Convolutional Neural Networks

Guanlin Li

Nov. 9 2016

Outline

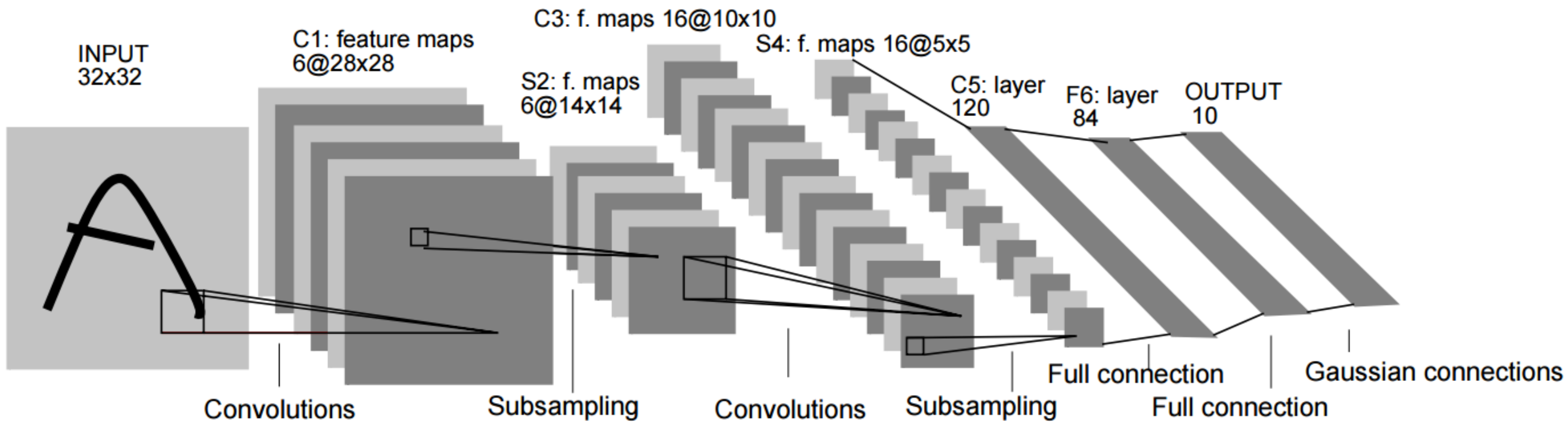
- Warming up
- Convolution
  - Filter, kernel
  - Deconvolution
- Pooling
  - Subsampling
  - Invariance
- Convolutional Neural Net
  - Architecture

Outline

- A Little bit Technical Warming up
- Convolution
  - Filter, kernel
  - Deconvolution
- Pooling
  - Subsampling
  - Invariance
- Convolutional Neural Net
  - Architecture

A Beginning Glimpse of CNN

- Modern CNN since Yann LeCun

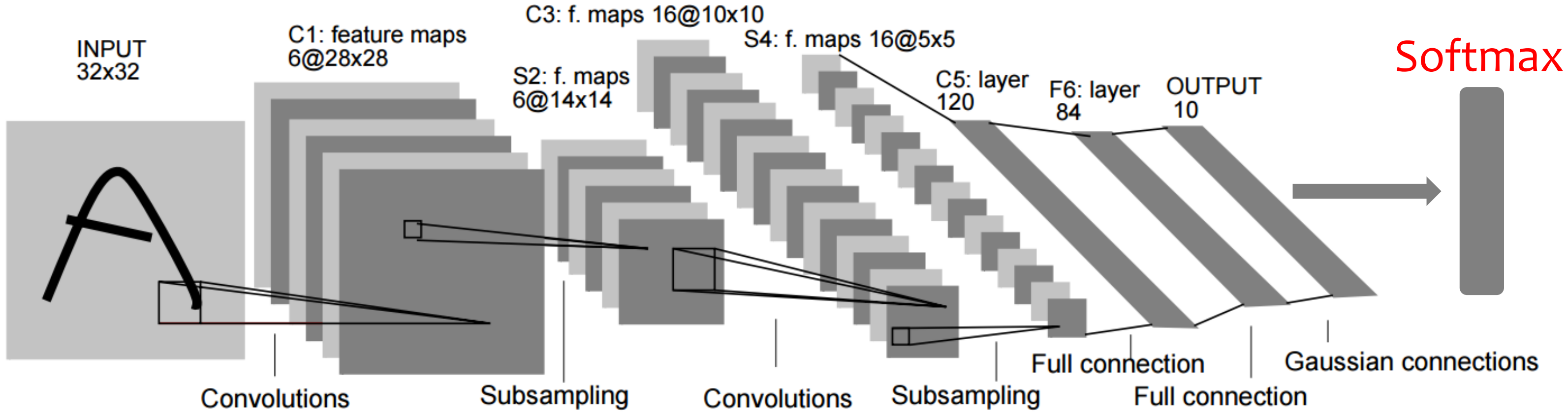


- "Gradient-based Learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns such as handwritten characters, with minimal preprocessing."
- "Convolutional neural networks, that are specifically designed to deal with the variability of 2D shapes are shown to outperform all other techniques."

Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

A Beginning Glimpse of CNN

- Modern CNN since Yann LeCun

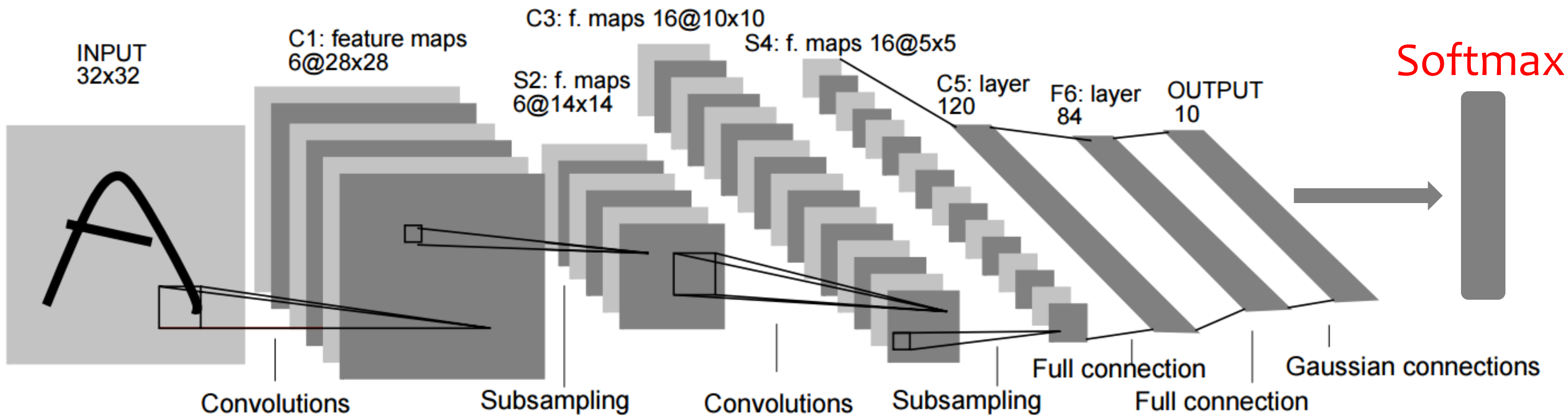


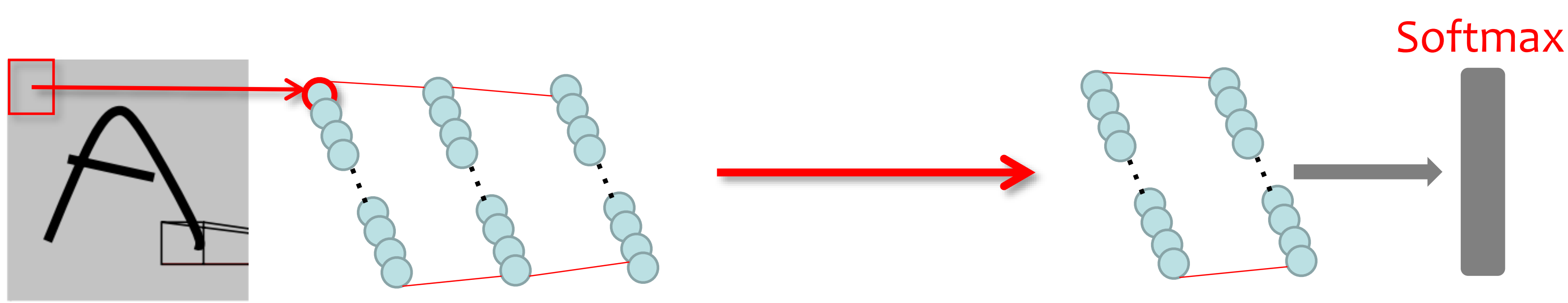
- As a classification problem, we are going to extract from the INPUT through the forward computation, and get a final feature vector, the OUTPUT.
- Then we can do *classification* upon this feature vector.

Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

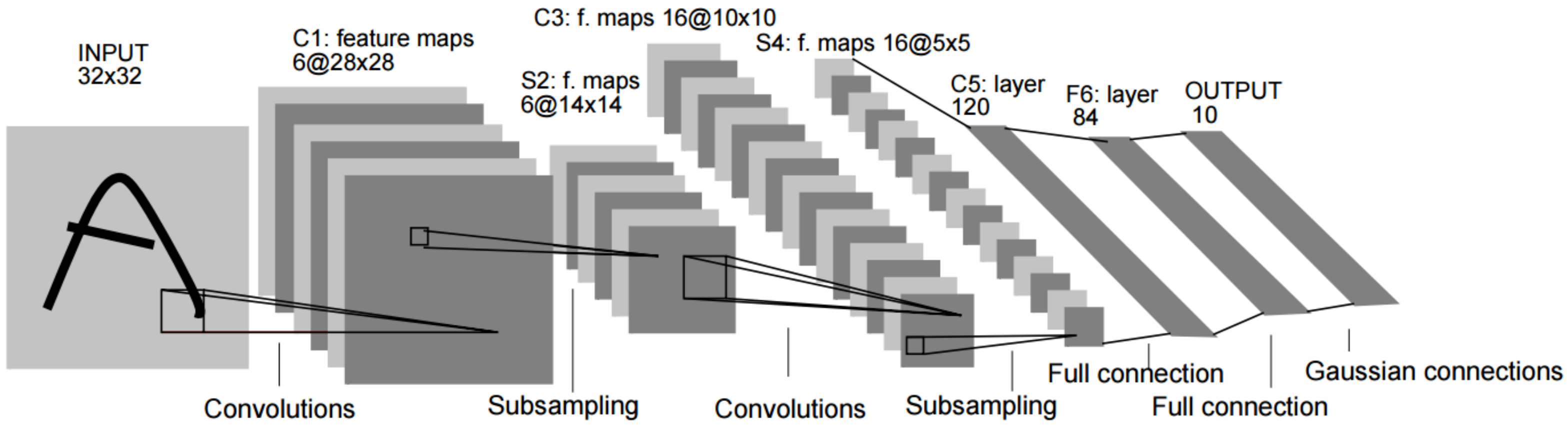
A Beginning Glimpse of CNN

- CNN is a kind of FFNN, with special design

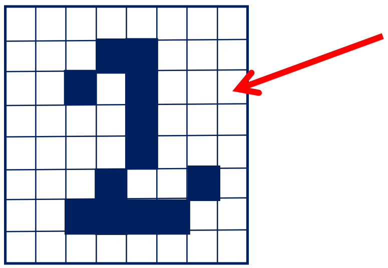




A Beginning Glimpse of CNN

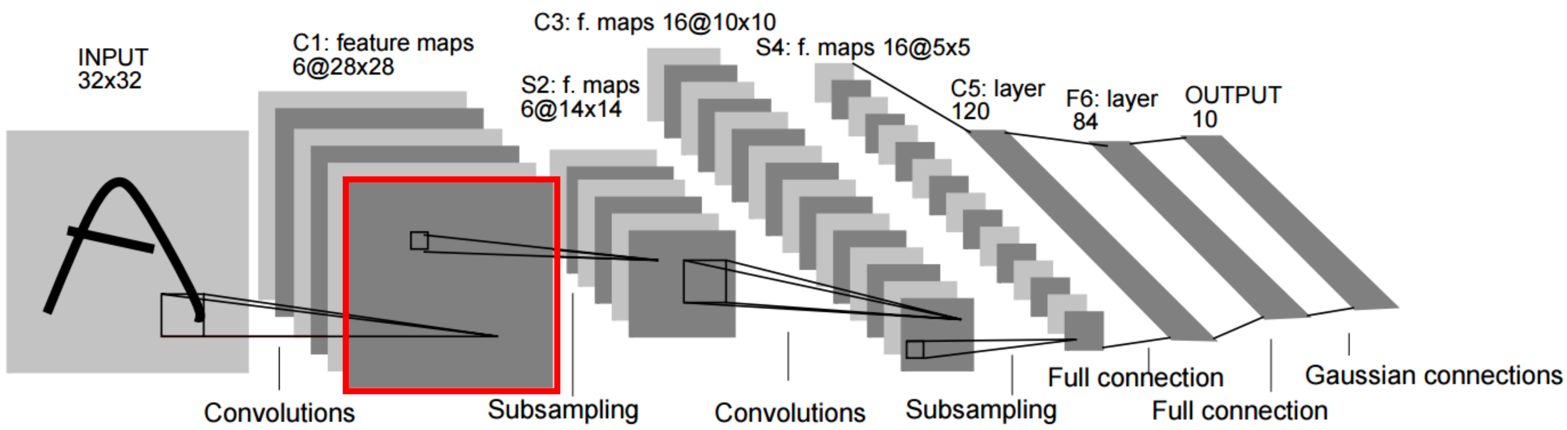


Now let us get into a little bit detail about how this work!

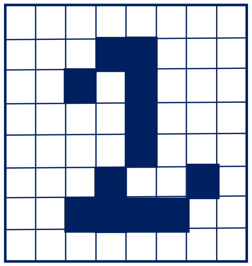


Every small grid with a real value

A Beginning Glimpse of CNN

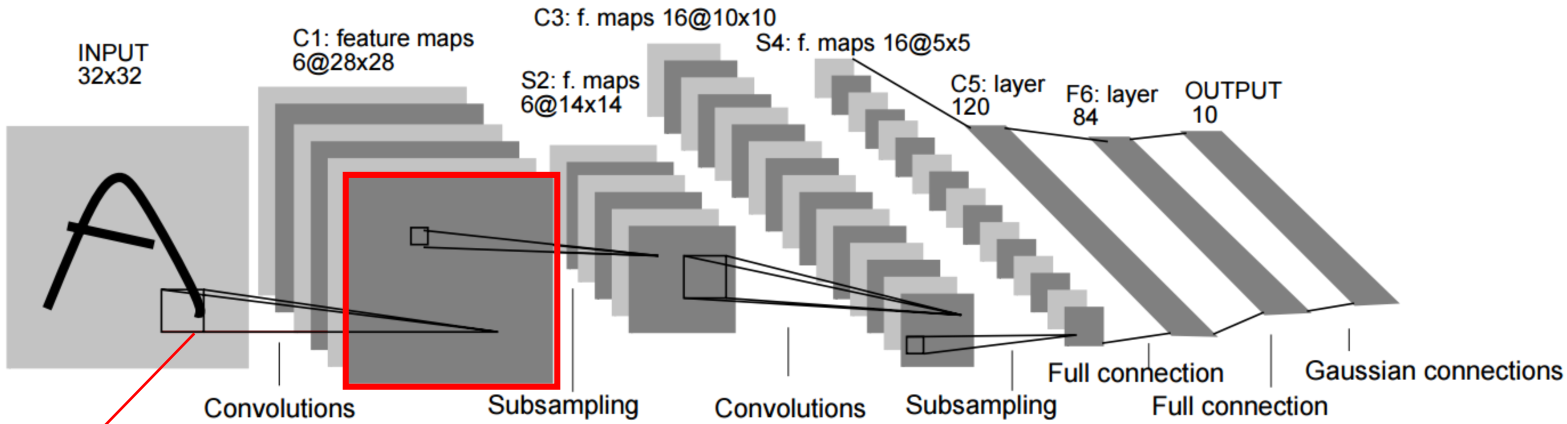


The red box is called a *feature map* after *convolution*, which is a matrix of real values.



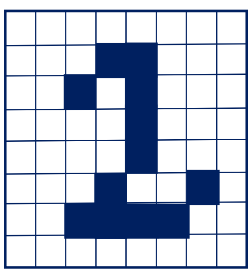
How we get it?

A Beginning Glimpse of CNN



We have a small window, say here 3×3, with real value parameters.

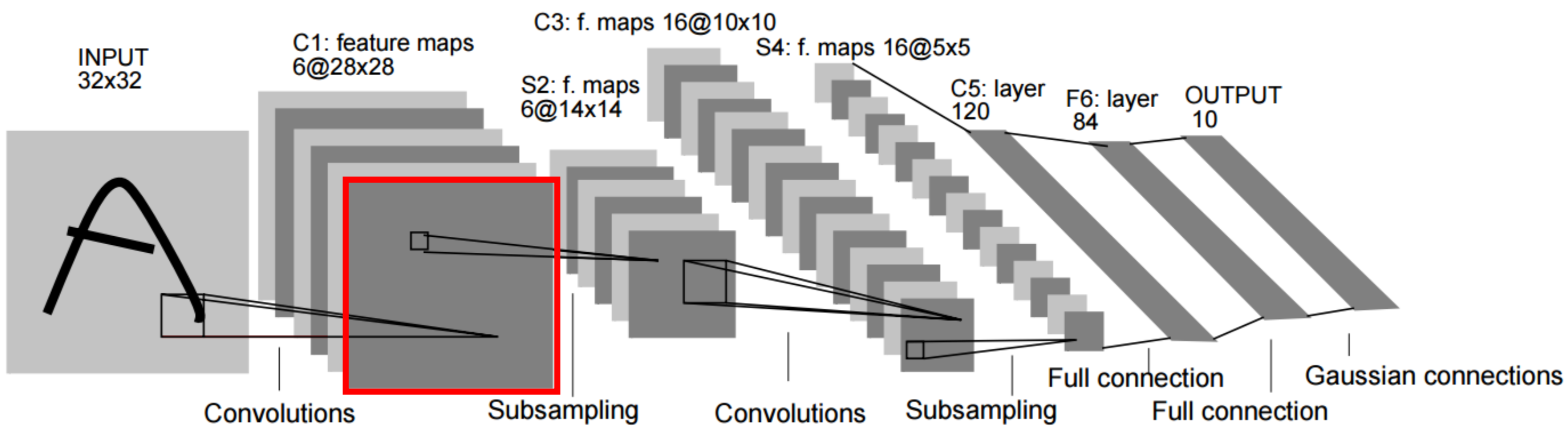
3	0	1
2	1	5
0	1	4



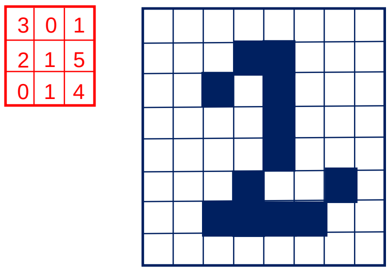
We always call it a *filter*, convolution *kernel*, or just *kernel*.

Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

A Beginning Glimpse of CNN



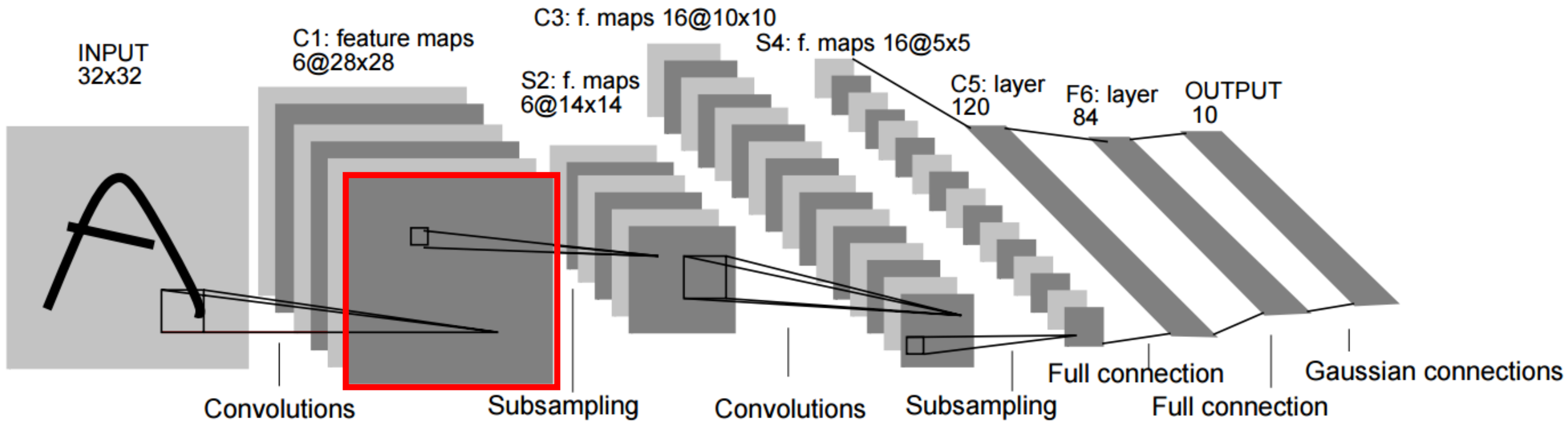
As the name indicates, a *filter* is like a *sieve*, to pick some thing "*important*".



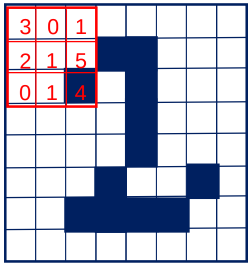
Since the filter is usually smaller than the *pattern*, we use it to *sweep* the *image*.

Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

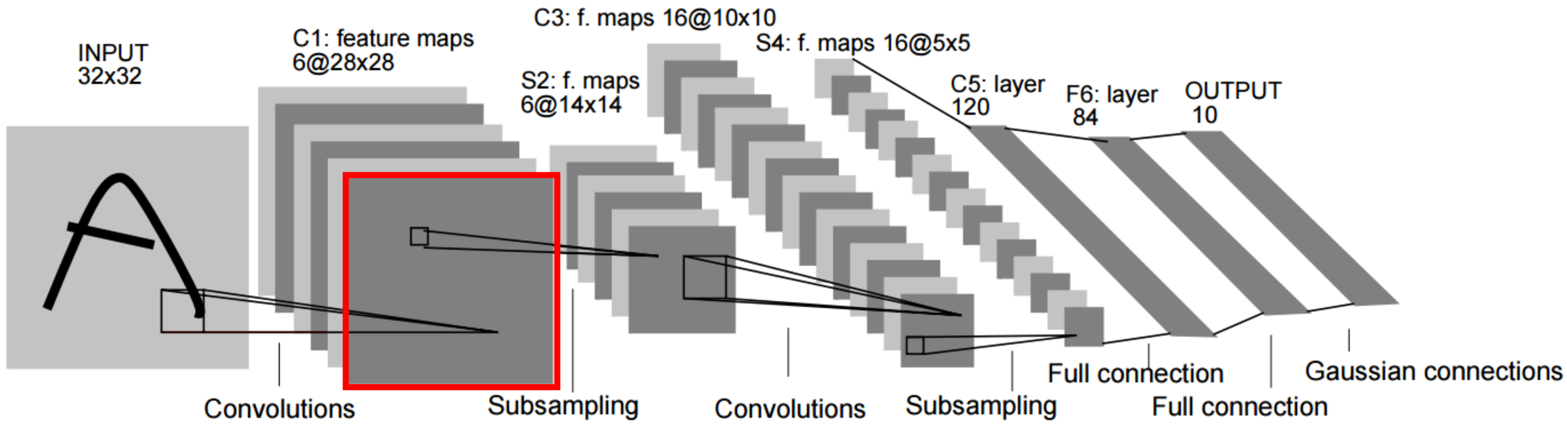
A Beginning Glimpse of CNN



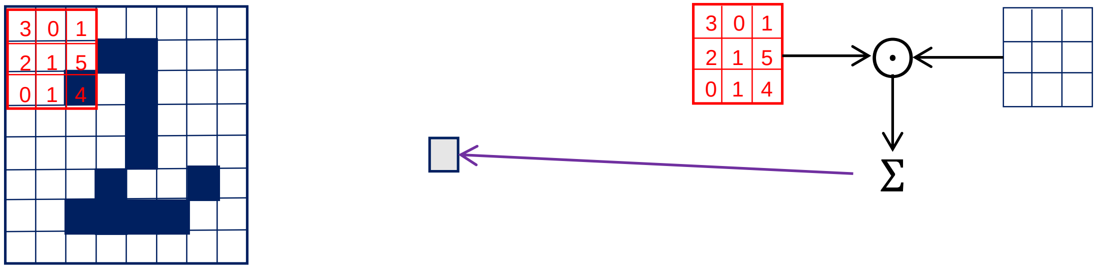
Like this:



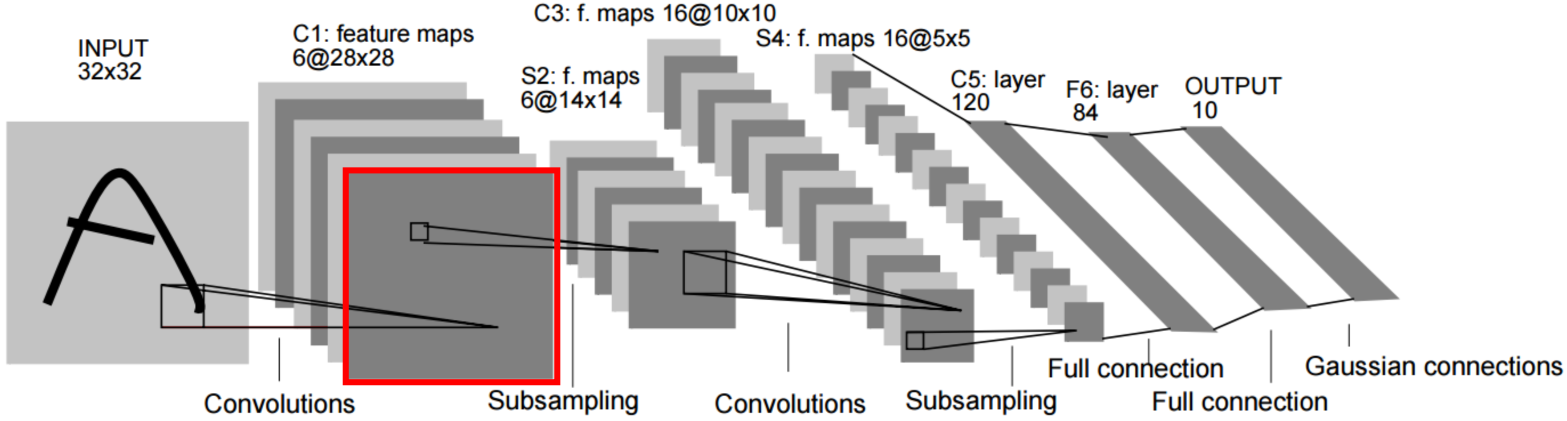
A Beginning Glimpse of CNN

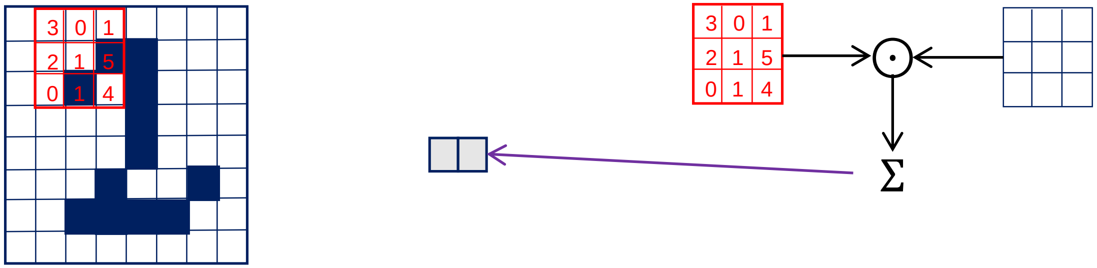


And every time, filter window does *element wise multiplication* with the box of values fitting with it. And *add* them all as one output.

