UiO : Department of Technology Systems
University of Oslo

Convolutional Neural Networks

Idar Dyrdal, Sigmund Rolfsjord

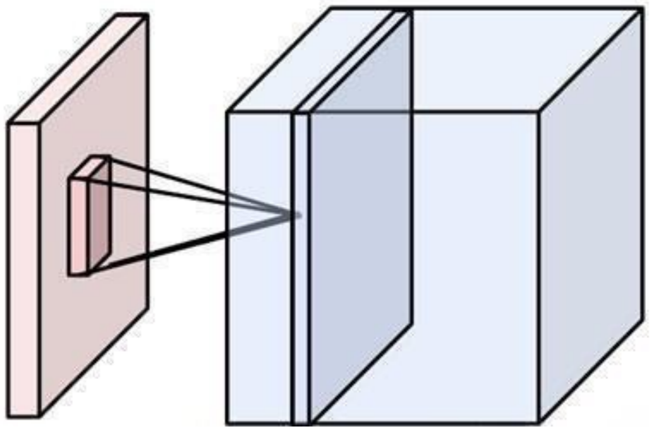
Convolutional Neural Network (CNN)

Used in Machine Vision and Image Analysis:

- Speech Recognition
- Image Recognition (Object detection)
- Video Recognition
- Image Segmentation
- …

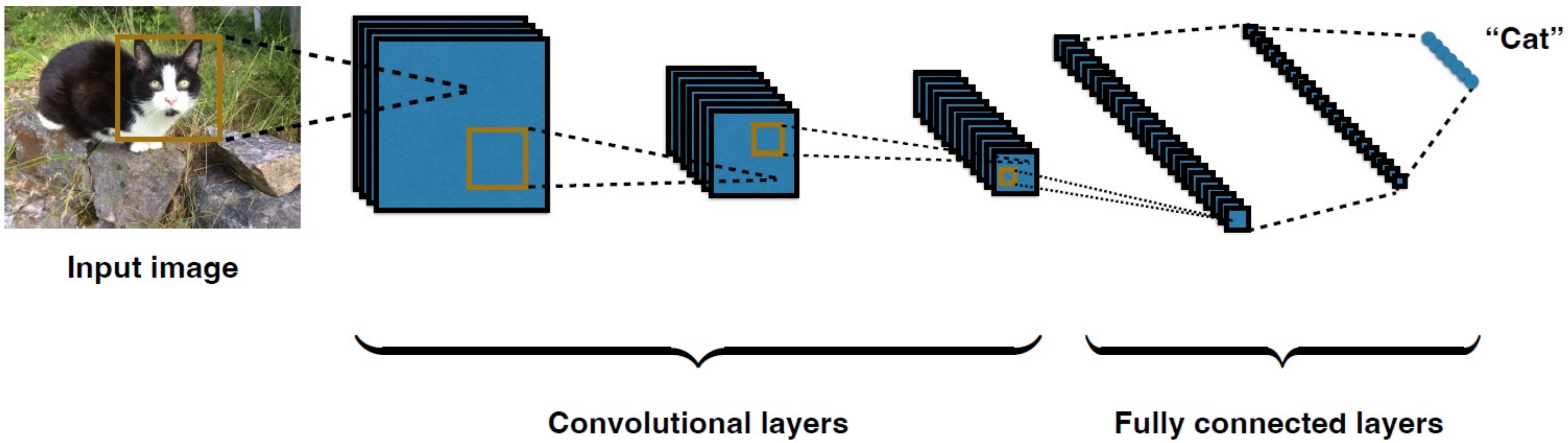
Convolutional neural network:

- Multi-layer feed-forward ANN
- Combinations of *convolutional* and fully connected layers
- Convolutional layers with *local* connectivity
- *Shared* weights across spatial positions
- Local or global pooling layers



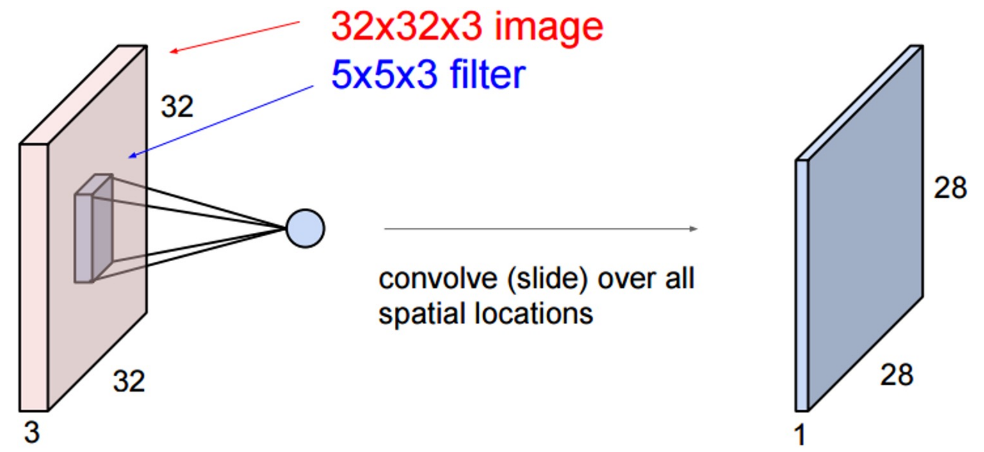
(A. Karpathy)

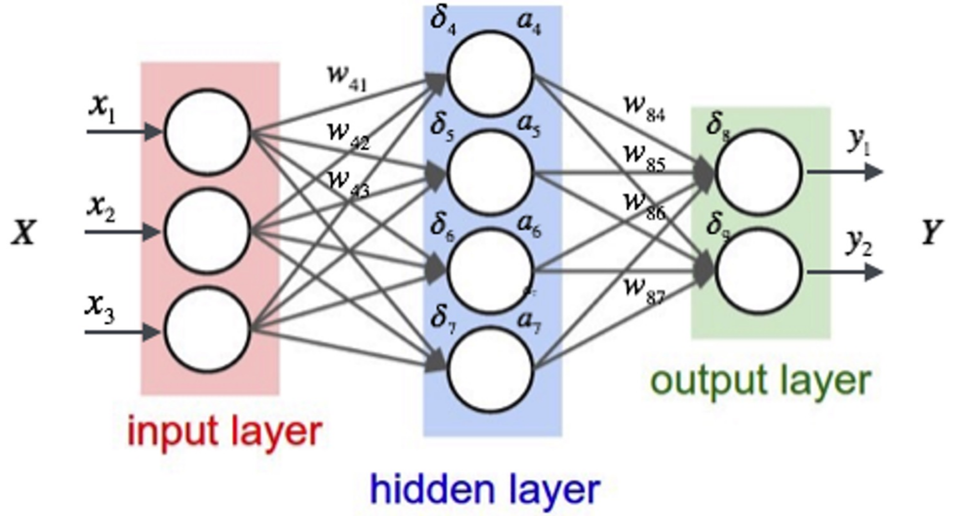
Convolutional Neural Network



Characteristics of a CNN

- Each layer is organized into *activation maps*
- Trainable multi-layer *convolutions*
- Local connections between layers (weighted sums over small local windows)
- Weight sharing (identical weights for all pixels)
- Shift-invariance
- Re-use of each training image multiple times (helps to prevent overfitting)





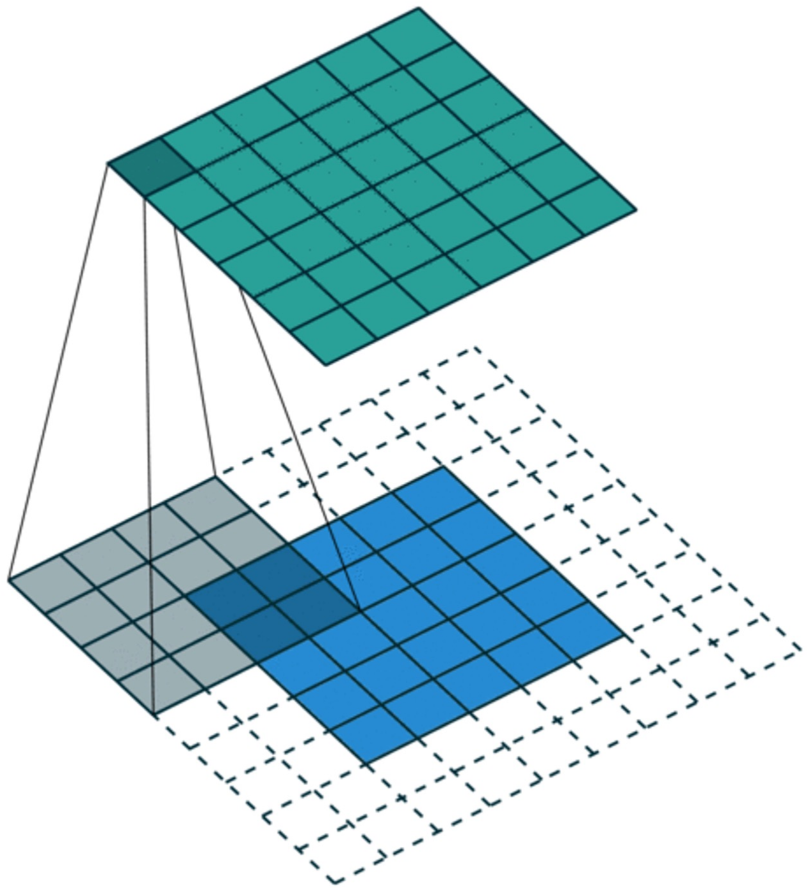
Object recognition – what is important?

- Relative position of pixels are more important than absolute position
- Translation is irrelevant
- It is still a cat!

Convolutions

- Replace matrix multiplications with convolutions

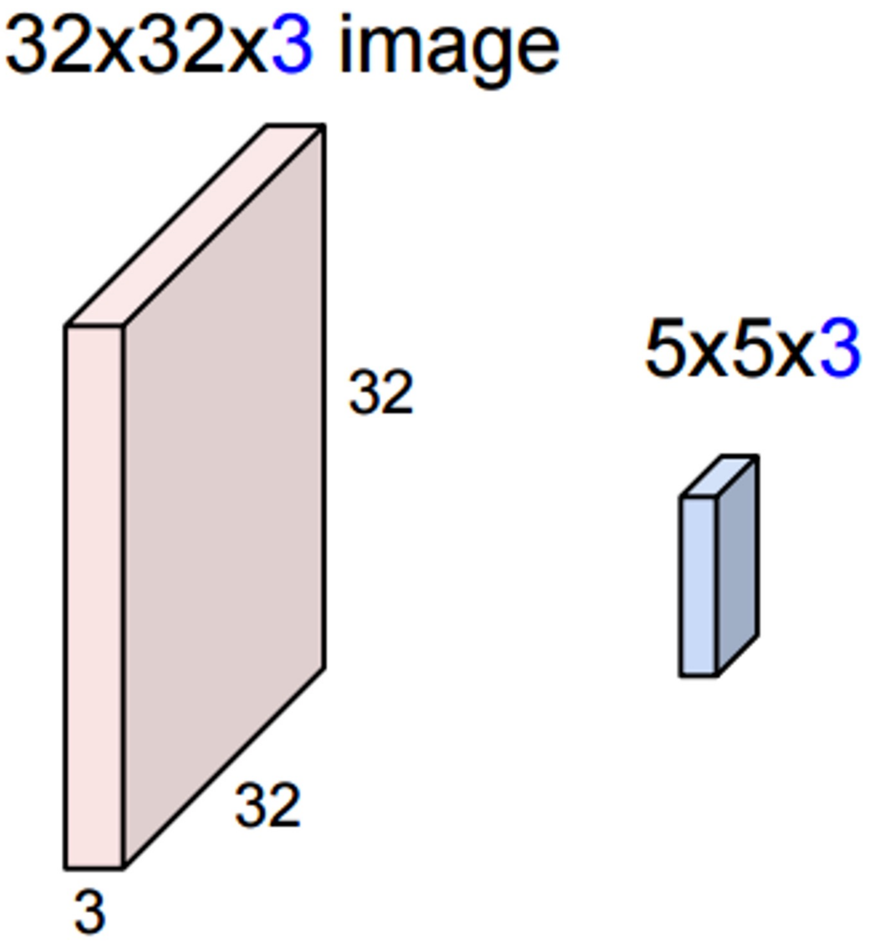
- Place the filter (kernel) over the image and compute the sum of all filter weights multiplied by the value of the corresponding pixel



