Deep Learning

Lecture 5

Fernando Perez-Cruz
based on Thomas Hofmann lectures

Swiss Data Science Center
ETH Zurich and EPFL – `datascience.ch`

October 22nd, 2018

Overview

1. Convolutional Layers

2. Convolutional Networks in Computer Vision

Section 1

Convolutional Layers

Integral Operators

Definition: Integral operator

A transform expressible with kernel H s.t. for any function f (for which Tf exists)

$$(Tf)(u) = \int_{t_1}^{t_2} H(u,t)f(t)\,dt$$

is called an integral operator.

- example: Fourier transform

$$(\mathcal{F}f)(u) := \int_{-\infty}^{\infty} e^{-2\pi itu} f(t)\,dt$$

Convolution

Definition: Convolution

Given two functions f, h, their convolution is defined as

$$(f * h)(u) := \int_{-\infty}^{\infty} h(u-t) f(t)\, dt = \int_{-\infty}^{\infty} f(u-t) h(t)\, dt$$

- integral operator with kernel $H(u,t) = h(u-t)$
- shift-invariant as $H(u-s, t-s) = h(u-t) = H(u,t)\ (\forall s)$
- convolution operator is commutative
- existence depends on properties of f, h
- typical use: $f =$ signal, $h =$ fast decaying kernel function

Linear Time-Invariant Transforms

Definition: Linear transform

T is linear, if for all functions f, g and scalars α, β,

$$T(\alpha f + \beta g) = \alpha T f + \beta T g$$

Definition: Translation invariant transform

T is translation (or shift) invariant, if for any f and scalar τ,

$$f_\tau(t) := f(t + \tau), \quad (T f_\tau)(t) = (T f)(t + \tau)$$

Theorem: Any linear, translation-invariant transformation T can be written as a convolution with a suitable h.

Signal Processing with Neural Networks

1. Transforms in deep networks: linear + simple non-linearity

2. Many signals (audio, images, etc.) obey translation invariance

$\Longrightarrow$ invariant feature maps: shift in input = shift in feature map

1.+2.

$\Longrightarrow$ learn convolutions, not (full connectivity) weight matrices

$\Longrightarrow$ convolutional layers for signal processing

Discrete Time Convolutions

For all practical purposes: signals are sampled, i.e. discrete.

Definition: Discrete convolution

For $f, h : \mathbb{Z} \to \mathbb{R}$, we can define the discrete convolution via

$$(f * h)[u] := \sum_{t=-\infty}^{\infty} f[t]\, h[u-t]$$

- use of rectangular brackets to suggest "arrays"
- 2D case left as an exercise
- typical: h with finite support (window size)

Discrete Time Convolutions: Example

Example: small Gaussian kernel with support $[-2:2] \subset \mathbb{Z}$

$$h[t] = \frac{1}{16} \begin{cases} 6 & t = 0 \\ 4 & |t| = 1 \\ 1 & |t| = 2 \\ 0 & \text{otherwise} \end{cases}$$

Consequence: convolution sum can be truncated

$$(f * h)[u] = \sum_{t=u-2}^{u+2} f[t]\, h[u-t] = \sum_{t=-2}^{2} h[t]\, f[u-t]$$

$$= \frac{1}{16} \left(6f[u] + 4f[u-1] + 4f[u+1] + f[u-2] + f[u+2]\right)$$

Discrete Cross–Correlations

Definition: Discrete cross-correlation

Let $f, h : \mathbb{Z} \to \mathbb{R}$, then

$$(h \star f)[u] := \sum_{t=-\infty}^{\infty} h[t]\, f[u+t]$$

- also called a "sliding inner product", $u+t$ instead of $u-t$
- note that cross-correlation and convolution are closely related

$$\begin{aligned}(h \star f)[u] &= \sum_{t=-\infty}^{\infty} h[t]\, f[u+t] = \sum_{t=-\infty}^{\infty} h[-t]\, f[u-t] \\ &= (\bar{h} * f)[u] = (f * \bar{h})[u], \quad \text{where } \bar{h}[t] := h[-t]\end{aligned}$$

only difference: kernel "flipped over" (non-commutative).

Convolution via Matrices

In practice: signal f and kernel h have finite support.

W.l.o.g.: $f[t] = 0$ for $t \notin [1:n]$, $h[t] = 0$ for $t \notin [1:m]$, $m \leq n$

We can think of f and h as vectors and define:

$$(f * h) = \underbrace{\begin{pmatrix} h_1 & 0 & 0 & 0 & \dots & 0 & 0 \\ h_2 & h_1 & 0 & 0 & \dots & 0 & 0 \\ h_3 & h_2 & h_1 & 0 & \dots & 0 & 0 \\ \vdots & & & & & & \vdots \\ 0 & 0 & 0 & 0 & \dots & h_m & h_{m-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & h_m \end{pmatrix}}_{:=\mathbf{H}_n^h \in \mathbb{R}^{(n+m-1)\times n}} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ \dots \\ f_n \end{pmatrix}$$

