

Deep Learning

2023-12-4 - 2023-12-8

Vecka 3

Ämne: Convolutional neural networks

Agenda

- Kort presentation av mig 😊
- Introduktion till CNN
- Applikationer av CNNs
- Lite historia
- Grunderna
 - Faltningar
 - Filter
 - Pooling
- Sätta ihop det!



Me, Myself and I

- Konsult inom AI och Machine Learning med fokus på innovation och projektledning.
- Studerat IT på Chalmers
- Tidigare arbetat med bl.a. deep learning för bildanalys och signalbehandling, dataanalys inom olika industrier. Senaste projektet var att bygga en applikation runt generativ AI.
- Mail: anna.nylander@iths.se



Me, Myself and I

- Måla, silversmide och keramik.
- Åker gärna på semester för att dyka.
- Försöker bli bra på att skattea och åka snowboard men det går sådär...
- Har två mysiga katter som gärna gör cameos vid videomötet.
- Gillar att laga mat men hatar att baka.



Agenda

- Kort presentation av mig 😊
- Introduktion till CNN
- Applikationer av CNNs
- Lite historia
- Grunderna
 - Faltningar
 - Filter
 - Pooling
- Sätta ihop det!



Introduktion

- Convolutional neural networks ("Faltande neurala nätverk") är neurala nätverk där minst 1 lager är ett convolutional layer.
- CNNs är väldigt bra på att hantera spatial data, t.ex. bilder.
- Kan lösa problem som
 - Image classification
 - Object detection
 - Semantisk segmentering

Image classification

- Olika typer av klassificeringar av hela bilden.

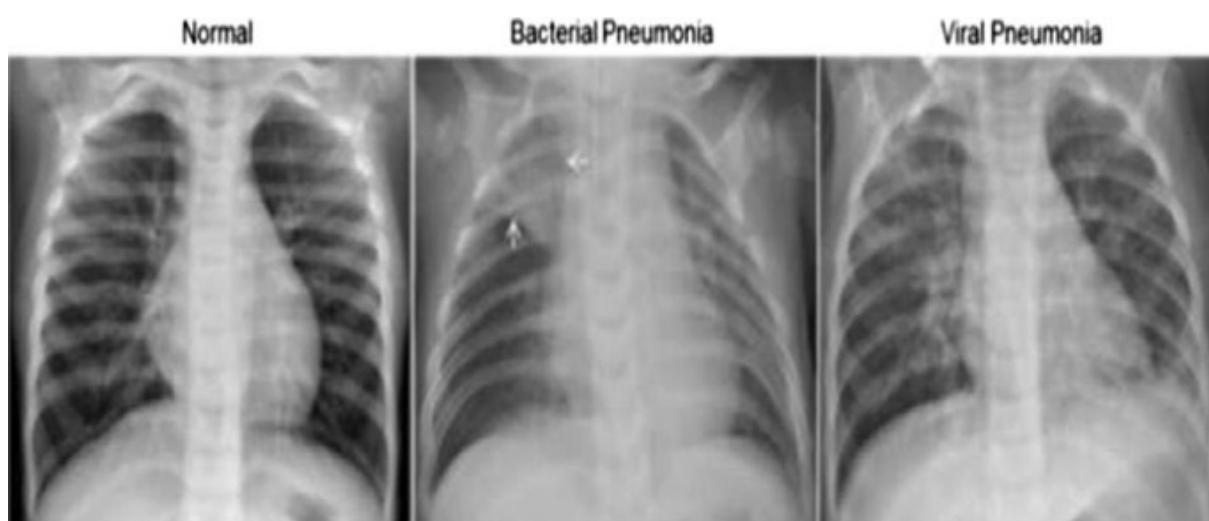


Image belongs
to Dog class



Image belongs
to Cat class

Image classification with localisation

Classification



CAT

Classification
+ Localization

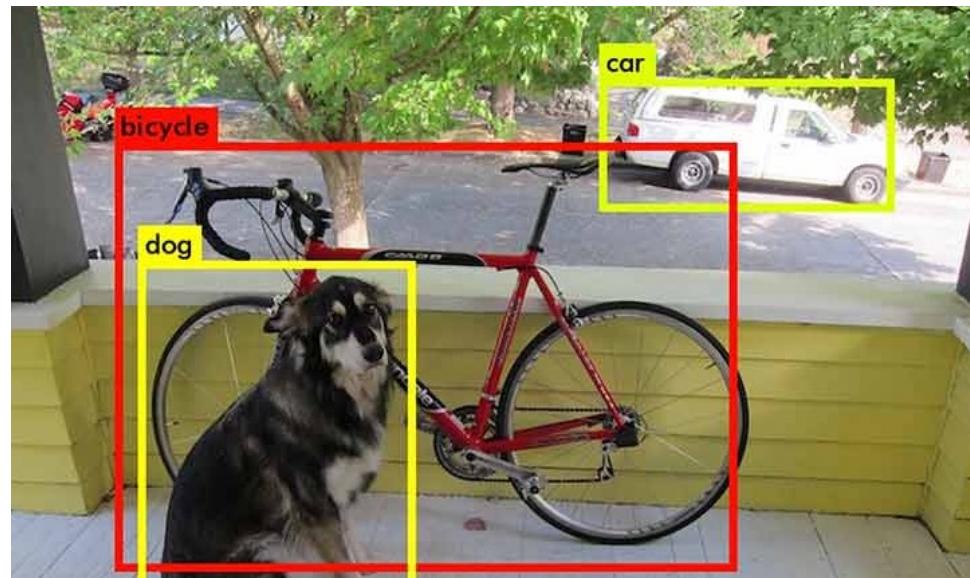
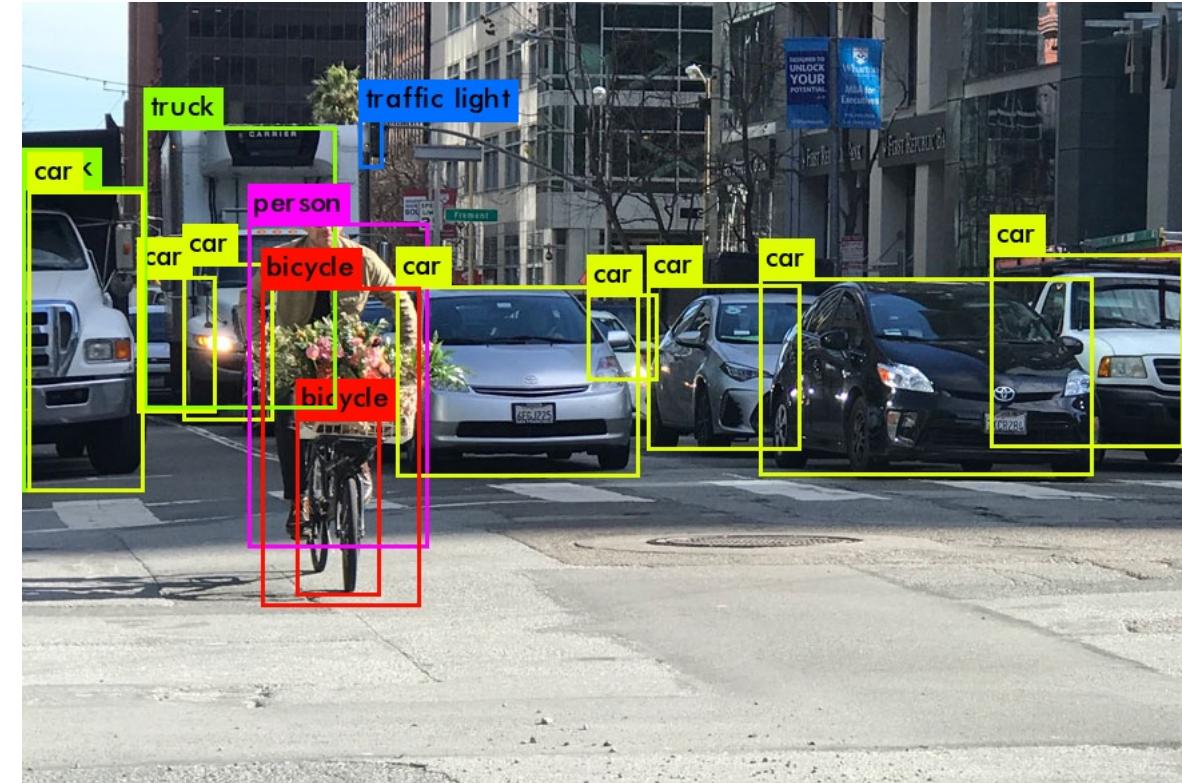


CAT

Single object

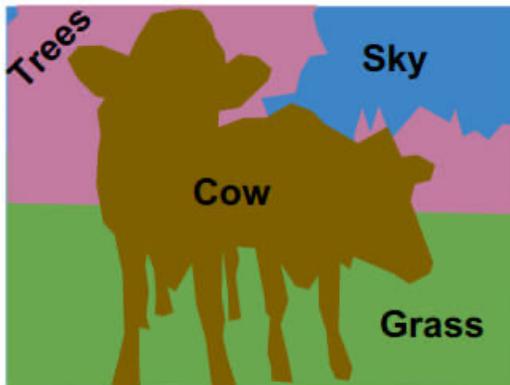
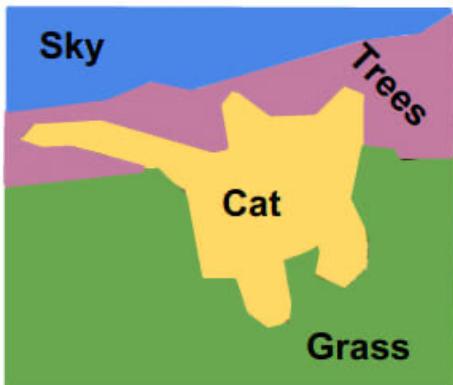
Object detection

- Hitta objekten (flera) i bilden med en **bounding box** samt klassificera dem.
- Kan hitta flera objekt av olika eller samma klass.



