COSI 165B
Deep Learning

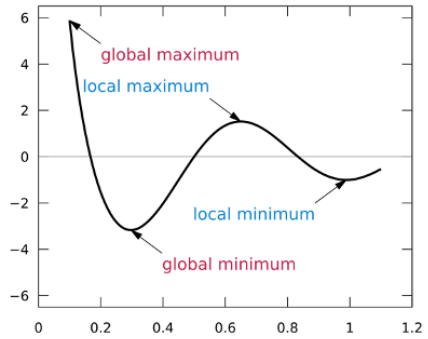
Chuxu Zhang
Computer Science Department
Brandeis University

3/1/2021

Last Lecture

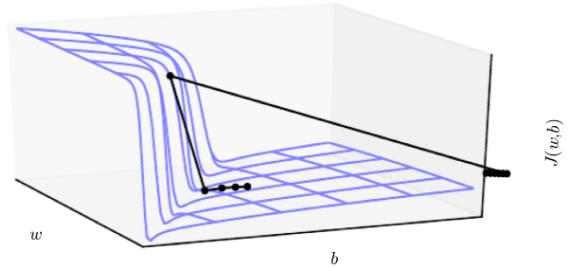
❑ Optimization

Local Minima



non-convex function (neural networks)

Exploding Gradients



cliff region (catapult parameters very far)

gradient clipping

Algorithm 8.1 Stochastic gradient descent (SGD) update at training iteration k

Require: Learning rate ϵ_k.
Require: Initial parameter $\boldsymbol{\theta}$
while stopping criterion not met **do**
  Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
  Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$
end while

Learning rate: it is necessary to gradually decrease the learning rate over time

$$\epsilon_k = (1 - \alpha)\epsilon_0 + \alpha\epsilon_\tau$$
$$\alpha = \frac{k}{\tau}.$$

linear decay

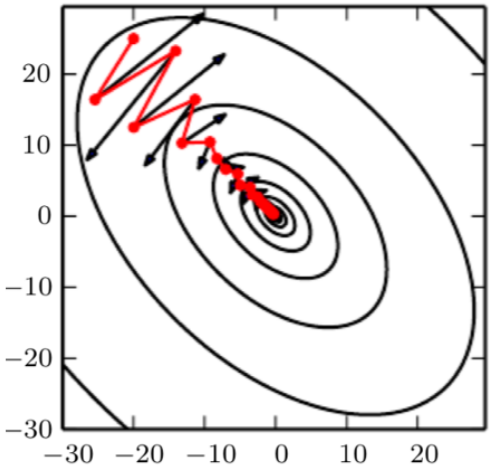
❑ Optimization

Algorithm 8.2 Stochastic gradient descent (SGD) with momentum

Require: Learning rate ϵ, momentum parameter α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
while stopping criterion not met **do**
  Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
  Compute gradient estimate: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
  Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
end while

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} L(\boldsymbol{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right),$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}.$$



The velocity v accumulates the gradient elements

The larger α is, the more previous gradients affect the current direction

Last Lecture

❑ Optimization

Algorithm 8.4 The AdaGrad algorithm

Require: Global learning rate ϵ
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant δ, perhaps 10^{-7}, for numerical stability
Initialize gradient accumulation variable $\boldsymbol{r} = \boldsymbol{0}$
while stopping criterion not met **do**
Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
Accumulate squared gradient: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$
Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$. (Division and square root applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end while

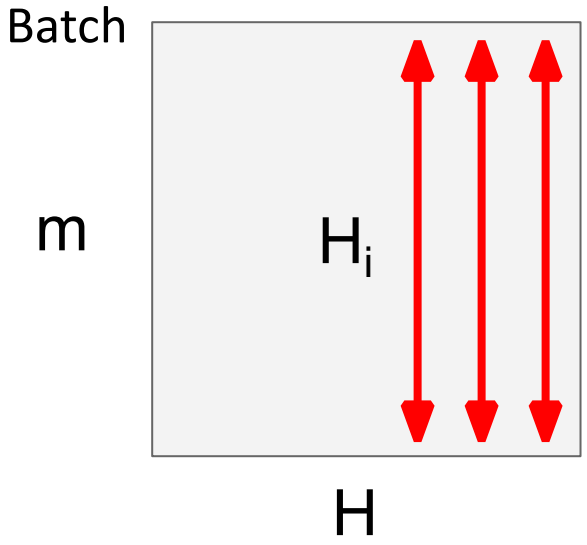
Algorithm 8.7 The Adam algorithm

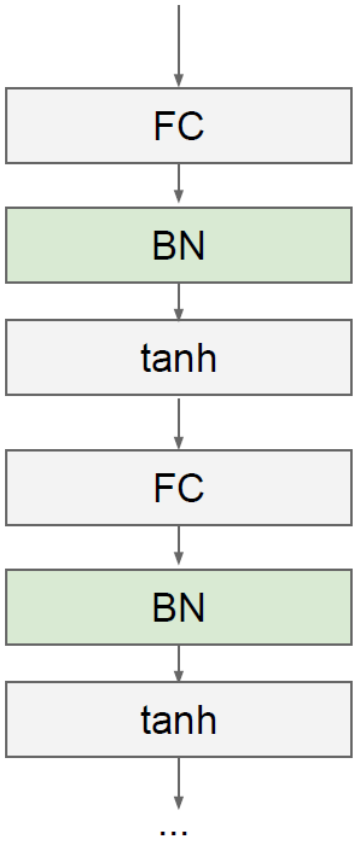
Require: Step size ϵ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in $[0, 1)$. (Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant δ used for numerical stabilization. (Suggested default: 10^{-8})
Require: Initial parameters $\boldsymbol{\theta}$
Initialize 1st and 2nd moment variables $\boldsymbol{s} = \boldsymbol{0}$, $\boldsymbol{r} = \boldsymbol{0}$
Initialize time step $t = 0$
while stopping criterion not met **do**
Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
$t \leftarrow t + 1$
Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t}$
Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t}$
Compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$ (operations applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end while