Toeplitz Matrices

Definition Toeplitz matrix

A matrix $\mathbf{H} \in \mathbb{R}^{k \times n}$ is a Toeplitz matrix, if there exists $n + k - 1$ numbers c_l ($l \in [-(n-1) : (k-1)] \subset \mathbb{Z}$) such that

$$H_{i,j} = c_{i-j}$$

- in plain English: all NW-SE diagonals are constant

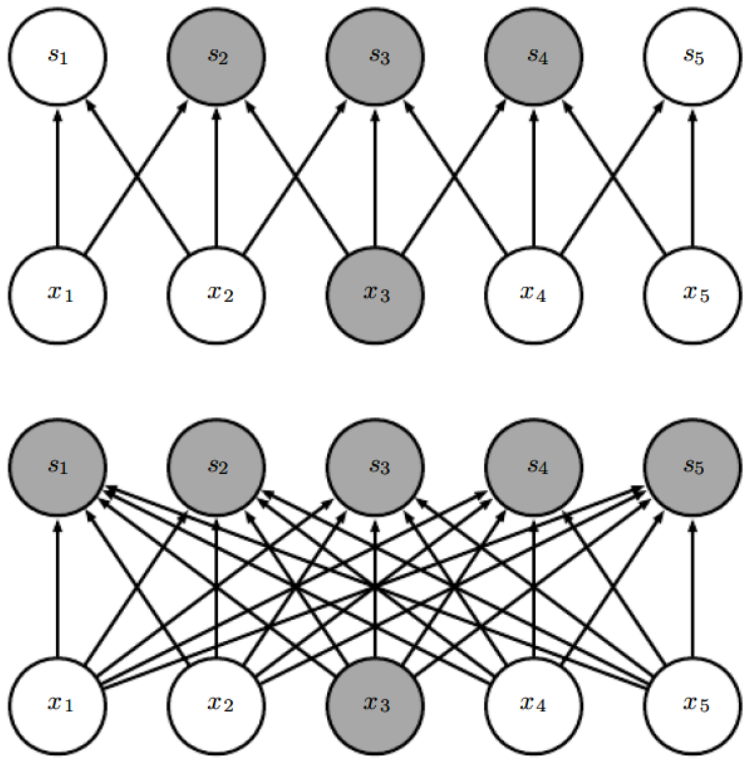
Proposition: $\mathbf{H}_n^h$ is a Toeplitz matrix.

- if $m \ll n$ (typical), a lot of additional sparseness (band matrix of diagonal width m)
- $\mathbf{H}_n^h$ has only m degrees of freedom
- locality (sparseness $m \ll n$) and weight sharing (kernel)

Sparse Connectivity

Schematic view sparse vs. dense connectivity.

(cf. DL, Figure 9.2)



Convolutions in Higher Dimensions

Generalize concept of convolution to ...

- $2D$: e.g. images, spectograms
- $3D$: e.g. color or multi-spectral images, voxel images, video
- or even higher dimensions

Replace vectors by

- matrices (e.g. in discrete case)

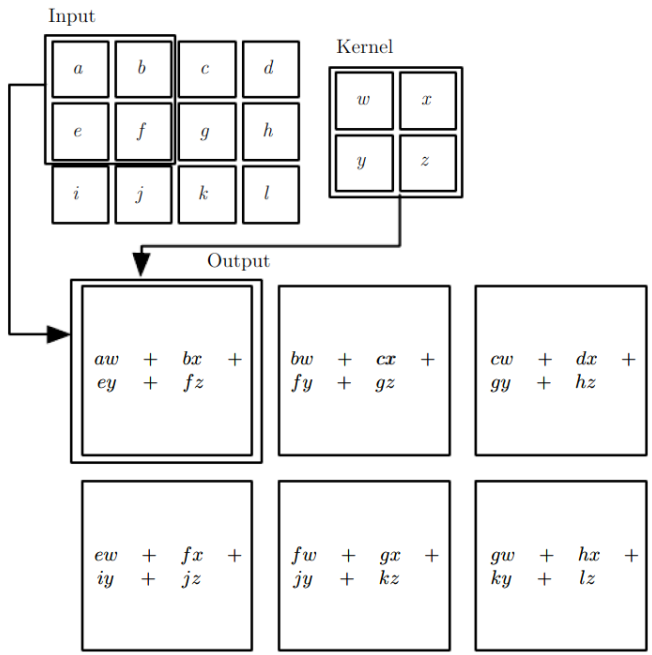
$$(F * G)[i, j] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} F[i-k, j-l] \cdot G[k, l]$$

- tensors: for $3D$ and higher

Convolutional Layers: Border Handling

Different options

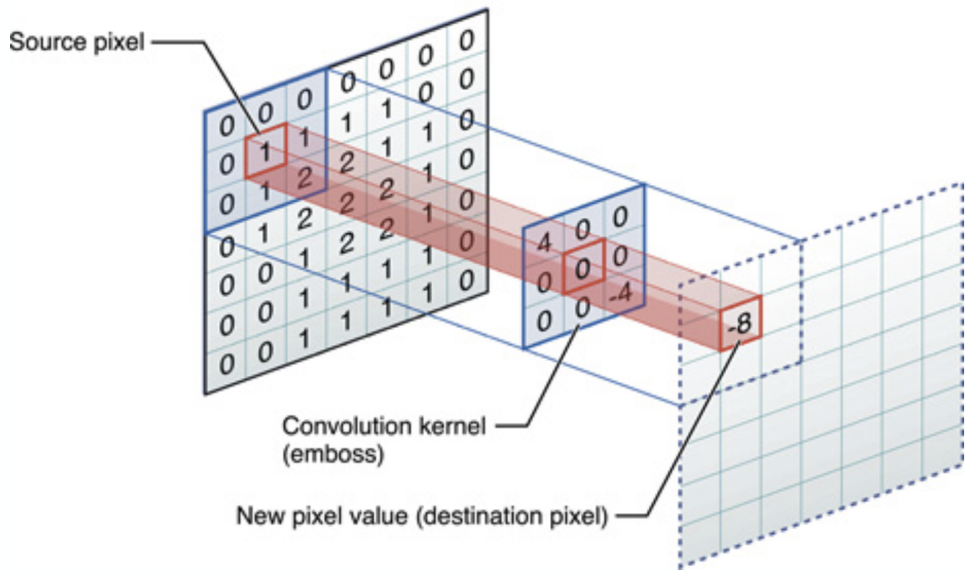
- our definition: padding with zeros = same padding
- only retain values from windows fully contained in support of signal f (see 2d example below, DL Fig. 9.1) = valid padding



