Neural Netoworks for Vision

EE 782 Advanved Topics in Machine Learning

July-Nov 2025

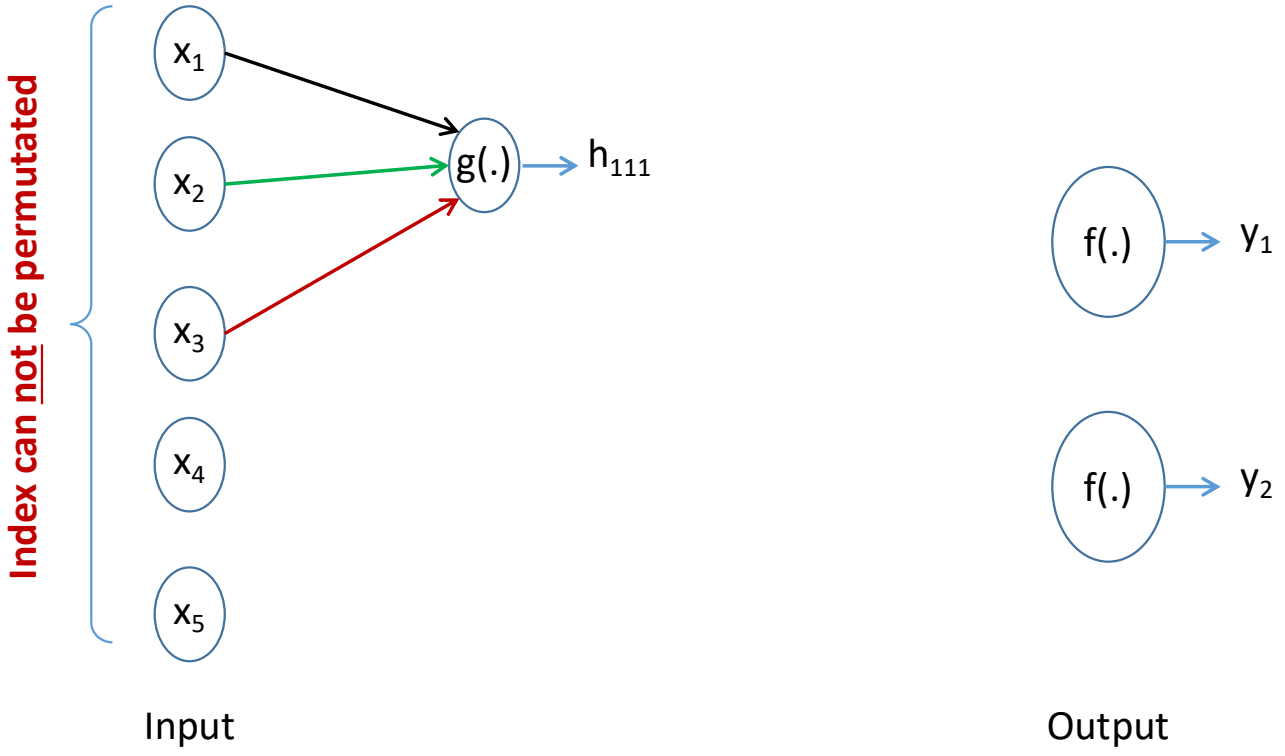
Amit Sethi, EE, IITB

Contact: asethi@iitb, 3528, 7483

Learning outcomes for the lecture

- List benefits of convolution
- Identify input types suited for convolution
- List benefits of pooling
- Identify input types not suited for convolution
- Write backprop through conv and pool

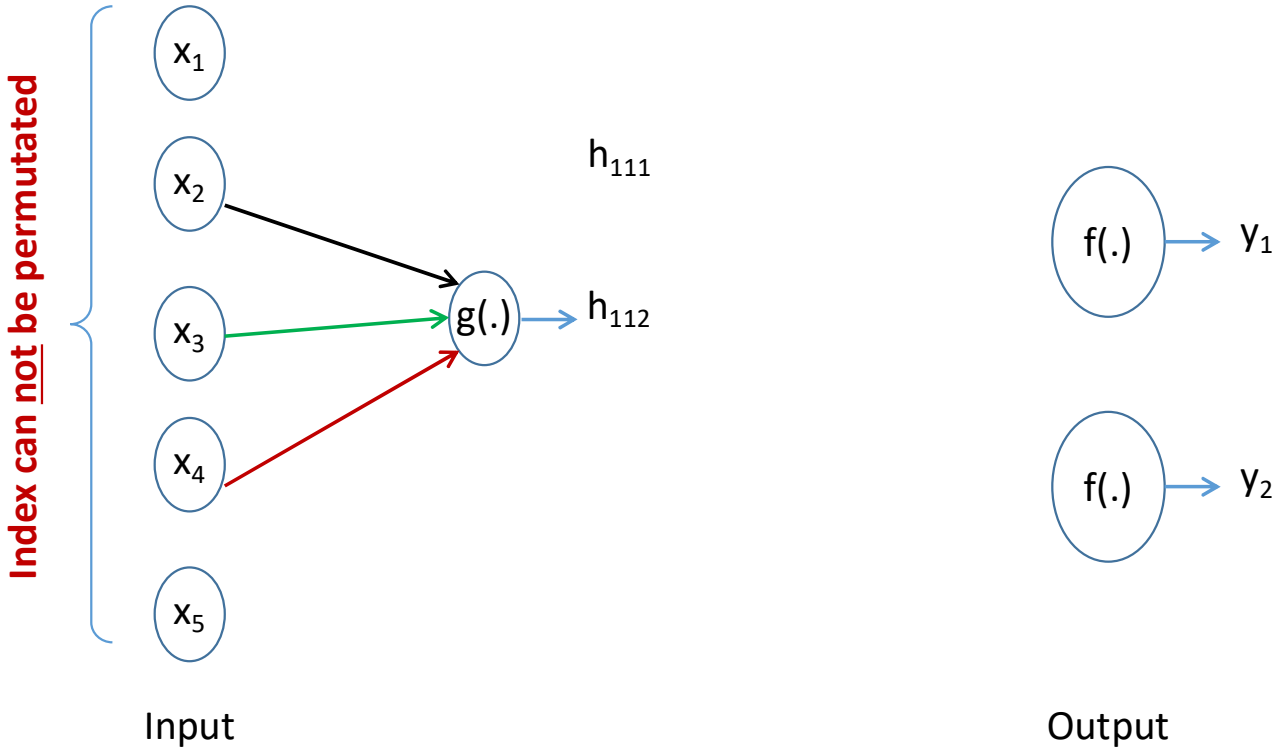
Convolutional layers



Idea: (1) Features are local
(2) Their presence/absence is stationary

Concept by Yann LeCun et al.

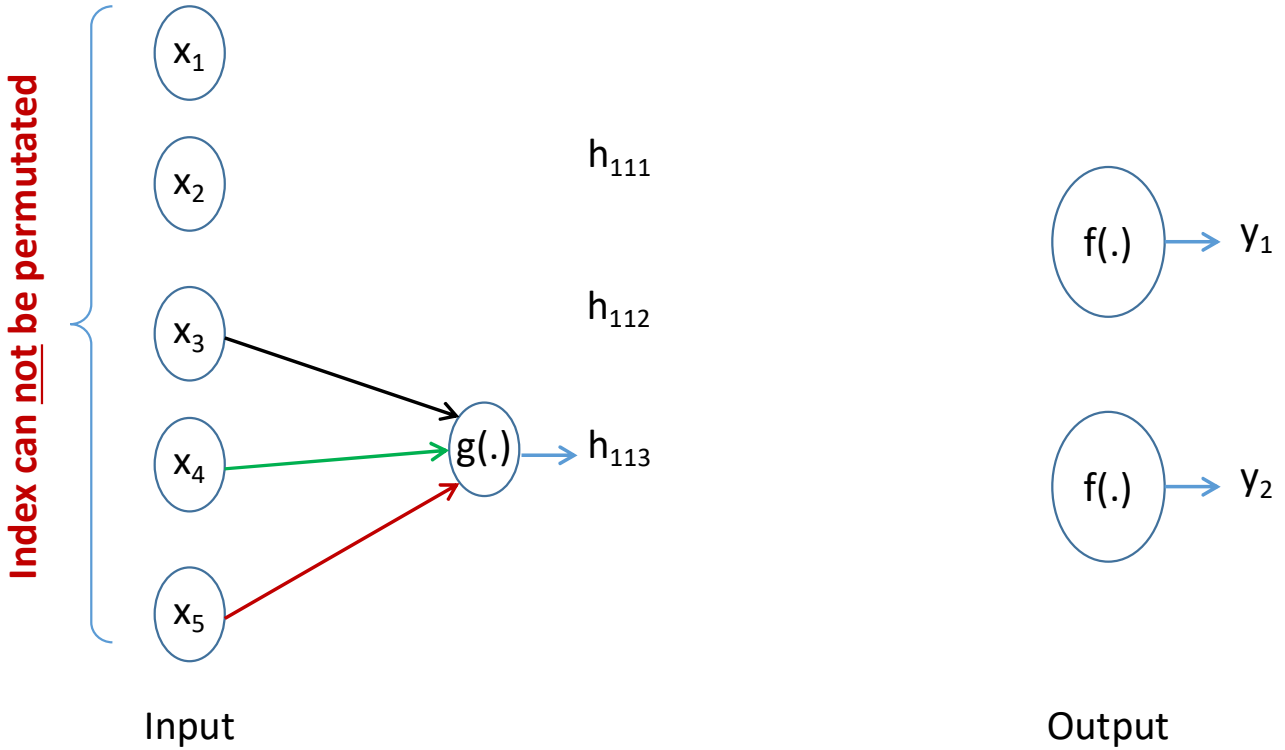
Convolutional layers



Idea: (1) Features are local
(2) Their presence/absence is stationary

Concept by Yann LeCun

Convolutional layers

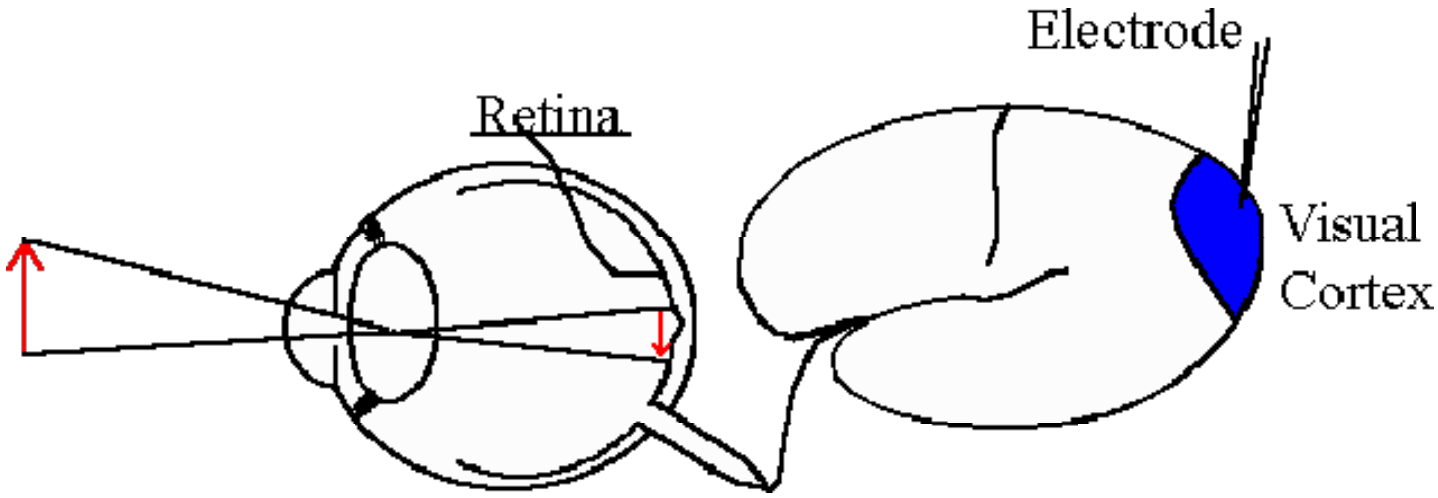


Idea: (1) Features are local, (2) Their presence/absence is stationary
(3) GPU implementation for inexpensive super-computing

LeNet, AlexNet

Receptive fields of neurons

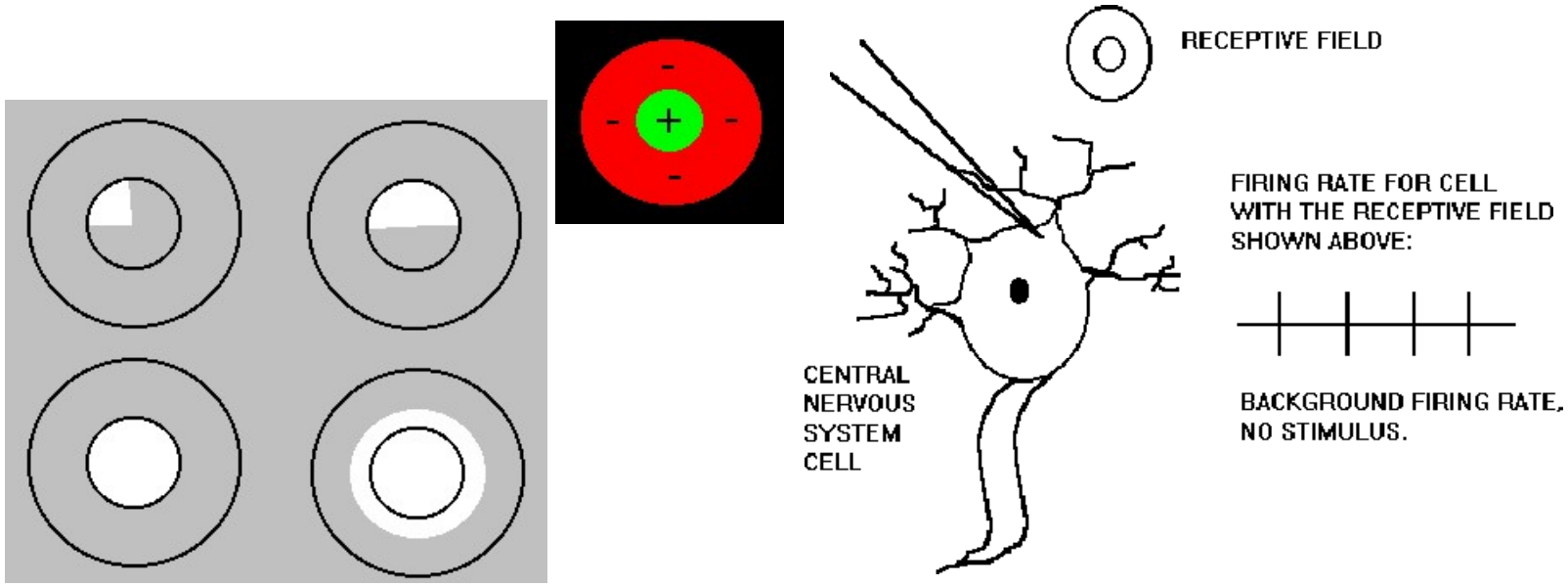
- Levine and Shefner (1991) define a receptive field as an "area in which stimulation leads to response of a particular sensory neuron" (p. 671).



