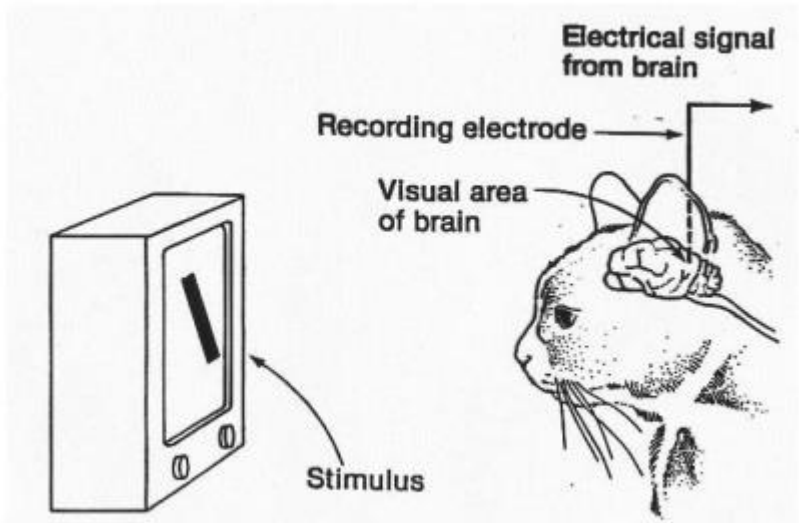
practical deep learning
convolutional neural networks

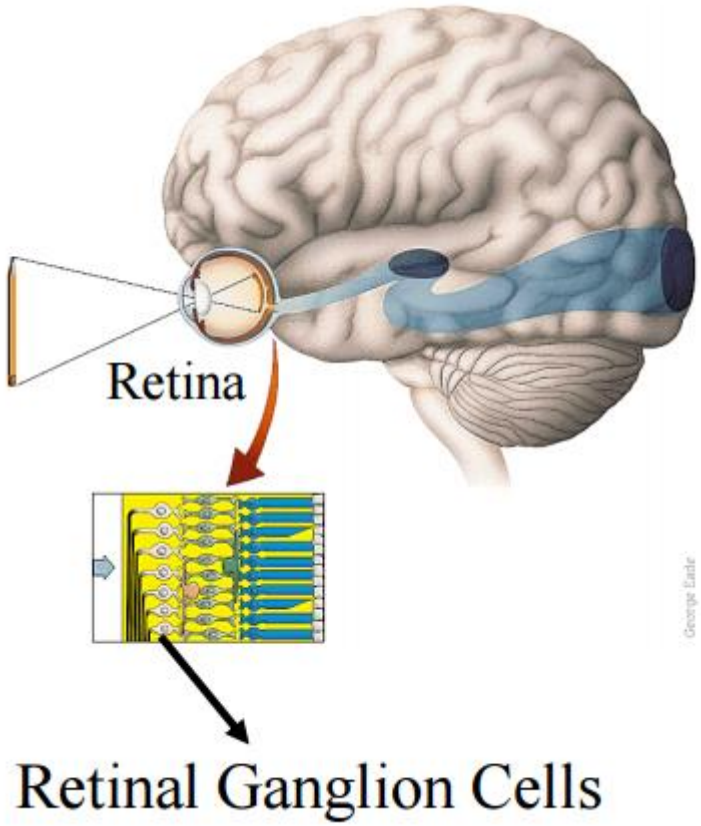
Alex Honchar
University of Verona

Day 3 goals

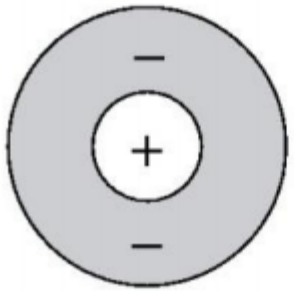
- You understand how convolutional neural network (CNN) works
- You understand modern approaches to CNNs
- You can train your own CNN for different computer vision problems
- You can use **Caffe** framework