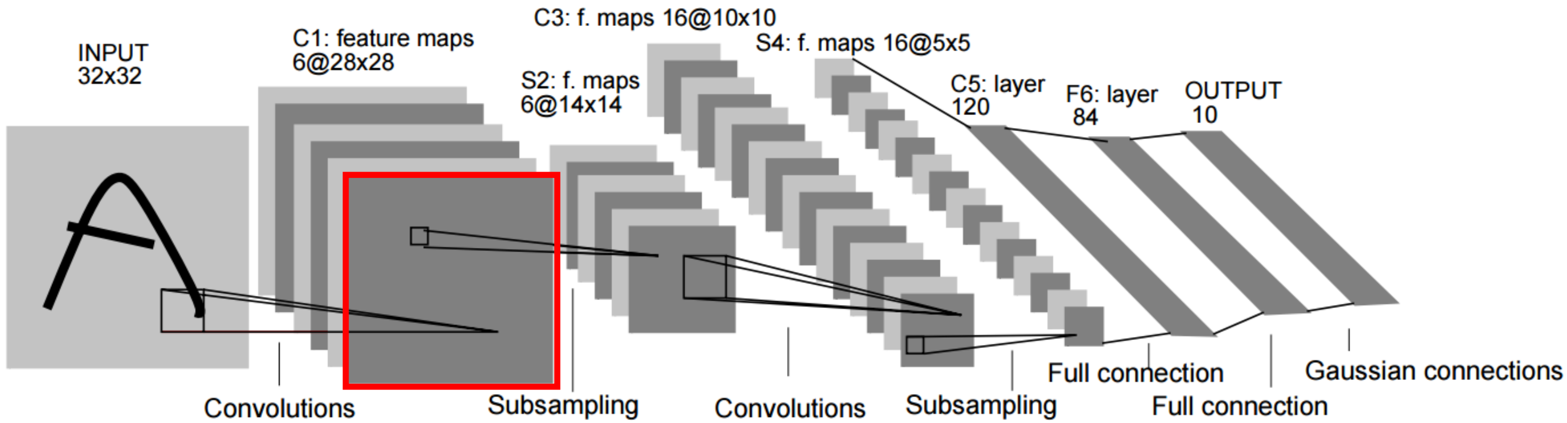


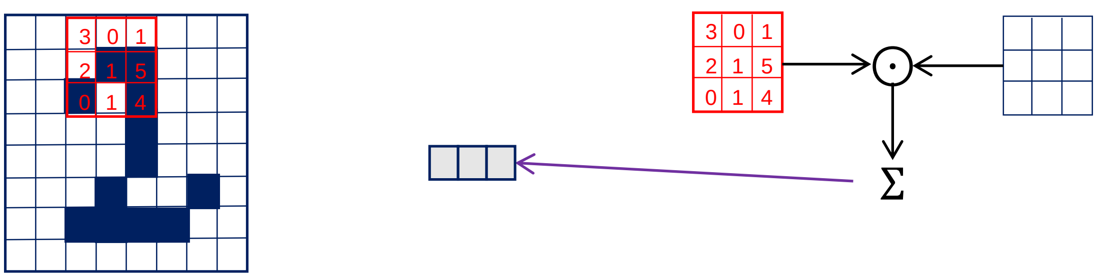
A Beginning Glimpse of CNN





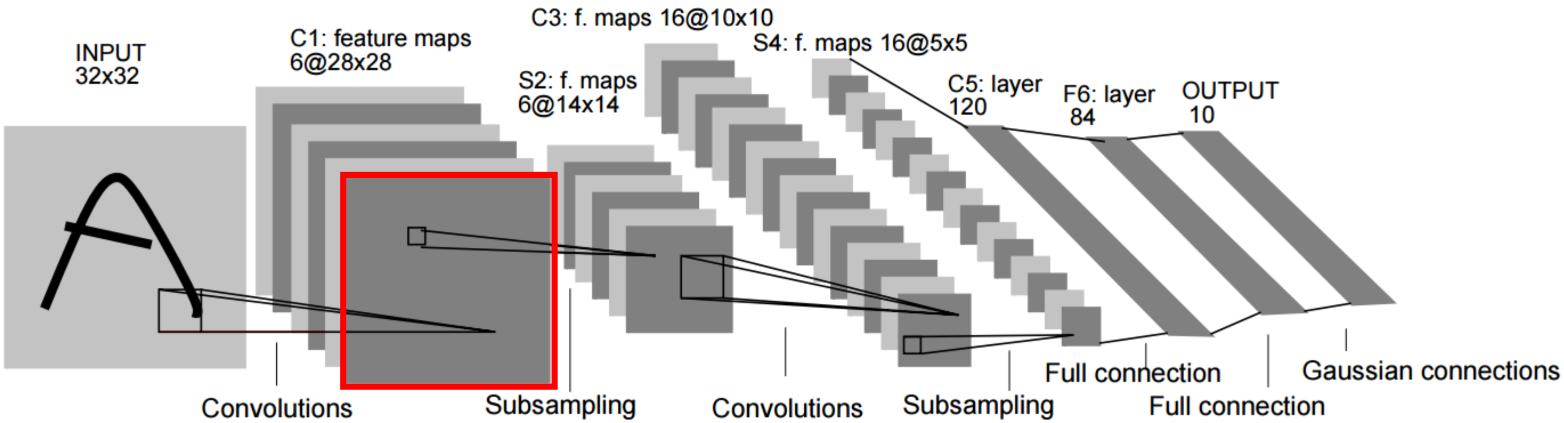
A Beginning Glimpse of CNN

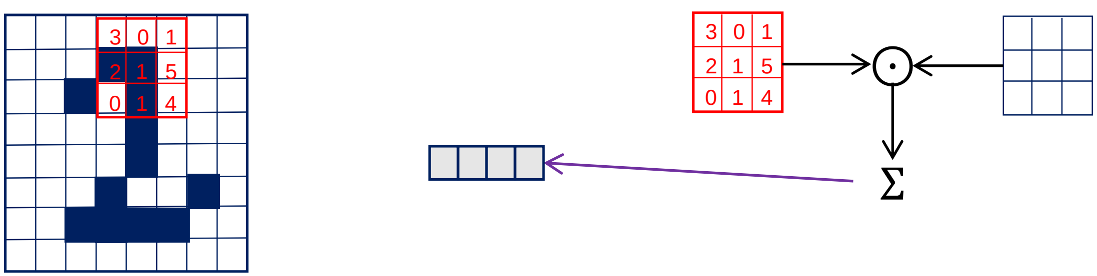




Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

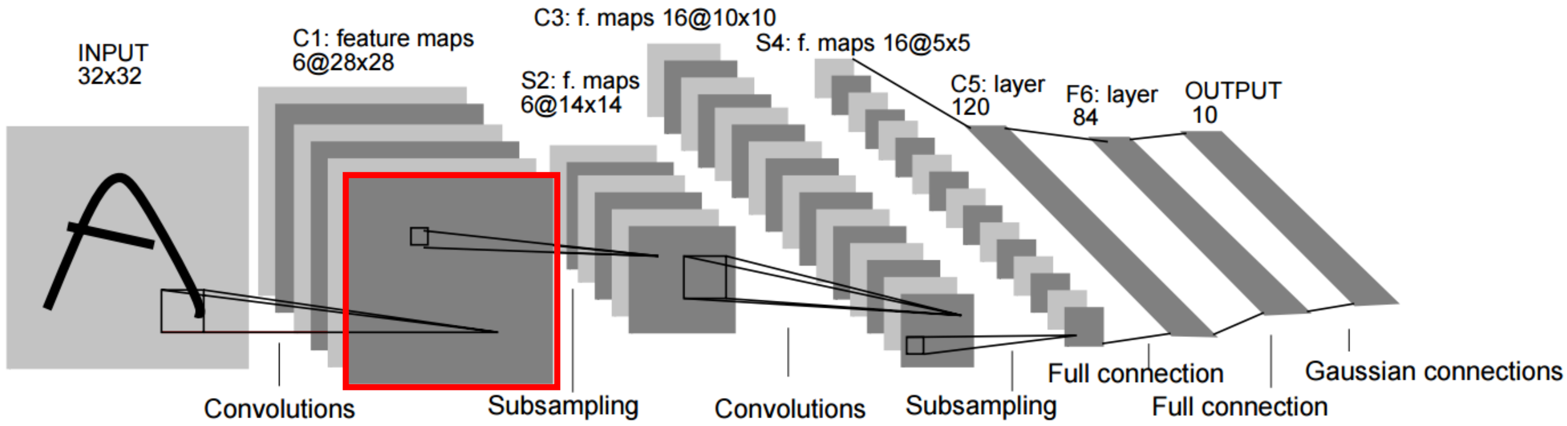
A Beginning Glimpse of CNN

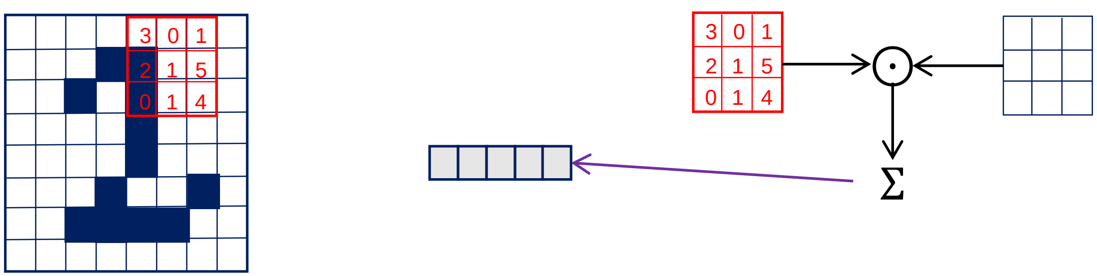




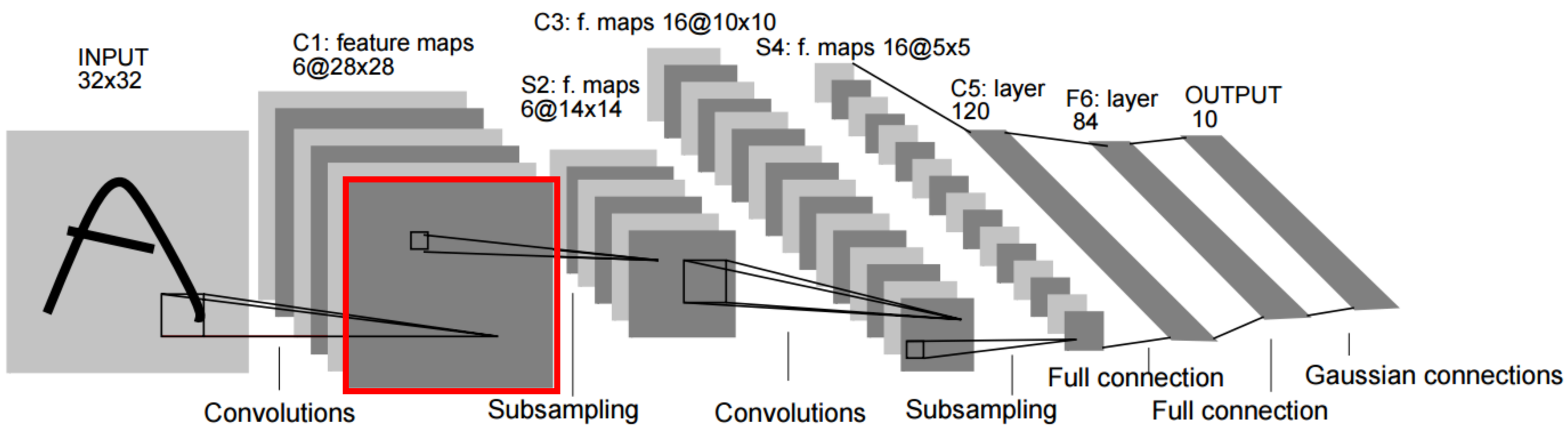
Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

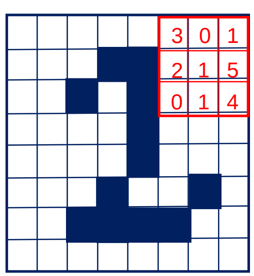
A Beginning Glimpse of CNN

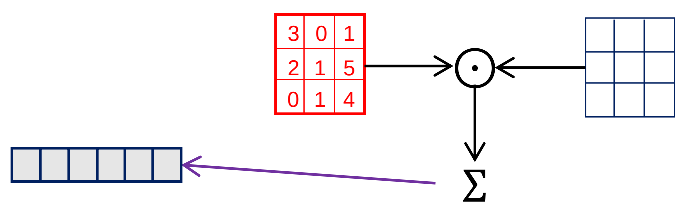




A Beginning Glimpse of CNN

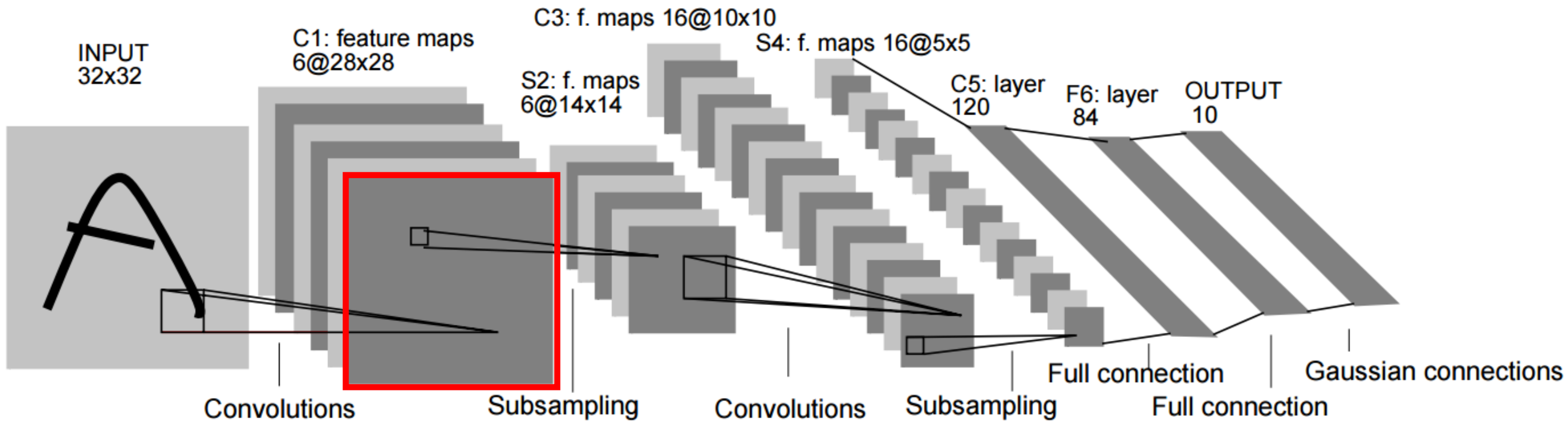


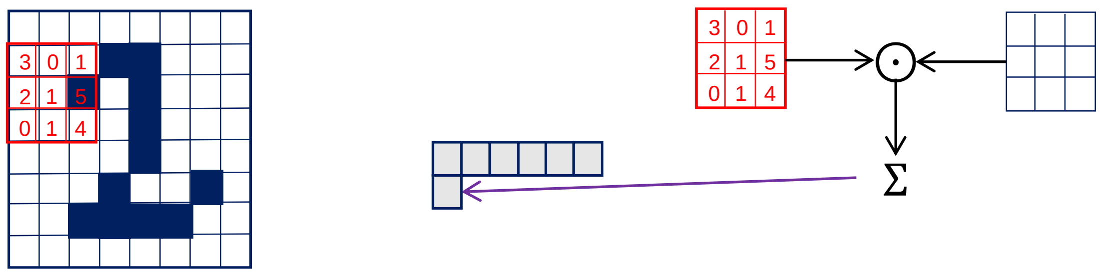




Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

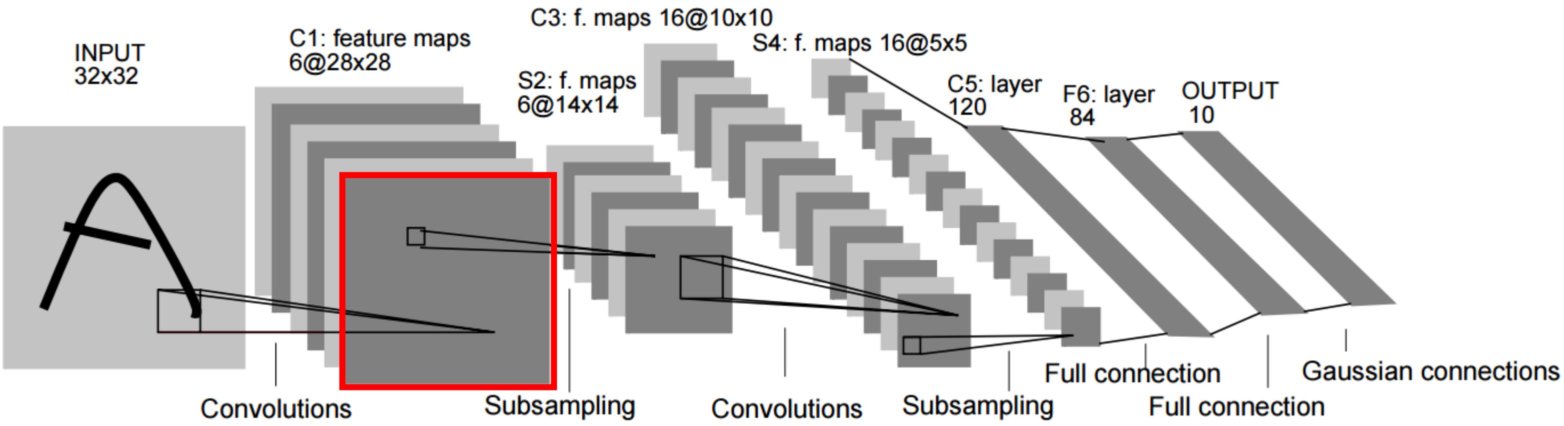
A Beginning Glimpse of CNN

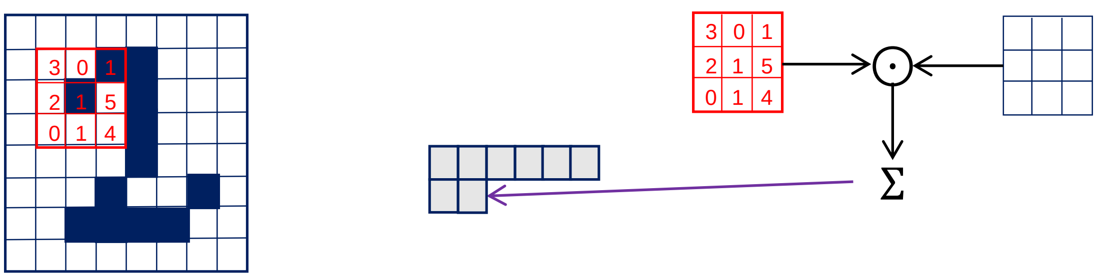




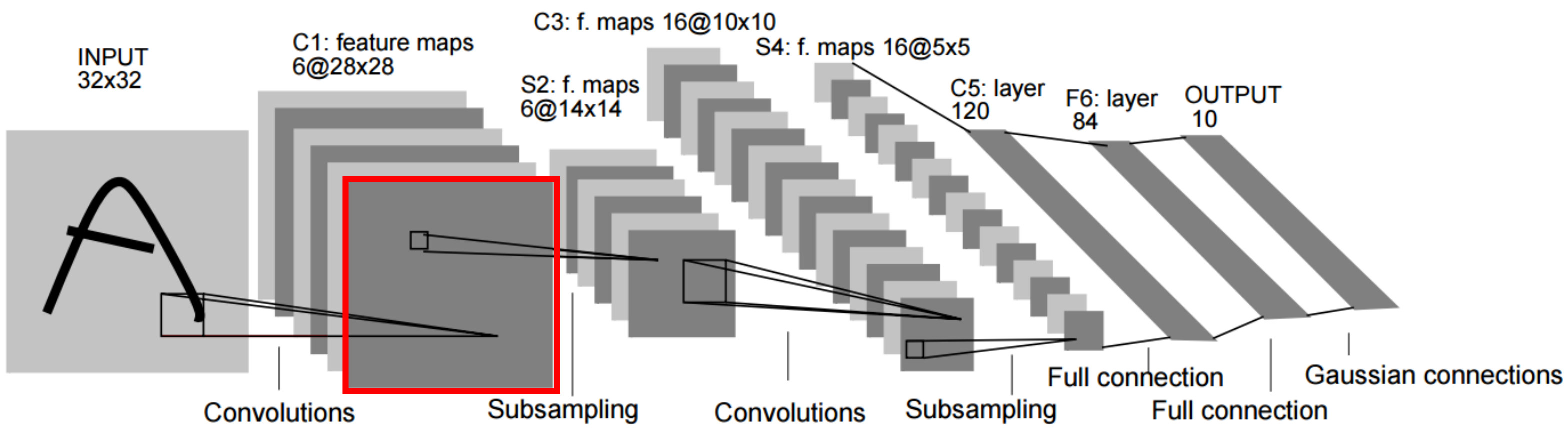
Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