❑ Optimization

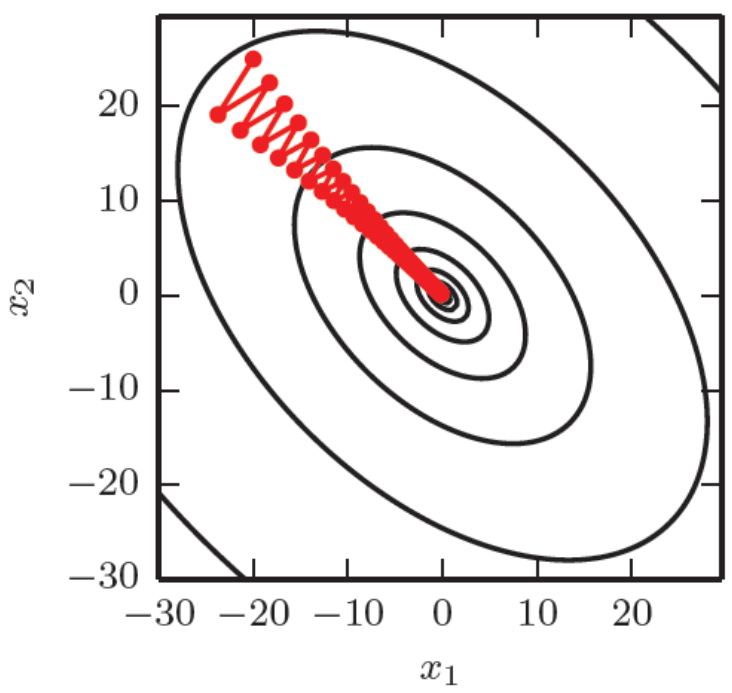
Batch Normalization

$$H' = \frac{H - \mu}{\sigma} \qquad \mu = \frac{1}{m}\sum_i H_{i,:} \qquad \sigma = \sqrt{\delta + \frac{1}{m}\sum_i (H - \mu)_i^2}$$





❑ Optimization



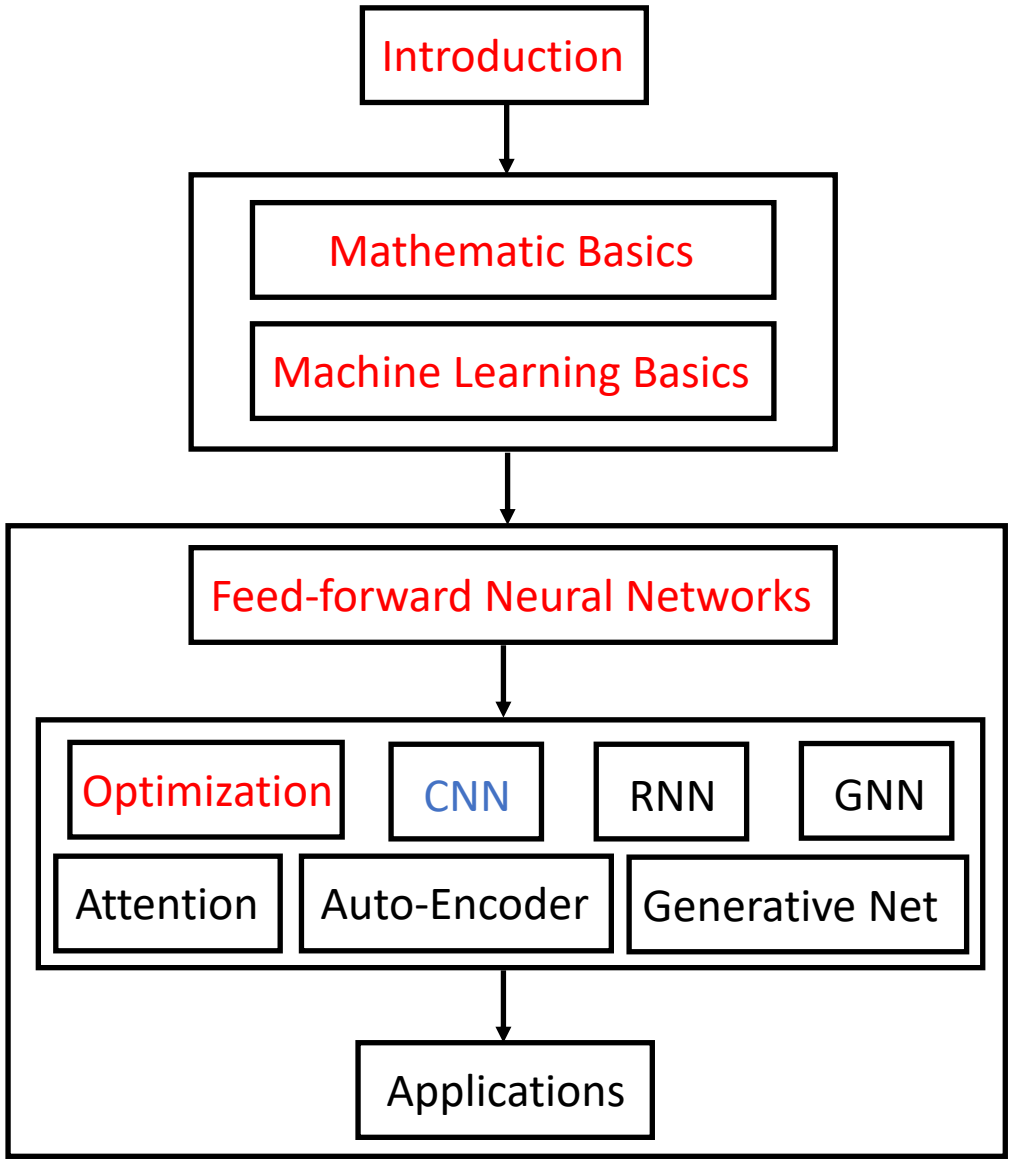
Ill-condition (Hessian matrix): the ratio of the largest singular value to the smallest singular value is large.

The loss function have contours that are very long ellipsoids.

The gradient directions away from the center are nearly orthogonal to the useful direction of descent.

Zig-zag path: slow learning.