Convolutional Layers: Layout

Convolved signal inherits topology of original signal.

Hence: units in a convolutional layer are typically arranged on the same grid (1D, 2D, 3D, ...)



Convolution Examples

original

filter (3 x 3)

0	0	0
0	1	0
0	0	0

identity

Convolution Examples

original

filter (5 x 5)

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

blur

Convolution Examples

original

filter (5 x 5)

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

sharpen

Convolution Examples

original

filter (3 x 3)

	0	0	0	
	-1	1	0	
	0	0	0	

vertical edge detector

Convolution Examples

original

filter (3 x 3)

	0	1	0	
	1	-4	1	
	0	1	0	

all edge detector

Convolutional Layers: Backpropagation

Exploit structural sparseness in computing $\frac{\partial x_i^l}{\partial x_j^{l-1}}$

- receptive field of x_i^l: $\mathcal{I}_i^l := \{j : W_{ij}^l \neq 0\}$
  (where $\mathbf{W}^l$ is the Toeplitz matrix of the convolution)
- obviously $\frac{\partial x_i^l}{\partial x_j^{l-1}} = 0$ for $j \notin \mathcal{I}_i^l$

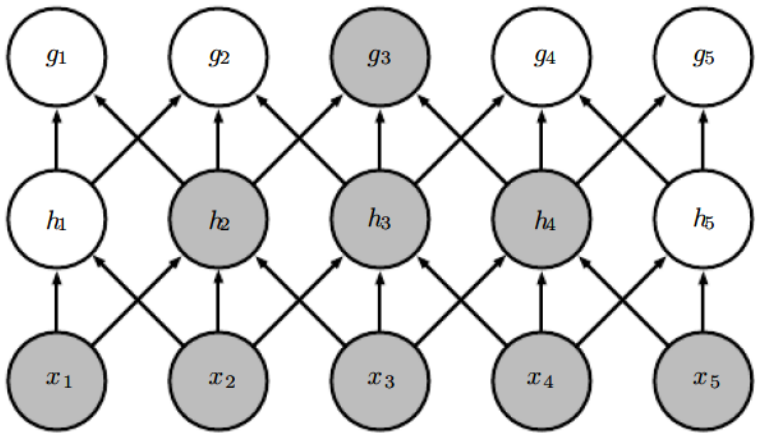
Weight sharing in computing $\frac{\partial \mathcal{R}}{\partial h_j^l}$, where h_j^l is a kernel weight

$$\frac{\partial \mathcal{R}}{\partial h_j^l} = \sum_i \frac{\partial \mathcal{R}}{\partial x_i^l} \frac{\partial x_i^l}{\partial h_j^l}$$

- weight is re-used for every unit within target layer
  $\Longrightarrow$ additive combination of derivatives in chain rule

Receptive Fields

Schematic view of the notion of a receptive field in a deep network. (cf. DL, Figure 9.4)



- nesting of convolutions: receptive fields grow

Efficient Computations of Convolutional Activities

FFT (Fast Fourier Transform): compute convolutions fast(er).

- Fourier transform of signal $f \to (\mathcal{F}f)$ and kernel $h \to (\mathcal{F}h)$
- pointwise multiplication and inverse Fourier transform

$$(f * h) = \mathcal{F}^{-1}\left((\mathcal{F}f) \cdot (\mathcal{F}h)\right)$$

- FFT: signal of length n, can be done in $\mathbf{O}(n \log n)$
- pays off, if many channels (amortizes computation of $\mathcal{F}f$)
- small kernels ($m < \log n$): favor time/space domain