Mammals vision system

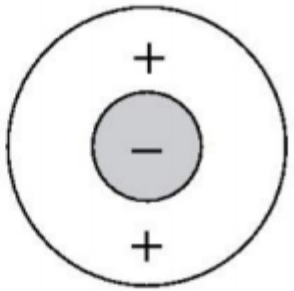




Mammals vision system

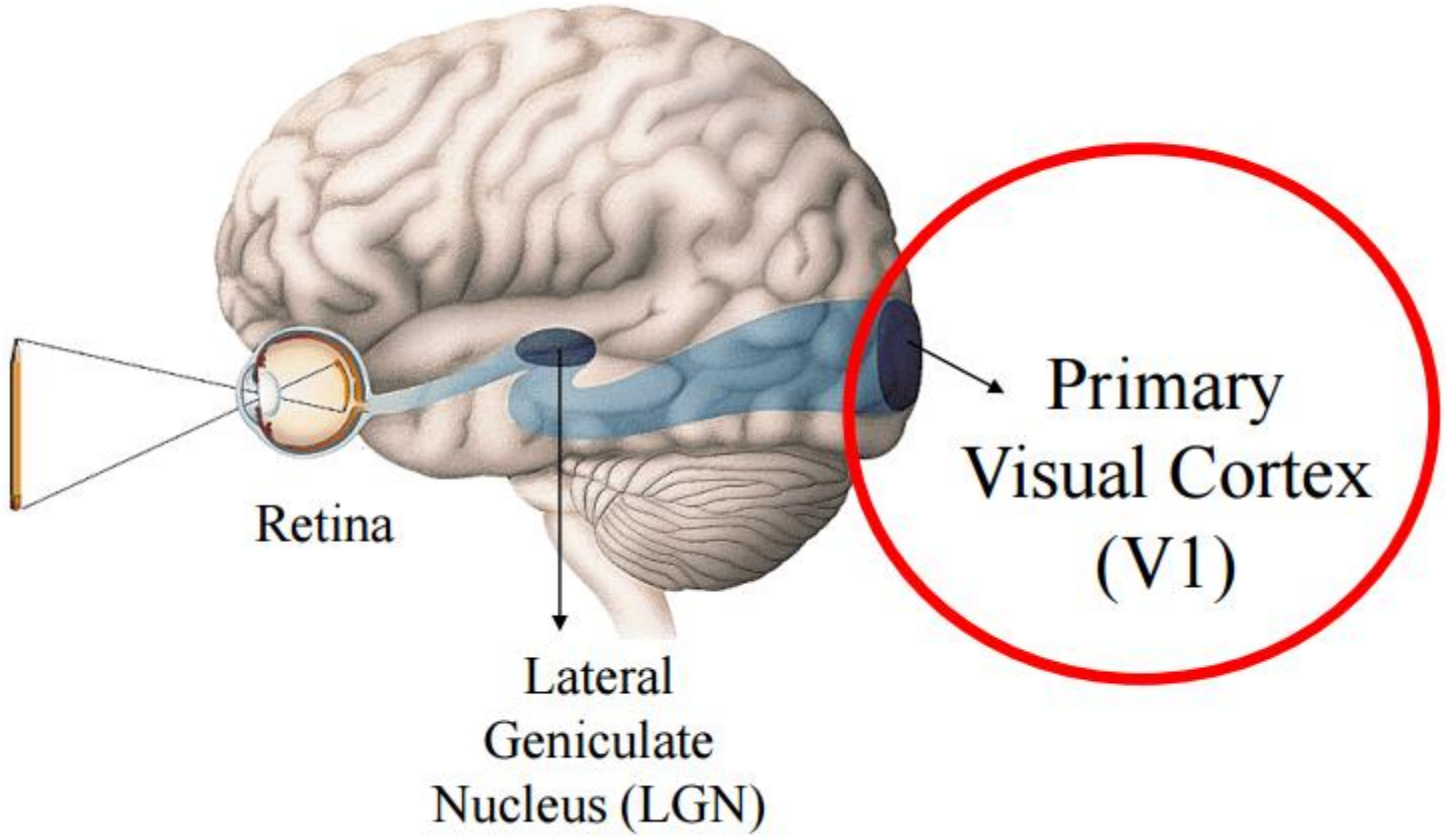


On-Center
Off-Surround
Receptive Field

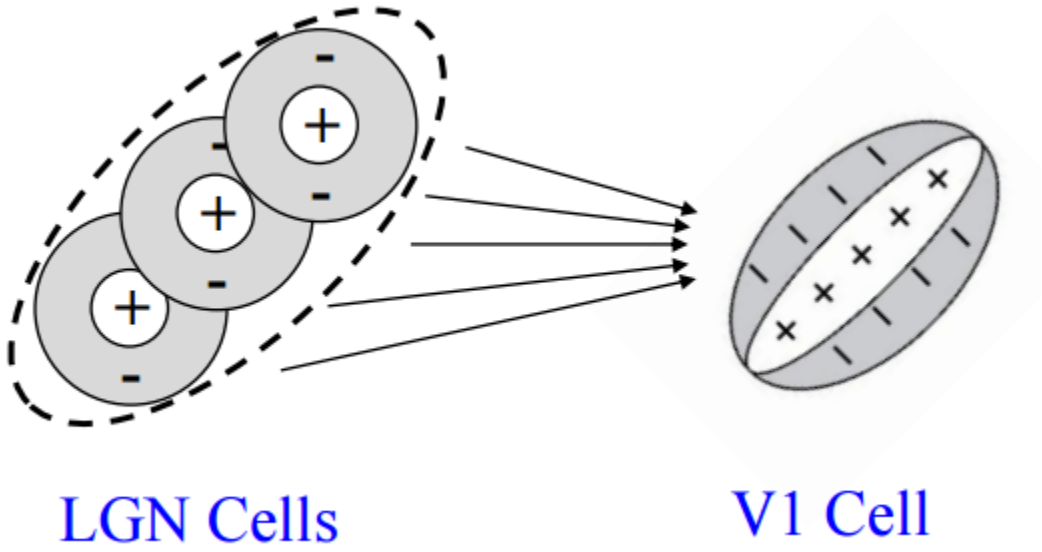


Off-Center
On-Surround
Receptive Field

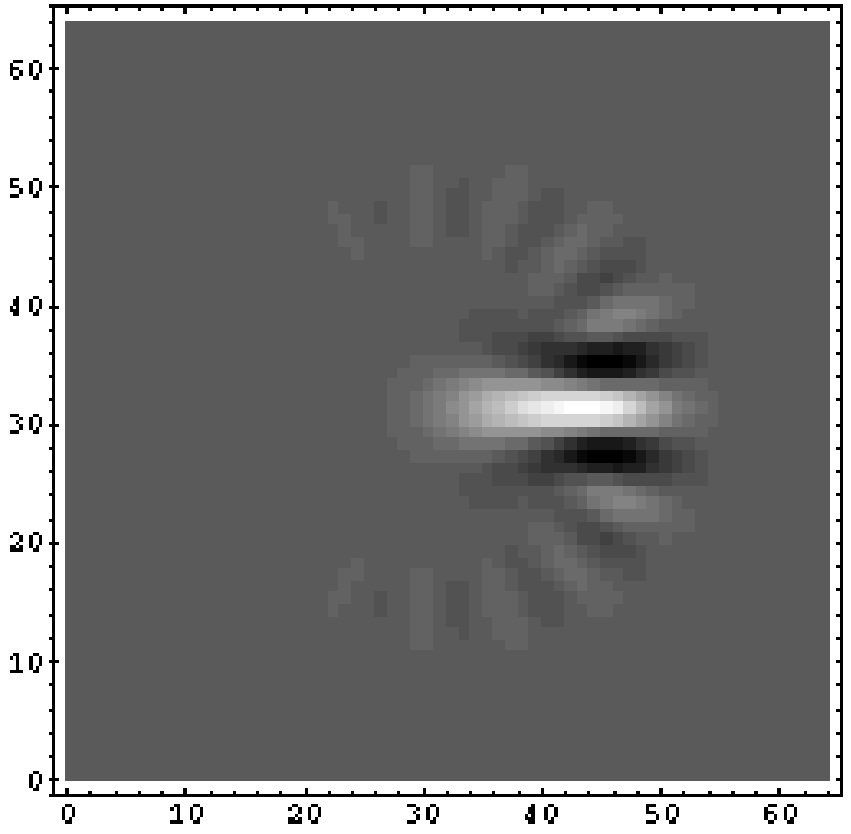
Mammals vision system



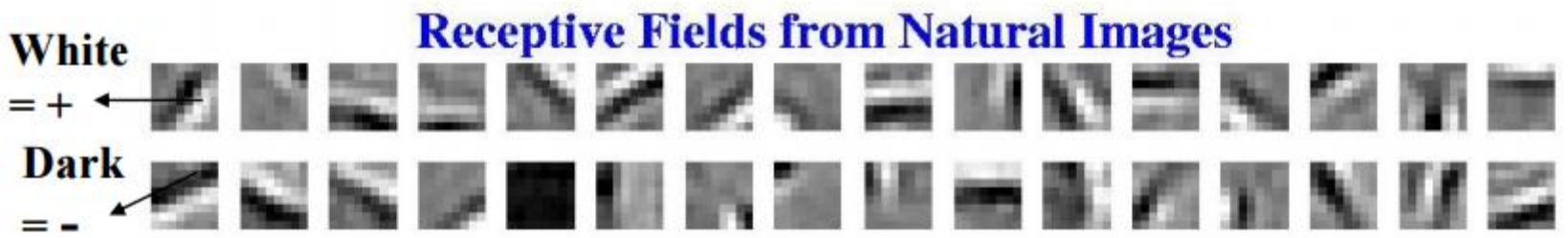
Mammals vision system

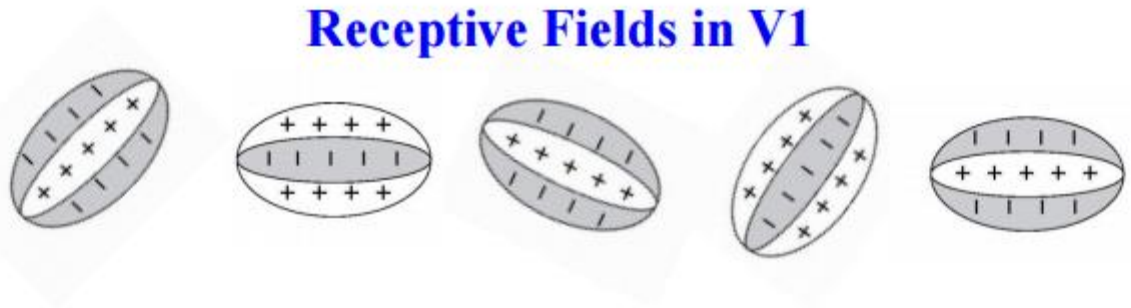


Mammals vision system



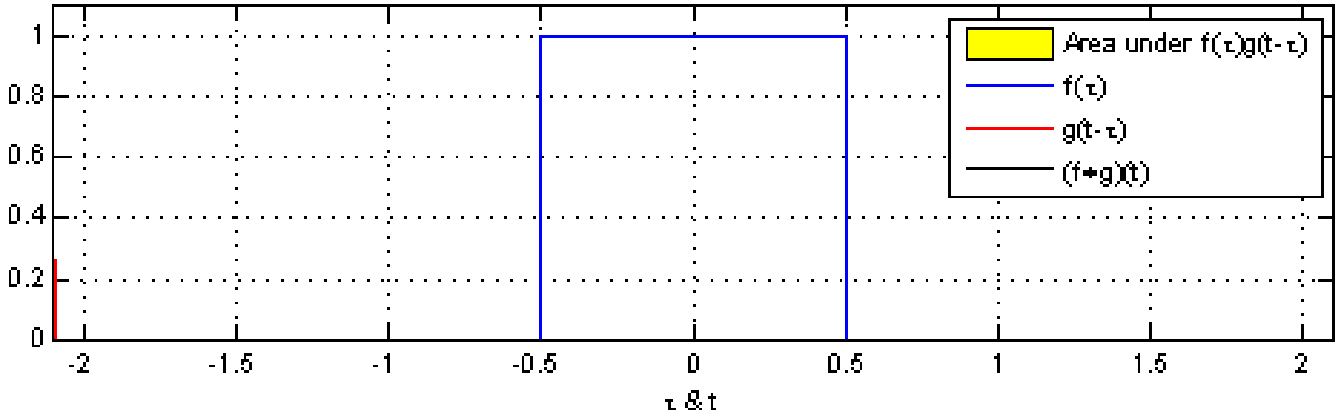
Mammals vision system



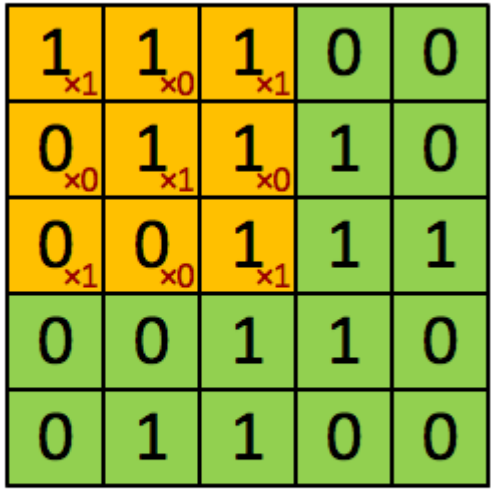


