Convolution Neural Networks (CNN)

- Introduced by Yann LeCun in 1989
- Convolution

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-t)g(t)\,dt$$

$$(f * g)(x) = \sum_{t=-\infty}^{\infty} f(x-t)g(t)$$

- In Math 3160, when X and Y are independent,

$$f_{X+Y} = f_X * f_Y.$$

- Convolution is translation-invariant, i.e., when $\tau_a f(x) = f(x-a)$,

$$\tau_a(f * g) = \tau_a(f) * g = f * \tau_a(g).$$

- In Machine Learning, the convolution of 2-D arrays is defined by

$$\text{input: } X = [X_{i,j}] \qquad \text{kernel (or filter): } \boldsymbol{w} = [w_{i,j}]$$

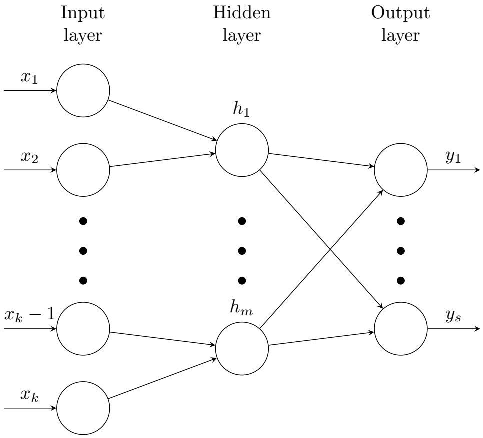
$$\text{output:} \quad (X * \boldsymbol{w})_{i,j} = \sum_{c,d} X_{i+c,j+d} w_{c,d}$$

- For example,

$$\begin{array}{|c|c|c|c|}\hline 1 & 2 & 0 & 3 \\ \hline 0 & 2 & 1 & 2 \\ \hline 1 & 0 & 0 & 2 \\ \hline 3 & 0 & 2 & 0 \\ \hline \end{array} * \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 1 & 2 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 5 & 6 & 5 \\ \hline 1 & 2 & 5 \\ \hline 4 & 4 & 2 \\ \hline \end{array}$$

- Clearly, this operation is translation-invariant.

- Convolution brings about parameter sharing and sparse connectivity.



- After convolution, pooling is performed. Two typical methods are the max pooling and the average pooling.
- Pooling is approximately translation-invariant and helps reduce noise.

  For example,

1	2	0
3	1	0
0	0	0

0	0	0
0	1	2
0	3	1

  Notice that the maximum or the average does not change.
- The result of convolution and pooling will go through activation and then move forward to the next layer.

Ideas of CNN

- Sparse connectivity

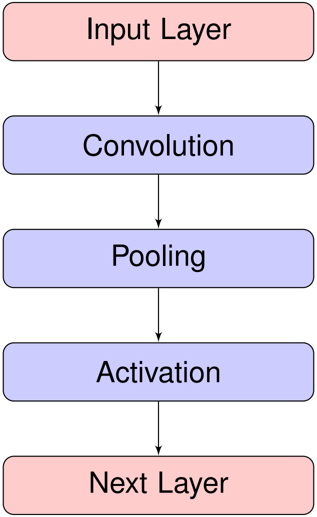
  Nearby pixels on an image are more strongly correlated than more distant pixels. Recognizing local features can enhance the performance of a classifier.

- Parameter sharing

  Any useful features that are detected in some portion of the input may be valid in other parts. The weights are shared across the layer, and the number of parameters is reduced.

- Translation invariance

  Features of images are invariant under translation.


Input Layer
Convolution
Pooling
Activation
Next Layer

Some specifics

- We use the max pooling and the relu function for activation.
- Multiple for-loops make computations very slow.

  We need to vectorize our computations.

  For a convolution with a 2×2 kernel, we convert

$$
\begin{array}{|c|c|c|c|}
\hline
x_{00} & x_{01} & x_{02} & x_{03} \\
\hline
x_{10} & x_{11} & x_{12} & x_{13} \\
\hline
x_{20} & x_{21} & x_{22} & x_{23} \\
\hline
x_{30} & x_{31} & x_{32} & x_{33} \\
\hline
\end{array}
\longrightarrow
\begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
x_{00} & x_{01} & x_{02} & x_{10} & x_{11} & x_{12} & x_{20} & x_{21} & x_{22} \\
\hline
x_{01} & x_{02} & x_{03} & x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} \\
\hline
x_{10} & x_{11} & x_{12} & x_{20} & x_{21} & x_{22} & x_{30} & x_{31} & x_{32} \\
\hline
x_{11} & x_{13} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} \\
\hline
\end{array}
$$

  and use a matrix multiplication once.
- We need to record where the maxima occur in the max pooling.
- We may use multiple kernels (or filters).