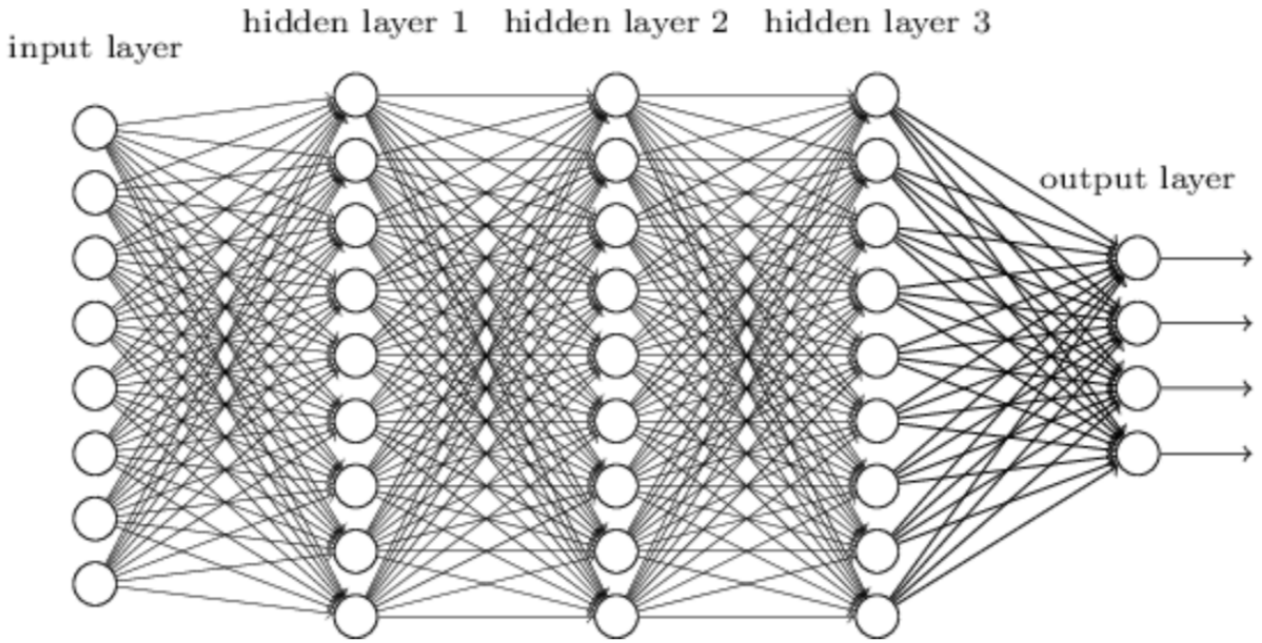
Structure of This Course



Convolutional Neural Networks

- Feed-forward neural network for image classification

Input: vector of image pixel values



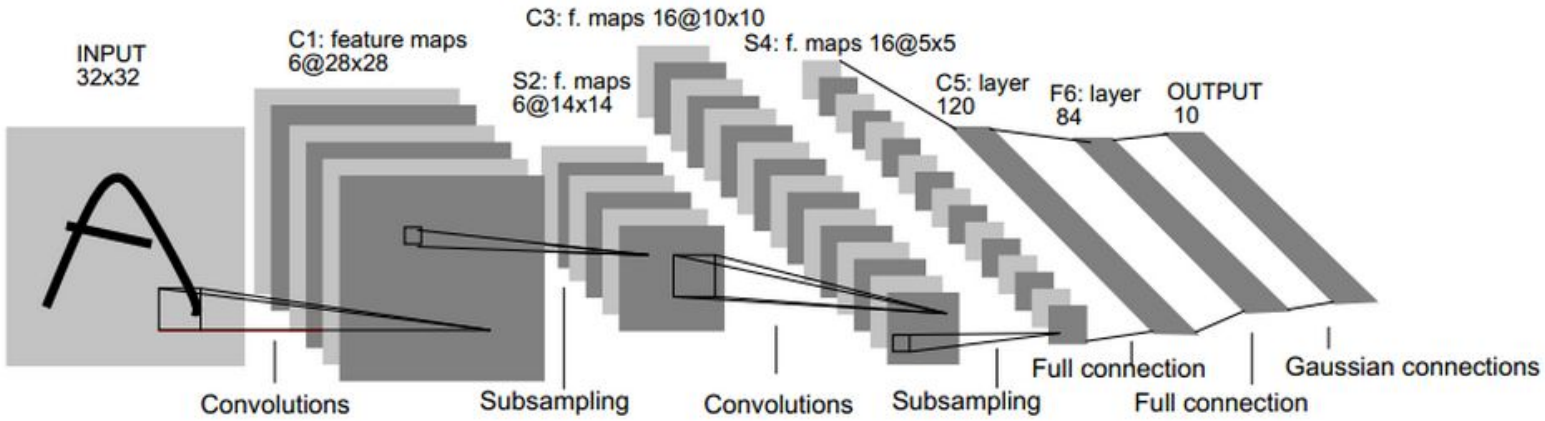
No spatial information!

Many parameters!

How can we use **spatial structure** in the input to inform the architecture of the network?

Convolutional Neural Networks

❑ What is CNN



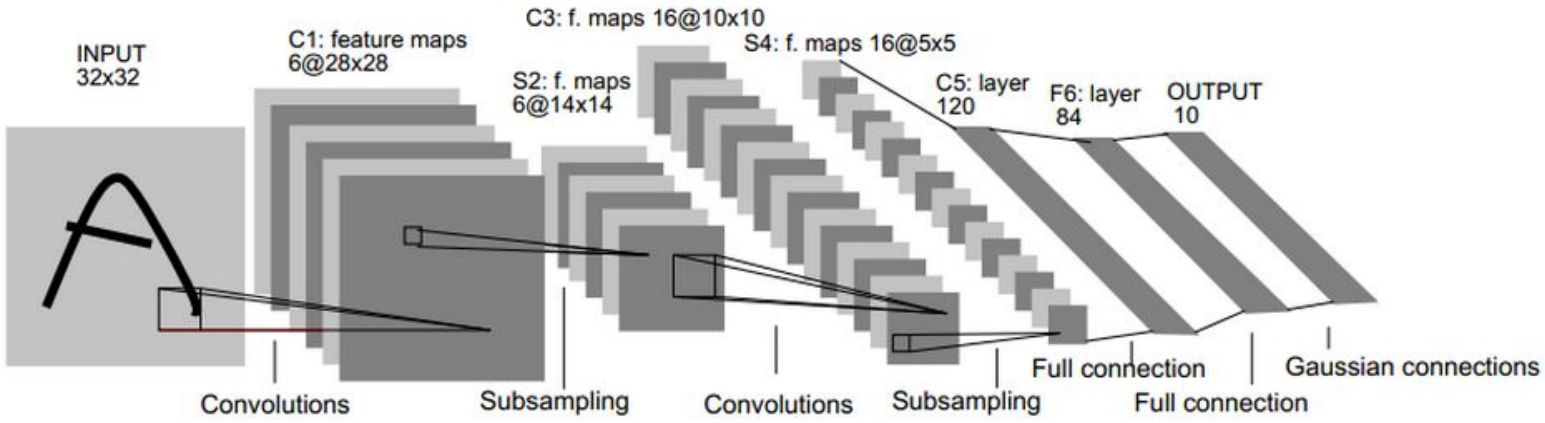
CNN is a specialized kind of neural network for processing data that has grid-like topology, such as image data which can be thought of as a 2D grid of pixels.

Convolutional networks have been tremendously successful in practical applications.

The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation.