Convolutions

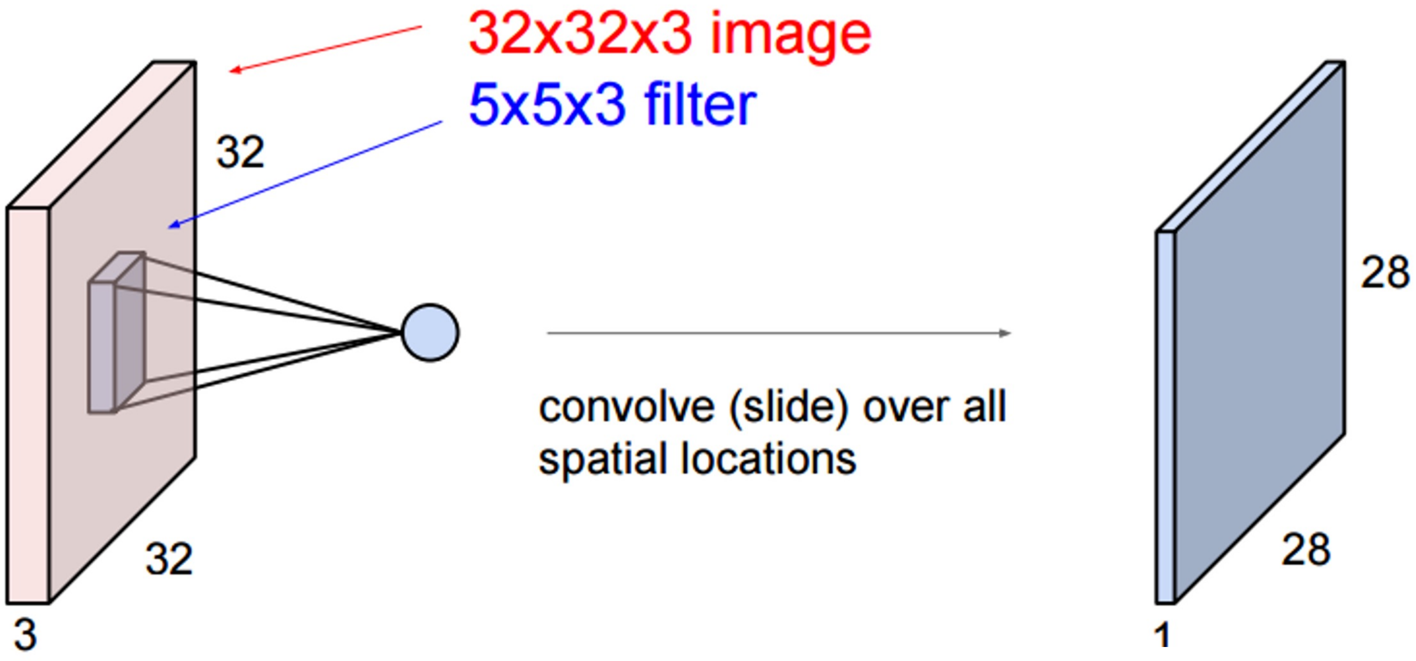
- Image and filter will often be 3-dimensional, i.e. height, width and number of channels (e.g. three channels for RGB colour image)

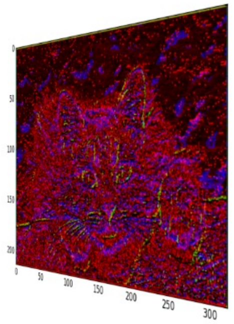
- The weighted sum is computed over all dimensions

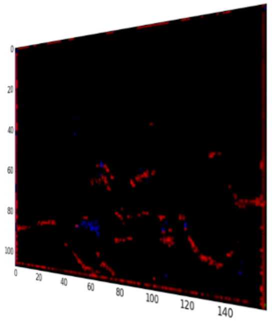


Convolutions

- Weighted sum computed over all input channels (dot product)
- Computed for all locations of the filter
- Output is a new single channel image (activation map)
- Remark: Padding (e.g. zero-padding or replication of pixels) can be applied



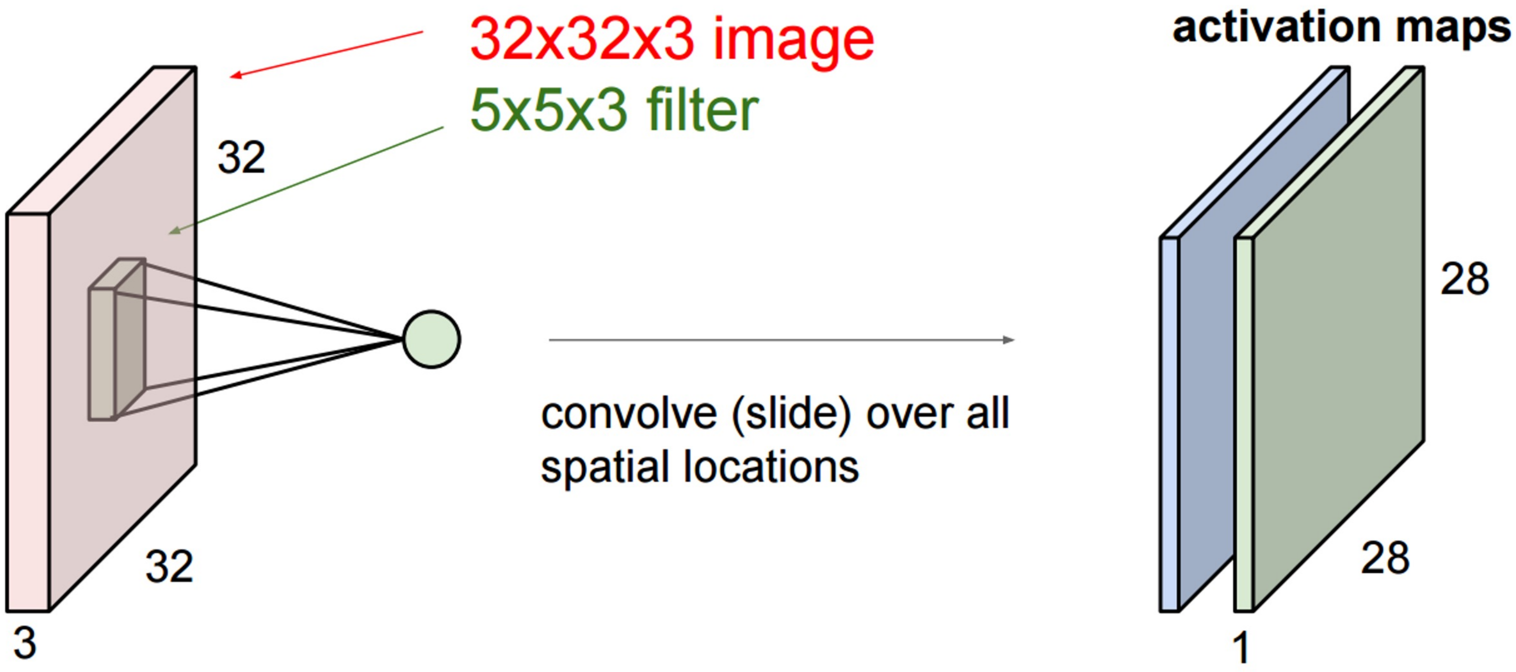




Convolutions – multiple filters per layer

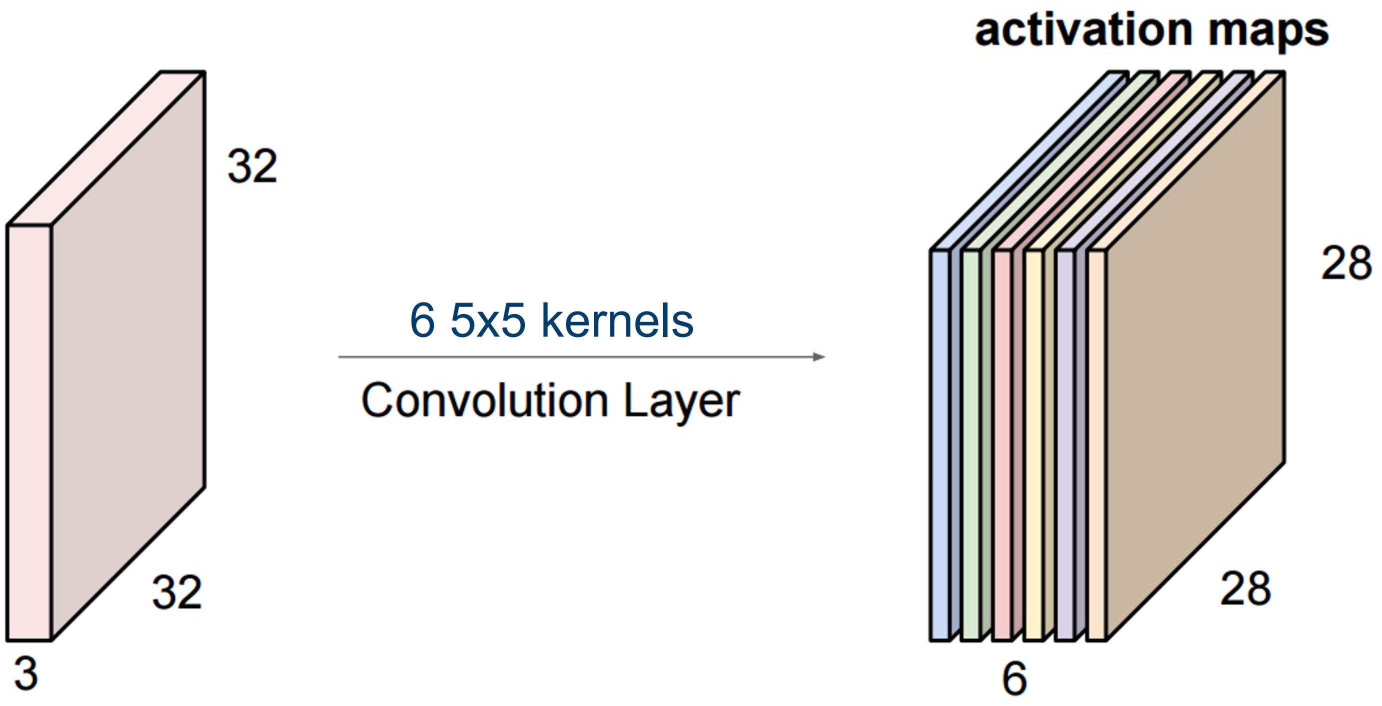
- Convolving the input image with a new kernel