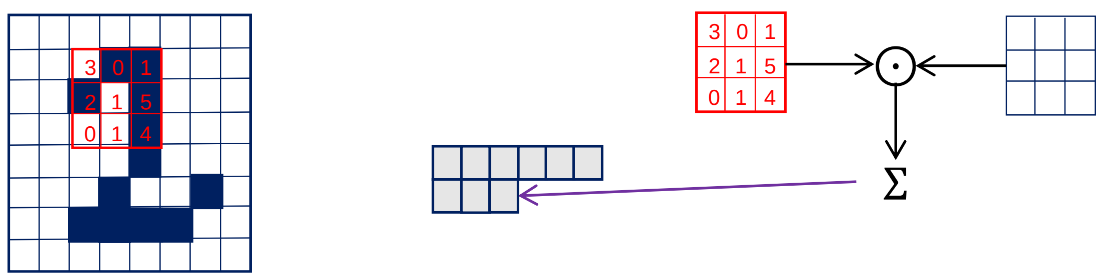
A Beginning Glimpse of CNN



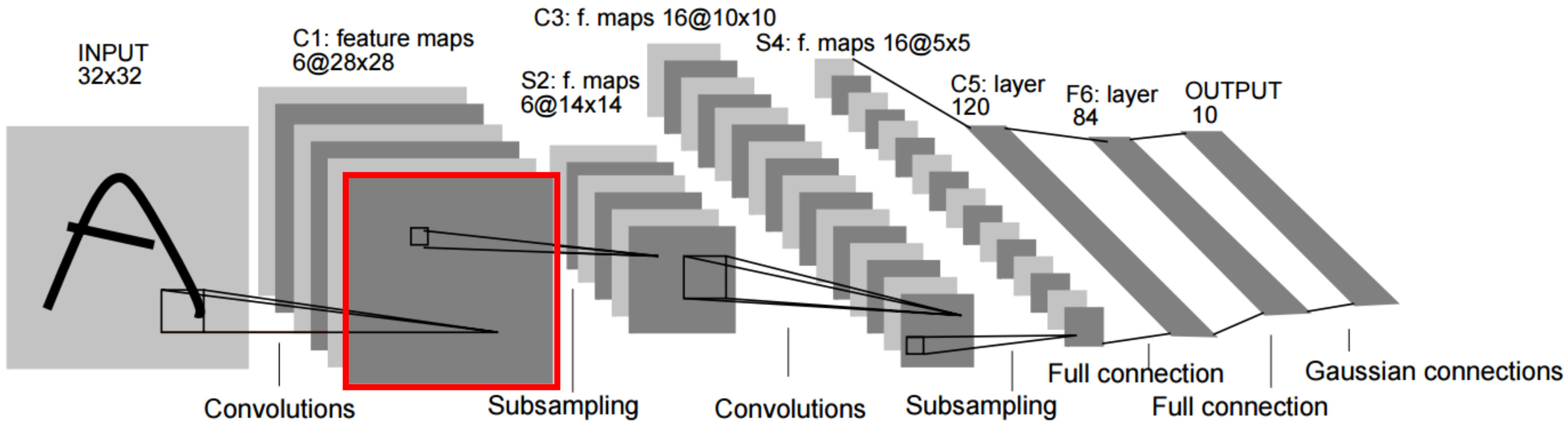


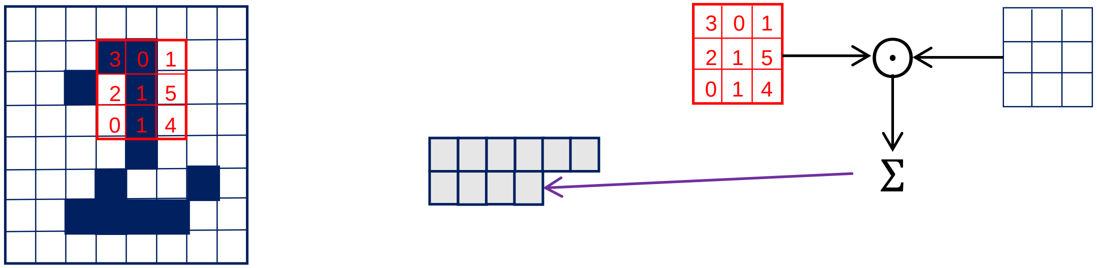
A Beginning Glimpse of CNN





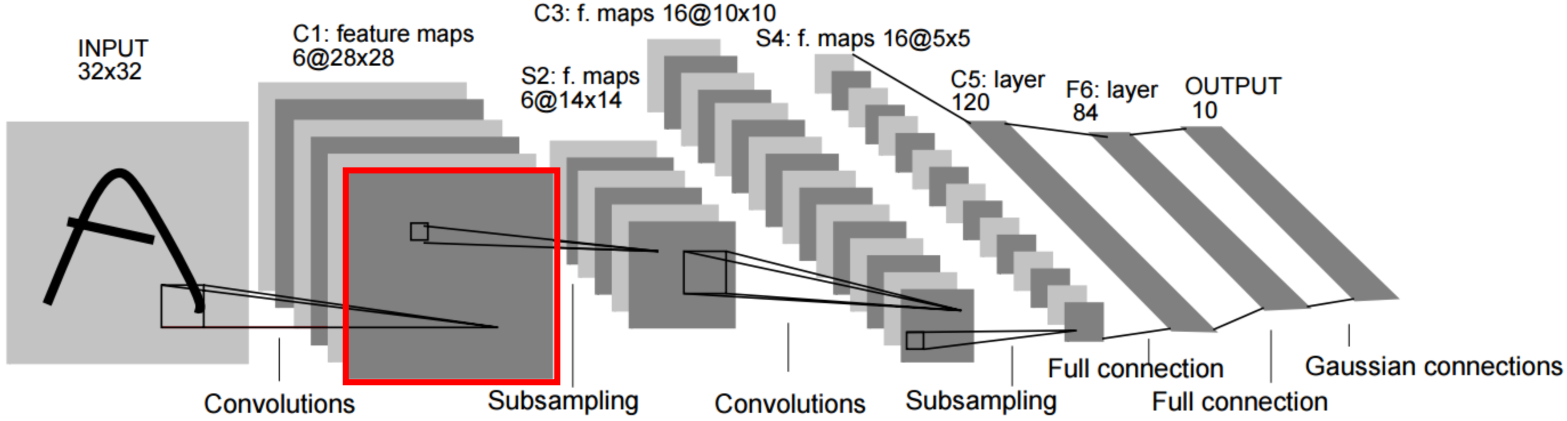
A Beginning Glimpse of CNN

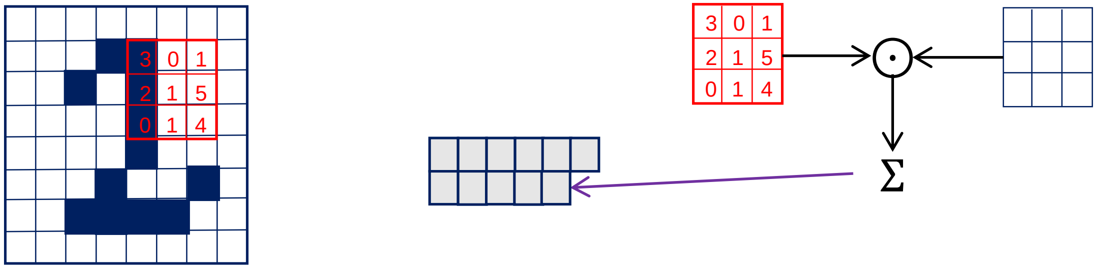




Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

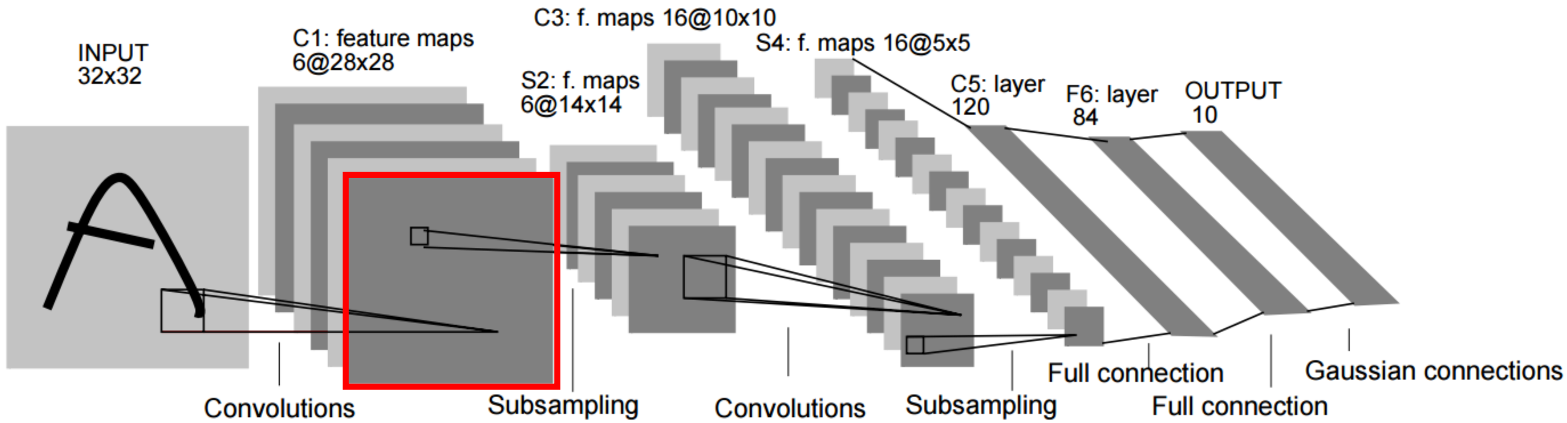
A Beginning Glimpse of CNN

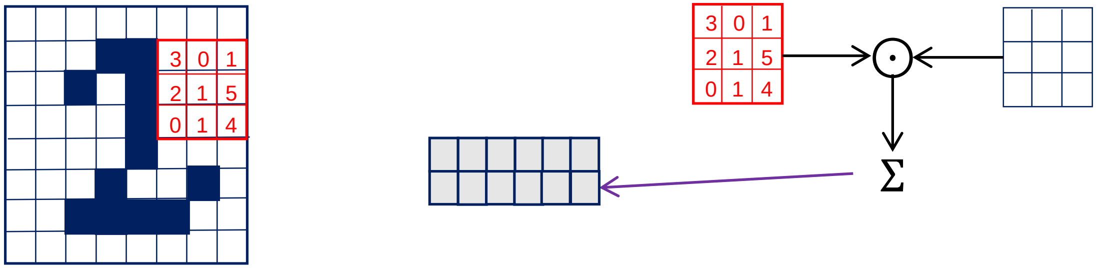




Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

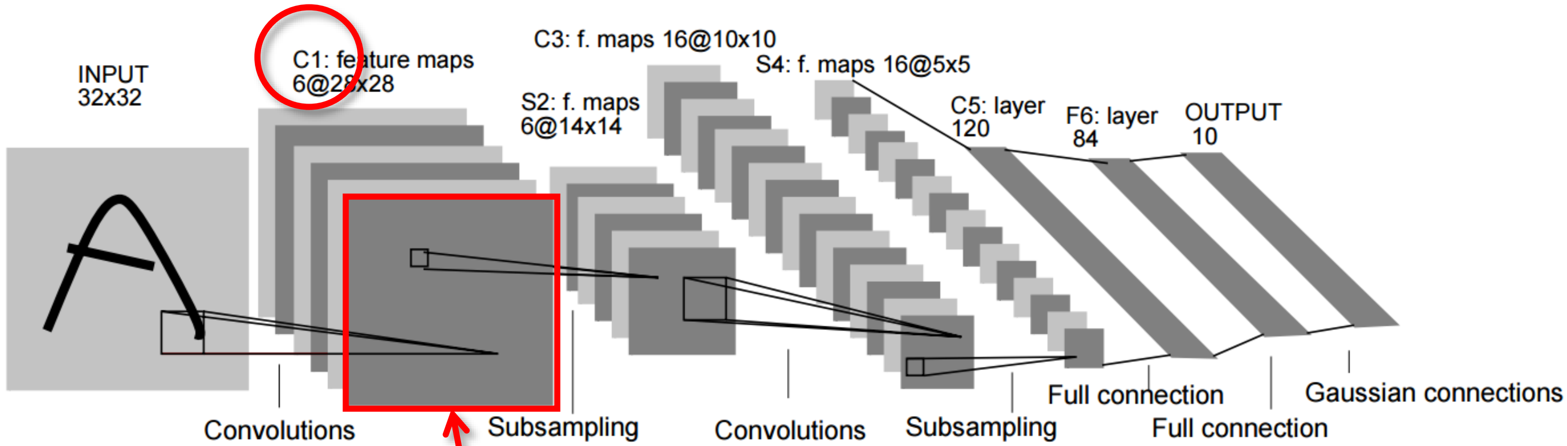
A Beginning Glimpse of CNN



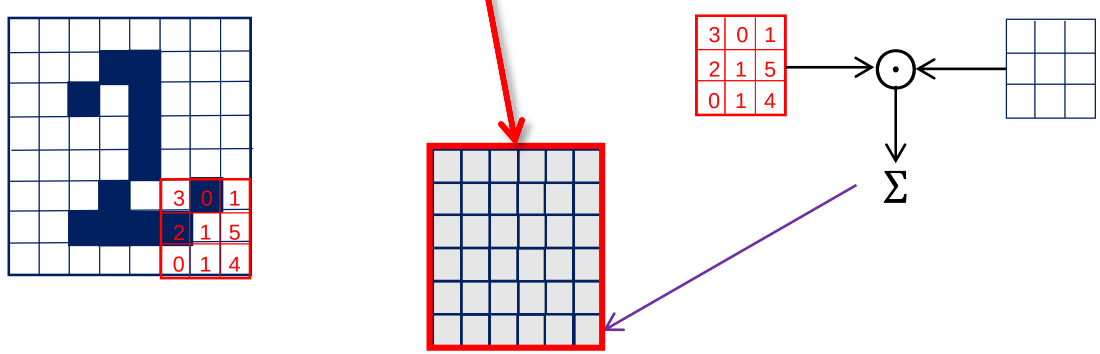


Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

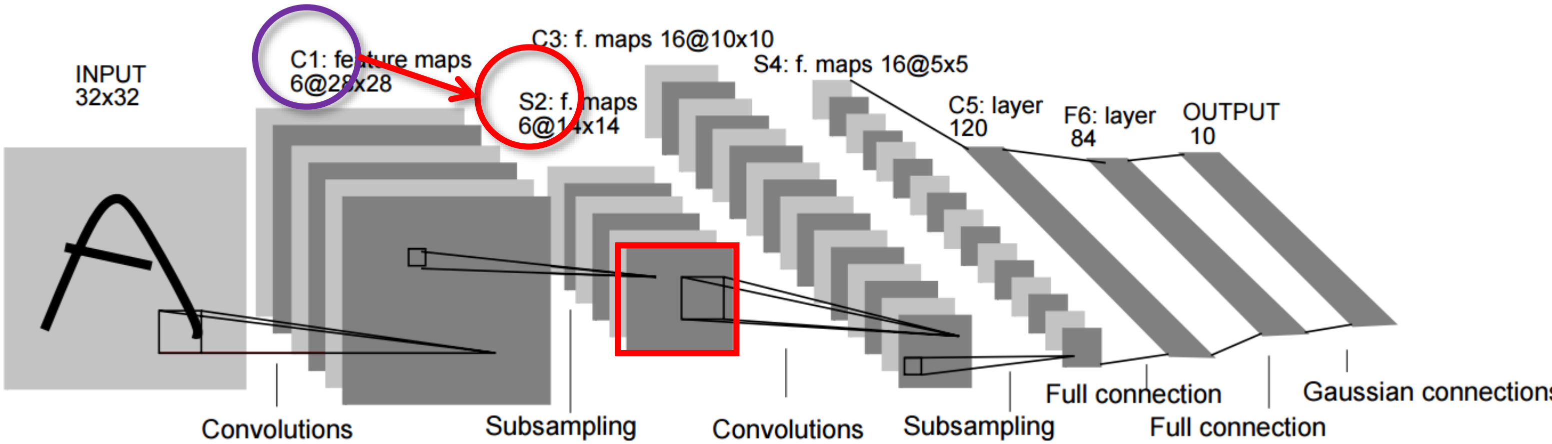
A Beginning Glimpse of CNN



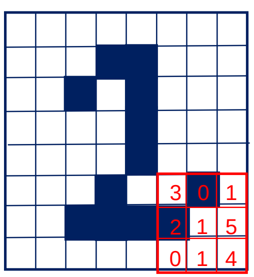
Finally, we get our feature map at C1 layer.

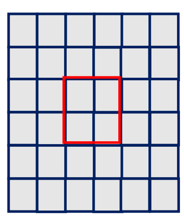


A Beginning Glimpse of CNN



Then from c1 layer to s2 layer, we call this *pooling* or *subsampling*. Mostly *un-overlapped*.

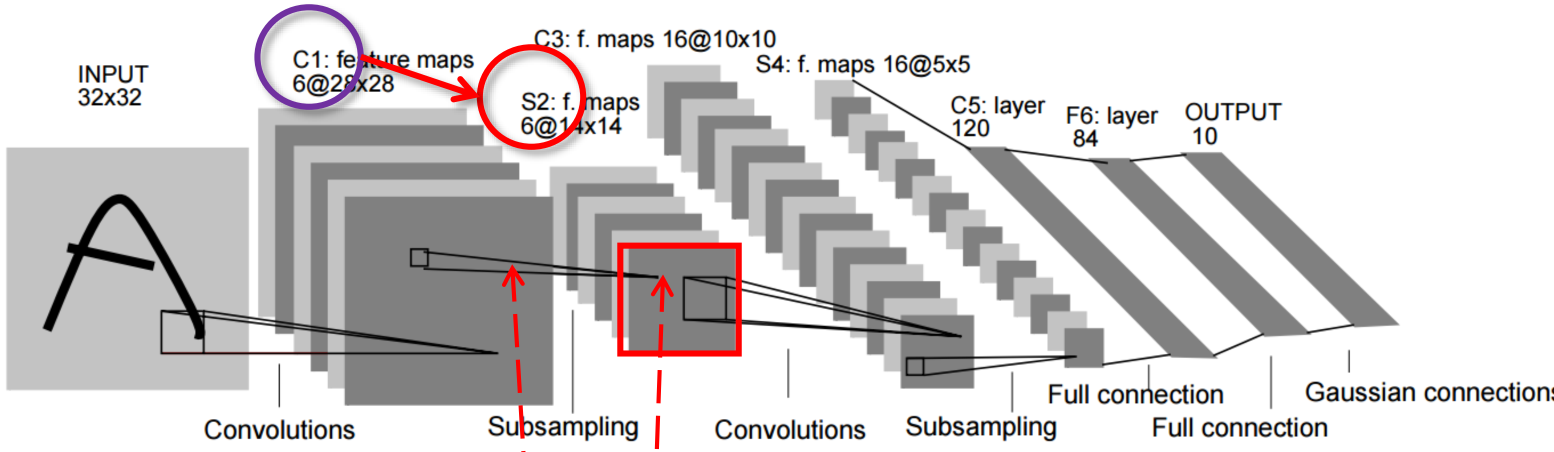




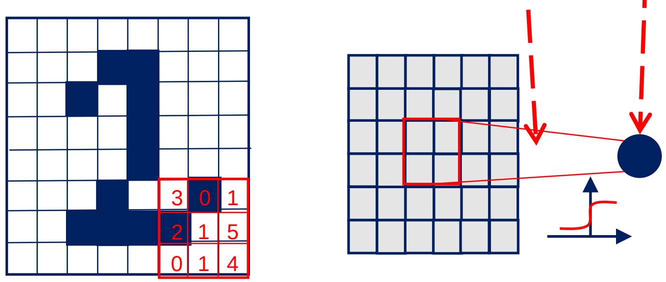
- We have a *window of size* 2×2, for example.
- In original paper: the word '*pooling*' was not used, they used the word '*subsampling*', and the actual computation is *a little bit different* from the current trend.

Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

A Beginning Glimpse of CNN



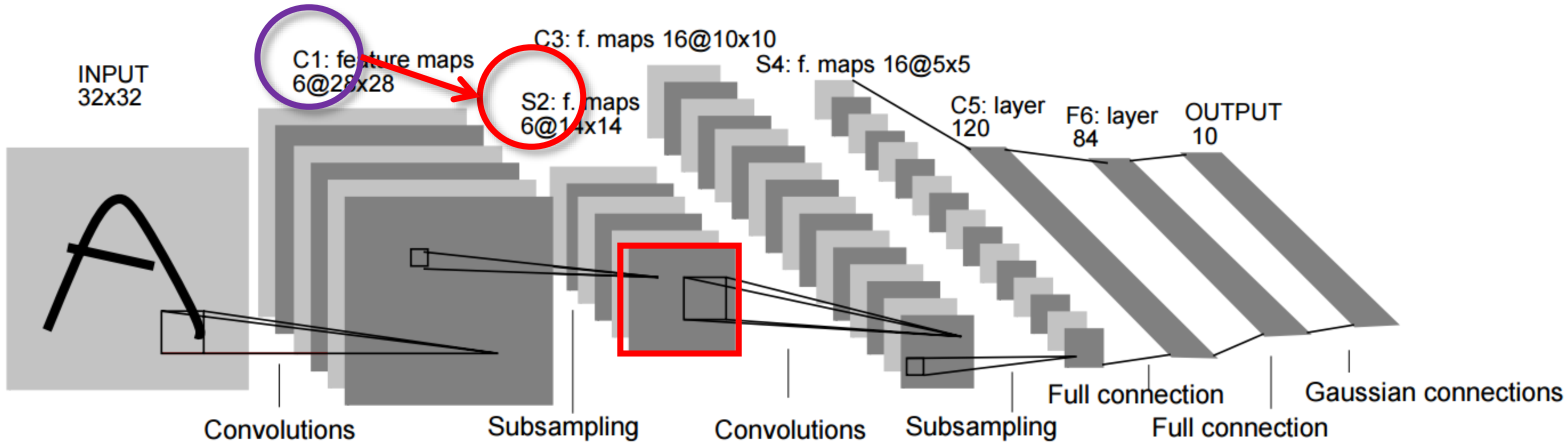
Then from c1 layer to s2 layer, we call this *pooling* or *subsampling*. Always *un-overlapped*.



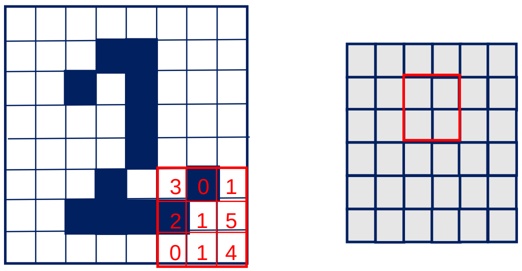
- *"The four inputs to a unit in s2 are added, then multiplied to a trainable bias."*
- *"The result is passed through a sigmoidal function"*
- *"The 2 × 2 receptive fields are non-overlapping"*

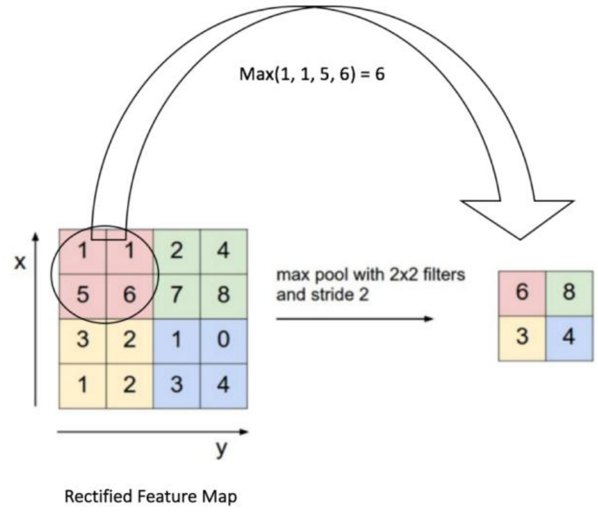
Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998

A Beginning Glimpse of CNN

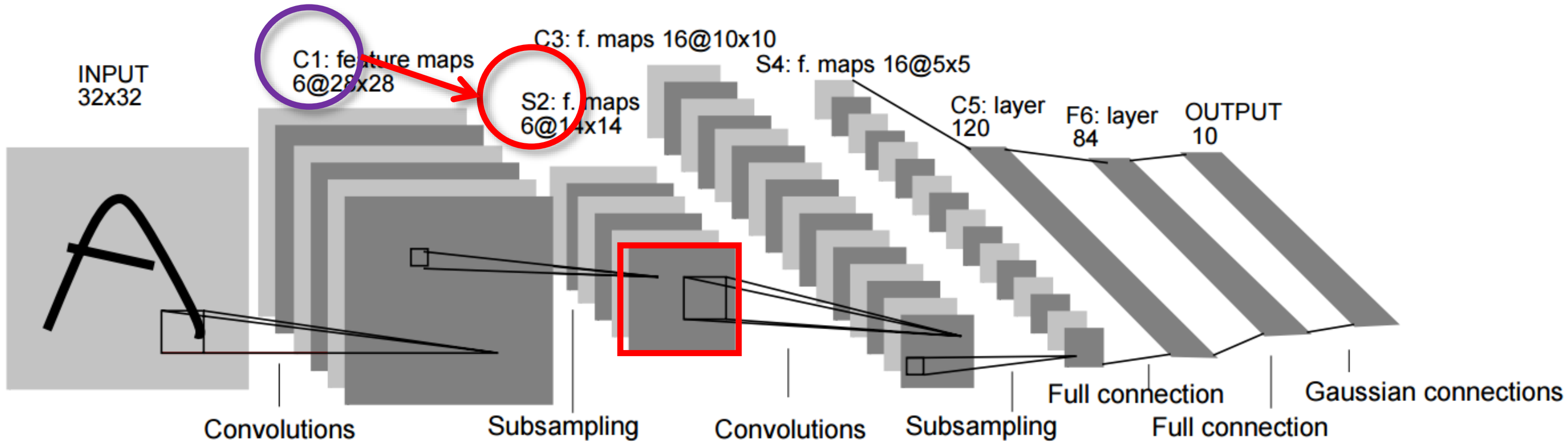


For simplicity, we can just do so-called *max-pooling*.

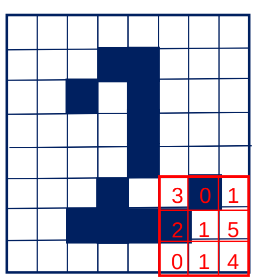


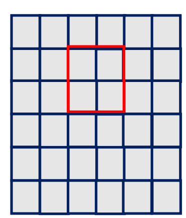


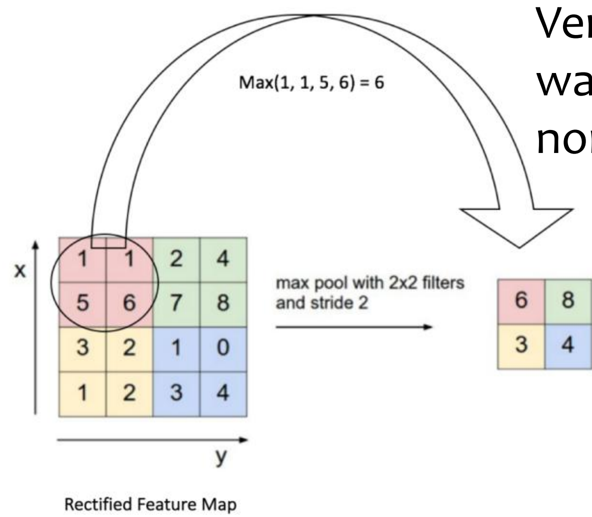
A Beginning Glimpse of CNN



For simplicity, we can just do so-called *max-pooling*.



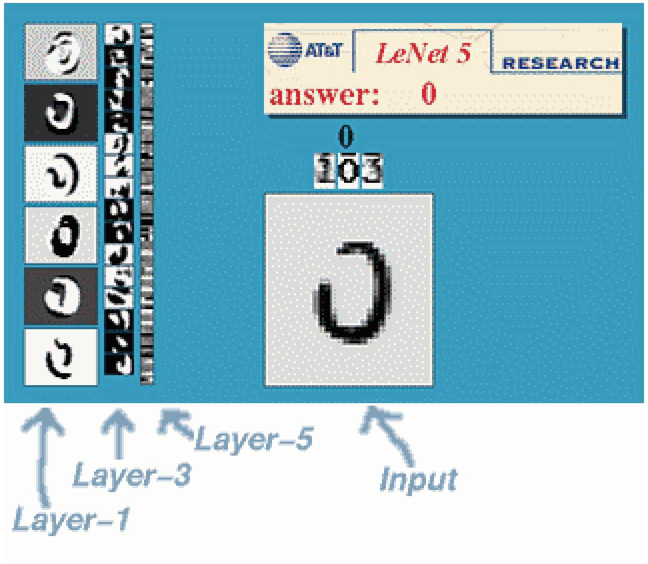




Very simple, Hah! BUT wait where comes the non-linearity?

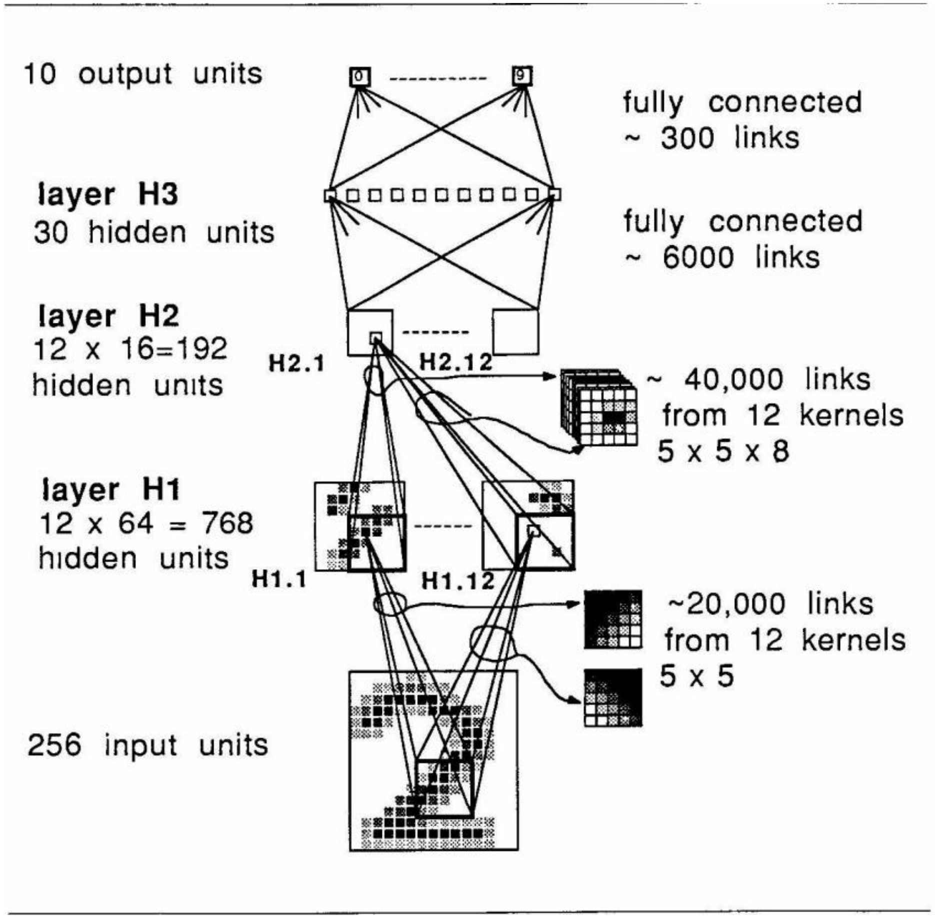
A Beginning Glimpse of CNN

- Demo from Yann LeCun
  - Transformation invariance



A Beginning Glimpse of CNN

- An early version of LeNet



Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1989

Who, Who, Who?
LeCun, LeCun, LeCun!

Yann LeCun, Né en 1960, français

Who, Who, Who? LeCun, LeCun, LeCun!

Yann LeCun was born near Paris, 1960. Got a Diplôme d'Ingénieur in 1983, and a PhD in 1987 during which proposed early BP algorithm. He was a postdoc in Geoffrey Hinton’s lab.

Who, Who, Who?
LeCun, LeCun, LeCun!

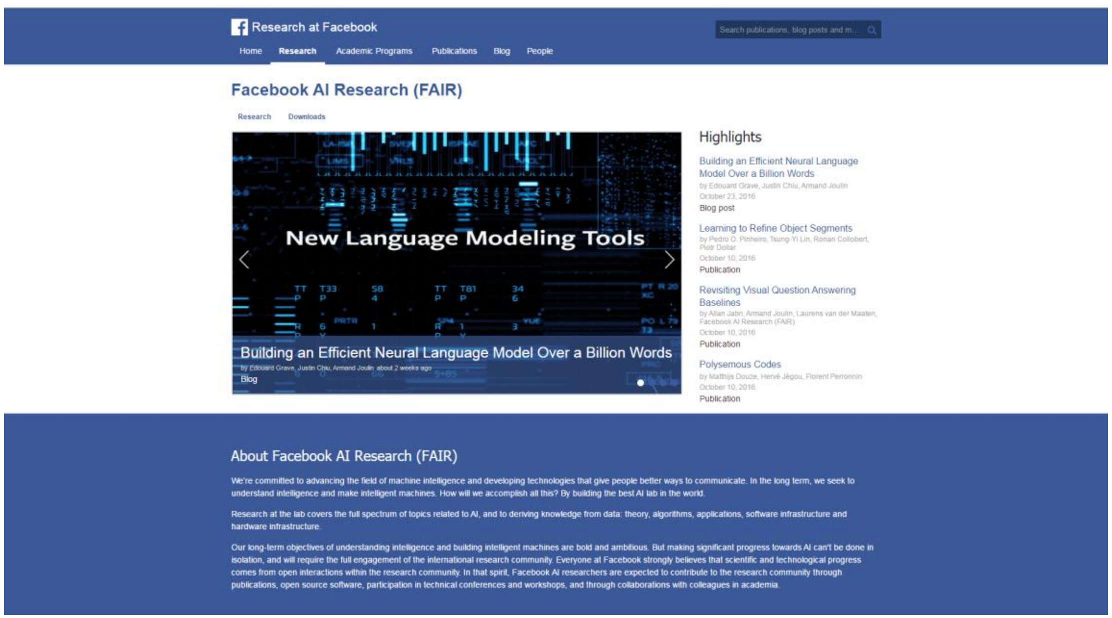
1988, joined AT&T ***Bell Lab***, and evented CNN, and in 1996, became head of Image Processing Department of AT&T Labs-Research, working on ***DjVu*** image compression techs. Collaborated with Leon Bottou and Vladimir Vapnik.

Who, Who, Who?
LeCun, LeCun, LeCun!

2003, he joined NYU, where he is Silver Professor of Computer Science Neural Science at the Courant Institute of Mathematical Science and the Center of Neural Science.

Who, Who, Who?
LeCun, LeCun, LeCun!

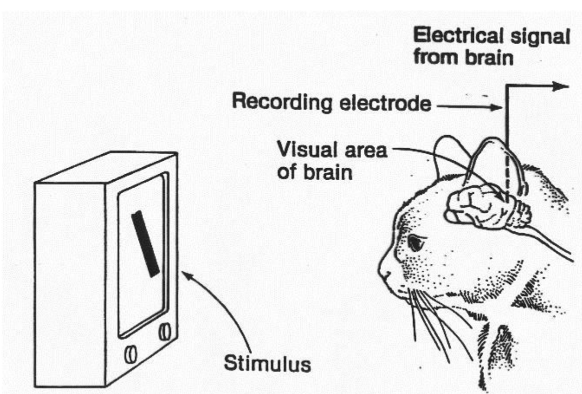
2012, he became Founding Director of the NYU Center of Data Science. On Dec. 2013, he became the first director of Facebook AI Research (FAIR) in New York City.