Convolutional Neural Networks

- What is CNN



Convolutional Layer: convolution operation

Pooling Layer: max/mean pooling

Fully Connected Layer: feed-forward neural network

❑ Convolution

Example: tracking the location of a spaceship with a laser sensor

weighting function

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

single measurement

provide measurements at every instant in time is not realistic

discrete convolution

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

multidimensional array input/parameters (tensors)

kernel

feature map ← $s(t) = (x * w)(t)$

input

Convolution

1D Convolution

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

2D Convolution

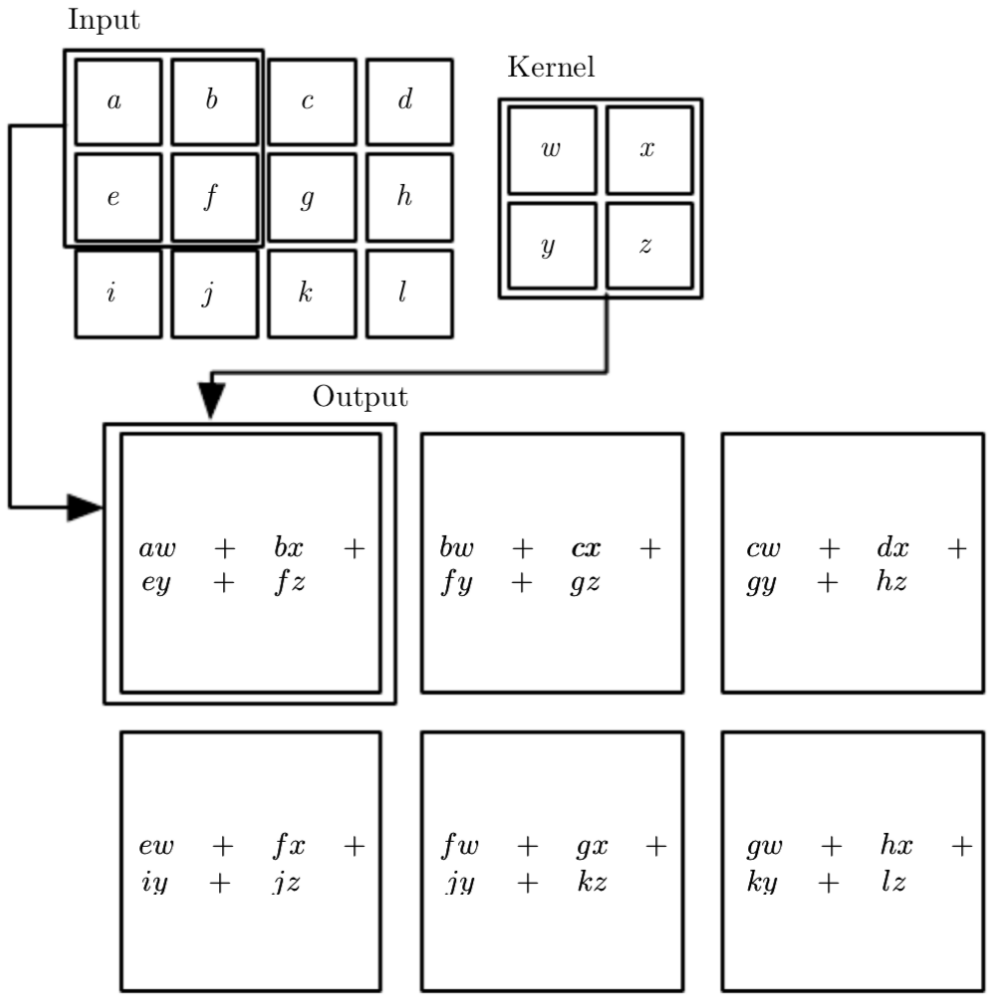
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