Convolution

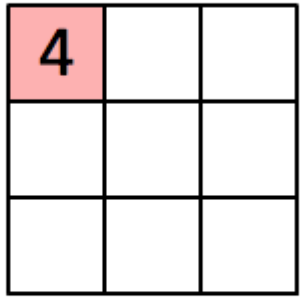
$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$
$$= \int_{-\infty}^{\infty} f(t-\tau)\, g(\tau)\, d\tau.$$



Convolution as kernel



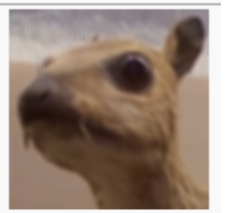
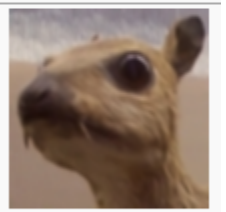
Image



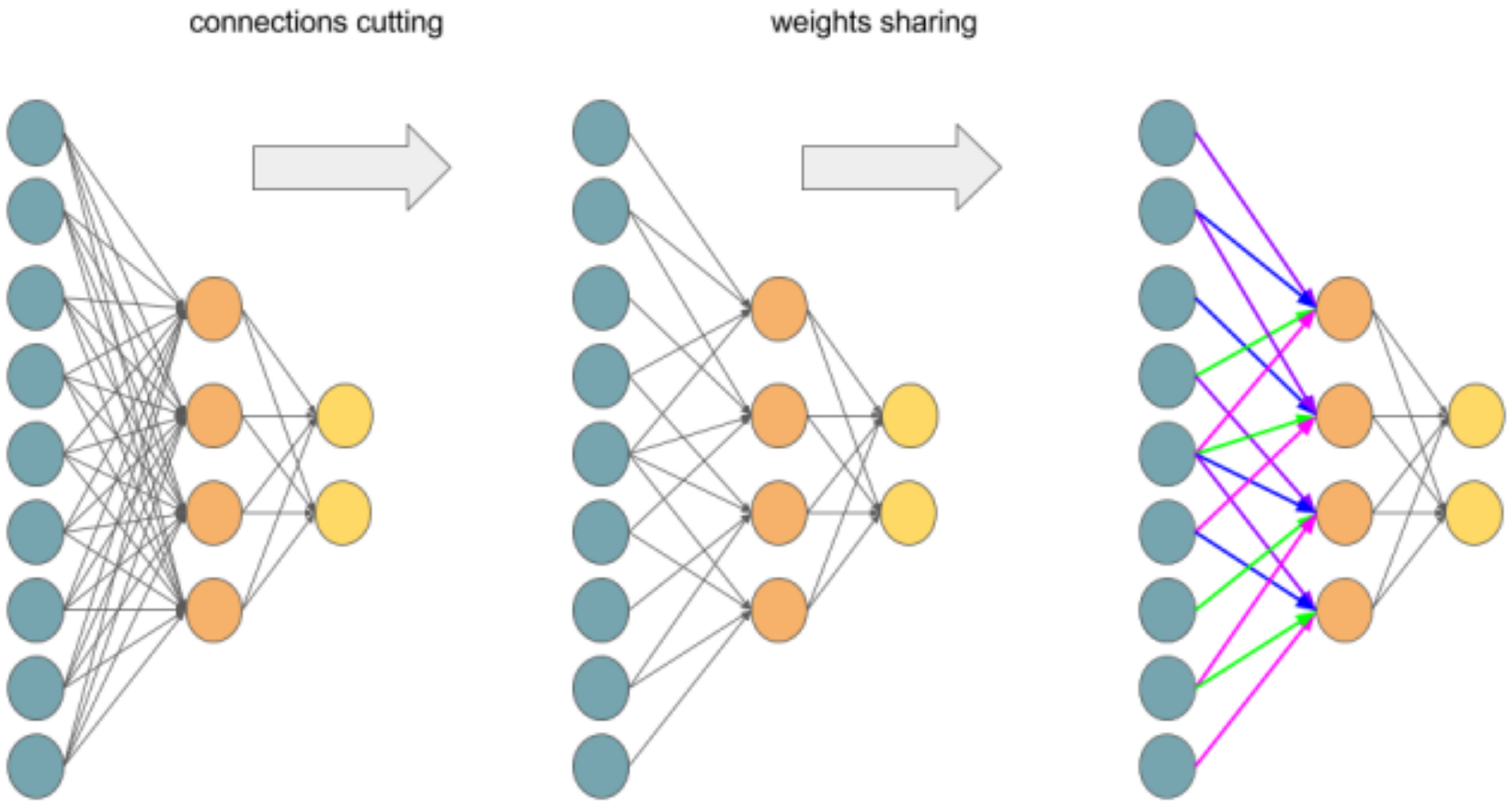
Convolved Feature

Convolution as kernel

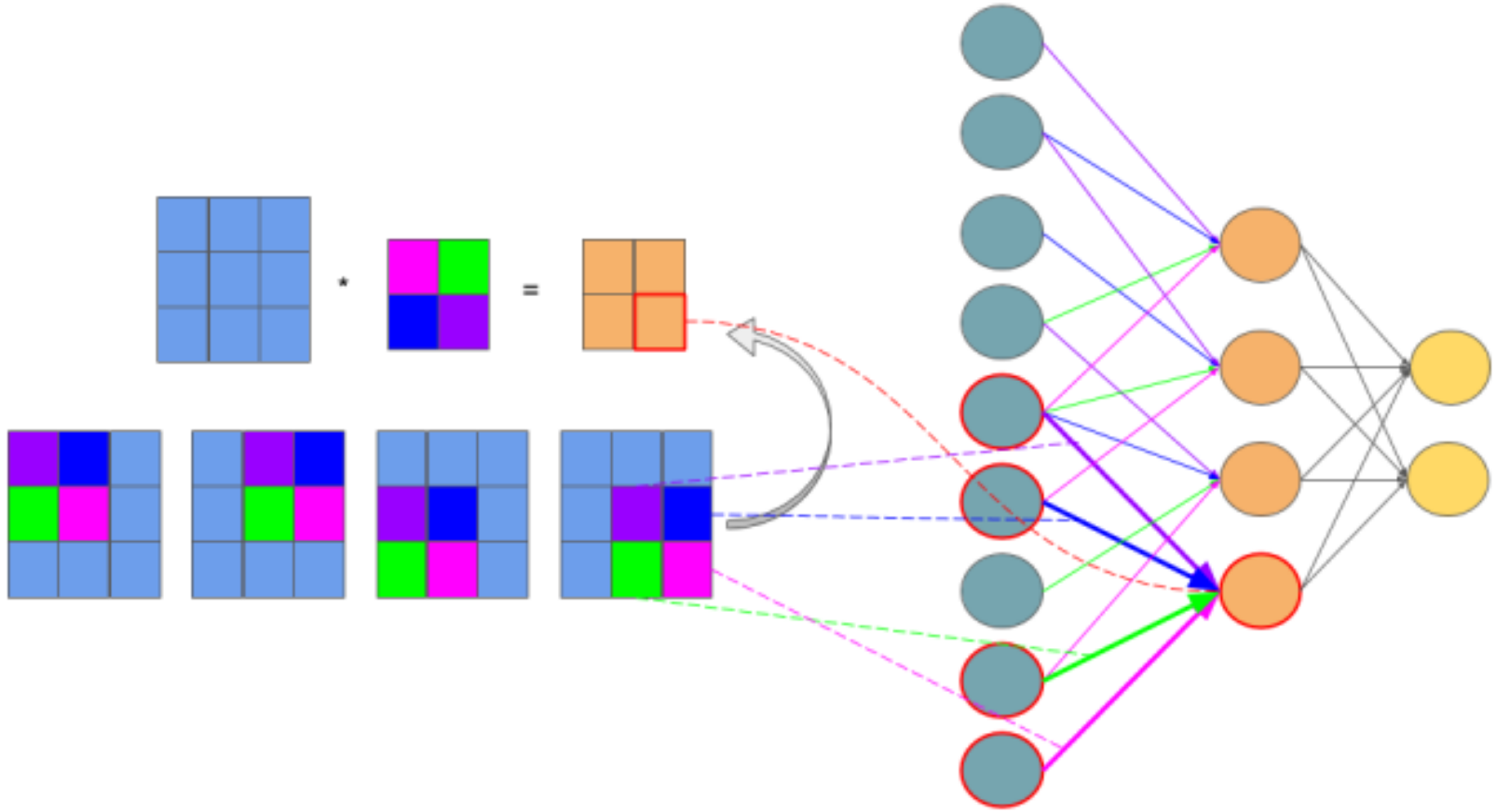
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
5×5 **Unsharp** (with no image mask)	$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