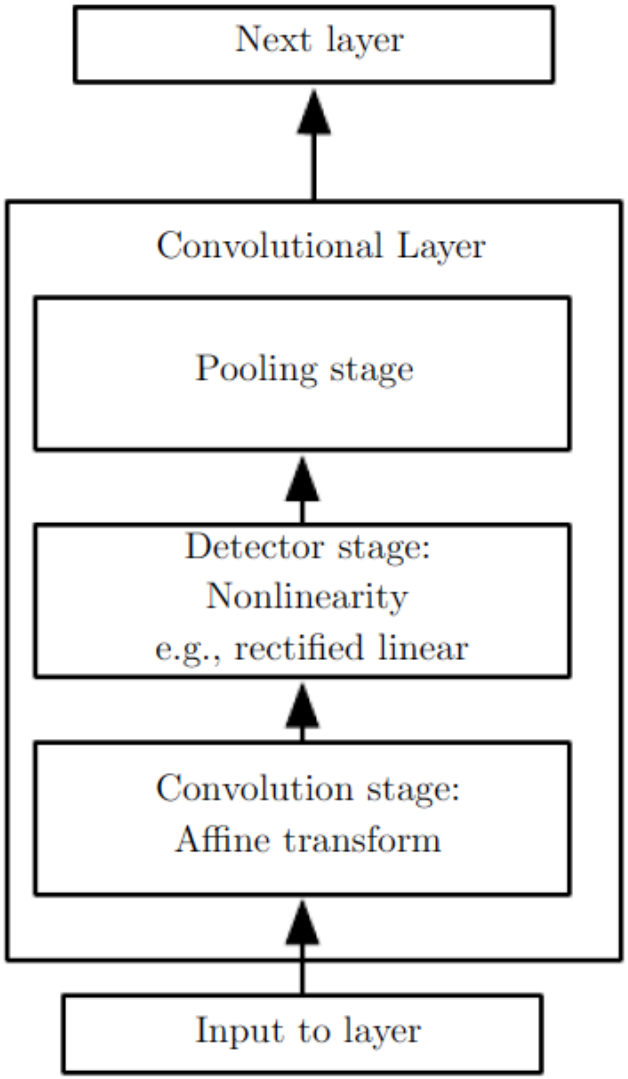
Convolutional Layers: Stages

Non-linearities: detector stage

as always: scalar non-linearities (activation function)

Pooling stage:

locally combine activities



Pooling

Most frequently used pooling function: max pooling

Define window size r (e.g. 3 or 3×3), then

$$\text{1D:} \quad x_i^{max} = \max\{x_{i+k} : 0 \leq k < r\},$$
$$\text{2D:} \quad x_{ij}^{max} = \max\{x_{i+k,j+l} : 0 \leq k, l < r\}$$

- maximum over a small "patch" of units
- other functions are possible: average, soft-maximization

Max-Pooling

Max-pooling: invariance

- set of invertible transformations $\mathcal{T}$: group w.r.t. composition
- $\mathcal{T}$-invariance through maximization

$$f_{\mathcal{T}}(\mathbf{x}) := \max_{\tau \in \mathcal{T}} f(\tau \mathbf{x})$$

Proposition: $f_{\mathcal{T}}$ is invariant under $\tau \in \mathcal{T}$

Proof:

$$f_{\mathcal{T}}(\tau \mathbf{x}) = \max_{\rho \in \mathcal{T}} f(\rho(\tau \mathbf{x})) = \max_{\rho \in \mathcal{T}} f((\rho \circ \tau)\mathbf{x}) = \max_{\sigma \in \mathcal{T}} f(\sigma \mathbf{x})$$

as $\forall \sigma$, $\sigma = \rho \circ \tau$ with $\rho = \sigma \circ \tau^{-1}$.

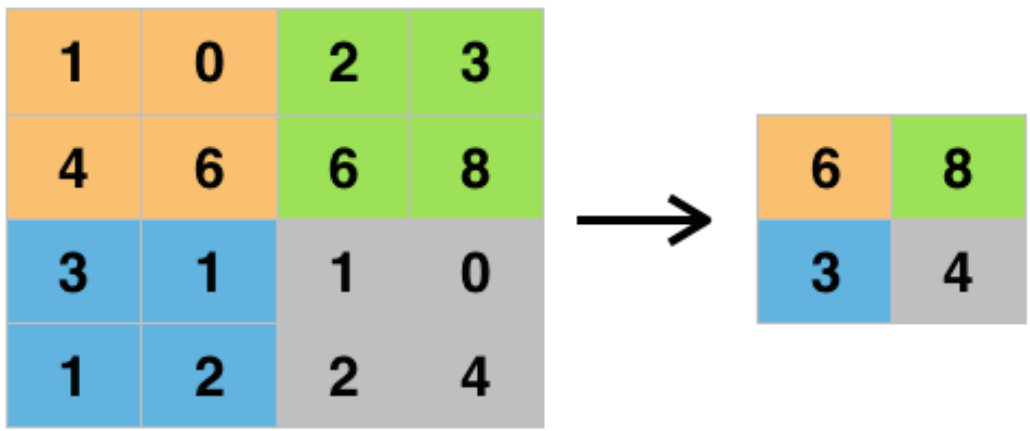
Sub-Sampling (aka "Strides")

Often, it is desirable to reduce the size of feature maps.

Sub-sampling: reduce temporal/spatial resolution

Often: combined with (max-)pooling (aka. stride)