Convolution is commutative, meaning we can equivalently write

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

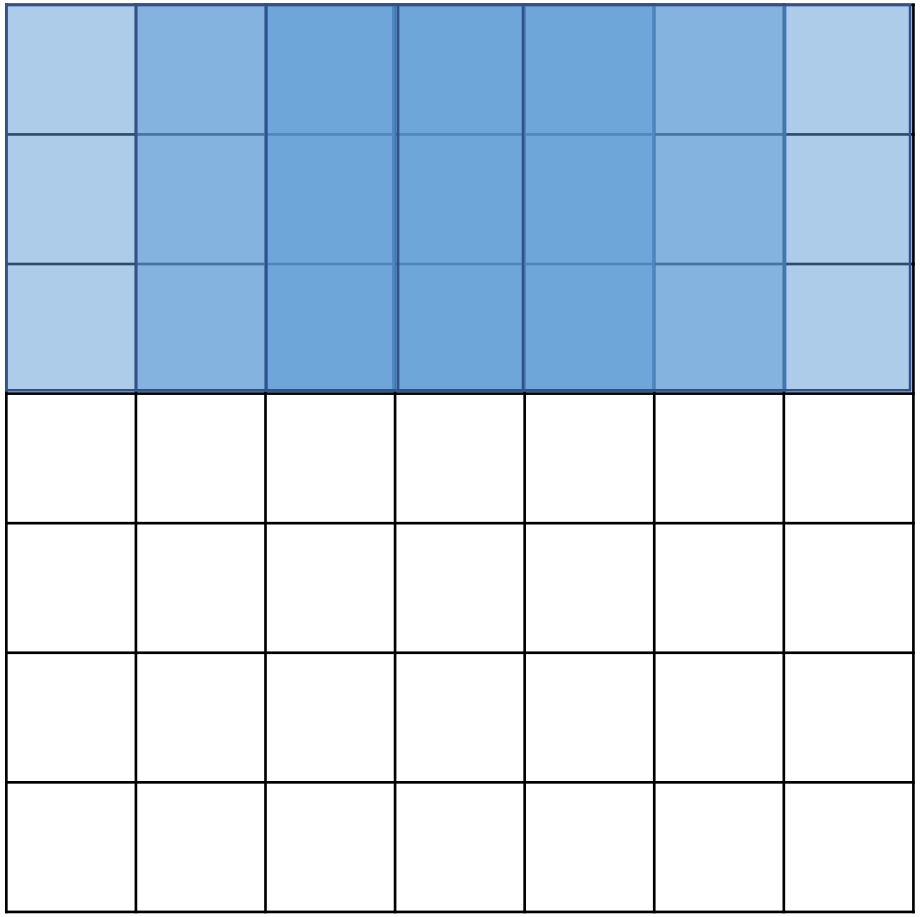
Convolution: example



$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

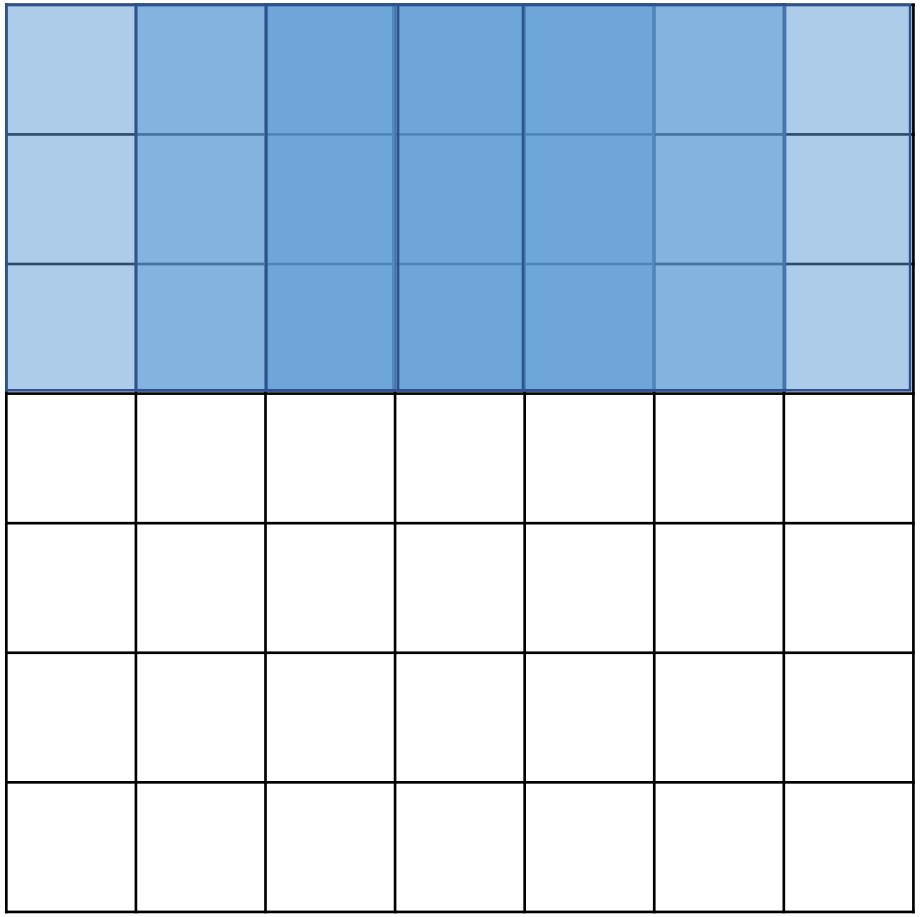
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

❑ Convolution: more examples



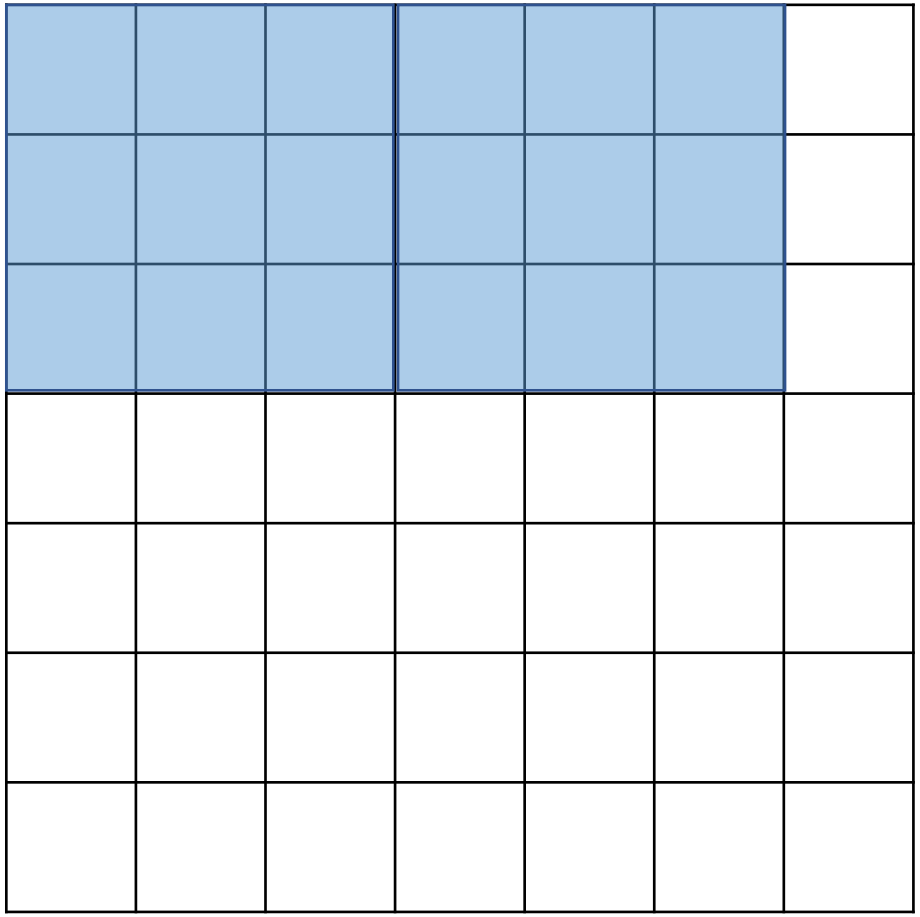
Input: 7x7
Kernel: 3x3
Stride: 1x1
Output: 5x5

❑ Convolution: more examples



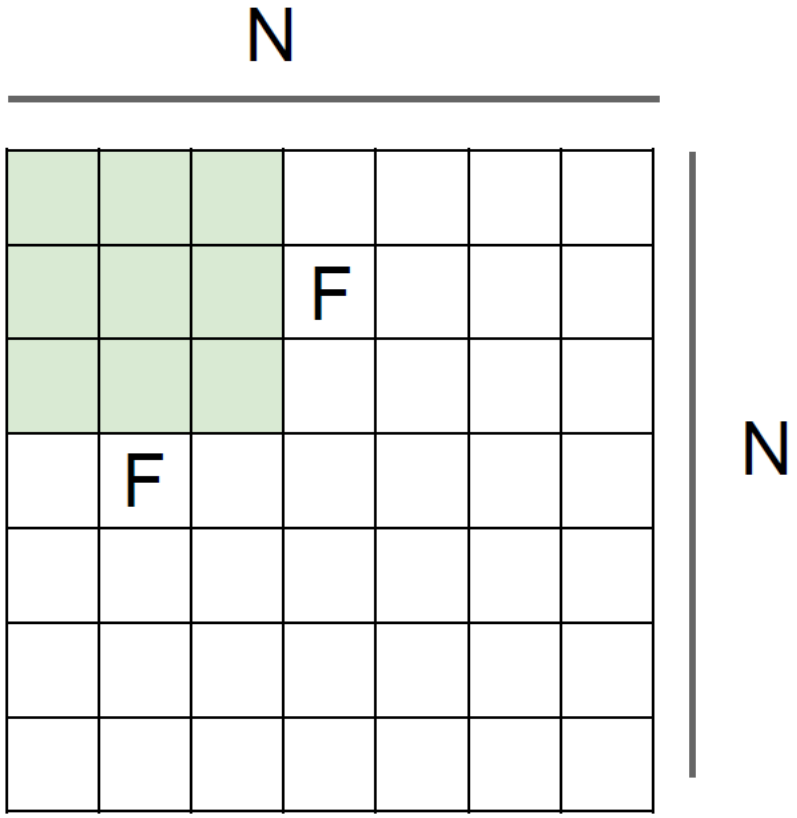
Input: 7x7
Kernel: 3x3
Stride: 2x2
Output: 3x3

