Convolutional Neural Network (CNN)

Special neural network architecture
for image analysis

To understand the construction of a CNN we need to introduce filters/kernels

$$I(i,j) = \text{Image}$$

$$K(m,n) = \text{Kernel}$$

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

Filtered image

Here is a small numerical example where a 4x4 image matrix is filtered by a 2x2 kernel.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} * \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} (-1+2+0+6) & (-2+3+0+7) & (-3+4+0+8) \\ (-5+6+0+10) & (-6+7+0+11) & (-7+8+0+12) \\ (-9+10+0+14) & (-10+11+0+15) & (-11+12+0+16) \end{bmatrix}$$

$$= \begin{bmatrix} 7 & 8 & 9 \\ 11 & 12 & 13 \\ 15 & 16 & 17 \end{bmatrix}$$

The box average kernel

$$\text{box average} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Emboss

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Outline

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Top Sobel

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Left Sobel

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Have fun with kernels!

http://setosa.io/ev/image-kernels/

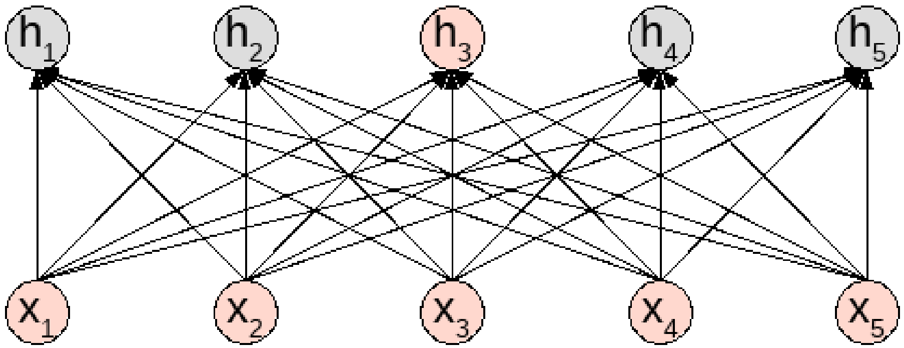
In CNNs we **train** filters of various sizes!

$$\begin{bmatrix} w & w_2 \\ w_3 & w_4 \end{bmatrix} \quad \begin{bmatrix} w & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \quad \begin{bmatrix} w & w_2 & w_3 & w_4 \\ w_5 & w_6 & w_7 & w_8 \\ w_9 & w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} & w_{16} \end{bmatrix} \quad ....$$

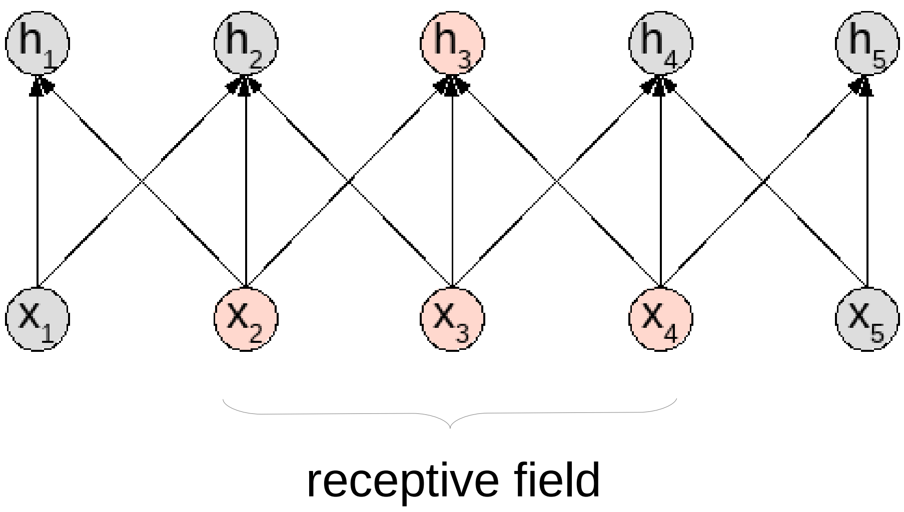
How can we “implement” the filter process
in a neural network architecture?

We need **sparse connectivity** and **weight sharing**

1D case

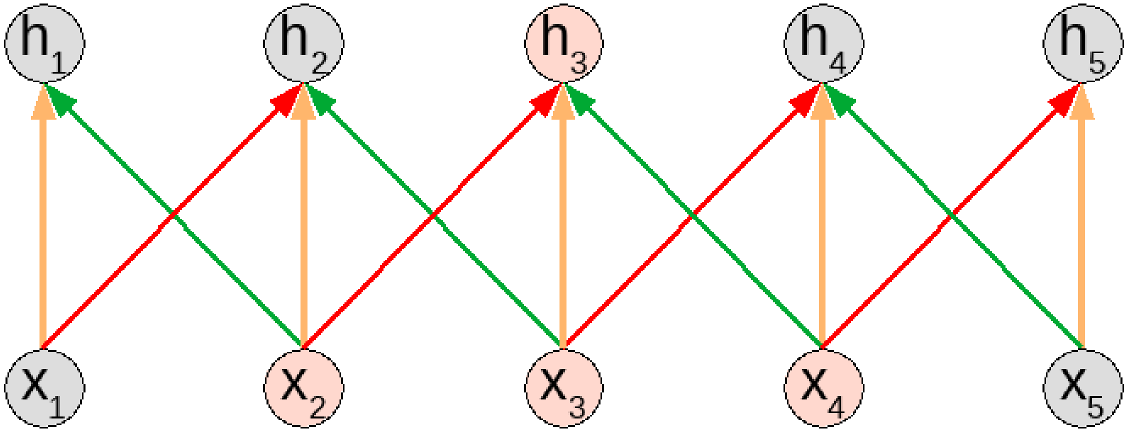


Fully connected input-to-hidden layer in an MLP. h_3 is receiving inputs from all input nodes.



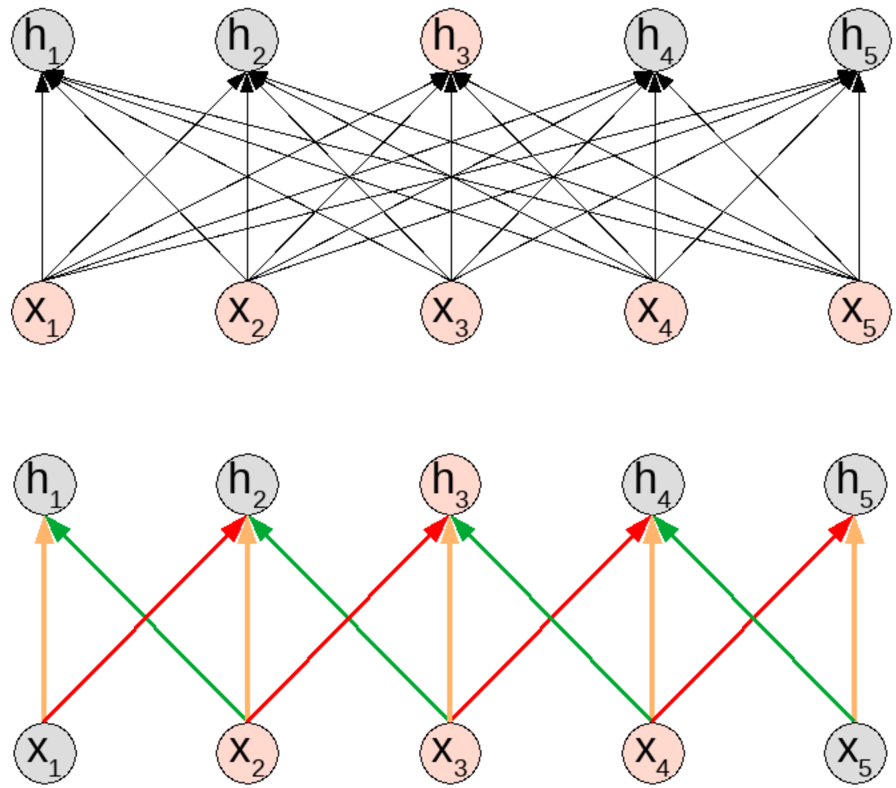
The same MLP but with fewer weights. h_3 is receiving inputs from only three input nodes.

