Convolutional Neural Networks

Nan Ye

School of Mathematics and Physics
The University of Queensland

Applications

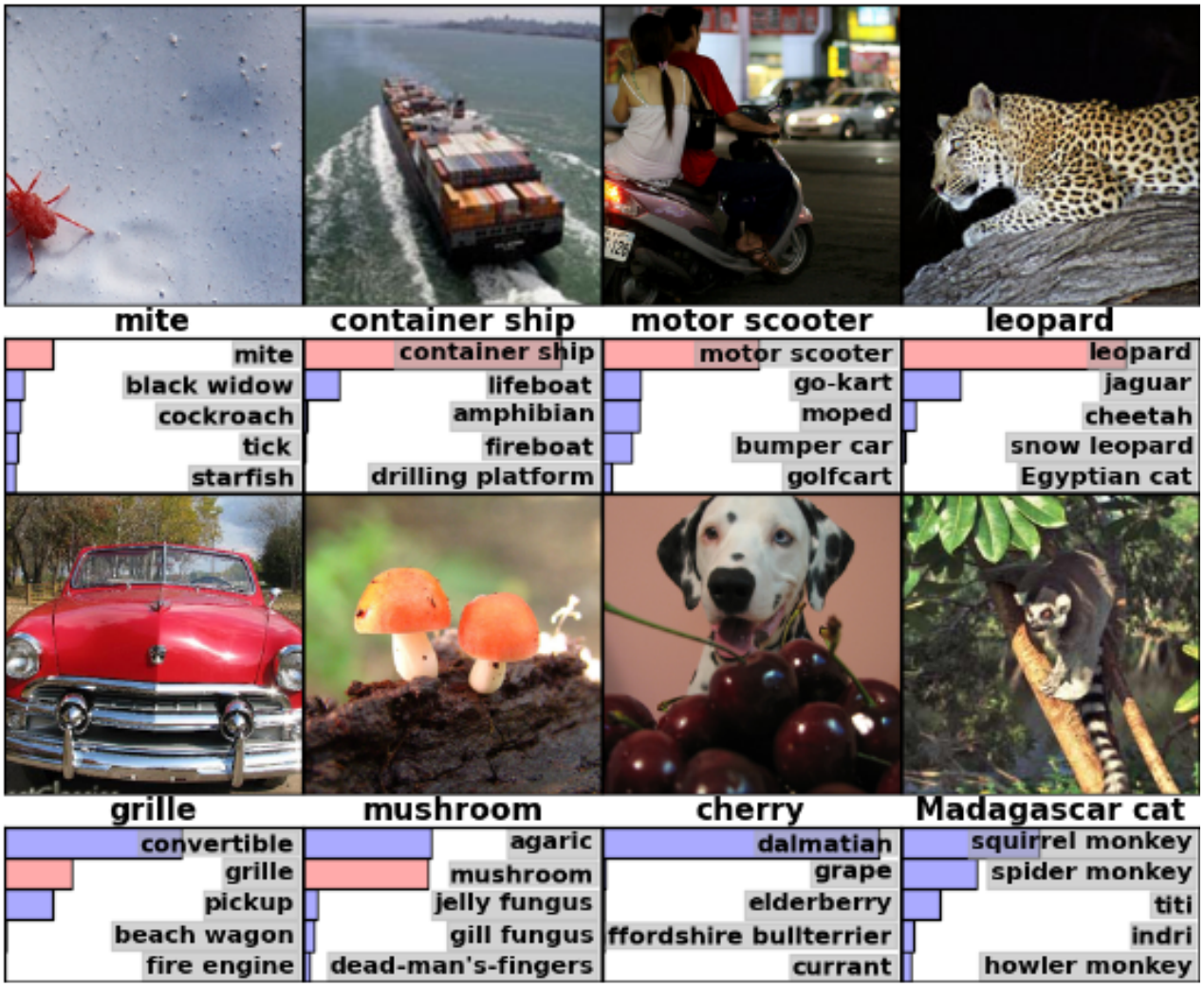
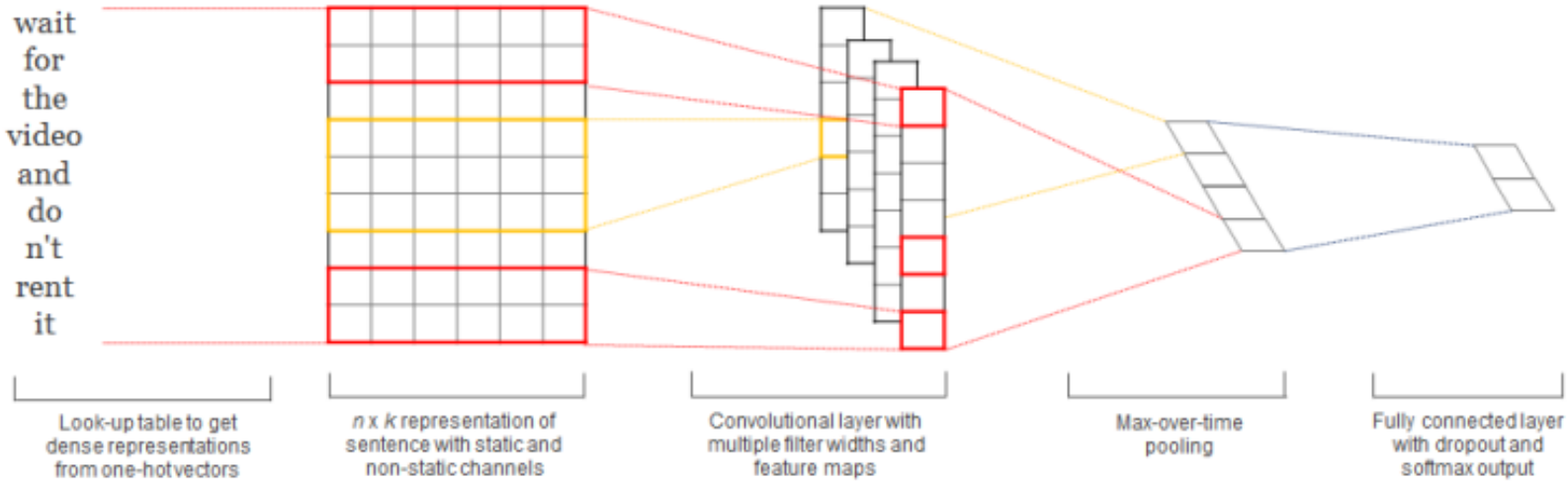