Source: http://psych.hanover.edu/Krantz/receptive/

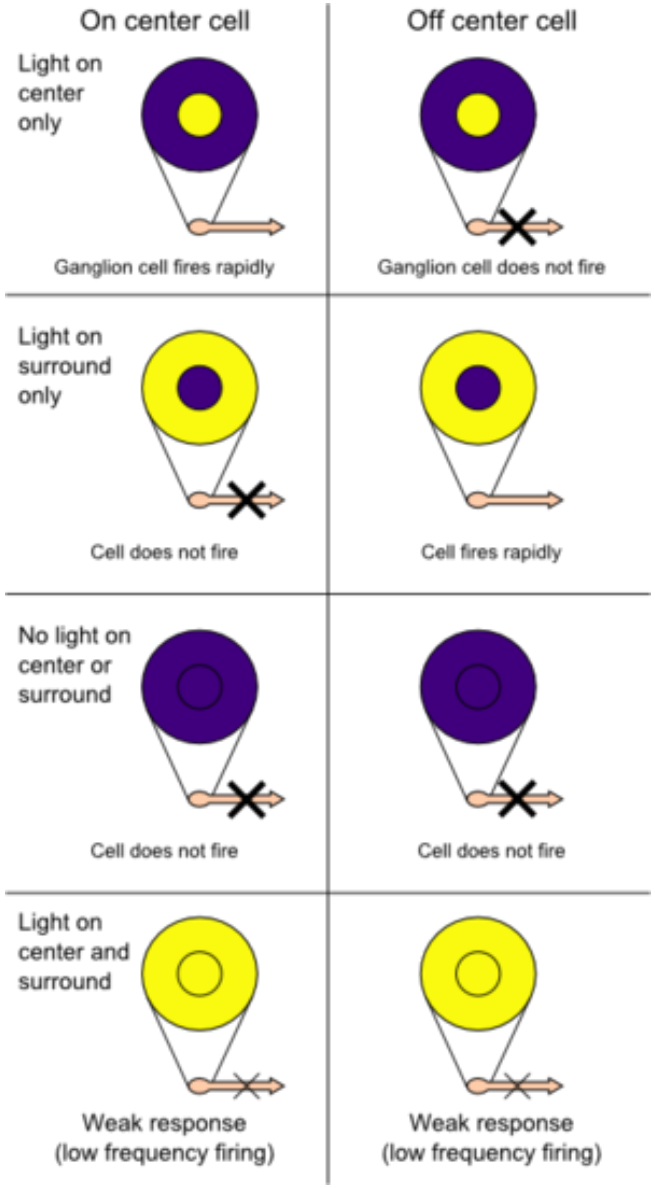
The concept of the best stimulus

- Depending on excitatory and inhibitory connections, there is an optimal stimulus that falls only in the excitatory region
- On-center retinal ganglion cell example shown here



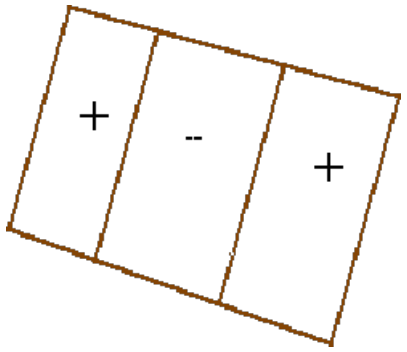
Source: http://psych.hanover.edu/Krantz/receptive/

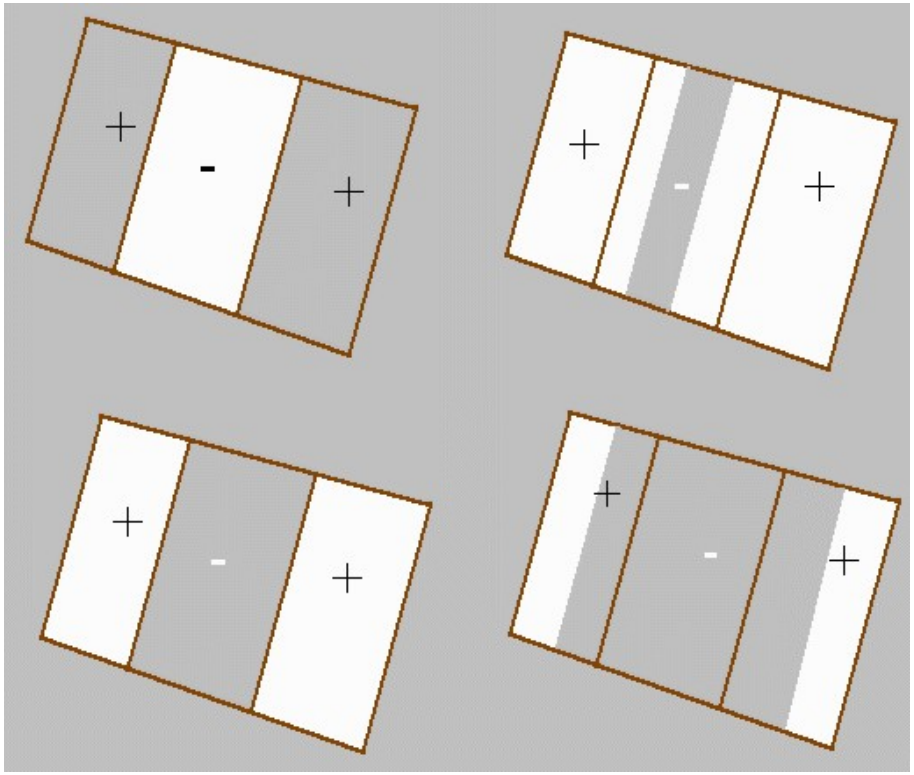
On-center vs. off-center



Source: https://en.wikipedia.org/wiki/Receptive_field

Bar detection example





Source: http://psych.hanover.edu/Krantz/receptive/

Gabor filters model simple cell in visual cortex

Complex

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right)$$

Real

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

Imaginary

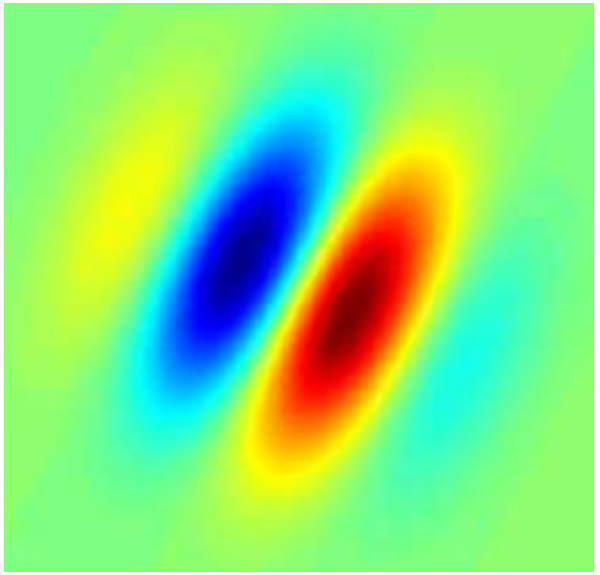
$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \sin\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

where

$$x' = x\cos\theta + y\sin\theta$$

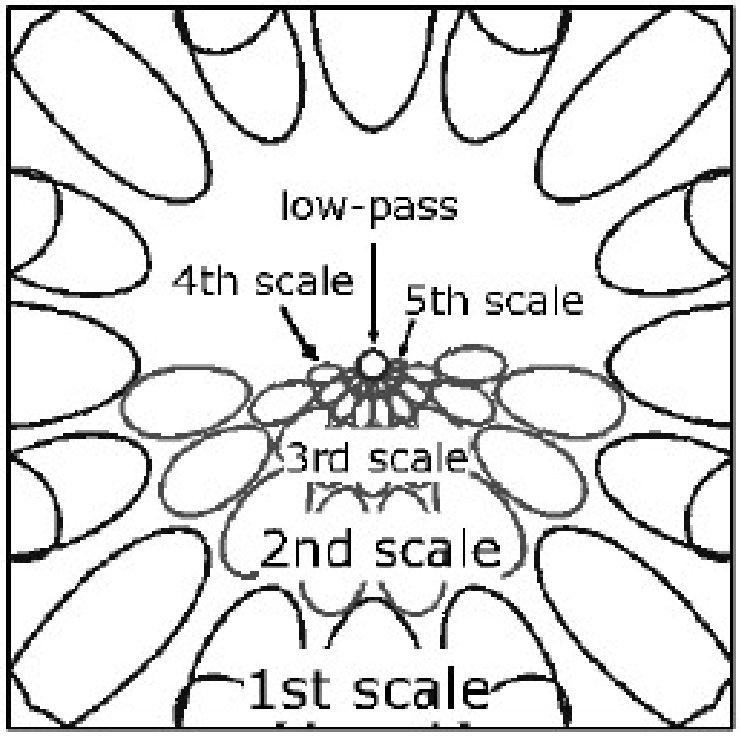
and

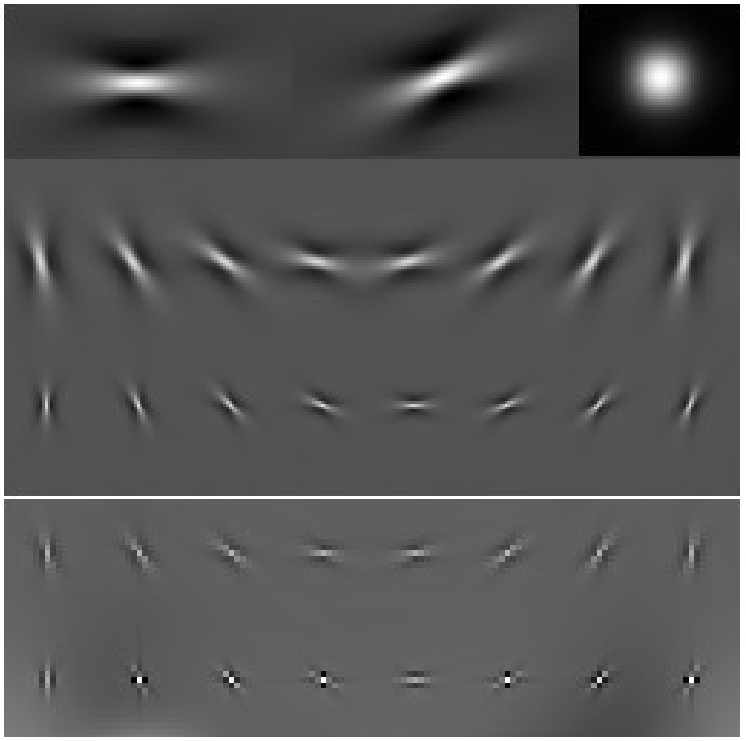
$$y' = -x\sin\theta + y\cos\theta$$



Source: https://en.wikipedia.org/wiki/Gabor_filter

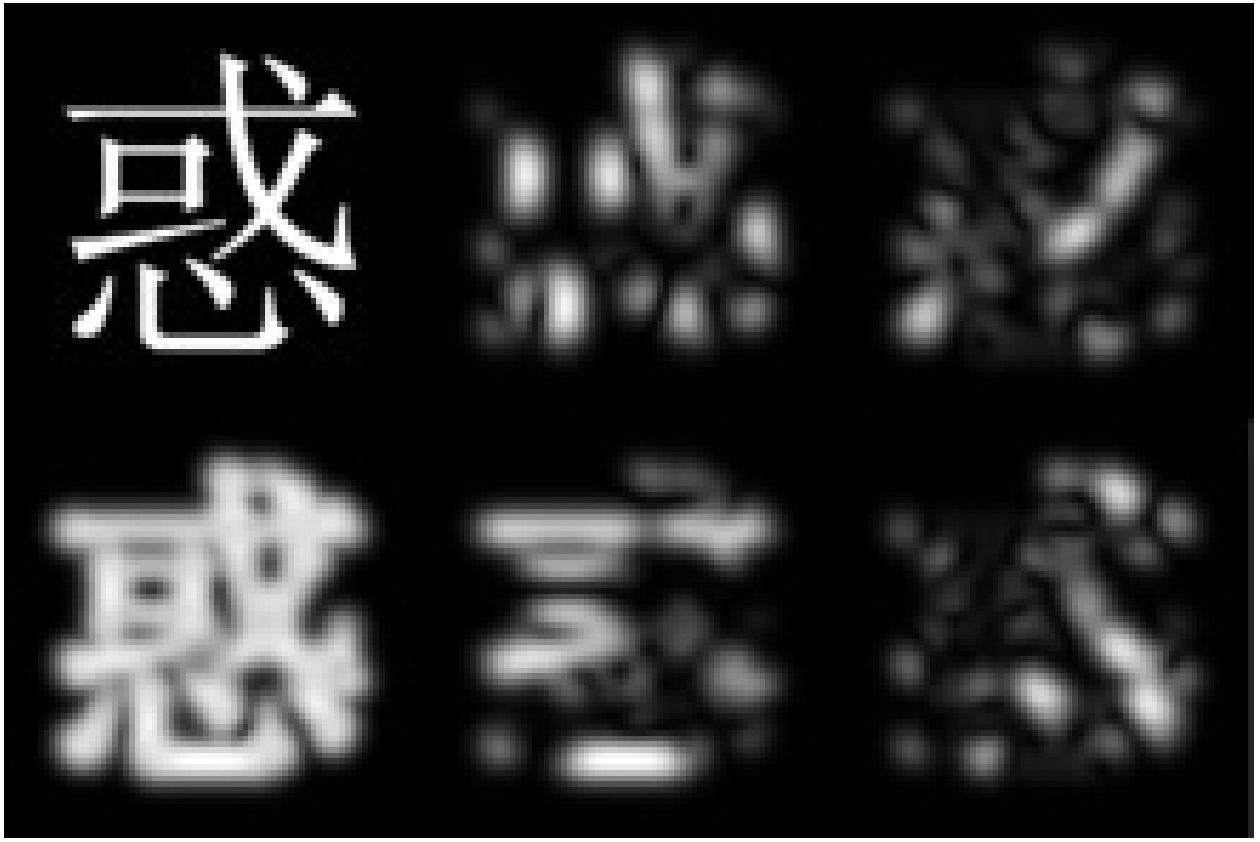
Modeling oriented edges using Gabor





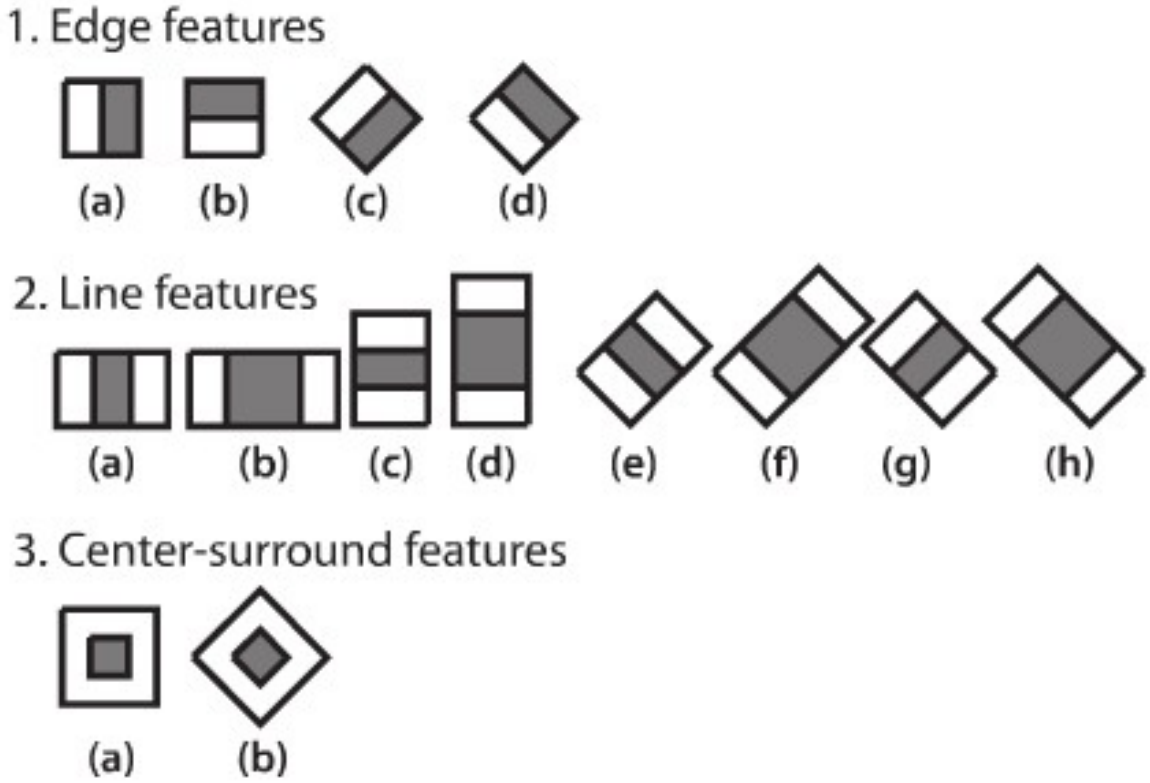
Source: https://en.wikipedia.org/wiki/Gabor_filter

Feature maps using Gabor filters



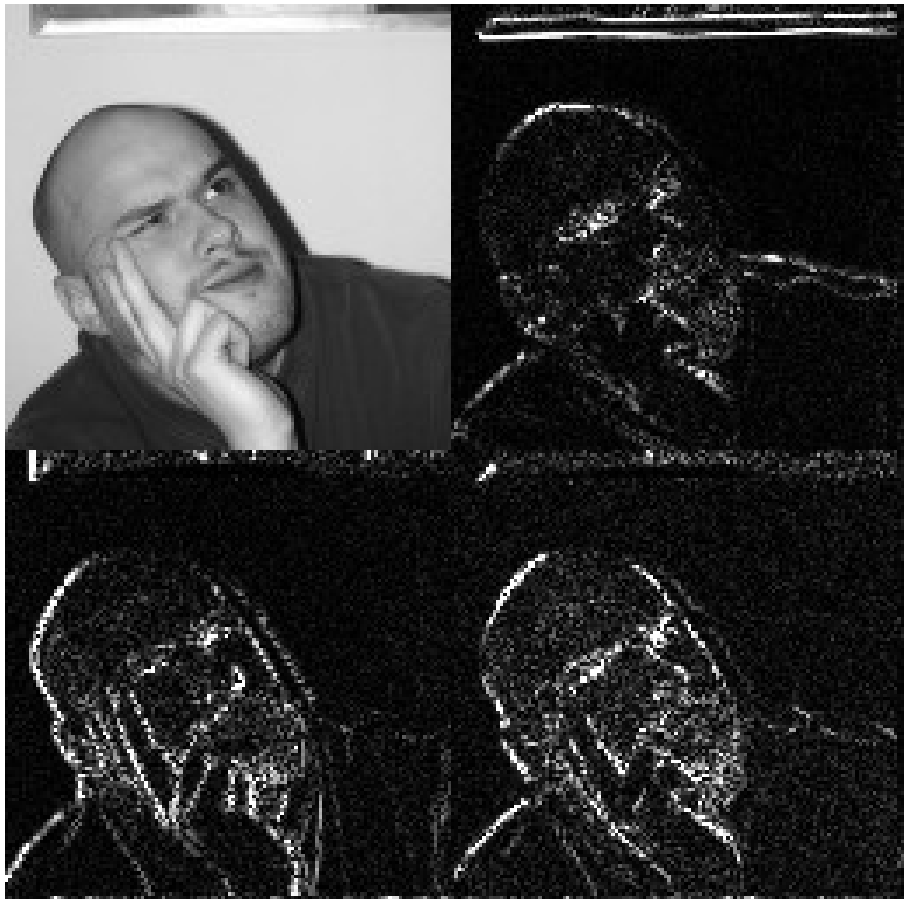
Source: https://en.wikipedia.org/wiki/Gabor_filter