But we also need weight sharing to
simulate the filter process.



Same color = shared weights

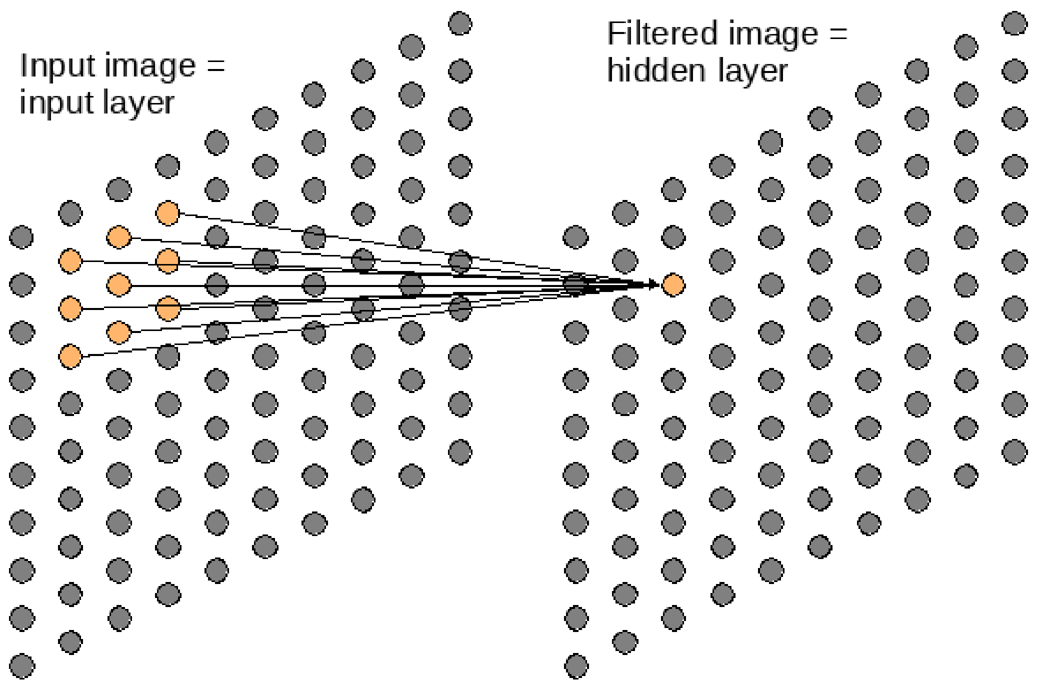
Dramatic reduction of trainable weights, compared to a fully connected network



Here, 25 (unique) weights (excluding bias)

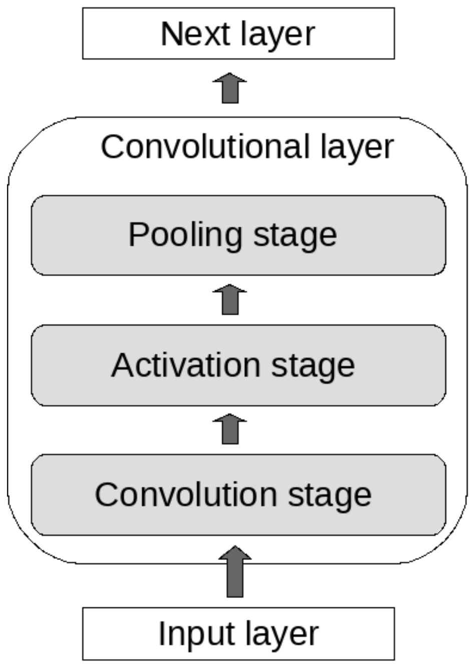
Here, 3 (unique) weights (excluding bias)

2D images have 2D kernels



Note: each hidden node share the weights with all other hidden nodes.

CNN building blocks

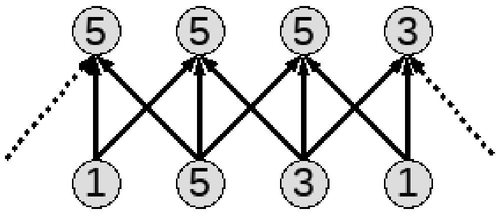


Pooling?

Some version of ReLU

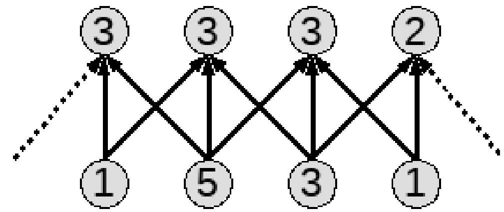
Pooling layer

Max pooling layer



Hidden layer

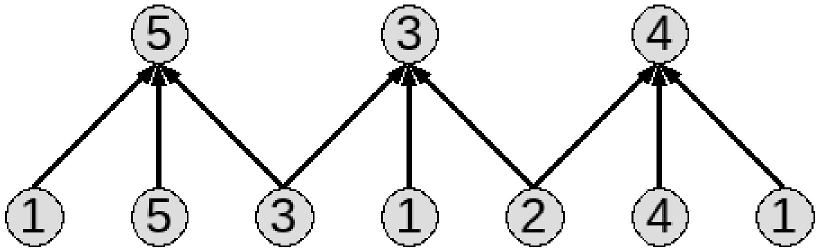
Average pooling layer



Hidden layer

No downsampling!

Max pooling, with downsampling

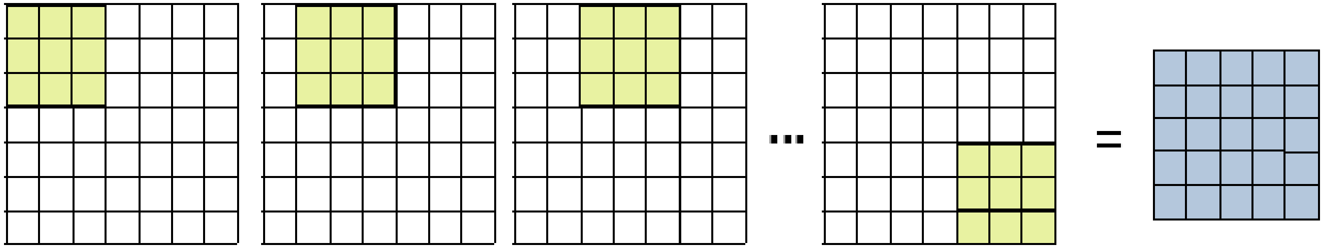


Hidden layer

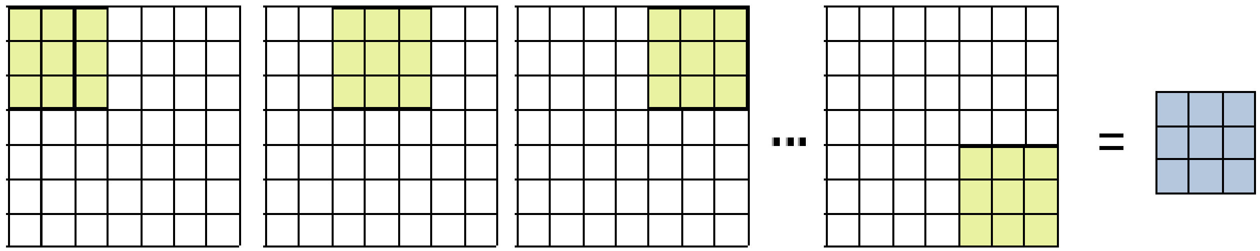
With downsampling!