❑ Convolution: more examples



Input: 7x7
Kernel: 3x3
Stride: 3x3
Output: invalid dimension

❑ Convolution: output size



Output Size = (N-F)/S + 1

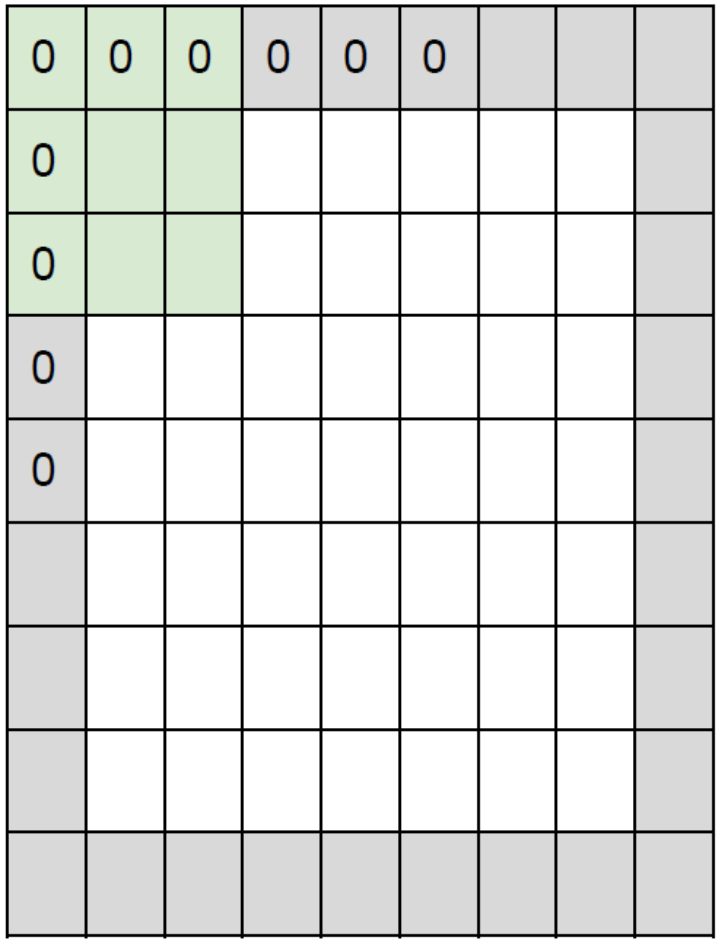
N = 7, F = 3, S = 1, Output = 5

N = 7, F = 3, S = 2, Output = 3

N = 7, F = 3, S = 3, Output = 2.3

How to fix?

❑ Convolution: zero padding

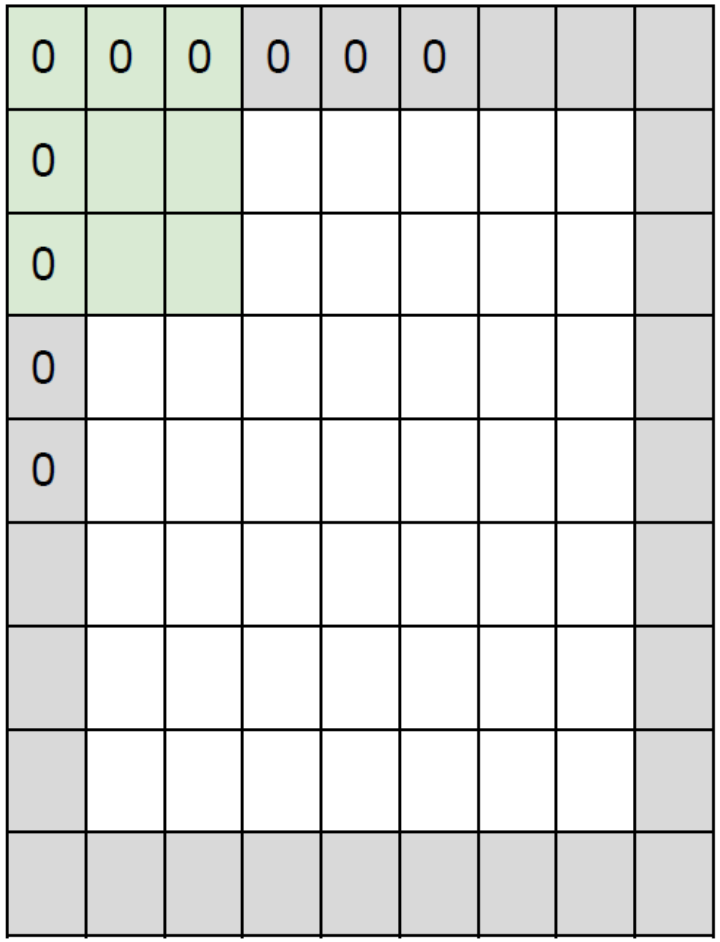


Input Size = (N-F)/S + 1

N = N + 2 (zero padding at boundary)

N = 9, F = 3, S = 3, Output = 3

❑ Convolution: zero padding



How many zero at boundary

$Z = [(M-1)S + F - N]/2$

When $S = 1$, $F-S/2$

$N = 7, F = 3, S = 3, M = 3, Z = 1$

❑ Motivation/Strengths

Sparse Interactions

MLP: fully connected, every output unit interacts with every input unit. $O(m \times n)$

Convolution: making the kernel smaller than the input, sparse connection $O(k \times n)$

Parameter Sharing

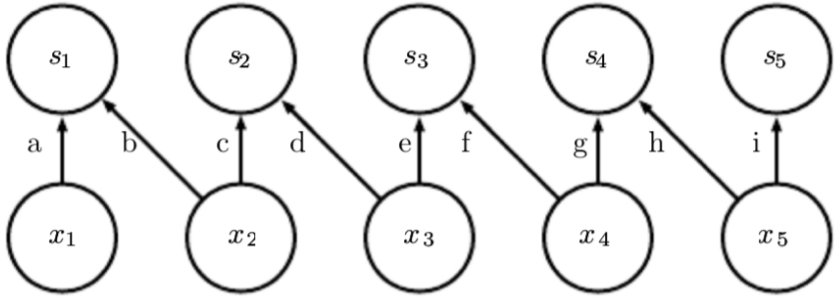
MLP: each element of the weight matrix is used exactly once when computing the output of a layer

Convolution: each member of the kernel is used at every position of the input, learn only one set of parameters for every position