Haar filters



Source: http://www.cosy.sbg.ac.at/~hegenbart/

More feature maps



Source: http://www.cosy.sbg.ac.at/~hegenbart/

Convolution

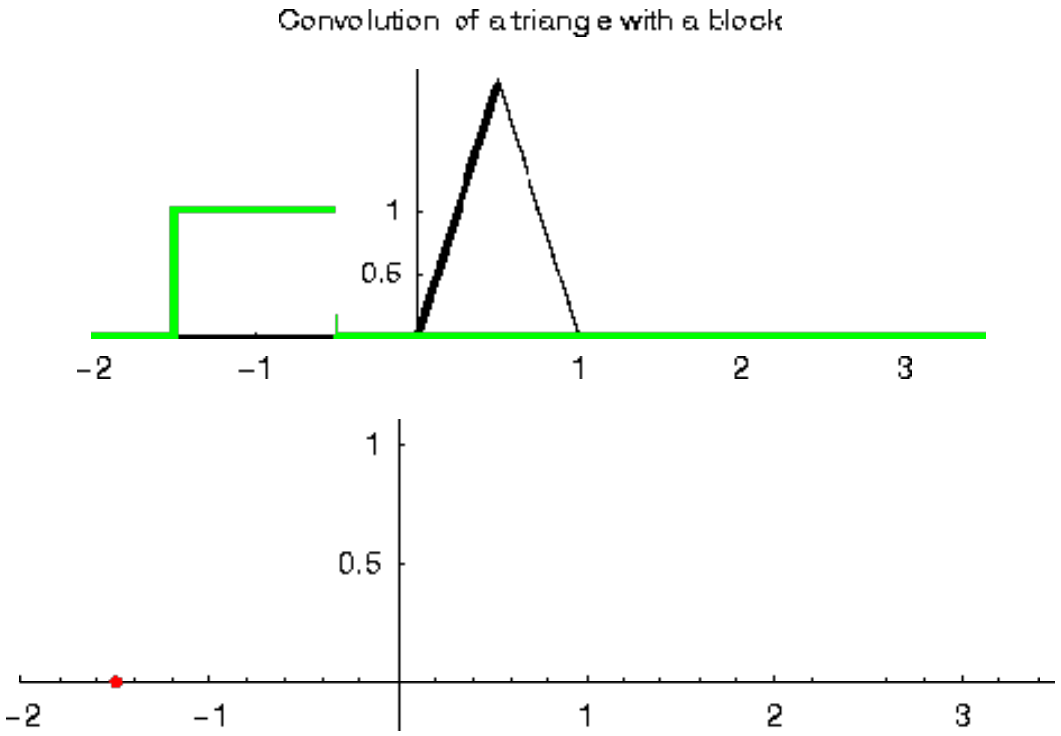
- Classical definitions

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

$$(f * g)[n] = \sum_{x=-\infty}^{\infty} f[n-x]g[x]$$

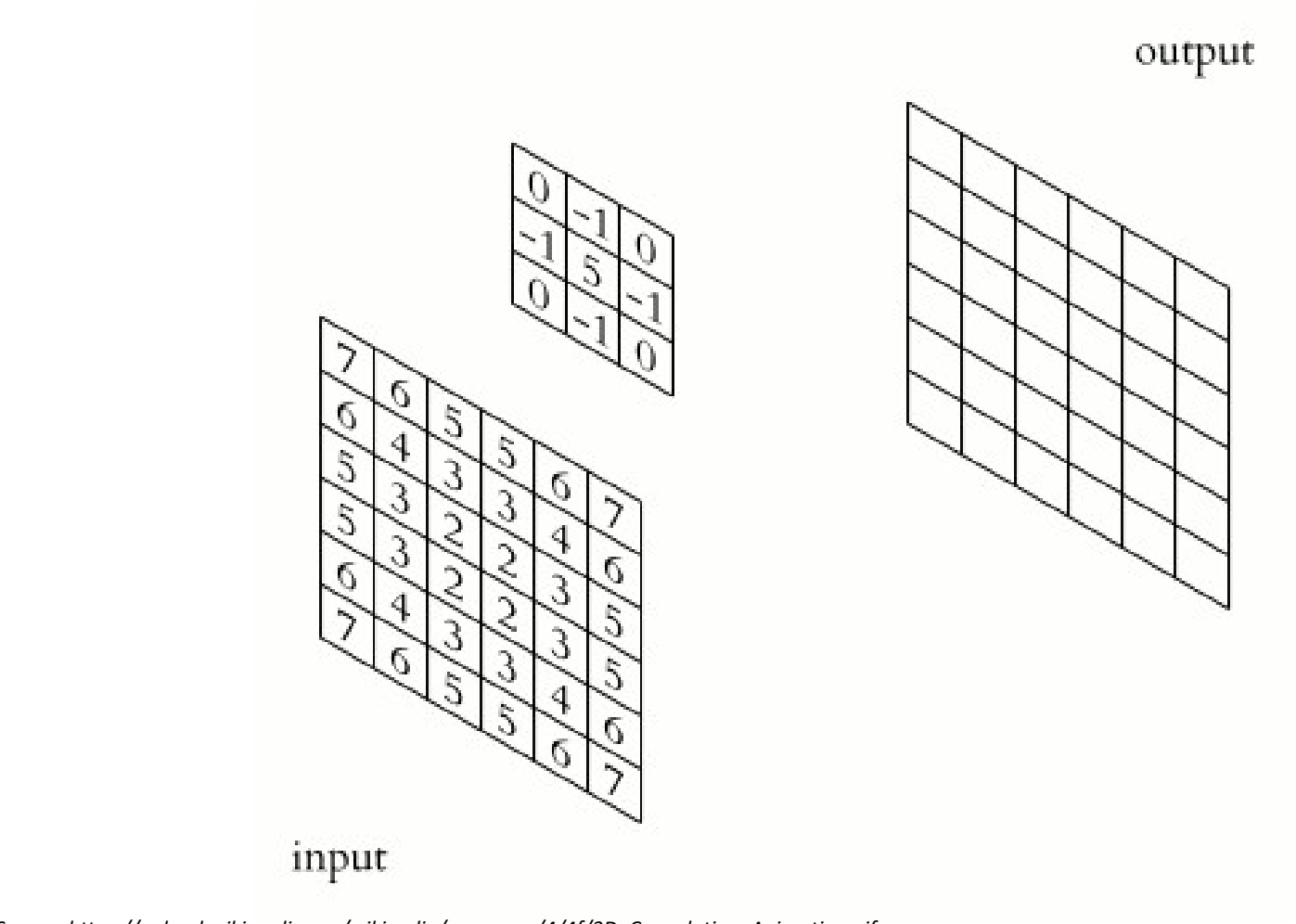
- Or, one can take cross-correlation between $f[n]$ and $g[-n]$
- In 2-D, it would be $\sum_{a=-\infty}^{\infty}\sum_{b=-\infty}^{\infty} f[n,m]g[n+x,m+y]$
- Fast implementation for multiple PUs

Convolution animation



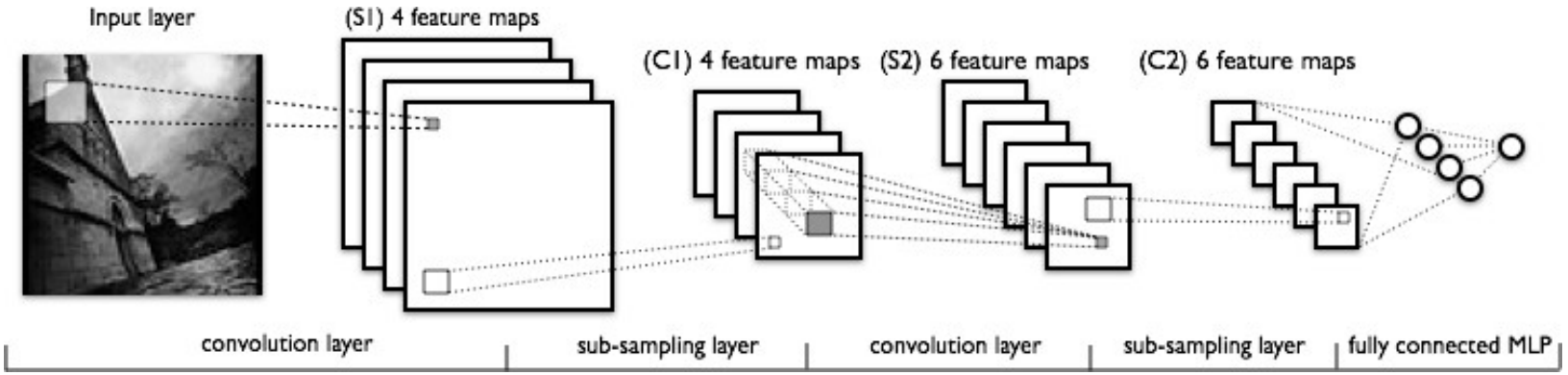
Source: http://bmia.bmt.tue.nl/education/courses/fev/course/notebooks/triangleblockconvolution.gif

Convolution in 2-D (sharpening filter)



Source: https://upload.wikimedia.org/wikipedia/commons/4/4f/3D_Convolution_Animation.gif

Let the network learn conv kernels



Original image source unknown

Number of weights with and without conv.

- Assume that we want to extract 25 features per pixel
- Fully connected layer:
  - Input 32x32x3
  - Hidden 28x28x25
  - Weights 32x32x3 x 28x28x25 = 60,211,200
- With convolutions (weight sharing):
  - Input 32x32x3
  - Hidden 28x28x25
  - Weights 5x5x3 x 25 = 1,875

How will backpropagation work?

- Backpropagation will treat each input patch (not image) as a sample!

Feature maps

- Convolutional layer:
  - Input → A (set of) layer(s)
    - Convolutional filter(s)
    - Bias(es)
    - Nonlinear squashing
  - Output → Another layer(s); AKA: Feature maps
- A map of where each feature was detected
- A shift in input => A shift in feature map

- Is it important to know where exactly the feature was detected?
- Notion of invariances: translation, scaling, rotation, contrast

Pooling is subsampling

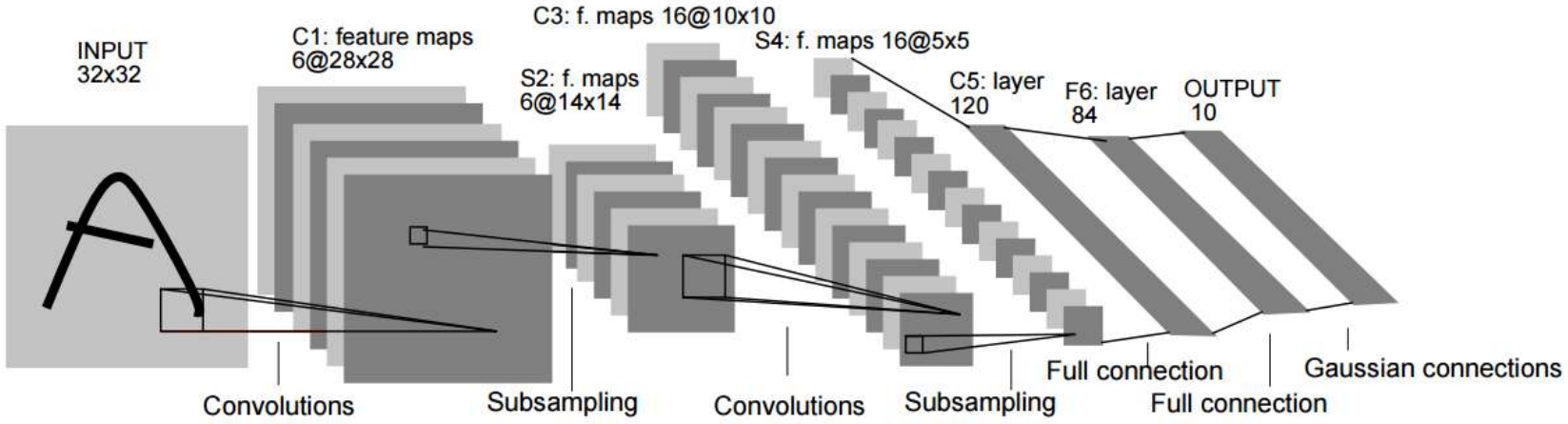


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

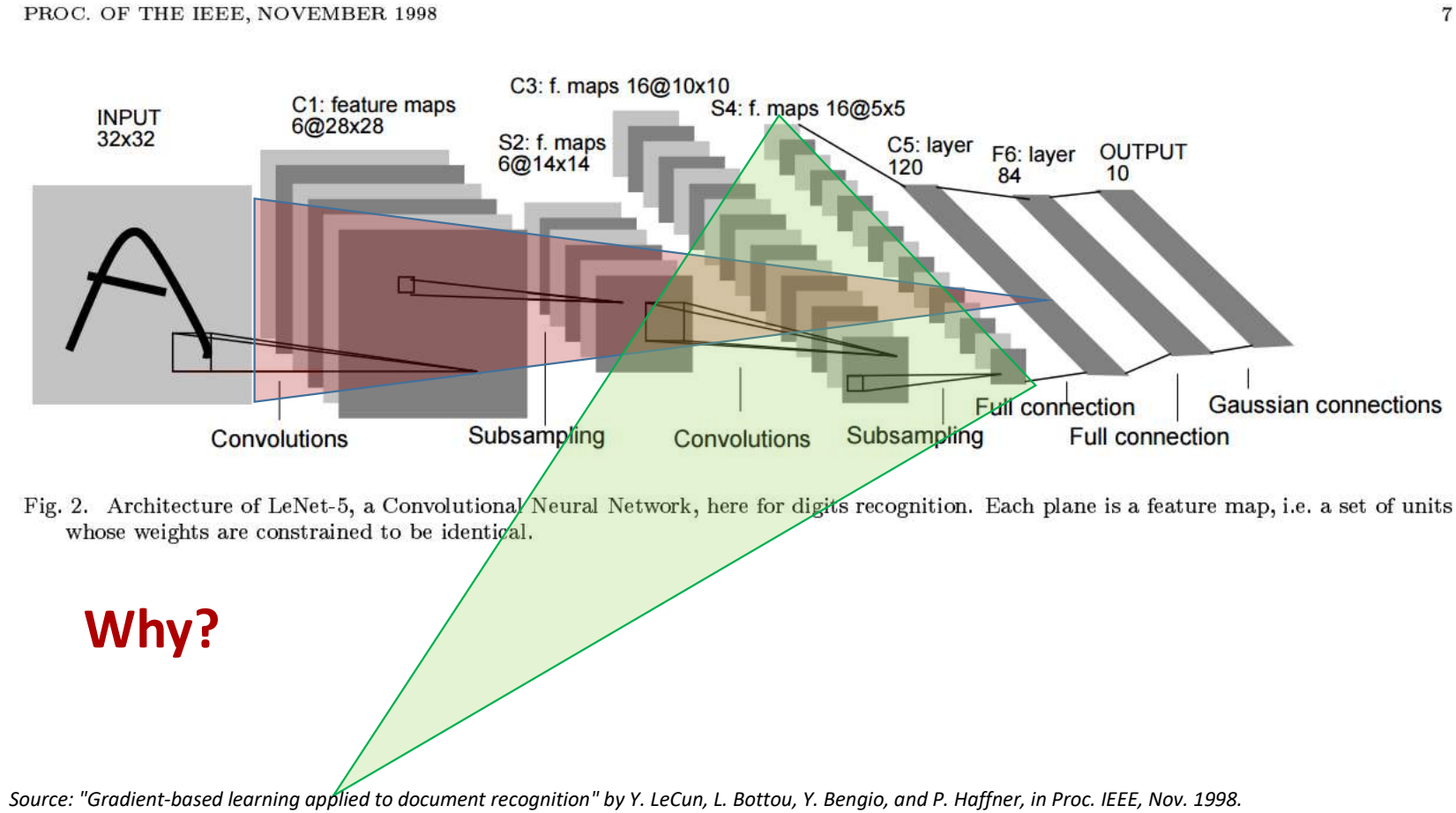
Source: "Gradient-based learning applied to document recognition" by Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, in Proc. IEEE, Nov. 1998.

Types of pooling

- Two types of popular pooling methods
  - Average
  - Max

- How do these differ?
- How do gradient computations differ?

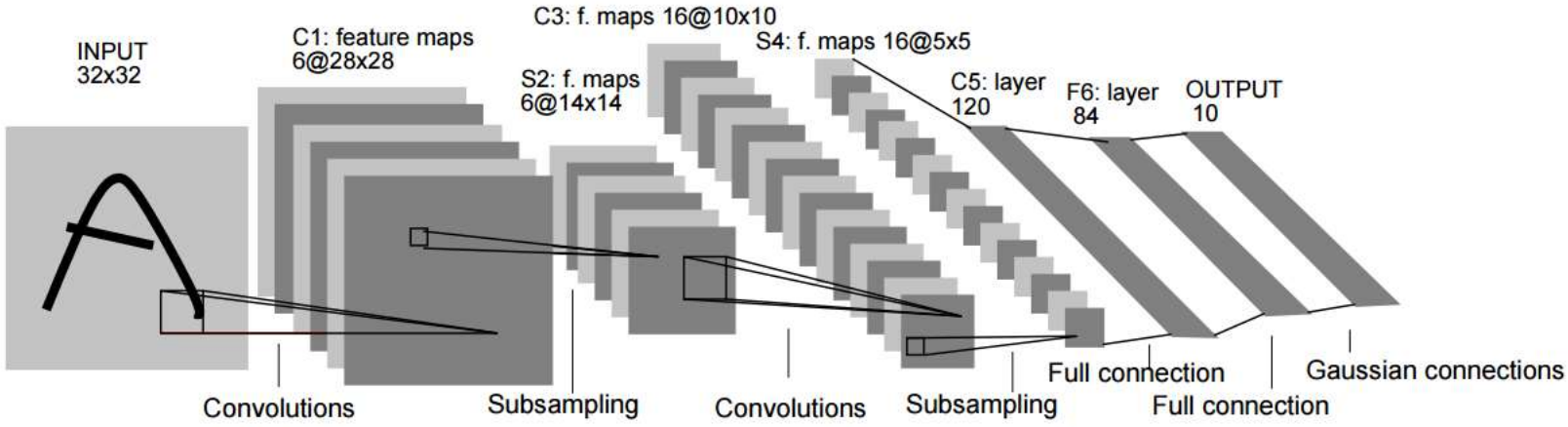
A bi-pyramid approach: Map size decreases, but number of maps increases



Why?