Semantic Segmentation

- Bestämmer klasstillhörighet för varje pixel



Varianter på segmentering



(a) Image



(b) Semantic Segmentation



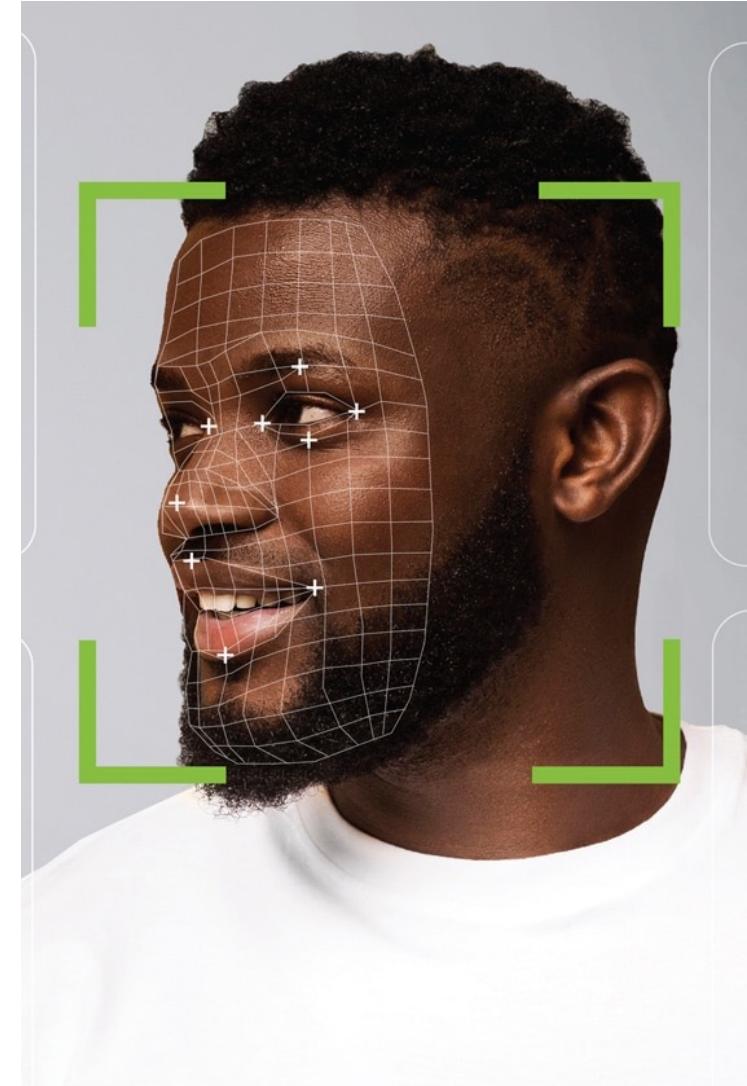
(c) Instance Segmentation



(d) Panoptic Segmentation

Applicerings exempel

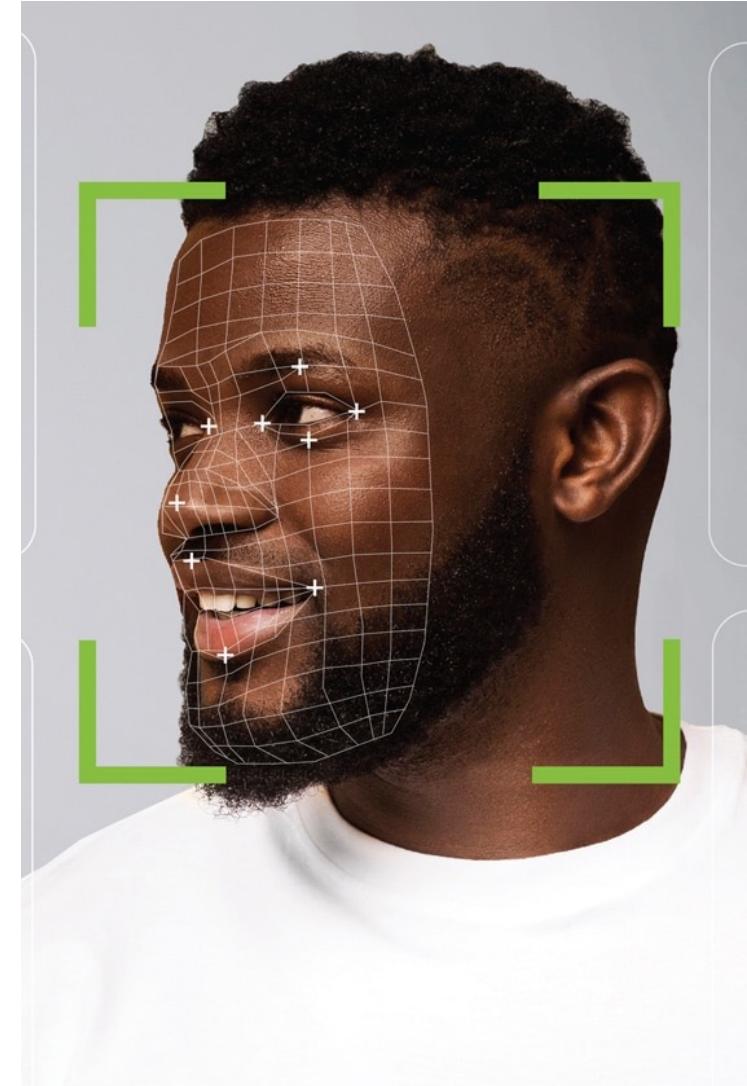
Face recognition - en samling tekniker.



Appliceringar exempel

Face recognition - en samling tekniker.

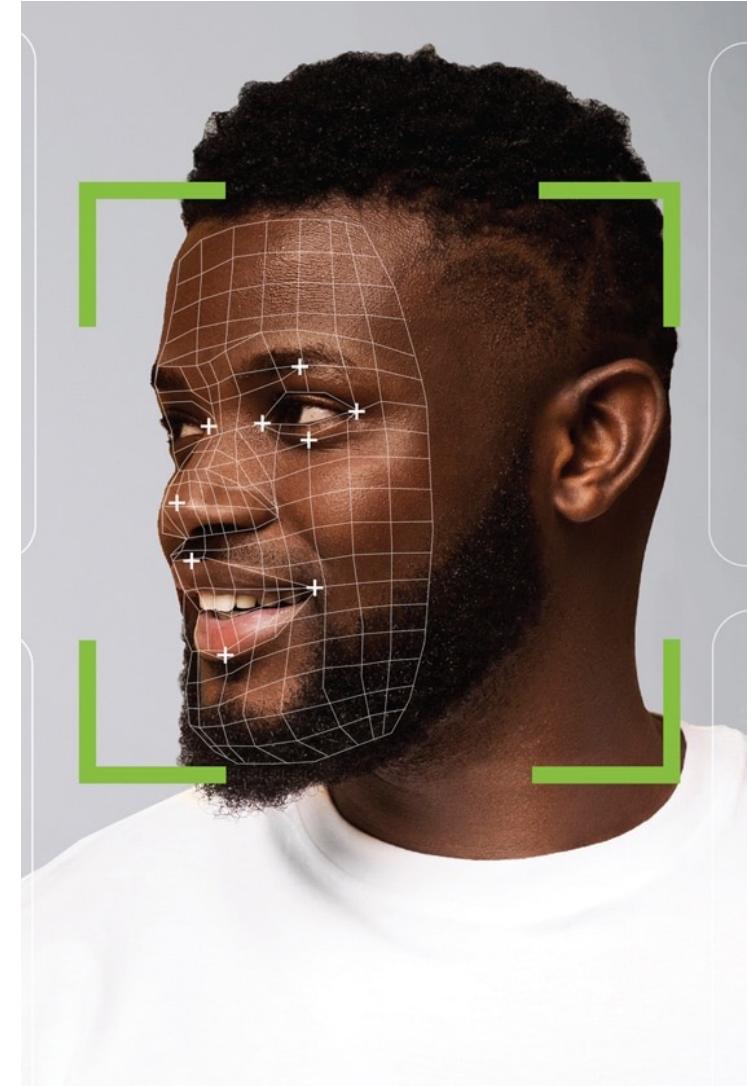
- Ansiktsdetektering
 - Finns ett ansikte i bilden?



Applicerings exempel

Face recognition - en samling tekniker.

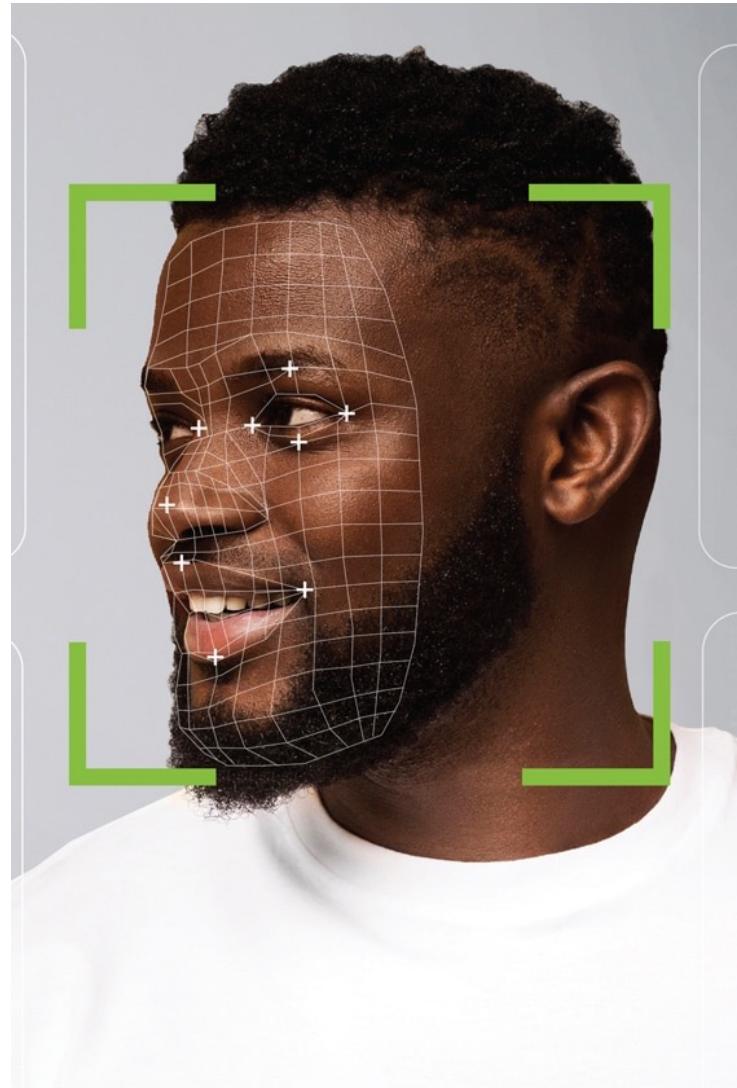
- Ansiktsdetektering
 - Finns ett ansikte i bilden?
- Analys av ansiktet
 - T.ex. hitta punkter i ansiktet och mäta avstånd.



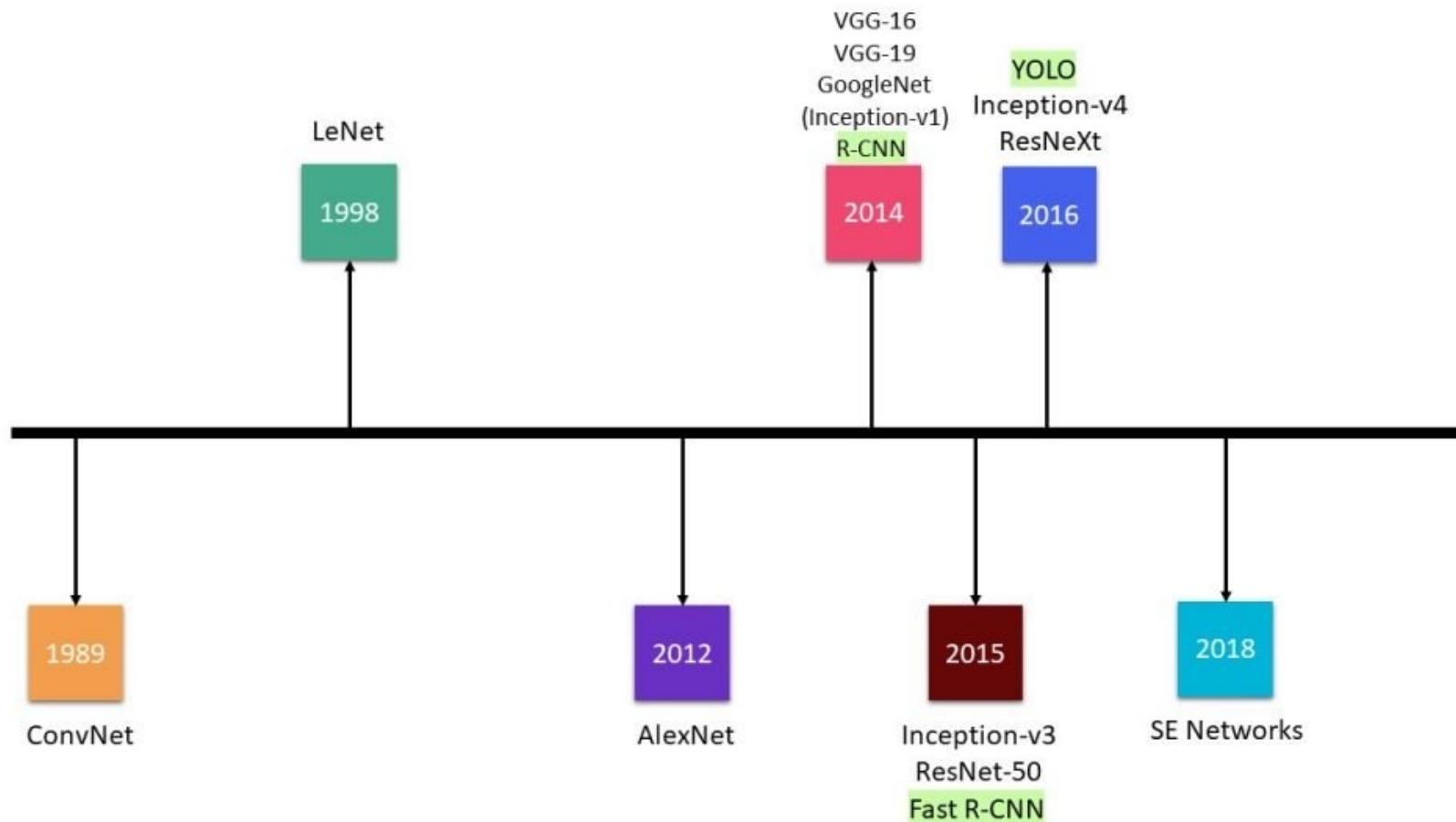
Applicerings exempel

Face recognition - en samling tekniker.