Example: max-pool, filter 2x2, stride 2x2



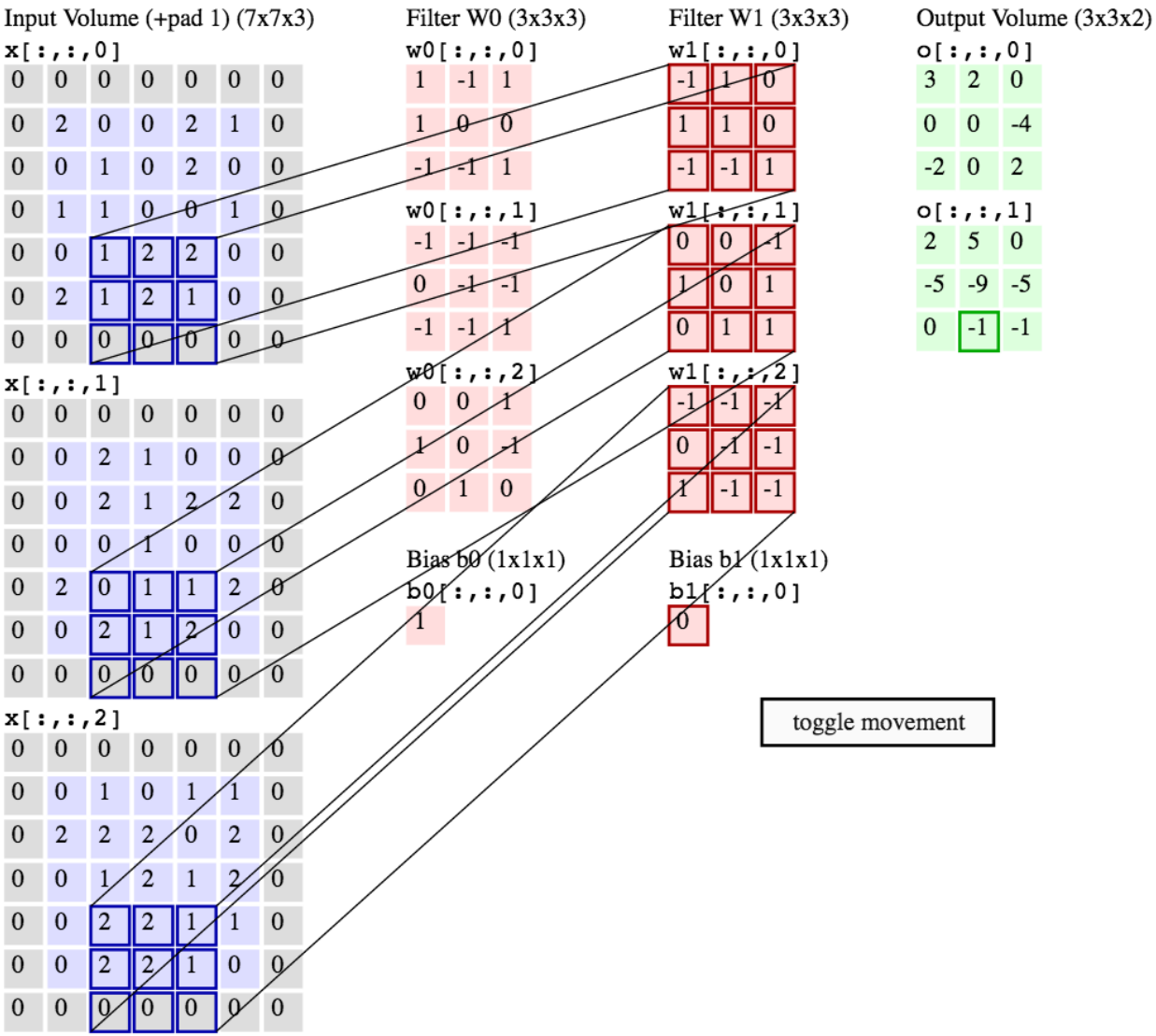
Disadvantage: loss of information.

Channels

Learn multiple convolution kernels (or filters) = multiple channels

- typically: all channels use same window size
- channels form additional dimension for next layers
  (e.g. 2D signal $\times$ channels = 3D tensor)
- number of channels: design parameter