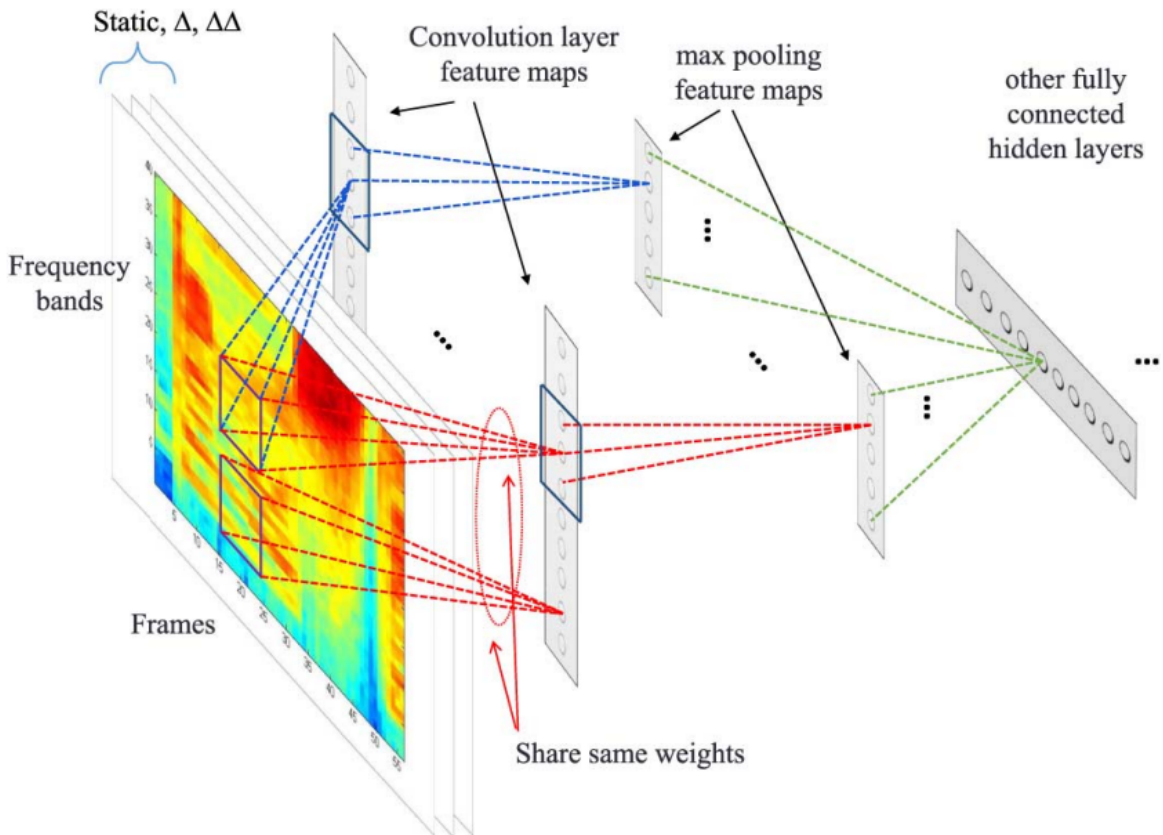
Image classification

Krizhevsky, Sutskever, and Hinton, Imagenet classification with deep convolutional neural networks, 2012