- Ansiktsdetektering
 - Finns ett ansikte i bilden?
- Analys av ansiktet
 - T.ex. hitta punkter i ansiktet och mäta avstånd.
- Konvertera data till vektor-rymd
 - T.ex. m.h.a. en autoencoder
- Hitta match
 - Finns en inlärd vektor (en känd person) som ligger nära den vi ser nu?



Kända CNNs



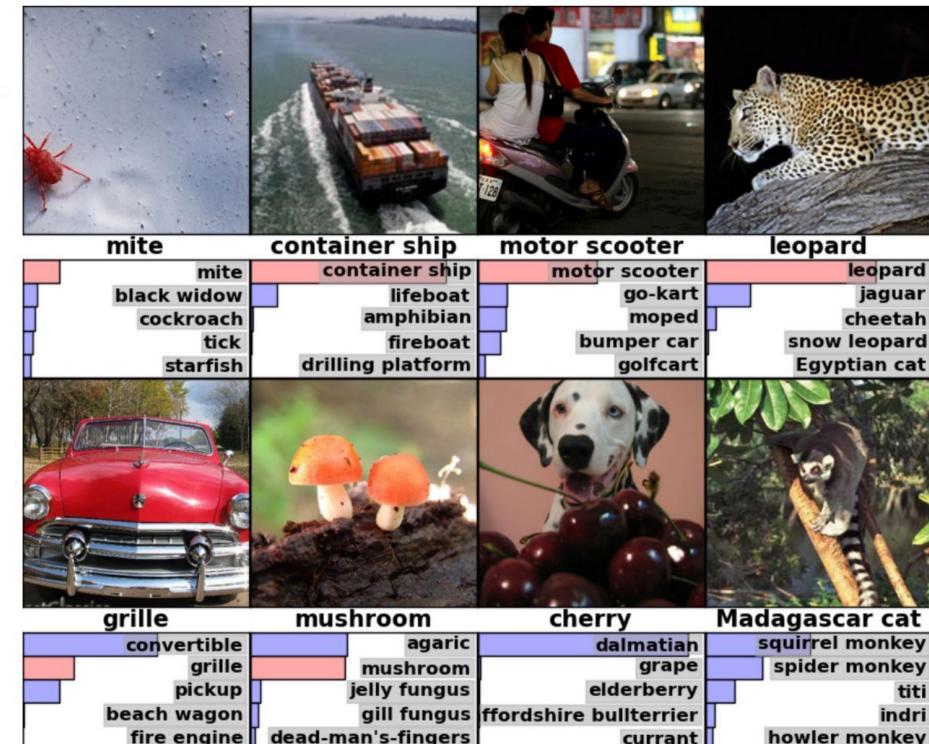
CNN historia

- Ca 2010 var detta ett superaktuellt problem.
- Imagenet project
- Annoterat en massa data som kunde användas för träning samt utvärdering.

ImageNet Challenge

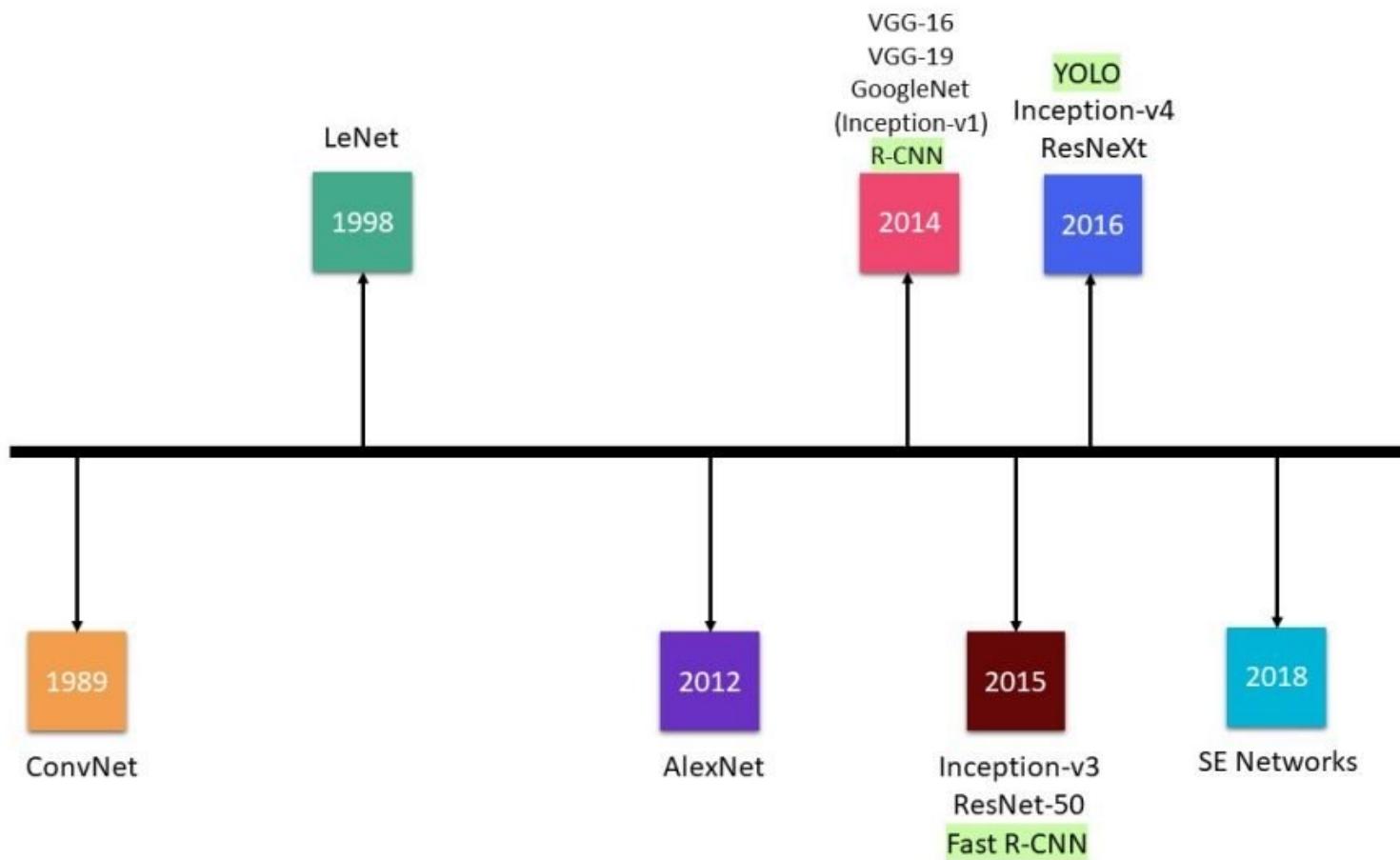
IMAGENET

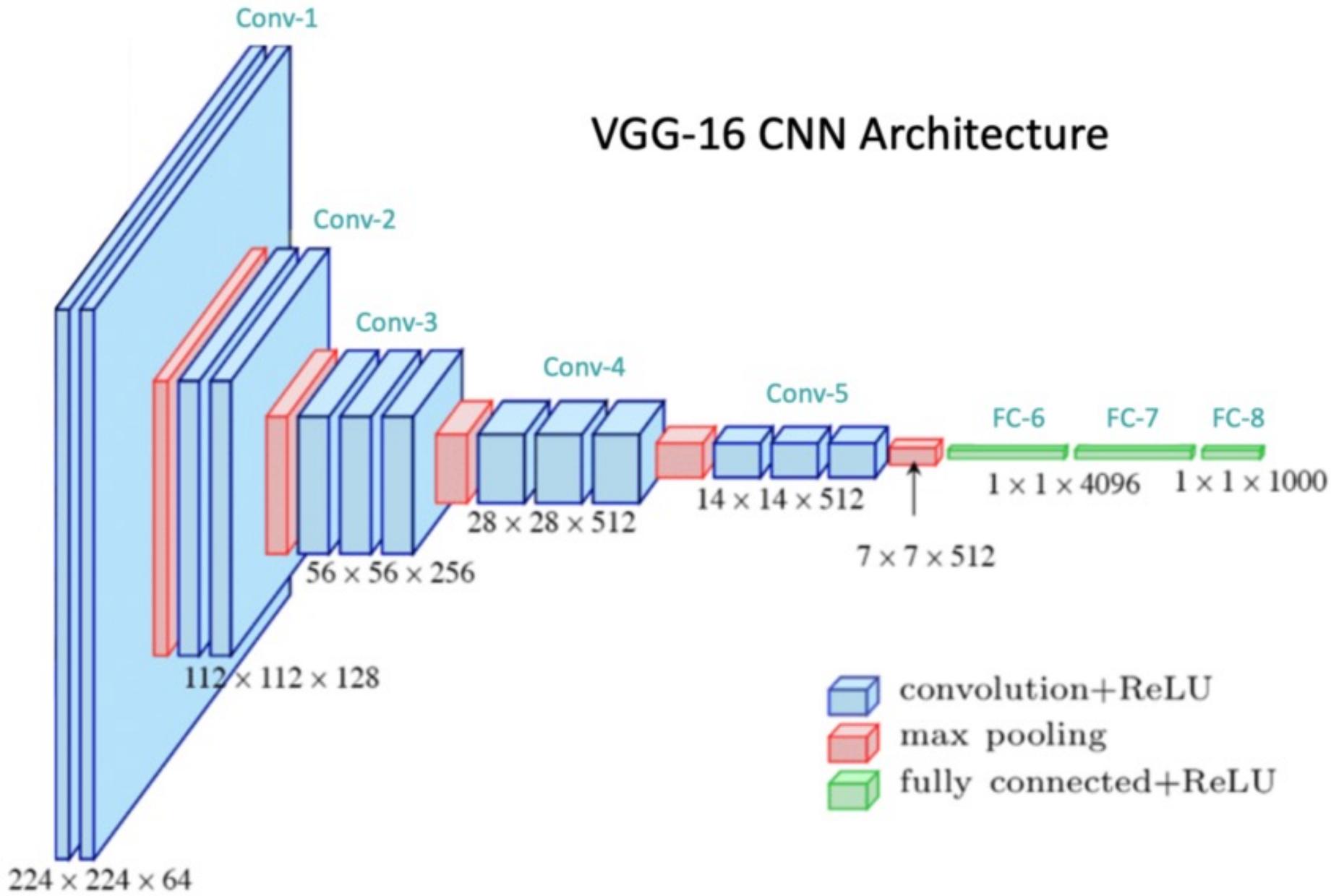
- 1,000 object classes (categories).
- Images:
 - 1.2 M train
 - 100k test.