- Result is a new output image

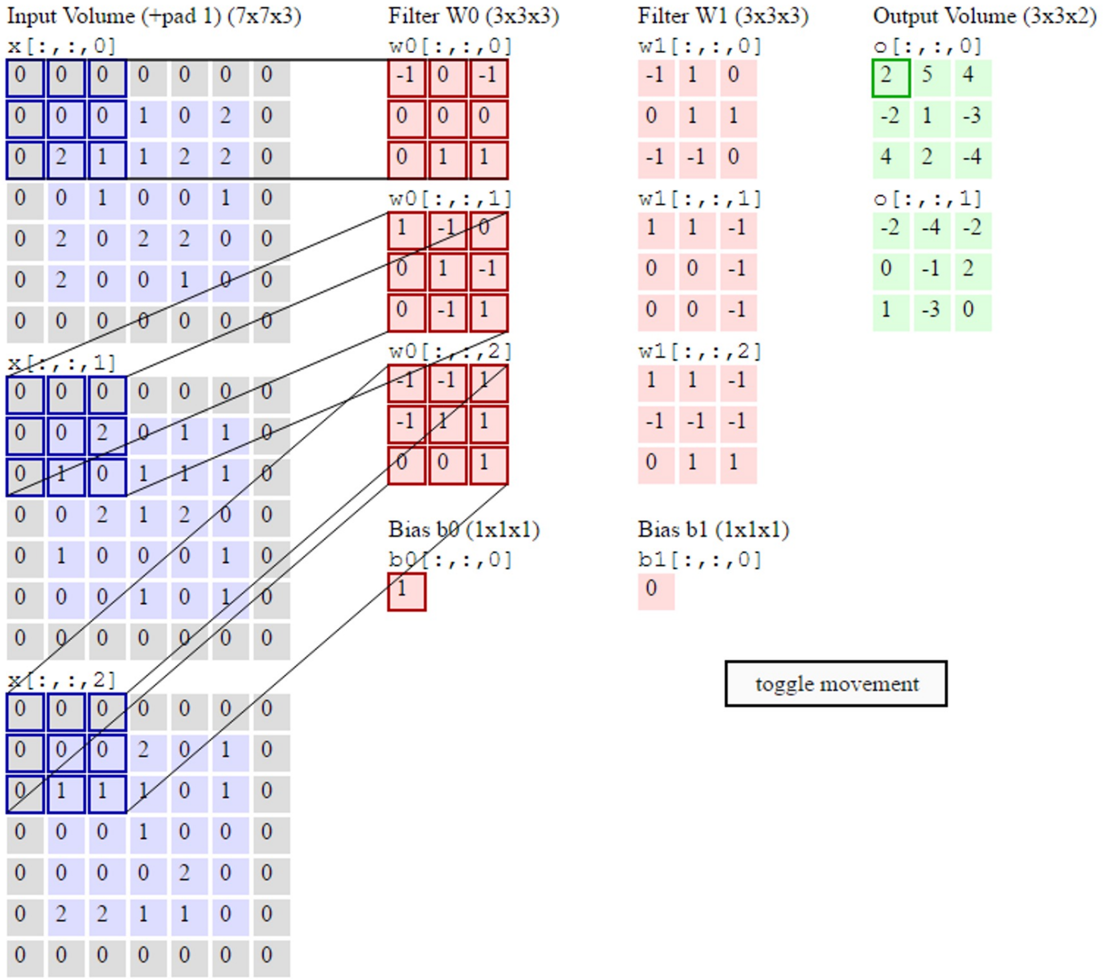


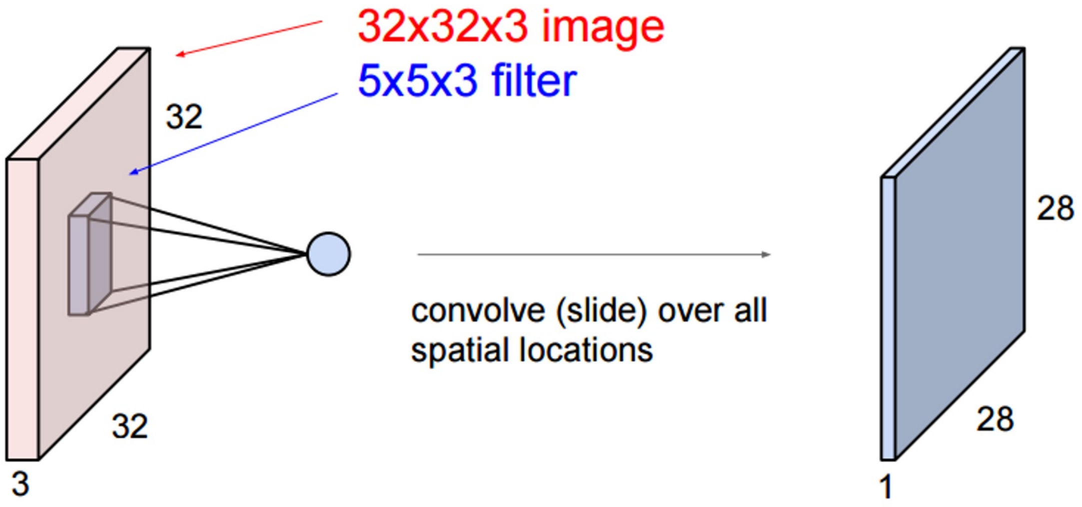
Convolutions – multiple filters per layer

- Convolving an input RBG image (3 channels) with 6 filters
- The result is a 6-channel output image