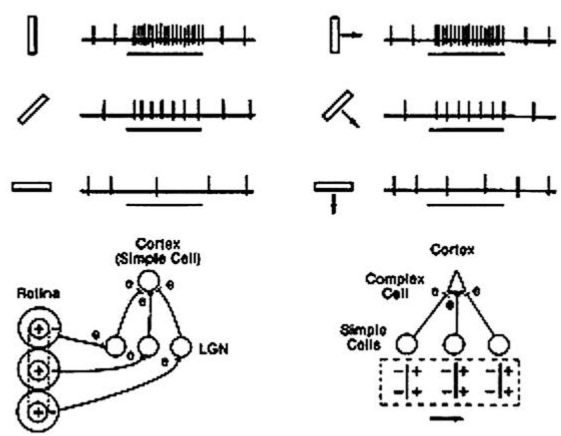


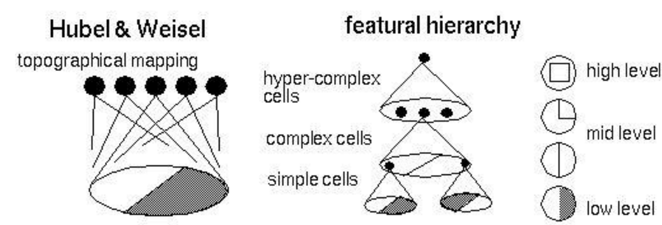
now, Real Warming Up

A Bit History

Torsten Wiesel   David Hunter Hubel

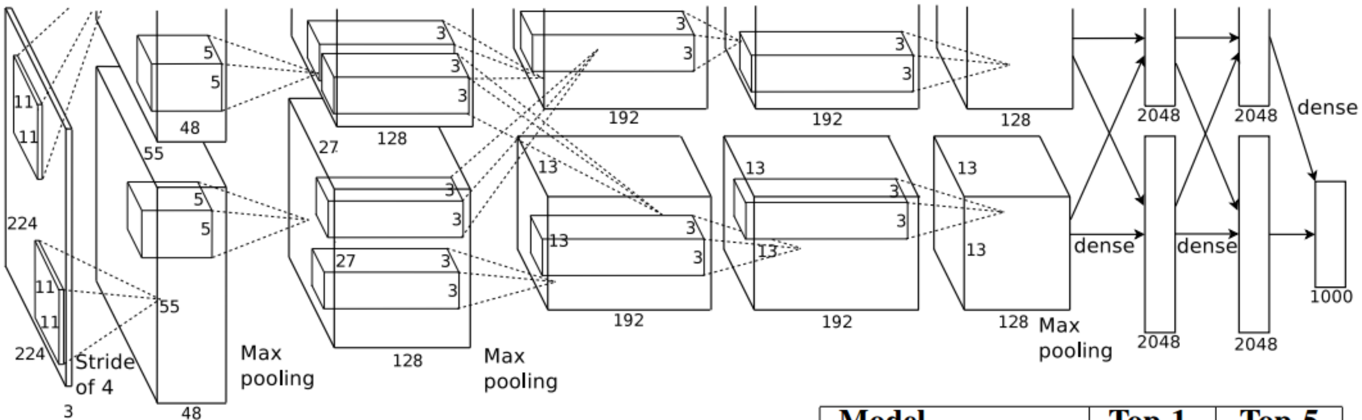






1981 Nobel Prize in Physiology or Medicine

now, Real Warming Up



Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

- First conv: 96 kernels $11 \times 11 \times 3$
- Second conv: 256 kernels $5 \times 5 \times 48$
- Third conv: 384 kernels $3 \times 3 \times 256$
- Fourth conv: 384 kernels $3 \times 3 \times 192$
- Fifth conv: 256 kernels $3 \times 3 \times 192$
- Full connect: 4096
- 1000 dim SoftMax

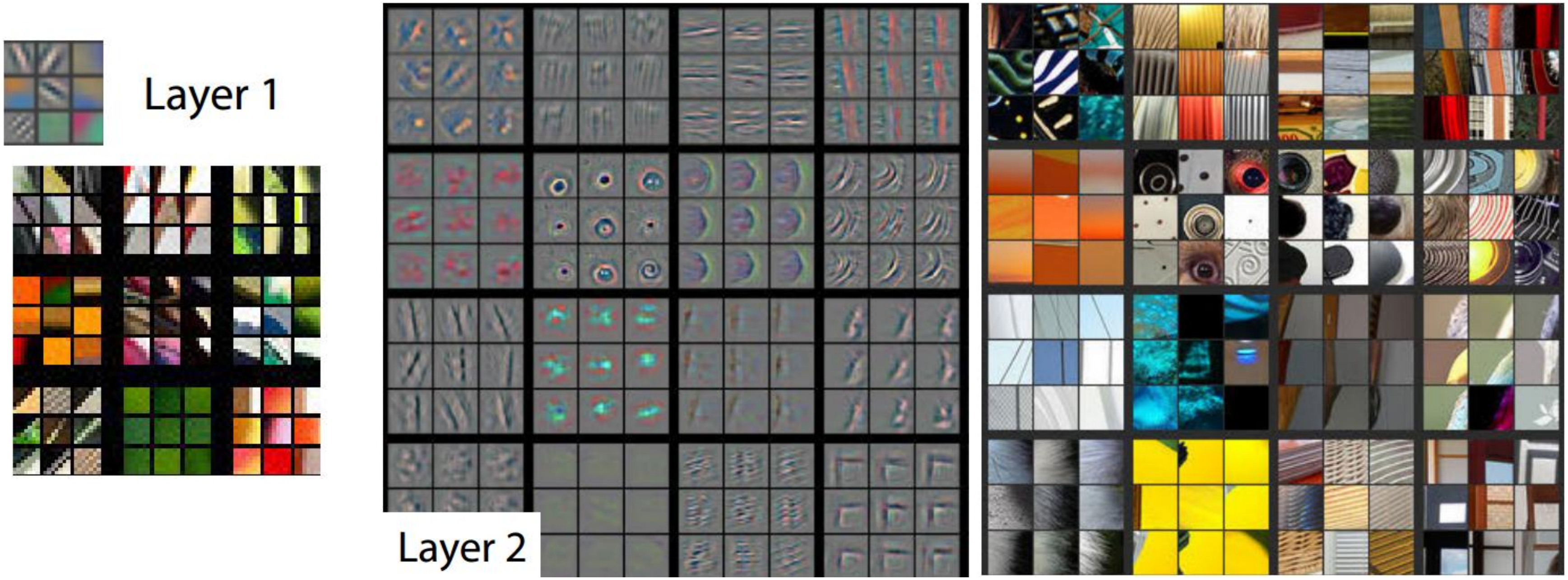
ImageNet

- "We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 context into 1000 different classes."
- "top-1 and top-5 error rates of 37.5% and 17%"

ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

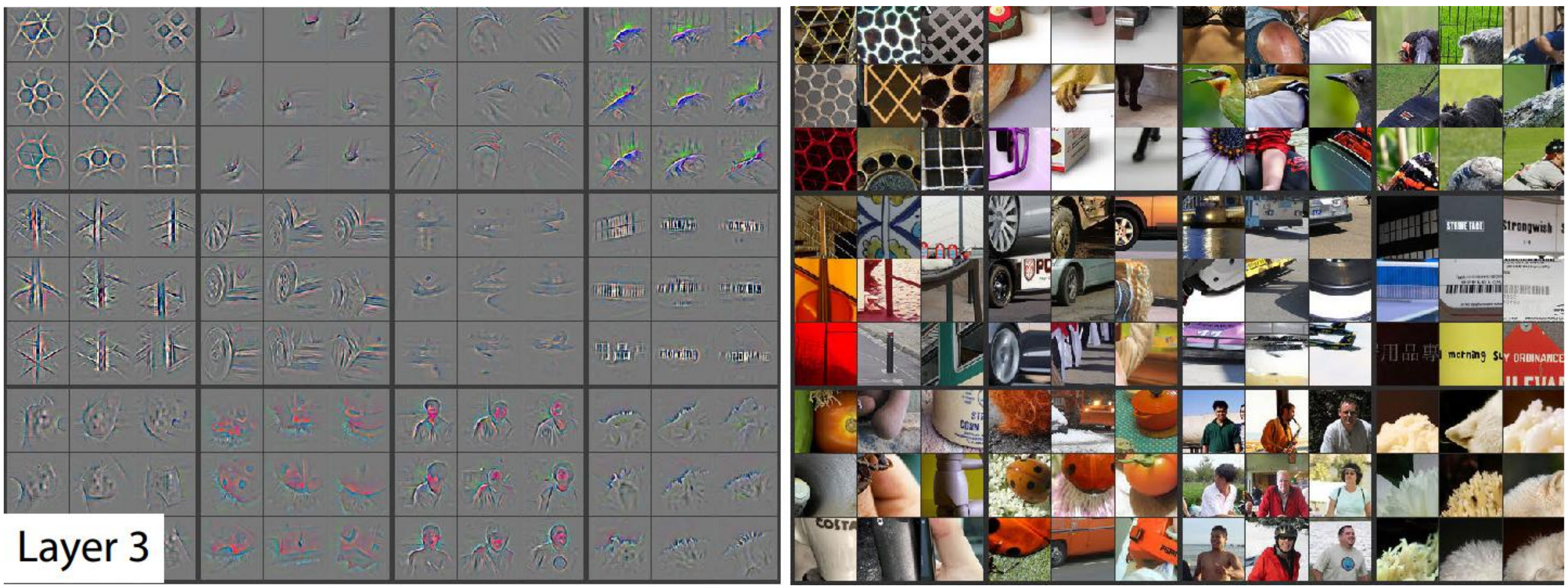
now, Real Warming Up

Visualization of CNNs



now, Real Warming Up

Different Layers Learn gradually complex features



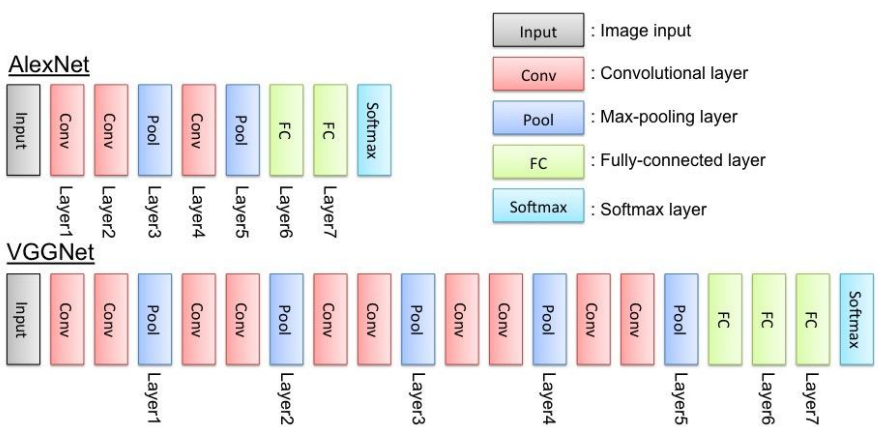
Visualizing and Understanding Convolutional Networks, arXiv 2013

now, Real Warming Up

VGGNet

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-



Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

Outline

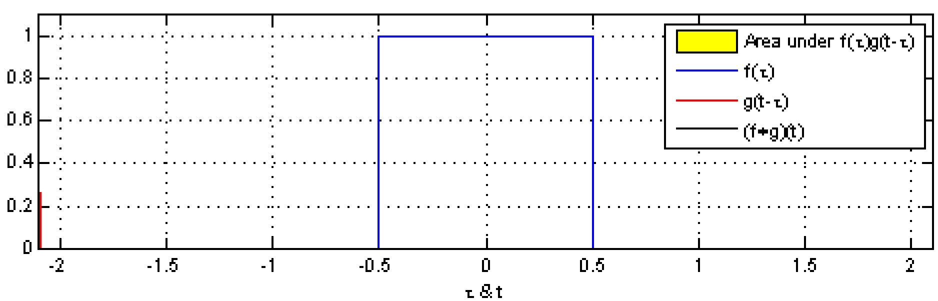
- Warming up
- Convolution
  - Filter, kernel
  - Deconvolution
- Pooling
  - Subsampling
  - Invariance
- Convolutional Neural Net
  - Architecture

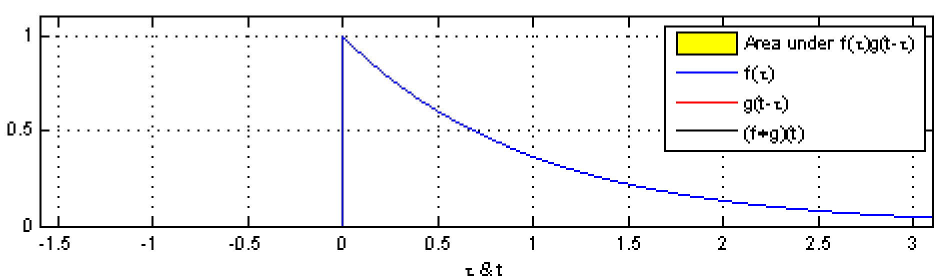
Convolution

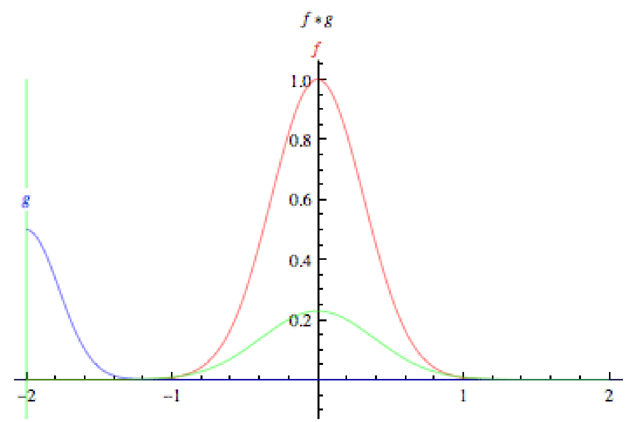
- Convolution is a very broadly used concept in many fields.
  - E.g. functional analysis, signal processing, probability theory, etc.
  - Abstractly speaking, *convolution* is a kind of *interaction* between 2 *changing* objects.

Convolution

- Mathematically, convolution is an integral calculation of two functions
  - $(f * g)(t) \stackrel{\text{def}}{=} \int f(a)g(t-a)da$

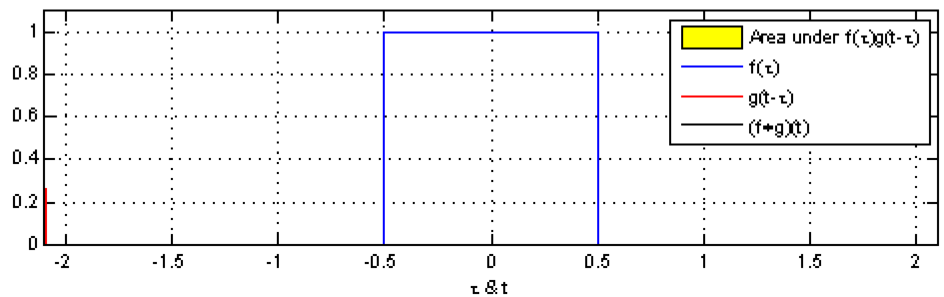




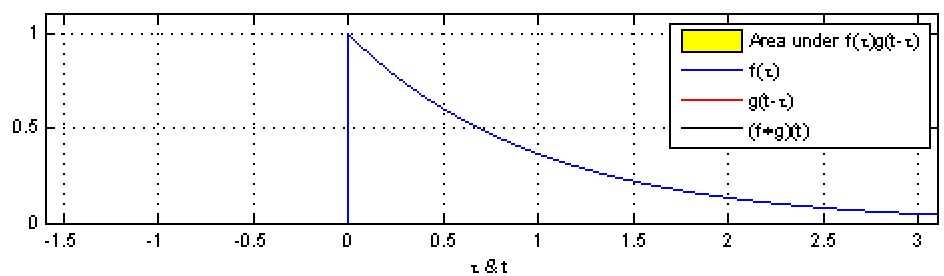


Convolution

- Mathematically, convolution is an integral calculation of two functions
  - $(f * g)(t) \stackrel{\text{def}}{=} \int f(a)g(t-a)da$

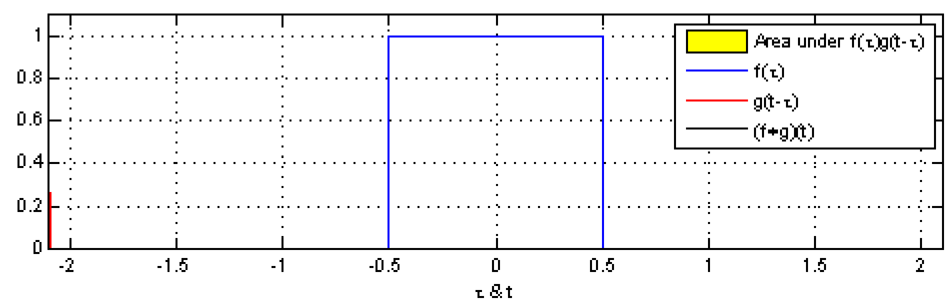


What does this help?

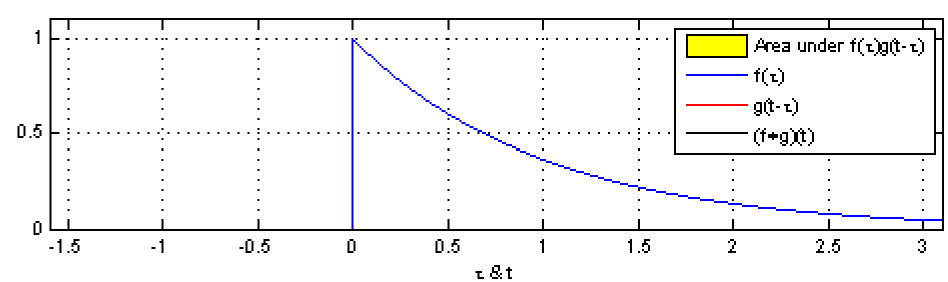


Convolution

- Let us imagine in a signal detection task
  - We use the convolution to recognize certain pattern in observed signal

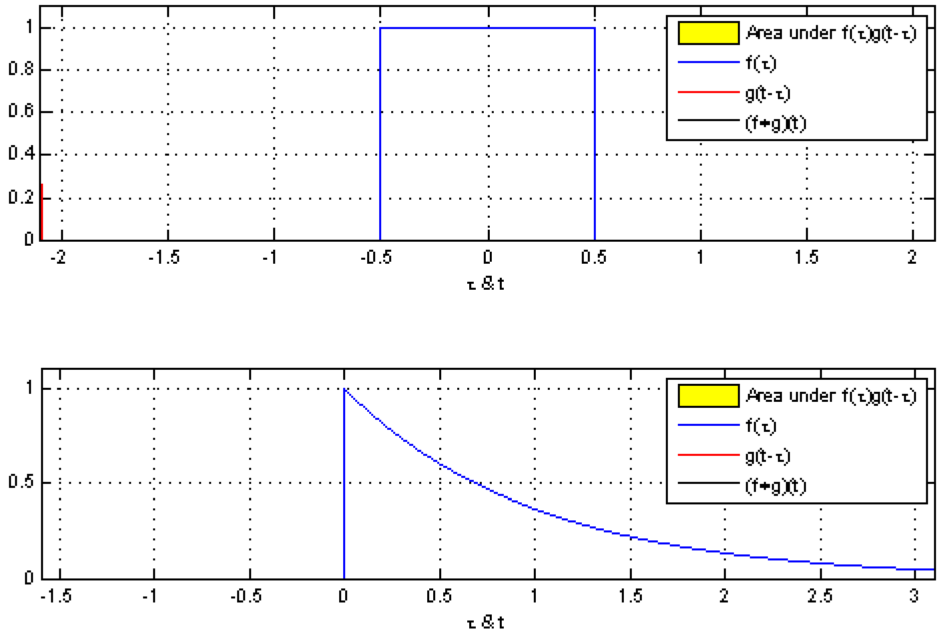


Signal detection



Convolution

- So convolution is away to *extract specific properties* in signals, or more broadly any kind of data.



Data pattern detection

Convolution

- In **Image Processing**
  - Convolution is always named *filtering*, and there are many famous *filters/convolution kernels* that extract intuitive features in images

Convolution

- In **Image Processing**
  - A filter is special designed matrix always squared, and could extract special pattern in an image

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

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

	0	0	0	
	-1	1	0	
	0	0	0	

	0	1	0	
	1	-4	1	
	0	1	0	

Convolution

- In computer vision
  - We always *Blur* an image

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

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

	0	0	0	
	-1	1	0	
	0	0	0	

	0	1	0	
	1	-4	1	
	0	1	0	

Convolution

- In computer vision
  - We always *Blur* an image

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

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

	0	0	0	
	-1	1	0	
	0	0	0	

	0	1	0	
	1	-4	1	
	0	1	0	

Convolution

- In computer vision
  - We always *Detect edge of* an image

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

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

	0	0	0	
	-1	1	0	
	0	0	0	

	0	1	0	
	1	-4	1	
	0	1	0	

Convolution

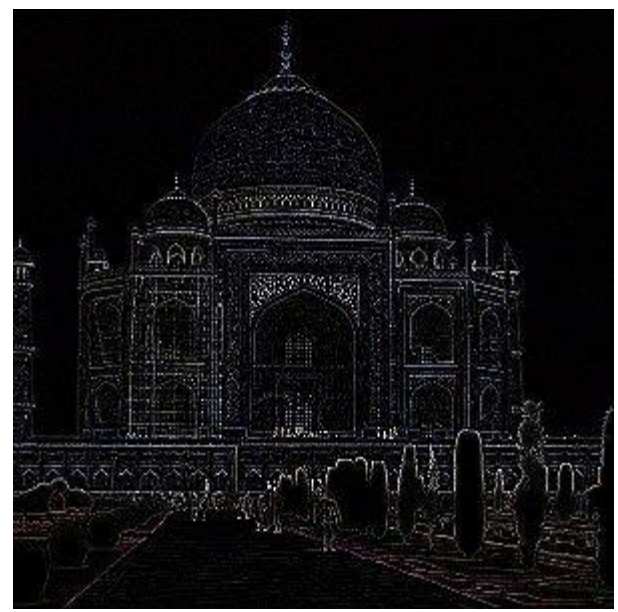
- In computer vision
  - We always *Detect edge of* an image

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

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

	0	0	0	
	-1	1	0	
	0	0	0	

	0	1	0	
	1	-4	1	
	0	1	0	



Convolution

- In computer vision
  - We always *Sharpen* an image

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

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

	0	0	0	
	-1	1	0	
	0	0	0	

	0	1	0	
	1	-4	1	
	0	1	0	

Convolution

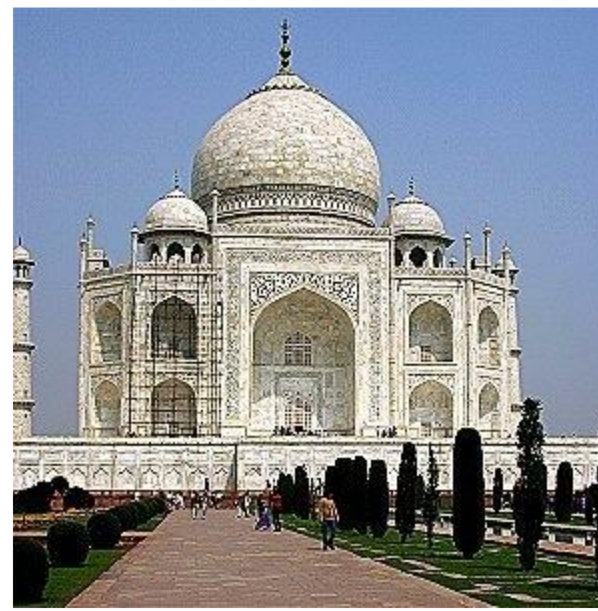
- In computer vision
  - We always *Sharpen* an image

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

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

	0	0	0	
	-1	1	0	
	0	0	0	

	0	1	0	
	1	-4	1	
	0	1	0	



Convolution

- In computer vision
  - We always *Enhance edge of* an image

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

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

	0	0	0	
	-1	1	0	
	0	0	0	

	0	1	0	
	1	-4	1	
	0	1	0	

Convolution

- In computer vision
  - We always *Enhance edge of* an image

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

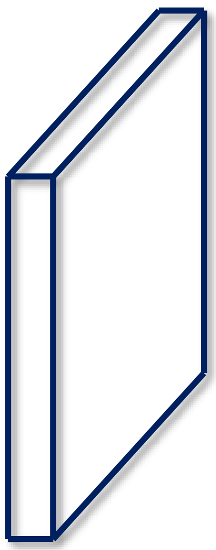
0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

	0	0	0	
	-1	1	0	
	0	0	0	

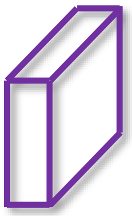
	0	1	0	
	1	-4	1	
	0	1	0	

Convolution, abstractly!

- Given an image and a filter: both tensors
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step



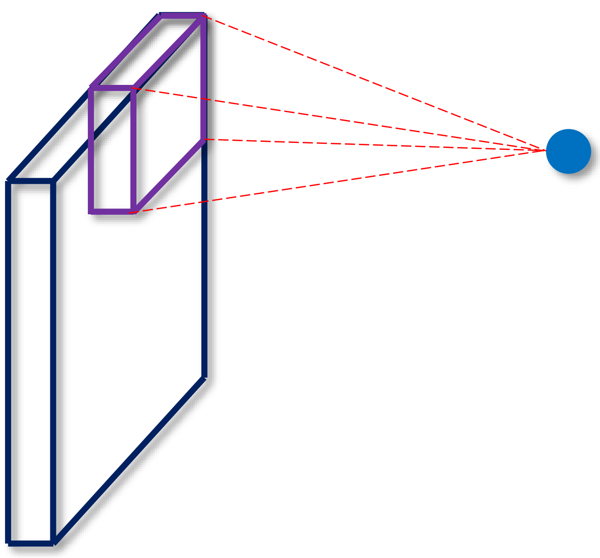
$32 \times 32 \times 3$



$5 \times 5 \times 3$

Convolution, abstractly!

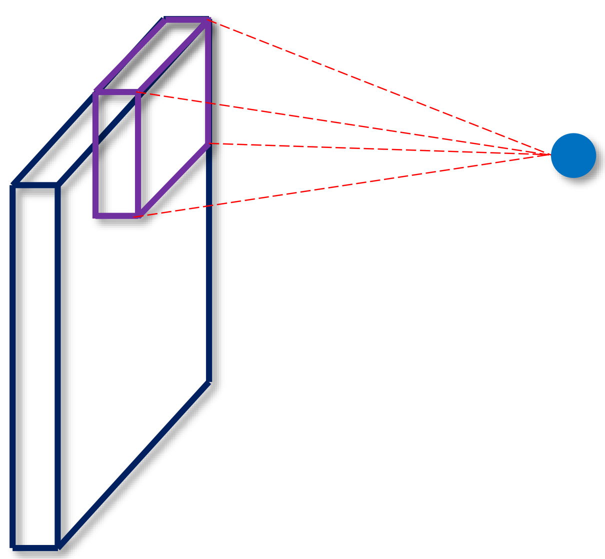
- Given an image and a filter: both tensor
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step



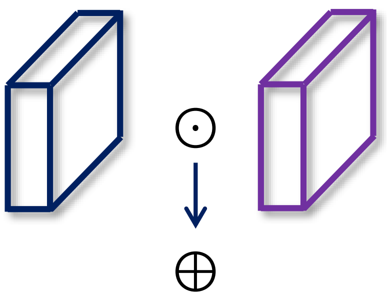
$32 \times 32 \times 3$

Convolution, abstractly!

- Given an image and a filter: both tensor
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step

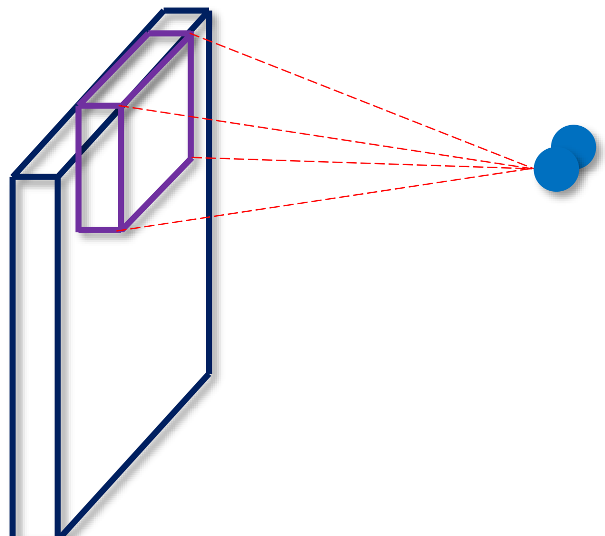


$32 \times 32 \times 3$



Convolution, abstractly!

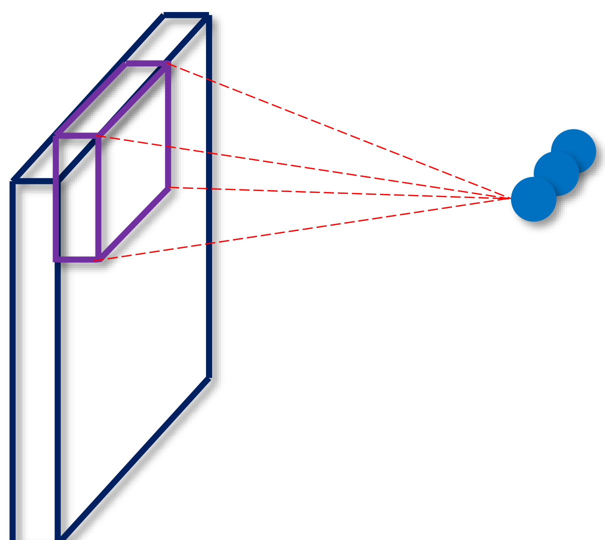
- Given an image and a filter: both tensor
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step



$32 \times 32 \times 3$

Convolution, abstractly!

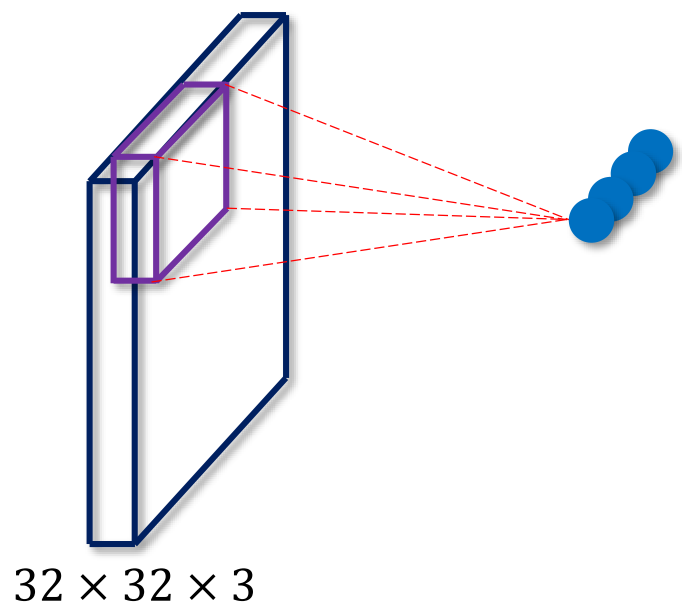
- Given an image and a filter: both tensor
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step



$32 \times 32 \times 3$

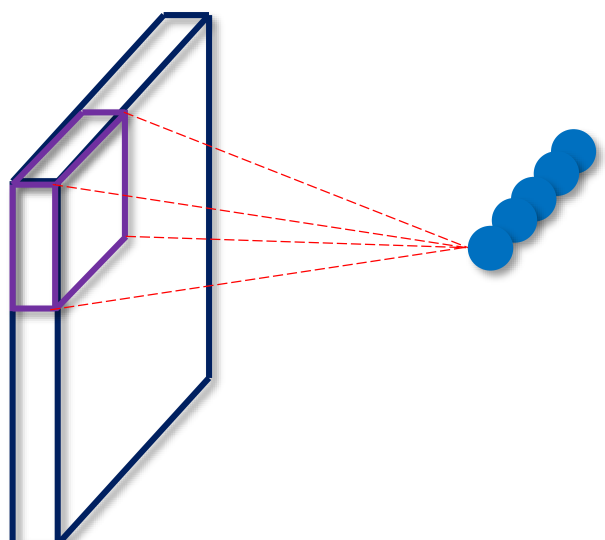
Convolution, abstractly!

- Given an image and a filter: both tensor
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step



Convolution, abstractly!