Kända CNNs

- AlexNet - först!
- VGG-16
- ResNet
- YOLO
 - snabb object detection



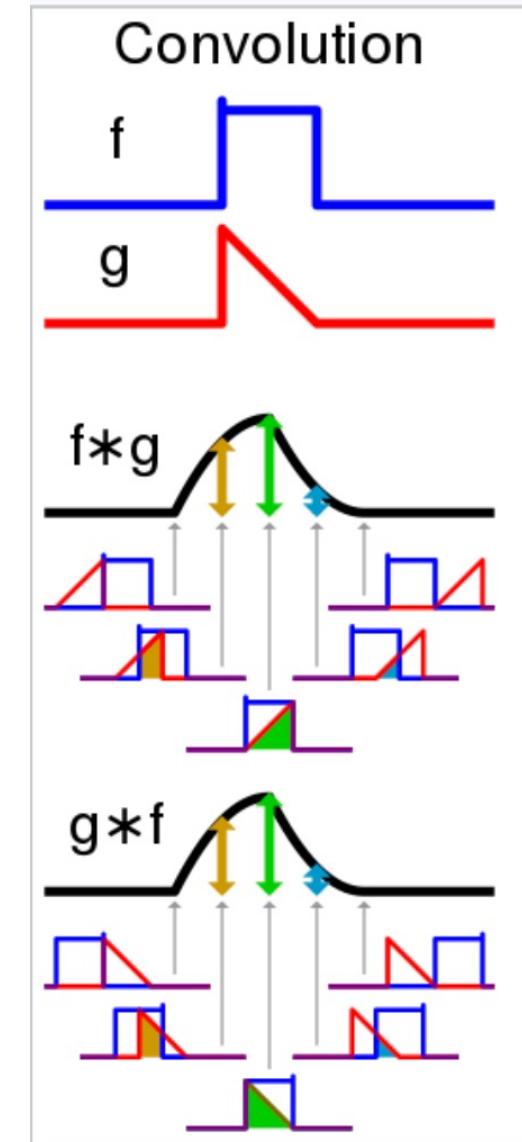


Introduktion

- Convolutional neural networks ("Faltande neurala nätverk") är neurala nätverk där minst 1 lager är ett convolutional layer.
- CNNs är väldigt bra på att hantera spatial data, t.ex. bilder.

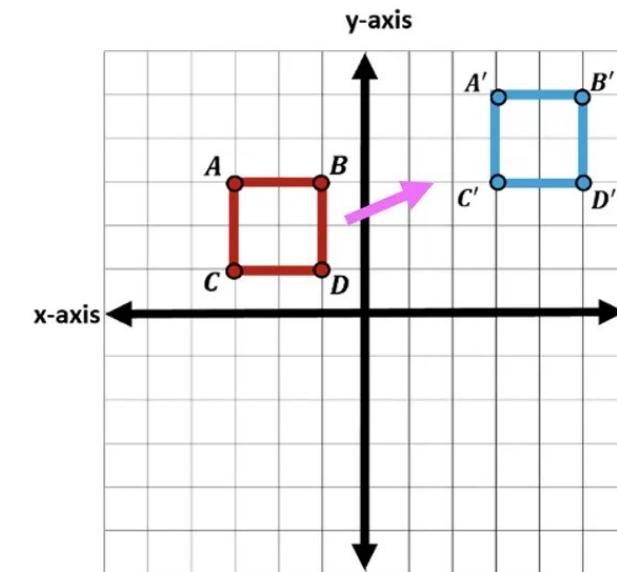
Convolution

- En "convolution" (faltung) är en matematisk funktion där 2 signaler kombineras för att skapa en tredje signal.
- Ett CNN använder då, på minst 1 ställe, en faltung istället för en matrismultiplikation.



Vad är så bra med CNN?

- Effektivare nätverk
 - Använder mindre minne samtidigt som att beräkningstiden inte påverkas
- Kan hantera små förändringar (t.ex. translationer)
- **Dynamisk input storlek!**



Convolutional layer

- Inputen multipliceras med "kernel":n och skapar en ny signal ("feature map")

2	4	9	1	4
2	1	4	4	6
1	1	2	9	2
7	3	5	1	3
2	3	4	8	5

Image

x

1	2	3
-4	7	4
2	-5	1

Filter /
Kernel

=

51		

Feature

Convolutional layer

Input

4	9	2	5	8	3
5	6	2	4	0	3
2	4	5	4	5	2
5	6	5	4	7	8
5	7	7	9	2	1
5	8	5	3	8	4

$$n_H \times n_W = 6 \times 6$$

Filter

1	0	-1
1	0	-1
1	0	-1

*

Parameters:

Size: $f = 3$

Stride: $s = 1$

Padding: $p = 0$

Result

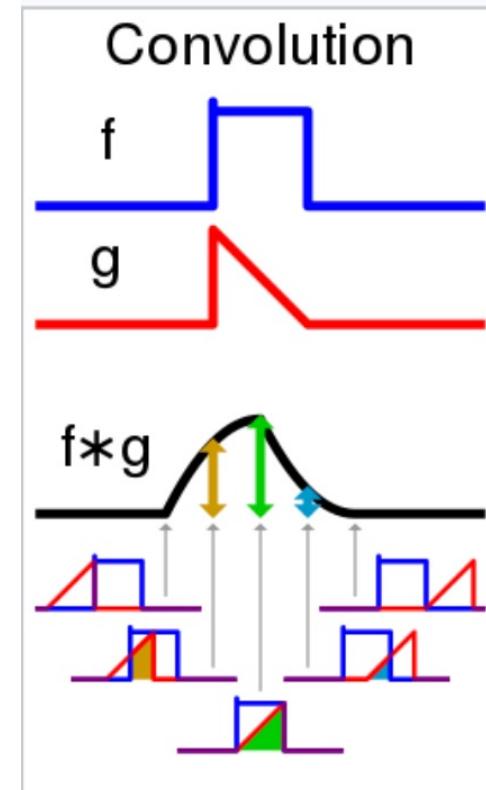
2			

$$\boxed{2} = 4*1 + 9*0 + 2*(-1) + \\ 5*1 + 6*0 + 2*(-1) + \\ 2*1 + 4*0 + 5*(-1)$$

<https://indoml.com>

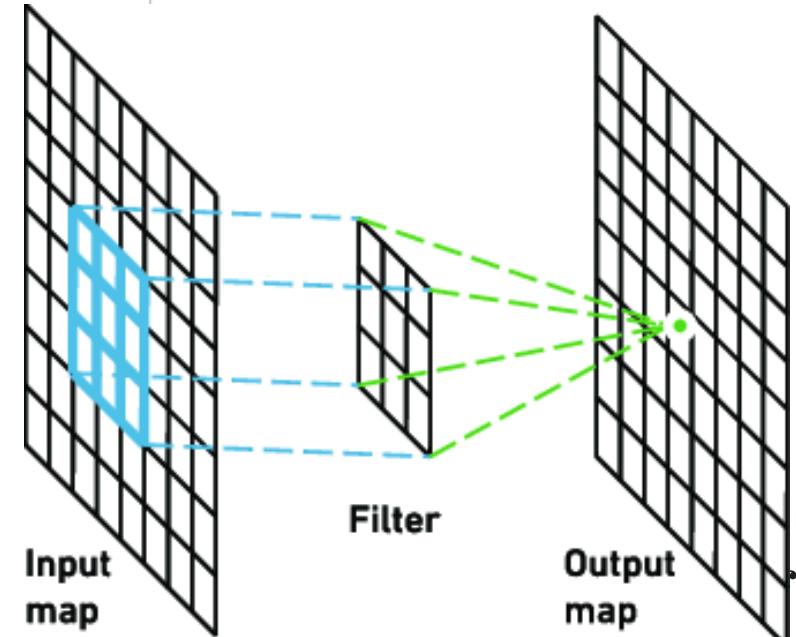
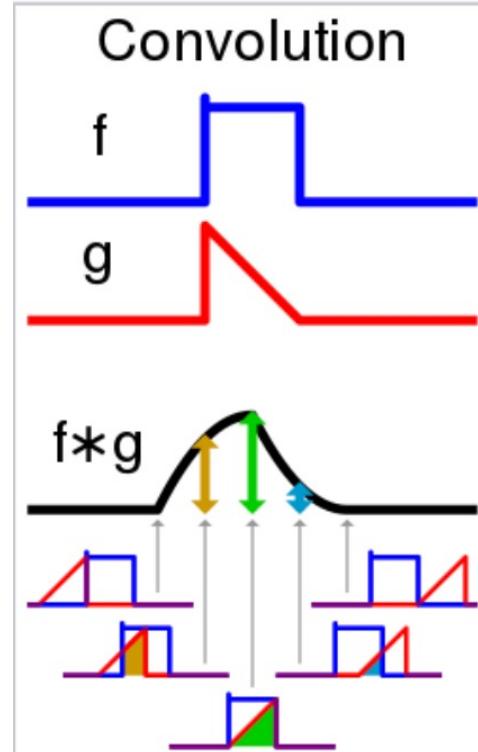
Kernel / filter

- I faltningen är inputen f och kerneln är g .