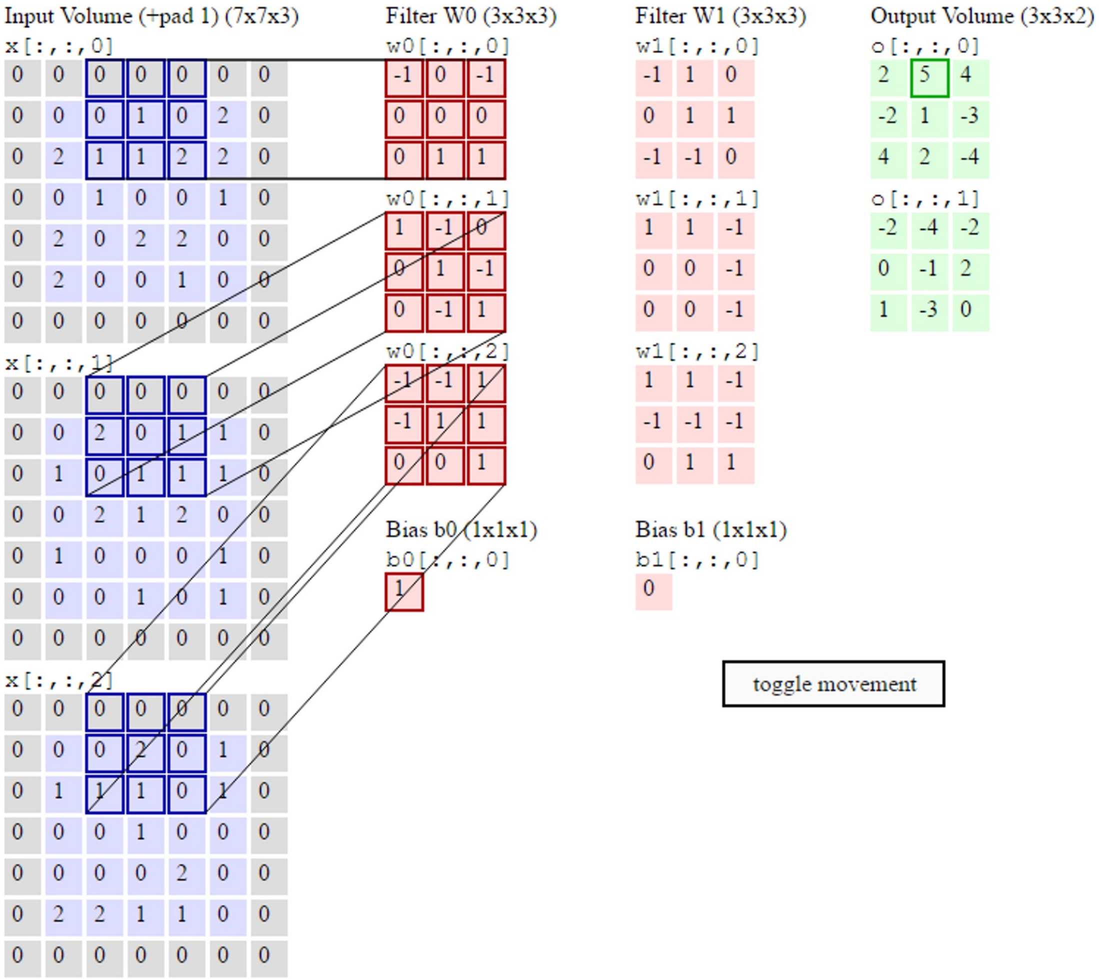


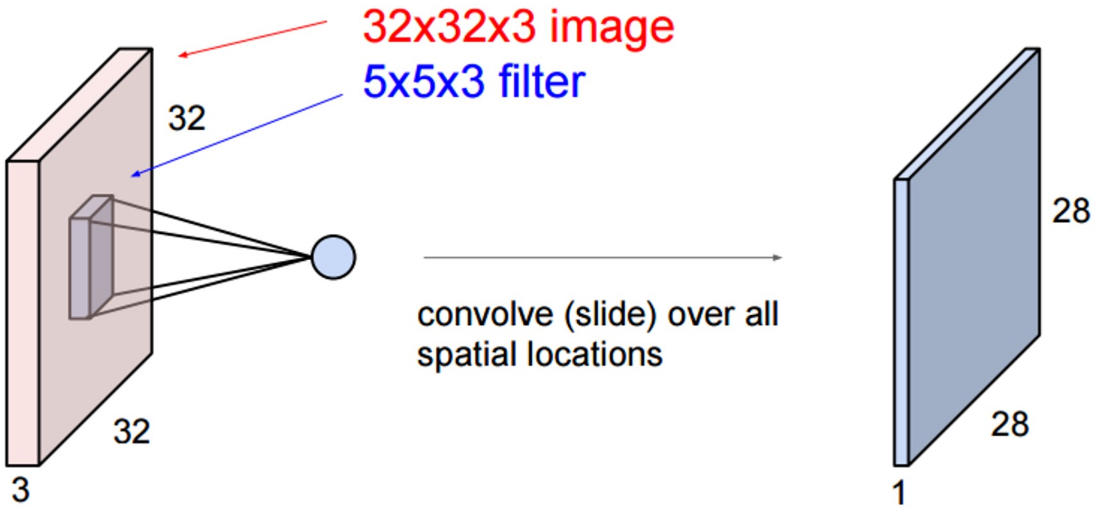
One layer network with two filters



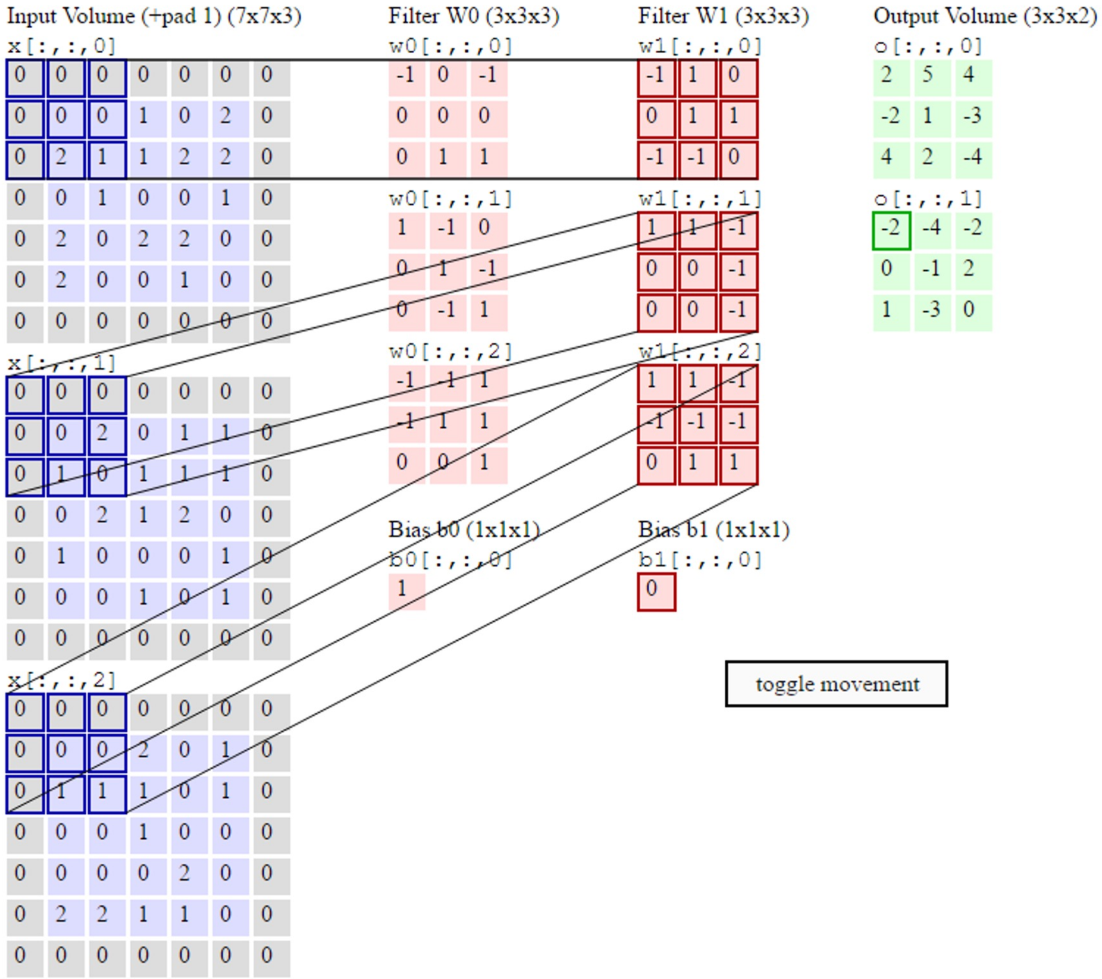


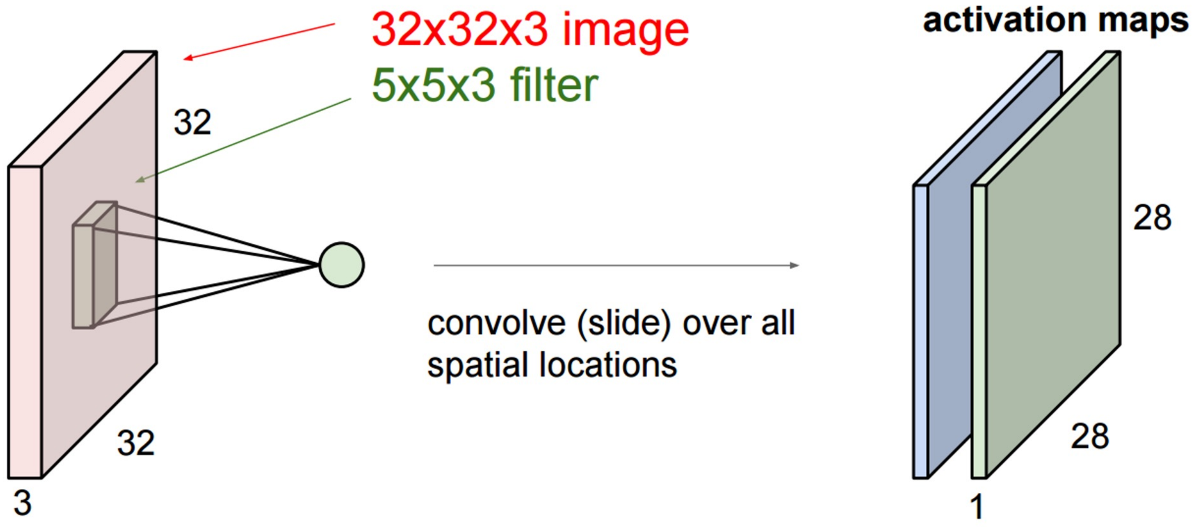
One layer network with two filters



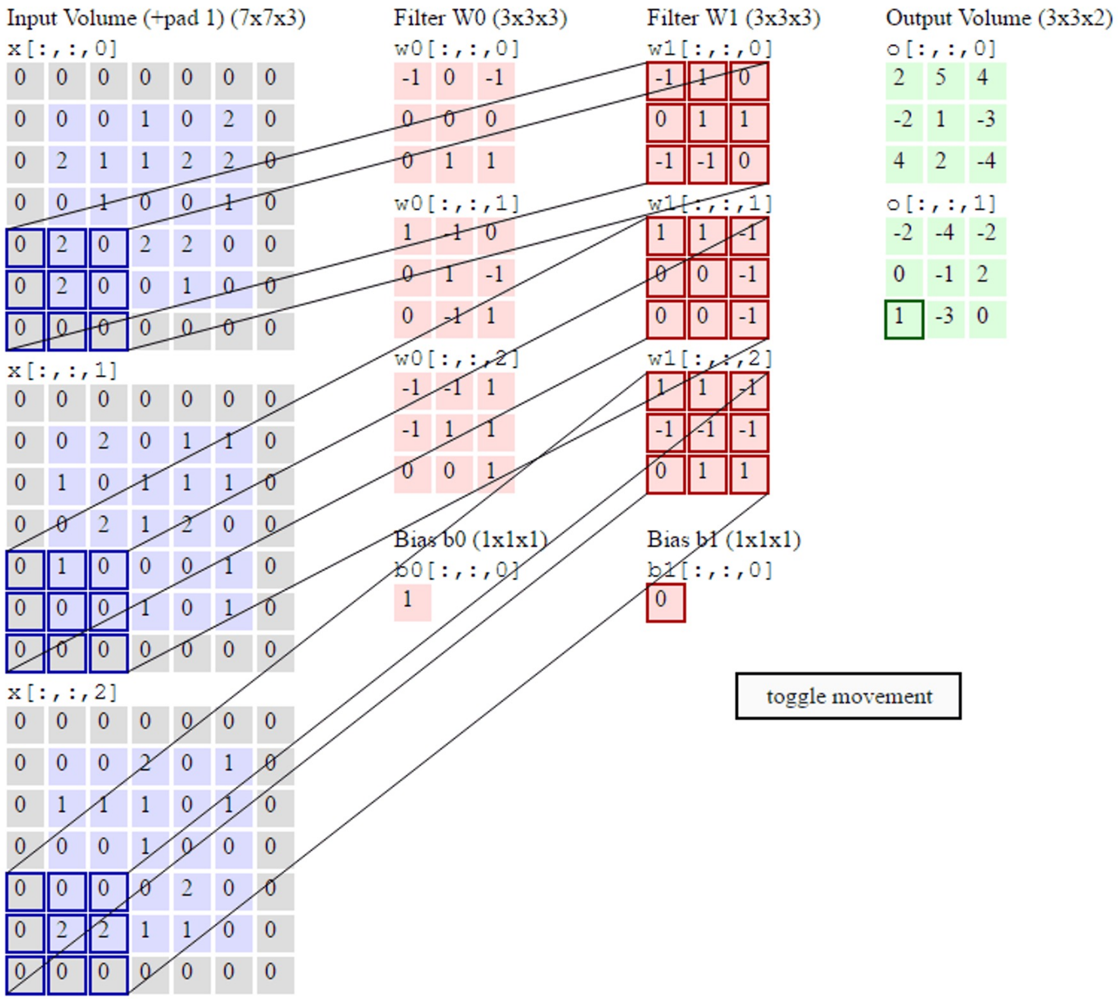


One layer network with two filters





One layer network with two filters

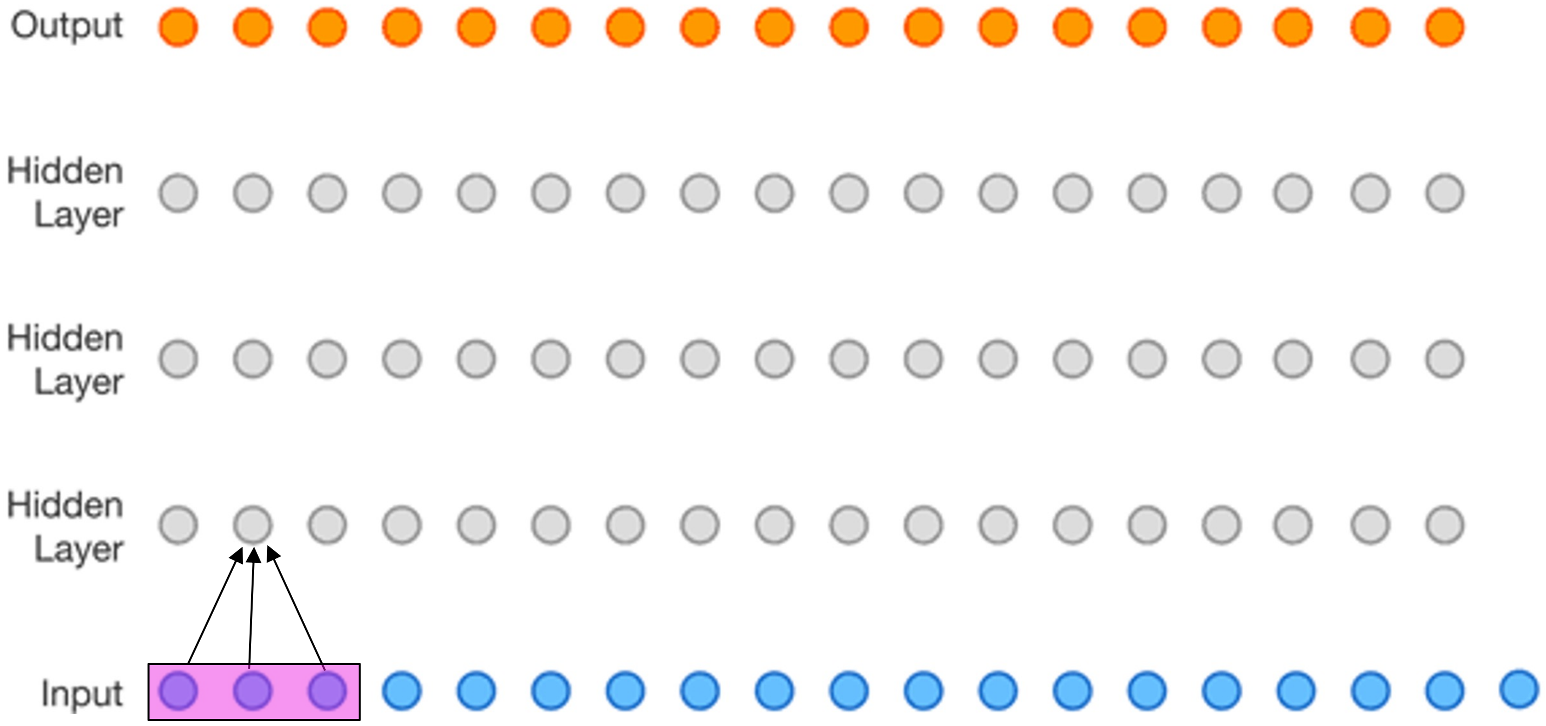


Field of view

- A convolution kernel has a field of view determined by the filter size
- A sliding classifier will thus have access to only a limited part of the image
- How can we go about to ensure a sufficiently wide field of view?