Source: "Gradient-based learning applied to document recognition" by Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, in Proc. IEEE, Nov. 1998.

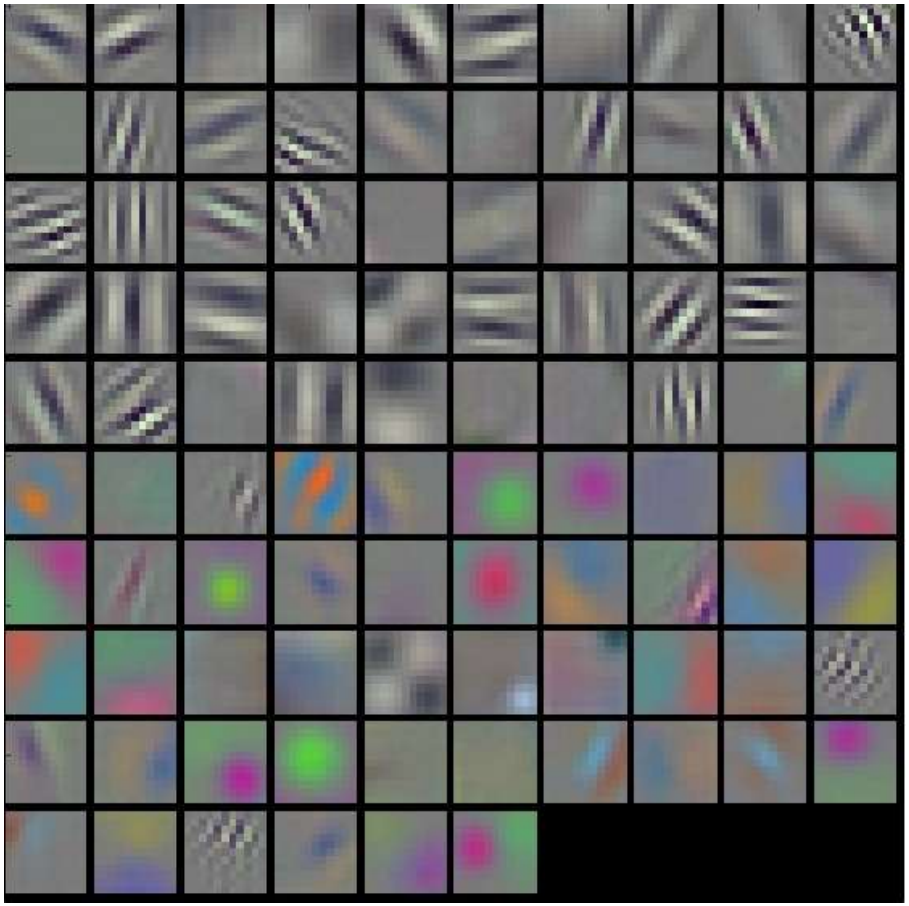
Fully connected layers



- Multi-layer non-linear decision making

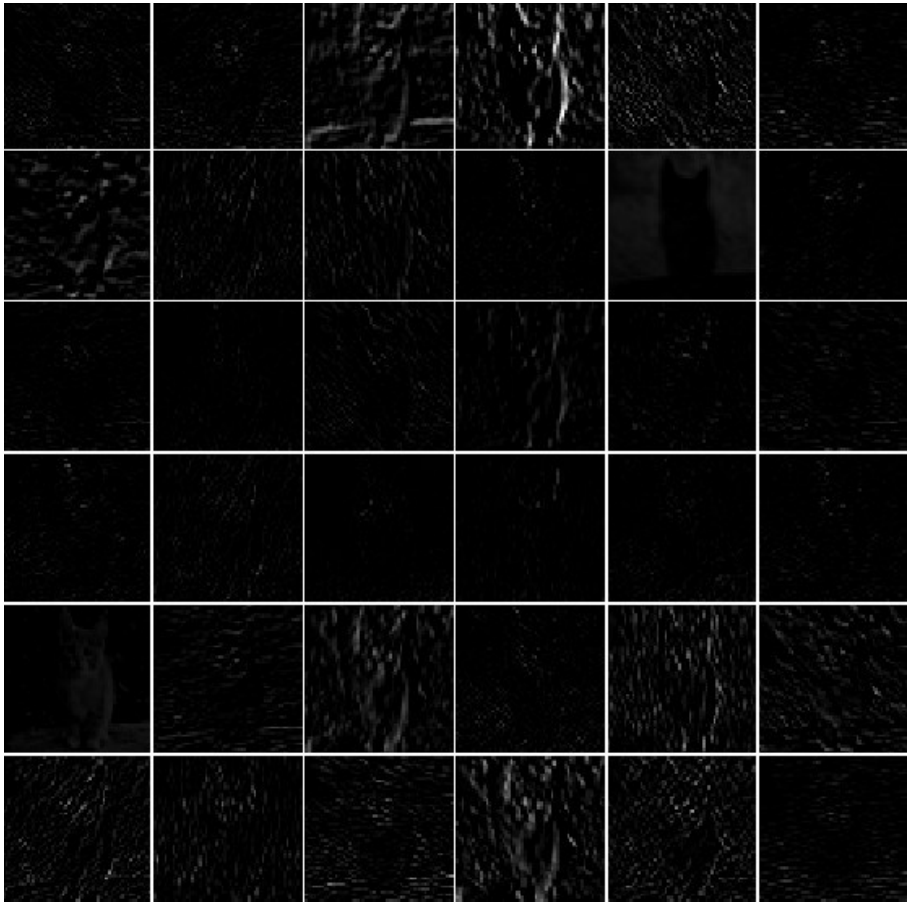
Source: "Gradient-based learning applied to document recognition" by Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, in Proc. IEEE, Nov. 1998.

Visualizing weights, conv layer 1



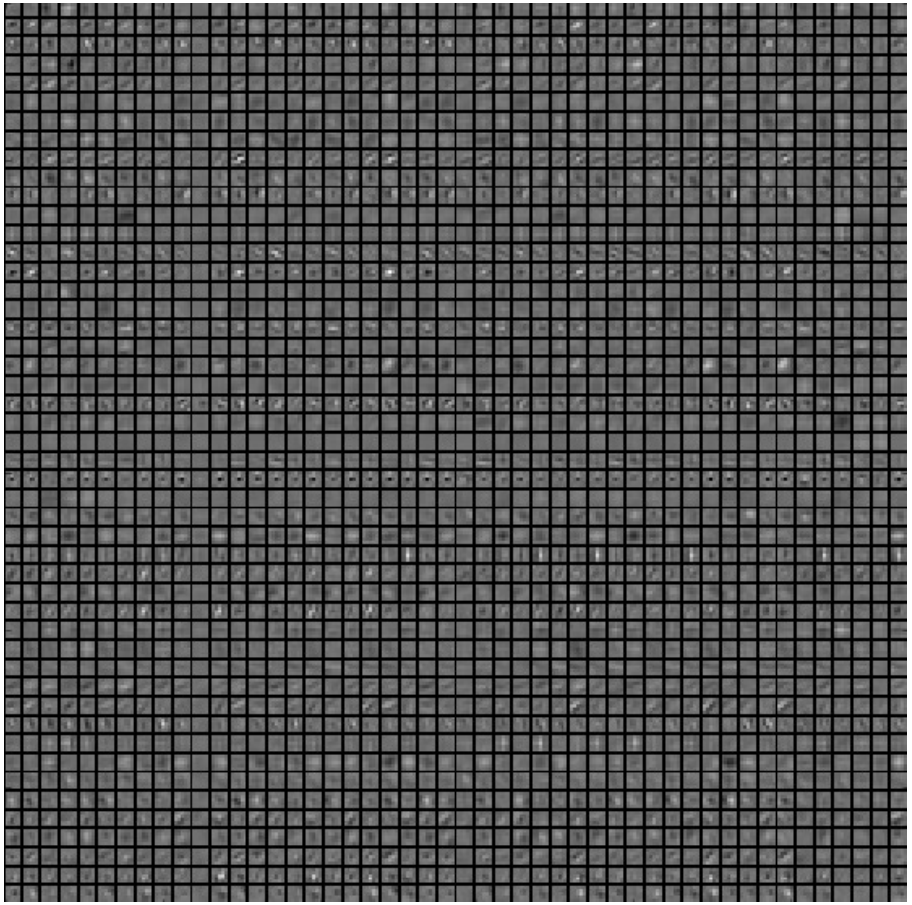
Source: http://cs231n.github.io/understanding-cnn/

Visualizing feature map, conv layer 1



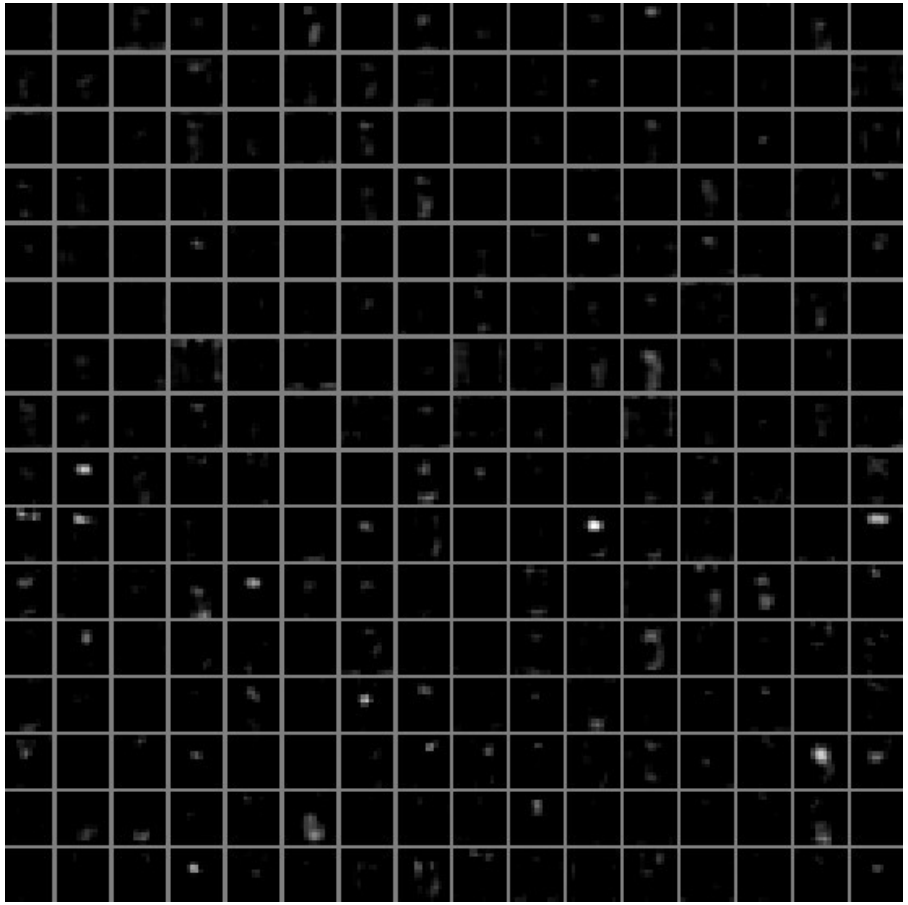
Source: http://cs231n.github.io/understanding-cnn/

Visualizing weights, conv layer 2



Source: http://cs231n.github.io/understanding-cnn/

Visualizing feature map, conv layer 2



Source: http://cs231n.github.io/understanding-cnn/

CNN for speech processing

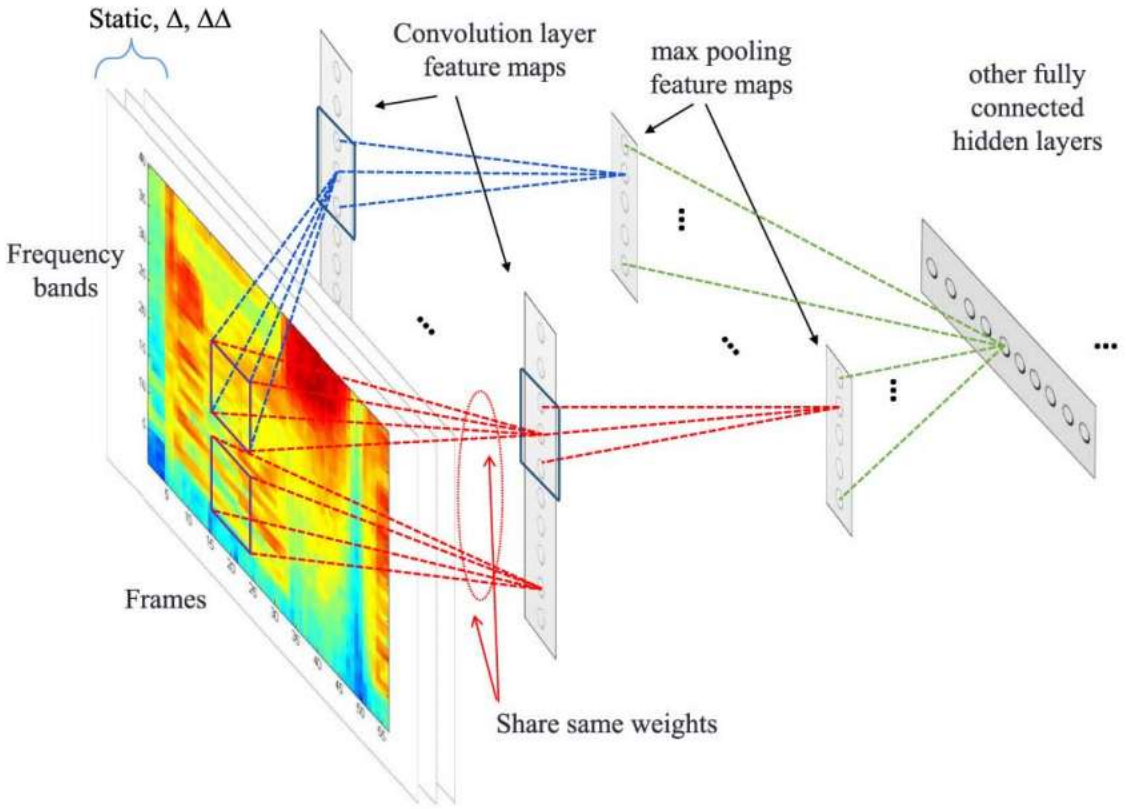
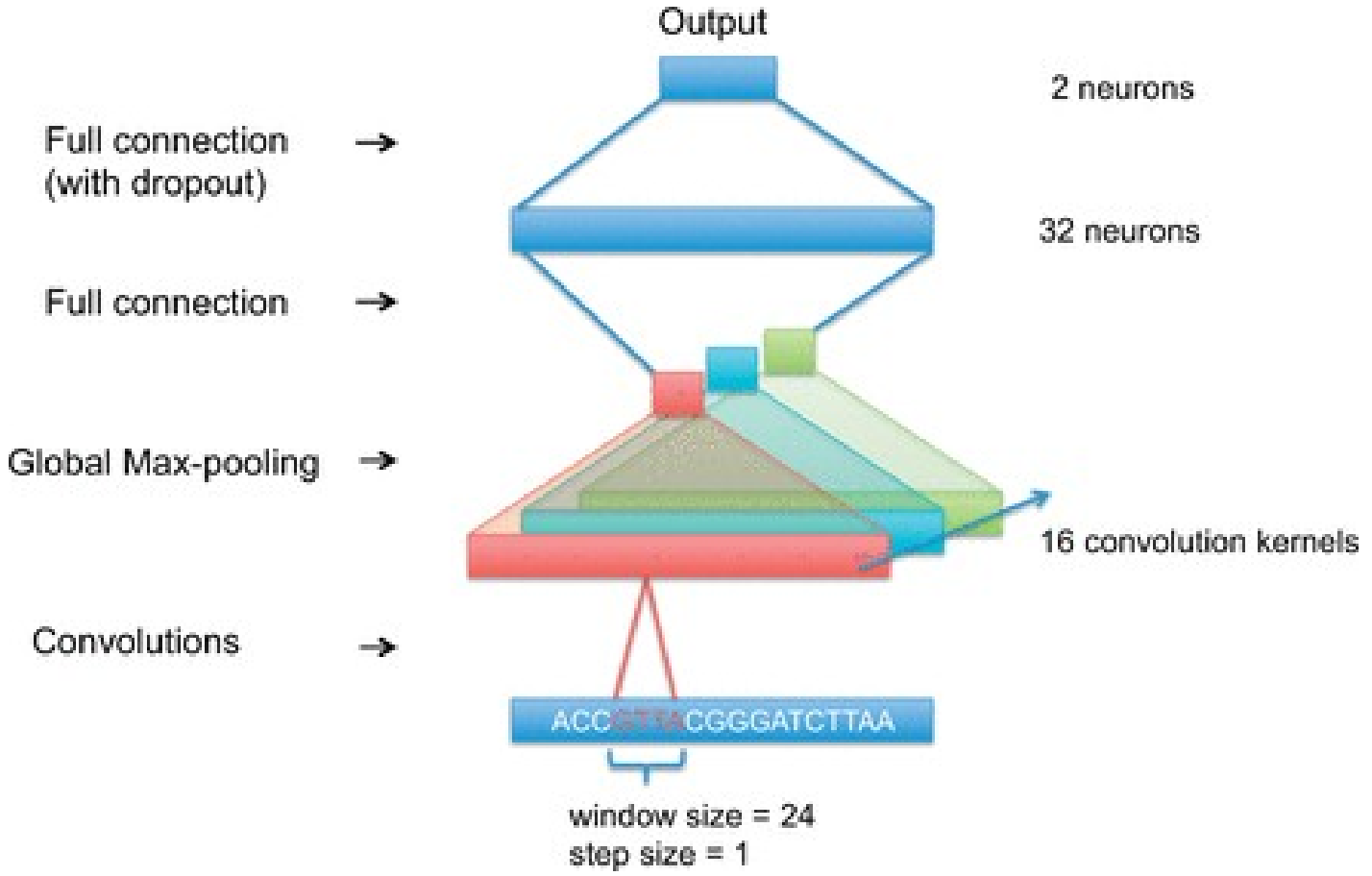


Fig. 3. An illustration of the regular CNN that uses so-called full weight sharing. Here, a 1-D convolution is applied along frequency bands.