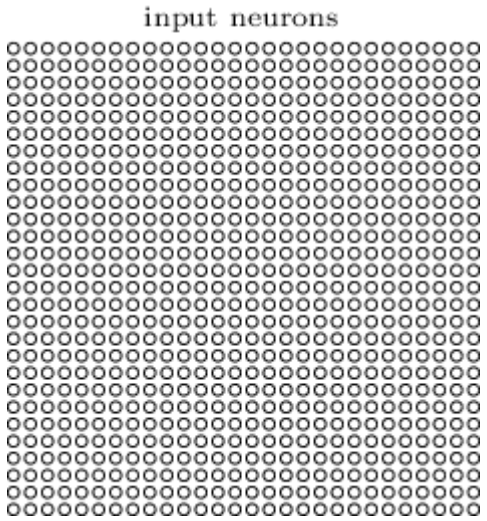
MLP -> CNN

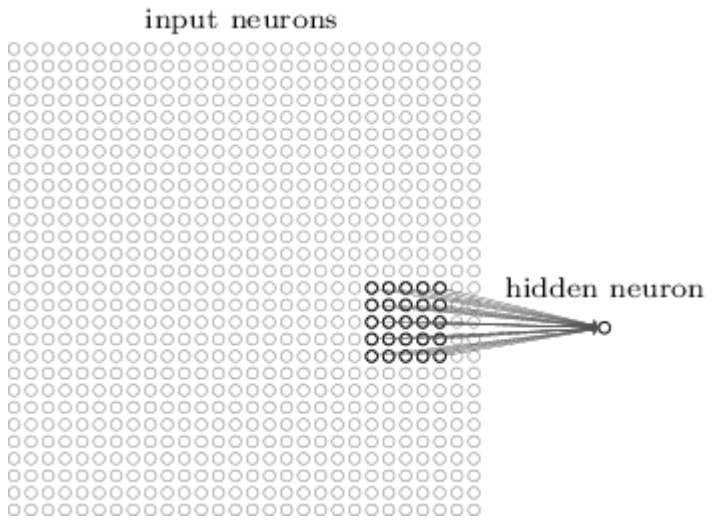


MLP -> CNN

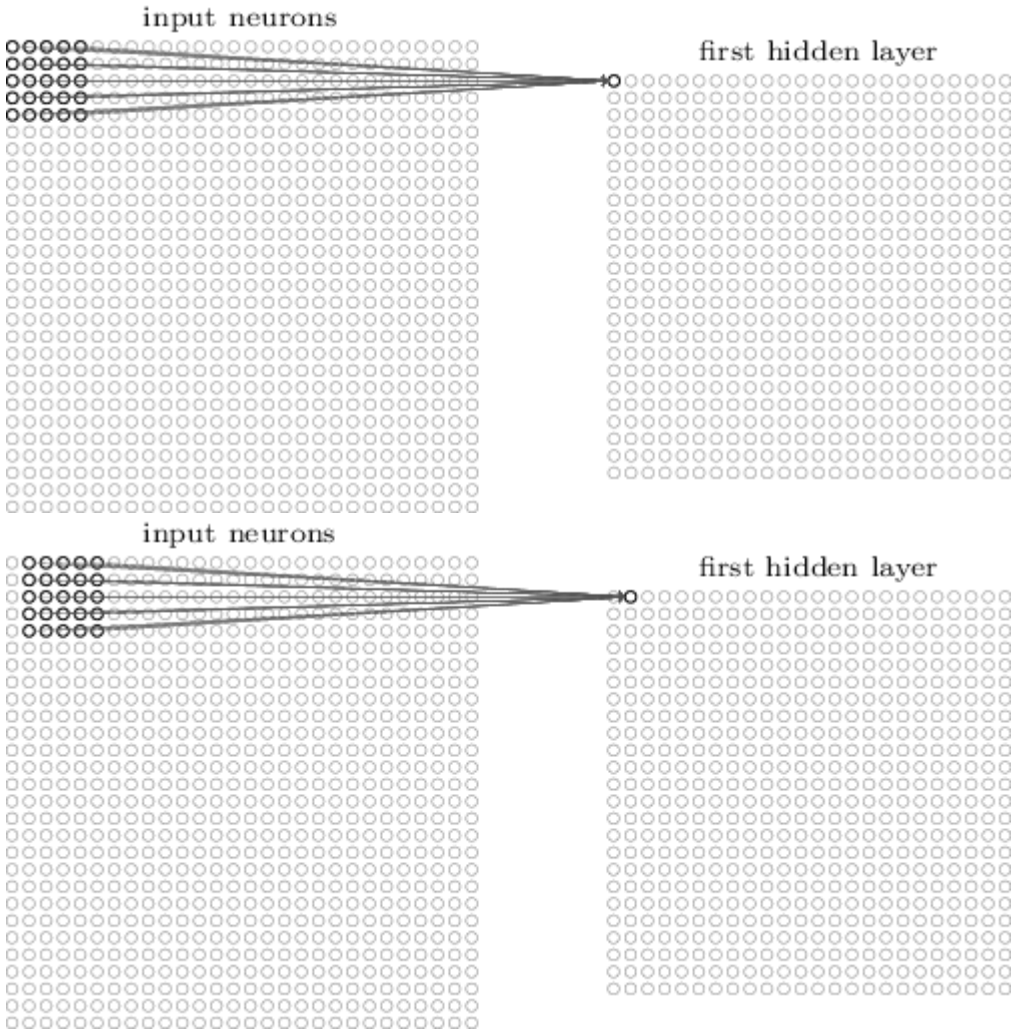


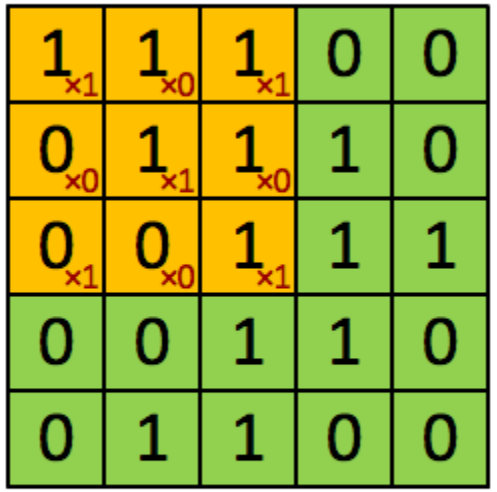
CNN



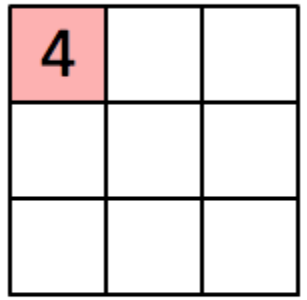


CNN: convolutional layer



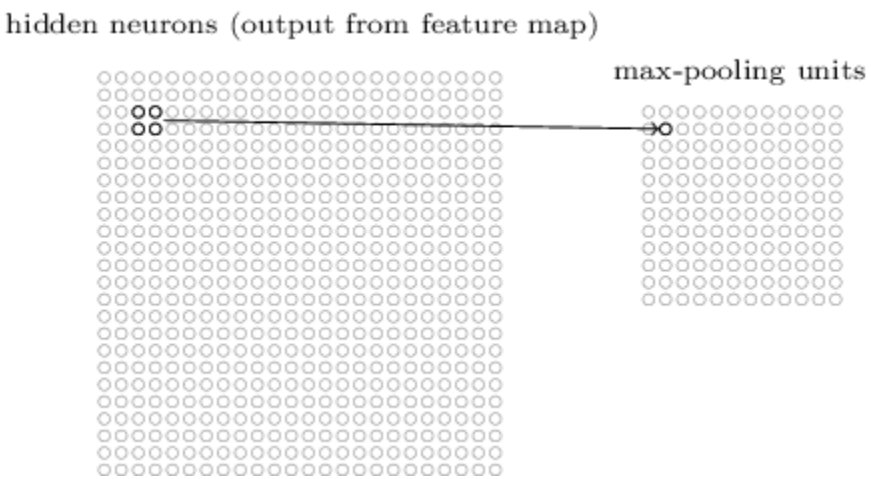


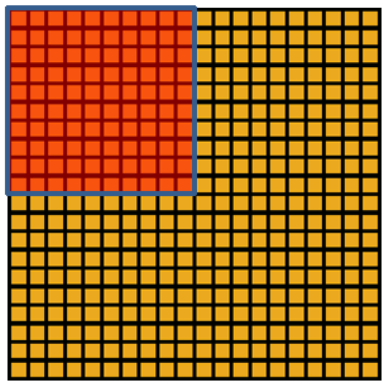
Image



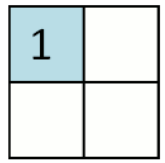
Convolved Feature

CNN: pooling layer





Convolved feature



Pooled feature

CNN:building block

- Convolution (linear activations)
- ReLU (detection stage)
- Pooling (modify output for next layer)

https://www.youtube.com/watch?v=AgkfIQ4IGaM&t=141s

https://www.youtube.com/watch?v=GhqOMJIHD8A

cnn backprobagation

http://www.simon-hohberg.de/2014/10/10/conv-net.html

CNN backpropagation

$$v\left(x,y\right)=\sum_{x_k=0}^{N_k-1}\sum_{y_k=0}^{N_k-1}i\left(x-x_k,y-y_k\right)k\left(x_k,y_k\right)$$

$$l\left(x,y\right)=\sum_{x_k=0}^{N_k-1}\sum_{y_k=0}^{N_k-1}i\left(x+x_k,y+y_k\right)k\left(x_k,y_k\right)$$

CNN backpropagation

$$i_{n,m}^{l}(x, y) = \sum o_{n}^{l-1}(x - x', y - y') \cdot w_{n,m}^{l}(x', y')$$

$$\begin{aligned} c_{m}^{l}(x, y) &= \sum_{n} i_{n,m}^{l}(x, y) + b_{m}^{l} \\ &= \sum_{n, x', y'} o_{n}^{l-1}(x - x', y - y') \cdot w_{n,m}^{l}(x', y') + b_{m}^{l} \end{aligned}$$

$$o_{m}^{l}(x, y) = a\left(c_{m}^{l}(x, y)\right)$$

