Convolutional neural network
(CNN, Conv. Net)

- Each square is a pixel
- Pixel is local intensity

2D image (100x100)

Start from 1D

\[
\begin{array}{cccccc}
1 & 2 & \ldots & k & \ldots & d \\
\hline
1 & 1 & \ldots & k & \ldots & d \\
0 & 1 & \ldots & k & \ldots & d \\
\end{array}
\]

Output y
Input x

2 adjacent layers of network

Both x and y are 1D maps

k: Index position of map

\[ y_k = w_{-1} x_{k-1} + w_0 x_k + w_1 x_{k+1} \]

For all \( k = 1, 2, \ldots, d \)

Kernel, filter \( \rightarrow \) 3 linear parameters

\[
\begin{bmatrix}
w_{-1} & w_0 & w_1 \\
\end{bmatrix}
\]

To be learned

Shared over all \( k = 1, \ldots, d \)

Sliding window, translation invariant

Zero padding at boundary

To be followed by adding bias, rectification

Bigger filter

\[
\begin{bmatrix}
w_3 & w_2 & w_1 & w_0 \end{bmatrix}
\]

Linear transformation can happen within any adjacent layers

\[
y_k = w_3 x_{k-3} + w_2 x_{k-2} + w_1 x_{k-1} + w_0 x_k + w_1 x_{k+1} + w_2 x_{k+2} + w_3 x_{k+3}
\]
Multiple channels

\[ Y_k = \mathbf{w}_0 \mathbf{x}_k + \mathbf{w}_1 \mathbf{x}_{k+1} \]

Think of \( Y_k \) as a vector.

Think of \( \mathbf{x}_k \) as a vector.

3D

\[ \mathbf{w} \]

is a 3x2 matrix.

Composition of 3 thought vectors in layer below:

- Thought vector \( \mathbf{y}_{k-2} \)
- Composition of 3 thought vectors in layer below:
  - 3x1
  - 3x2
  - 2x1

\[ \mathbf{w}_0 \mathbf{x}_k \]

\[ \mathbf{w}_1 \mathbf{x}_{k+1} \]

\[ \mathbf{y}_k \]

\[ \mathbf{w}_{-1} \mathbf{x}_{k-1} \]

Computed by combining all the channels below:

\[ + \text{bias} \]

Rectification (applied elementwise)

Back to 2D:

Need to generalize formula & sliding window to 2D:

One channel

Kernel matrix

3x3

Sliding window

\[ \mathbf{y}_{ij} = \mathbf{w}_{-1,0} \mathbf{x}_{i-1,j} + \mathbf{w}_{-1,1} \mathbf{x}_{i-1,j+1} + \mathbf{w}_{1,0} \mathbf{x}_{i,j-1} + \mathbf{w}_{1,1} \mathbf{x}_{i,j+1} \]
+ bias, rectification

can make this bigger (i.e. 5x5, 7x7...)

Now generalize multiple channels to 2D

a channel, feature map

channels 1 2 \ldots 10

1 2 \ldots 20

\[
Y_{ij} = \frac{1}{\Delta i = -1 \Delta j = -1} \sum \sum w_{oi,oj} X_{i+oi, j+oj}
\]

each value = a channel

each \text{vec} = 10\text{-dim}

\text{spatial range} = 3 \times 3

20\text{-dim}

thought vector

1 \times 1 \text{ conv., no spatial pooling}

\[
Y_{ij} = w_{0,0} X_{ij}
\]

20\text{-dim} 20 \times 10 10\text{-dim}

+ bias

+ ReLU

1 \times 1 \text{ conv., no spatial pooling}

\[
Y_{ij} = w_{0,0} X_{ij}
\]

20\text{-dim} 20 \times 10 10\text{-dim}

+ bias

+ ReLU
network within network

Subsampling
- makes map smaller, computation faster
  - don’t need to calculate redundant observations

4x4 reduced to 2x2

stride = 2
  OR max pooling (find maximum win each block)
  OR 2x2 filter (may be better than direct subsampling)

edges

conv

RGB

100x100

100x100

12

12

128

50x50

50x50

128

25x25

class prob / soft max

1000 logits

fully connected

4096

1x1

12
image net 1000 categories 1 million images

thought vectors

load-d

→ 128-d

→ 512-d

→ ...

spatial composition

\[ y_{ij} = \sum_{\delta i} \sum_{\delta j} w_{\delta i \delta j} x_{i+\delta i, j+\delta j} \]