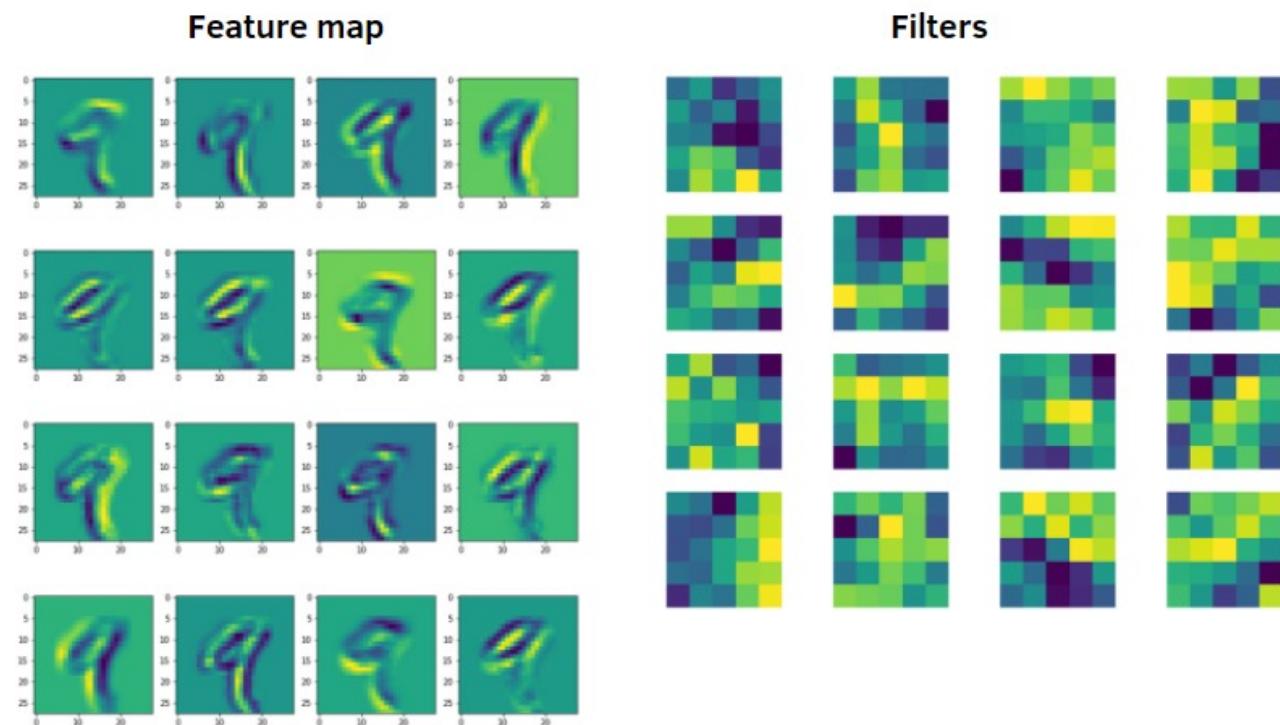
Kernel / filter

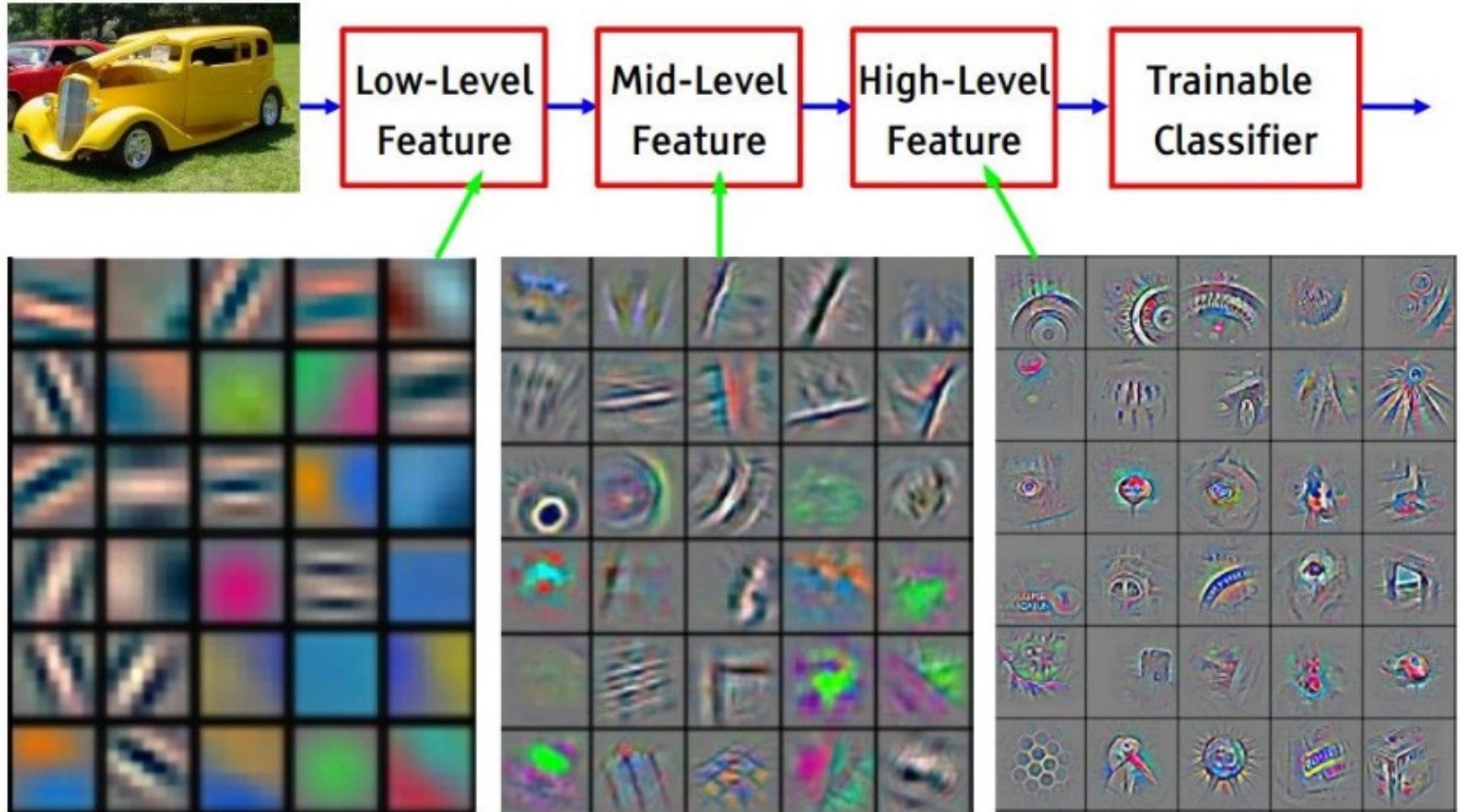
- I faltningen är inputen f och kerneln är g .
- Kerneln kan ha olika storlek, men brukar vara kvadratisk.
- Kerneln "går" över bilden och skapar en **feature map** (activation map), vilken sen blir inputen till nästa lager.



Feature map

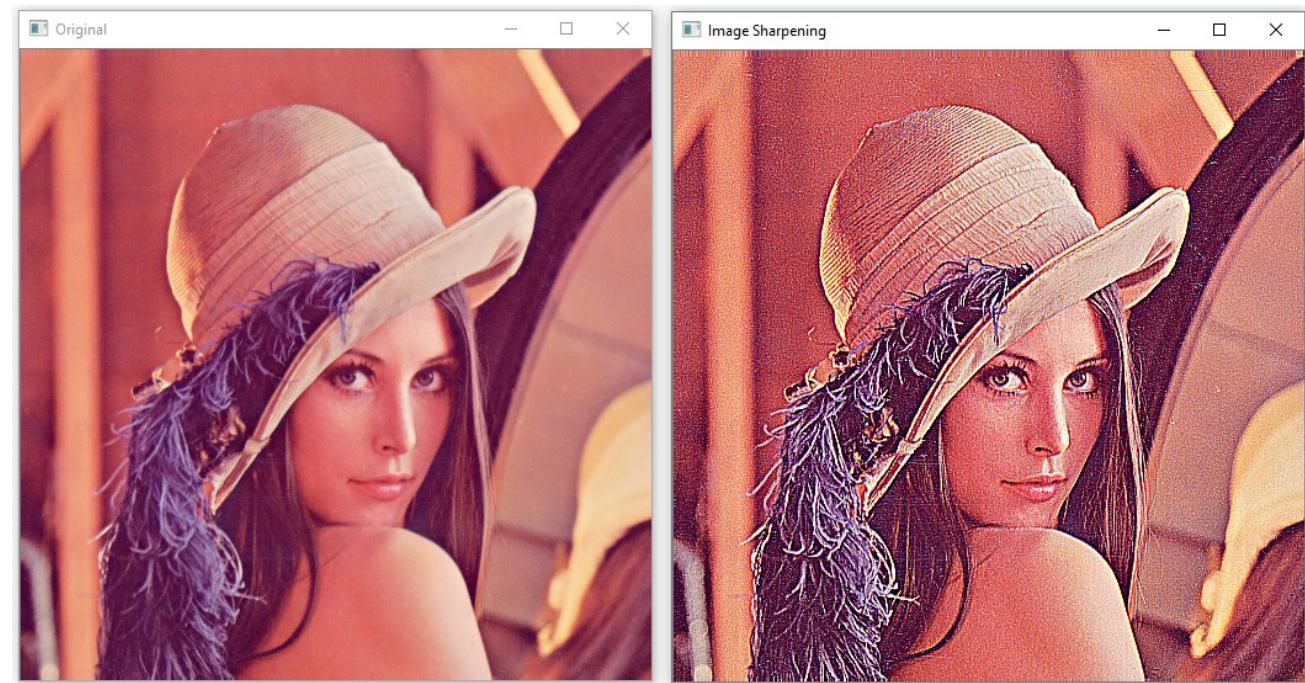
- Outputen från ett convolutional lager
- Representerar **särdrag** i datan, t.e.x hittar ett filter bara horisontella kanter
- Flera olika kernels kan användas i samma lager för att hitta olika särdrag

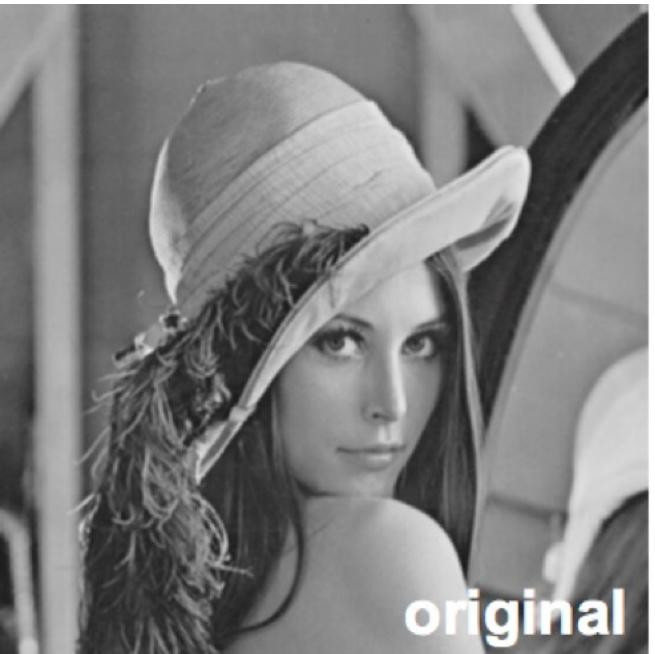




Filter från computer vision

- I klassisk signalbehandling används filter.
- T.ex. blur eller sharpening

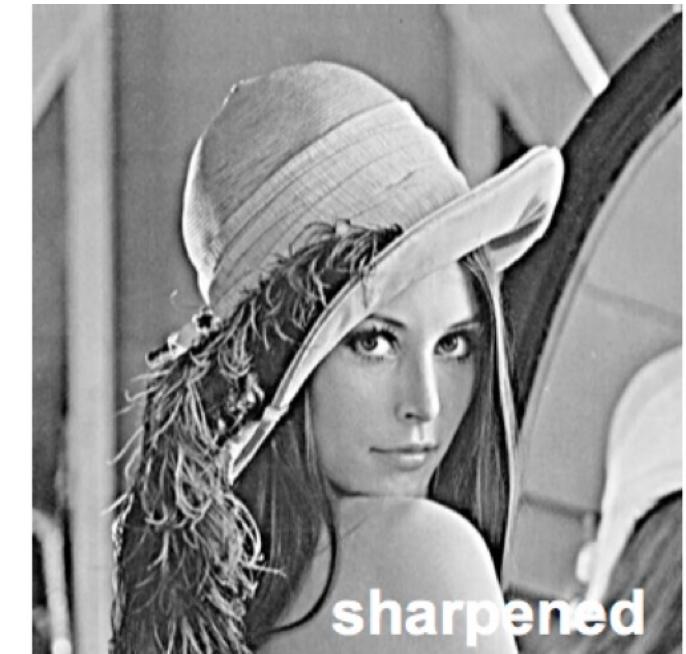




+ a



=



•0	•0	•0
•0	•1	•0
•0	•0	•0

+

•0	•0	•0
•0	•1	•0
•0	•0	•0

$$- \frac{1}{9}$$

•1	•1	•1
•1	•1	•1
•1	•1	•1

=

•0	•0	•0
•0	•2	•0
•0	•0	•0

•1	•1	•1
•1	•1	•1
•1	•1	•1

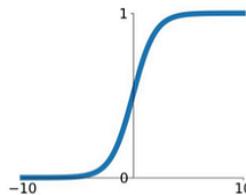
Aktiveringsfunktion för conv lager

- ReLU funkar ofta bra för CNN

Activation Functions

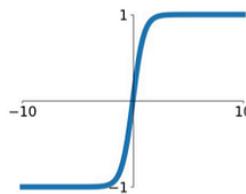
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



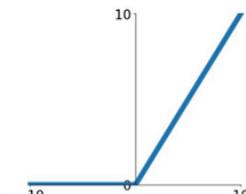
tanh

$$\tanh(x)$$



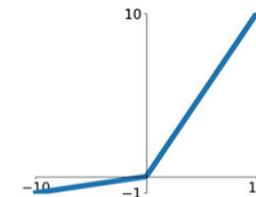
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