CNN weight update

$$\frac{\partial E}{\partial w^l_{n,m}(x,\,y)} = \sum_{x',y'} \underbrace{\frac{\partial E}{\partial c^l_m(x',\,y')}}_{\delta^l_m(x',\,y')} \cdot \frac{\partial c^l_m(x',\,y')}{\partial w^l_{n,m}(x,\,y)}$$

$$\begin{aligned}
\frac{\partial c^l_m(x',\,y')}{\partial w^l_{n,m}(x,\,y)} &= \frac{\partial}{\partial w^l_{n,m}(x,\,y)} \left(\sum_{n,\,x'',\,y''} o^{l-1}_n(x'-x'',\,y'-y'') \cdot w^l_{n,m}(x'',\,y'') + b^l_m \right) \\
&= \frac{\partial}{\partial w^l_{n,m}(x,\,y)} \left(o^{l-1}_0(x'-0,\,y'-0) \cdot w^l_{0,m}(0,\,0) + \ldots + o^{l-1}_n(x'-x,\,y'-y) \cdot w^l_{n,m}(x,\,y) + \ldots + b^l_m \right) \\
&= o^{l-1}_n(x'-x,\,y'-y)
\end{aligned}$$

$$\frac{\partial E}{\partial w^l_{n,m}(x,\,y)} = \sum_{x',y'} o^{l-1}_n(x'-x,\,y'-y) \cdot \delta^l_m(x',\,y')$$

CNN weight update

$$\frac{\partial E}{\partial w_{n,m}^{l}(x,\,y)} = \mathrm{rot}_{180}\left(\underbrace{\sum_{x',y'} o_{n}^{l-1}(x+x',\,y+y')\cdot\delta_{m}^{l}(x',\,y')}_{\text{Cross-Correlation}}\right)$$

CNN delta calculation

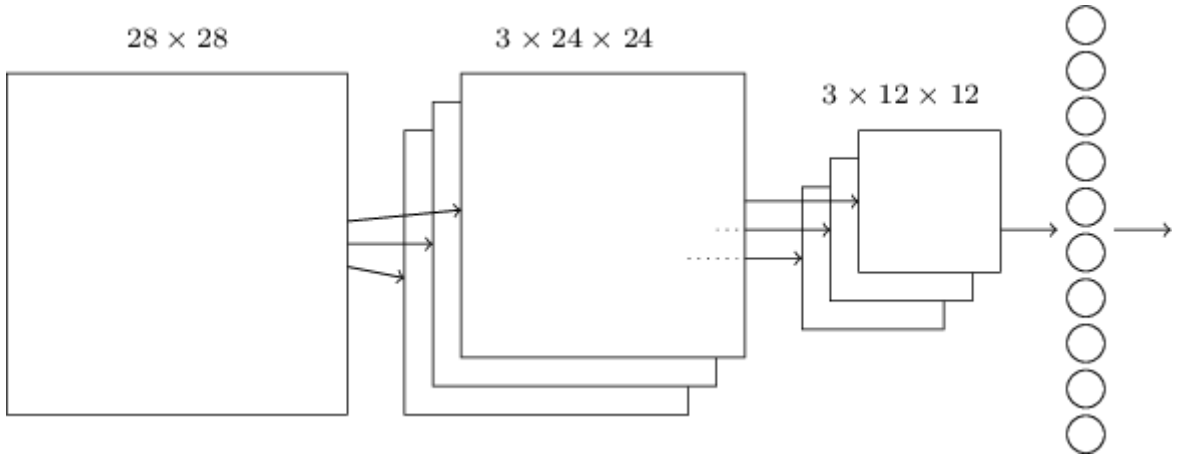
$$\delta_m^l(x, y) = \frac{\partial E}{\partial c_m^l(x, y)} = \sum_o \sum_{x', y'}^{N_w, N_w} \underbrace{\frac{\partial E}{\partial c_o^{l+1}(x + x', y + y')}}_{\delta_o^{l+1}(x+x', y+y')} \cdot \frac{\partial c_o^{l+1}(x + x', y + y')}{\partial c_m^l(x, y)}$$