Convolutional Layers: Animation

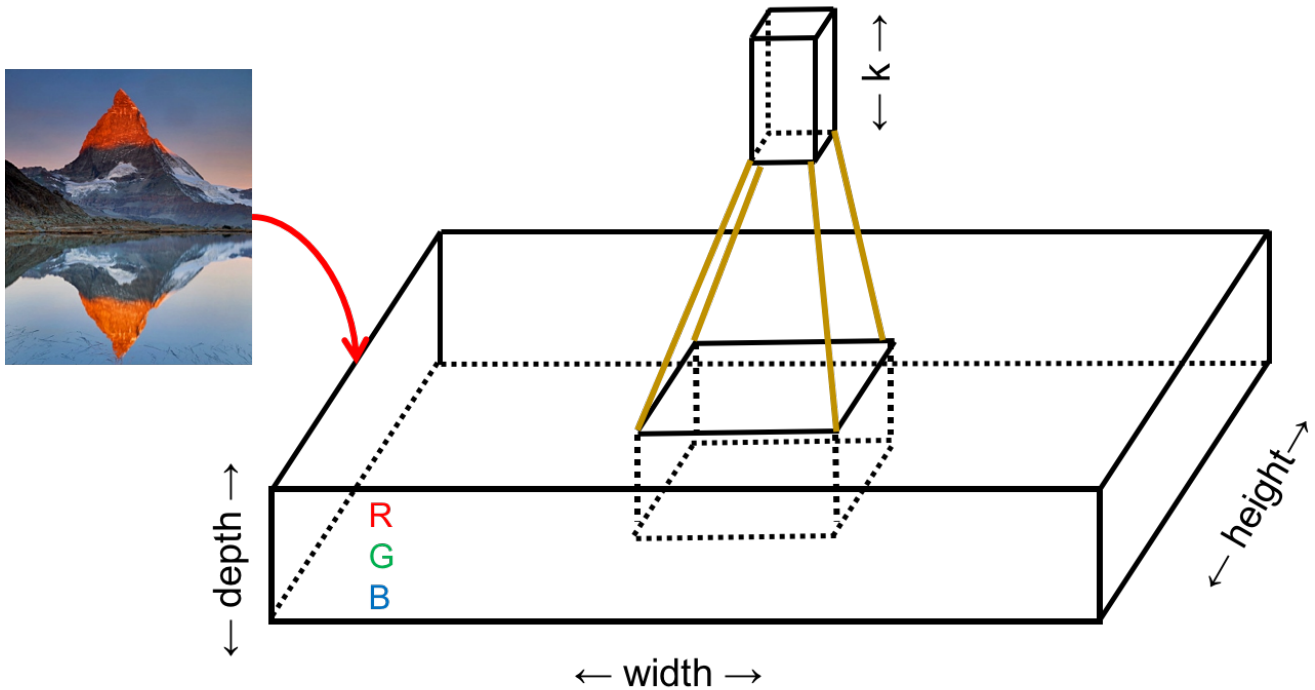


cs231n.github.io/assets/conv-demo/index.html

Section 2

Convolutional Networks in Computer Vision

Convolutional Layers for Vision



Note that "kernels" (across channels) form a linear map:

$$h : \mathbb{R}^{r^2 \times d} \to \mathbb{R}^k$$

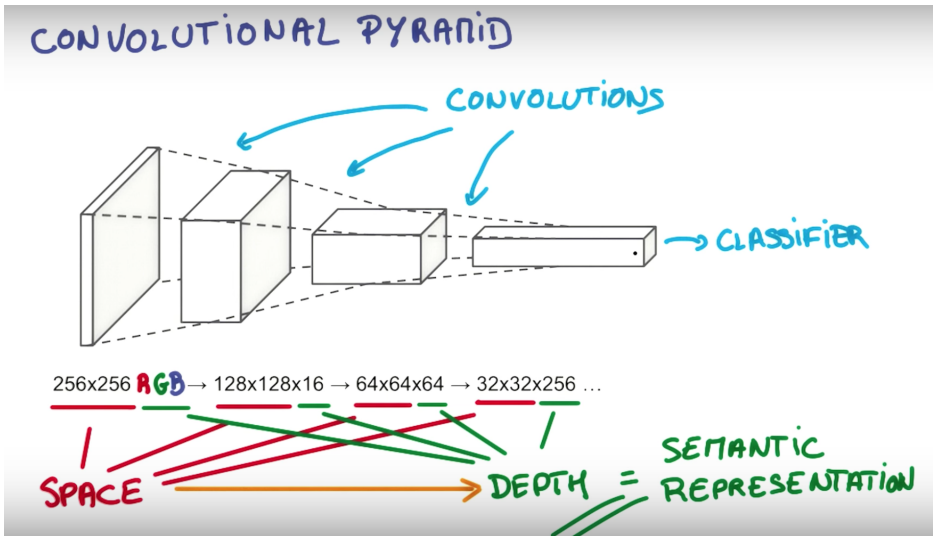
where $r \times r$ is the window size and d is the depth

Convolutional Pyramid

Typical use of convolution in vision: sequence of convolutions that

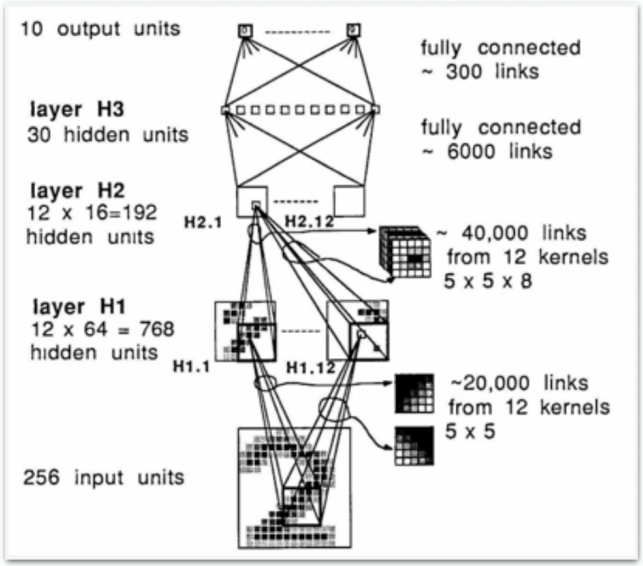
1. reduce spatial dimensions (sub-sampling) and
2. increase number of channels

$\Longrightarrow$ smaller, but more feature maps



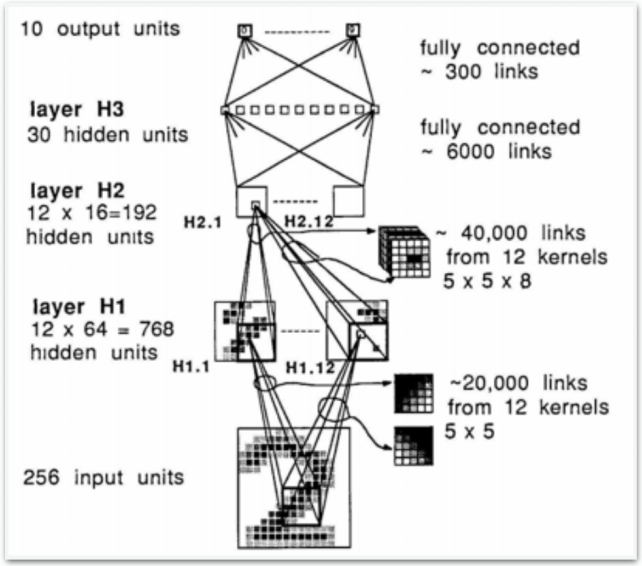
(from `www.udacity.com/course/deep-learning--ud730`)

