Face Recognition: A Convolutional Neural Network Approach

Instructor: Bhiksha Raj
Student: T. Hoang Ngan Le
The Problem

Testing

Training

Recognition
Proposed System - Flowchart

Images

Image Sampling

Dimensionality Reduction

• SOM
• KL transform

Convolutional Neural Network

• Full Connected
• Nearest Neighbor
• Multi-layer Perceptron

Classification

Identification
A window is stepped over the image and a vector is created at each location.
Dimensionality Reduction - SOM

- Image Sampling
- Dimensionality Reduction
- Convolutional Neural Network
- Classification

Input vector $x_1, x_2, \ldots, x_n$

$w_{ij}$

$R^{25}$

$R^3$
Dimensionality Reduction - SOM

Operations:
- Select random input
- Compute winner neuron
- Update neurons
- Repeat for all input data
- Classify input data

1. Select random input
2. Compute winner neuron
3. Update neurons
4. Repeat for all input data
5. Classify input data
6. Repeat for all input data
Dimensionality Reduction - KL Transform
Dimensionality Reduction - KL Transform

• PCA
  – Objective function: \( u^T C u - \lambda (u^T u - 1) \)

• Karhunen-Loeve (KL) transform
  – Objective function:

\[
\begin{align*}
  u_2^T C u_2 - \lambda (u_2^T u_2 - 1) - \phi u_2^T u_1 \\
  \hat{x}_{k-1} = x - \sum_{i=1}^{k-1} u_i u_i^T x \\
  u_k = \arg \max_{\|u\|=1} E[(u^T \hat{x}_{k-1})^2]
\end{align*}
\]
Convolutional Network

[Diagram of convolutional network with layers and operations]
Convolutional Network

**Motivation**

Example: 1000x1000 image
1M hidden units

\[10^{12} \text{ parameters!!!}\]

Share the same parameters across different locations:
Convolutions with learned kernels

Example: 1000x1000 image
1M hidden units
Filter size: 10x10

100M parameters

Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10

10K parameters
Convolutional Network

Convolution

1D

2D

\[ y_{ij} = \frac{1}{4} \left( x_{2i,2j} + x_{2i+1,2j} + x_{2i,2j+1} + x_{2i+1,2j+1} \right) \]

Subsample

local averaging operator
Convolutional Network
Convolutional Network

Backpropagation gradient-descent procedure

Backpropagation algorithm for standard MLP

\[ E_p = \frac{1}{2} \| o_p - t_p \| ^2 = \frac{1}{2} \sum_{k=1}^{K} (o_{pk} - t_{pk})^2 \]

\[ w_{ji}^{(l)} \leftarrow w_{ji}^{(l)} + \Delta w_{ji}^{(l)} = w_{ji}^{(l)} - \lambda \frac{\partial E_p}{\partial w_{ji}^{(l)}} \]
Convolutional Neural Network - System

Images → Image Sampling → Dimensionality Reduction → Convolutional Neural Network → Classification

Convolution Neural Network

MLP Style Classifier

Nearest – Neighbor Classifier

Classification

Feature Extraction

Multi-Layer Perceptron

SOM

Dimensionality Reduction

K-L Transform

Images
Convolutional Neural Network – Extensions

LeNet-5

http://yann.lecun.com/exdb/lenet/

C1,C3,C5 : Convolutional layer.
5 × 5 Convolution matrix.
S2 , S4 : Subsampling layer.
Subsampling by factor 2.
F6 : Fully connected layer.

About 187,000 connection.
About 14,000 trainable weight
Convolutional Neural Network –
Extension and variants

Space Displacement Neural Networks (SDNN)

Shunting Inhibitory Convolutional Neural Networks (SICoNNet)

Siamese CNNs

Sparse Convolutional Neural Networks (Sparse CNN)
Convolutional Neural Network – Experiment & Comparison

200 training images and 200 test images from ORL database (AT&T).

### Various Experiments

- Variation of the number of output classes
- Variation of the dimensionality of the SOM
- Variation of the quantization level of the SOM
- Variation of the image sample extraction algorithm
- Substituting the SOM with the KL transform
- Replacing the CN with an MLP

<table>
<thead>
<tr>
<th>Number of classes</th>
<th>10</th>
<th>20</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error rate</td>
<td>1.33%</td>
<td>4.33%</td>
<td>5.75%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SOM Dimension</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error rate</td>
<td>8.25%</td>
<td>6.75%</td>
<td>5.75%</td>
<td>5.83%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SOM Size</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error rate</td>
<td>8.5%</td>
<td>5.75%</td>
<td>6.0%</td>
<td>5.75%</td>
<td>3.83%</td>
<td>3.83%</td>
<td>4.16%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input type</th>
<th>Pixel intensities</th>
<th>Differences w/base intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error rate</td>
<td>5.75%</td>
<td>7.17%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dimensionality reduction</th>
<th>Linear PCA</th>
<th>SOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error rate</td>
<td>5.33%</td>
<td>3.83%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear PCA</th>
<th>SOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>41.2%</td>
</tr>
<tr>
<td>CN</td>
<td>5.33%</td>
</tr>
</tbody>
</table>
Convolutional Neural Networks are a special kind of multi-layer neural networks.

Like almost every other neural networks they are trained with a version of the back-propagation algorithm.

Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing.

Shared weights: all neurons in a feature share the same weights.

In this way all neurons detect the same feature at different positions.

Reduce the number of free parameters in the input image.
Thank you