Statistical Machine Learning (BE4M33SSU)
Lecture 7b: Deep Neural Networks

Jan Drchal

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science
Overview

Topics covered in the lecture:

- Deep Architectures
- Convolutional Neural Networks (CNNs)
- Transfer learning
Is it better to use deep architectures rather than the shallow ones for complex nonlinear mappings?

We know that deep architectures evolved in Nature (e.g., cortex)

Universal approximation theorem: one layer is enough so why to bother with more layers?


- deep neural networks can have exponentially less units than shallow networks for learning the same function
- functions such as those realized by current deep convolutional neural networks are considered

Handcrafted features vs. automatic extraction

Gradually increasing complexity, intermediate representations: each successive layer brings higher abstraction
Fig. 2. Visualization of features in a fully trained model. For layers 2-5 we show the top activation signatures and corresponding feature maps across the validation data, projected down to pixel space using our deconvolutional network approach. Our reconstructions are not samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form. The compression artifacts are a consequence of the 30Mb submission limit, not the reconstruction algorithm itself.
Fig. 2. Visualization of features in a fully trained model. For layers 2-5 we show the top activations as a network of top 9 activations in the corresponding feature maps across the validation set, projected down to pixel space using our deconvolutional network approach. Our reconstructions are not samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g., eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form. The compression artifacts are a consequence of the 30Mb submission limit, not the reconstruction algorithm itself.
Fig. 2. Visualization of features in a fully trained model. For layers 2-5 we show the top 9 activations as a Neuromap and sub feat map across the validation data, projected down to pixel space using our deconvolutional network approach. Our reconstructions are not samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form. The compression artifacts are a consequence of the 30Mb submission limit, not the reconstruction algorithm itself.
Processing Images

- Input: grayscale image $32 \times 32$ pixels
- Output: layer of $32 \times 32$ features
- How many parameters do we need when input and output is fully connected?
Processing Images

- Input: grayscale image $32 \times 32$ pixels
- Output: layer of $32 \times 32$ features
- How many parameters do we need when input and output is fully connected?

$$\frac{32^2}{\text{outputs}} \times \left( \frac{32^2}{\text{inputs}} + 1 \right) \approx 1 \text{M}$$
Locally Connected Layer

- Motivation: topographical mapping in the visual cortex - nearby cells process nearby regions in the visual field
- Each neuron has a **receptive field** of $3 \times 3$ pixels
- It is fully connected only to the corresponding set of 9 inputs
- How many parameters do we need now?
Locally Connected Layer

- Motivation: topographical mapping in the visual cortex - nearby cells process nearby regions in the visual field
- Each neuron has a **receptive field** of $3 \times 3$ pixels
- It is fully connected only to the corresponding set of 9 inputs
- How many parameters do we need now?

$$30^2 \times (3^2 + 1) = 9k$$
Multiple Input Channels

- We can have more input channels, e.g., colors
- Now the input is defined by width, height, and depth: $32 \times 32 \times 3$
- The number of parameters is $30^2 \times (3 \times 3^2 + 1) \approx 25k$
Sharing Parameters

- We can further reduce the number of parameters by sharing weights.
- Use the same set of weights and bias for all outputs, define a *filter*.
- The number of parameters drops to $3 \times 3^2 + 1 = 28$.
- Translation *equivariance*.

![Diagram showing translation equivariance](image)
Multiple Output Channels

- Extract multiple different features
- Use multiple filters to get more feature maps
- For 4 filters we have $4 \times (3 \times 3^2 + 1) = 112$ parameters
- This is the convolutional layer
- Processes volume into volume
### Convolution Applied to an Image

<table>
<thead>
<tr>
<th>Identity</th>
<th><img src="https://en.wikipedia.org/wiki/Kernel_(image_processing)" alt="Identity Image" /></th>
</tr>
</thead>
</table>
|         | \[
|         | \begin{bmatrix} 0 & 0 & 0 \\
|         | 0 & 1 & 0 \\
|         | 0 & 0 & 0 \end{bmatrix} |
|         | ![Identity Output](https://en.wikipedia.org/wiki/Kernel_(image_processing)) |