Some details of the filter process

Example: 7x7 image, with a 3x3 kernel
moving 1 pixel each time (**stride=1**)
gives a 5x5 filtered image (= hidden nodes)

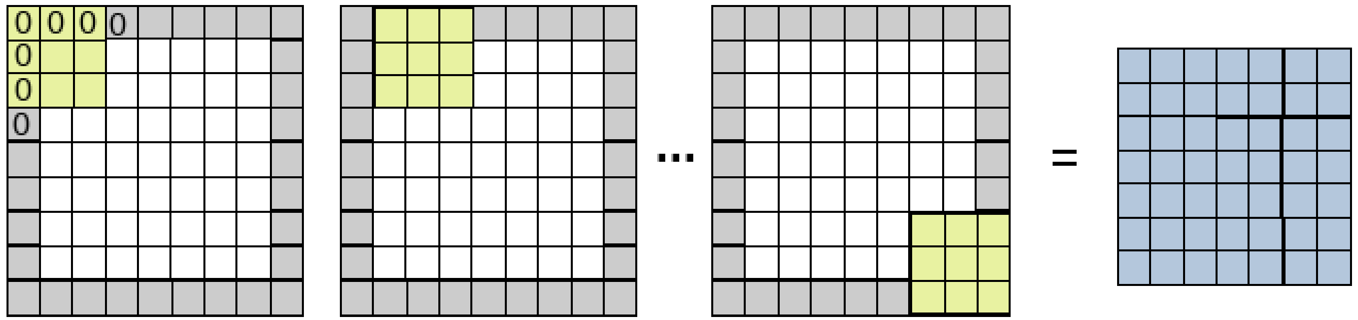


Example: 7x7 image, with a 3x3 kernel
moving 2 pixel each time (**stride=2**)
gives a 3x3 filtered image



Keep the original size by using zero padding

Example: 7x7 image, with a 3x3 kernel
stride=1, zero padding
gives a 7x7 filtered image