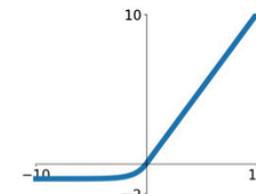


Maxout

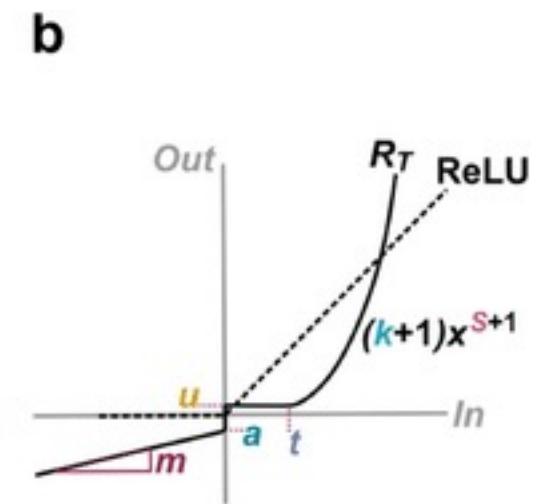
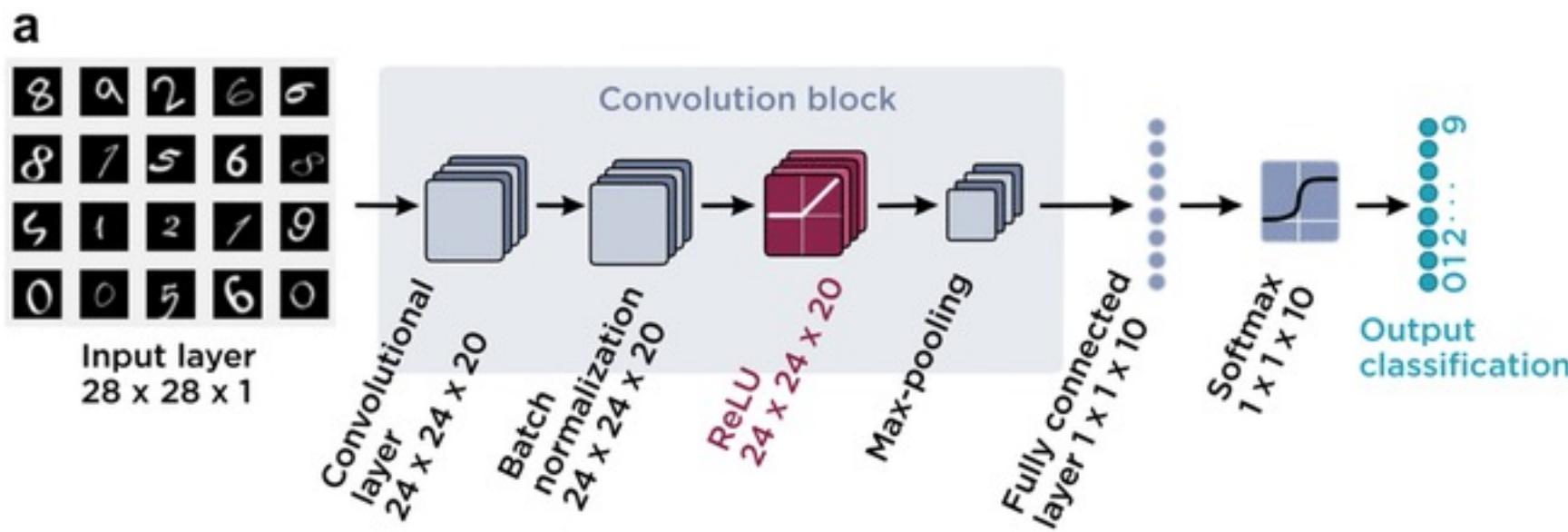
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

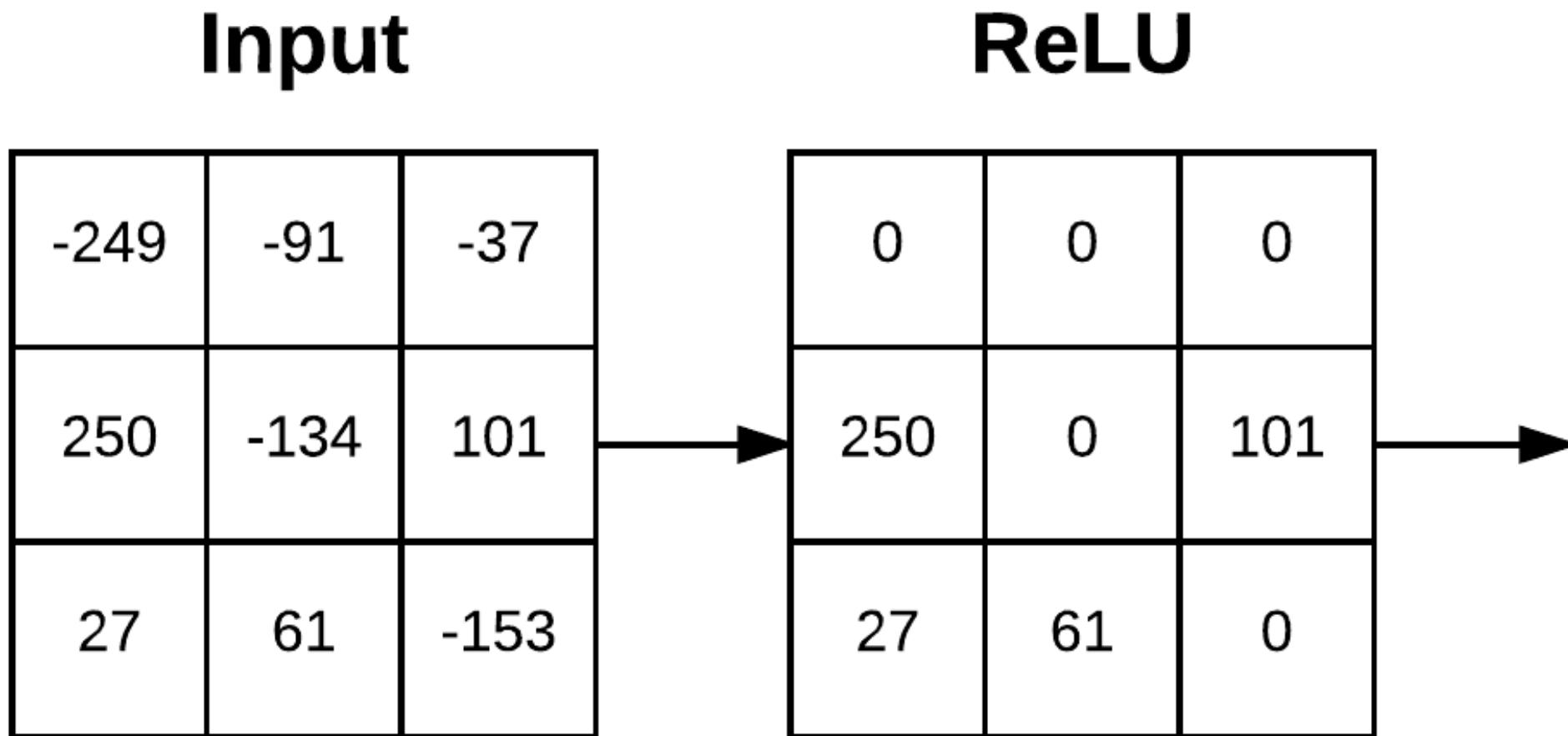
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation function



Activation function



Pooling layer

- Pooling används (ofta) efter ett convolutional lager.

Pooling layer

- Pooling används (ofta) efter ett convolutional lager.
- Pooling hjälper med att göra att representationerna kan hantera mot små translationer.
 - I stort sett så ”zoomar vi ut” lite. Är det en skillnad på ett par pixlar då kommer vi inte märka det.

Pooling layer

- Pooling används (ofta) efter ett convolutional lager.
- Pooling hjälper med att göra att representationerna kan hantera mot små translationer.
 - I stort sett så ”zoomar vi ut” lite. Är det en skillnad på ett par pixlar då kommer vi inte märka det.
- Det reducerar dimensionerna på feature maps samtidigt som den viktigaste datan behålls.
 - Vi vill oftast reducera dimensionerna, t.ex. gör du binär klassificering är det ju bara 1 neuron som är output, då bör vi reducera dimensionerna gradvis genom hela nätet.

Max pooling och avg pooling

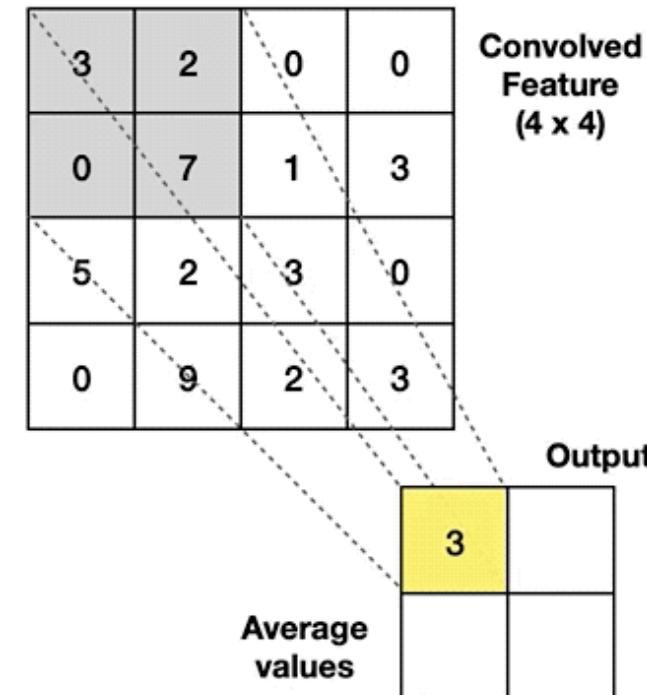
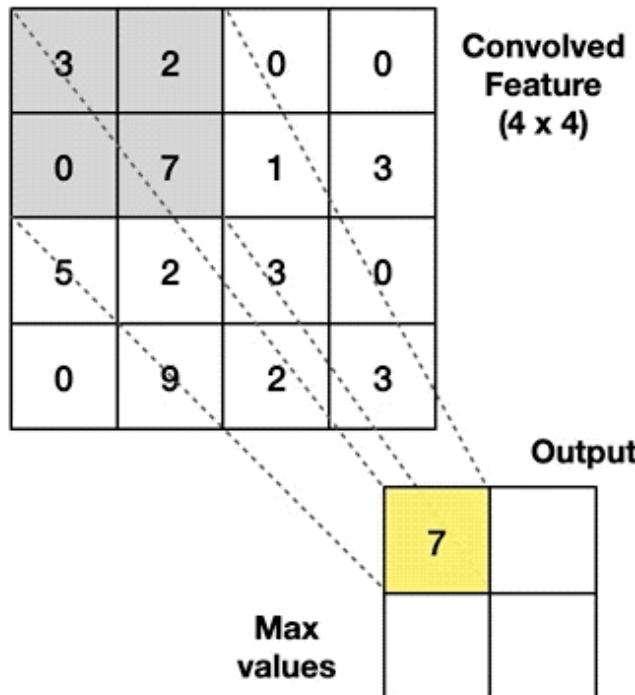
Max Pooling

Take the **highest** value from the area covered by the kernel

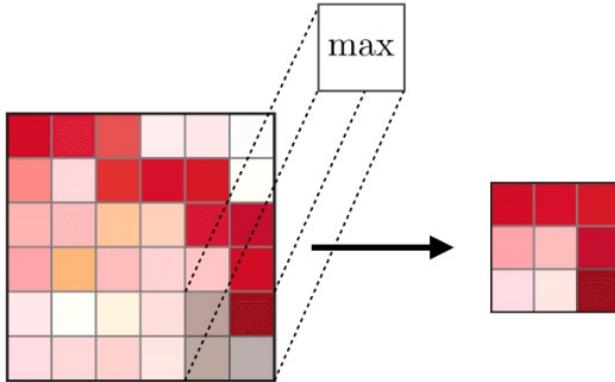
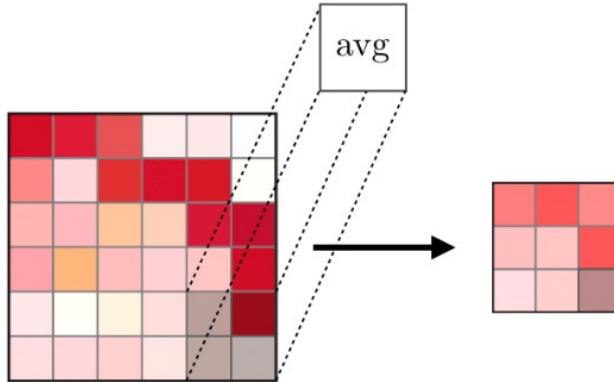
Average Pooling

