Convolutional Neural Networks

- appropriate for computer vision tasks (where they were originally introduced, but they are useful in other domains as well)

- computer vision tasks:
  - classification
  - segmentation
  - bounding box determination

- key to good performance: choice of our feature map

\[ \phi : \mathbb{R}^{d_1 \times d_2} \rightarrow \mathbb{R}^D \]

- so far in this class:
  - flattened images into vectors
  - kernel approach (implicitly specify \( \phi \))
  - autoencoders to get \( \phi \)
new approach: convolutional filters
treat our image directly as a two-dimensional table of numbers and compute local features:

```
| 1 1 1 0 |
| 0 1 1 1 |
| 0 0 1 1 |
| 0 1 1 0 |
```

* filter/kernel

```
| 1 0 1 |
| 0 1 0 |
| 1 0 1 |
```

```
| 4 3 4 |
| 2 4 3 |
| 2 3 4 |
```
Note that the filters have size $k_1 \times k_2$ and $k_1$ & $k_2$ must be odd.

\[
\begin{array}{c}
\text{I - image} \\
\text{K - kernel}
\end{array}
\]

\[
(I \ast K)_{i,j} = \sum_{m=-\frac{k_1}{2}}^{\frac{k_1}{2}} \sum_{n=-\frac{k_2}{2}}^{\frac{k_2}{2}} I_{i-m, j-n} \cdot K_{mn}
\]
Idea of CNNs: use convolutional filters to build our nonlinear feature map

Why?
- Intuition that image features should be local
- Turns out to be cheap compared to a fully connected layers
- Parameter sharing which makes learning better-behaved
Convolutional Layer:

Input image

$K$ - 3x3 filter

$I \ast K$ - the preactivation

$\sigma(I \ast K)$ - output from the convolutional layer

$K$ is the parameter for this convolutional layer
**Convolutional Layer**

*Input:* \(d_1 \times d_2\)

*Filter:* \(k_1 \times k_2\)

*Output:* \((d_1 - k_1 + 1) \times (d_2 - k_2 + 1)\)
Convolutional Layers w/ Stride

Input: \( d_1 \times d_2 \)
Kernel: \( K_1 \times K_2 \)
Stride: \( S_1 \times S_2 \)

- evaluate the filter at \( x \)-intervals of \( S_1 \)
  and \( y \)-intervals of \( S_2 \)
- equivalent to taking \( I \times K \) and sampling
  in the \( x \) dimension each \( S_1 \) pixels and
  in the \( y \) dimension each \( S_2 \) pixels
$I - 5 \times 5$

$K - 3 \times 3$

$I \times K - 3 \times 3$

$I \times K$ stride of $(2,2) - 2 \times 2$
CNN diagram

Image

Conv, Conv, Flatten, FC, FC
Pooling reduces the dimensionality of a convolutional layer by aggregating locally.

Typical examples:
- max pooling
- average pooling

2x2 max pooling

Stride: 2x2
Effective Receptive Fields

- Effective Receptive Field for a feature is the set of pixels on which it depends, in the original image.
**Ex convolutional NN: LeNet (1997)**

5-layer CNN

<table>
<thead>
<tr>
<th>Layer</th>
<th>Features</th>
<th>Filter Size</th>
<th>Stride</th>
<th>Activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Conv2D</td>
<td>6</td>
<td>5x5</td>
<td>1</td>
<td>tanh</td>
</tr>
<tr>
<td>2 Avg Pooling</td>
<td>6</td>
<td>2x2</td>
<td>2</td>
<td>tanh</td>
</tr>
<tr>
<td>3 Conv2D</td>
<td>16</td>
<td>5x5</td>
<td>1</td>
<td>tanh</td>
</tr>
<tr>
<td>4 Avg Pooling</td>
<td>16</td>
<td>2x2</td>
<td>2</td>
<td>tanh</td>
</tr>
<tr>
<td>5 Conv</td>
<td>120</td>
<td></td>
<td>1</td>
<td>softmax</td>
</tr>
<tr>
<td>6 FC</td>
<td>84</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output FC</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>