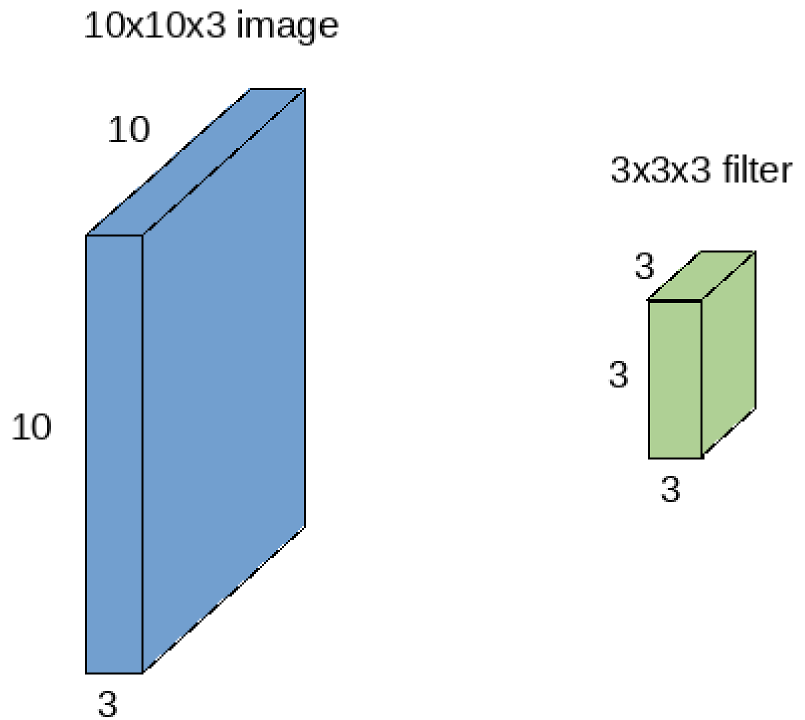


$$\text{output width} = \frac{W - F_w + 2P}{S_w} + 1$$

$$\text{output height} = \frac{H - F_h + 2P}{S_h} + 1$$

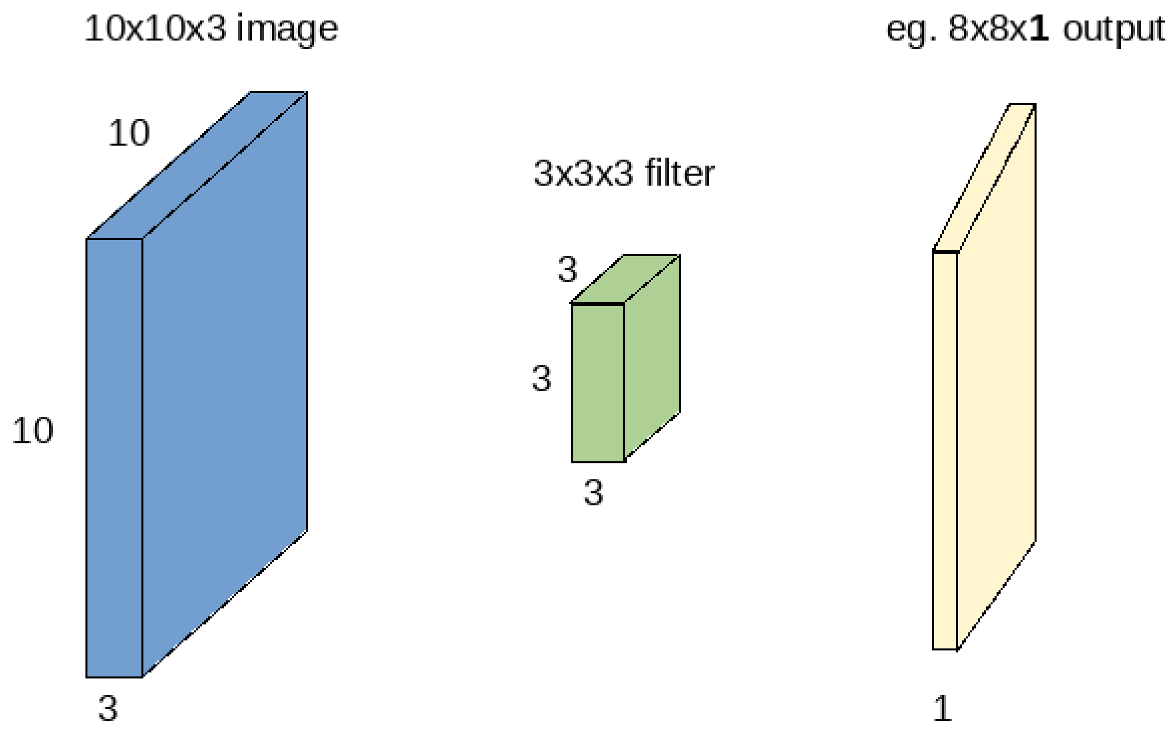
P = padding
S = stride
F = filter size

With multichannel images (color) we just add one (2D) filter for each channel



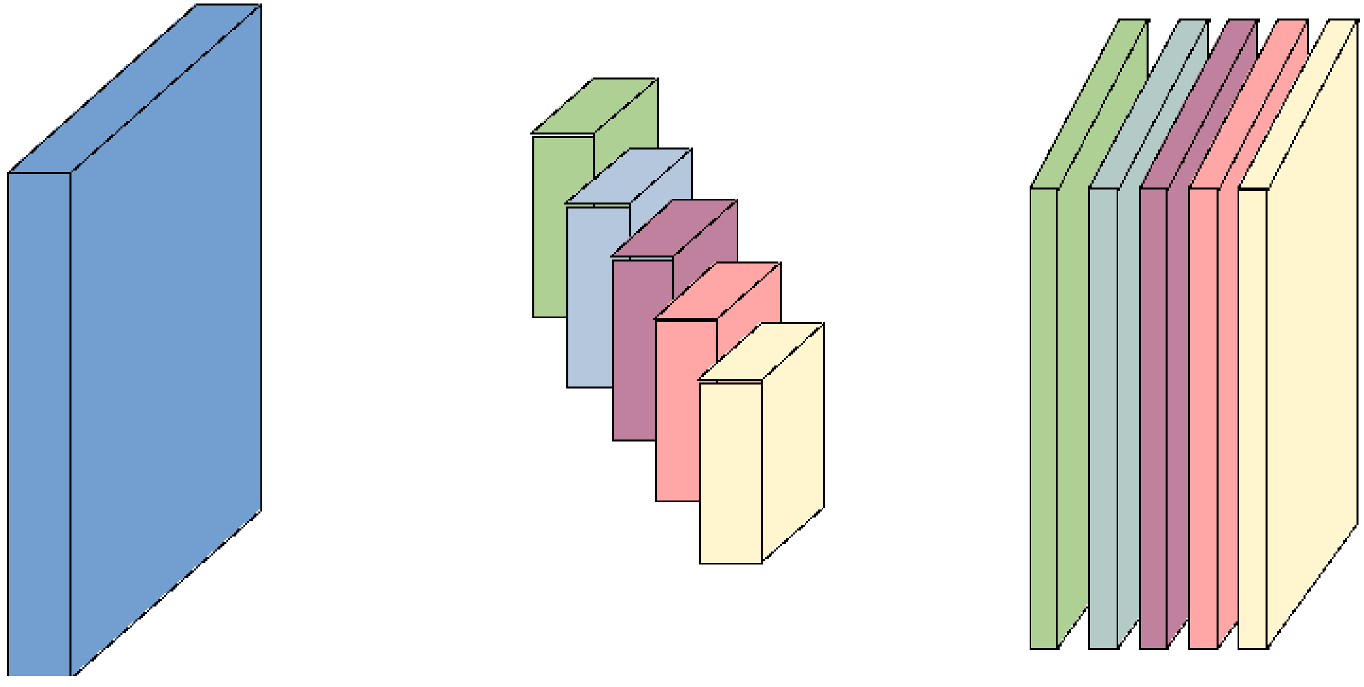
Remember: Filters always extend to the full dept of the input image

Regardless of input depth, the output has depth = 1

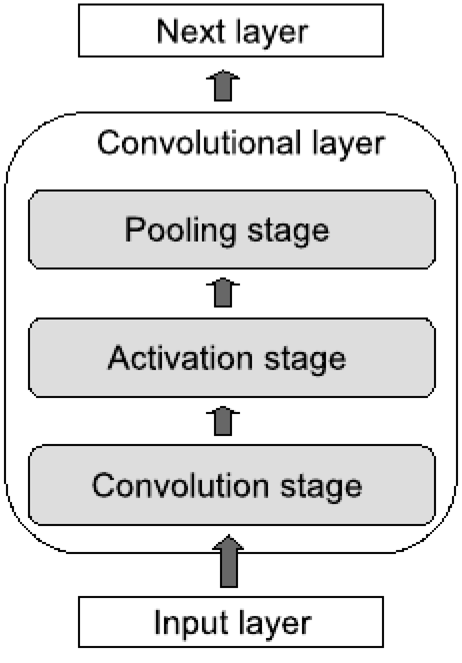


We have 27 weights in the filter + 1 bias weight!!

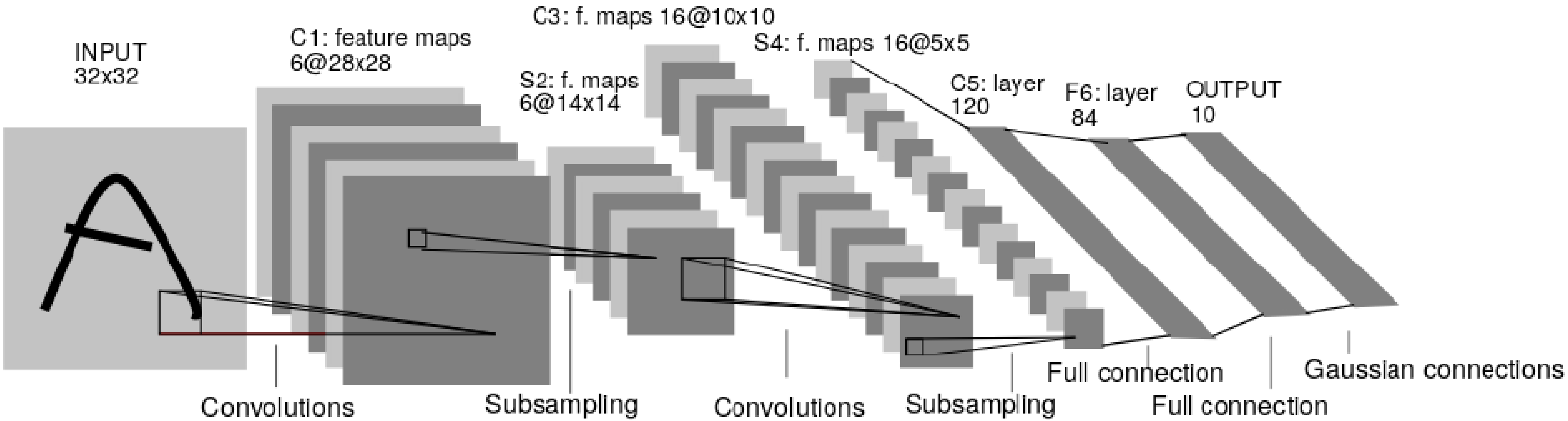
Example where 5 filters will result
in a 5-channel filtered image.