Calculate the **average** value from the area covered by the kernel

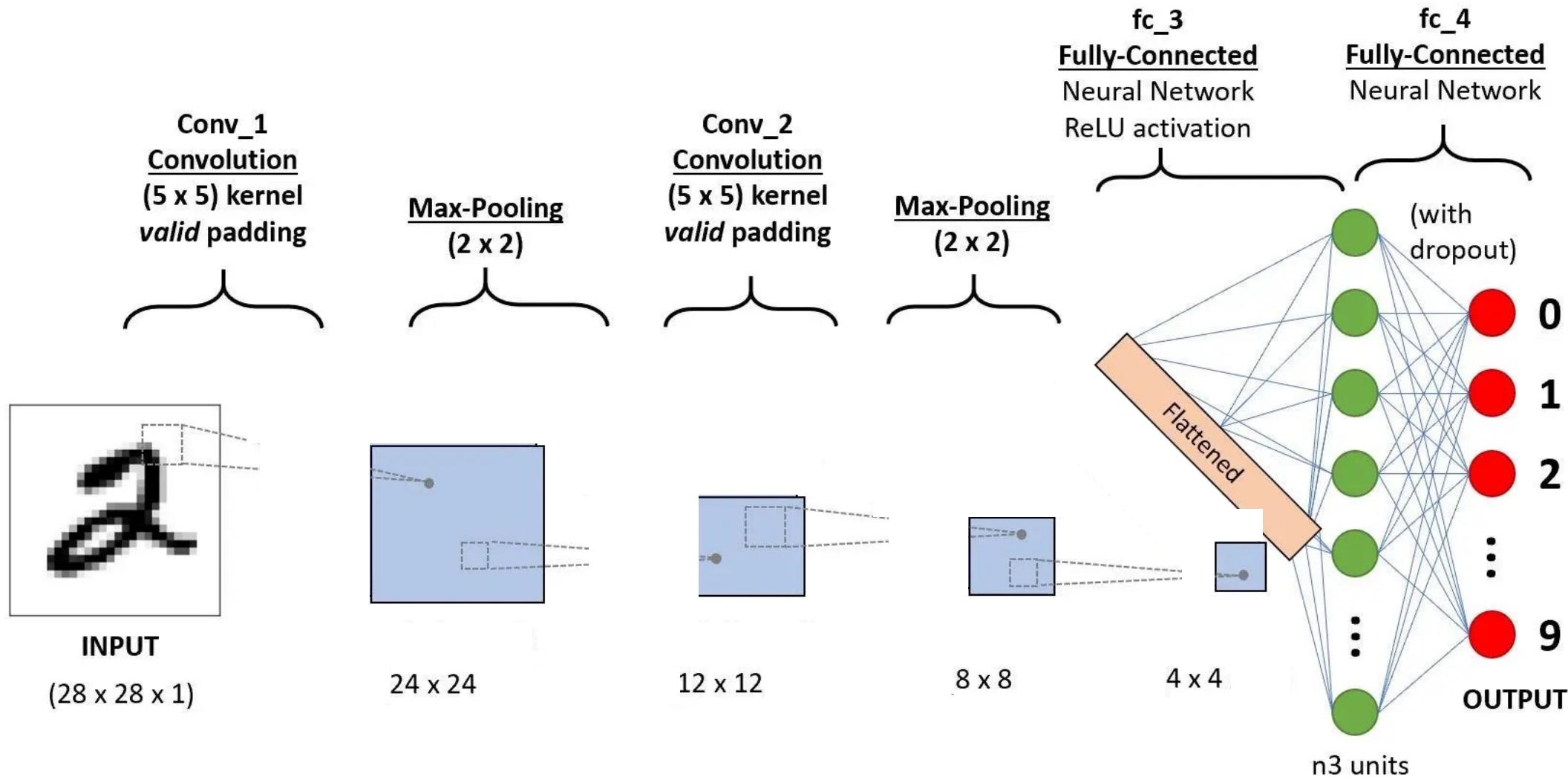
Example: Kernel of size 2×2 ; stride=(2,2)



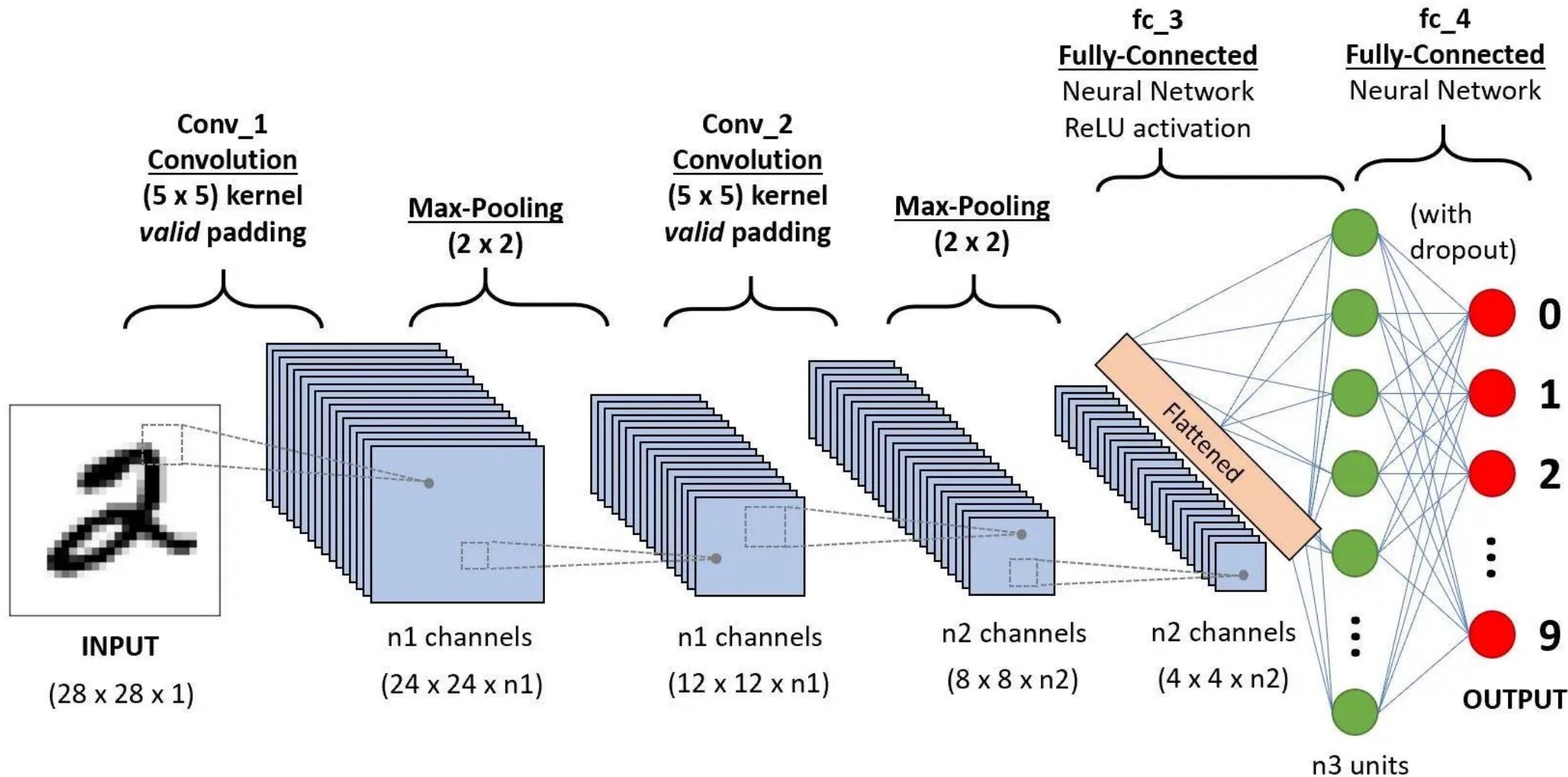
Max pooling och avg pooling

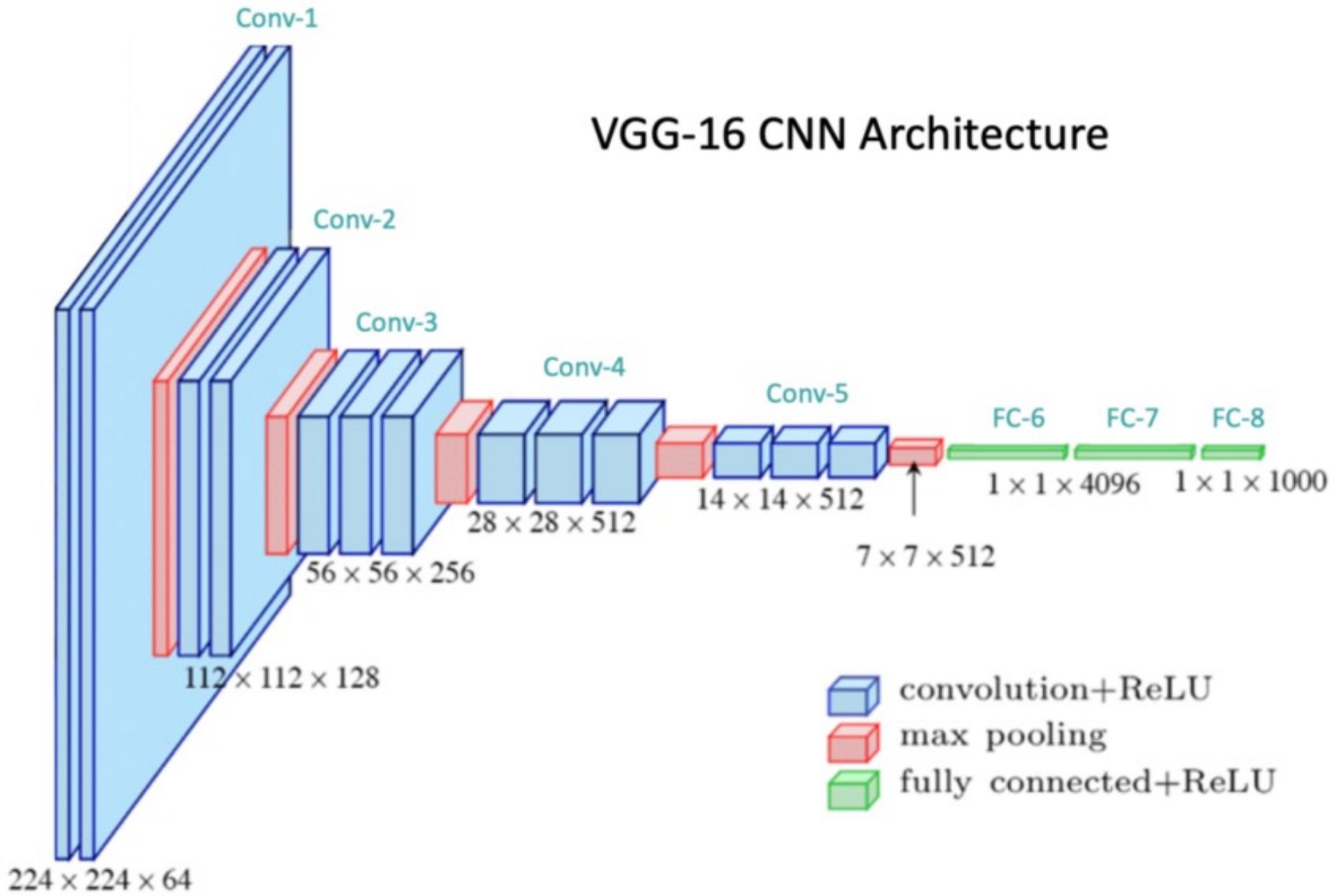
Type	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	<ul style="list-style-type: none">• Preserves detected features• Most commonly used	<ul style="list-style-type: none">• Downsamples feature map• Used in LeNet

Nu sätter vi ihop det!