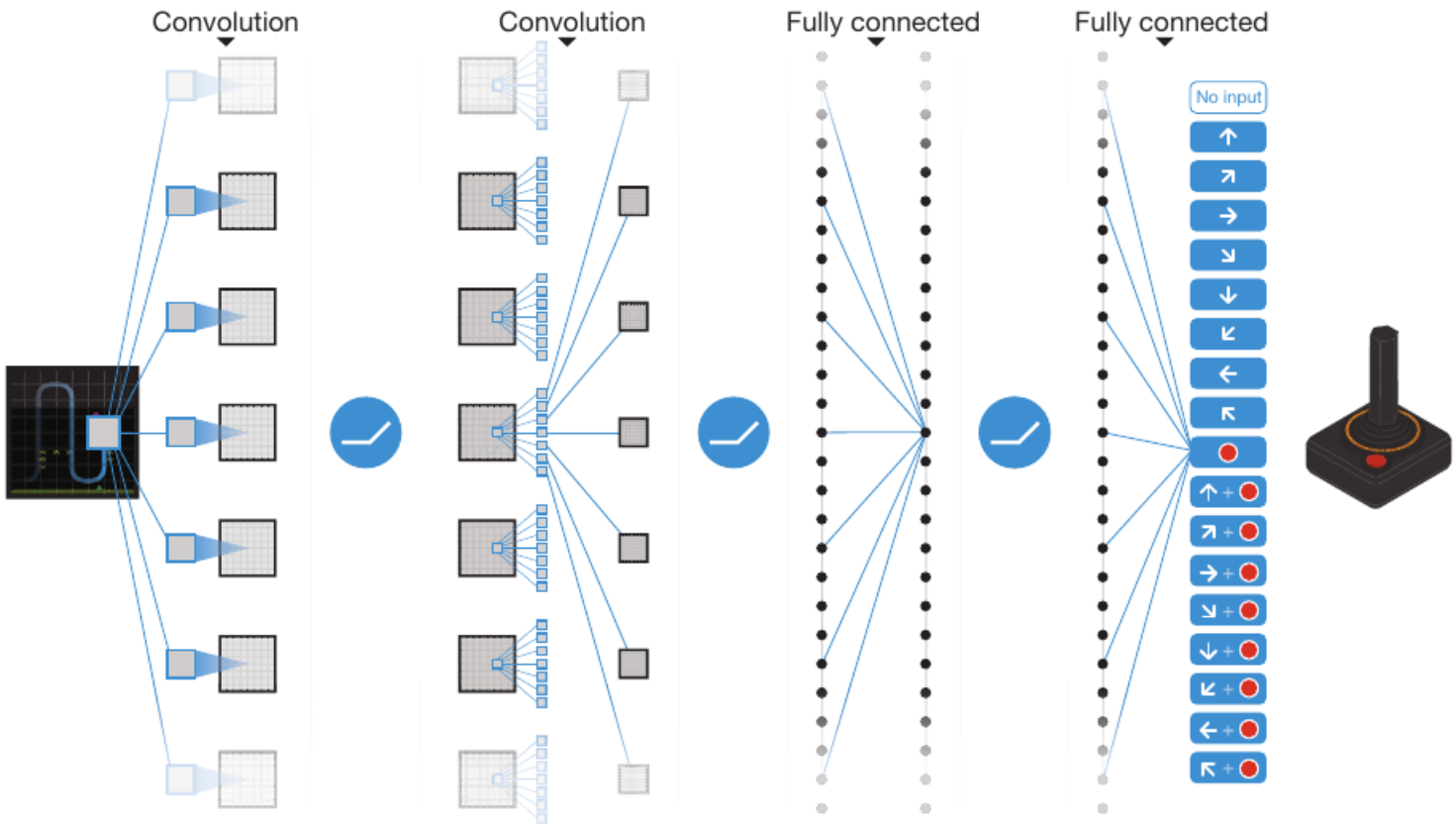
Natural language processing

Kim, Convolutional neural networks for sentence classification, 2014



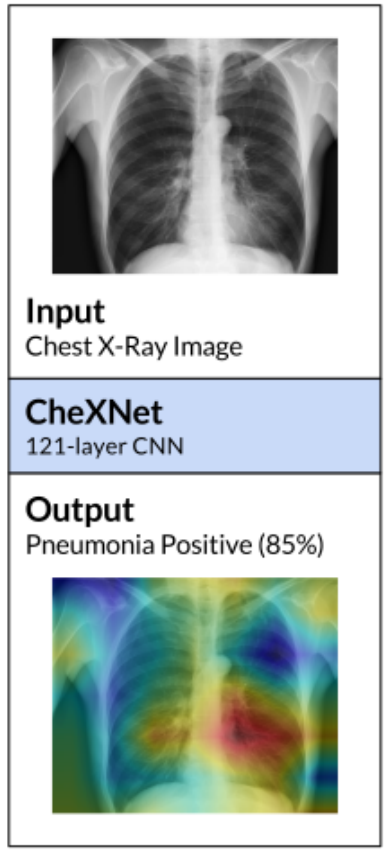
Speech recognition

Abdel-Hamid, Mohamed, Jiang, Deng, Penn, and Yu, Convolutional neural networks for speech recognition, 2014



Playing video games

Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland, and Ostrovski, Human-level control through deep reinforcement learning, 2015



Pneumonia detection from chest X-rays

Rajpurkar, Irvin, Zhu, Yang, Mehta, Duan, Ding, Bagul, Langlotz, and Shpanskaya, CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning, 2017

History

Biological vision

- Hubel & Wiesel (1950s and 1960s) showed that cat and monkey visual cortexes contain neurons that individually respond to small regions of the visual field.
- The firing of a single neuron is affected by a certain region of the visual space, known as the the receptive field of the neuron.
- Neighboring cells have similar and overlapping receptive fields.
- Some cells can detect edges irrespective of where they occur.

Neocognitron

- Fukushima (1980) introduced the neocognitron as an artificial neural net based on Hubel & Wiesel's works on biological vision.
- He introduced two basic types of layers in CNNs (convolutional layers, and downsampling layers) that makes the network able to tolerate some degree of input distortion

LeNet

- LeCun et. al. (1989) proposed a convolutional neural network that extends the neocognitron architecture.
- They developed a fully automatic algorithm for learning the network.

Modern CNNs

- Since 2010s, larger and deeper CNNs have been developed for various tasks, including the examples that we see earlier.
- The inputs for such tasks can be seen as arrays where nearby values are correlated.

Convolutional Neural Nets (CNNs)

- CNNs are multilayer feedforward neural networks
  - they are MLPs where the weights have been constrained to mimic how biological vision works
- Three architectural ideas
  - Local receptive fields
  - Shared weights
  - Spatial or temporal sub-sampling

  These ensure some degree of shift, scale, and distortion invariance.

- There are two key building blocks
  - The convolutional layer, which consists of a number of filters
    - filters are also called kernels, feature detectors
    - each filter scans small patches in the input to detect features
  - The downsampling layer, which reduces the resolution of the image for learning higher-level features.

Convolution

Convolution (in CNN) is not convolution (in maths)!