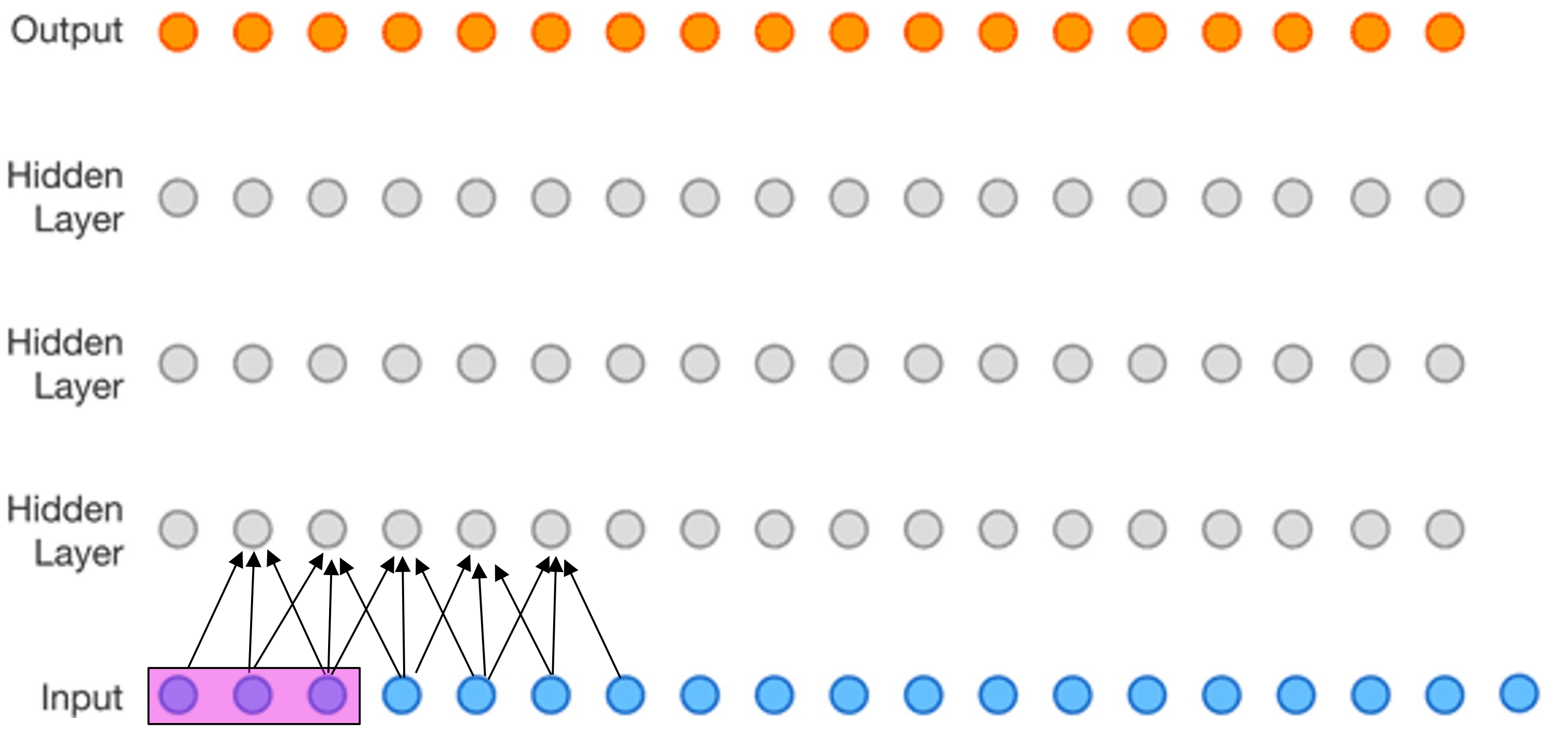
Field of view

- The **field of view** will increase when moving to a higher level in the CNN



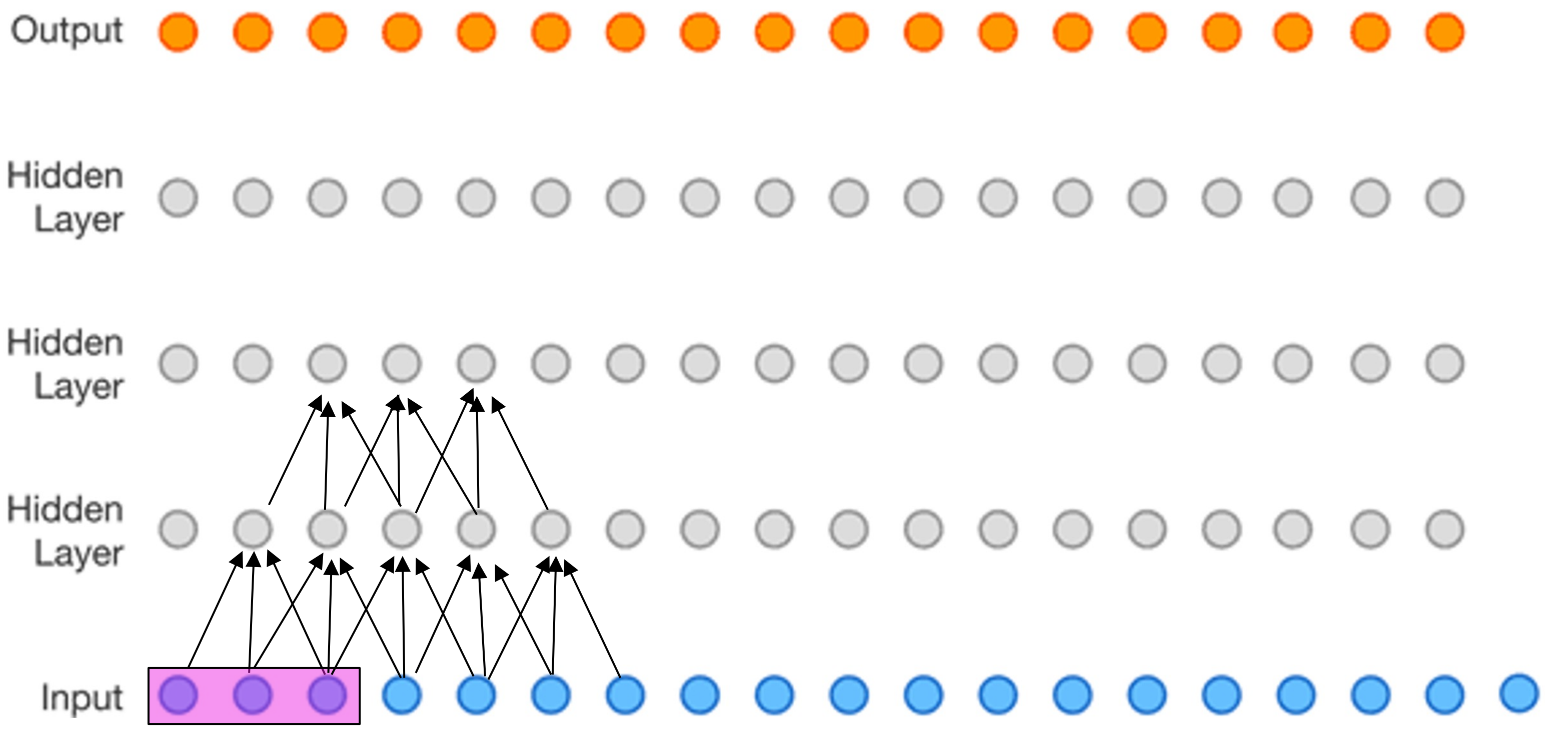
Field of view

- The **field of view** will increase when moving to a higher level in the CNN



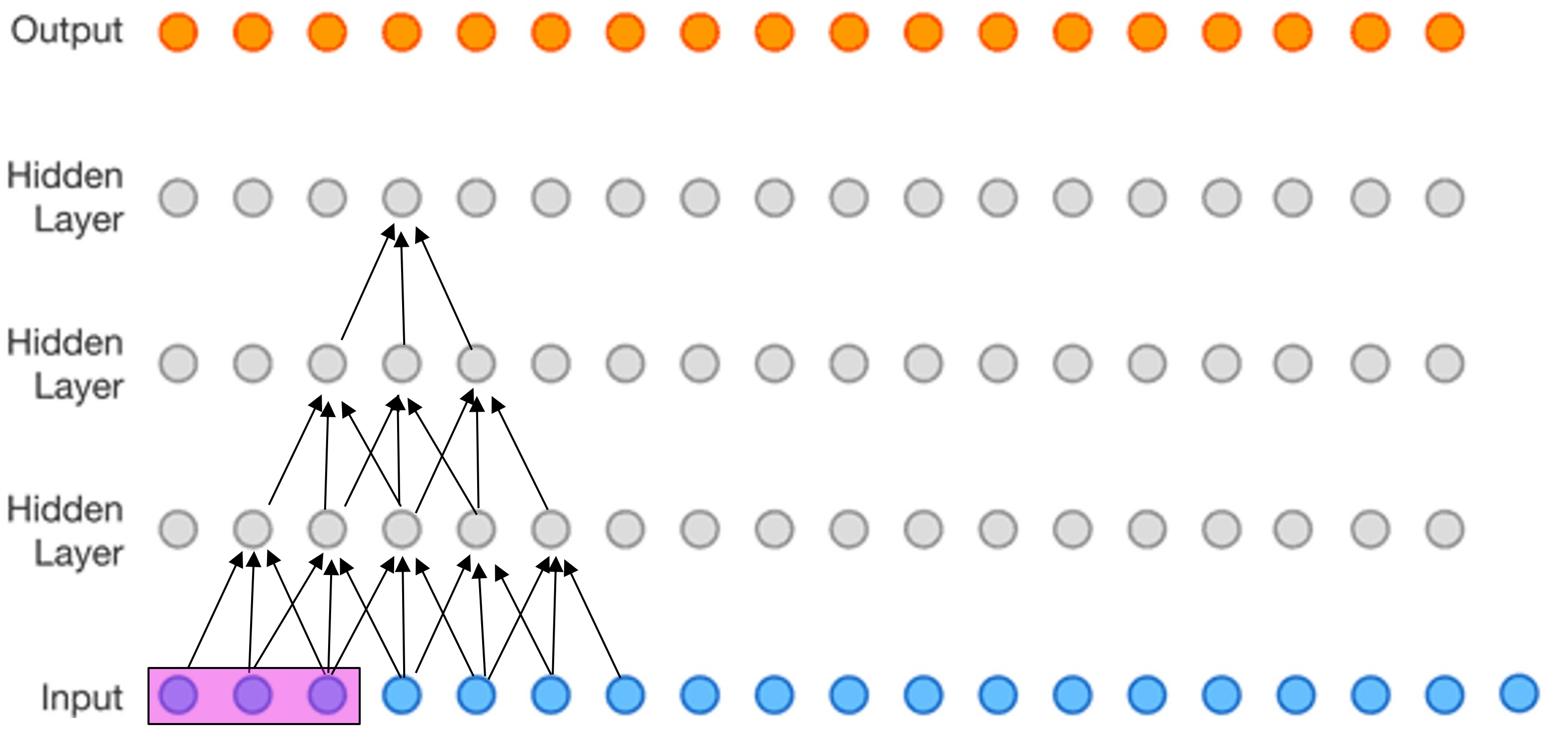
Field of view

- The **field of view** will increase when moving to a higher level in the CNN



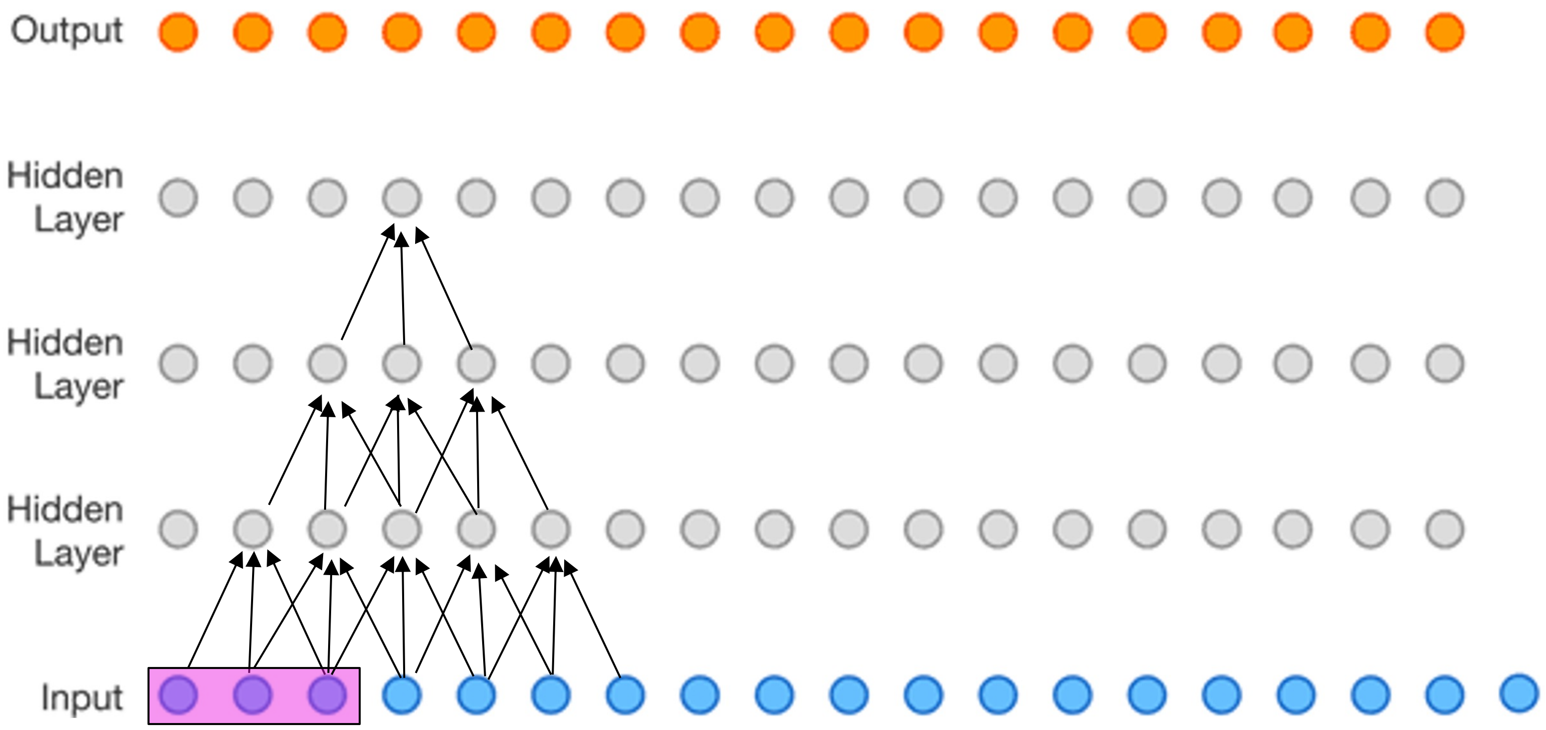
Field of view

- The **field of view** will increase when moving to a higher level in the CNN