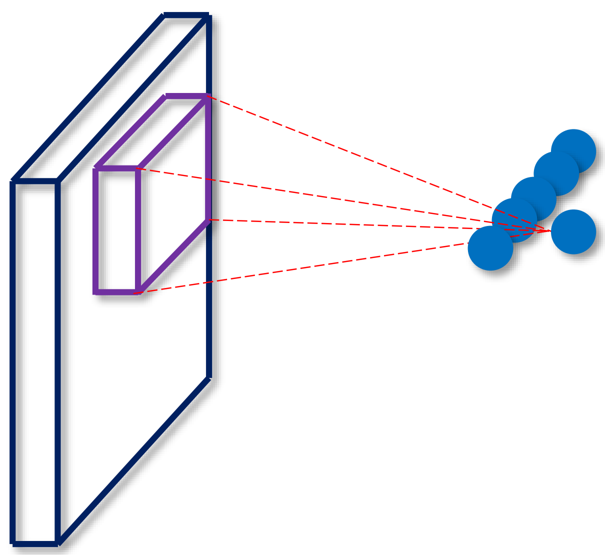
- Given an image and a filter: both tensor
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step



$32 \times 32 \times 3$

Convolution, abstractly!

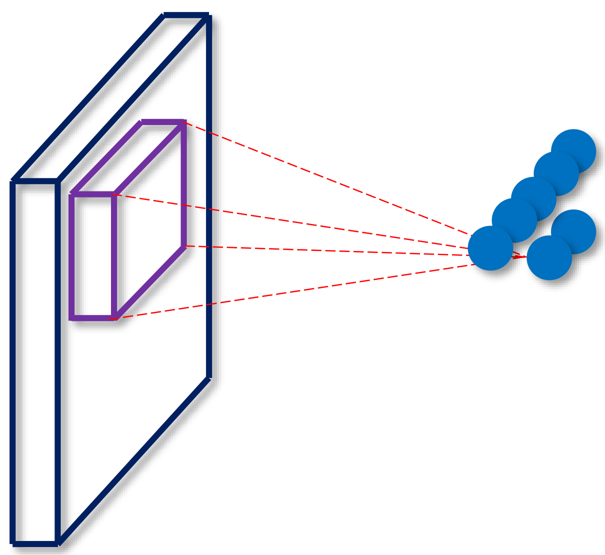
- Given an image and a filter: both tensor
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step



$32 \times 32 \times 3$

Convolution, abstractly!

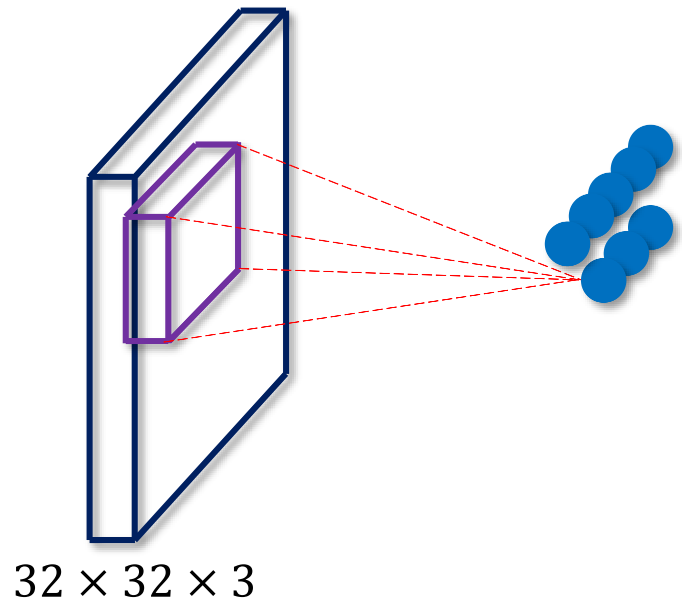
- Given an image and a filter: both tensor
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step



$32 \times 32 \times 3$

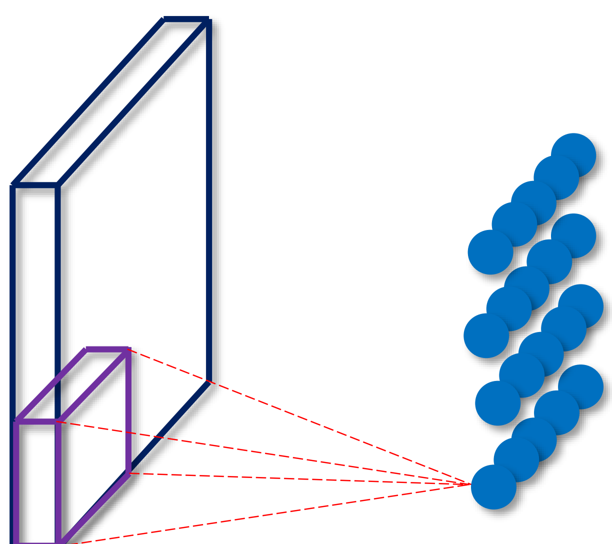
Convolution, abstractly!

- Given an image and a filter: both tensor
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step



Convolution, abstractly!

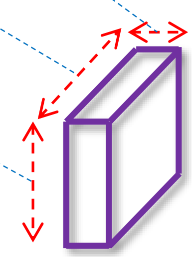
- Given an image and a filter: both tensor
  - Convolving is to use the filter to *sweep* the image *spatially*, and compute convolution value at each step



$32 \times 32 \times 3$

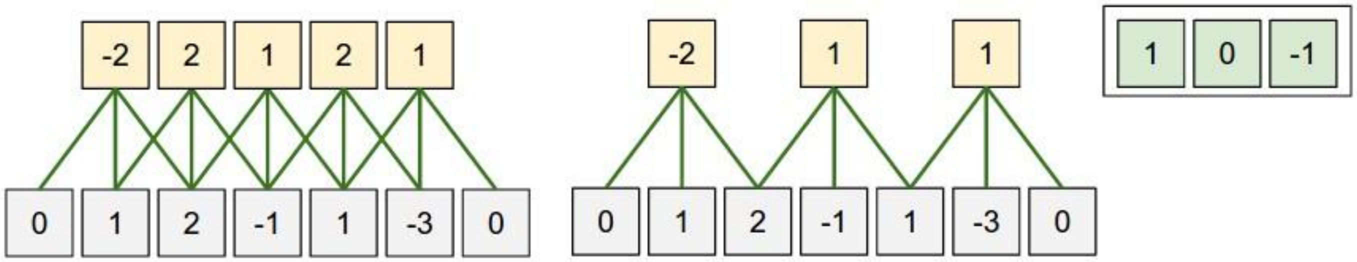
Bite the bones!
Fight the Notations!

- filter/Kernel size
  - Height, width, depth



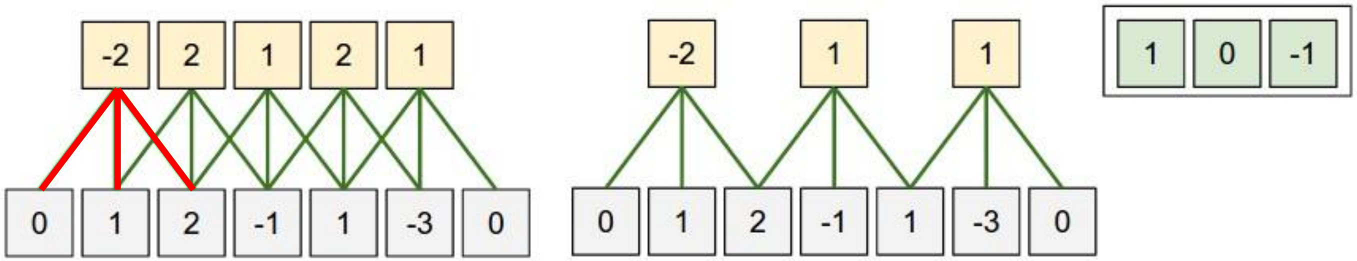
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



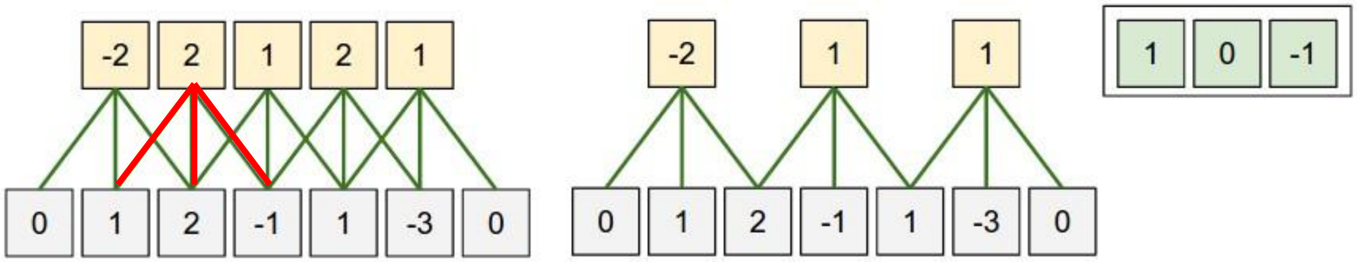
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



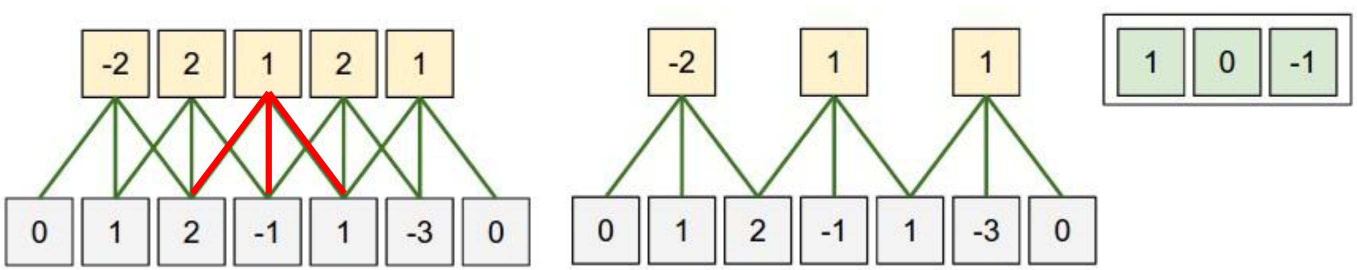
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



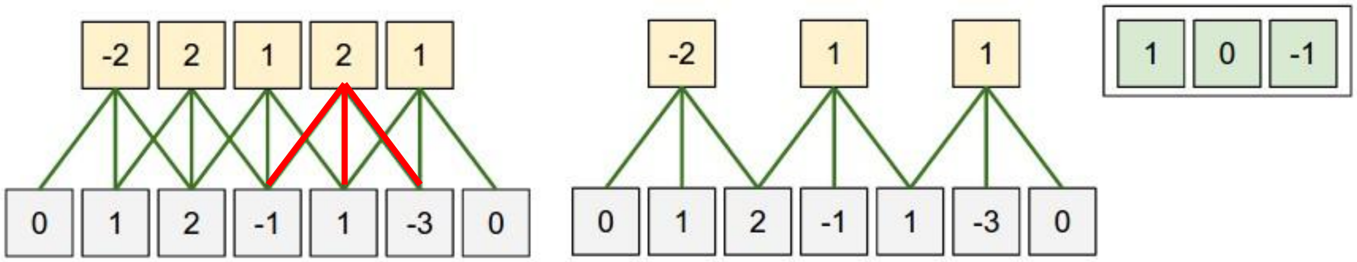
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



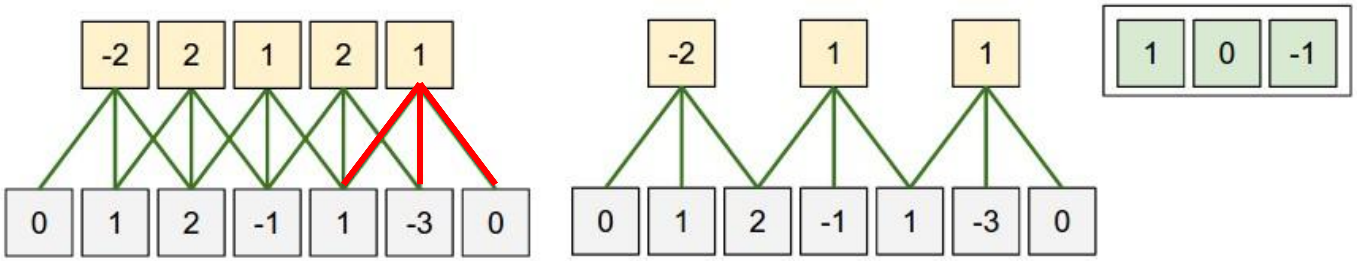
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



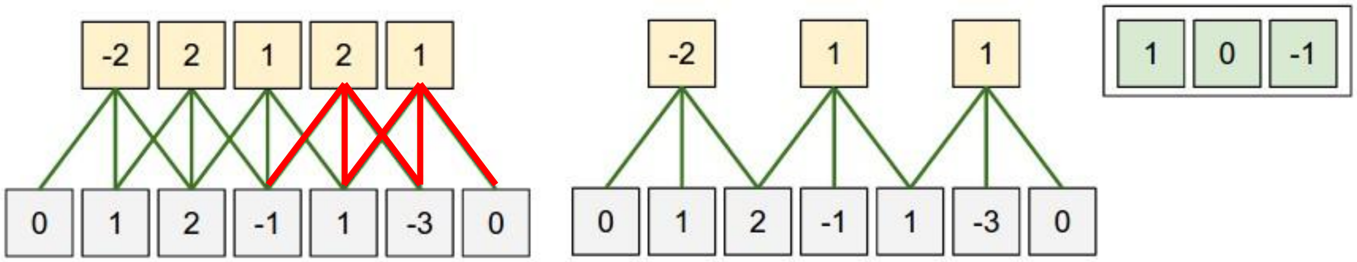
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



Bite the bones!
Fight the Notations!

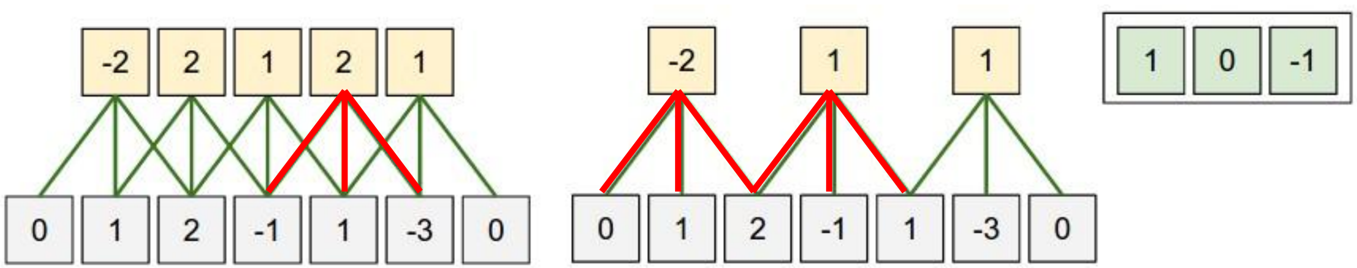
- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



Stride 1

Bite the bones!
Fight the Notations!

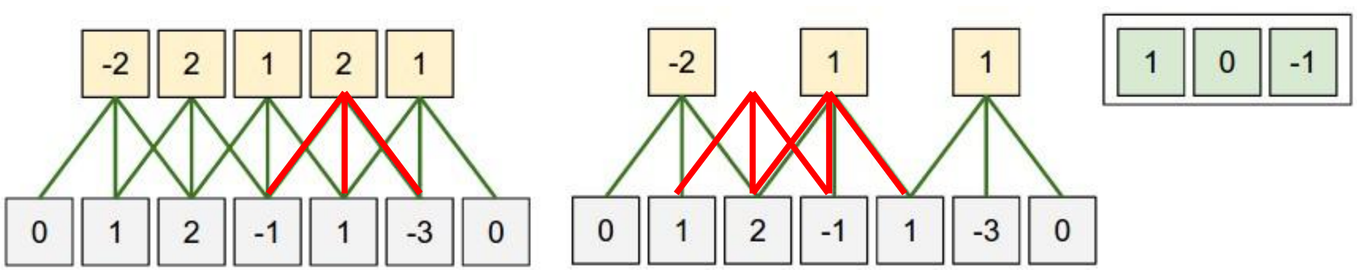
- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



Stride 1

Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image

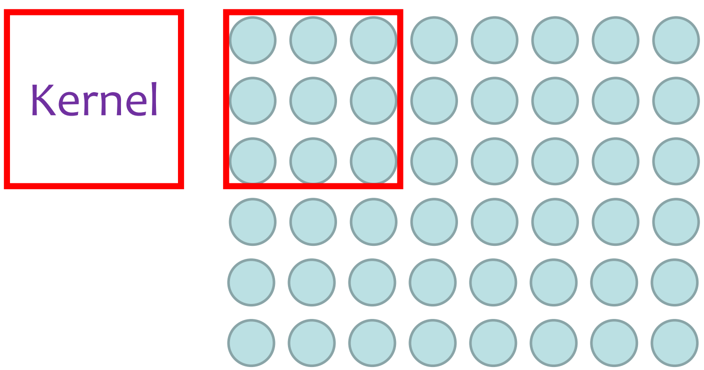


Stride 1

Stride 2

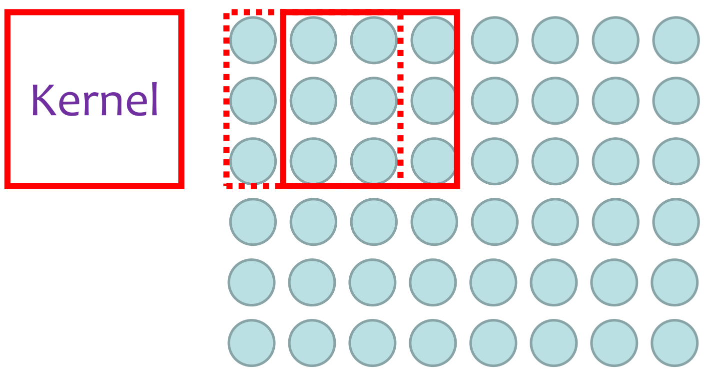
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



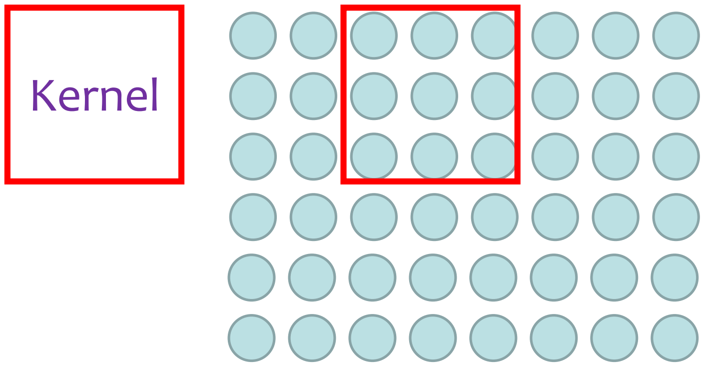
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



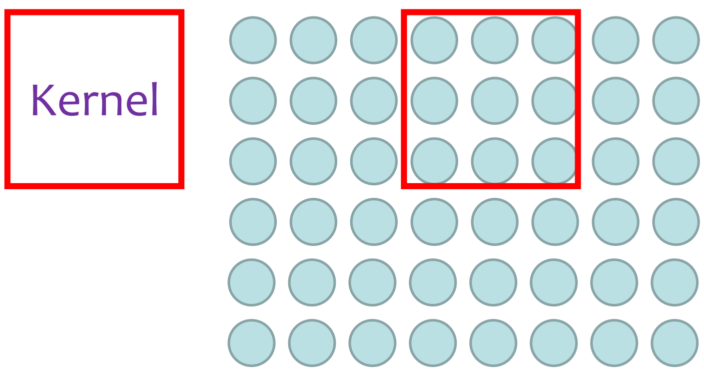
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



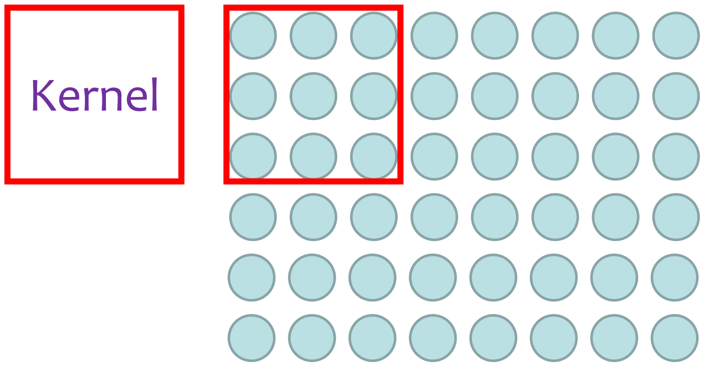
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



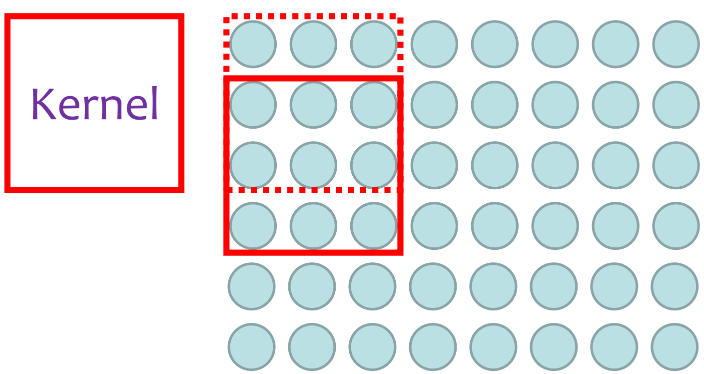
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



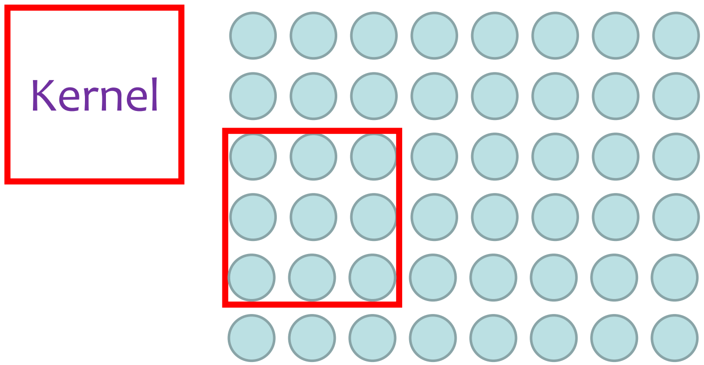
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



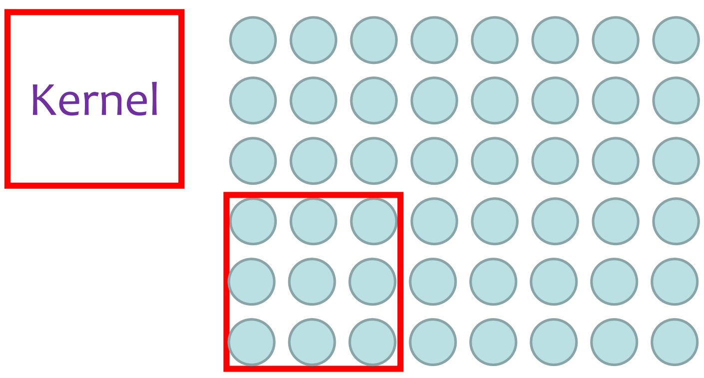
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



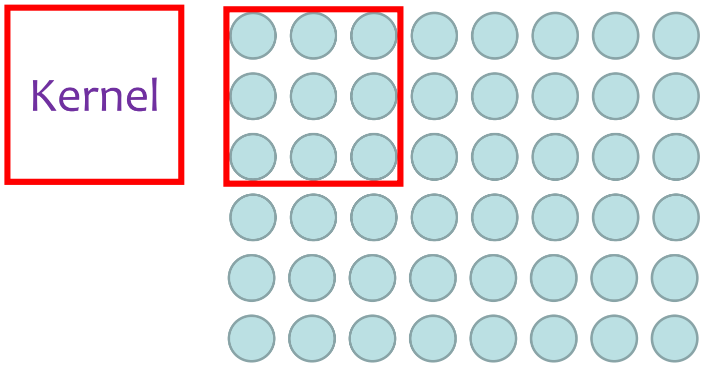
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



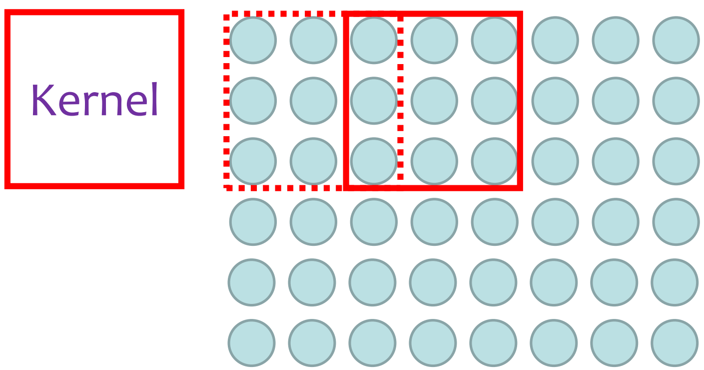
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



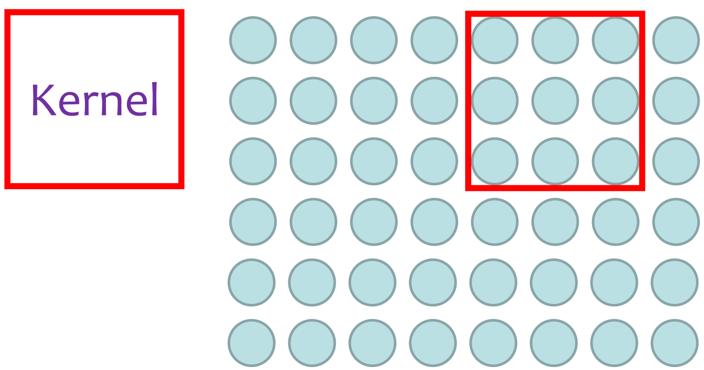
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



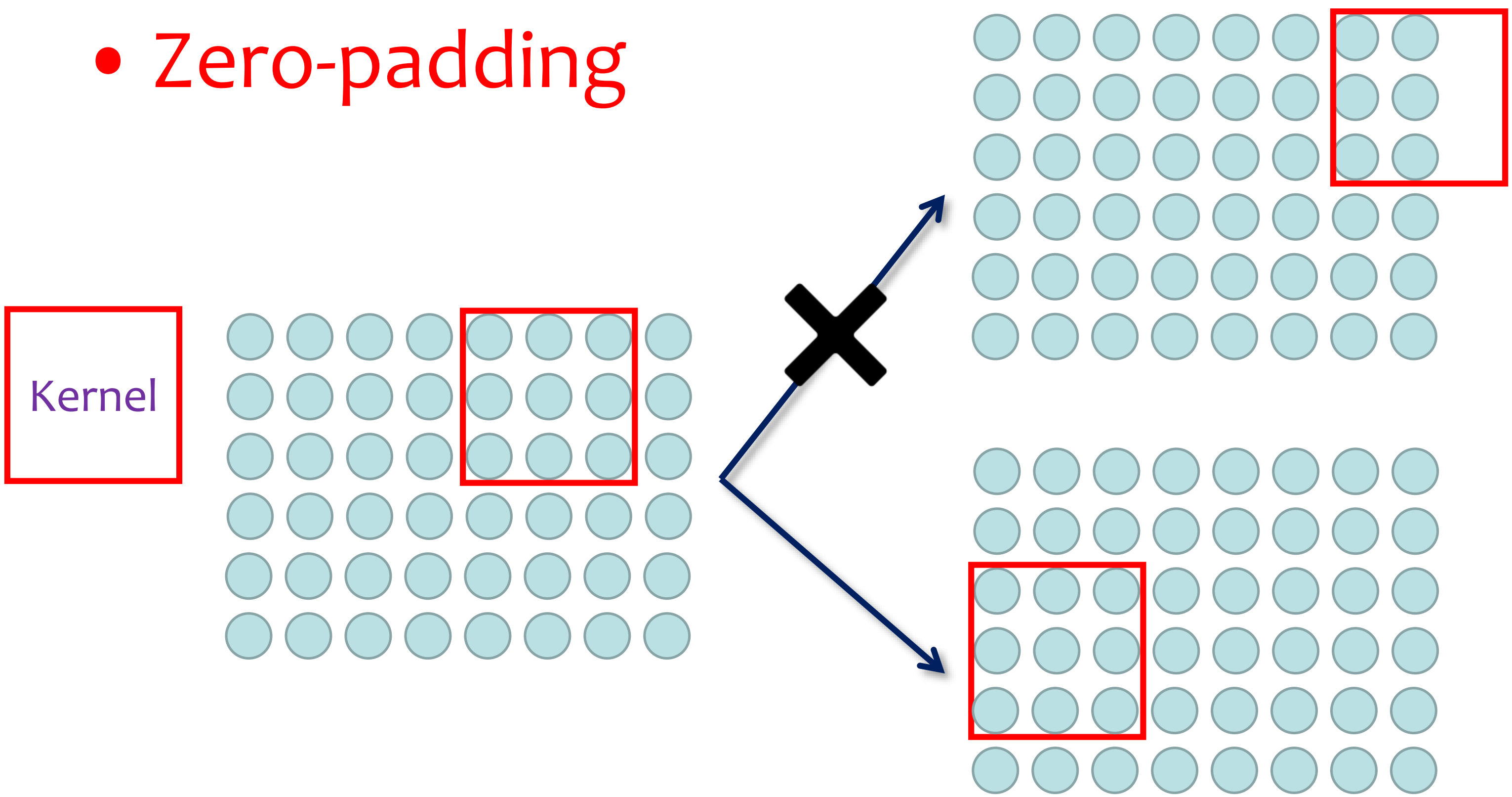
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
  - The *step size* you take the filter to sweep the image