<table>
<thead>
<tr>
<th>Sharpen</th>
<th><img src="https://en.wikipedia.org/wiki/Kernel_(image_processing)" alt="Sharpen Image" /></th>
</tr>
</thead>
</table>
|         | \[
|         | \begin{bmatrix} 0 & -1 & 0 \\
|         | -1 & 5 & -1 \\
|         | 0 & -1 & 0 \end{bmatrix} |
|         | ![Sharpen Output](https://en.wikipedia.org/wiki/Kernel_(image_processing)) |

<table>
<thead>
<tr>
<th>Box blur (normalized)</th>
<th><img src="https://en.wikipedia.org/wiki/Kernel_(image_processing)" alt="Box blur Image" /></th>
</tr>
</thead>
</table>
| \[
|         | \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\
|         | 1 & 1 & 1 \\
|         | 1 & 1 & 1 \end{bmatrix} |
|         | ![Box blur Output](https://en.wikipedia.org/wiki/Kernel_(image_processing)) |

<table>
<thead>
<tr>
<th>Gaussian blur (approximation)</th>
<th><img src="https://en.wikipedia.org/wiki/Kernel_(image_processing)" alt="Gaussian blur Image" /></th>
</tr>
</thead>
</table>
| \[
|         | \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\
|         | 2 & 4 & 2 \\
|         | 1 & 2 & 1 \end{bmatrix} |
|         | ![Gaussian blur Output](https://en.wikipedia.org/wiki/Kernel_(image_processing)) |

*References:*

- [Convolution Applied to an Image](https://en.wikipedia.org/wiki/Kernel_(image_processing))
Convolution in 2D: Forward Message

\[ z_{kld} = f_{kld}(\mathbf{x}, \mathbf{w}, \mathbf{b}) = b_d + \sum_{i=1}^{F} \sum_{j=1}^{F} \sum_{c=1}^{C} x_{k+i-1, l+j-1, c} w_{ijcd} \]
**Stride**

- Stride hyper parameter, typically $S \in \{1, 2\}$
- Higher stride produces smaller output volumes spatially
**Stride**

- Stride hyper parameter, typically $S \in \{1, 2\}$
- Higher stride produces smaller output volumes spatially
Zero Padding

- Convolutional layer reduces the spatial size of the output w.r.t. the input
- For many layers this might be a problem
- This is often fixed by *zero padding* the input
- The size of the zero padding is denoted $P$

$$P = 1, \ S = 1$$

![Diagram showing zero padding with $P = 1$ and $S = 1$]
Convolutional Layer Summary

- **Input volume:** $W_{\text{input}} \times H_{\text{input}} \times C$

- **Output volume:** $W_{\text{output}} \times H_{\text{output}} \times D$

- **Having** $D$ **filters:**
  - receptive field of $F \times F$ units,
  - stride $S$
  - zero padding $P$

$$
W_{\text{output}} = (W_{\text{input}} - F + 2P)/S + 1
$$
$$
H_{\text{output}} = (H_{\text{input}} - F + 2P)/S + 1
$$

- **Needs** $F^2CD$ **weights and** $D$ **biases**

- **The number of activations and $\delta$s to store:** $W_{\text{output}} \times H_{\text{output}} \times D$
Convolutional Layer: Nonlinearities

- In most cases a nonlinearity (sigmoid, tanh, ReLU) is applied to the outputs of the convolutional layer.
- Example: ReLU units
Max Pooling

- Reduces spatial resolution $\rightarrow$ less parameters $\rightarrow$ helps with overfitting
- Introduces translation invariance and invariance to small rotations
- Depth is not affected