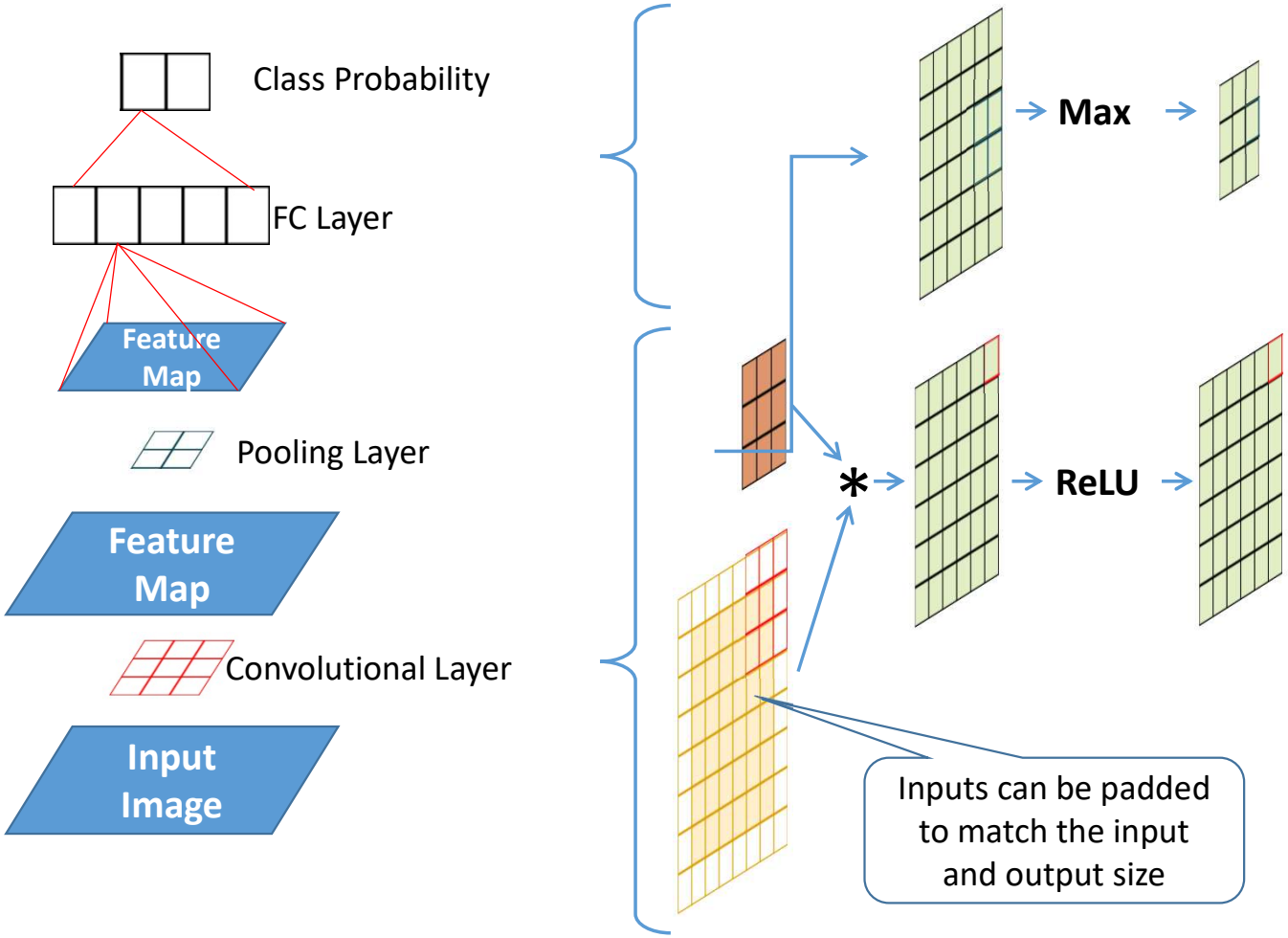
Source: "Convolutional neural networks for speech recognition" by Ossama Abdel-Hamid et al., in IEEE/ACM Trans. ASLP, Oct, 2014

CNN for DNA-protein binding



Source: "Convolutional neural network architectures for predicting DNA–protein binding" by Haoyang Zeng et al., Bioinformatics 2016, 32 (12)

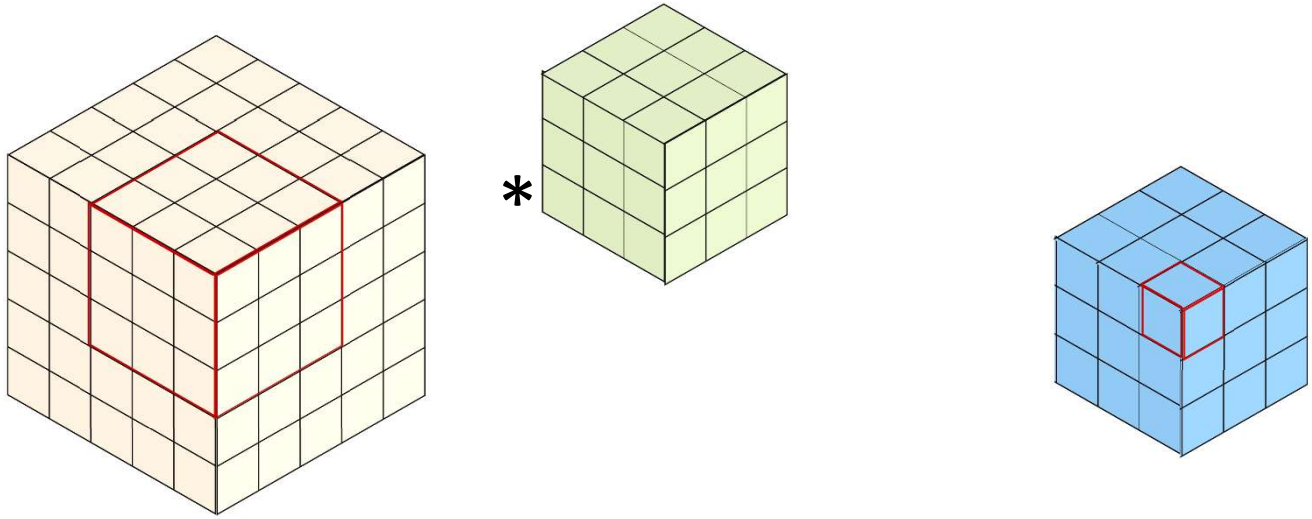
Convolution and pooling revisited



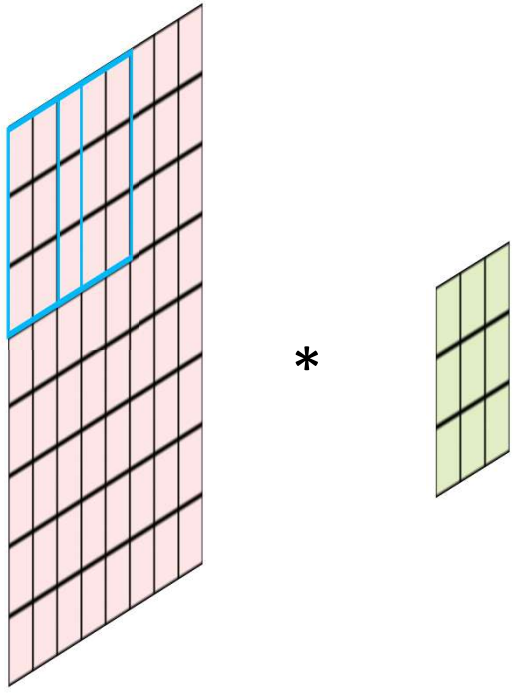
Variations of convolutional filter achieve various purposes

- N-D convolutions generalize over 2-D
- Stride variation leads to pooling
- Atrous (dilated) convolutions cover more area with less parameters
- Transposed convolution increases the feature map size
- Layer-wise convolutions reduce parameters
- 1x1 convolutions reduce feature maps
- Separable convolutions reduce parameters
- Network-in-network learns a nonlinear conv

Convolutions in 3-D



Convolutions with stride > 1



Atrous (dilated) convolutions can increase the receptive field without increasing the number of weights

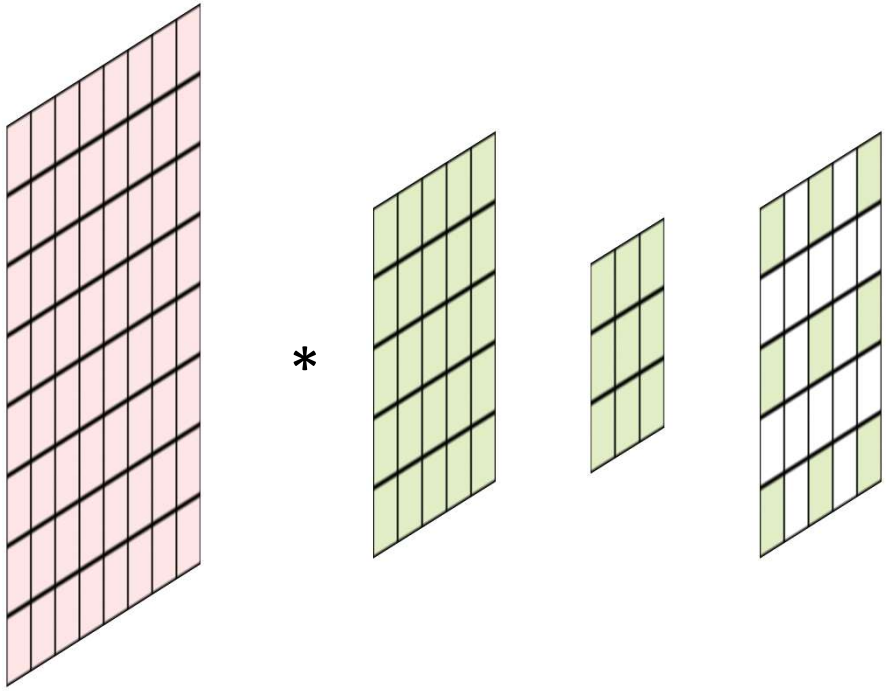
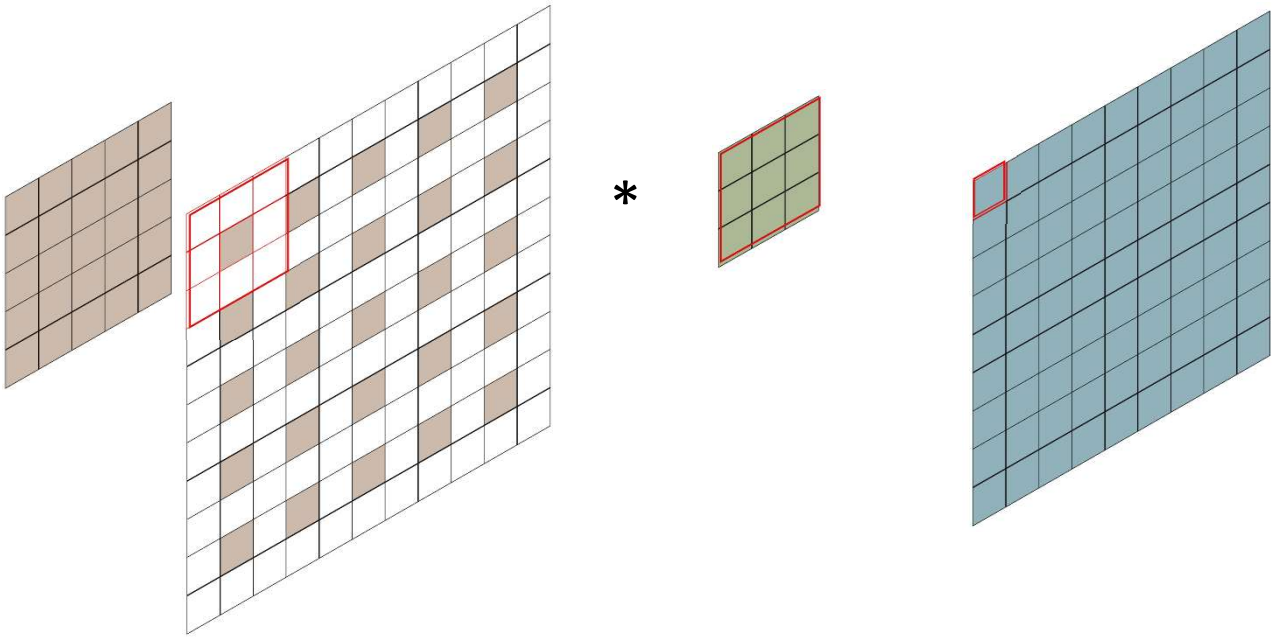
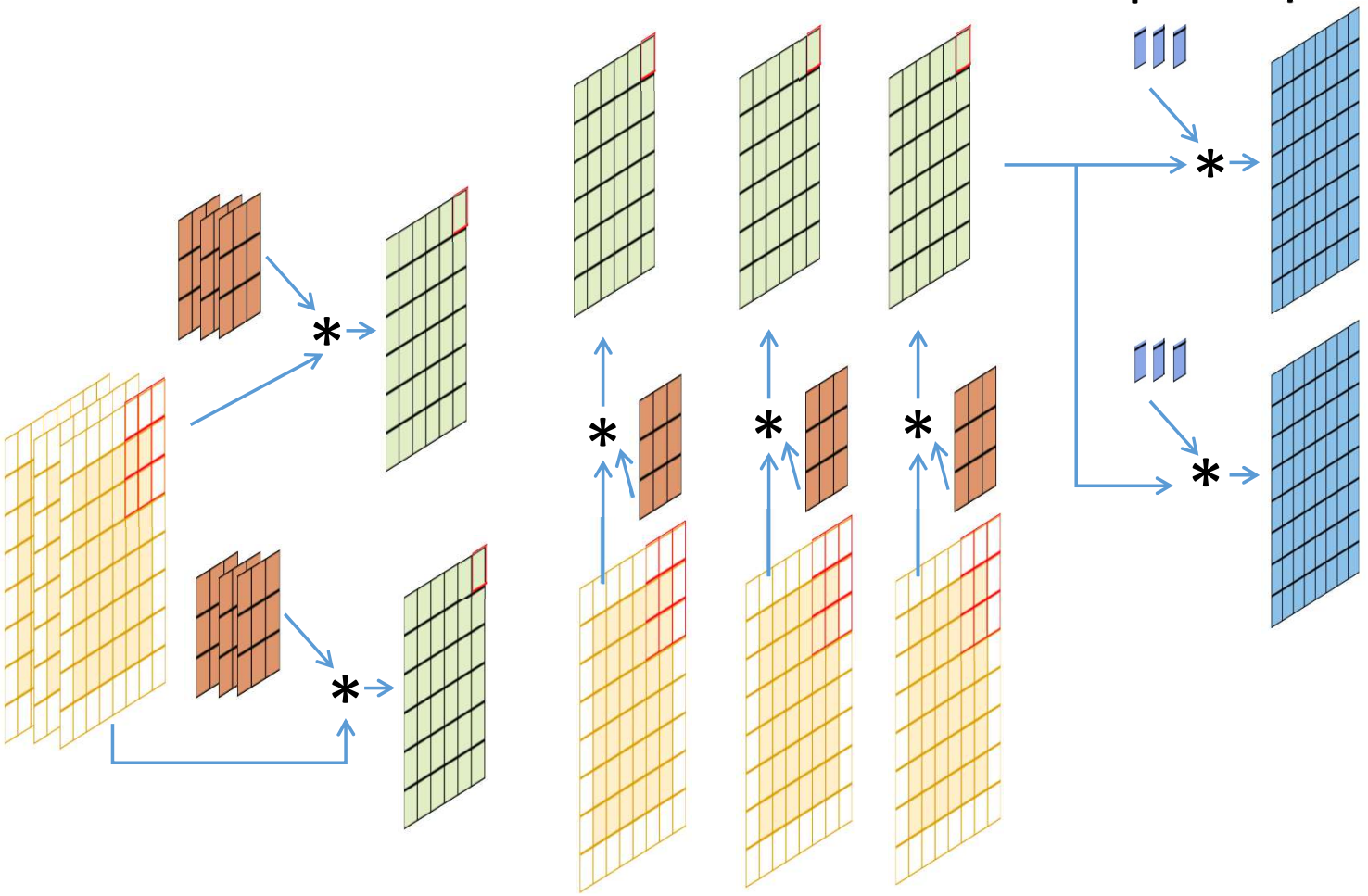


Image pixels 5x5 kernel 3x3 kernel 5x5 dilated kernel with only 3x3 trainable weights