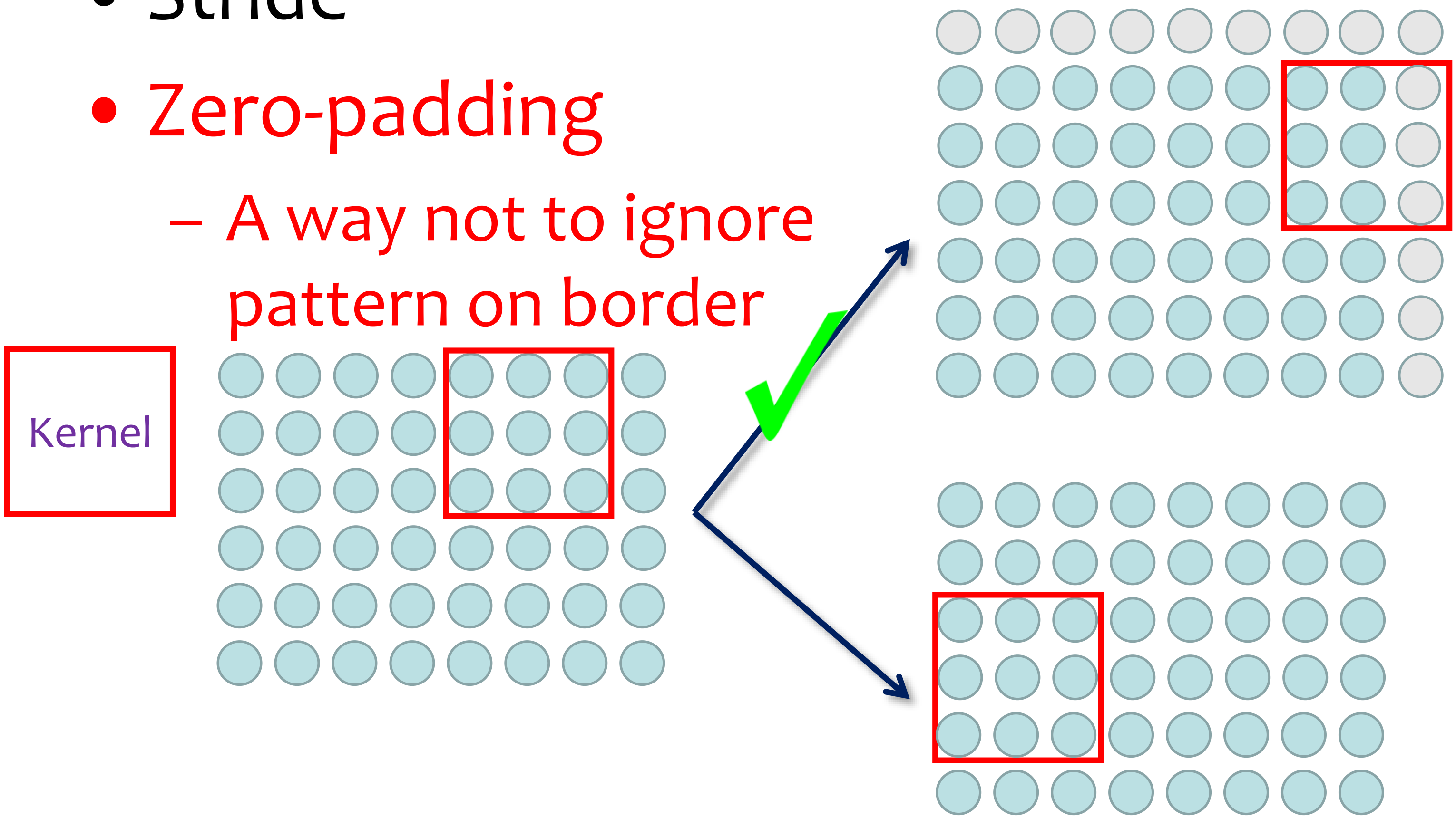
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
- Zero-padding



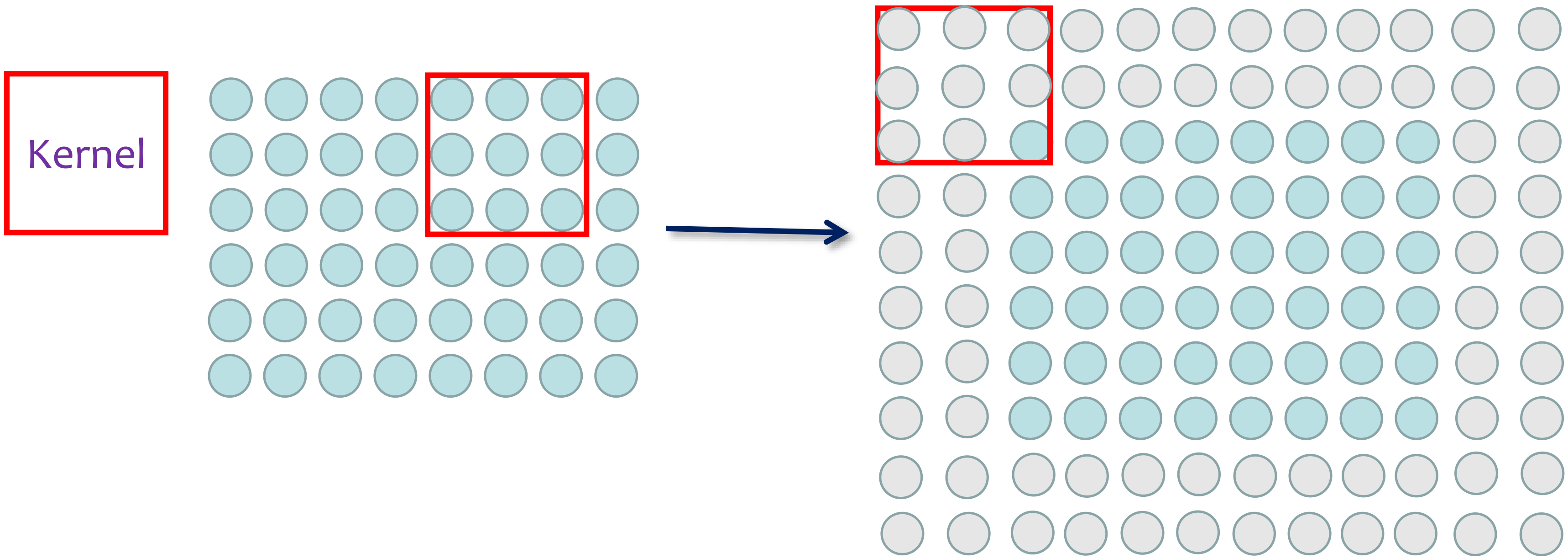
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
- Zero-padding
  - A way not to ignore pattern on border



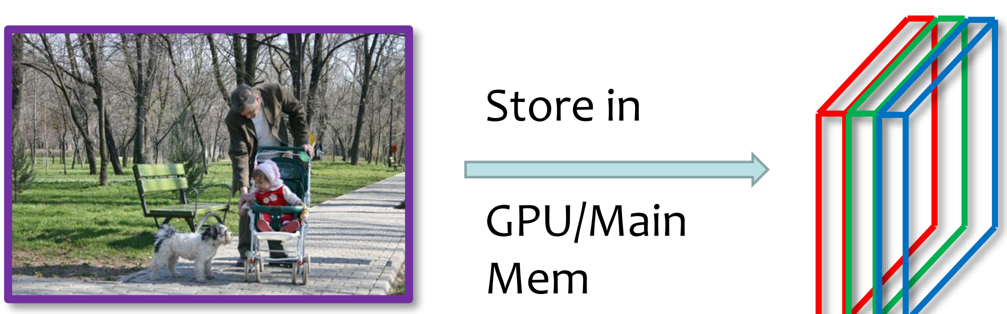
Bite the bones!
Fight the Notations!

- filter/Kernel size
- Stride
- Zero-padding
  - Padding can be adapted to kernel size



Bite the bones!
Other terms

- Channel
  - A **channel** in this context is the grayscale image of the same size as a color image, made of just one of these primary colors.
    - An RGB image has three channels: red, green, and blue. RGB channels roughly follow the color receptors in the human eye, and are used in computer display and image scanner.



Bite the bones!
Other terms

- Channel

A 24-bit RGB image

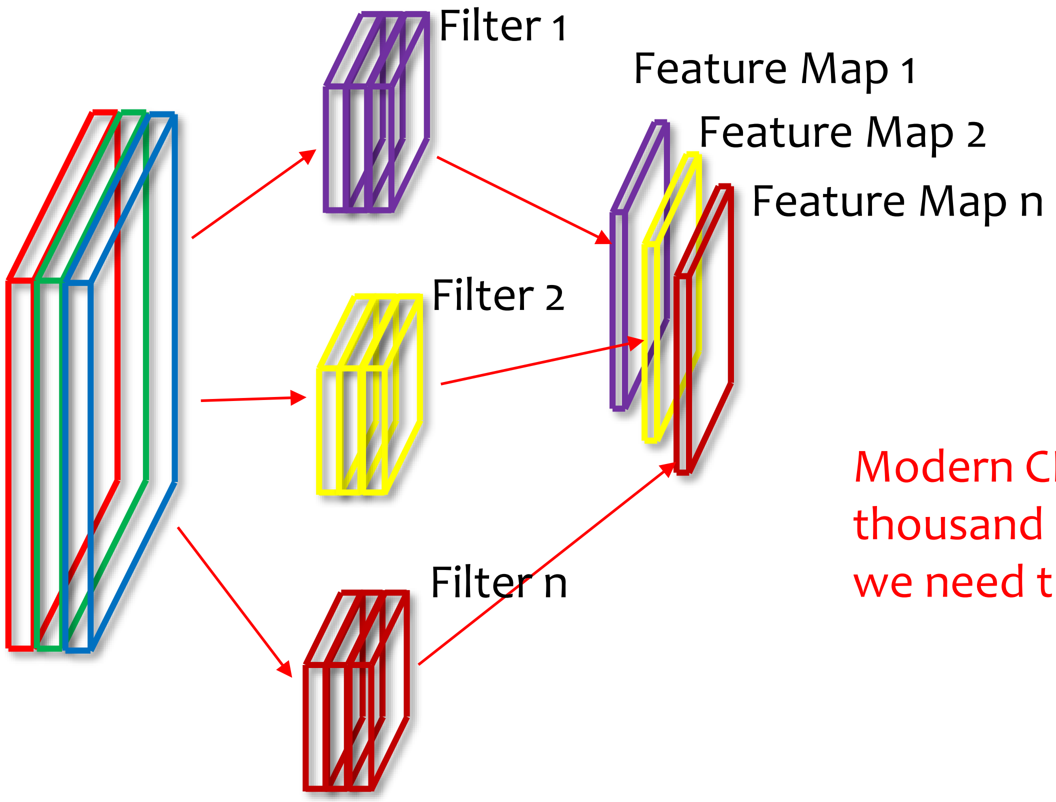
The RED channel of the original RGB image (converted to greyscale for easier viewing)

The GREEN channel of the original RGB image (converted to greyscale for easier viewing)

The BLUE channel of the original RGB image (converted to greyscale for easier viewing)

Bite the bones!
Other terms

- Channel
- #filter⇒#Feature map

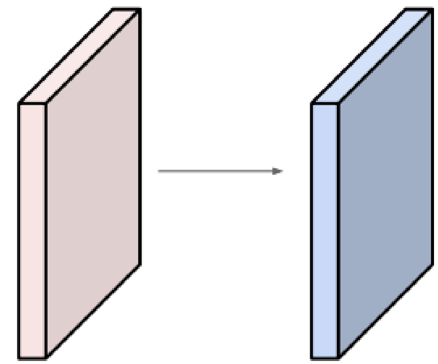


Modern CNN has hundred of thousand of filters, so why we need that much?

Bonus! Let us Relax

Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

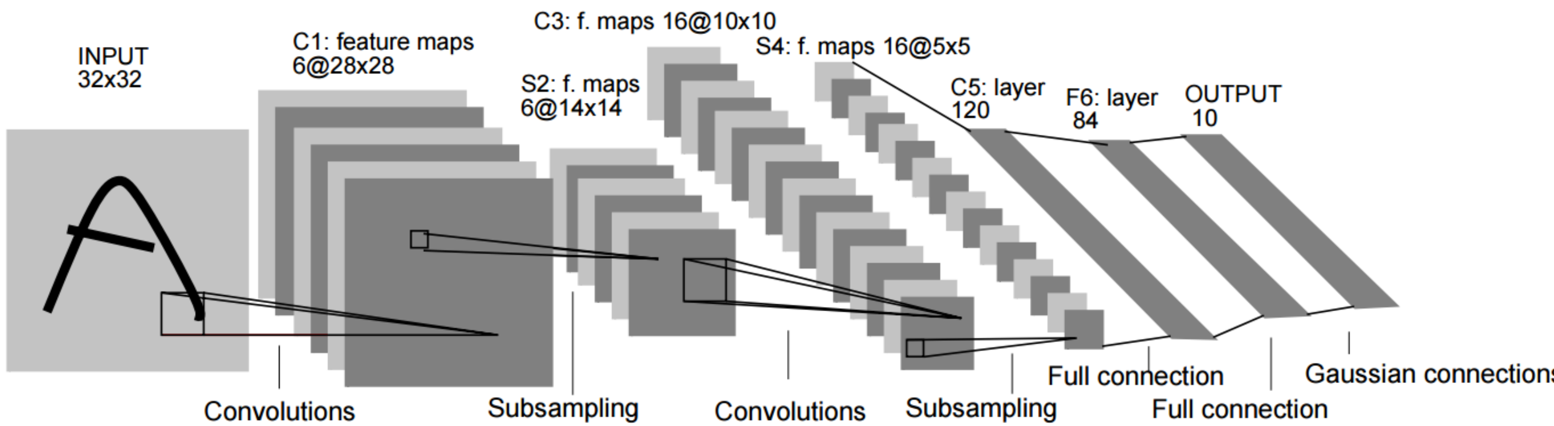


Output volume size: ?

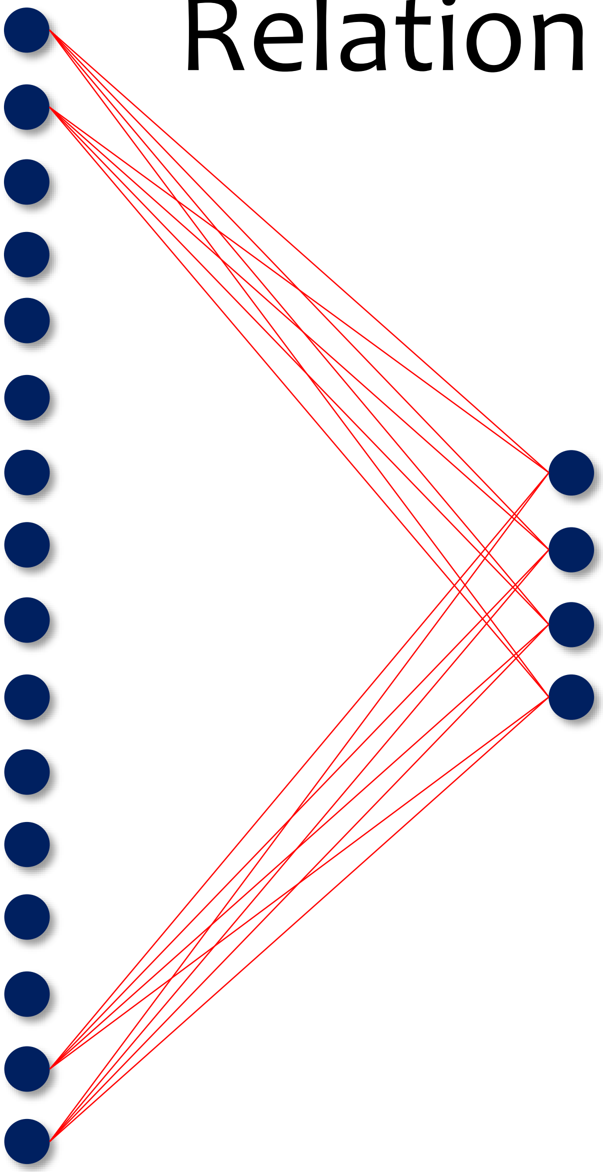
Courtesy slide, http://cs231n.stanford.edu/slides/winter1516_lecture7.pdf

The position of convolution

- It can be anywhere while there are enough region to extract features
- Convolution is always the first to be applied to a raw image (*DATA*) matrix



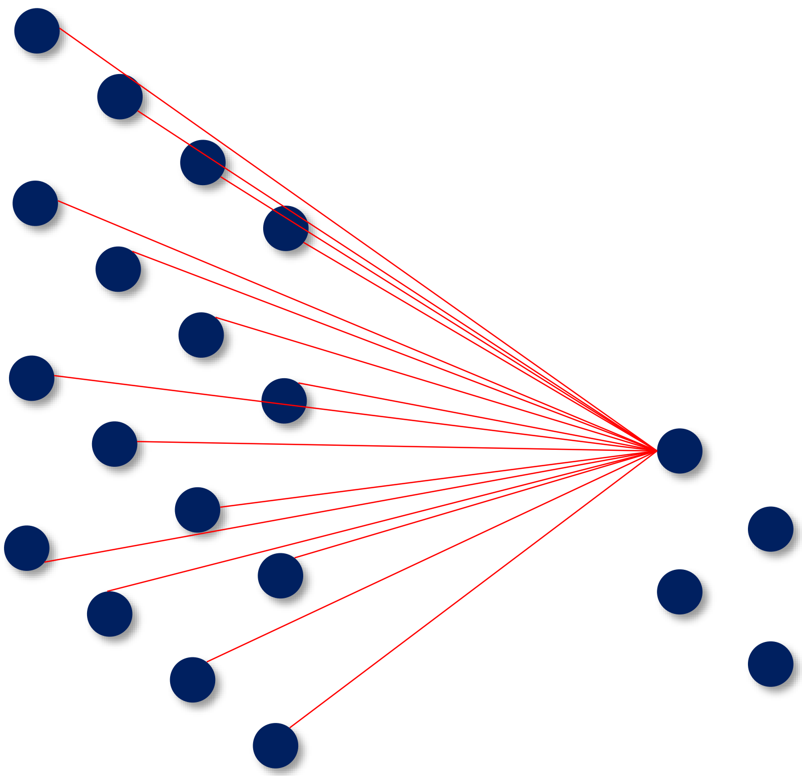
Relation to FFNN, again!



- Let us visualize using FFNN for zip code image classification.

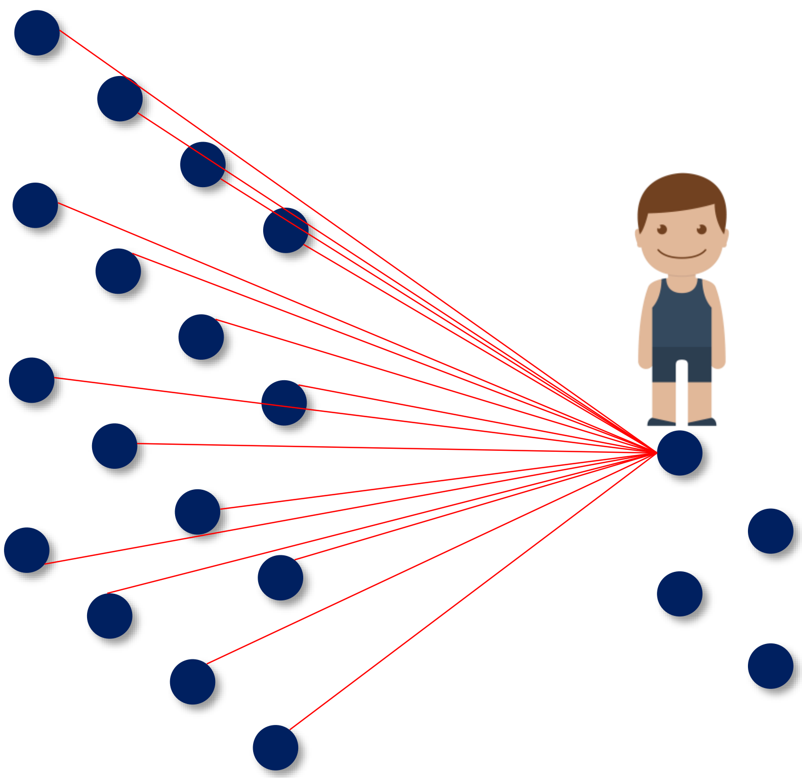
Relation to FFNN, again!

- Let us rearrange them to look more handsome



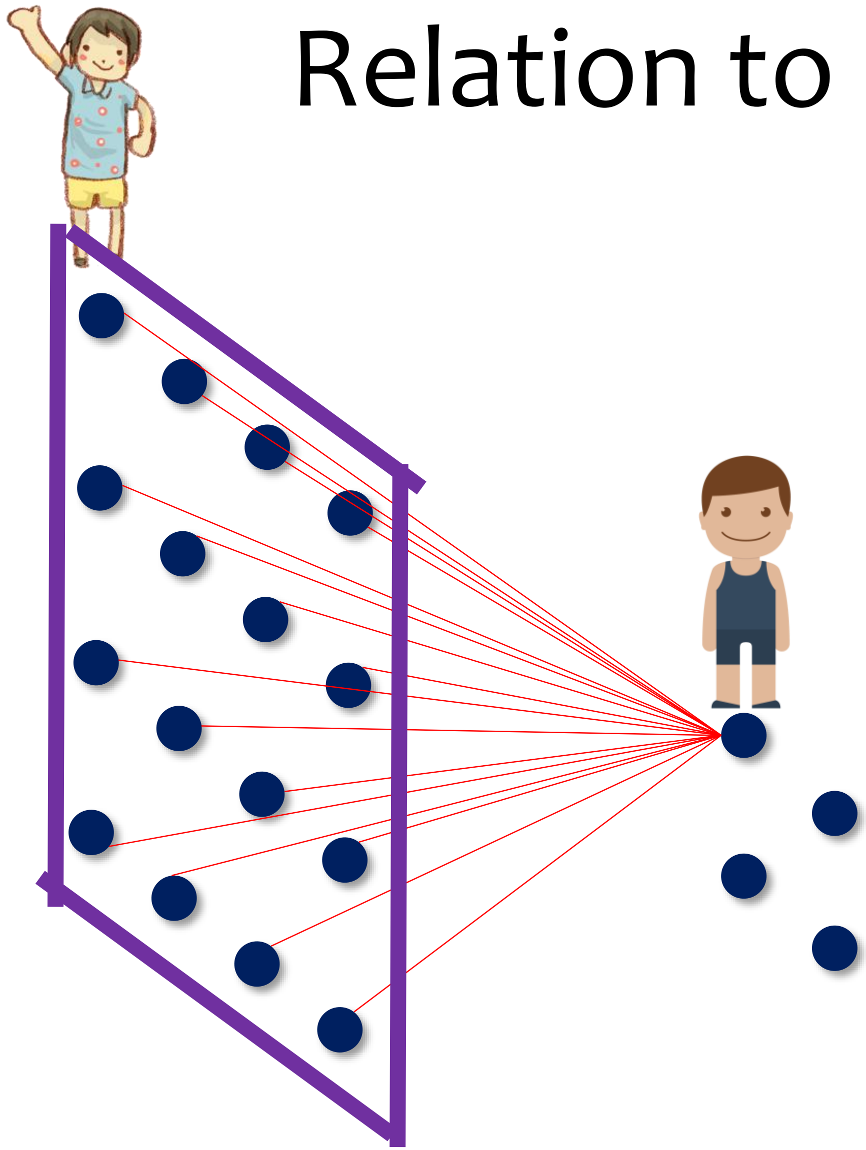
- Every neuron in 2nd layer will look into *all* signals from last layer
- And every red line is *distinctive* with its weight

Relation to FFNN, again!



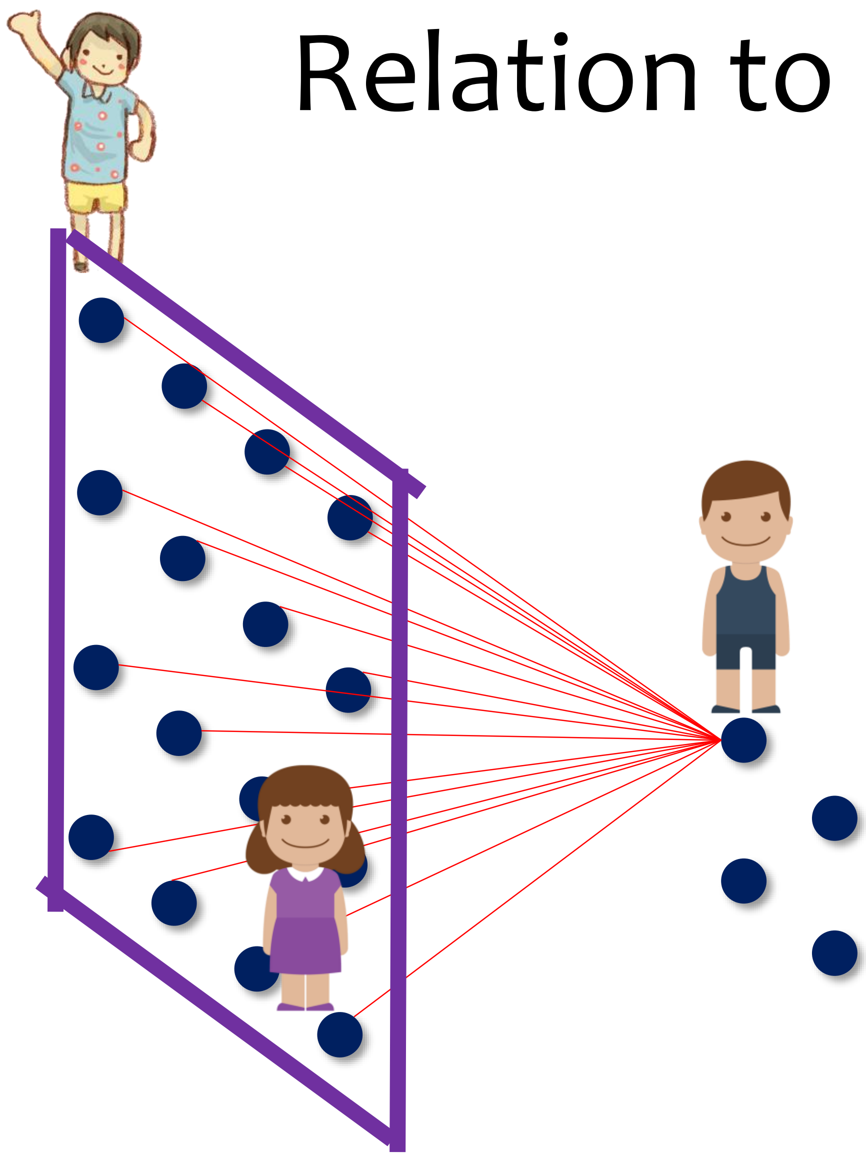
- Intuition:
  - At every neuron there is a *small man* exchange information with men in the last layer
  - To find his *special interest*, and then pass on his *observation*

Relation to FFNN, again!



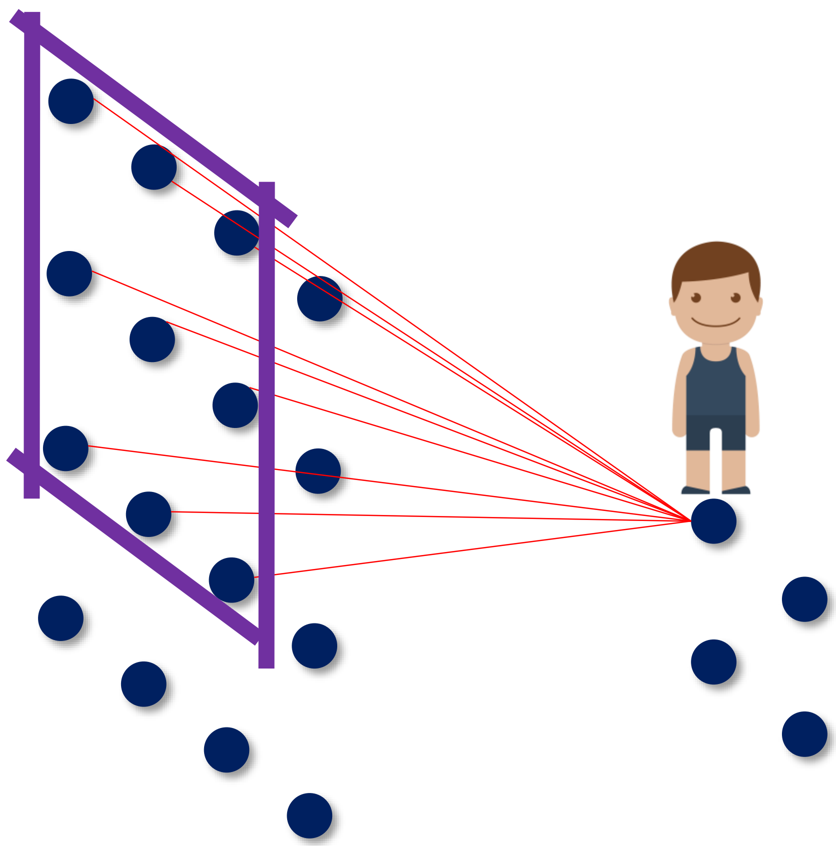
- Intuition:
  - Every time, he only analysis info gained within a *local neighborhood*
  - Since *neighborhood* has much info to share in common
  - *Men from remote districts have few thing to share*

Relation to FFNN, again!



