs
- Python sklearn - RF, XGBoost
- R - CATCH (Pan et al., 2018a,b)
1 Background
   - Protein subcellular localization
   - Data description
   - Implementation

2 A Convolutional Neural Network: DeepYeast
   - CNN model structure
   - Result

3 Residual Neural Network
   - ResNet model structures
   - Result

4 Feature extraction and transfer learning
   - Definition
   - Result

5 Summary
   - Comparison of different methods
   - Discussion
DeepYeast (11-layer CNN) Model structure

- A 11-layered CNN; similar to the first few layers of VGG-19;
- VGG-19 was trained on the (ImageNet) ILSVRC dataset consisting of natural objects, aircraft, etc.
- Several papers: similar CNNs for the current problem.
## VGG-19 and DeepYeast (11-layered) model structure

<table>
<thead>
<tr>
<th>VGG-19</th>
<th>DeepYeast</th>
</tr>
</thead>
<tbody>
<tr>
<td>19 weight layers</td>
<td>11 weight layers</td>
</tr>
<tr>
<td>Input: 224×224×3</td>
<td>Input: 64×64×3</td>
</tr>
<tr>
<td>conv3-64</td>
<td>conv3-64</td>
</tr>
<tr>
<td>conv3-64</td>
<td>conv3-64</td>
</tr>
<tr>
<td>maxpool 2×2</td>
<td>maxpool 2×2</td>
</tr>
<tr>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
<tr>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
<tr>
<td>maxpool 2×2</td>
<td>maxpool 2×2</td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td>maxpool 2×2</td>
<td>maxpool 2×2</td>
</tr>
<tr>
<td>conv3-512</td>
<td>Fully-connected layer-512</td>
</tr>
<tr>
<td>conv3-512</td>
<td>Dropout-0.5</td>
</tr>
<tr>
<td>conv3-512</td>
<td>Fully-connected layer-512</td>
</tr>
<tr>
<td>conv3-512</td>
<td>Dropout-0.5</td>
</tr>
<tr>
<td>maxpool 2×2</td>
<td>Fully-connected layer-12 (softmax)</td>
</tr>
<tr>
<td>conv3-512</td>
<td>(BN added except for the last FC layer)</td>
</tr>
<tr>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>conv3-512</td>
<td></td>
</tr>
<tr>
<td>maxpool 2×2</td>
<td></td>
</tr>
<tr>
<td>Fully-connected layer-4096</td>
<td></td>
</tr>
<tr>
<td>Dropout-0.5</td>
<td></td>
</tr>
<tr>
<td>Fully-connected layer-4096</td>
<td></td>
</tr>
<tr>
<td>Dropout-0.5</td>
<td></td>
</tr>
<tr>
<td>Fully-connected layer-1000 (softmax)</td>
<td></td>
</tr>
<tr>
<td># of parameters is 144,000,000</td>
<td># of parameters is 3,128,908</td>
</tr>
</tbody>
</table>
CNN Model structure

Input image, $64 \times 64 \times 3$

Conv layer with $64 \times 3 \times 3$ filters, padding=1, stride=1
Output dimension: $64 \times 64 \times 64$

Conv layer with $64 \times 3 \times 3$ filters, padding=1, stride=1
Output dimension: $64 \times 64 \times 64$

2 × 2 Maxpooling
Output dimension: $32 \times 32 \times 64$

Conv layer with $128 \times 3 \times 3$ filters, padding=1, stride=1
Output dimension: $32 \times 32 \times 128$

Conv layer with $128 \times 3 \times 3$ filters, padding=1, stride=1
Output dimension: $32 \times 32 \times 128$

2 × 2 Maxpooling
Output dimension: $16 \times 16 \times 128$

Conv layer with $256 \times 3 \times 3$ filters, padding=1, stride=1
Output dimension: $16 \times 16 \times 256$

Conv layer with $256 \times 3 \times 3$ filters, padding=1, stride=1
Output dimension: $16 \times 16 \times 256$

Conv layer with $256 \times 3 \times 3$ filters, padding=1, stride=1
Output dimension: $16 \times 16 \times 256$

Conv layer with $256 \times 3 \times 3$ filters, padding=1, stride=1
Output dimension: $16 \times 16 \times 256$

2 × 2 Maxpooling
Output dimension: $8 \times 8 \times 256$

Fully connected with 512 neurons
Output dimension: $512 \times 1$

Fully connected with 512 neurons
Output dimension: $512 \times 1$

Fully connected with 12 neurons
Output dimension: $12 \times 1$
The test accuracy is 0.8512 (vs 0.8671 in the paper).
1 Background
   - Protein subcellular localization
   - Data description
   - Implementation

2 A Convolutional Neural Network: DeepYeast
   - CNN model structure
   - Result

3 Residual Neural Network
   - ResNet model structures
   - Result

4 Feature extraction and transfer learning
   - Definition
   - Result

5 Summary
   - Comparison of different methods
   - Discussion
Motivation

Figure 2: Poorer performance with deeper layers (He et al., 2016)

Figure 3: Convolution layer learns the residual features left by the identity skip connection/shortcut
Residual neural networks

- Convolutional layer blocks; no fully-connected layers;
- Identity shortcut was shown to perform well.
- We tried 18- and 50-layered ResNet, Res18 and Res50.
## Table 3: Model structure