- In maths, the convolution of two discrete functions f and g defined on the set of integers is denoted by $f * g$ and defined as

$$(f * g)(n) = \sum_{i=-\infty}^{\infty} f(i)g(n-i)$$

- $(f * g)(n)$ is the inner product of the following two infinite vectors

$$\begin{array}{lcccc} \mathbf{f} = \ldots & f(-1) & f(0) & f(1) & \ldots \\ \bar{\mathbf{g}}_n = \ldots & g(n+1) & g(n) & g(n-1) & \ldots \end{array}$$

  $\bar{\mathbf{g}}_n$ is obtained by reversing $\mathbf{g} = (\ldots, g(-1), g(0), g(1), \ldots)$.
- Convolution slides a reversed $\mathbf{g}$ to obtain inner products with $\mathbf{f}$.

- In maths, there is also a concept called cross correlation

$$(f \star g)(n) = \sum_{i=-\infty}^{\infty} f(i)g(n+i)$$

- Cross correlation slides $\mathbf{g}$ to obtain inner products with $\mathbf{f}$.
- Cross correlation is aka sliding inner product (or sliding dot product).
  - Thus convolution is a kind of reversed sliding inner product.
- Convolution in CNN is actually cross correlation (for finite vectors/matrices/arrays)!

2D Convolution (in CNN)

- Given an $N \times N$ input, the convolution operation slides one filter through the input to extract features
  - An $F \times F$ filter is simply an $F \times F$ weight matrix.
  - We slide the filter over all $F \times F$ subarrays.
  - For each subarray, we compute the weighted sum of its elements (i.e., the dot product between the filter and the sub-array).
  - This gives us an $(N - F + 1) \times (N - F + 1)$ feature/activation map.

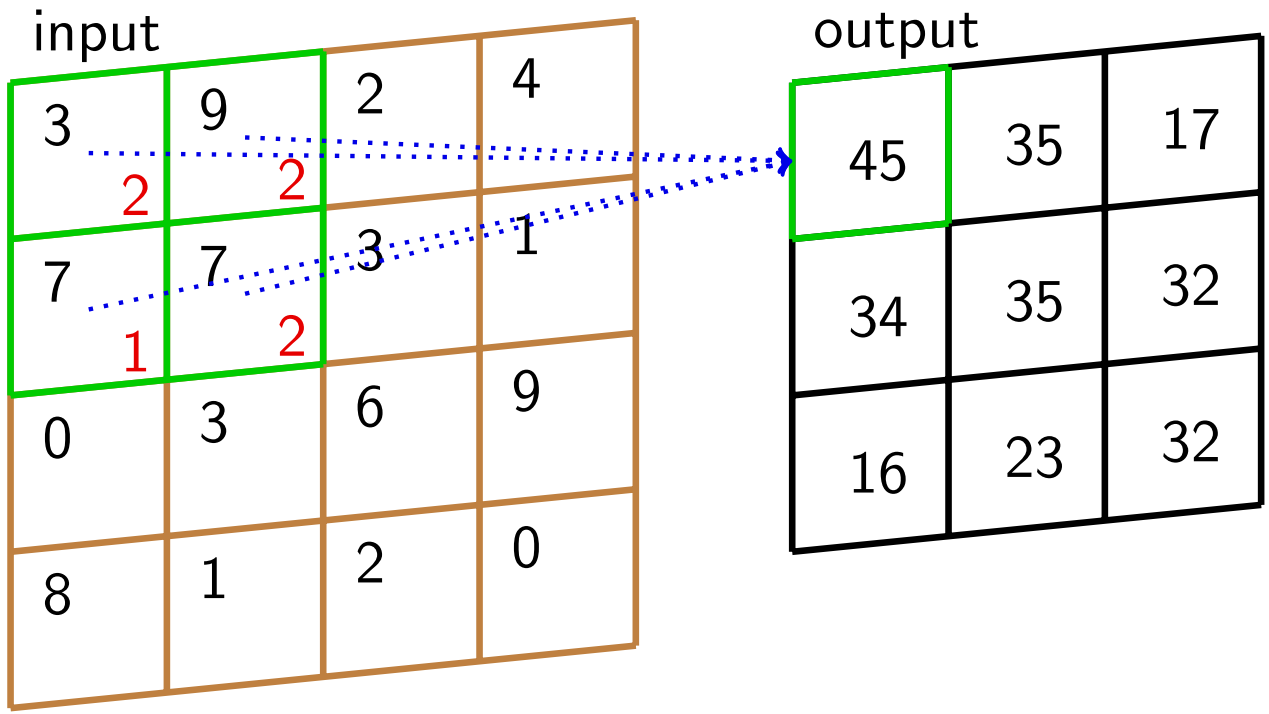
Example. 2x2 filter applied to 4x4 input