$F = 2, S = 2$

\[
\begin{array}{cccccc}
2 & 2 & 0 & 4 & 3 & 4 \\
0 & 0 & 5 & 0 & 4 & 1 \\
4 & 5 & 2 & 5 & 1 & 4 \\
5 & 2 & 1 & 0 & 2 & 1 \\
2 & 3 & 3 & 3 & 5 & 3 \\
0 & 3 & 0 & 4 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{ccc}
2 & 5 & 4 \\
5 & 5 & 4 \\
3 & 4 & 5 \\
\end{array}
\]
Convolutional Neural Networks (CNNs)
VGGNet 2014

- Simonyan, Zisserman: *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014
- Lowering filter spatial resolution \((F = 3, S = 1, P = 1)\), increasing depth
- A sequence of \(3 \times 3\) filters can emulate a single large one
- Top five error 7.3%, 6.8% for an ensemble of 2 CNNs

<table>
<thead>
<tr>
<th>layer</th>
<th>activations</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>150k</td>
<td>1.7k</td>
</tr>
<tr>
<td>conv3-64</td>
<td>224 x 224 x 3</td>
<td>37k</td>
</tr>
<tr>
<td>conv3-128</td>
<td>112 x 112 x 64</td>
<td>74k</td>
</tr>
<tr>
<td>MP</td>
<td>112 x 112 x 64</td>
<td>147k</td>
</tr>
<tr>
<td>conv3-128</td>
<td>56 x 56 x 128</td>
<td>295k</td>
</tr>
<tr>
<td>conv3-256</td>
<td>28 x 28 x 256</td>
<td>590k</td>
</tr>
<tr>
<td>conv3-256</td>
<td>28 x 28 x 256</td>
<td>1.2M</td>
</tr>
<tr>
<td>conv3-512</td>
<td>14 x 14 x 512</td>
<td>2.4M</td>
</tr>
<tr>
<td>MP</td>
<td>14 x 14 x 512</td>
<td>2.4M</td>
</tr>
<tr>
<td>conv3-512</td>
<td>7 x 7 x 512</td>
<td>2.4M</td>
</tr>
<tr>
<td>MP</td>
<td>7 x 7 x 512</td>
<td>2.4M</td>
</tr>
<tr>
<td>FC - 4096</td>
<td>1x1x4096</td>
<td>103M</td>
</tr>
<tr>
<td>FC - 4096</td>
<td>1x1x4096</td>
<td>16.7M</td>
</tr>
<tr>
<td>FC - 1000</td>
<td>1x1x1000</td>
<td>4M</td>
</tr>
<tr>
<td>softmax</td>
<td></td>
<td>1000</td>
</tr>
</tbody>
</table>
Convolutional vs. Fully-Connected Layers

- Convolutional layer can be simply transformed to a Fully-connected layer → sparse weight matrix

- The other direction is also possible:
  FC layer of $D$ units following a $F \times F \times C$ convolutional layer can be replaced by a $1 \times 1 \times D$ convolutional layer using $F \times F$ filters ($P = 0$, $S = 1$)

- In both cases you do not have to recompute the weights, you just rearrange them
Fully-Connected Layer to Convolutional Example

input

CONV, MP layers

224 × 224 × 3

7 × 7 × 512

FC

4096

FC

4096

softmax

FC

1000
Fully-Connected Layer to Convolutional Example

input

CONV, MP layers

224 × 224 × 3

7 × 7 × 512

1 × 1 × 4096

1 × 1 × 1000

F = 7

F = 1

softmax
Fully-Connected Layer to Convolutional Example

input

CONV, MP layers

12 x 12 x 512

6 x 6 x 4096

6 x 6 x 1000

F = 7

F = 1

softmax
Transfer Learning

- Idea: use an existing model as a base to solve a *similar problem*
- Often used when not enough data available to solve the target problem directly
- Example: reuse an ImageNet network for object localization
Transfer Learning

- Idea: use an existing model as a base to solve a *similar problem*
- Often used when not enough data available to solve the target problem directly
- Example: reuse an ImageNet network for object localization
Transfer Learning