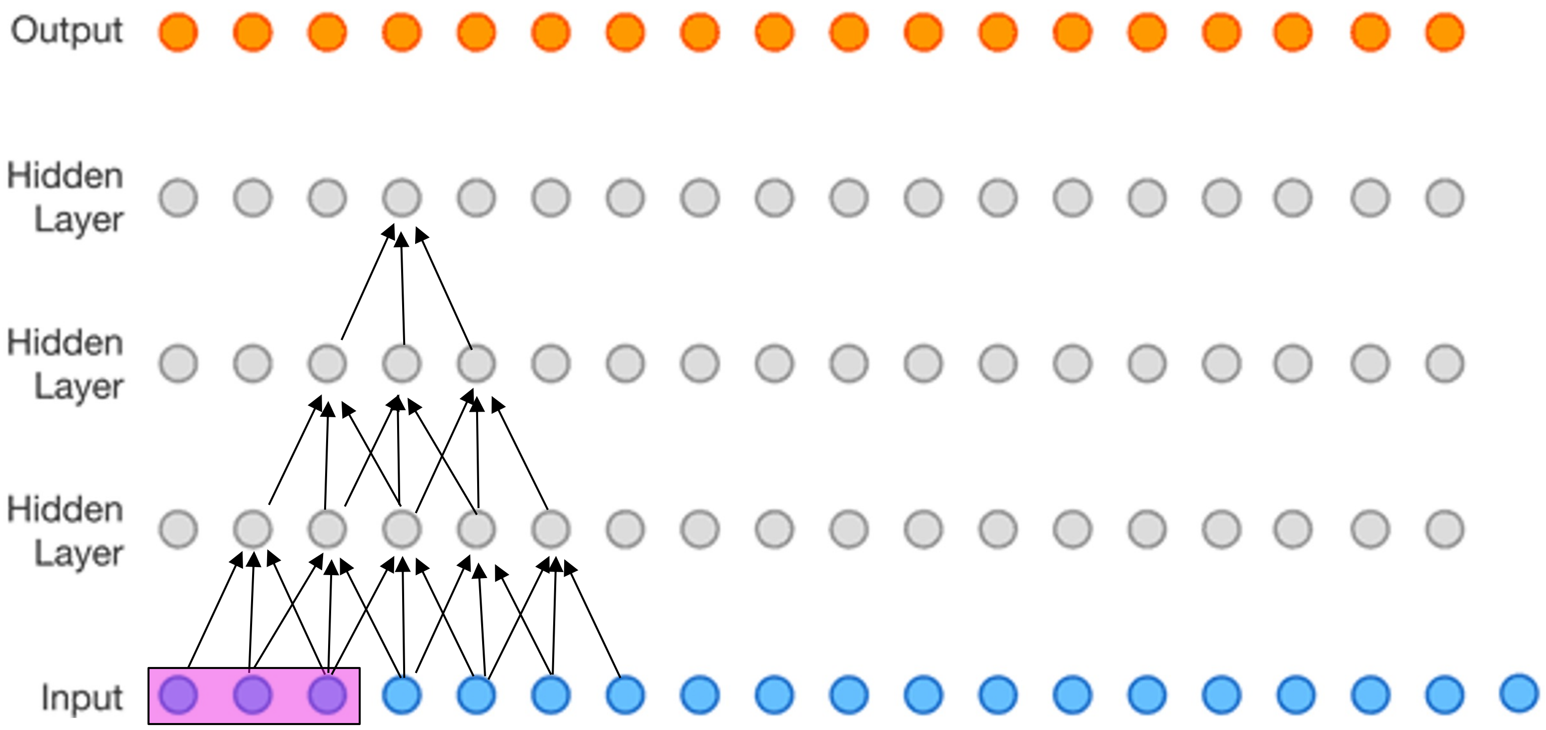
Field of view increases with k-1 for each layer

- Here **k=3** is the kernel size
- Field of view increases by two pixels for each layer in this case



Large kernels or many layers?

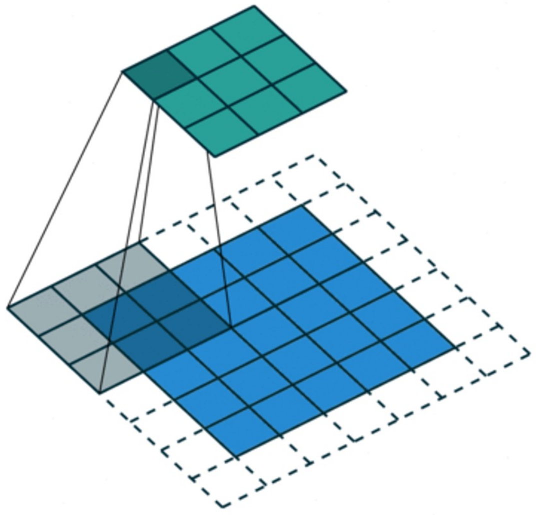
- Small kernels are more efficient with respect to number of weights
- More layers may thus be required to give sufficient field of view
- Other possibilities?



How to increase the field of view more efficiently?

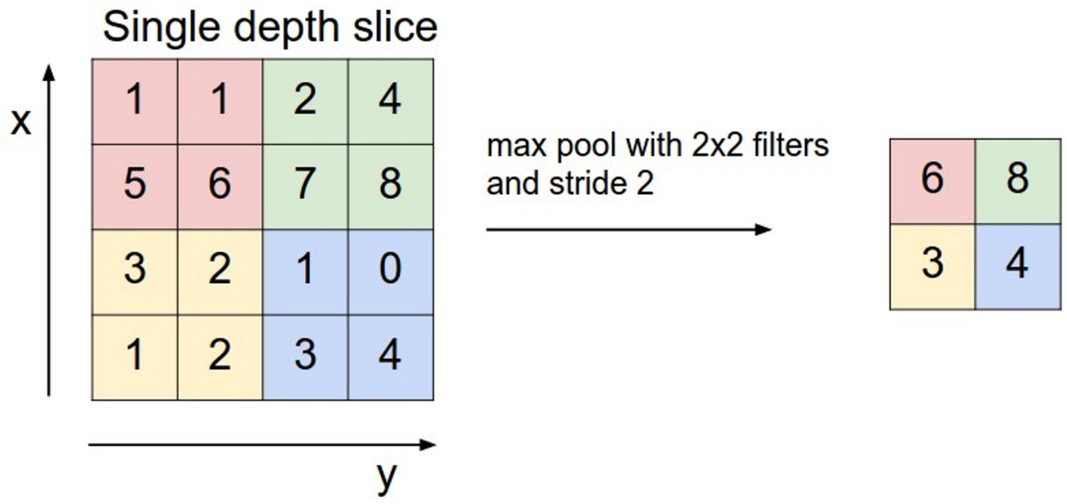
Strided convolutions:

- Most common method
- Simplest solution



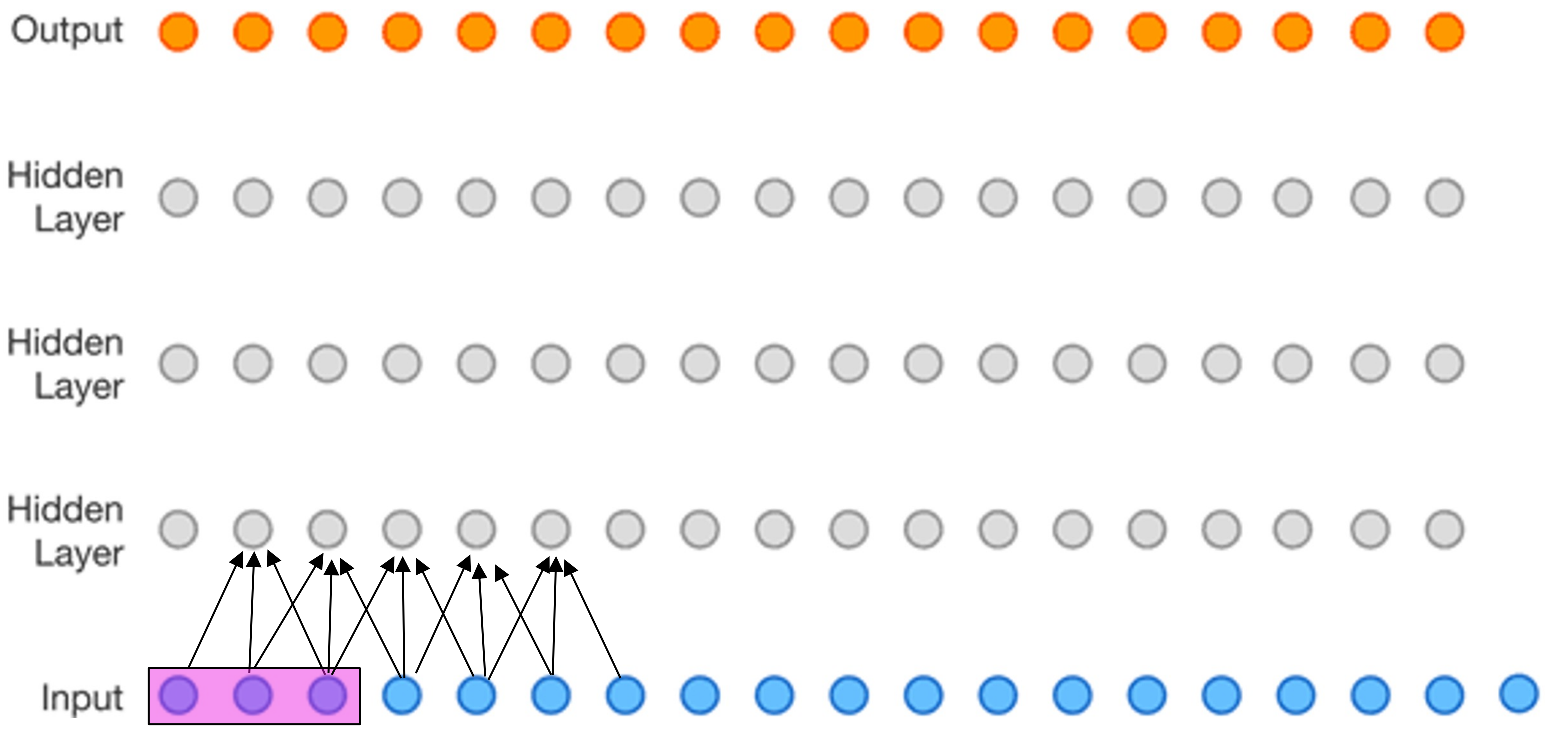
Some form of pooling:

- Max pooling
- Average pooling



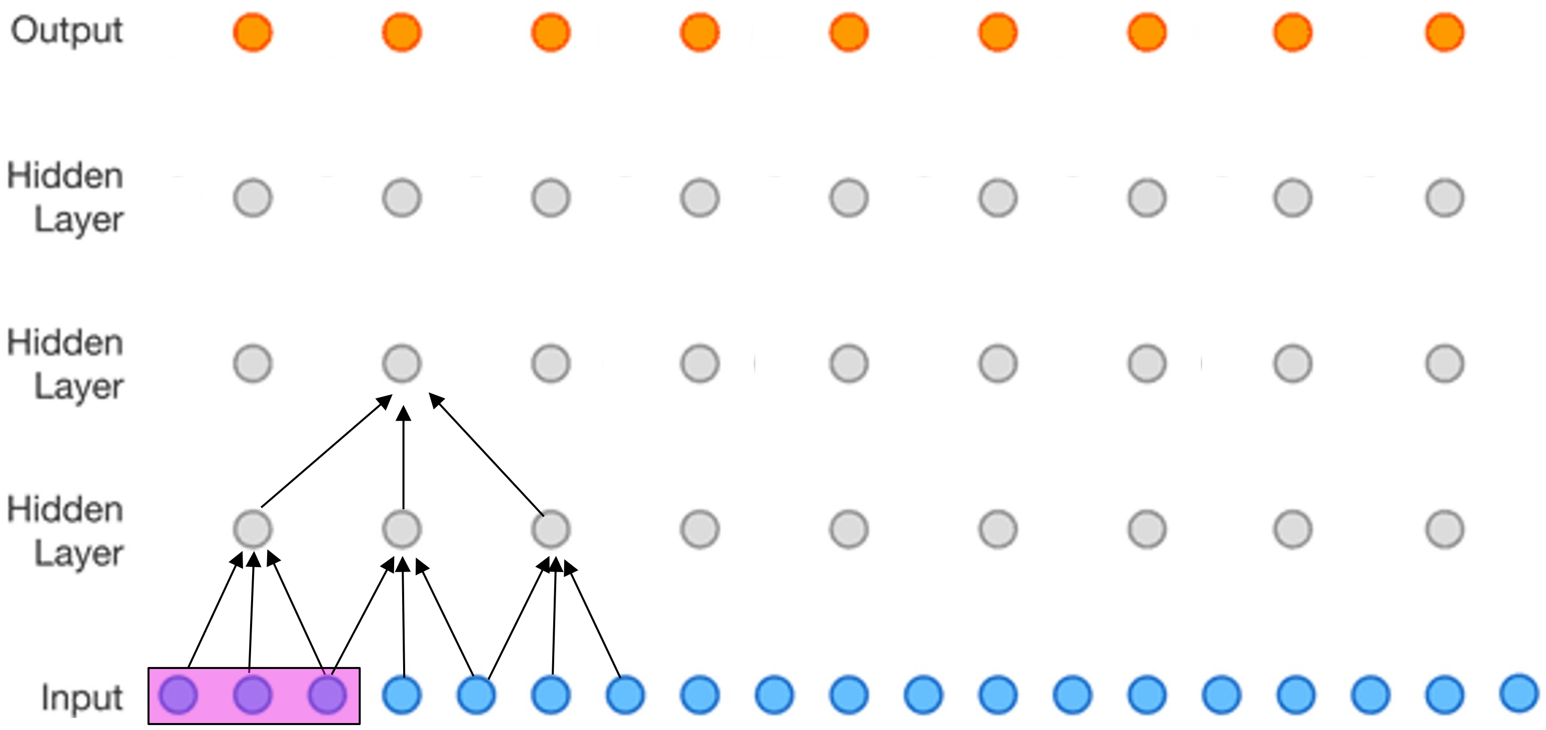
Strided convolutions

- No striding (i.e. **Stride=1**)
- Field of view = 3 pixels in first layer



Strided convolutions

- Striding (**stride=2**)
- Input is fully covered (no holes)
- Increased field of view
- Field of view = 7 pixels in second layer
- Without striding an extra layer was needed



Strided convolutions

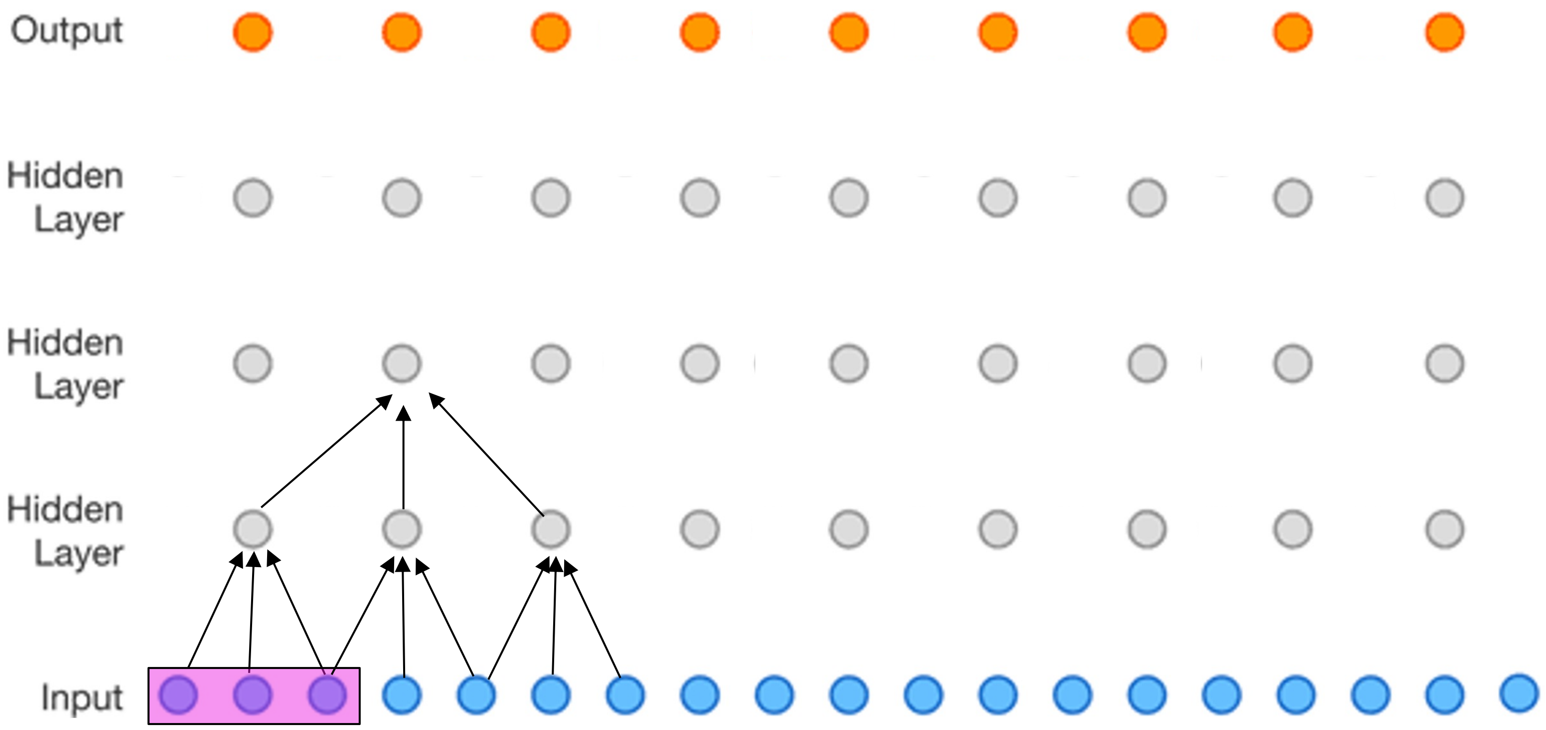
The field of view for layer k is given by the function l_k :