Transposed (de-) convolution increases feature map size



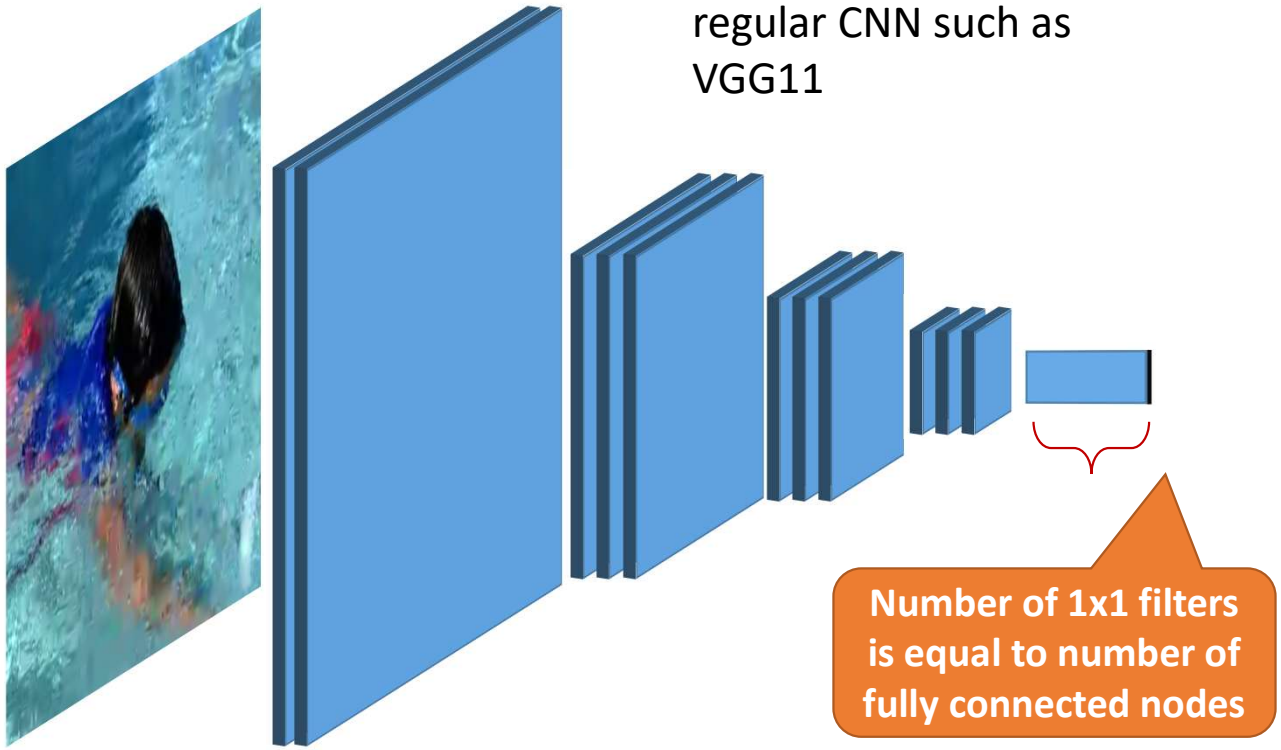
MobileNet filters each feature map separately



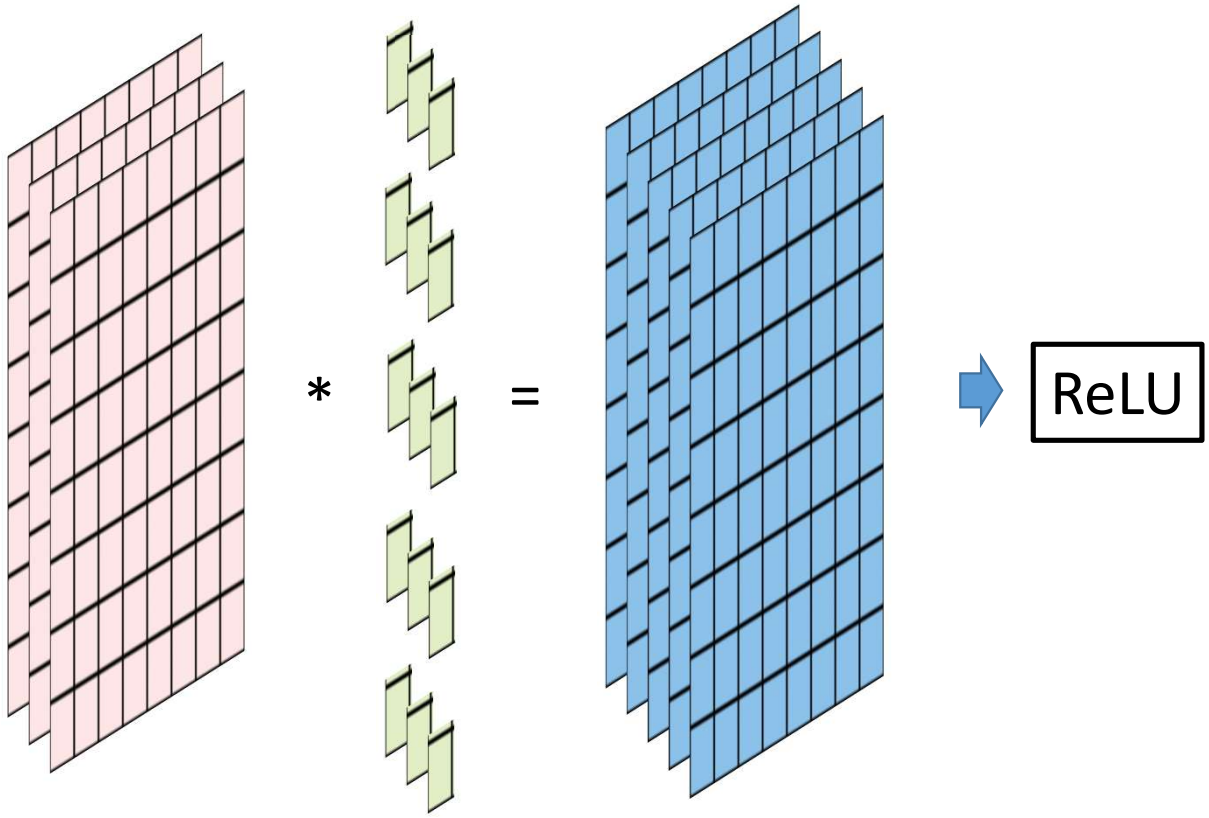
"MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" by Andrew G. Howard Menglong Zhu Bo Chen Dmitry Kalenichenko Weijun Wang Tobias Weyand Marco Andreetto Hartwig Adam, 2017

Using 1x1 convolutions is equivalent to having a fully connected layer

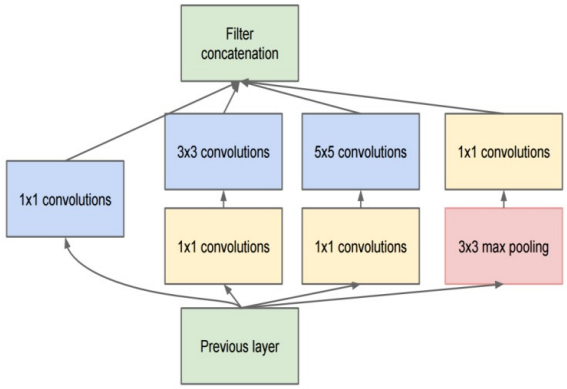
- This way, a fully convolutional network can be constructed from a regular CNN such as VGG11



1x1 convolutions can also be used to change the number of feature maps



Inception uses multiple sized convolution filters



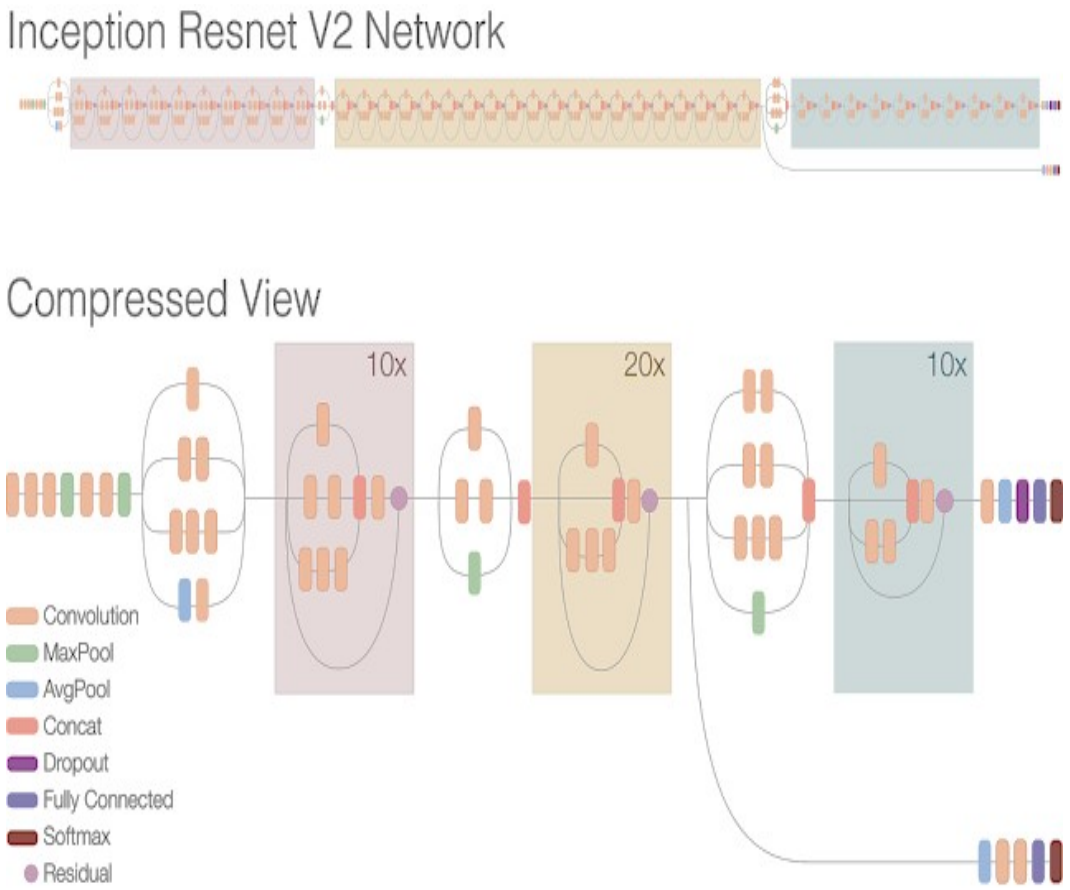
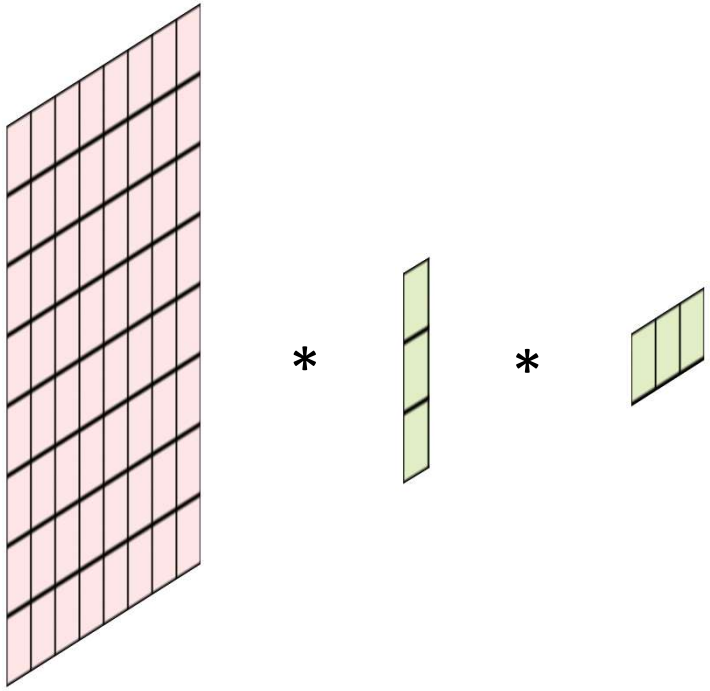


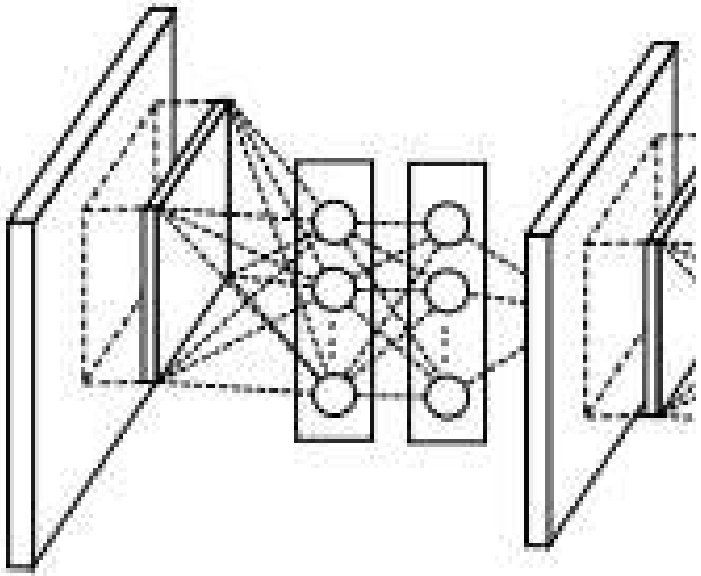
Image source: https://ai.googleblog.com/2016/08/improving-inception-and-image.html

Separable convolutions



Network in network

- Instead of a linear filter with a nonlinear squashing function, N-i-N uses an MLP in a convolutional (sliding) fashion



Source: "Network in Network" by Min Lin, Qiang Chen, Shuicheng Yan, https://arxiv.org/pdf/1312.4400v3.pdf

Variations of pooling are also available, e.g. stochastic pooling

- Average pooling (subsampling):

$$s_j = \frac{1}{|R_j|} \sum_{i \in R_j} a_i$$

- Max pooling:

$$s_j = \max_{i \in R_j} a_i$$

- Stochastic pooling:
  - Define probability:

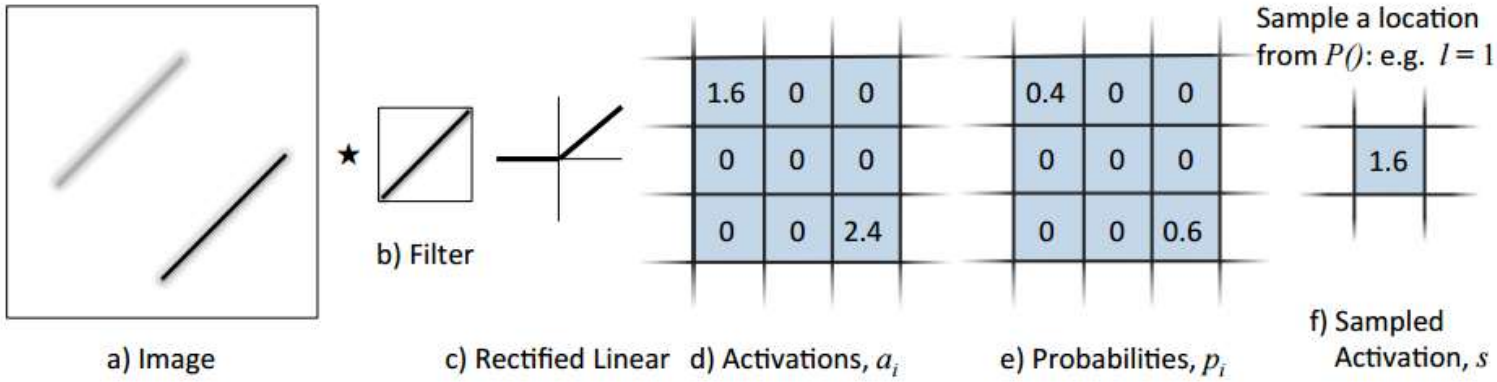
$$p_i = \frac{a_i}{\sum_{k \in R_j} a_k}$$

  - Select activation from multinomial dist $s_j = a_l \quad \text{where } l \sim P(p_1, \ldots, p_{|R_j|})$
  - Backpropagation works just like max pooling
    - Keep track of l that was chosen (sampled)
  - During testing, take a weighted average of activations

$$s_j = \sum_{i \in R_j} p_i a_i$$

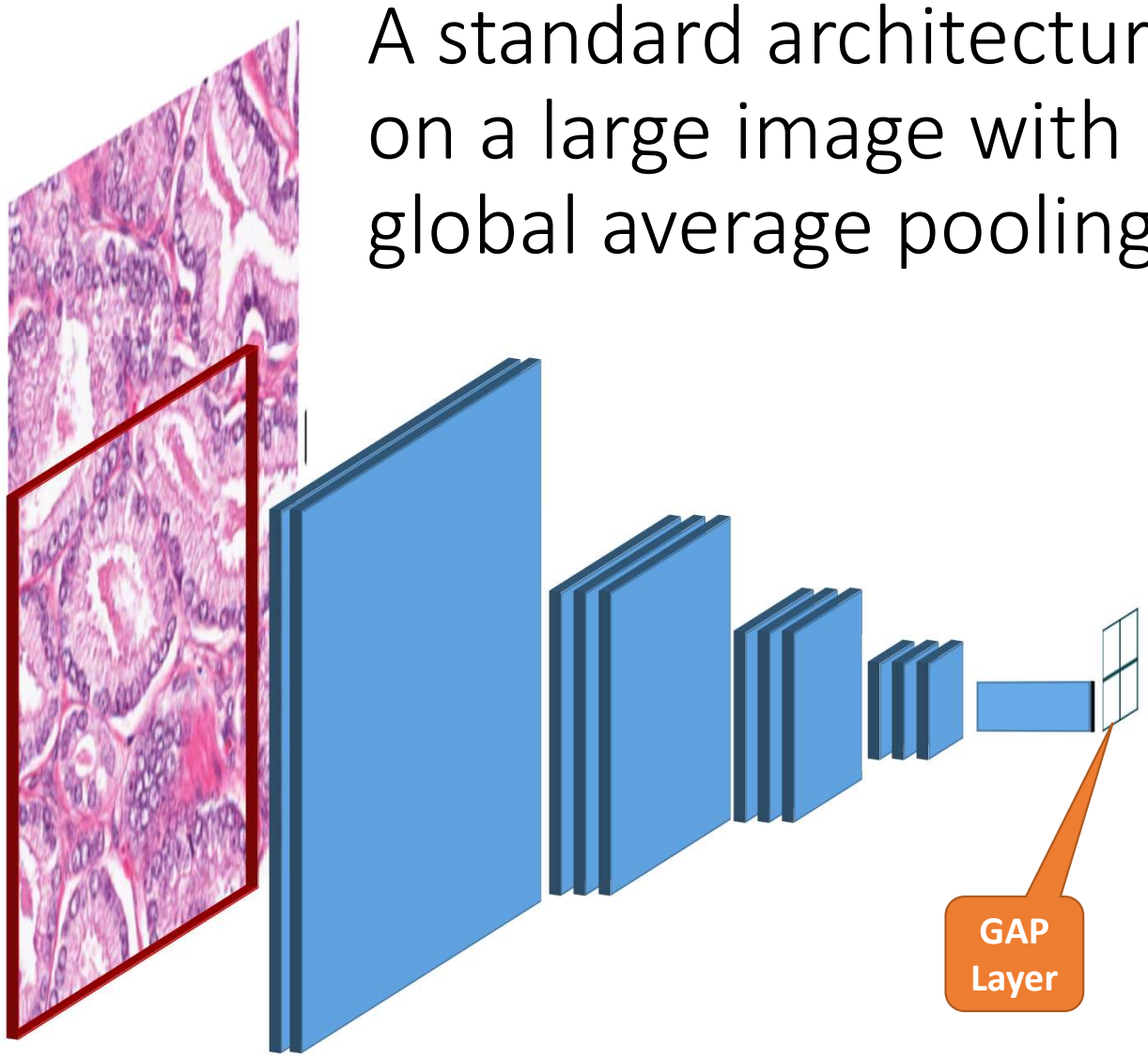
Source: "Stochastic Pooling for Regularization of Deep Convolutional Neural Networks", by Zeiler and Fergus, in ICLR 2013.