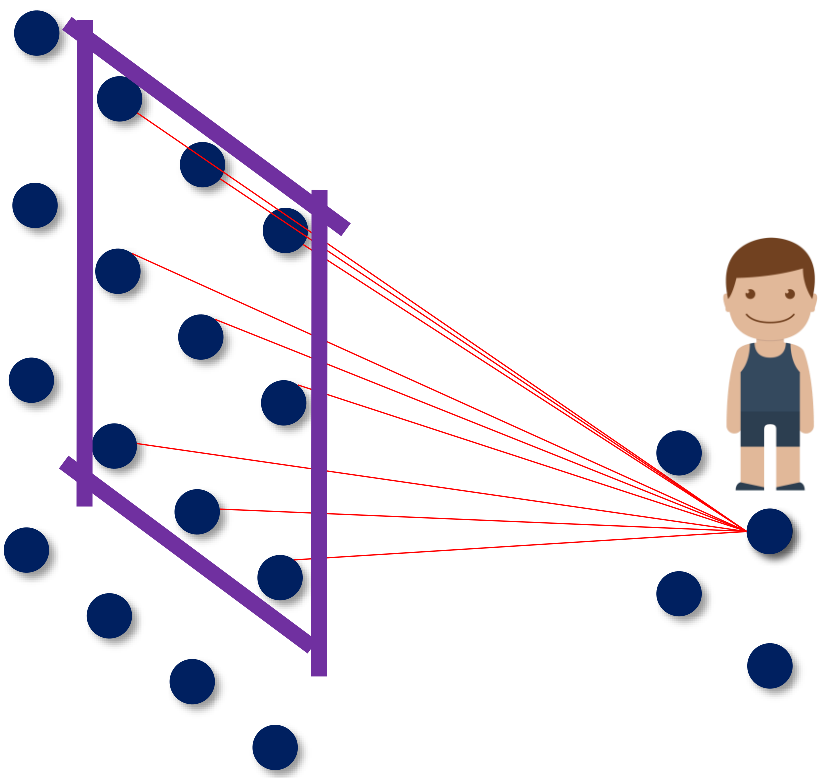
- Intuition:
  - Every time, he only analysis info gained within a *local neighborhood*
  - Since *neighborhood* has much info to share in common
  - *Men from remote districts have few thing to share*

Relation to FFNN, again!



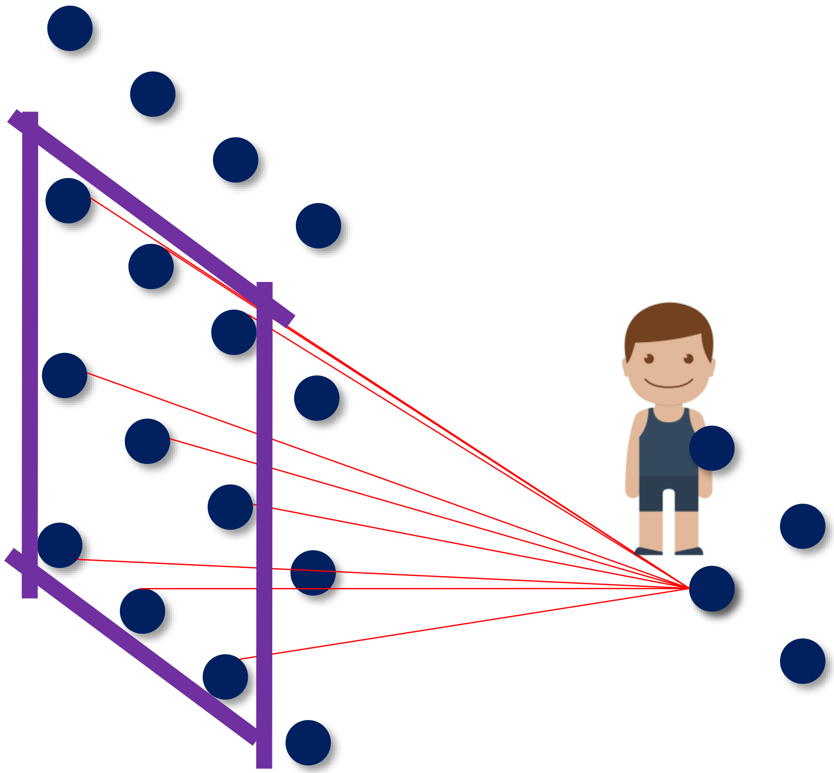
- Intuition:
  - Actually a man with special interest is more like a bunch of *weights* instead of a neuron
  - His *knowledge* is *encoded* in the weights

Relation to FFNN, again!



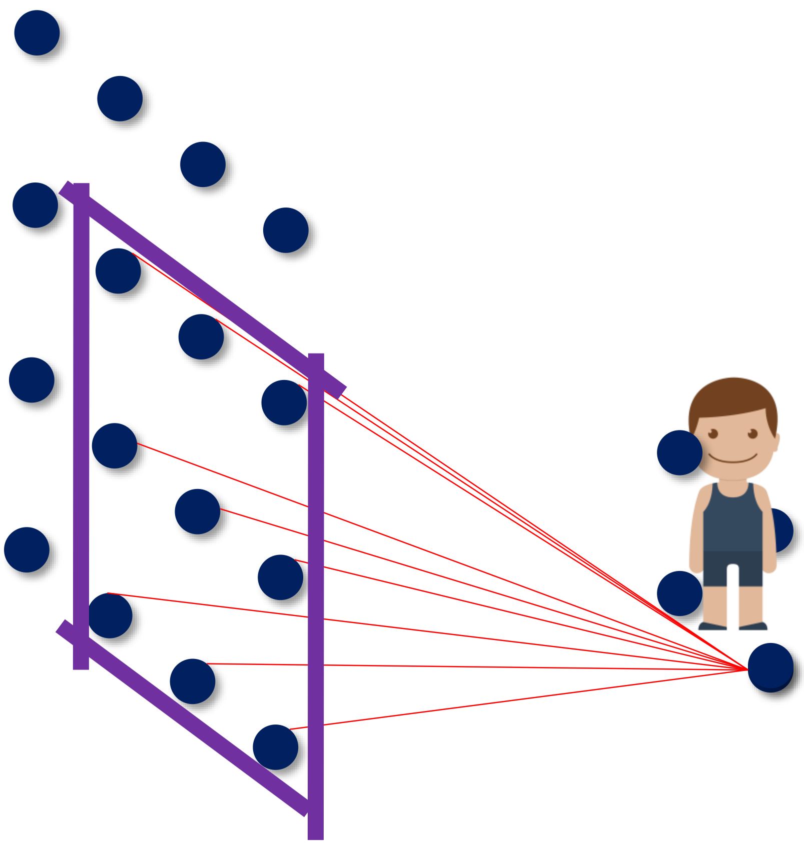
- Intuition:
  - So he use his *knowledge* to investigate *all* the community divided into neighbors

Relation to FFNN, again!



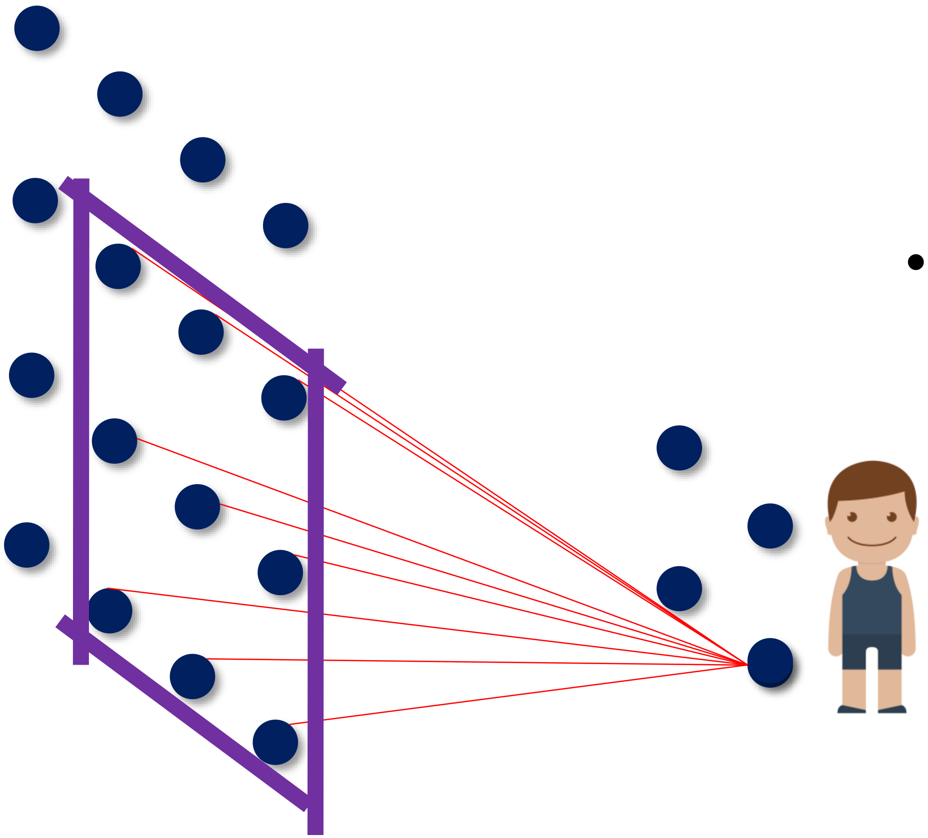
- Intuition:
  - So he use his *knowledge* to investigate *all* the community divided into neighbors

Relation to FFNN, again!



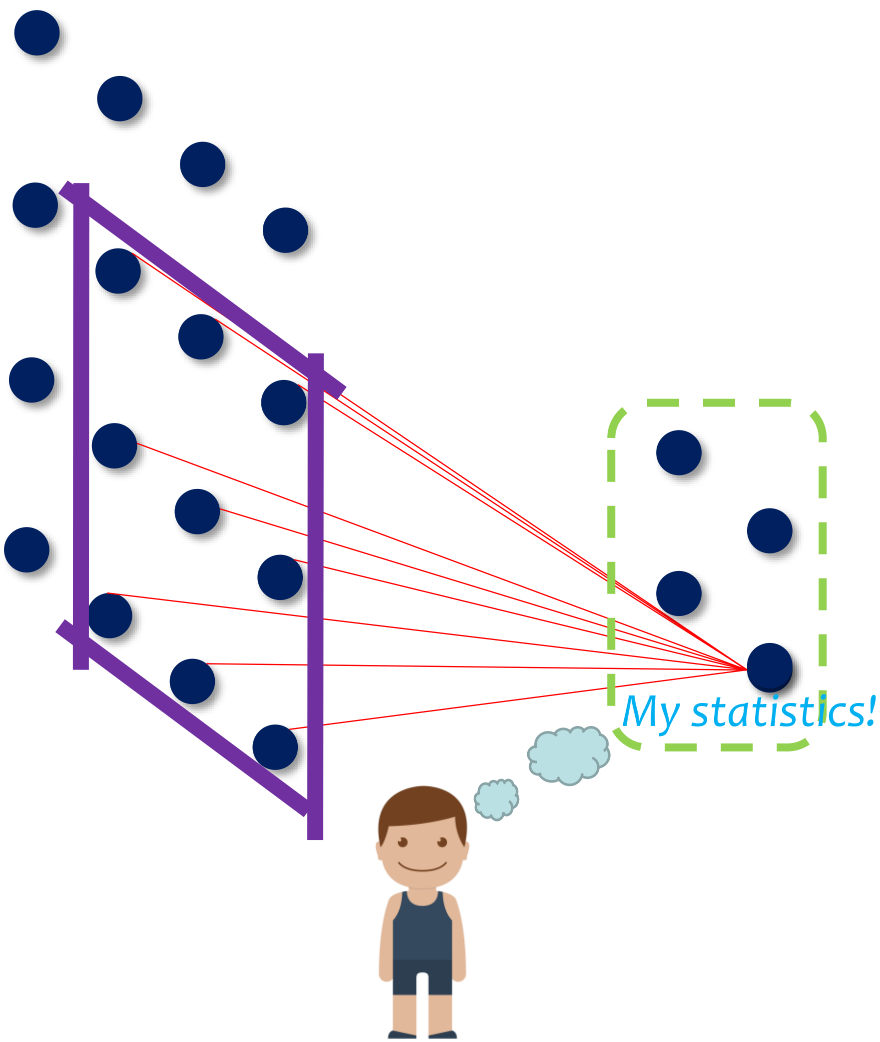
- Intuition:
  - So he use his *knowledge* to investigate the *whole* city which is divided into neighbors

Relation to FFNN, again!



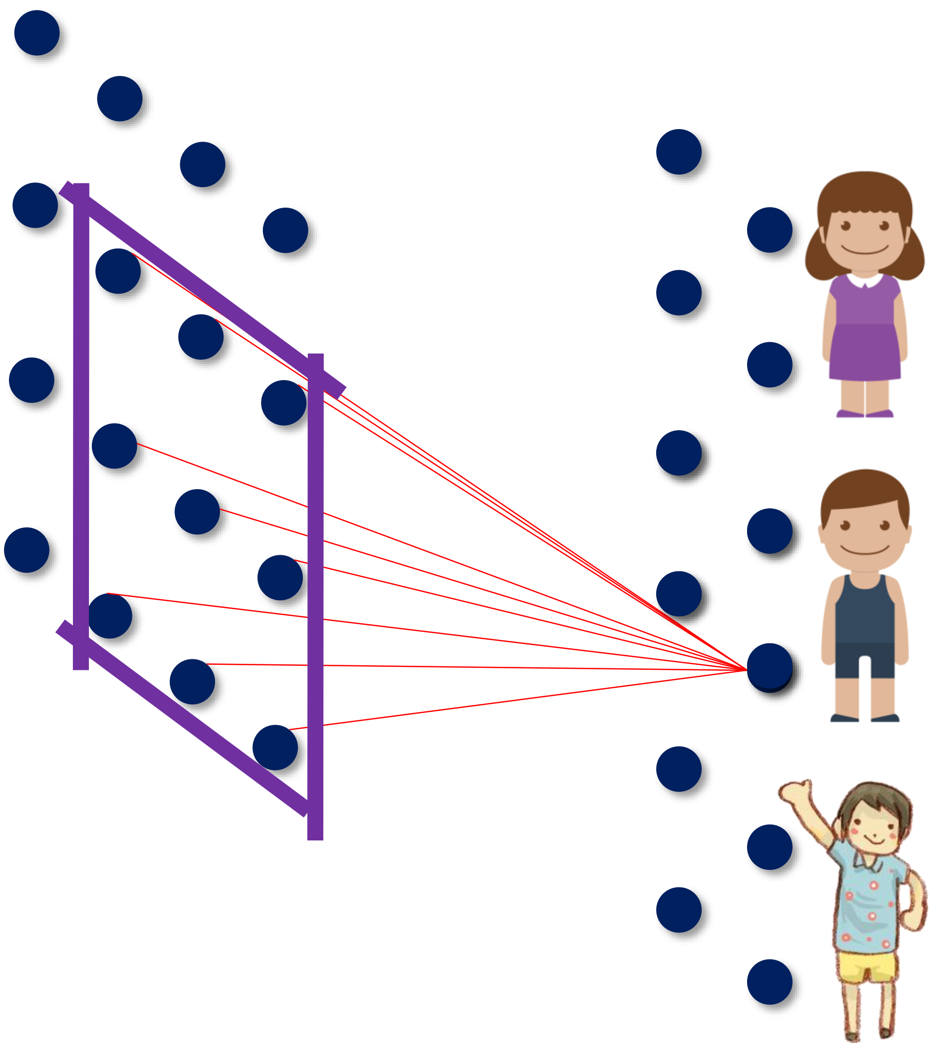
- Intuition:
  - So he use his *knowledge* to investigate *all* the community divided into neighbors
  - And produce his *statistics!*

Relation to FFNN, again!



- Intuition:
  - One man is weak

Relation to FFNN, again!

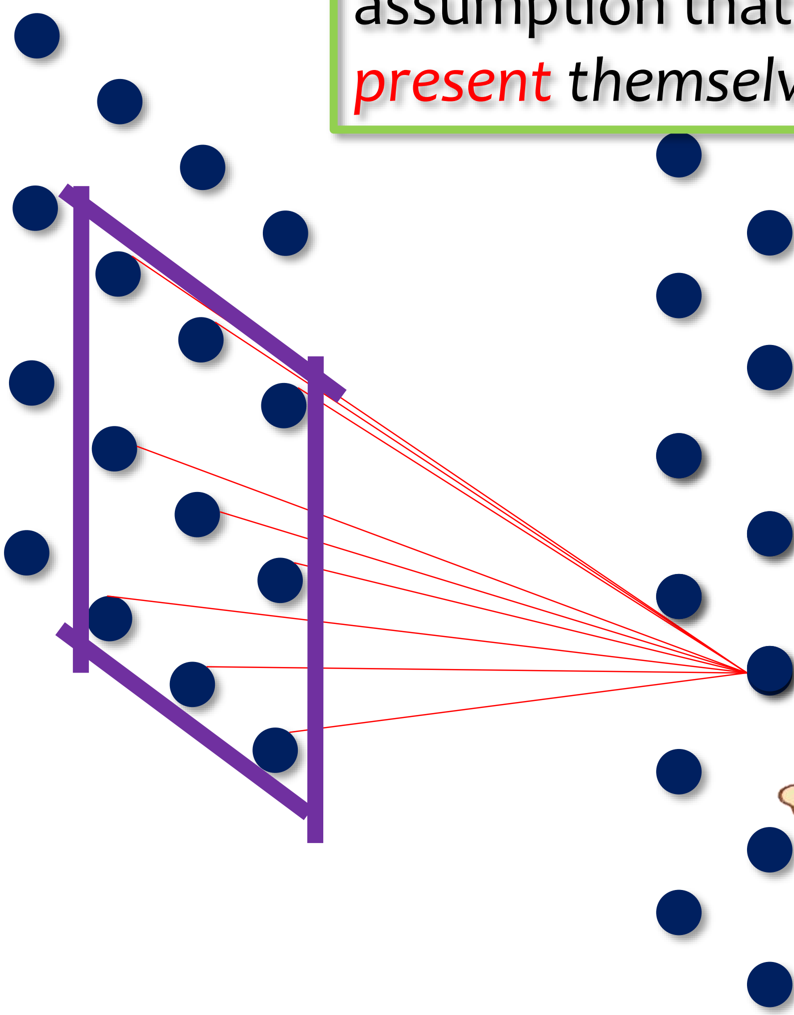


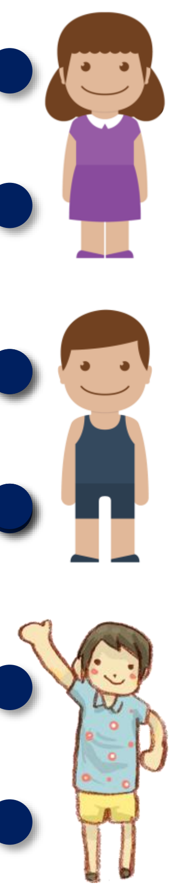
Intuition:

- One man is weak, but a *group of man is strong*!
- They have *distinctive* knowledge (weights)
- One man *share* his knowledge across the whole community!

Relation to FFNN, again!

All this should work under the assumption that *patterns locally present themselves*





Intuition:

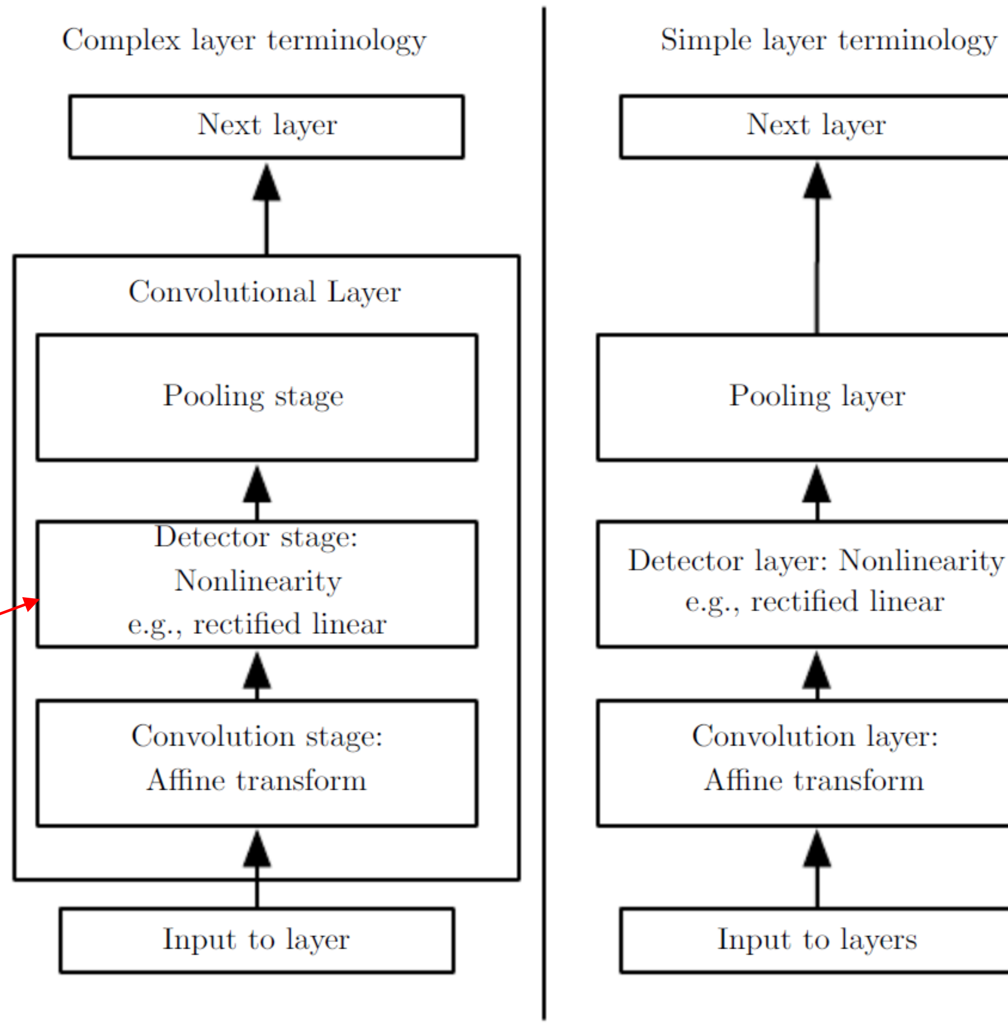
- One man is weak, but a *group of man is strong*!
- They have *distinctive* knowledge (weights)
- One man *share* his knowledge across the whole community!

Outline

- Warming up
- Convolution
  - Filter, kernel
  - Deconvolution
- Pooling
  - Subsampling
  - Invariance
- Convolutional Neural Net
  - Architecture

Typical Stages of A CNN Layer

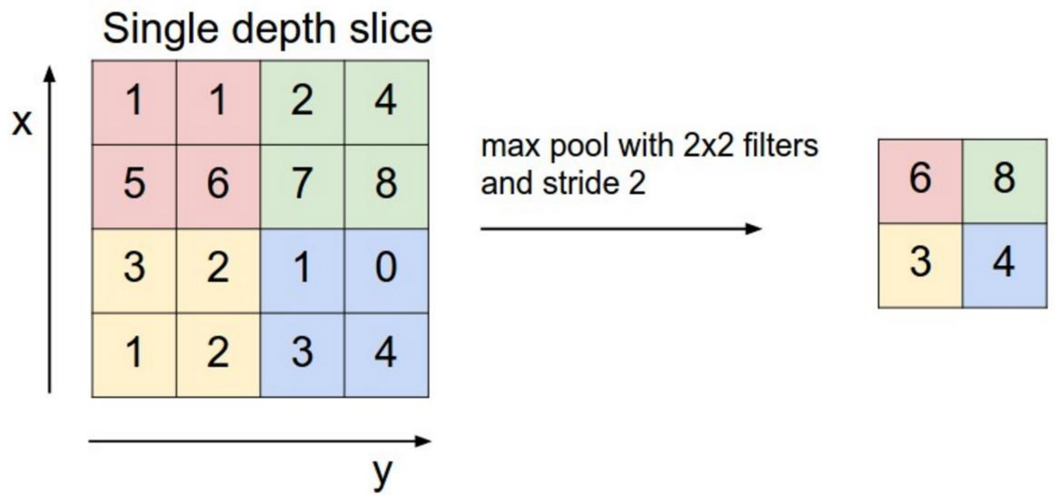
- Three stages
  - Convolution stage
  - Detector stage: Activation
  - Pooling stage



Hah! Non-linearity is here!

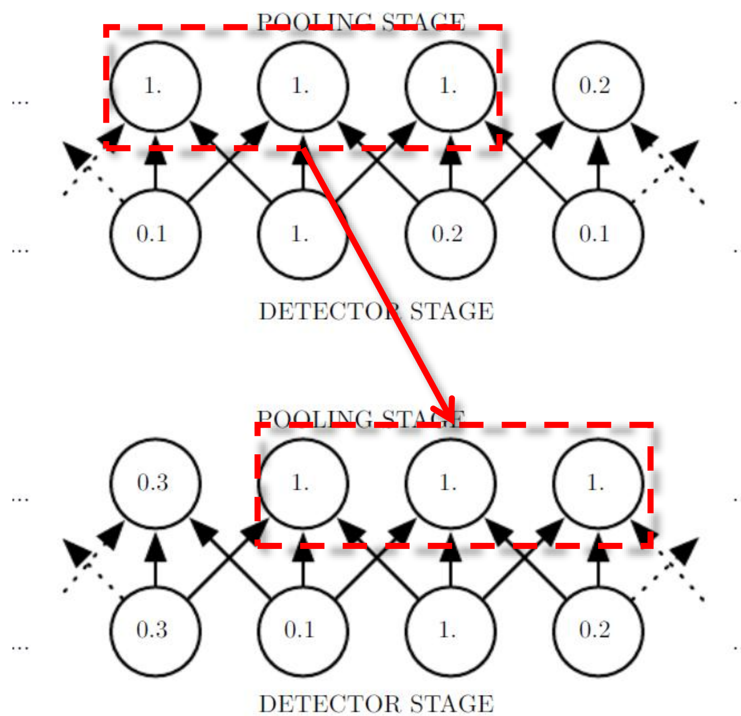
Pooling

- Pooling layers *subsample* their input
  - Spatial pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map
  - Retains the most important information



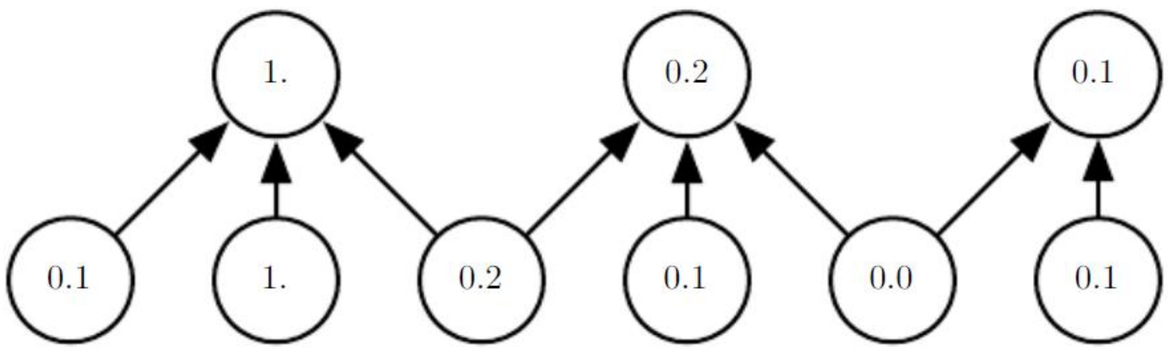
Gandalf, the Gray!
Pooling, the Invariance!

- Pooling is *unsensitive* to local *translation*.
  - "if we translate the input by a small amount, the values of most of the pooled outputs do not change."



Gandalf, the Gray!
Pooling, the Accelerator

- Pooling *ease* the *computational burden* of next layer.



"Stride 2 max-pooling" Figure 9.10 Deep Learning Book

"This reduces the computational and statistical burden on the next layer"

Gandalf, the Gray!
Pooling, the Flexible

- Pooling should be *designed* to *fit* specific applications.
  - Max pooling
  - Average pooling
  - Min pooling
  - l_2-norm pooling
  - Dynamic k-pooling
  - Etc.