Again the building blocks

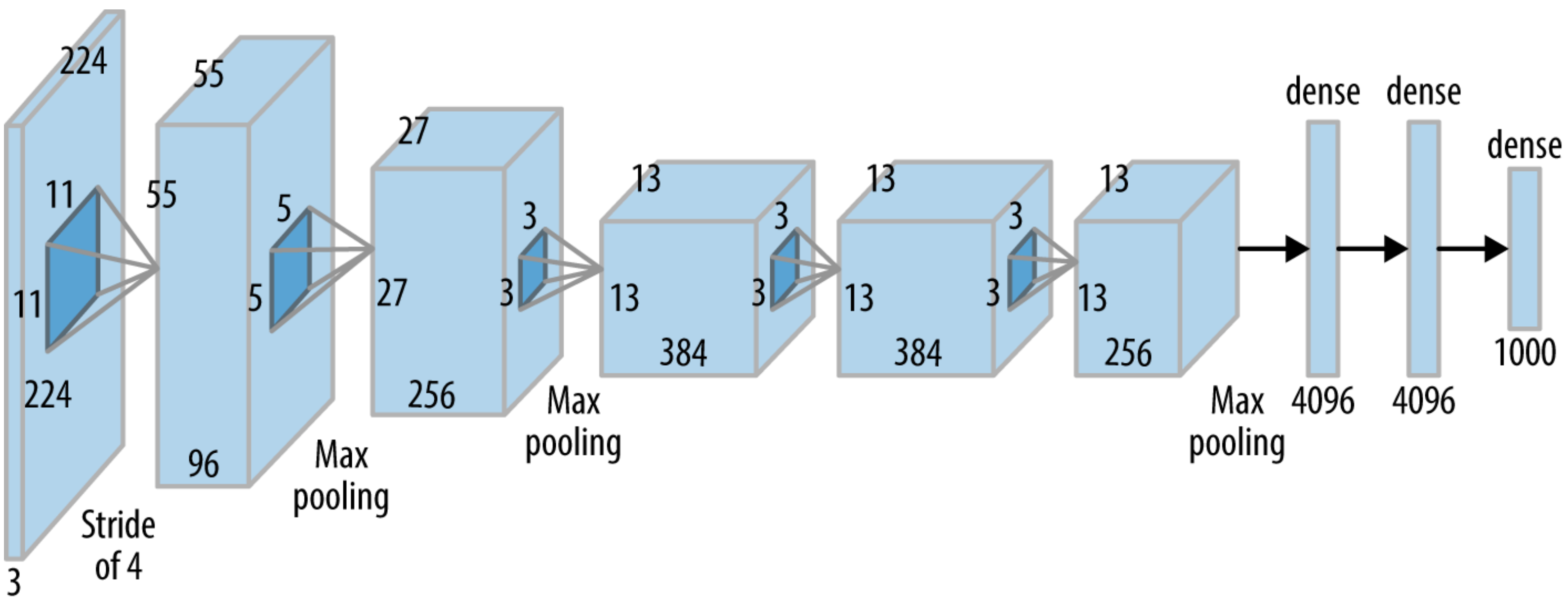


One of the very first CNNs (Lecun, 1998). This one is called LeNet-5.



Some examples of “famous” CNNs

Alexnet, Winner of ILSVRC competition 2012.



ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

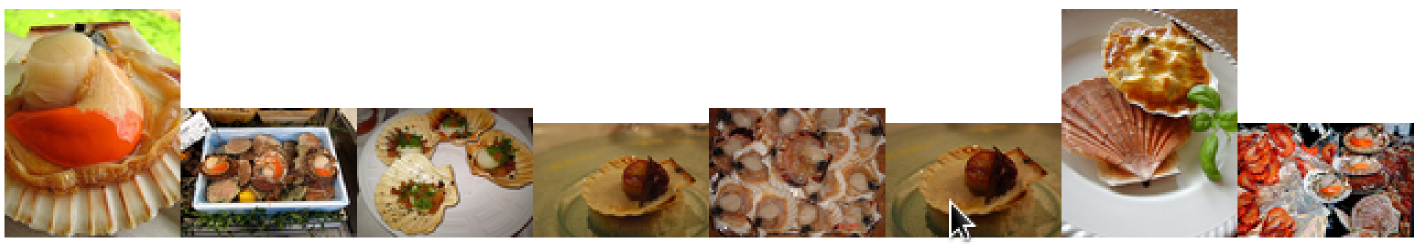
IMAGENET

14,197,122 images, 21841 synsets indexed

Explore Download Challenges Publications CoolStuff About

Not logged in. Login | Signup

ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.
Click here to learn more about ImageNet, Click here to join the ImageNet mailing list.



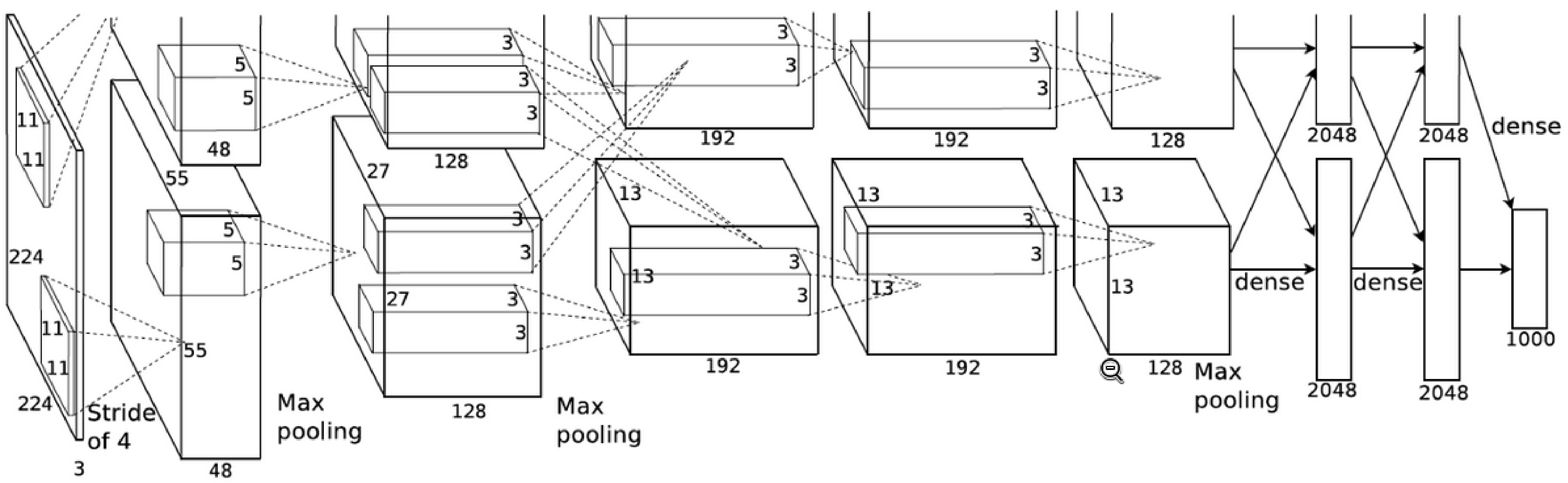
What do these images have in common? *Find out!*

Check out the ImageNet Challenge on Kaggle!