Example of stochastic pooling



Source: "Stochastic Pooling for Regularization of Deep Convolutional Neural Networks", by Zeiler and Fergus, in ICLR 2013.

A standard architecture on a large image with global average pooling

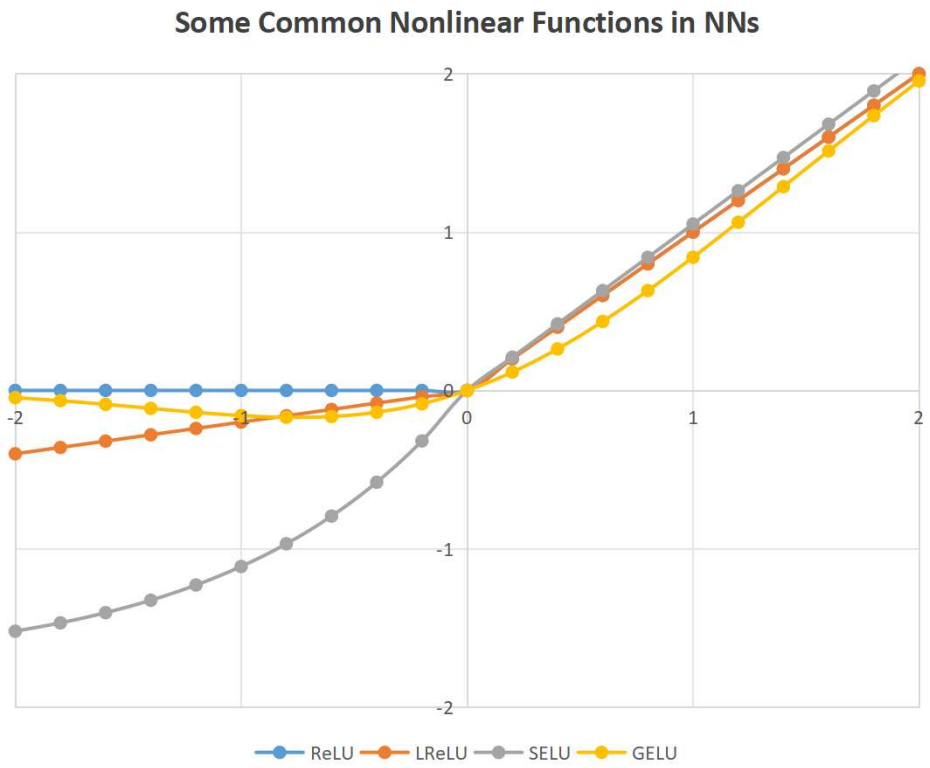


Summary of convolutional layers

- Purpose: Detect **local** features that can occur in **any location**
- Neuron with limited receptive field
- Applied to multiple locations of previous layer output
- Filter can slide along 1D (audio), 2D (image), 3D (MRI/CT, video), ...
- Channel is (usually) **not the same** as D+1
- Can be square, rectangular, or depthwise, or 1x1
- Output has same H and W (almost) as the input
- Has a stride:
  - Conventionally stride is 1
  - Stride > 1 leads to a pooling type effect
- Sparse / atrous filter can increase receptive field without increasing params
- Upconv: Inserting interveneing zeros in input can lead to a larger sized output

Summary of nonlinear layer

- Purpose: Introduce nonlinearity in the NN function
- Should be:
  - Non-polynomial
  - Easy to optimize with
    - Continuous
    - Mostly smooth
    - Gradient does not blow up
    - Gradient is not zero almost everywhere
    - Has few modes of behavior (ReLU has two: on, off)
- Examples: ReLU, LReLU, SELU, GeLU



Summary of pooling layer

- Purpose: pool/collect features from larger receptive field
- Effect is reduction of H and W
- Can be done in different ways:
  - Average
  - Max
  - Stochastic
  - Learnable
- Can have variable footprint
  - 2x2 is the most common
  - Global average pooling (take average over entire variable H x W, channel-wise)

Summary of flatten layer

- Purpose: Bridge between feature maps and fully connected layer
- Usually, when feature map has reduced a lot due to multiple pooling layers
- Convert H x W x C to HWC x 1
  - Or $H/2^n$ x $W/2^n$ x C_m to ($HWC_m/2^{2n}$ x 1)

Summary of fully connected layers

- Purpose: Final feature extraction and classification / regression
- Similar to a multi-layer perceptron layers
  - Every output of the previous layer is connected to every neuron in the next layer
  - Has a nonlinearity (usually), e.g.,
    - ReLU if not the final layer
    - Softmax for final classification layer (class probability)
      - Logits are BEFORE the sigmoid or softmax nonlinearity
      - Class probabilities are AFTER the sigmoid or softmax nonlinearity
    - Sigmoid for binary classification or multi-class classification (more than one correct class)
  - For regression, final layer has no nonlinearity (usually)

Skip connections

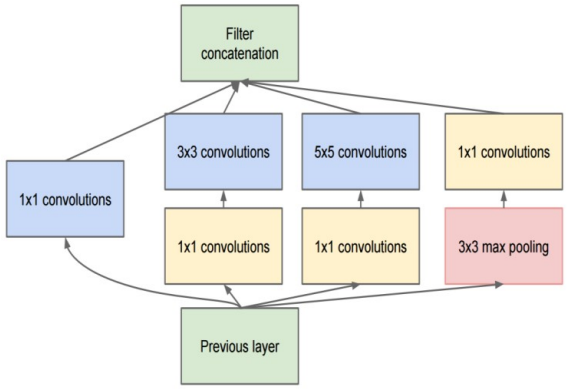
- Purpose: Ease gradient flow, multi-resolution processing, expand hypothesis space
- One layer's output *x* can **skip (bypass)** the next layer *f*
- It can then be an input to a subsequent layer *h*
- Types of skips:
  - Addition (residual): $h(x + f(x))$ — All three dims must match
  - Concatenation: $h(x \copyright f(x))$ — All but one dims must match
  - Multiplication (point-wise): $h(x \odot f(x))$ — All three dims must match
  - Custom, e.g., attention: $h(g(x) \odot f(x))$ — All three dims must match
- Dimension matching is important in skip connections

Layer design: Deep, but smaller convolutions

- Starting with VGGNet, most conv nets use small filter sizes
  - Why? Detect spatial co-occurrence of features that aren't distant
- Only the first conv layer may have a larger filter size
  - Why? In the image space, we need to detect complex features

Source: "Going Deeper with Convolutions" Szegedy et al., CVPR' 15

Inception uses multiple sized convolution filters



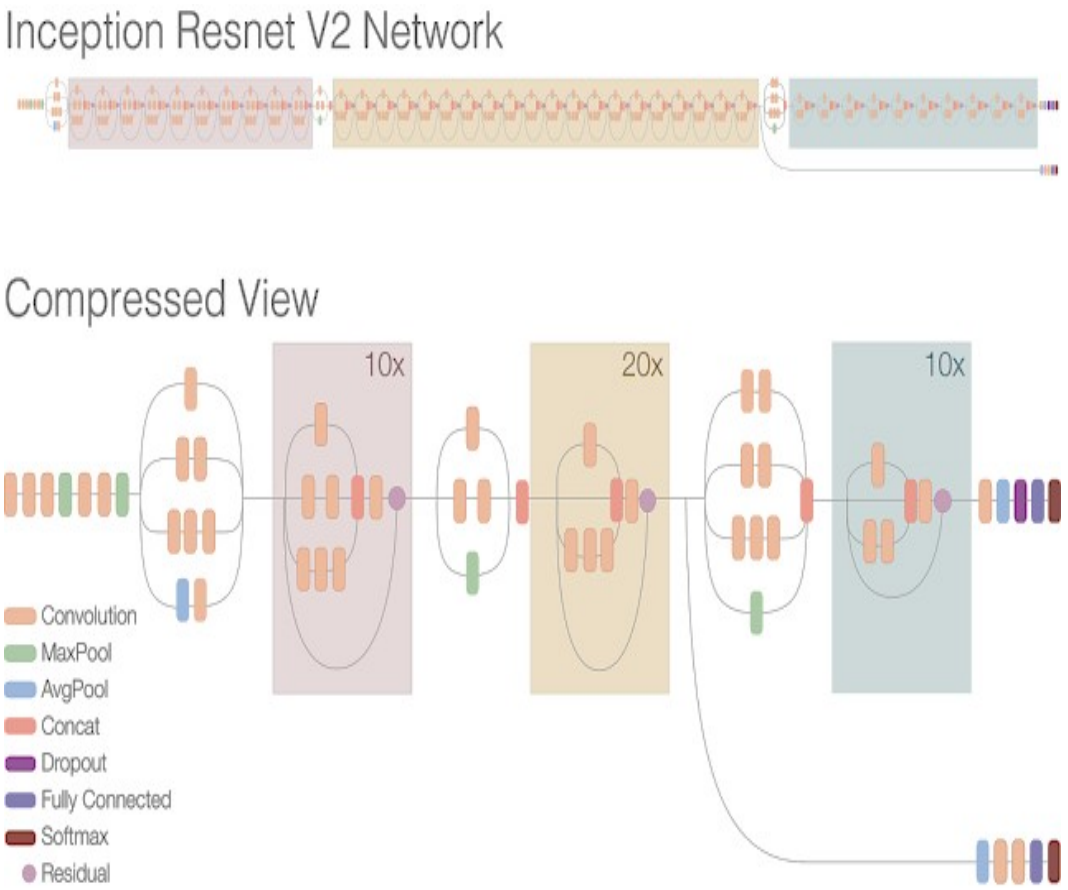
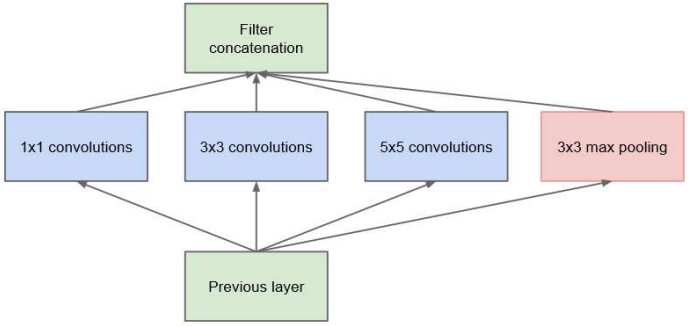
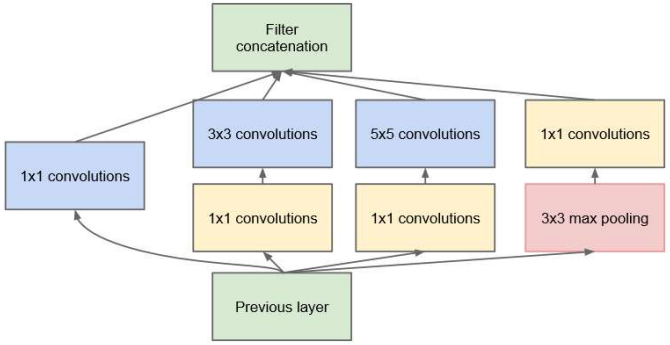


Image source: https://ai.googleblog.com/2016/08/improving-inception-and-image.html

InceptioNet makes heavy use of 1x1 convolutions to detect sparse features



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

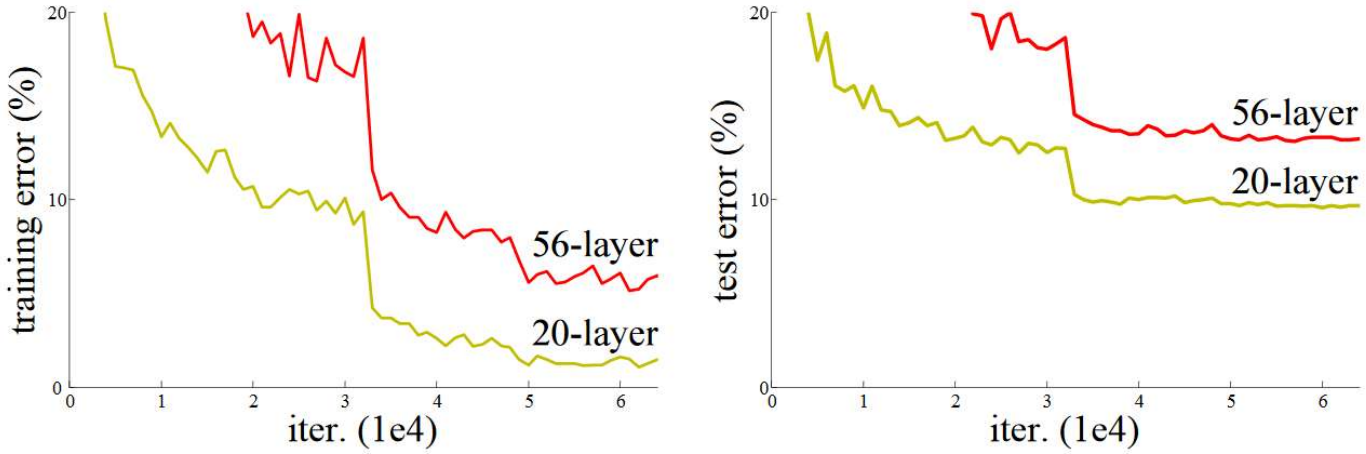
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture.

Source: “Going Deeper with Convolutions” Szegedy et al., CVPR’ 15

Empirically, deep networks are hard to train

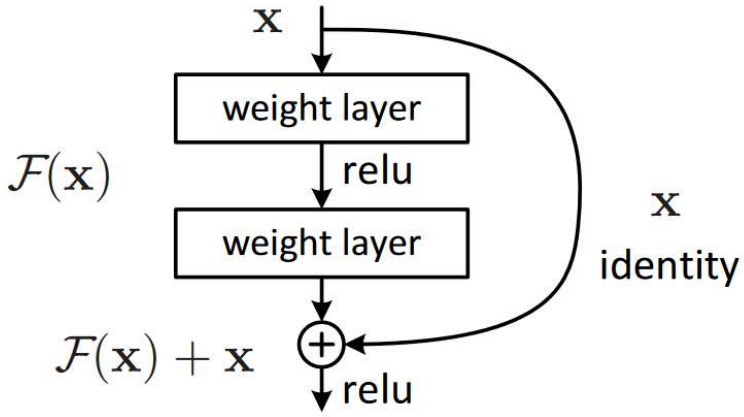
- Shouldn't <u>training</u> error decrease with more layers? But, it doesn't due to the gradient getting stuck



Source: "Deep Residual Learning for Image Recognition" by He, Zhang, Ren, Sun, https://arxiv.org/pdf/1512.03385v1.pdf