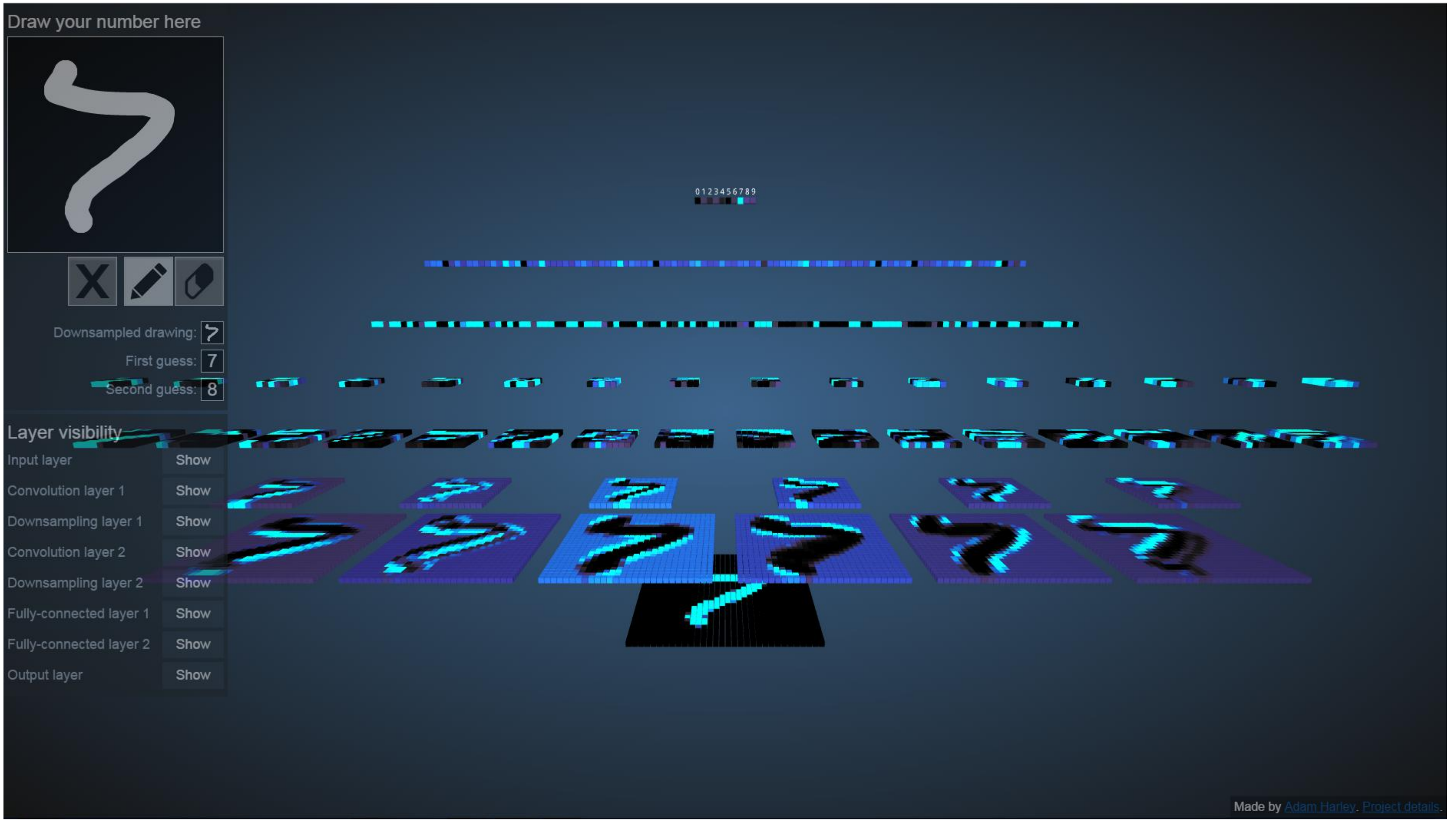
Properties of Pooling

- Makes the input representation (feature dim) smaller and more manageable
- Reduces number of parameters and computations in the network, therefor, controlling *overfitting*
- Makes the network invariant to small transformations, distortions and translations in the input image
- Help us arrive at almost scale invariant representation of our image

Outline

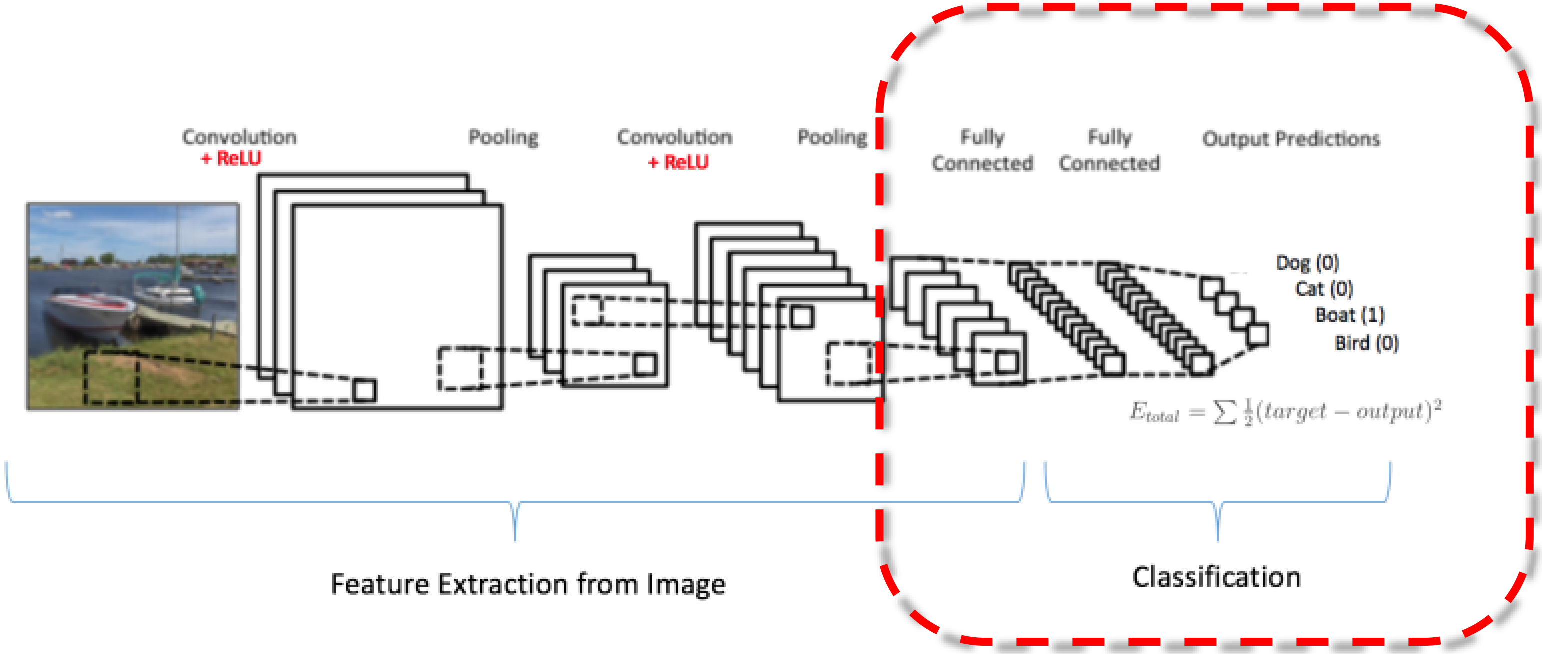
- Warming up
- Convolution
  - Filter, kernel
  - Deconvolution
- Pooling
  - Subsampling
  - Invariance
- Convolutional Neural Net
  - Architecture

An Visualization



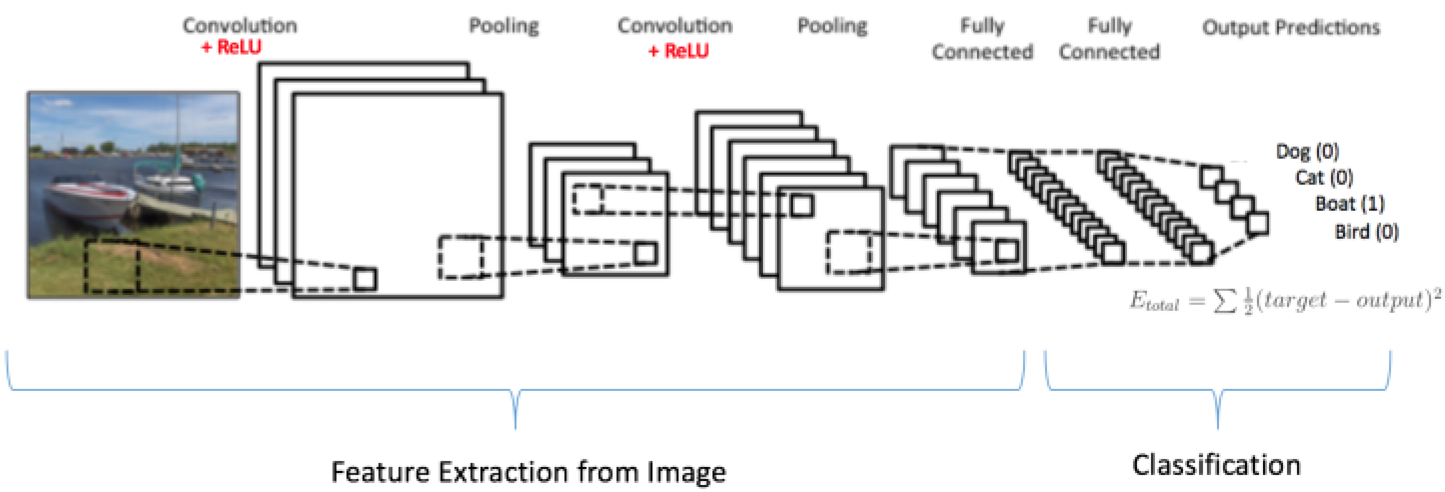
Fully Connected Layer

- FCLs are just naive FFNNs
  - In CNN architecture, it is used *integrate* info



CNN as a Whole

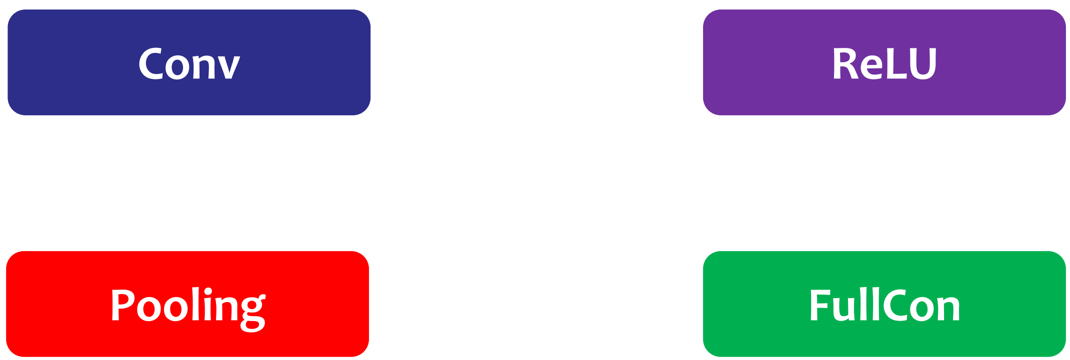
- So as a whole, CNN is composed of
  - Convolution
  - Nonlinearity: e.g. ReLU
  - Pooling
  - FC Layers



Courtesy image, https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

CNN as a Whole

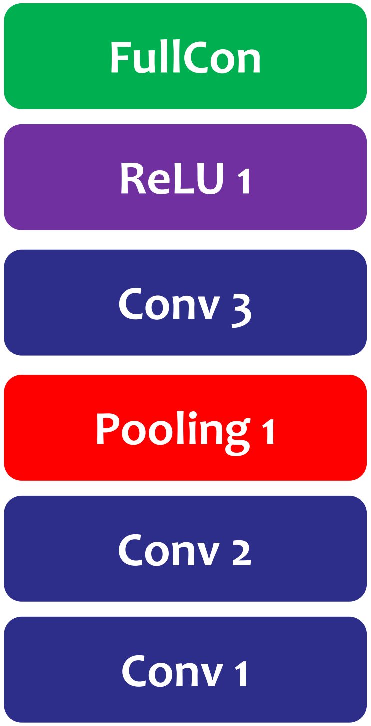
- If they are denoted using small icons, we can stack them almost arbitrarily

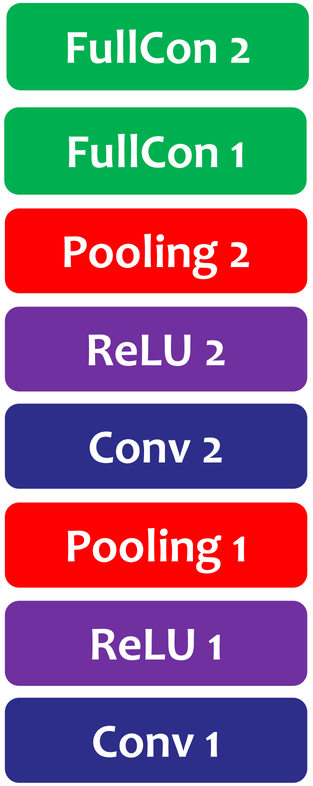


Courtesy image, https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

CNN as a Whole

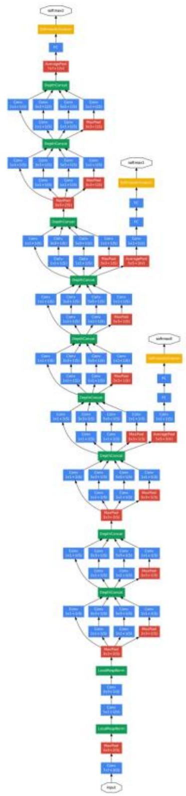
- If they are denoted using small icons, we can stack them almost arbitrarily

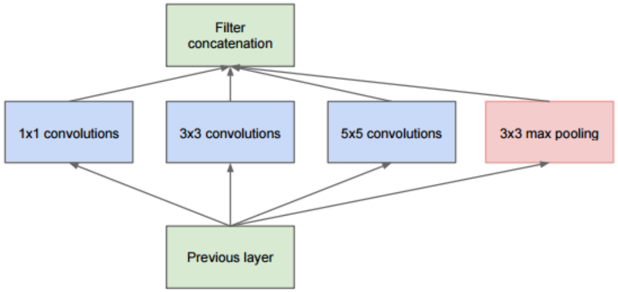




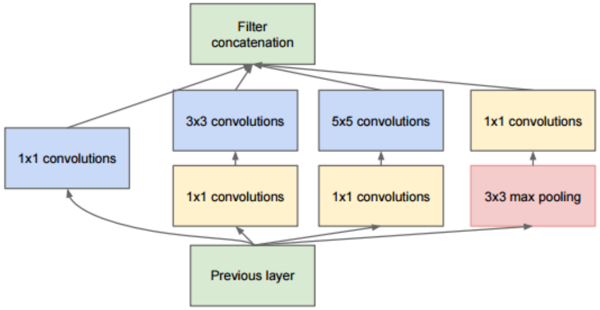
Courtesy image, https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Bonus: GoogLeNet

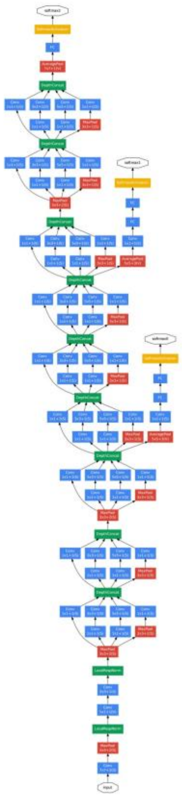


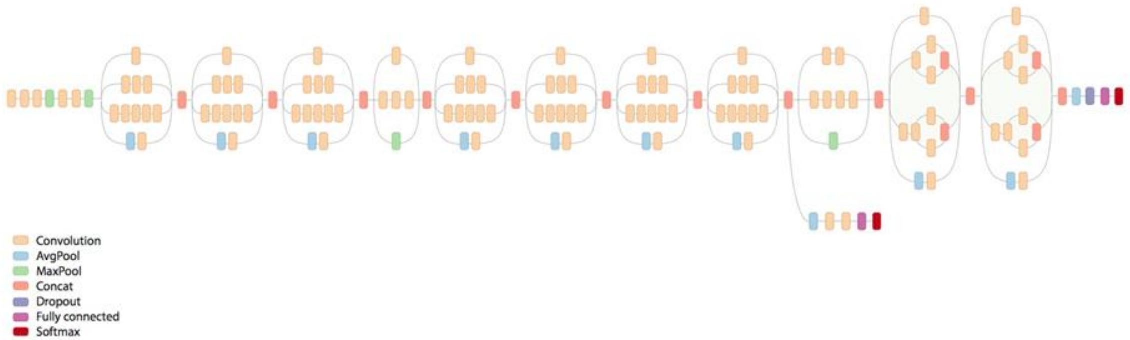


(a) Inception module, naïve version



(b) Inception module with dimensionality reduction





Another view of GoogLeNet's architecture.

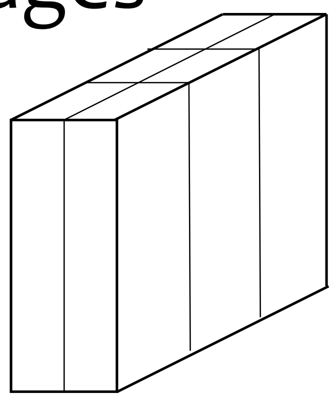
Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

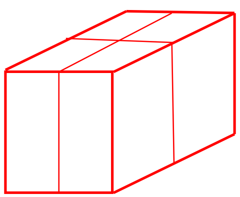
Going Deeper with Convolutions, CVPR 2015

Bonus: formalize CONV. & BP

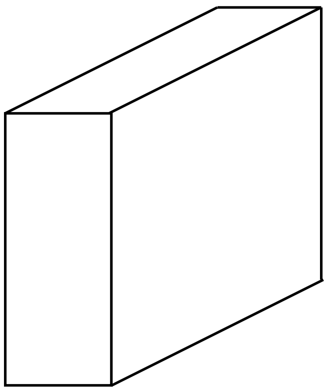
- Let $x \in \mathbb{R}^{N\times H\times W\times C}$ be N images, and $f \in \mathbb{R}^{I\times J\times C\times K}$ be K filters, with same #channel as images

$\mathbb{R}^{H\times W\times C}$





$\mathbb{R}^{H\times W\times C}$

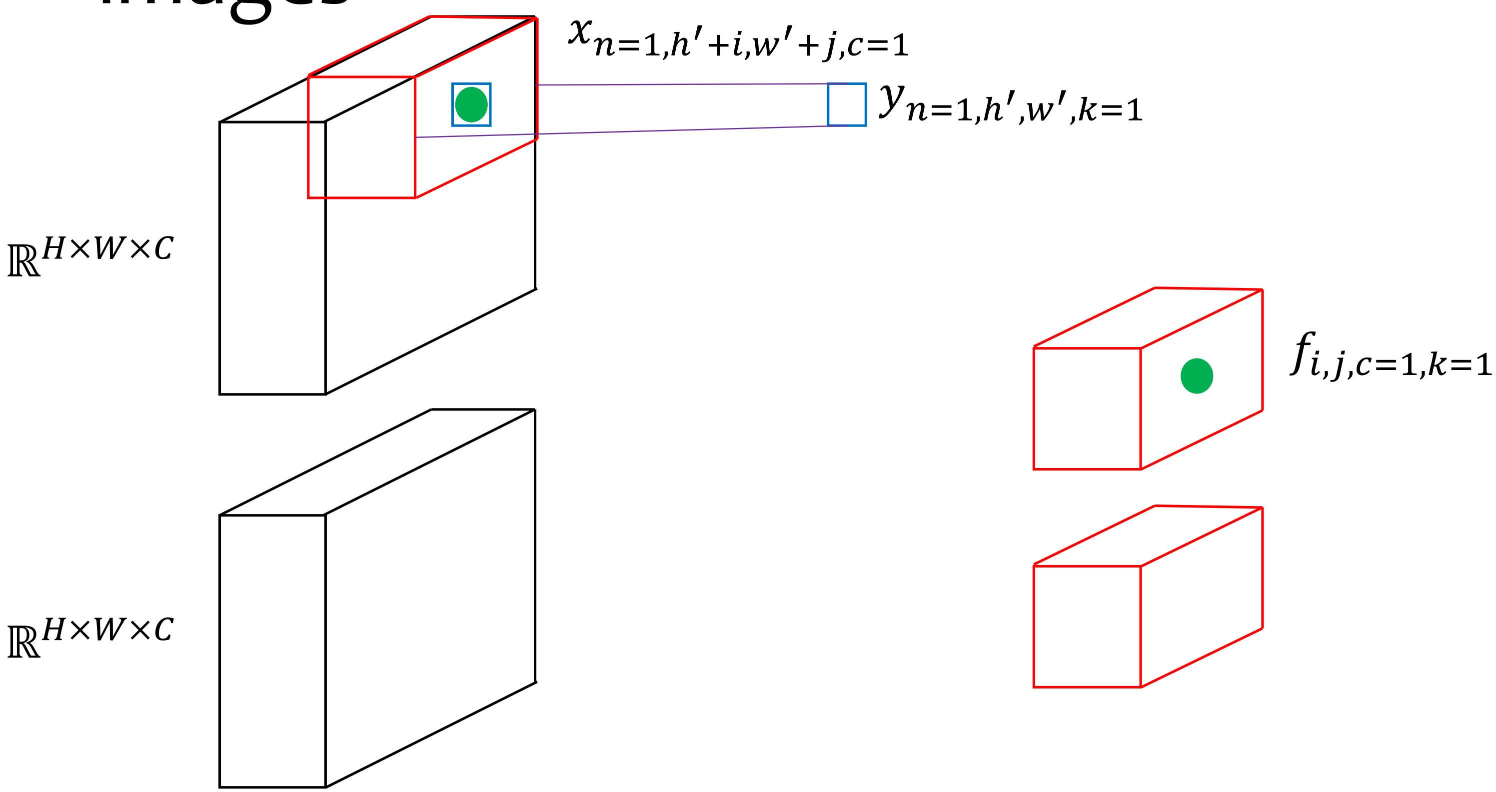


Bonus: formalize CONV. & BP

- Let $x \in \mathbb{R}^{N \times H \times W \times C}$ be N images, and $f \in \mathbb{R}^{I \times J \times C \times K}$ be K filters, with same #channel as images

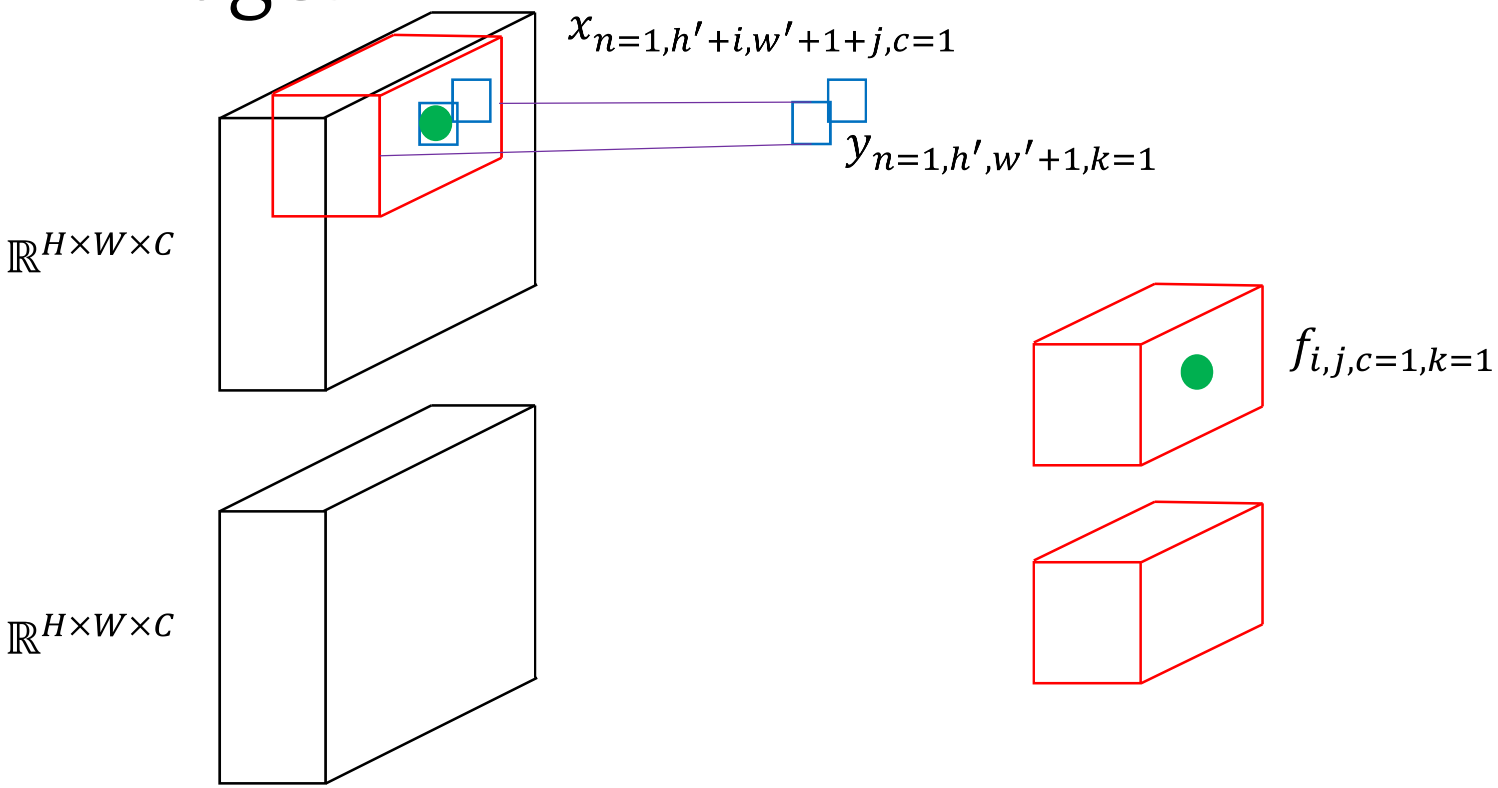
Bonus: formalize CONV. & BP

- Let $x \in \mathbb{R}^{N \times H \times W \times C}$ be N images, and $f \in \mathbb{R}^{I \times J \times C \times K}$ be K filters, with same #channel as images



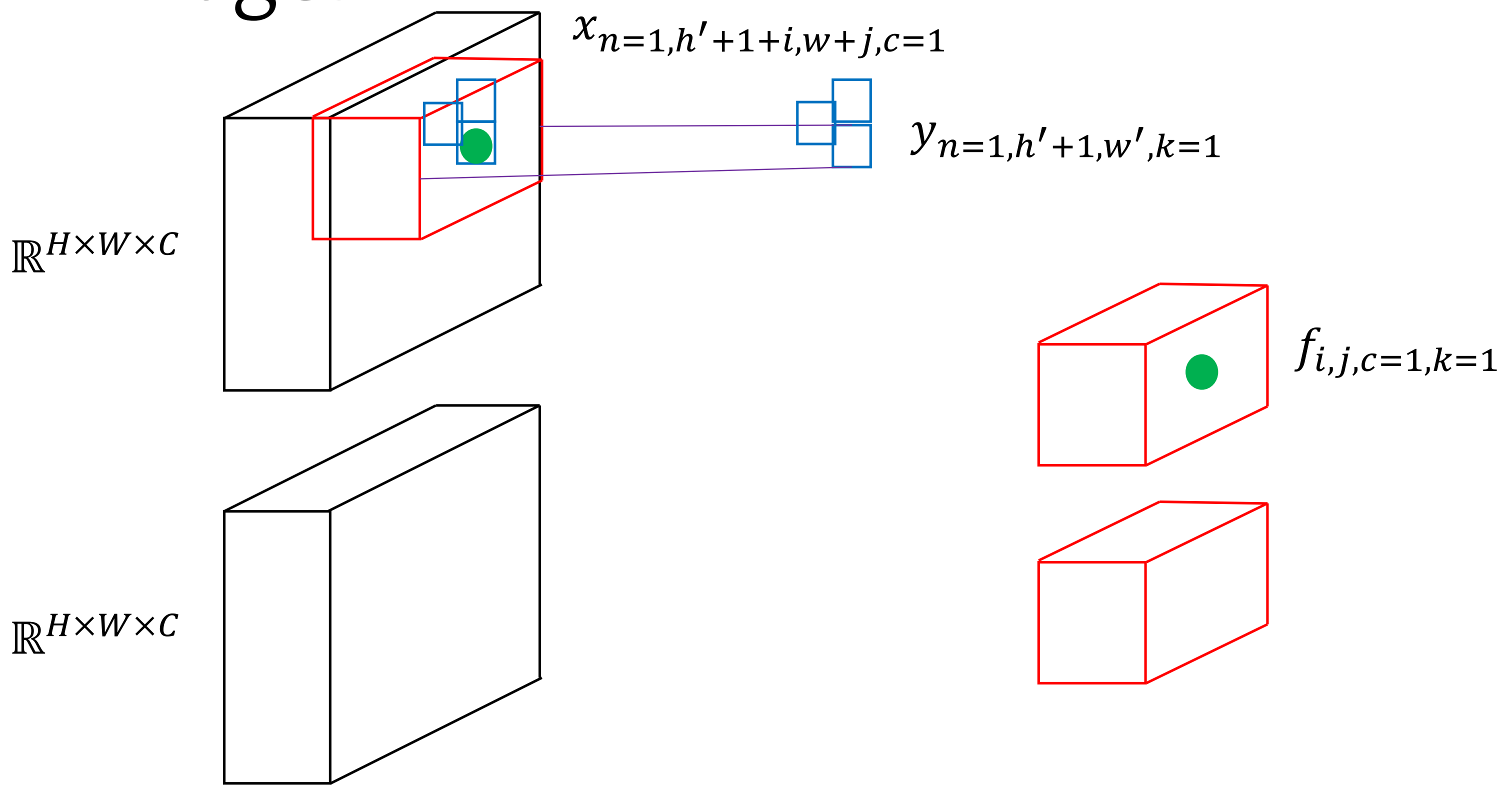
Bonus: formalize CONV. & BP

- Let $x \in \mathbb{R}^{N\times H\times W\times C}$ be N images, and $f \in \mathbb{R}^{I\times J\times C\times K}$ be K filters, with same #channel as images



Bonus: formalize CONV. & BP

- Let $x \in \mathbb{R}^{N\times H\times W\times C}$ be N images, and $f \in \mathbb{R}^{I\times J\times C\times K}$ be K filters, with same #channel as images



Bonus: formalize CONV. & BP

- $\frac{\partial E}{\partial f_{i,j,c,k}} = \Sigma_h \Sigma_w \Sigma_\mathrm{n} x_{n,h,w,c} \cdot \frac{\partial E}{\partial y_{n,h,w,k}}$