- Idea: use an existing model as a base to solve a *similar problem*
- Often used when not enough data available to solve the target problem directly
- Example: reuse an ImageNet network for object localization
Transfer Learning

- Idea: use an existing model as a base to solve a similar problem

- Often used when not enough data available to solve the target problem directly

- Example: reuse an ImageNet network for object localization

- You can:
  
  - cut the original network at various layers,
  
  - fix or not the weights of the original network or use different learning rates
  
  - use different type of model instead of the output layers, e.g., linear SVM
Fig. 2. Visualization of features in a fully trained model. For layers 2-5 we show the top 9 activation sites and subsystem feature maps across the validation data, projected down to pixel space using our deconvolutional network approach. Our reconstructions are not samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form. The compression artifacts are a consequence of the 30Mb submission limit, not the reconstruction algorithm itself.
Fig. 2. Visualization of features in a fully trained model. For layers 2-5 we show the top activations for the most active features in each map, projected down to pixel space using our deconvolutional network approach. Our reconstructions are not samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form. The compression artifacts are a consequence of the 30Mb submission limit, not the reconstruction algorithm itself.
Fig. 2. Visualization of features in a fully trained model. For layers 2-5 we show the top 9 active activations and maps to features across the validation data, projected down to pixel space using our deconvolutional network approach. Our reconstructions are not samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form. The compression artifacts are a consequence of the 30Mb submission limit, not the reconstruction algorithm itself.
| Identity | \[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\] |
|---|---|
| Edge detection | \[
\begin{bmatrix}
1 & 0 & -1 \\
0 & 0 & 0 \\
-1 & 0 & 1 \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0 \\
\end{bmatrix}
\] | \[
\begin{bmatrix}
-1 & -1 & -1 \\
-1 & 8 & -1 \\
-1 & -1 & -1 \\
\end{bmatrix}
\] |
| Sharpen          | \[
|                 | \begin{bmatrix}
|                 | 0 & -1 & 0 \\
|                 | -1 & 5 & -1 \\
|                 | 0 & -1 & 0 \\
|                 | \end{bmatrix} |
| Box blur        | \[
|                 | \frac{1}{9} \begin{bmatrix}
|                 | 1 & 1 & 1 \\
|                 | 1 & 1 & 1 \\
|                 | 1 & 1 & 1 \\
|                 | \end{bmatrix} |
| (normalized)    | |
| Gaussian blur   | \[
|                 | \frac{1}{16} \begin{bmatrix}
|                 | 1 & 2 & 1 \\
|                 | 2 & 4 & 2 \\
|                 | 1 & 2 & 1 \\
|                 | \end{bmatrix} |
| (approximation) | |
\[(i, j, c)\]
$S = 1$

$S = 2$
\[ S = 1 \]

\[ S = 2 \]
$P = 1, \ S = 1$
Non-Linearity

**Rectified linear function**

- **Applied per-pixel**
- output = max(0, input)

Input feature map

Black = negative; white = positive values

Only non-negative values

Output feature map
\[ F = 2, \quad S = 2 \]

\begin{array}{cccccc}
2 & 2 & 0 & 4 & 3 & 4 \\
0 & 0 & 5 & 0 & 4 & 1 \\
4 & 5 & 2 & 5 & 1 & 4 \\
5 & 2 & 1 & 0 & 2 & 1 \\
2 & 3 & 3 & 3 & 5 & 3 \\
0 & 3 & 0 & 4 & 0 & 1 \\
\end{array}
<table>
<thead>
<tr>
<th>Layer</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv3-64</td>
<td>224 x 224 x 3</td>
</tr>
<tr>
<td>conv3-64</td>
<td>112 x 112 x 64</td>
</tr>
<tr>
<td>MP</td>
<td>56 x 56 x 256</td>
</tr>
<tr>
<td>conv3-128</td>
<td>28 x 28 x 256</td>
</tr>
<tr>
<td>conv3-128</td>
<td>14 x 14 x 512</td>
</tr>
<tr>
<td>MP</td>
<td>7 x 7 x 512</td>
</tr>
<tr>
<td>conv3-256</td>
<td>1 x 1 x 4096</td>
</tr>
<tr>
<td>MP</td>
<td>1 x 1 x 1000</td>
</tr>
<tr>
<td>conv3-512</td>
<td>1 x 1 x 1000</td>
</tr>
<tr>
<td>FC</td>
<td>1 x 1 x 1000</td>
</tr>
<tr>
<td>FC</td>
<td>1 x 1 x 1000</td>
</tr>
<tr>
<td>softmax</td>
<td>1 x 1 x 1000</td>
</tr>
</tbody>
</table>

