Neural Network Part 3: Convolutional Neural Networks

CS 760@UW-Madison
Goals for the lecture

you should understand the following concepts
  • convolutional neural networks (CNN)
  • convolution and its advantage
  • pooling and its advantage
Convolutional neural networks

• Strong empirical application performance

• Convolutional networks: neural networks that use convolution in place of general matrix multiplication in at least one of their layers

\[ h = \sigma(W^T x + b) \]

for a specific kind of weight matrix \( W \)
Convolution
Convolution: math formula

• Given functions $u(t)$ and $w(t)$, their convolution is a function $s(t)$

$$s(t) = \int u(a)w(t - a)da$$

• Written as

$$s = (u * w) \quad \text{or} \quad s(t) = (u * w)(t)$$
Convolucion: discrete version

- Given array \( u_t \) and \( w_t \), their convolution is a function \( s_t \)

\[
s_t = \sum_{a=-\infty}^{+\infty} u_a w_{t-a}
\]

- Written as

\[
s = (u \ast w) \quad \text{or} \quad s_t = (u \ast w)_t
\]

- When \( u_t \) or \( w_t \) is not defined, assumed to be 0
Illustration 1

\[ w = [z, y, x] \]
\[ u = [a, b, c, d, e, f] \]
Illustration 1

xc+yd+ze
Illustration 1

![Diagram with variables and equation: xd + ye + zf]
Illustration 1: boundary case

\[ xe+yf \]
Illustration 1 as matrix multiplication

<table>
<thead>
<tr>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>f</td>
</tr>
</tbody>
</table>
Illustration 2: two dimensional case

\[ wa + bx + ey + fz \]
Illustration 2

\[
\begin{array}{cccc}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
\end{array}
\]

\[
\begin{array}{cc}
  w & x \\
  y & z \\
\end{array}
\]

\[
\begin{align*}
  &wa + bx \\
  + &ey + fz \\
\end{align*}
\]

\[
\begin{align*}
  &bw + cx \\
  + &fy + gz \\
\end{align*}
\]
• All the units used the same set of weights (kernel)
• The units detect the same “feature” but at different locations
Advantage: sparse interaction

Fully connected layer, $m \times n$ edges

$m$ output nodes

$n$ input nodes

Figure from *Deep Learning*, by Goodfellow, Bengio, and Courville
Advantage: sparse interaction

Convolutional layer, \( \leq m \times k \) edges

- \( m \) output nodes
- \( k \) kernel size
- \( n \) input nodes

Figure from *Deep Learning*, by Goodfellow, Bengio, and Courville
Advantage: sparse interaction

Multiple convolutional layers: larger receptive field

Figure from *Deep Learning*, by Goodfellow, Bengio, and Courville
Advantage: parameter sharing/weight tying

The same kernel are used repeatedly. E.g., the black edge is the same weight in the kernel.

Figure from *Deep Learning*, by Goodfellow, Bengio, and Courville
Advantage: equivariant representations

- Equivariant: transforming the input = transforming the output
- Example: input is an image, transformation is shifting
  \[
  \text{Convolution(shift(input))} = \text{shift(Convolution(input))}
  \]
- Useful when care only about the \textit{existence} of a pattern, rather than the \textit{location}
Pooling
Terminology

Figure from Deep Learning, by Goodfellow, Bengio, and Courville
Pooling

- Summarizing the input (i.e., output the max of the input)

Figure from *Deep Learning*, by Goodfellow, Bengio, and Courville
• Each unit in a pooling layer outputs a max, or similar function, of a subset of the units in the previous layer.
Advantage

Induce invariance

Figure from *Deep Learning*, by Goodfellow, Bengio, and Courville
Motivation from neuroscience

• David Hubel and Torsten Wiesel studied early visual system in human brain (V1 or primary visual cortex), and won Nobel prize for this

• V1 properties
  • 2D spatial arrangement
  • Simple cells: inspire convolution layers
  • Complex cells: inspire pooling layers
Example: LeNet
LeNet-5


- Apply convolution on 2D images (MNIST) and use backpropagation

- Structure: 2 convolutional layers (with pooling) + 3 fully connected layers
  - Input size: 32x32x1
  - Convolution kernel size: 5x5
  - Pooling: 2x2
Figure from *Gradient-based learning applied to document recognition*, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner
LeNet-5

Figure from *Gradient-based learning applied to document recognition*, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner
LeNet-5

Figure from *Gradient-based learning applied to document recognition*, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner

Filter: 5x5, stride: 1x1, #filters: 6
LeNet-5

Figure from *Gradient-based learning applied to document recognition*, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner
LeNet-5

Figure from Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner

Filter: 5x5x6, stride: 1x1, #filters: 16
LeNet-5

Figure from Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner
LeNet-5

Weight matrix: 400x120

Figure from Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner
LeNet-5

- Weight matrix: 84x10
- Weight matrix: 120x84

Figure from *Gradient-based learning applied to document recognition*, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner
Example: ResNet
ResNet


- Apply very deep networks with repeated residue blocks

- Structure: simply stacking residue blocks
Plain Network

• “Overly deep” plain nets have higher training error
• A general phenomenon, observed in many datasets

Residual Network

• Naïve solution
  • If extra layers are an identity mapping, then a training error does not increase

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun.
Residual Network

• Deeper networks also maintain the tendency of results
  • Features in same level will be almost same
  • An amount of changes is fixed
  • Adding layers makes smaller differences
  • Optimal mappings are closer to an identity

Residual Network

- Plain block
  - Difficult to make identity mapping because of multiple non-linear layers

Residual Network

- Residual block
  - If identity were optimal, easy to set weights as 0
  - If optimal mapping is closer to identity, easier to find small fluctuations

-> Appropriate for treating perturbation as keeping a base information

Network Design

- Basic design (VGG-style)
  - All 3x3 conv (almost)
  - Spatial size/2 => #filters x2
  - Batch normalization
  - Simple design, just deep

- Other remarks
  - No max pooling (almost)
  - No hidden fc
  - No dropout

Results

• Deep Resnets can be trained without difficulties
• Deeper ResNets have lower training error, and also lower test error

Results

• 1\textsuperscript{st} places in all five main tracks in “ILSVRC & COCO 2015 Competitions”
  • ImageNet Classification
  • ImageNet Detection
  • ImageNet Localization
  • COCO Detection
  • COCO Segmentation

Quantitative Results

• ImageNet Classification

Qualitative Result

- Object detection
  - Faster R-CNN + ResNet

Qualitative Results

- Instance Segmentation

THANK YOU

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Matt Gormley, Elad Hazan, Tom Dietterich, and Pedro Domingos.