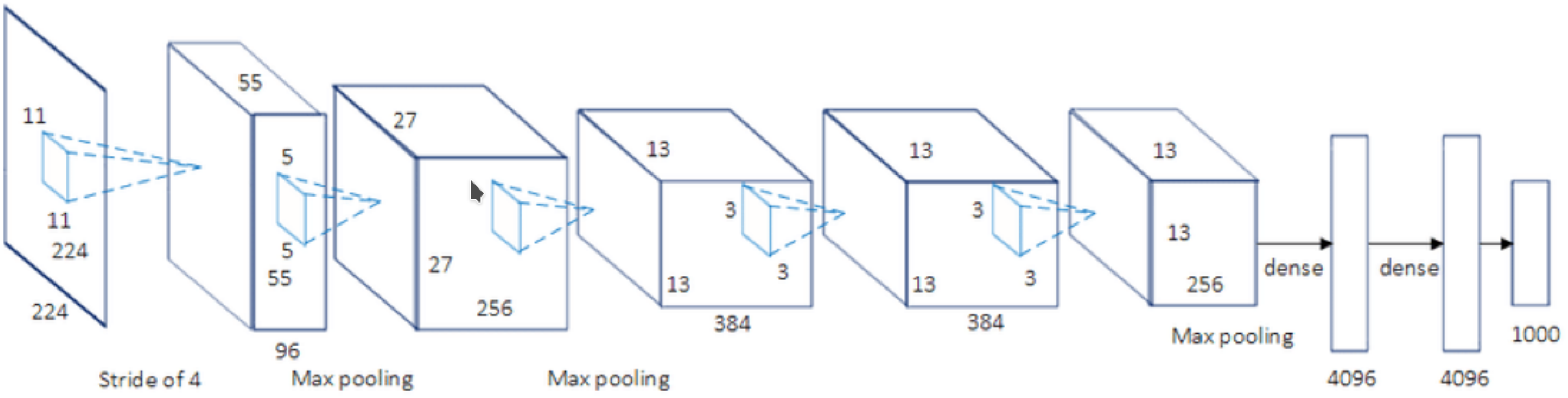
ImageNet classification (ILSVRC)
Large Scale Visual Recognition Challenge (2012)

- 1000 objects
- 1.2 million training images
- 100 000 test images

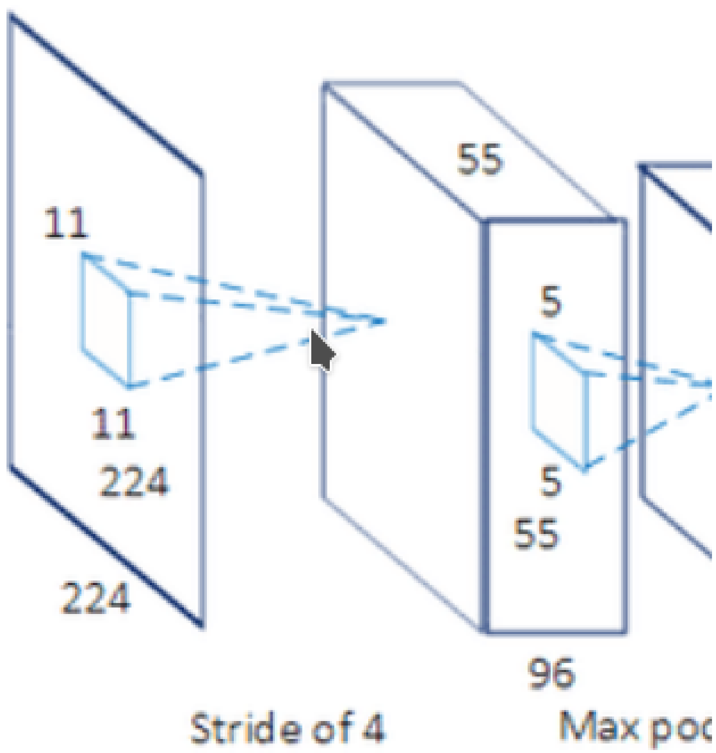
Winner: AlexNet



AlexNet single GPU equivalent

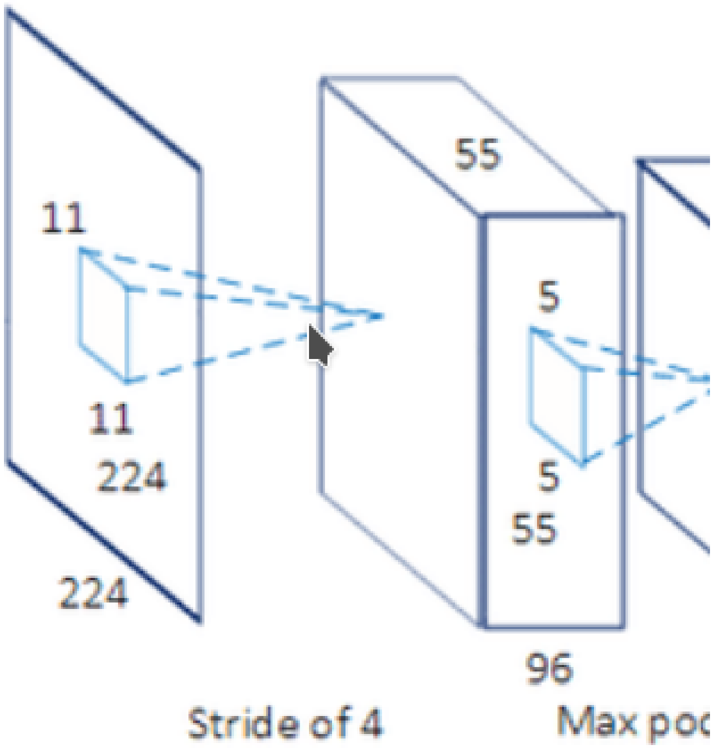


First convolutional layer



- Images: 227x227x3
- Filter size: 11x11
- Stride: 4
- Conv layer output: 55x55x**96**

First convolutional layer

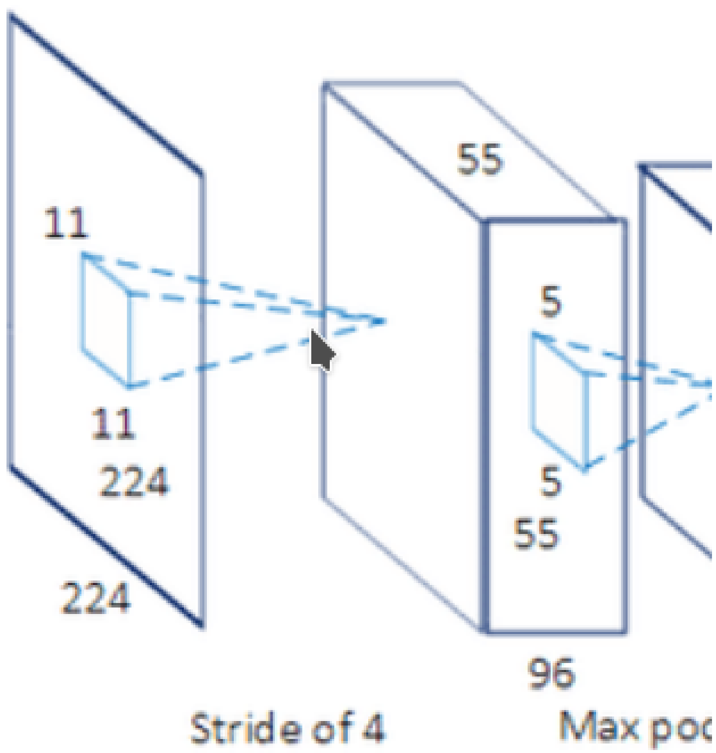


- Images: 227x227x3
- Filter size: 11x11
- Stride: 4
- Conv layer output: 55x55x**96**

Why this output size?