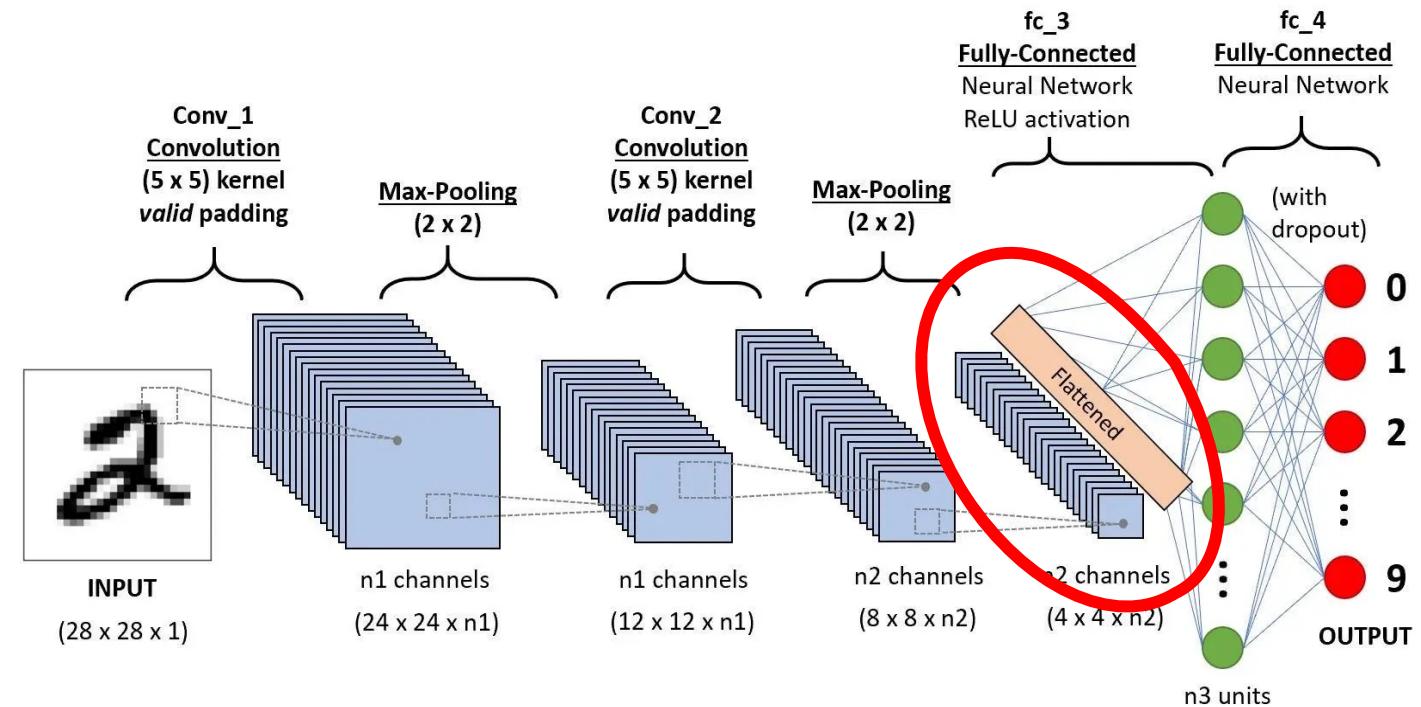
Nu sätter vi ihop det!

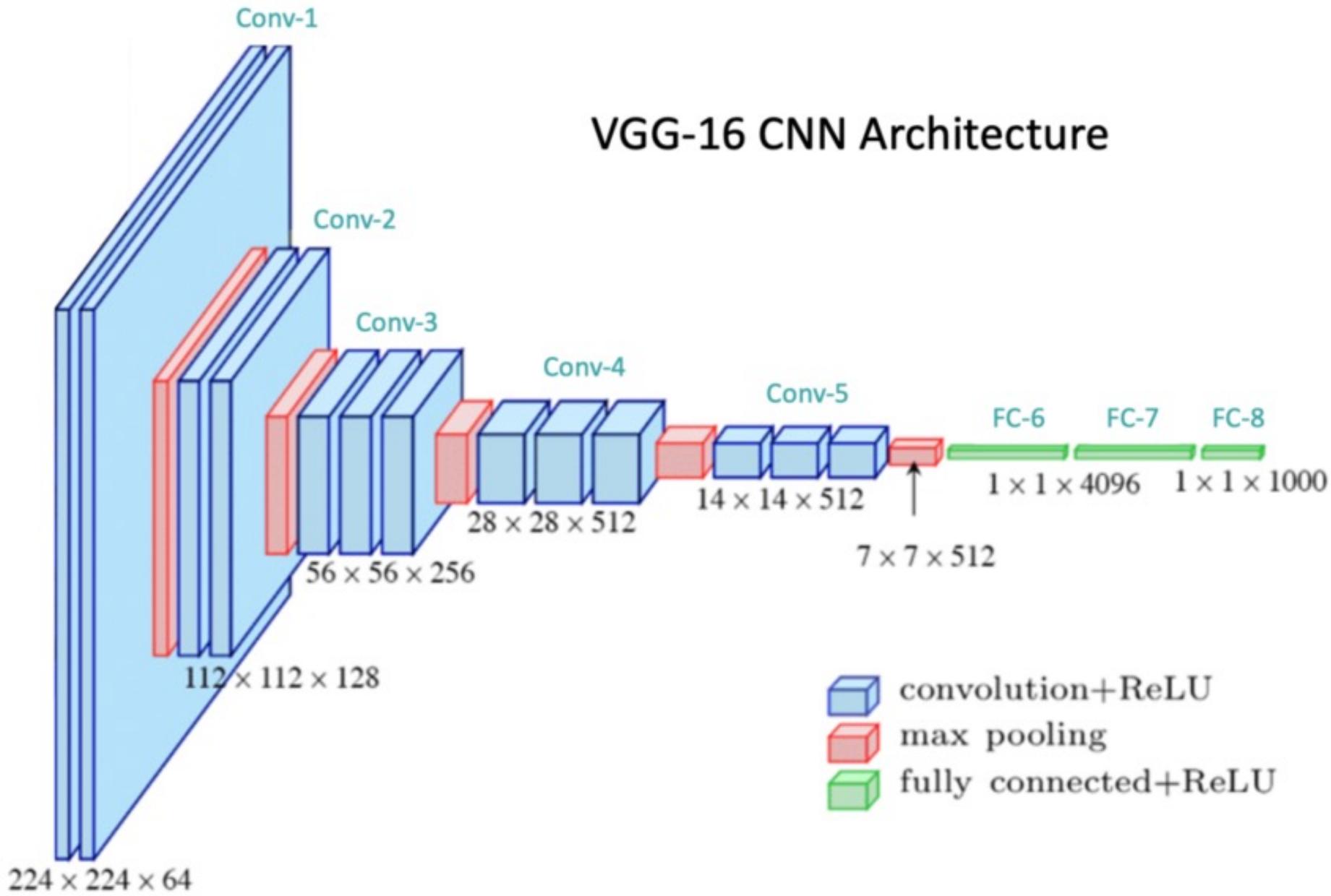




Från convolutions till fully connected

- "Flatten network"
- Ta alla data i feature maps och lägg dem i en rad
- Voilá - nu är det ett fully connected nätverk





Vad är så bra med CNNs?

- Effektivare nätverk
 - Samma vikter (kernel) används över hela input datan. I ett FCNN så används en vikt en gång, här så ”återanvänder” vi vikterna när kerneln rör sig över bilden.
 - Kerneln är mycket mindre än bilden och är också en anledning att vi använder färre parametrar.
- Kan hantera små förändringar (t.ex. translationer) i datan.
- **Dynamisk input storlek!!**

Hur mycket mindre är ett conv lager?

- Tänk att vi har en bild som är 150x150 pixlar.
- Vi vill att nästa lager ska ha 100 neuroner.

Hur mycket mindre är ett conv lager?

- Tänk att vi har en bild som är 150x150 pixlar.
- Vi vill att nästa lager ska ha 100 neuroner.
- $150 \times 150 \times 100 = 2\ 250\ 000$ vikter.

Hur mycket mindre är ett conv lager?

- Tänk att vi har en bild som är 150x150 pixlar.
 - Vi vill att nästa lager ska ha 100 neuroner.
 - $150 \times 150 \times 100 = 2\ 250\ 000$ vikter.
-
- Vi använder istället ett conv lager. Låt oss säga 64 st 9×9 filter.
 - (Vår output feature map blir då 141×141)

Hur mycket mindre är ett conv lager?

- Tänk att vi har en bild som är 150x150 pixlar.
- Vi vill att nästa lager ska ha 100 neuroner.
- $150 \times 150 \times 100 = 2\ 250\ 000$ vikter.
- Vi använder istället ett conv lager. Låt oss säga 64 st 9x9 filter.
- (Vår output feature map blir då 141x141)
- $64 \times 9 \times 9 = 5184$ vikter

Hur mycket mindre är ett conv lager?

- Tänk att vi har en bild som är 150x150 pixlar.
 - Vi vill att nästa lager ska ha 100 neuroner.
 - $150 \times 150 \times 100 = 2\ 250\ 000$ vikter.
-
- Vi använder istället ett conv lager. Låt oss säga 100x100 filter.
 - (Vår output feature map blir då 50x50)

Hur mycket mindre är ett conv lager?

- Tänk att vi har en bild som är 150x150 pixlar.
- Vi vill att nästa lager ska ha 100 neuroner.
- $150 \times 150 \times 100 = 2\ 250\ 000$ vikter.

- Vi använder istället ett conv lager. Låt oss säga 100x100 filter.
- (Vår output feature map blir då 50x50)
- $100 \times 100 = 10\ 000$ vikter (i filtret).
- Vi kan använda oss av **225** olika filter för att använda samma mängd minne. (och så många behöver man inte)

Hur mycket mindre är ett conv lager?

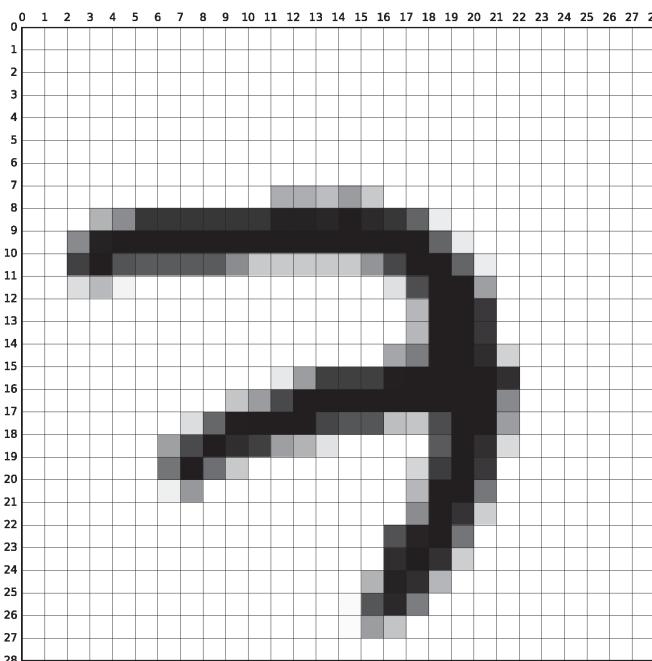
- Tänk att vi har en bild som är 150x150 pixlar.
- Vi vill att nästa lager ska ha 100 neuroner.
- $150 \times 150 \times 100 = 2\ 250\ 000$ vikter.

För stort

- Vi använder istället ett conv lager. Låt oss säga ~~100x100~~ filter.
- (Vår output feature map blir då 50x50)
- $100 \times 100 = 10\ 000$ vikter (i filtret).
- Vi kan använda oss av **225** olika filter för att använda samma mängd minne. (och så många behöver man inte)

Bilder som inputdata

- Vi börjar med svart-vita bilder
- Varje pixel har ett värde
- Ofta går värdenen på pixlarna mellan 0-255
- 0=svart, 255=vitt



(a) MNIST sample belonging to the digit '7'.



(b) 100 samples from the MNIST training set.

Normalisering

- Som med annan data för deep learning är det en bra idé att normalisera inputdatan.
- Pixelvärde / 255 = normaliserat pixelväde.