**Activations**

<table>
<thead>
<tr>
<th>Layer</th>
<th>1.7k</th>
<th>37k</th>
<th>295k</th>
<th>590k</th>
<th>200k</th>
<th>1.2M</th>
<th>2.4M</th>
<th>2.4M</th>
<th>2.4M</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv3-64</td>
<td>150k</td>
<td>3.2M</td>
<td>3.2M</td>
<td>800k</td>
<td>800k</td>
<td>800k</td>
<td>800k</td>
<td>800k</td>
<td>800k</td>
</tr>
<tr>
<td>conv3-128</td>
<td>1.6M</td>
<td>1.6M</td>
<td>400k</td>
<td>400k</td>
<td>400k</td>
<td>400k</td>
<td>400k</td>
<td>400k</td>
<td>400k</td>
</tr>
<tr>
<td>conv3-256</td>
<td>56k</td>
<td>56k</td>
<td>256k</td>
<td>256k</td>
<td>256k</td>
<td>256k</td>
<td>256k</td>
<td>256k</td>
<td>256k</td>
</tr>
<tr>
<td>conv3-512</td>
<td>100k</td>
<td>100k</td>
<td>100k</td>
<td>100k</td>
<td>100k</td>
<td>100k</td>
<td>100k</td>
<td>100k</td>
<td>100k</td>
</tr>
<tr>
<td>FC</td>
<td>103M</td>
<td>16.7M</td>
<td>4M</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
</tr>
</tbody>
</table>

**Parameters**

| Layer     | 1.7k | 37k | 74k  | 147k | 295k | 590k | 590k | 200k | 1.2M | 2.4M | 2.4M | 2.4M | 2.4M | 2.4M | 2.4M | 2.4M | 2.4M |
|-----------|------|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| conv3-64  | 150k | 3.2M| 3.2M | 800k | 800k | 800k | 800k | 800k | 800k | 800k | 800k | 800k | 800k | 800k | 800k | 800k | 800k | 800k |
| conv3-128 | 1.6M | 1.6M| 400k | 400k | 400k | 400k | 400k | 400k | 400k | 400k | 400k | 400k | 400k | 400k | 400k | 400k | 400k | 400k |
| conv3-256 | 56k  | 56k | 256k | 256k | 256k | 256k | 256k | 256k | 256k | 256k | 256k | 256k | 256k | 256k | 256k | 256k | 256k | 256k |
| conv3-512 | 100k | 100k| 100k | 100k | 100k | 100k | 100k | 100k | 100k | 100k | 100k | 100k | 100k | 100k | 100k | 100k | 100k | 100k |
| FC        | 103M | 16.7M| 4M   | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 |
The diagram illustrates a neural network architecture with the following layers:

- **Input Layer**: 224 x 224 x 3
- **CONV, MP Layers**: 7 x 7 x 512
- **Fully Connected (FC) Layers**: 4096
- **Softmax Layer**: 1000

The network starts with an input image of size 224 x 224 x 3, followed by convolutional and max pooling (MP) layers, then several fully connected (FC) layers, and finally a softmax layer for classification.
input
CONV, MP
layers

384 × 384 × 3

12 × 12 × 512

6 × 6 × 4096

6 × 6 × 4096

6 × 6 × 1000

softmax

F = 7

F = 1
input

CONV & MP

( x, y, w, h )

( x, y, w, h )