(227-11)/4 + 1 = 55

First convolutional layer



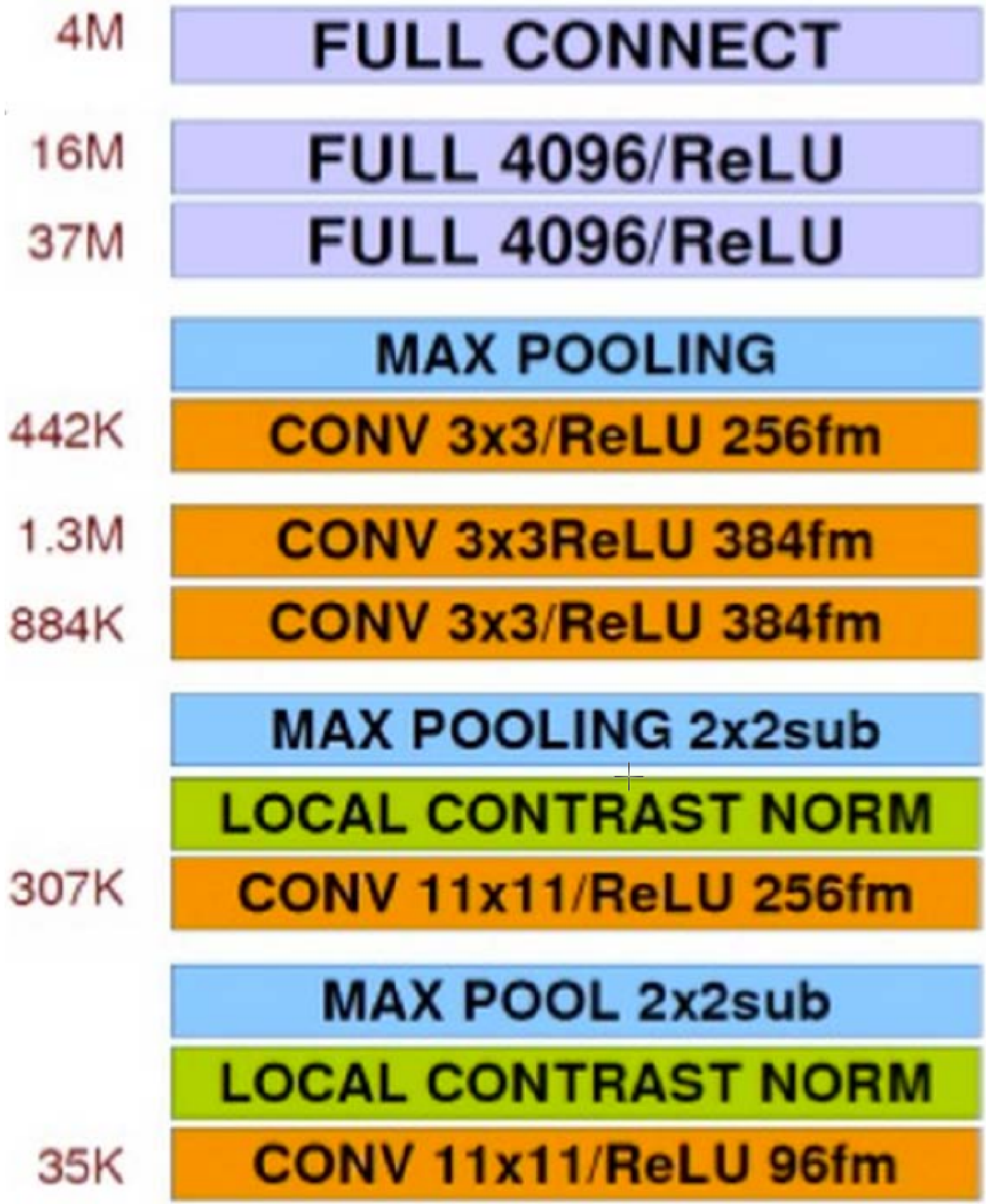
- Images: 227x227x3
- Filter size: 11x11
- Stride: 4
- Conv layer output: 55x55x**96**

How many weights?

(11*11*3+1) weights per filter
96 filters

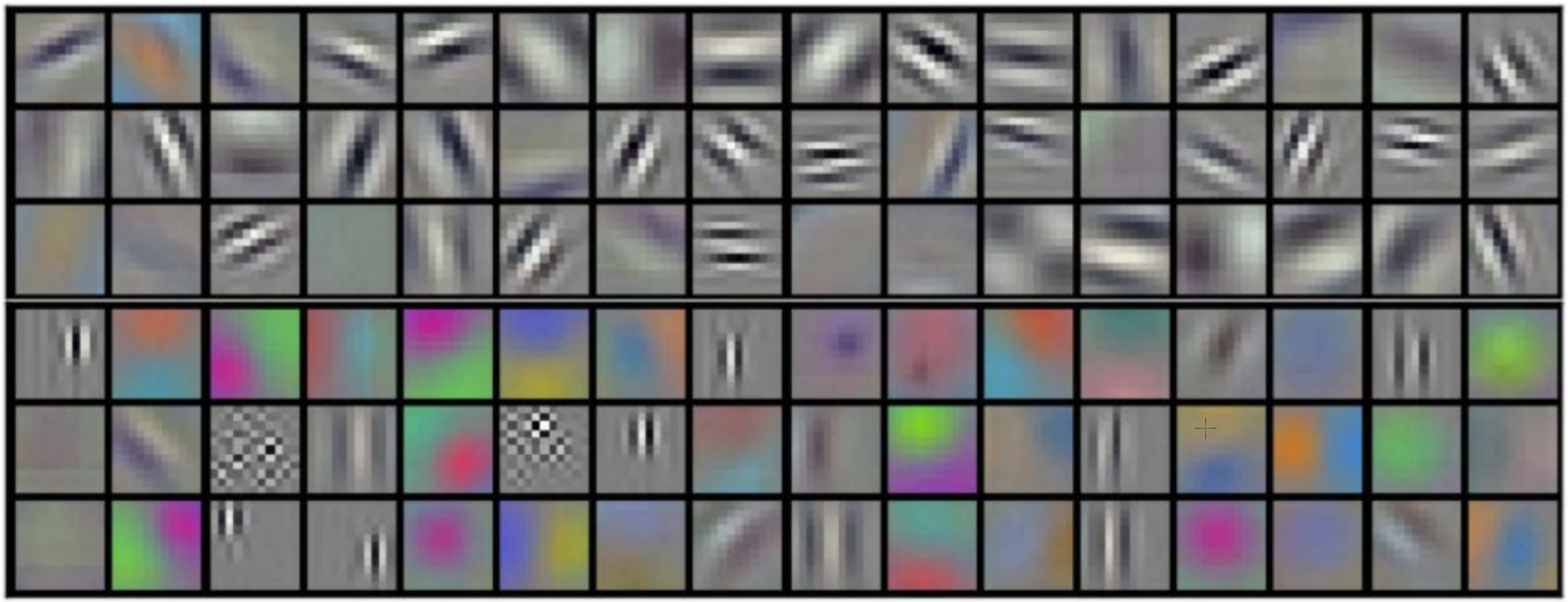
(11*11*3+1)*96 = 34944 weights

Size / Operation	Filter	Depth	Stride	Padding	Number of Parameters
3* 227 * 227					
Conv1 + Relu	11 * 11	96	4		(11*11*3 + 1) * 96=34944
96 * 55 * 55					
Max Pooling	3 * 3		2		
96 * 27 * 27					
Norm					
Conv2 + Relu	5 * 5	256	1	2	(5 * 5 * 96 + 1) * 256=614656
256 * 27 * 27					
Max Pooling	3 * 3		2		
256 * 13 * 13					
Norm					
Conv3 + Relu	3 * 3	384	1	1	(3 * 3 * 256 + 1) * 384=885120
384 * 13 * 13					
Conv4 + Relu	3 * 3	384	1	1	(3 * 3 * 384 + 1) * 384=1327488
384 * 13 * 13					
Conv5 + Relu	3 * 3	256	1	1	(3 * 3 * 384 + 1) * 256=884992
256 * 13 * 13					
Max Pooling	3 * 3		2		
256 * 6 * 6					
Dropout (rate 0.5)					
FC6 + Relu					256 * 6 * 6 * 4096=37748736
4096					
Dropout (rate 0.5)					
FC7 + Relu					4096 * 4096=16777216
4096					
FC8 + Relu					4096 * 1000=4096000
1000 classes					
Overall					62369152=62.3 million
Conv VS FC					Conv:3.7million