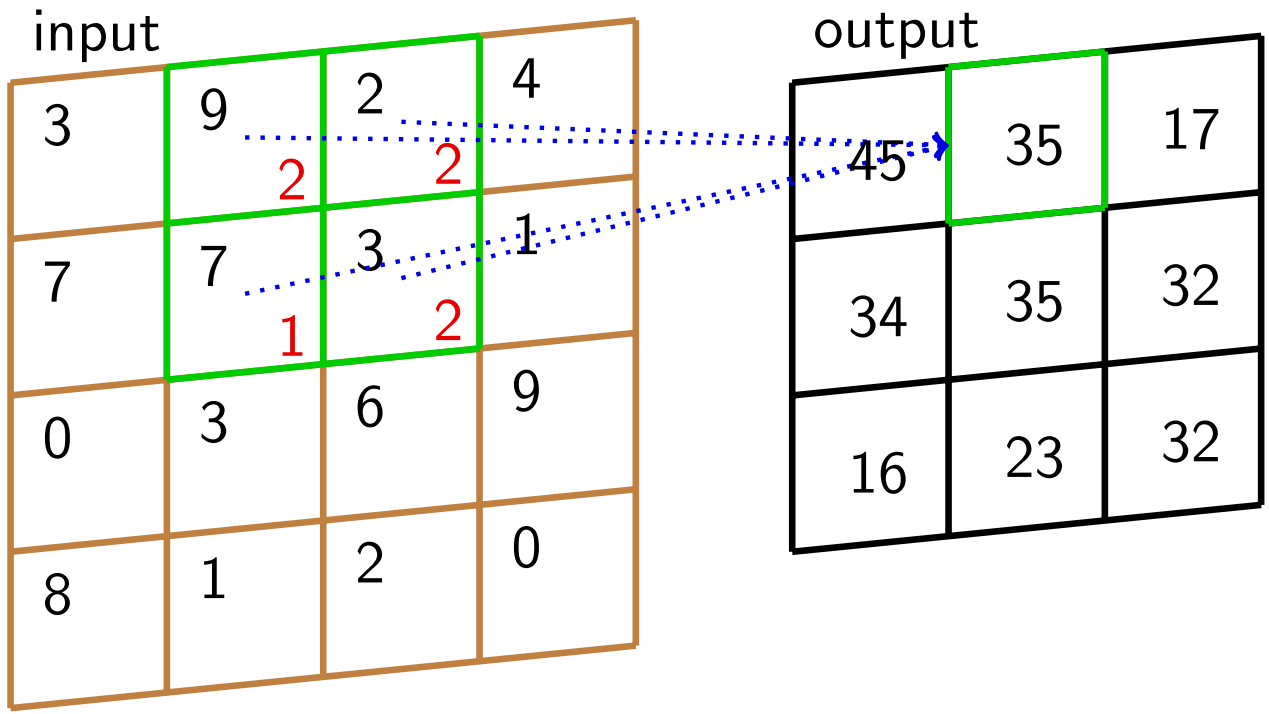
input

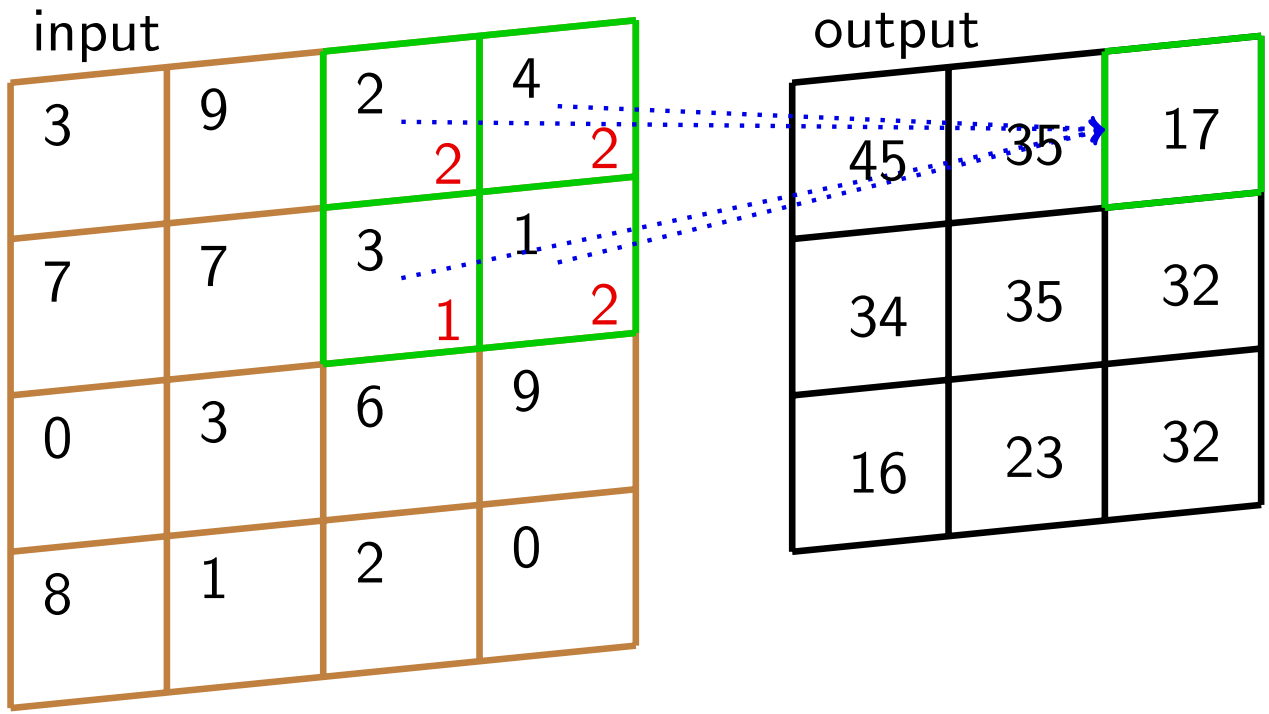
3	9	2	4
7	7	3	1
0	3	6	9
8	1	2	0

filter

2	2
1	2


input
3
9
2
4
2
2
7
7
3
1
1
2
0
3
6
9
8
1
2
0
output
45
35
17
34
35
32
16
23
32


input
3
9
2
4
2
2
7
7
3
1
1
2
0
3
6
9
8
1
2
0
output
45
35
17
34
35
32
16
23
32


input
3
9
2
4
2
2
7
7
3
1
1
2
0
3
6
9
8
1
2
0
output
45
35
17
34
35
32
16
23
32

In the language of neural nets...