$$\begin{aligned}
\frac{\partial c_o^{l+1}(x + x', y + y')}{\partial c_m^l(x, y)} &= \frac{\partial}{\partial c_m^l(x, y)} \left(\sum_{m'} \sum_{x'', y''}^{N_w, N_w} o_{m'}^l(x + x' - x'', y + y' - y'') \cdot w_{m',o}^{l+1}(x'', y'') \right) \\
&= \frac{\partial}{\partial c_m^l(x, y)} \left(\sum_{m'} \sum_{x'', y''}^{N_w, N_w} a\left(c_{m'}^l(x + x' - x'', y + y' - y'')\right) \cdot w_{m',o}^{l+1}(x'', y'') \right) \\
&= \frac{\partial}{\partial c_m^l(x, y)} \left(w_{0,o}^{l+1}(0, 0) \cdot a\left(c_0^l(x' - 0, y' - 0)\right) + \ldots + w_{m,o}^{l+1}(x'', y'') \cdot a\left(c_m^l(x, y)\right) + \ldots \right) \\
&= w_{m,o}^{l+1}(x + x', y + y') \cdot \frac{\partial a\left(c_m^l(x, y)\right)}{\partial c_m^l(x, y)}
\end{aligned}$$

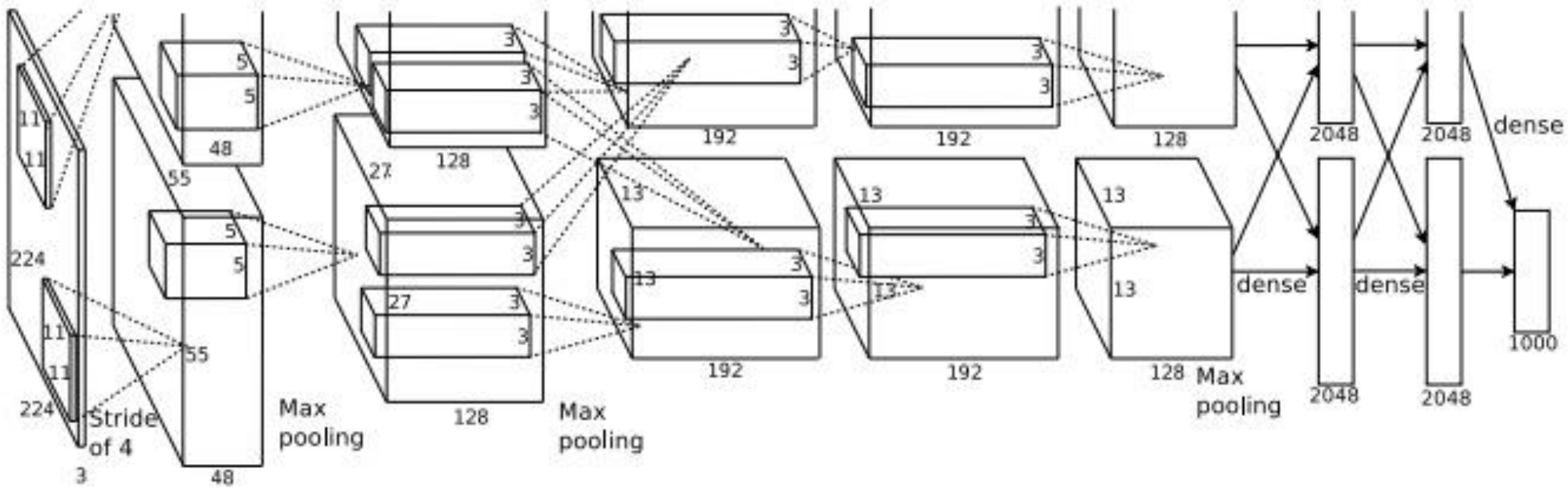
CNN delta calculation

$$\delta_m^l(x, y) = \underbrace{\sum_o \sum_{x', y'}^{N_w, N_w} \delta_o^{l+1}(x + x', y + y') \cdot w_{m,o}^{l+1}(x', y')}_{\text{Cross-Correlation, Backpropagated Error}} \cdot \frac{\partial a\left(c_m^l(x, y)\right)}{\partial c_m^l(x, y)}$$