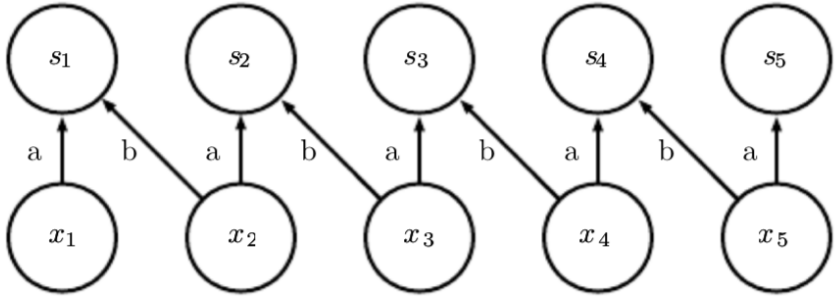
Equivariant Representations

If the input changes, the output changes in the same way: shift-convolution, convolution-shift

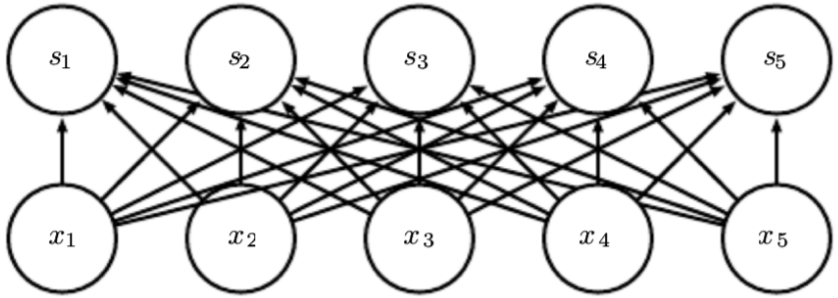
❑ Comparison



Local Connection: patch size = 2, different parameters



Convolution: patch size = 2, same parameters



Fully Connection: path size = 5, different parameters

❑ Pooling

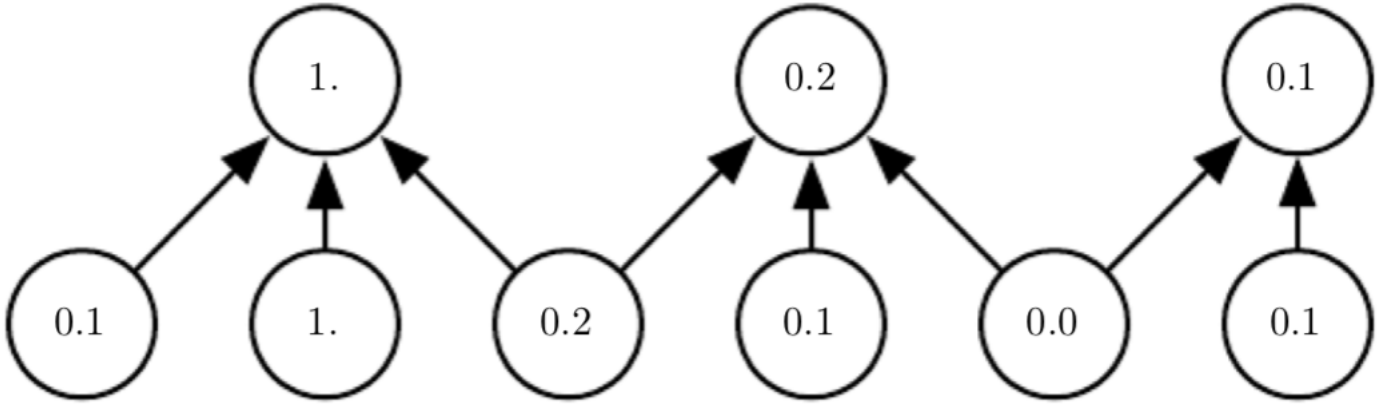
Use a pooling function to modify the output of the layer.

A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs.
e.g., max pooling: reports the maximum output within a rectangular neighborhood

In all cases, pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change.

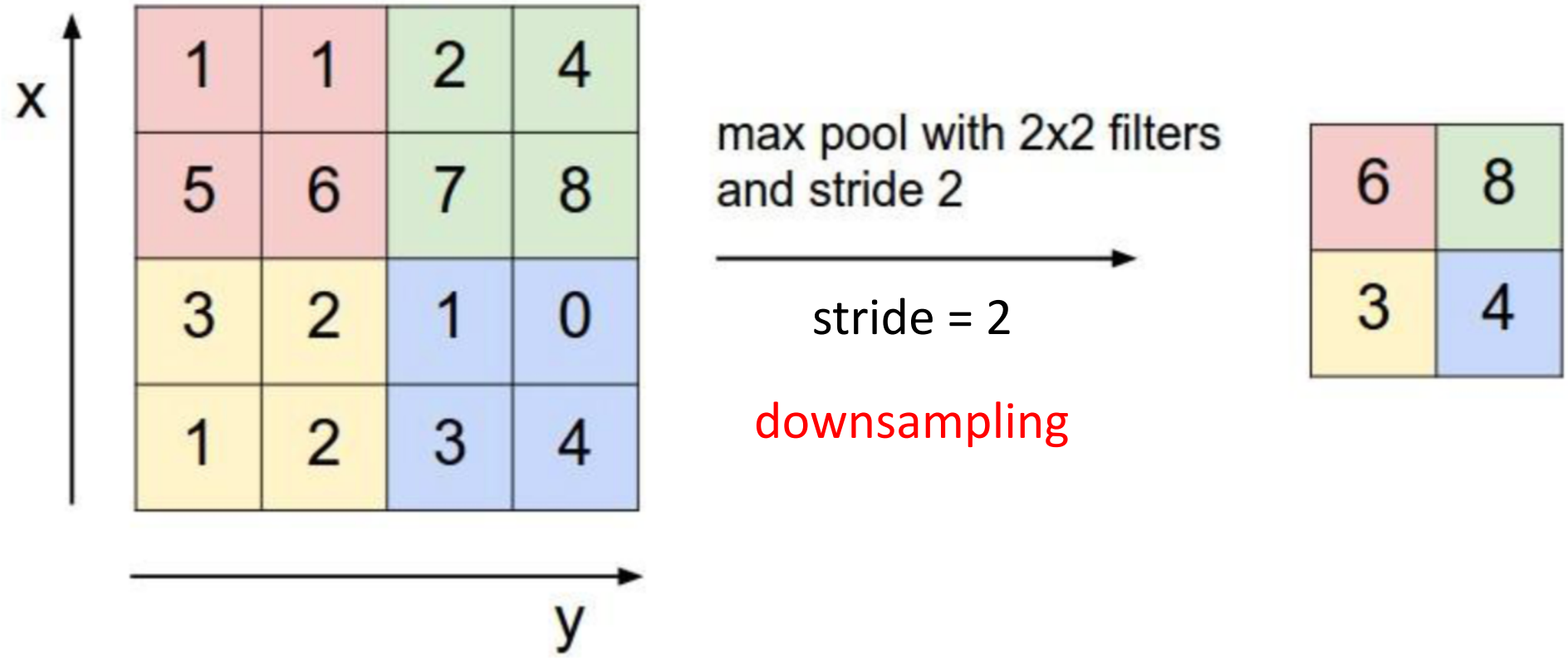
❑ Pooling: example

Report summary statistics for pooling regions spaced k pixels apart rather than 1 pixel apart. This improves the computational efficiency of the network because the next layer has roughly k times fewer inputs to process.