4M
FULL CONNECT
16M
FULL 4096/ReLU
37M
FULL 4096/ReLU
MAX POOLING
442K
CONV 3x3/ReLU 256fm
1.3M
CONV 3x3ReLU 384fm
884K
CONV 3x3/ReLU 384fm
MAX POOLING 2x2sub
LOCAL CONTRAST NORM
307K
CONV 11x11/ReLU 256fm
MAX POOL 2x2sub
LOCAL CONTRAST NORM
35K
CONV 11x11/ReLU 96fm

The 96 filters from the first conv. Layer.

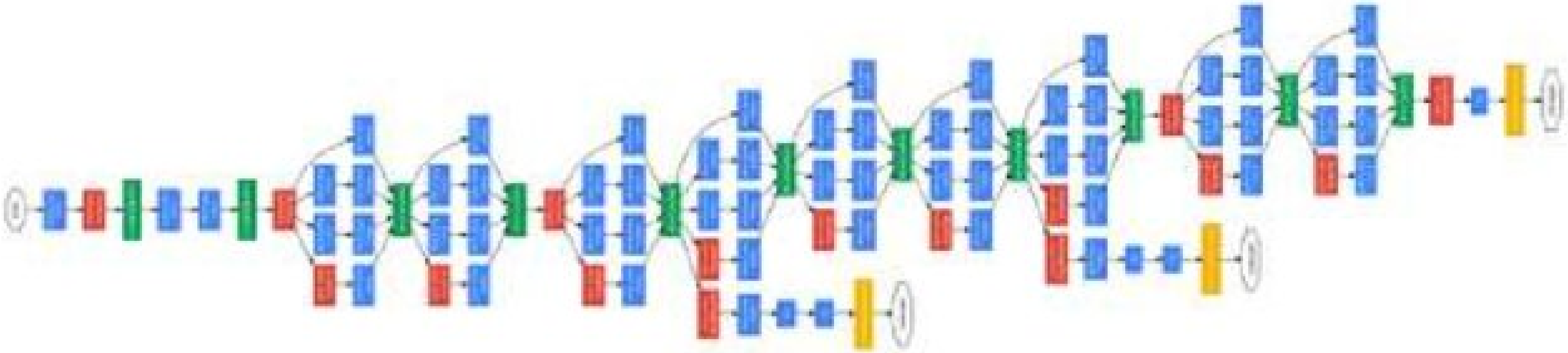
(11x11x3)



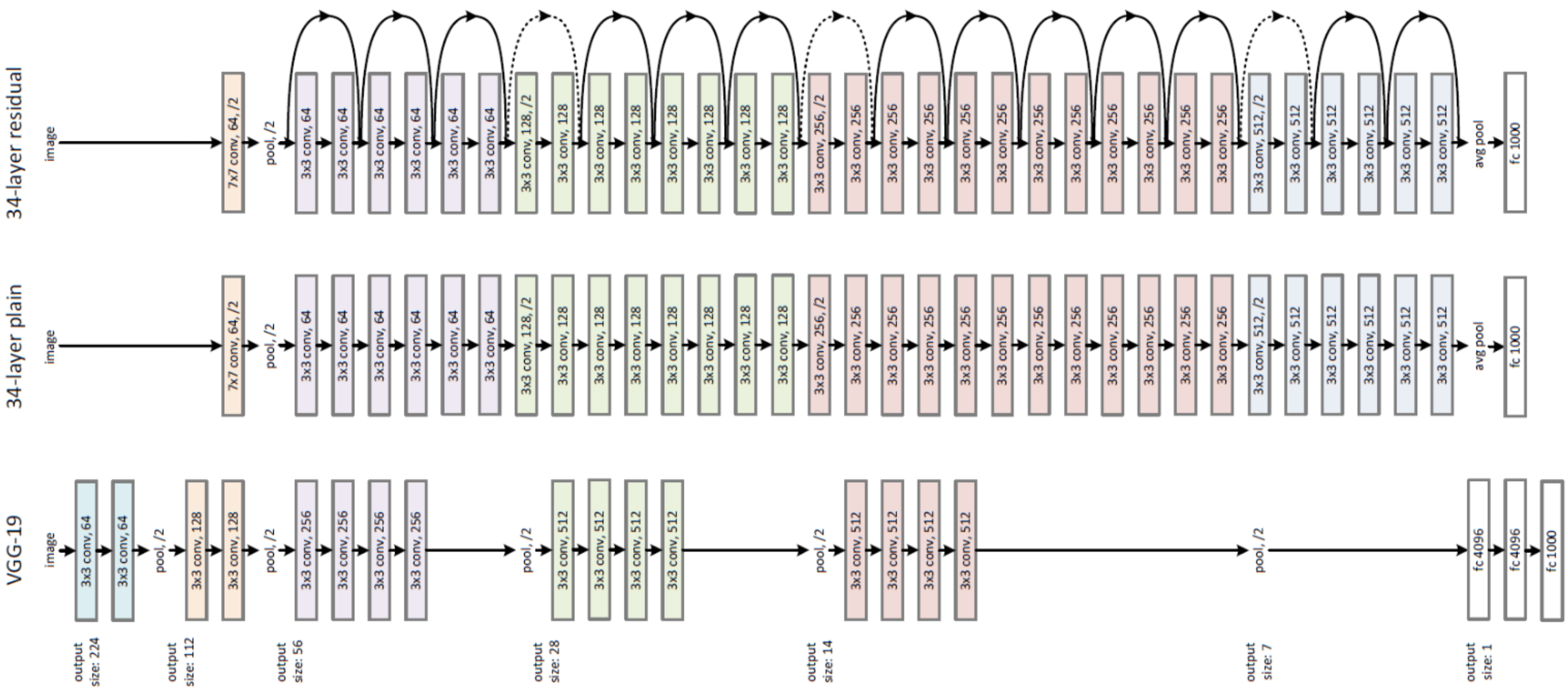
Remember the idea of deep learning:

Basic features → features → more advanced features → advanced features → → classification

GoogleNet, Winner of ILSVRC competition 2014.



Microsoft ResNet, Winner of ILSVRC competition 2015.



Machine learning playgrounds

Machine Learning Playground

http://ml-playground.com/

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.

https://playground.tensorflow.org/

ConvNetJS

Deep Learning in your browser

https://cs.stanford.edu/people/karpathy/convnetjs/

Experiments with Google

COLLECTION

AI Experiments

https://experiments.withgoogle.com/collection/ai