- 4x4 input matrix = outputs of 4x4 input neurons
- 3x3 output matrix = outputs of 3x3 neurons in the conv. layer
- Each output neuron is connected to 4 of the 4x4 input neurons.
- The 4 weights are shared for all the output neurons.

input

3	9	2	4
7	7	3	1
0	3	6	9
8	1	2	0

output

45	35	17
34	35	32
16	23	32

Example. 2x2 filter applied to 5x5 input with stride 2

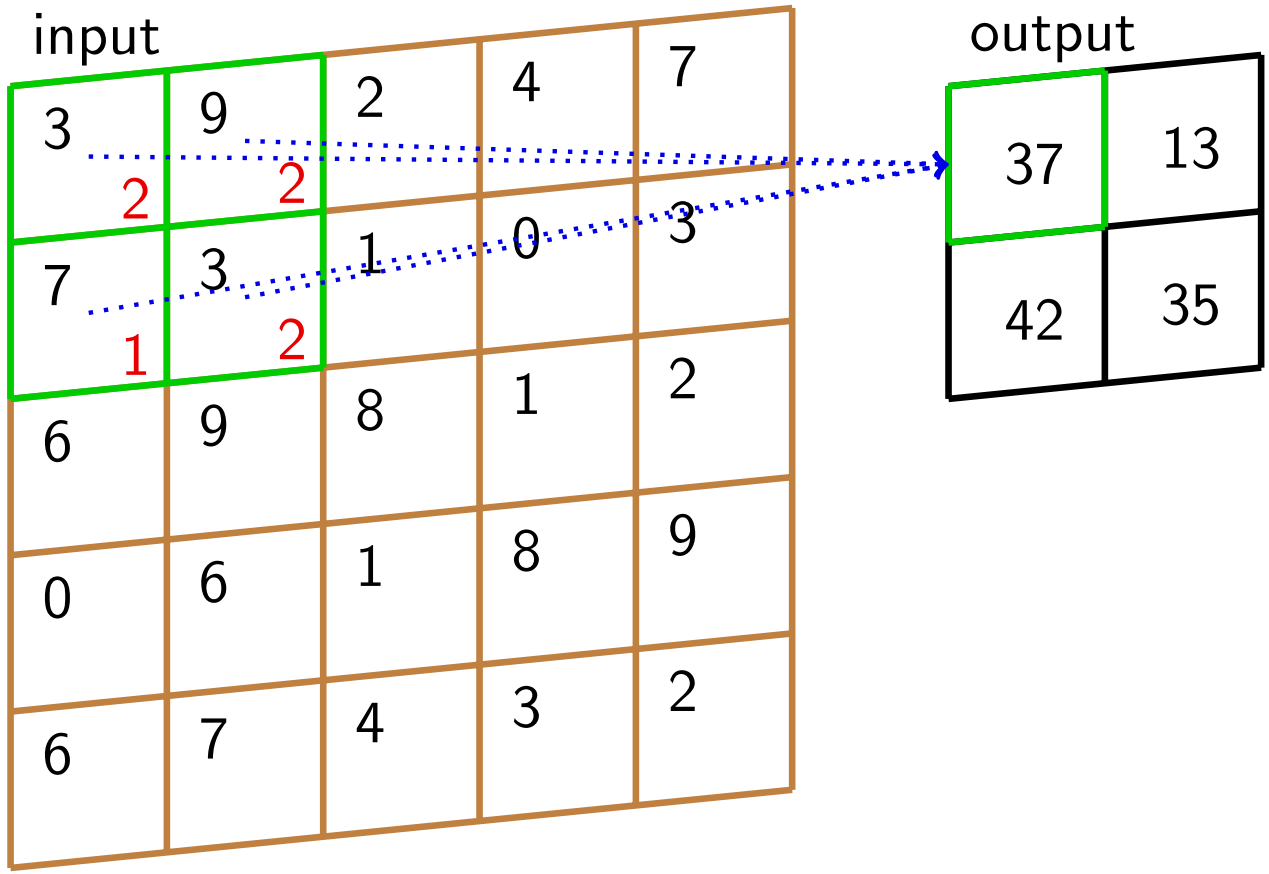
input

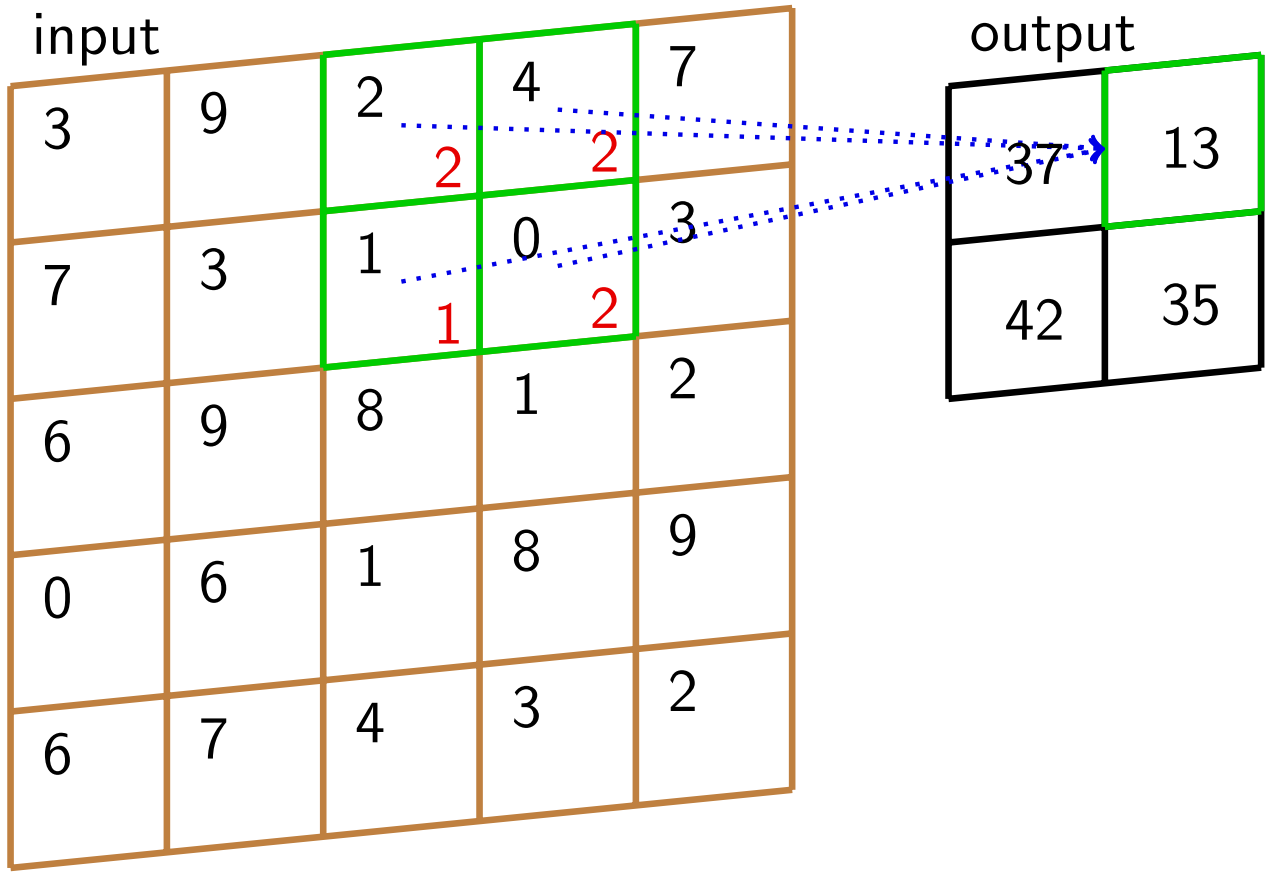
3	9	2	4	7
7	3	1	0	3
6	9	8	1	2
0	6	1	8	9
6	7	4	3	2