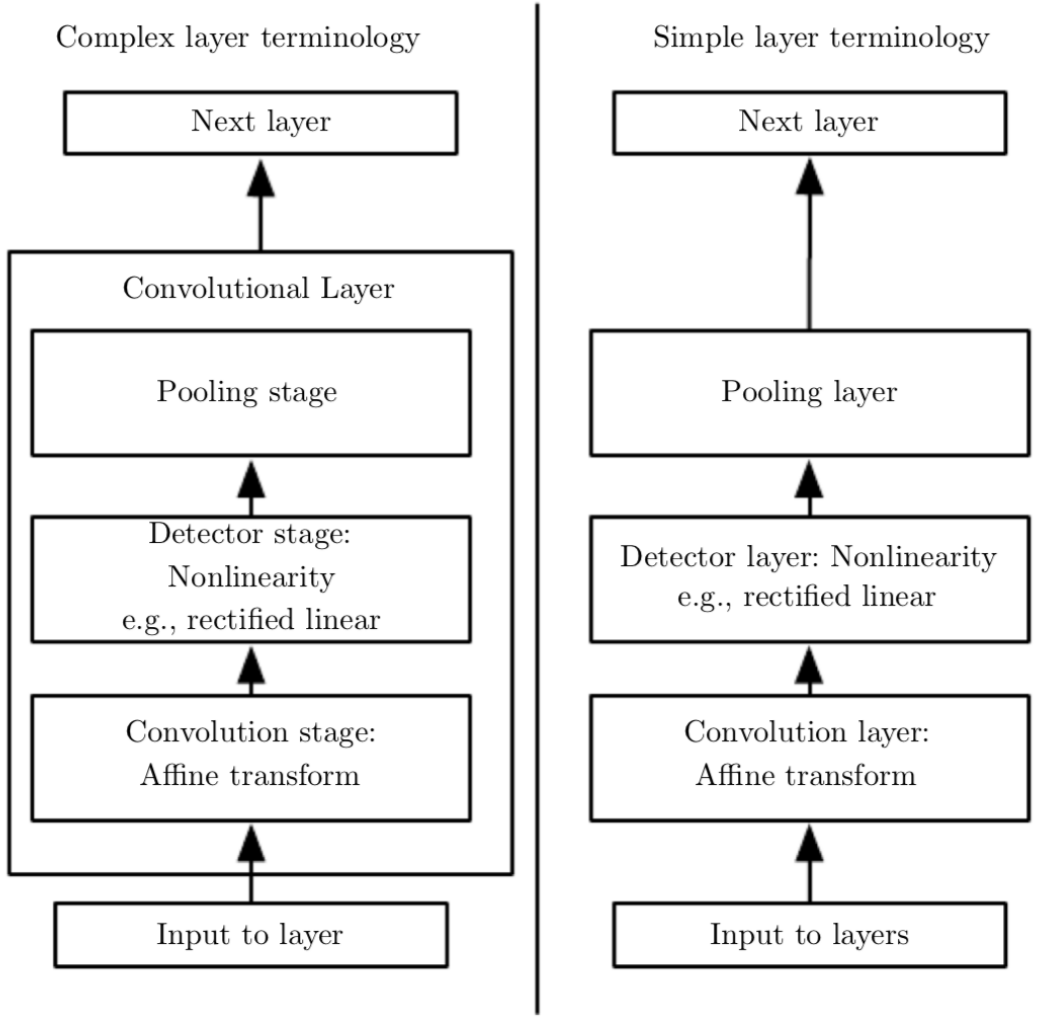
Pooling: max-pooling with a pool width of three and a stride between pools of two.

- Pooling: 2D example

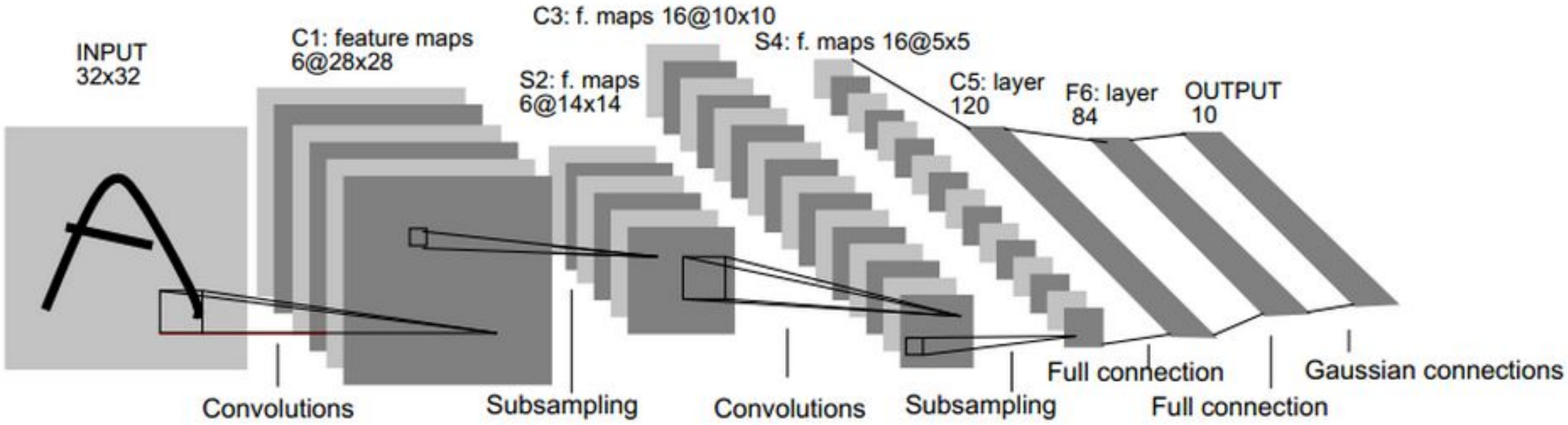


Convolutional Neural Networks

- The components of a convolutional layer



- A Full Model


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

Q & A