$$l_k = l_{k-1} + ((f_k - 1) * \prod_{i=1}^{k-1} s_i)$$

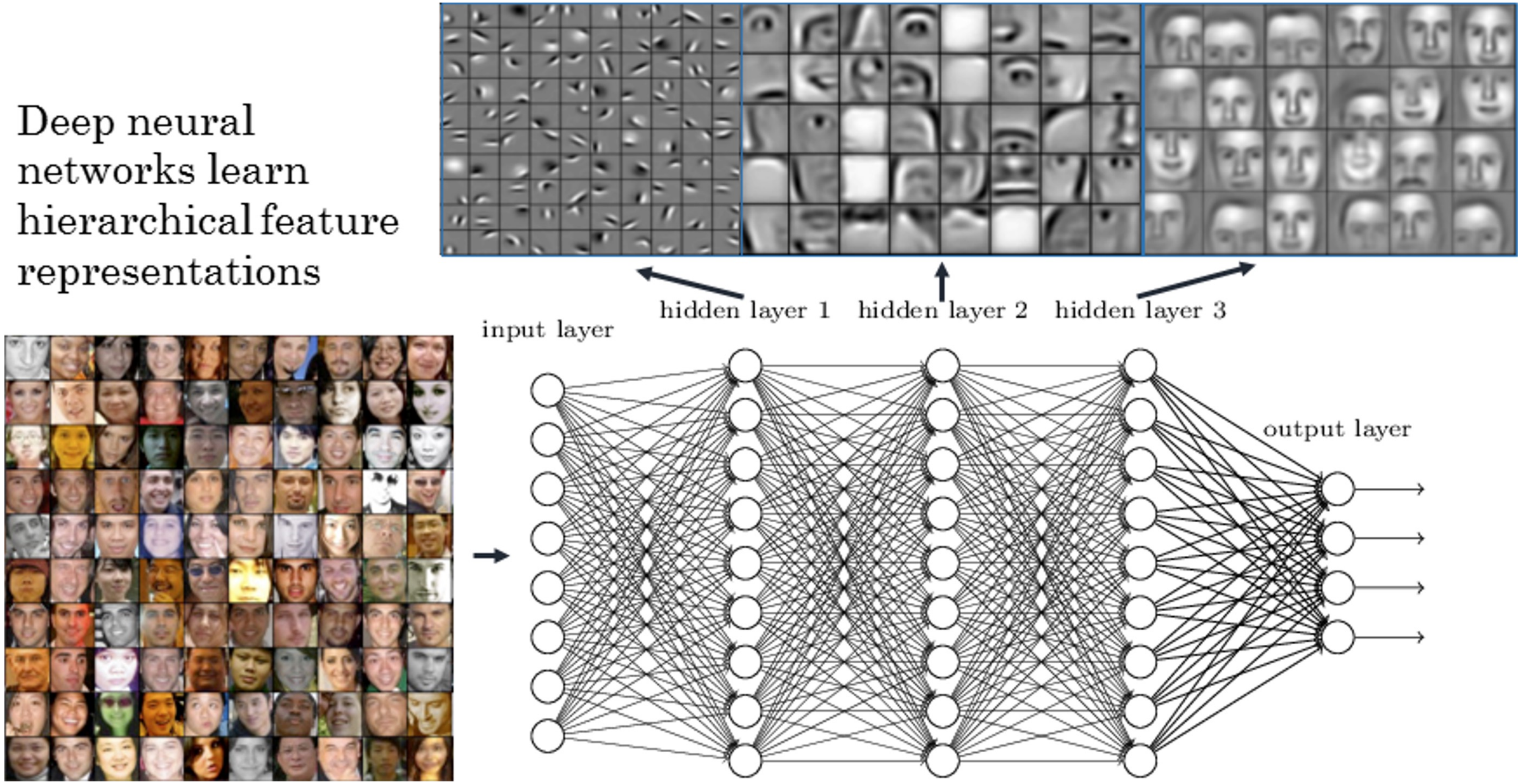
Here l_{k-1} is the field of view of the previous layer, f_i is the filter size and s_i is the stride for level *i.*

The field of view for the second layer is thus l_2=7.

Striding can provide a much wider field of view for the same number of layers

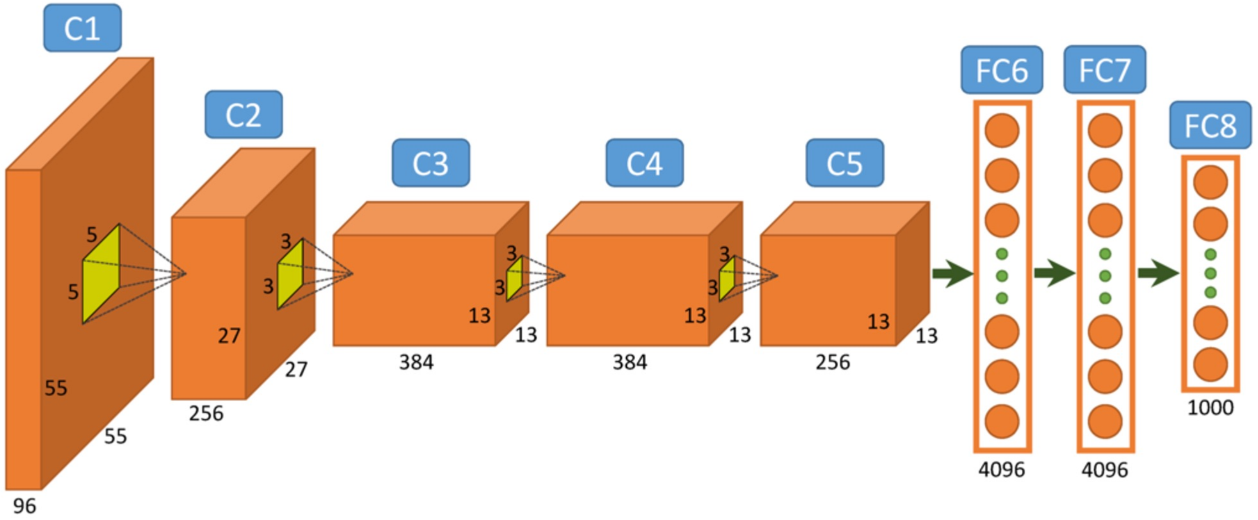


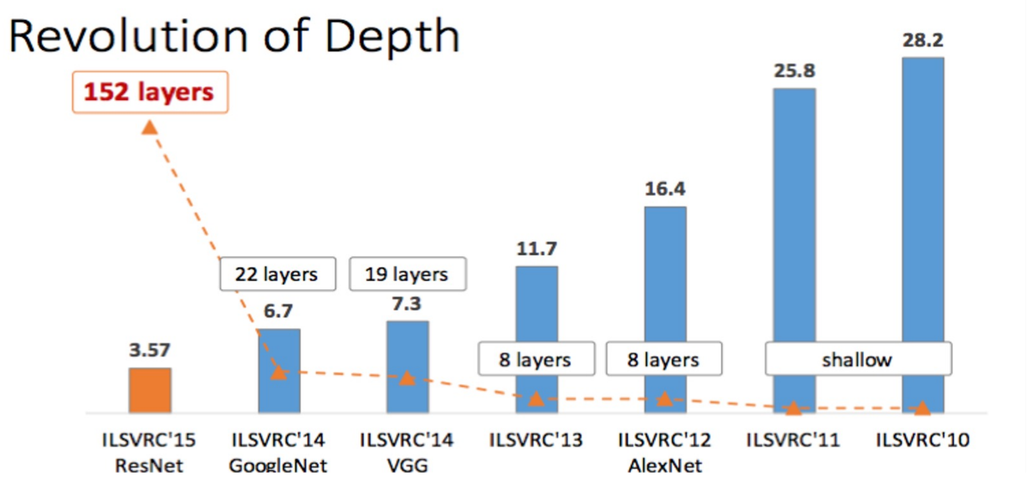
CNNs will learn hierarchical features



Layers in CNN

- Convolutional layers
- Non-linearities (ReLu-layers)
- Pooling layers
- Normalization layers
- Dropout layers
- Fully connected layers
- Softmax layer
- Classification layer





Example - AlexNet

- **SuperVision deep neural network** (2012)
- 5 convolution layers
- 3 fully connected layers
- 60 million parameters
- Trained on 1.2 million images from the ImageNet database
- Augmentation used to increase the training set by a factor of 2048
- Dropout layers
- ReLu activation function
- Won the «2012 ImageNet competition»
- Top-5 error rate 15.3%

	Name	Type	Activations	Learnable Properties
1	data 227×227×3 images with 'zerocenter' nor...	Image Input	227(S) × 227(S) × 3(C) × 1(B)	-
2	conv1 96 11×11×3 convolutions with stride [4 4...	2-D Convolution	55(S) × 55(S) × 96(C) × 1(B)	Weights 11 × 11 × 3 × 96 Bias 1 × 1 × 96
3	relu1 ReLU	ReLU	55(S) × 55(S) × 96(C) × 1(B)	-
4	norm1 cross channel normalization with 5 chan...	Cross Channel Nor...	55(S) × 55(S) × 96(C) × 1(B)	-
5	pool1 3×3 max pooling with stride [2 2] and pa...	2-D Max Pooling	27(S) × 27(S) × 96(C) × 1(B)	-
6	conv2 2 groups of 128 5×5×48 convolutions wi...	2-D Grouped Conv...	27(S) × 27(S) × 256(C) × 1(B)	Weights 5 × 5 × 48 × 128 × 2 Bias 1 × 1 × 128 × 2
7	relu2 ReLU	ReLU	27(S) × 27(S) × 256(C) × 1(B)	-
8	norm2 cross channel normalization with 5 chan...	Cross Channel Nor...	27(S) × 27(S) × 256(C) × 1(B)	-
9	pool2 3×3 max pooling with stride [2 2] and pa...	2-D Max Pooling	13(S) × 13(S) × 256(C) × 1(B)	-
10	conv3 384 3×3×256 convolutions with stride [1 ...	2-D Convolution	13(S) × 13(S) × 384(C) × 1(B)	Weights 3 × 3 × 256 × 384 Bias 1 × 1 × 384
11	relu3 ReLU	ReLU	13(S) × 13(S) × 384(C) × 1(B)	-
12	conv4 2 groups of 192 3×3×192 convolutions ...	2-D Grouped Conv...	13(S) × 13(S) × 384(C) × 1(B)	Weights 3 × 3 × 192 × 192 × 2 Bias 1 × 1 × 192 × 2
13	relu4 ReLU	ReLU	13(S) × 13(S) × 384(C) × 1(B)	-
14	conv5 2 groups of 128 3×3×192 convolutions ...	2-D Grouped Conv...	13(S) × 13(S) × 256(C) × 1(B)	Weights 3 × 3 × 192 × 128 × 2 Bias 1 × 1 × 128 × 2
15	relu5 ReLU	ReLU	13(S) × 13(S) × 256(C) × 1(B)	-
16	pool5 3×3 max pooling with stride [2 2] and pa...	2-D Max Pooling	6(S) × 6(S) × 256(C) × 1(B)	-
17	fc6 4096 fully connected layer	Fully Connected	1(S) × 1(S) × 4096(C) × 1(B)	Weights 4096 × 9216 Bias 4096 × 1
18	relu6 ReLU	ReLU	1(S) × 1(S) × 4096(C) × 1(B)	-
19	drop6 50% dropout	Dropout	1(S) × 1(S) × 4096(C) × 1(B)	-
20	fc7 4096 fully connected layer	Fully Connected	1(S) × 1(S) × 4096(C) × 1(B)	Weights 4096 × 4096 Bias 4096 × 1
21	relu7 ReLU	ReLU	1(S) × 1(S) × 4096(C) × 1(B)	-
22	drop7 50% dropout	Dropout	1(S) × 1(S) × 4096(C) × 1(B)	-
23	fc8 1000 fully connected layer	Fully Connected	1(S) × 1(S) × 1000(C) × 1(B)	Weights 1000 × 4096 Bias 1000 × 1
24	prob softmax	Softmax	1(S) × 1(S) × 1000(C) × 1(B)	-
25	output crossentropyex with 'tench' and 999 oth...	Classification Output	1(S) × 1(S) × 1000(C) × 1(B)	-

Other types of convolutional networks

Object detection:

- Yolo (You only look once) networks (pyramid structure for images and activation maps)
- SSD (Single Shot MultiBox Detector)

Semantic segmentation:

- U-nets
- Downsampling (encoding, e.g. max pooling)
- Bootleneck
- Upsampling (decoding, interpolation by convolution)
- Skip connections between encoder and decoder

Summary

Convolutional Neural Networks:
- Deep neural networks
- Convolutional layers
- Field of view
- Other layers in a CNN

Recommended reading:
- Szeliski 5.4