LeNet 1989



- C1: 12 channels, 5x5 kernels, (768 units)
- C2: 12 channels, 6x6x8 kernels, (1600 units)
- F1: 196 channels: fully-connected
- F2: 30 channels: fully-connected
- output: Gaussian noise model (squared loss)

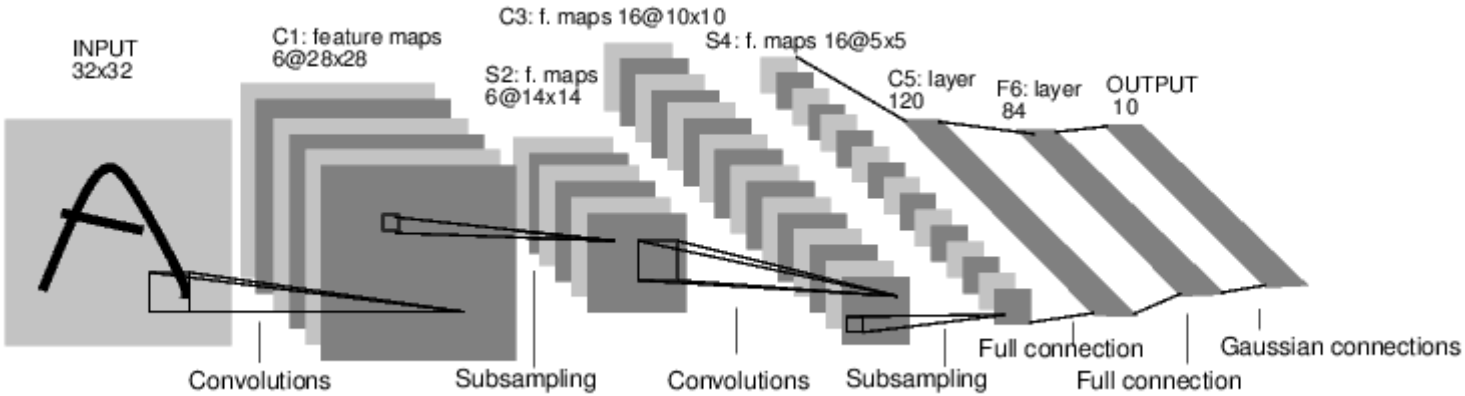
LeNet 1989



to have identical weight vectors. The eight maps in H1 on which a map in H2 takes its inputs are chosen according a scheme that will not be described here. Connections falling off the boundaries are treated like

- C1: 12 channels, 5x5 kernels, (768 units)
- C2: 12 channels, 6x6x8 kernels, (1600 units)
- F1: 196 channels: fully-connected
- F2: 30 channels: fully-connected
- output: Gaussian noise model (squared loss)

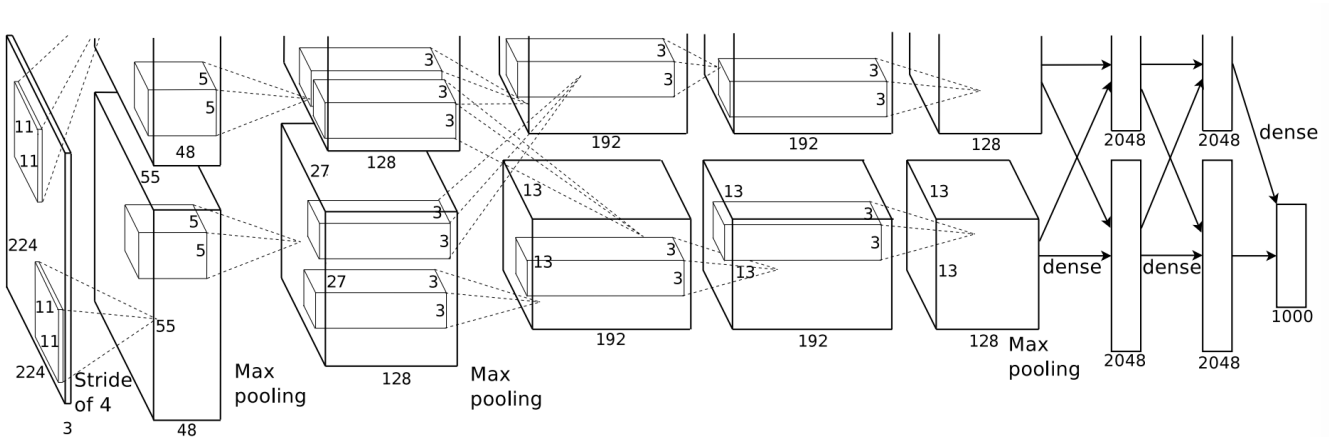
LeNet5



Architecture LeNet5 (LeCun et al, 1989; LeCun et al, 1998)

- C1/S2: 6 channels, (5x5 kernels), 2x2 sub (4704 units)
- C3/S4: 16 channels, (6: 6x6x3, 9 6x6x4 and 1 6x6x6 kernels), 2x2 sub (1600 units)
- C5: 120 channels, F6: fully-connected
- output: Gaussian noise model (squared loss)

AlexNet



AlexNet

- Krizhevsky, Sutskever, Hinton, 2012
  ImageNet Classification with Deep Convolutional NN
- 60 million parameters and 500,000 neurons
- 5 convolutional layers, some followed by max-pooling
- 2 globally connected layers with a final 1000-way softmax

Inception Module: 1x1 Convolution

Many channels needed for high accuracy, typically $k \approx 200 - 1000$ (e.g. AlexNet: $2\text{x}192$).