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1_x</td>
<td>3 x 3, 64</td>
<td>7 x 7, 64</td>
<td>7 x 7, 64</td>
<td>7 x 7, 64</td>
<td>3 x 3, 16</td>
<td>3 x 3, 16</td>
</tr>
<tr>
<td>conv2_x</td>
<td>3 x 3, 128</td>
<td>3 x 3, 64</td>
<td>1 x 1, 64</td>
<td>3 x 3, 16 x 4</td>
<td>3 x 3, 16 x 2</td>
<td>3 x 3, 16 x 2</td>
</tr>
<tr>
<td>conv3_x</td>
<td>3 x 3, 256</td>
<td>3 x 3, 64</td>
<td>1 x 1, 128</td>
<td>3 x 3, 32 x 4</td>
<td>3 x 3, 32 x 2</td>
<td>3 x 3, 32 x 2</td>
</tr>
<tr>
<td>conv4_x</td>
<td>3 x 3, 64</td>
<td>1 x 1, 256</td>
<td>1 x 1, 1024</td>
<td>3 x 3, 32 x 4</td>
<td>3 x 3, 32 x 2</td>
<td>3 x 3, 32 x 2</td>
</tr>
<tr>
<td>conv5_x</td>
<td>3 x 3, 64</td>
<td>1 x 1, 512</td>
<td>3 x 3, 512</td>
<td>1 x 1, 64</td>
<td>3 x 3, 16 x 4</td>
<td>3 x 3, 16 x 2</td>
</tr>
</tbody>
</table>

max pooling

[512-d fc] x 2

12-d fc (softmax)

average pooling, 12-d fc (softmax)
Test accuracy of residual neural networks

- Res18 and Res50 performed better than their plain versions;
- Plain 50 worse than plain 18; but Res50 better than Res18;
- More benefits with 50 layers.

Table 4: Comparison of accuracy among different methods

<table>
<thead>
<tr>
<th>Network</th>
<th>Training time</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>plain 18</td>
<td>1.75 h</td>
<td>0.8432</td>
</tr>
<tr>
<td>Res 18</td>
<td>1.75 h</td>
<td>0.8708</td>
</tr>
<tr>
<td>plain 50</td>
<td>13 h</td>
<td>0.8190</td>
</tr>
<tr>
<td>Res 50</td>
<td>12.75 h</td>
<td>0.8856</td>
</tr>
</tbody>
</table>
Feature extraction and transfer learning

1. Background
   - Protein subcellular localization
   - Data description
   - Implementation

2. A Convolutional Neural Network: DeepYeast
   - CNN model structure
   - Result

3. Residual Neural Network
   - ResNet model structures
   - Result

4. Feature extraction and transfer learning
   - Definition
   - Result

5. Summary
   - Comparison of different methods
   - Discussion
Definition and advantages

- The last one or few layers of a pretrained neural network are replaced by new classifiers.
- More accurate and faster (vs. without feature extraction).
Use trained network as a feature extractor

- Replace the last fully-connected layer of the base CNN model (DeepYeast) with a random forest and an XGBoost:
  - Compared to using vectorizing-image input, the test accuracy is improved (0.85 vs 0.6)
  - Faster: with 512 extracted features vs the original 12288 (= 64 × 64 × 3) features

- Replace all the fully-connected layers of the VGG-19 model with new fully-connected layers, a random forest and an XGBoost respectively,
  - Very quick compared to training a neural network from scratch; decent test accuracy (0.73, 0.66, 0.72)
1 Background
   - Protein subcellular localization
   - Data description
   - Implementation

2 A Convolutional Neural Network: DeepYeast
   - CNN model structure
   - Result

3 Residual Neural Network
   - ResNet model structures
   - Result

4 Feature extraction and transfer learning
   - Definition
   - Result

5 Summary
   - Comparison of different methods
   - Discussion
# Summary

## Table 5: Comparison of accuracy between different methods

<table>
<thead>
<tr>
<th>Network</th>
<th>Training time</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepYeast (11-layer CNN)</td>
<td>6 h</td>
<td>0.851</td>
</tr>
<tr>
<td>CNN (18-layer)</td>
<td>1.75 h</td>
<td>0.843</td>
</tr>
<tr>
<td>Res18</td>
<td>1.75 h</td>
<td>0.871 – 0.891</td>
</tr>
<tr>
<td>ResNet 18 (He et al., 2016)</td>
<td>2.45 h</td>
<td>0.853</td>
</tr>
<tr>
<td>CNN (50-layer)</td>
<td>13 h</td>
<td>0.819</td>
</tr>
<tr>
<td>ResNet 50 (He et al. 2016)</td>
<td>12.75 h</td>
<td>0.886</td>
</tr>
<tr>
<td>Wide ResNet (widening factor 2)</td>
<td>46 h</td>
<td>0.853</td>
</tr>
<tr>
<td>Random Forest (v-images; 1000 trees)</td>
<td>1.68 h</td>
<td>0.600</td>
</tr>
<tr>
<td>XGBoost (v-images 800 trees)</td>
<td>10 h</td>
<td>0.6</td>
</tr>
<tr>
<td>Feature extraction by DeepYeast:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random Forest</td>
<td>10 min</td>
<td>0.850</td>
</tr>
<tr>
<td>XGBoost</td>
<td>1 h</td>
<td>0.840</td>
</tr>
<tr>
<td>Feature extraction by VGG-19 (transfer learning):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC layers</td>
<td>3 min</td>
<td>0.730</td>
</tr>
<tr>
<td>Random Forest (800 trees)</td>
<td>12 min</td>
<td>0.660</td>
</tr>
<tr>
<td>XGBoost (1000 trees)</td>
<td>14 h</td>
<td>0.722</td>
</tr>
</tbody>
</table>
Discussion

- CNNs performed best, though not a thorough evaluation! Why? images, a large dataset, ...
- Other statistical methods: RF, Boosting, SVM good, but not tailored to images...
  BUT, some new stat methods: could not even run...
Acknowledgment

- Funded by NIH, NSF.
- Thank you!
References