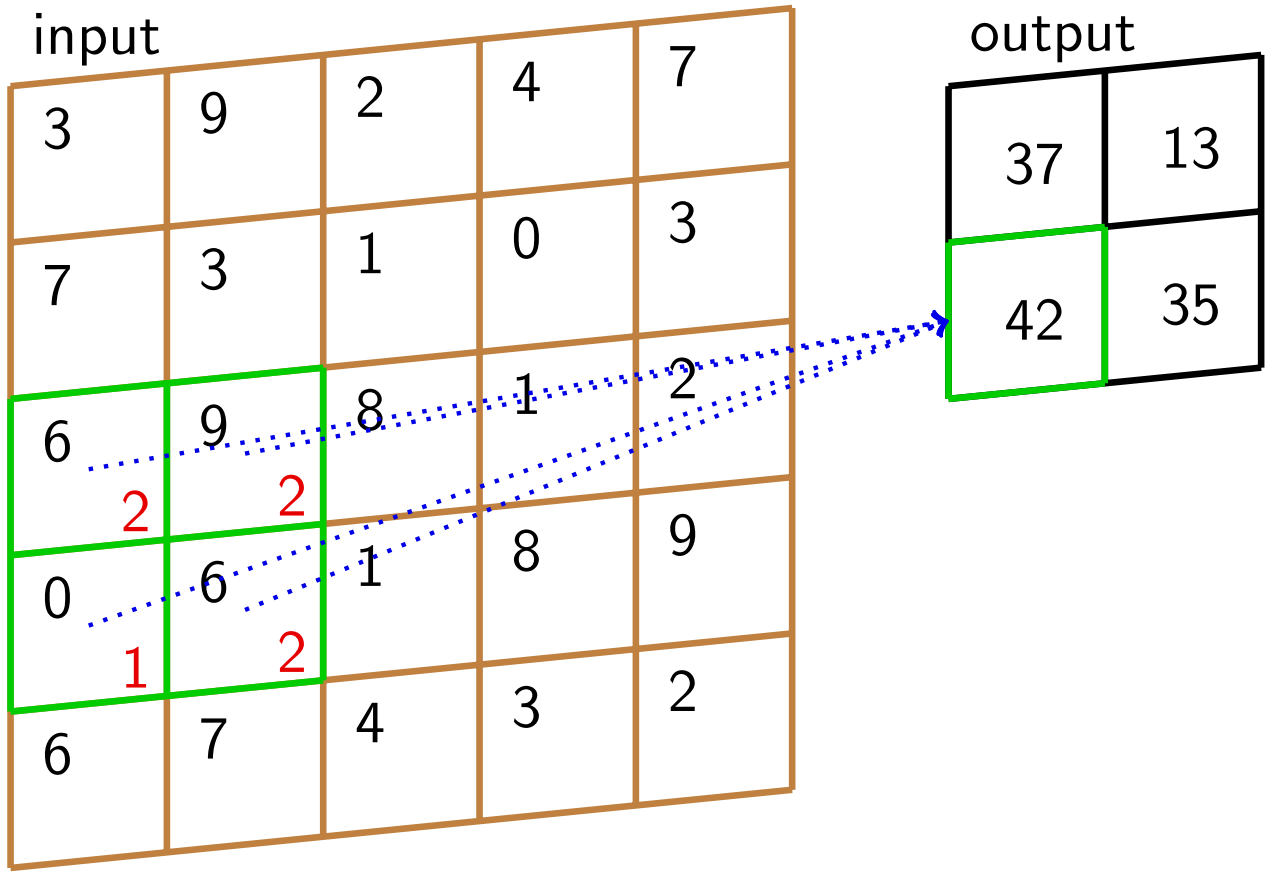
filter

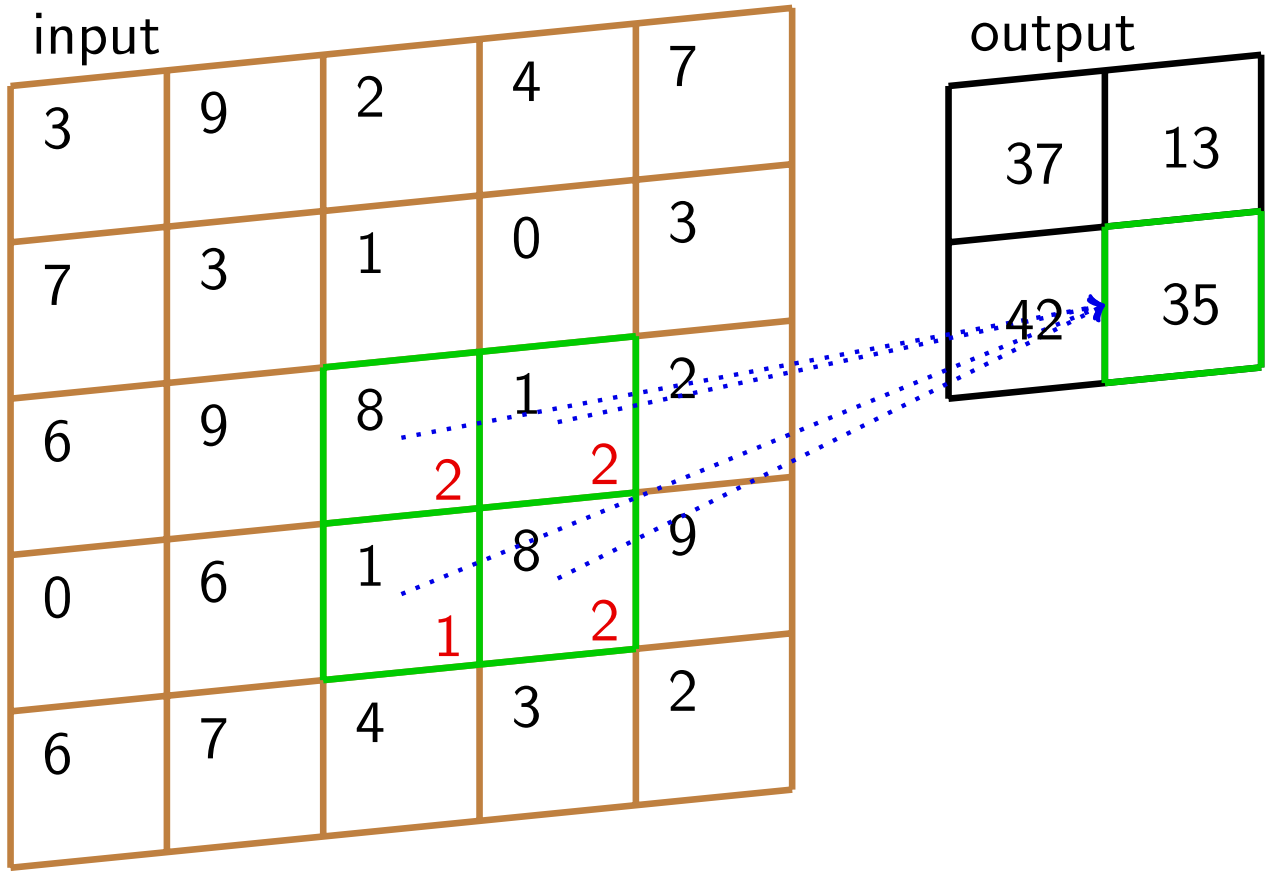
2	2
1	2

With stride=2, we skip 2 cells each time.


input
3
9
2
4
7
2
2
7
3
1
0
3
1
2
6
9
8
1
2
0
6
1
8
9
6
7
4
3
2
output
37
13
42
35


input
3
9
2
4
7
2
2
7
3
1
0
3
1
2
6
9
8
1
2
0
6
1
8
9
6
7
4
3
2
output
37
13
42
35


input
3 9 2 4 7
7 3 1 0 3
6 9 8 1 2
2 2
0 6 1 8 9
1 2
6 7 4 3 2
output
37 13
42 35



$N \times N$ input, $F \times F$ filter with stride S $\Rightarrow$ output size $\lfloor \frac{N-F}{S} \rfloor + 1$

Zero-padding, dilation and bias

- We often pad each side of the input with P zeros (or other constants)
  - this allows the filters to scan elements near the borders
- Sometimes, in a filter with dilation D, its cells are D cells apart ($D = 1$ in previous examples).
- $N \times N$ input, $F \times F$ filter, pad P zeros on each side, dilation D, stride $S \Rightarrow$ output size $\lfloor \frac{N+2P-D(F-1)-1}{S} \rfloor + 1$
- In general, each filter has a bias term as well.

Convolution beyond 2D

- In general, the input is not necessarily a 2D matrix, but can be a general N-dimensional array (1D, 2D, 3D,...)
- Similarly, a filter can be a general M-dimensional array (you can slide it through the input array as long as $M \leq N$).

Convolutional layer

- A convolutional layer often has several filters.
- Each filter produces a separate activation map.
- Filter weights are typically learned from data.

Your Turn

Which of the following statement is correct? (Multiple choice)

(a) A convolutional layer is a special kind of fully connected layer.

(b) Each neuron in a convolutional layer has to be connected to all input neurons.

(c) Convolutional layer is designed to extract features from array data.

Sub-sampling

- Sub-sampling (or pooling) is very similar to convolution.
- In average pooling, when we slide the filter through the input, we simply take the average of the input elements being scanned as the output.
- In max pooling, we replace average by max.
- The default stride is equal to the filter size (i.e. we do not pool the same element twice).

What You Need to Know...

Convolutional neural nets

- They are special types of MLPs with sparse connections between layers.
- Three key architectural ideas: local receptive fields, weight sharing, sub-sampling.
- Two special types of layers
  - Convolutional layers
  - Sub-sampling layers