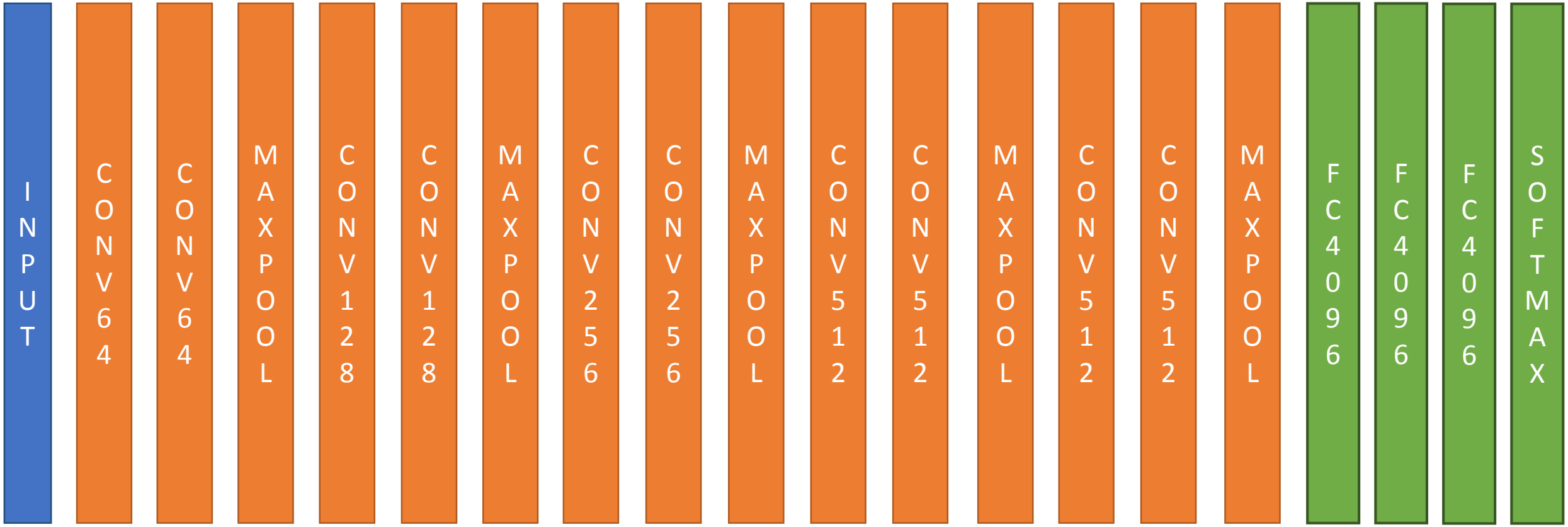
CNN: basic structure



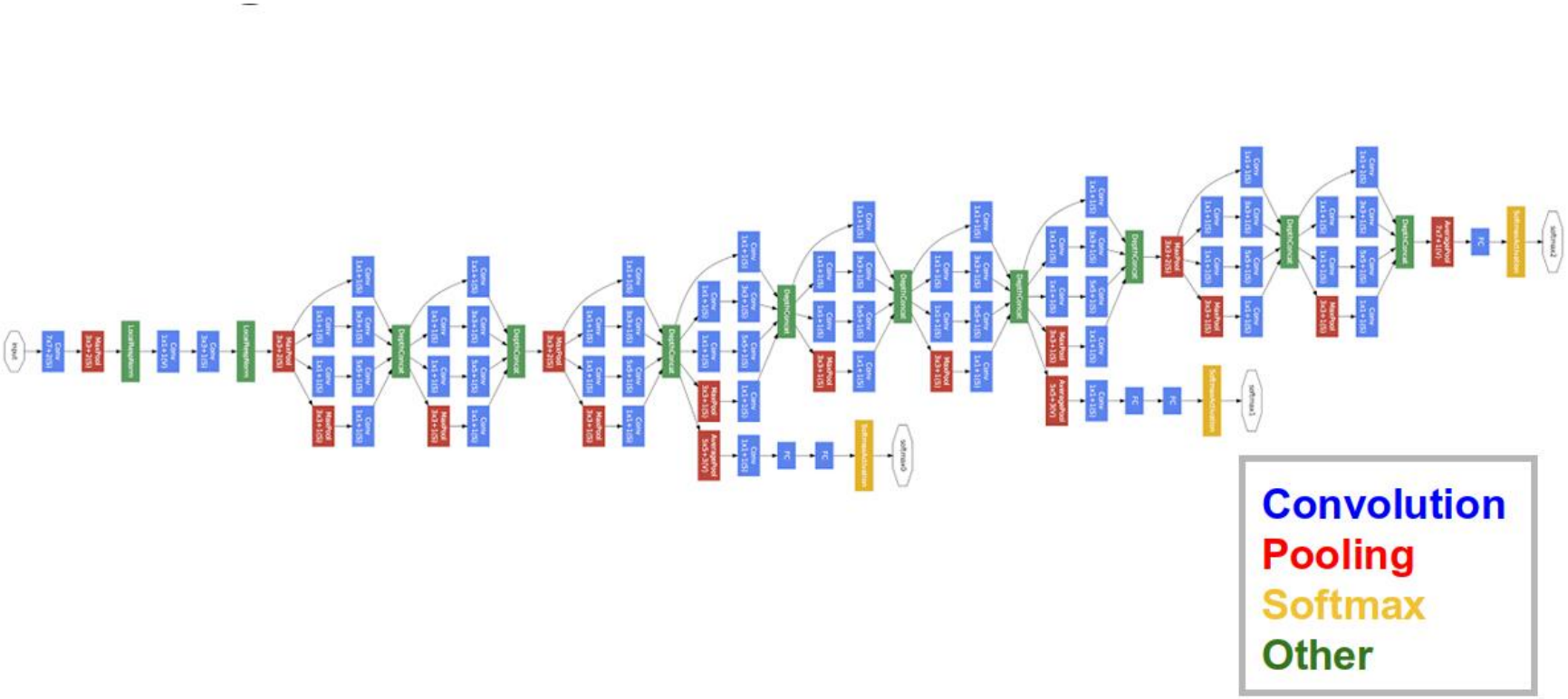
CNN: AlexNet 2012



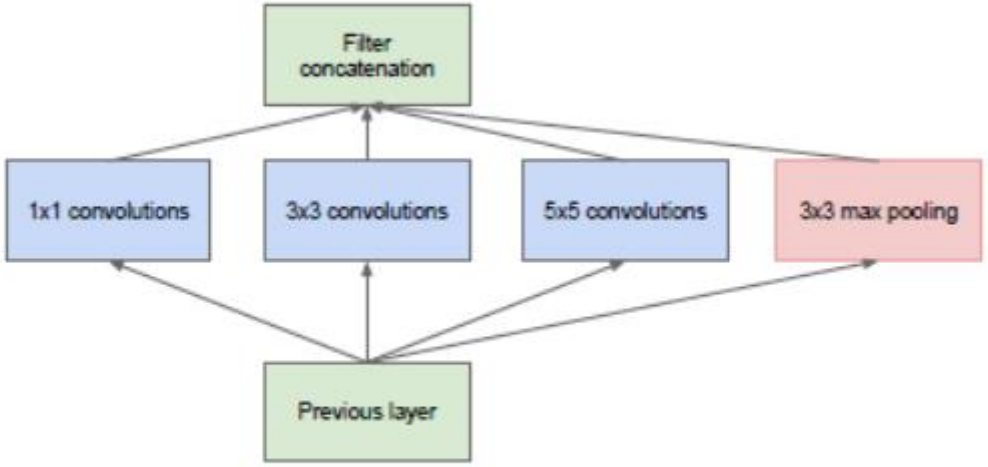
CNN: VGG Oxford 2014



CNN: GoogleNet 2014

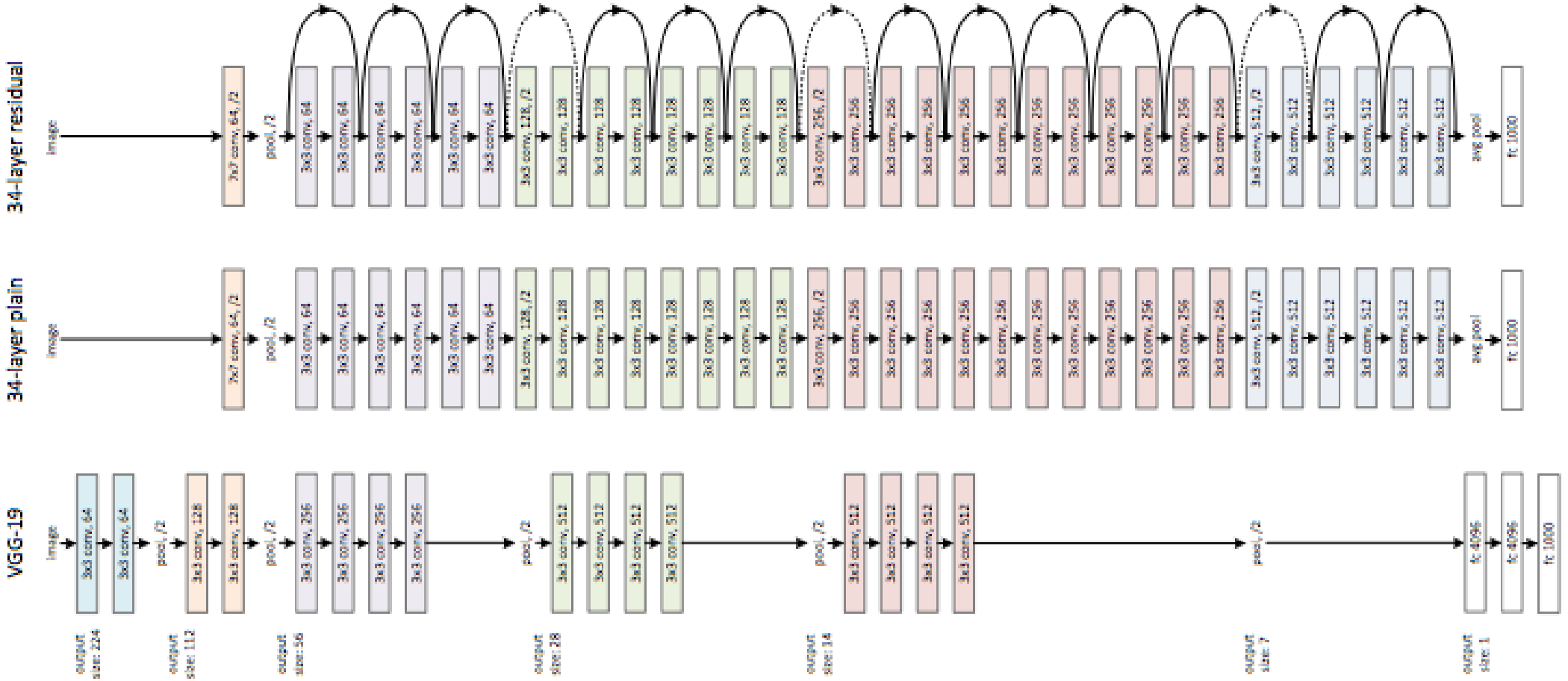


CNN: GoogleNet 2014



(a) Inception module, naïve version

CNN: ResNet 2015



CNN: ResNet 2015

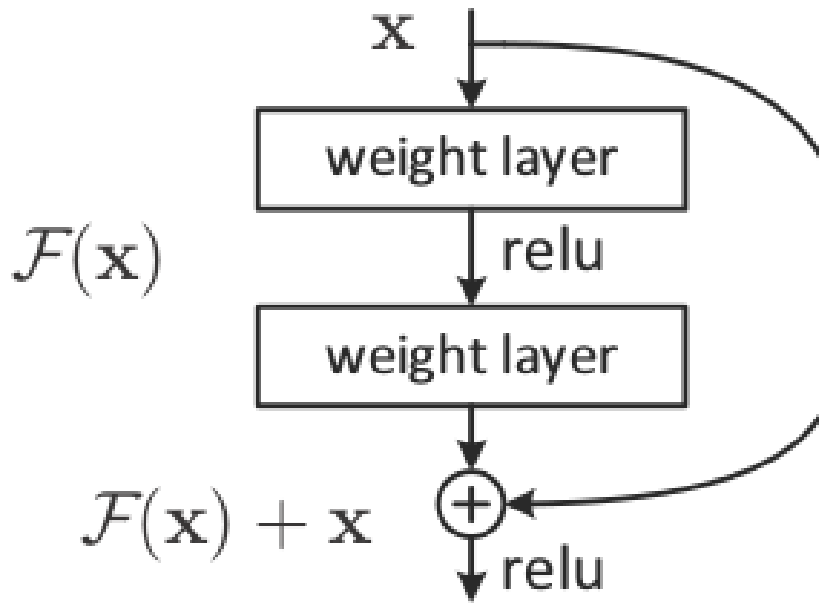
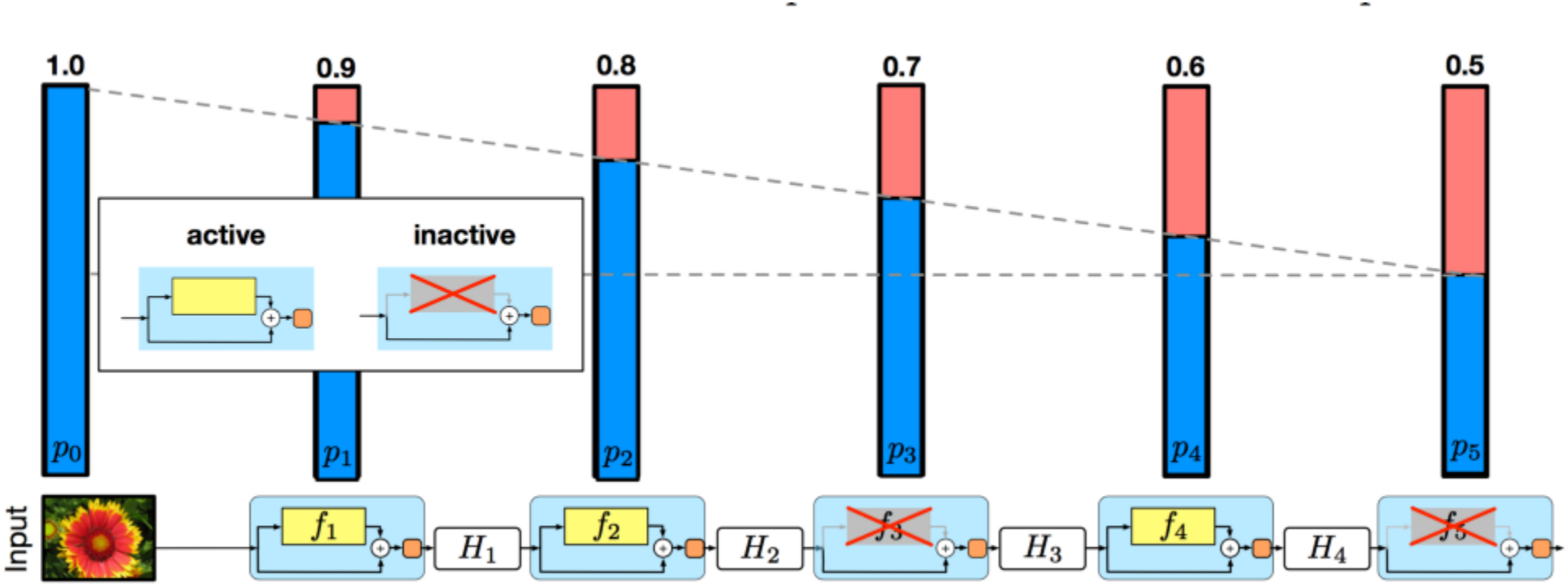
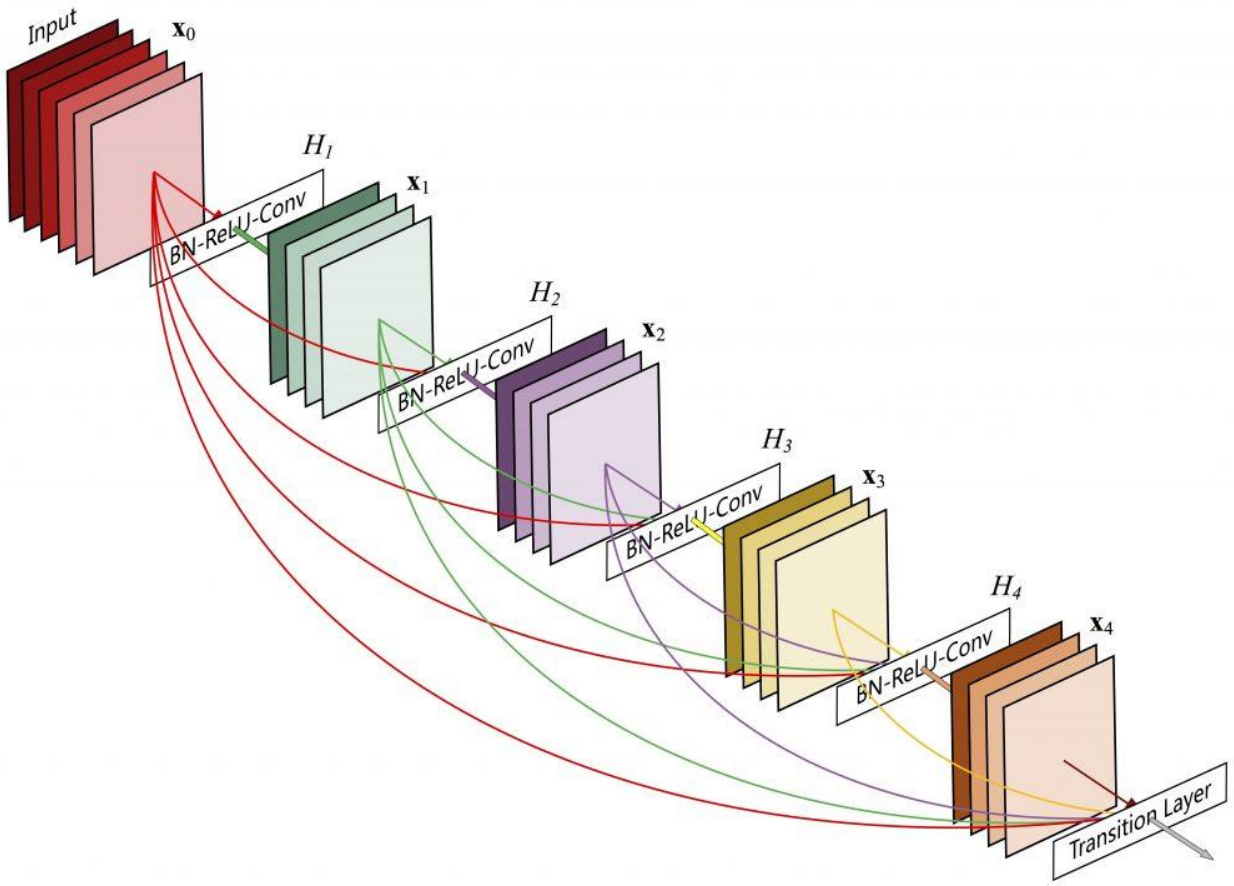


Figure 2. Residual learning: a building block.

CNN: Stochastic depth 2016



CNN: DenseNet 2016



CNN: Recent advances

- Batch normalization
- Different convolution types (dilated etc)
- Knowledge transfer
- Multi-task learning
- Convolution as FFT
- Several outputs
- Weights pruning / quantization

CNN: applications

- Classification (any object)
- Localization (any object)
- Segmentation (any object)
- Identification (face for example)
- Pose estimation
- Image captioning