Observation (motivated by Arora et al, 2013): when convolving, dimension reduction across channels may be acceptable.

Dimension reduction: m channels of a 1x1xk convolution $m \leq k$:

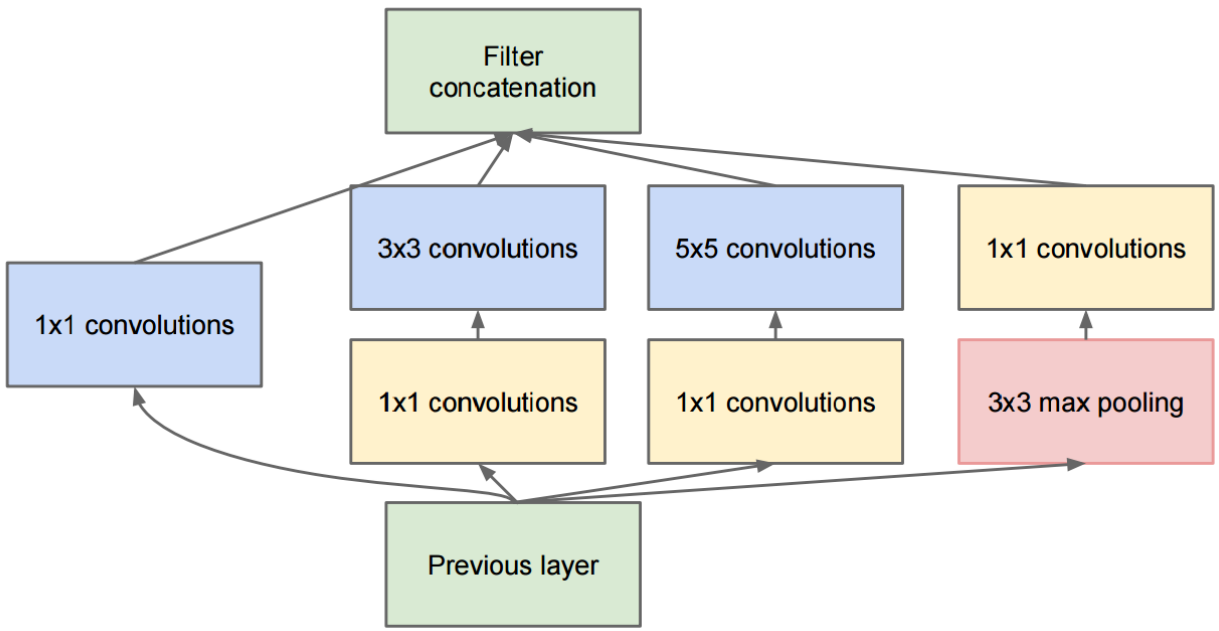
$$\mathbf{x}_{ij}^{+} = \sigma\left(\mathbf{W}\mathbf{x}_{ij}\right), \quad \mathbf{W} \in \mathbb{R}^{m \times k}$$

- "1x1 convolution" = no convolution
- inception module (Szegedy et al., 2014)
- network within a network (Lin et al., 2013)
- i.e. $\mathbf{W}$ is shared for all (i, j) (translation invariance)

Inception Module: Mixing

Instead of fixed window size convolution: mix 1x1 with 3x3 and 5x5, max-pooling.

Use 1x1 convolutions for dimension reduction before convolving with larger kernels



Google Inception Network

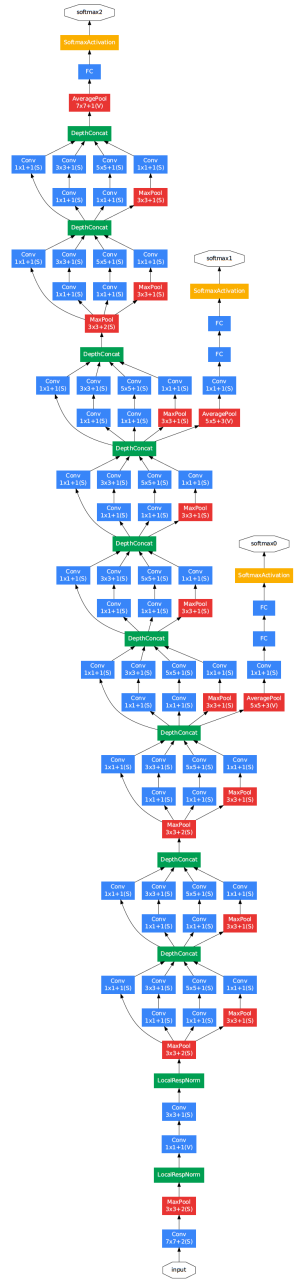
Very deep network: many inception modules (green boxes: concatenation points)

Additional trick: connect softmax layer (and loss) at intermediate stages (yellow boxes)

$\Longrightarrow$ gradient "shortcuts"

ImageNet results

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no



Google Inception Network

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								