Residual skip connections

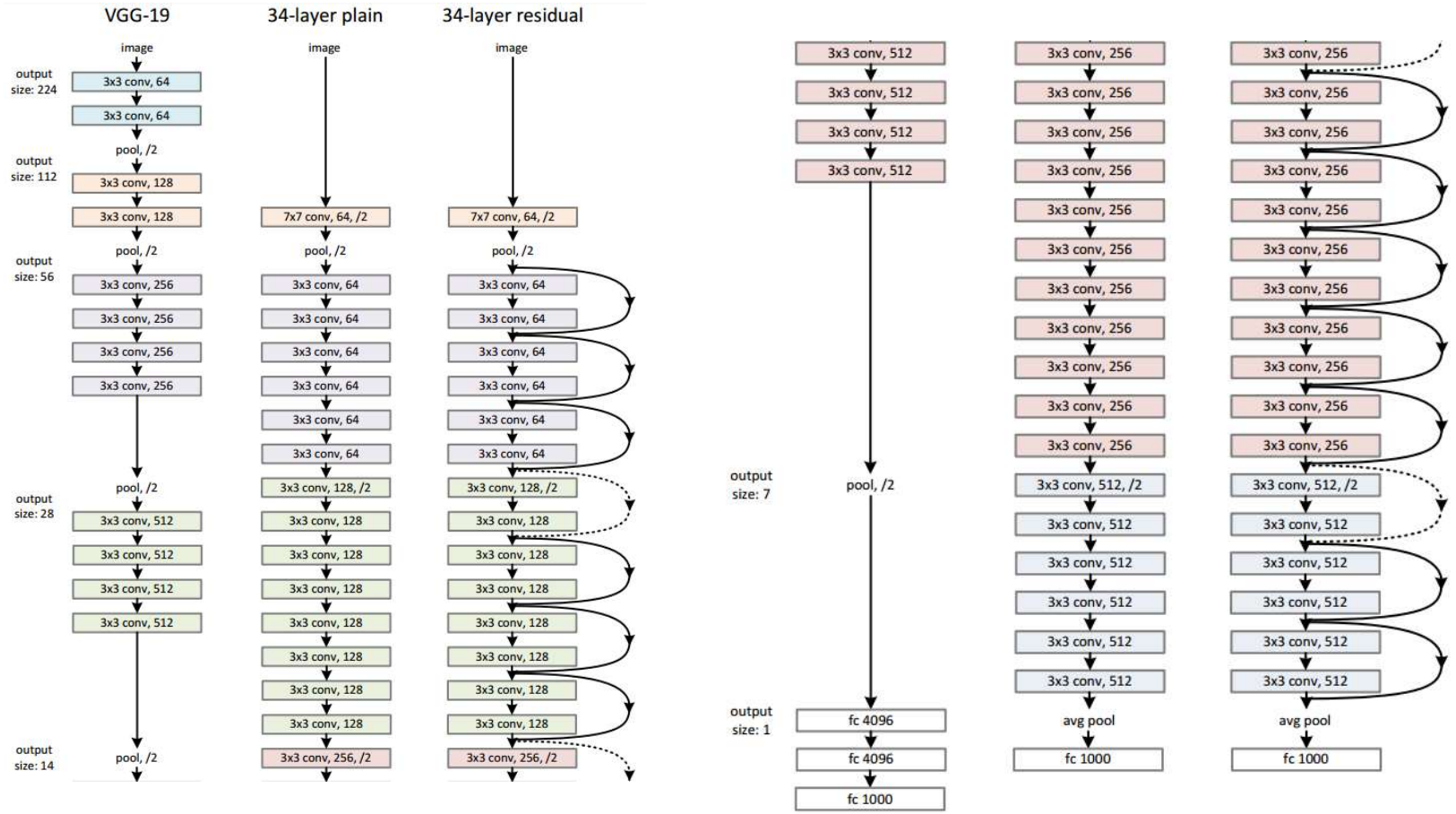
- Train H(x) of the form F(x) + x
- F(x) can be thought of as a residual



- Hierarchica[illegible] idea
- Makes the problem well-conditioned
- Can learn identity mapping; can't be any worse
- More layers, lower complexity!

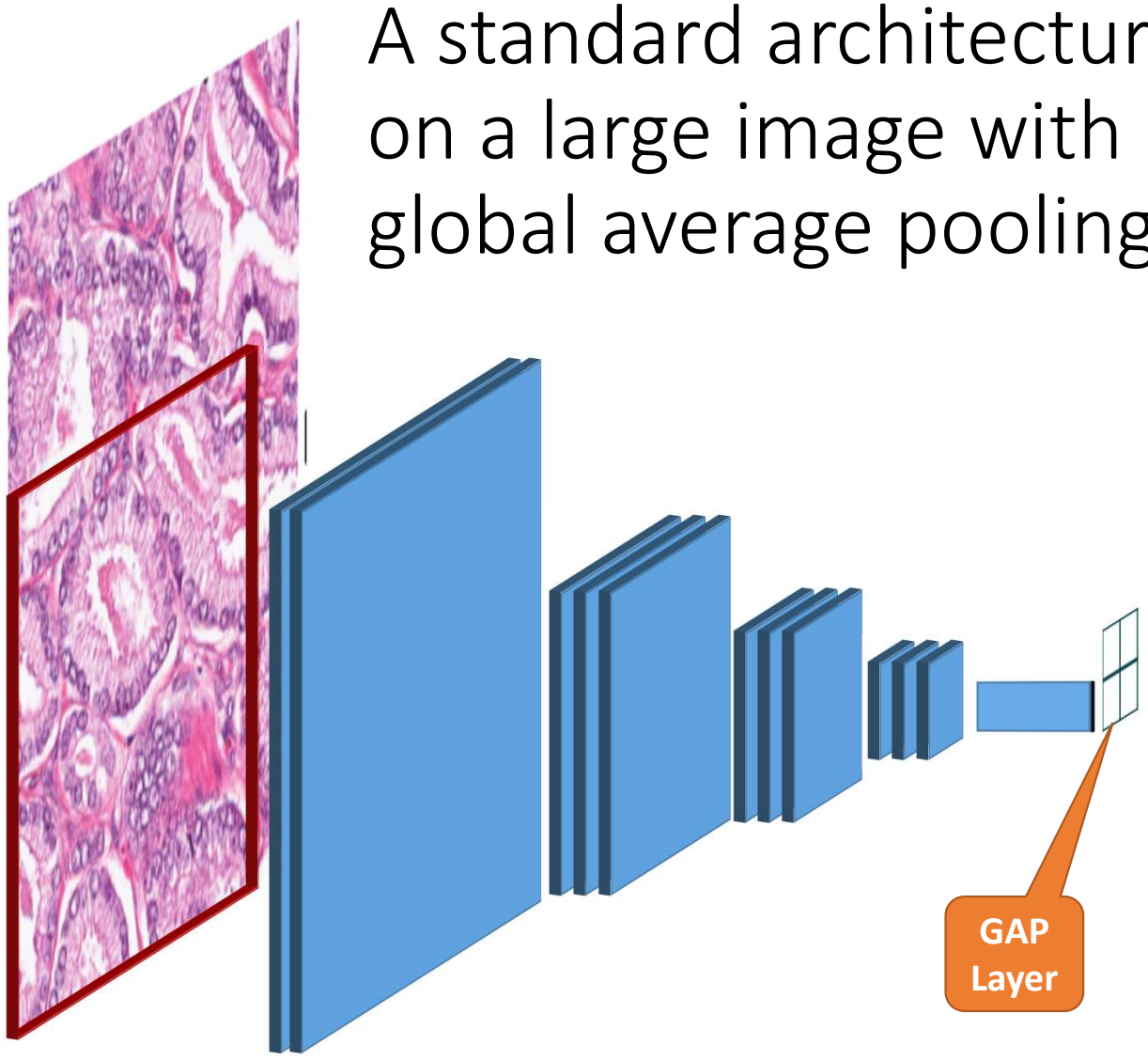
Source: "Deep Residual Learning for Image Recognition" by He, Zhang, Ren, Sun, https://arxiv.org/pdf/1512.03385v1.pdf

Compared to VggNet



Source: “Deep Residual Learning for Image Recognition” by He, Zhang, Ren, Sun, https://arxiv.org/pdf/1512.03385v1.pdf

A standard architecture on a large image with global average pooling



Semantic segmentation is labeling pixels according to their classes

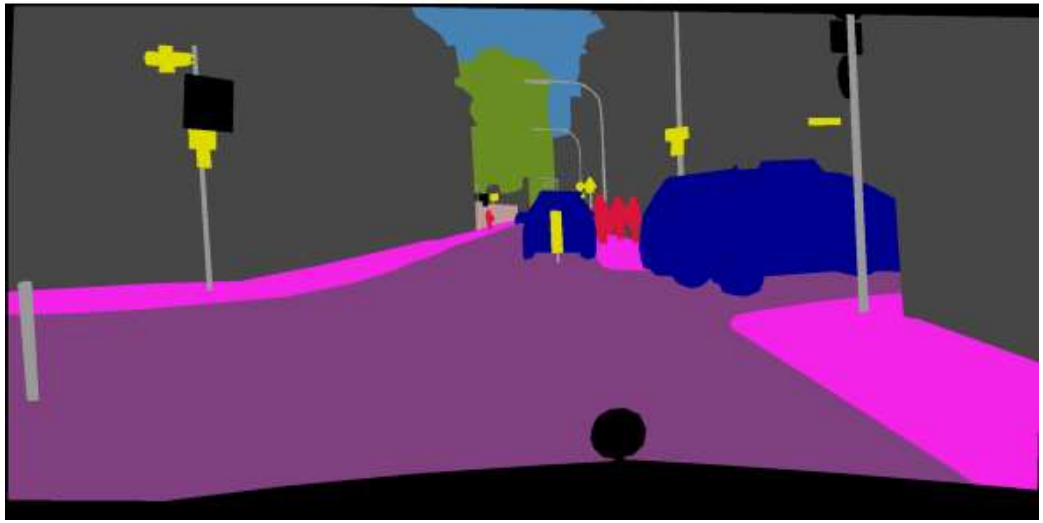
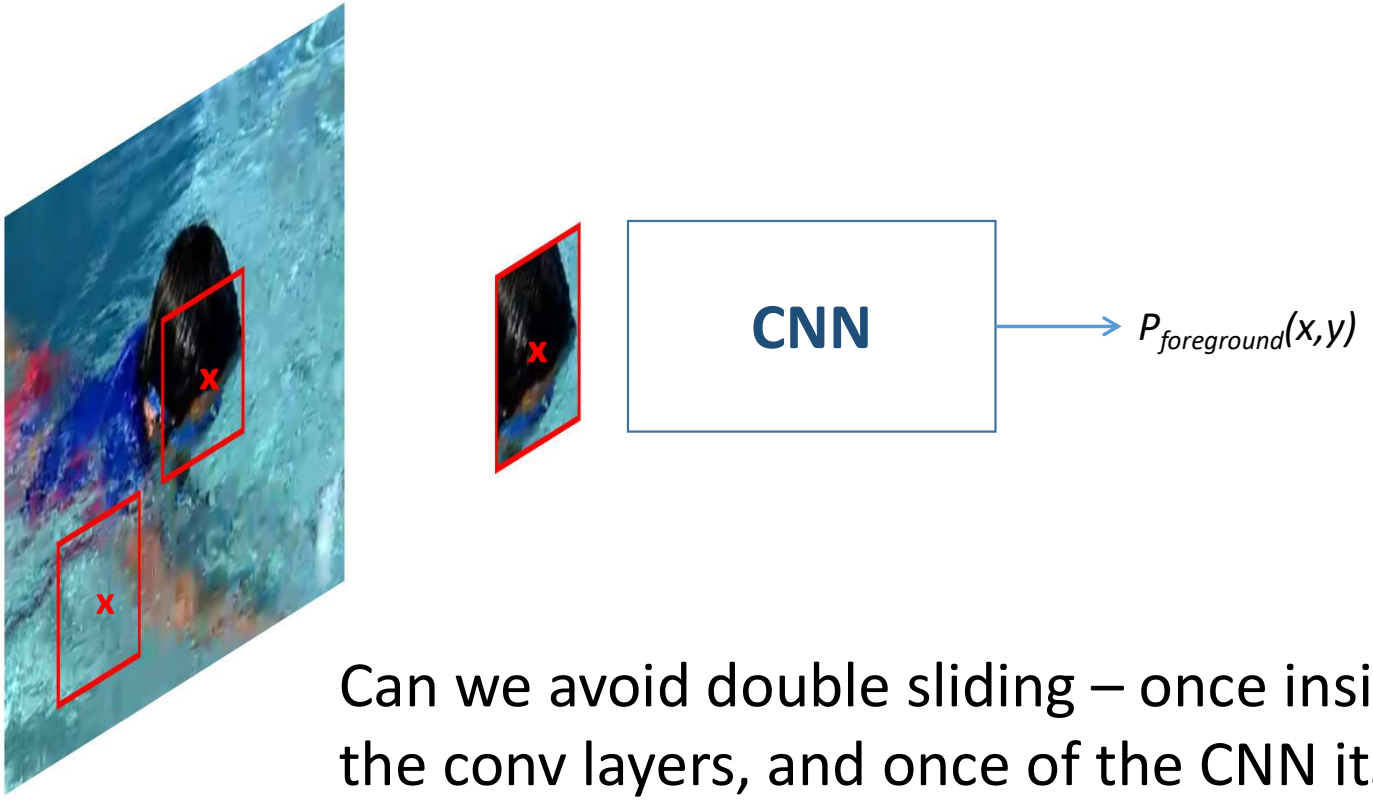


Image Source: "ICNet for Real-Time Semantic Segmentation on High-Resolution Images" Hengshuang Zhao1, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, Jiaya Jia, ECCV'18

For segmentation, a pixel class can be predicted using some spatial context



Can we avoid double sliding – once inside the conv layers, and once of the CNN itself?

Pixel labels for training images must be known to train for semantic segmentation

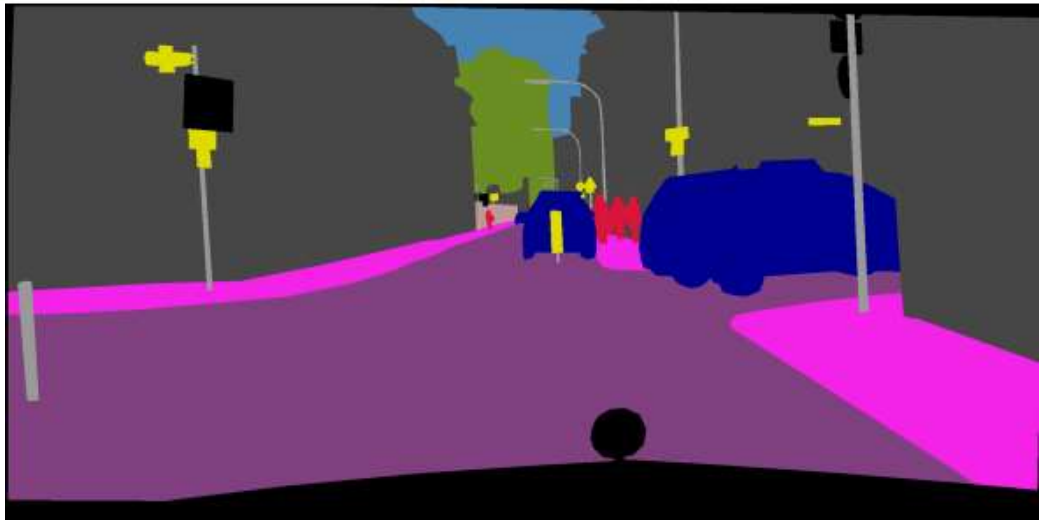
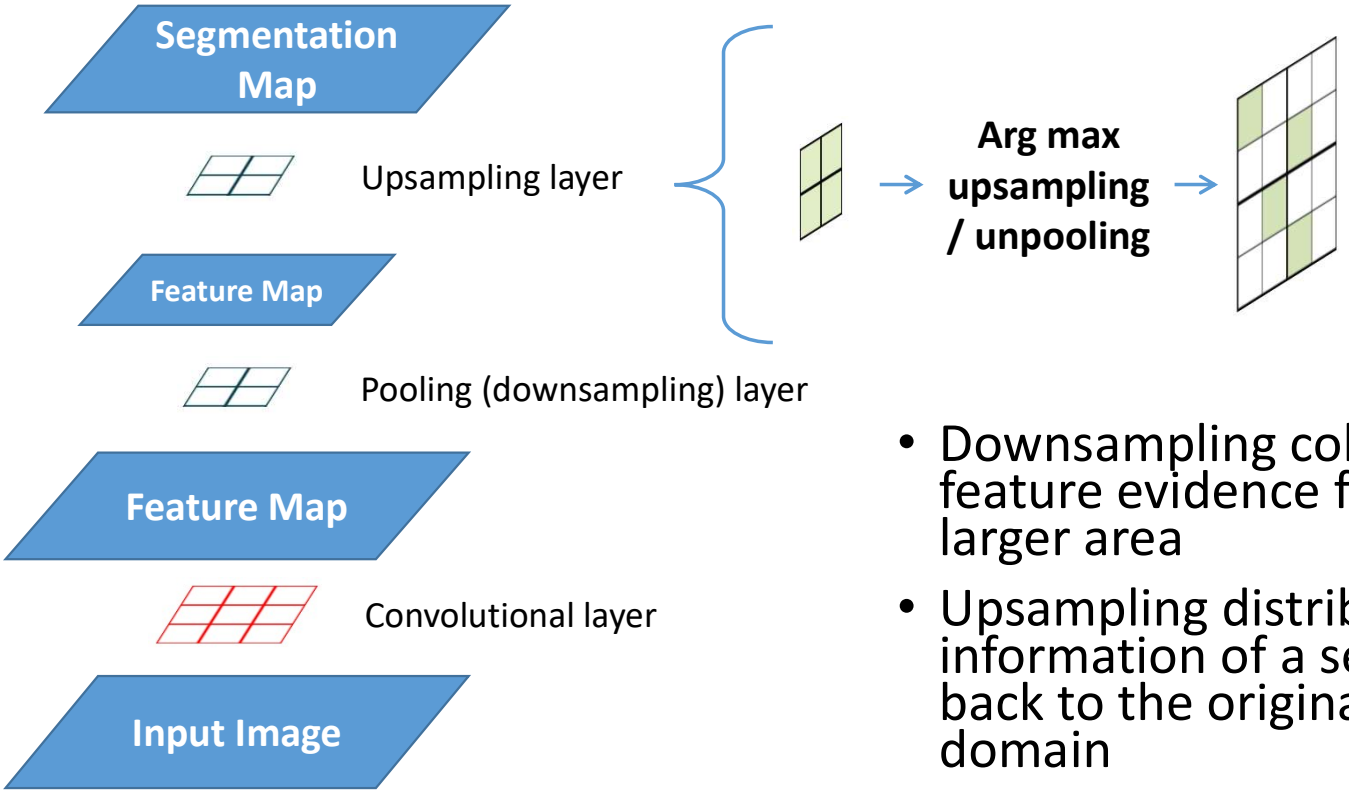


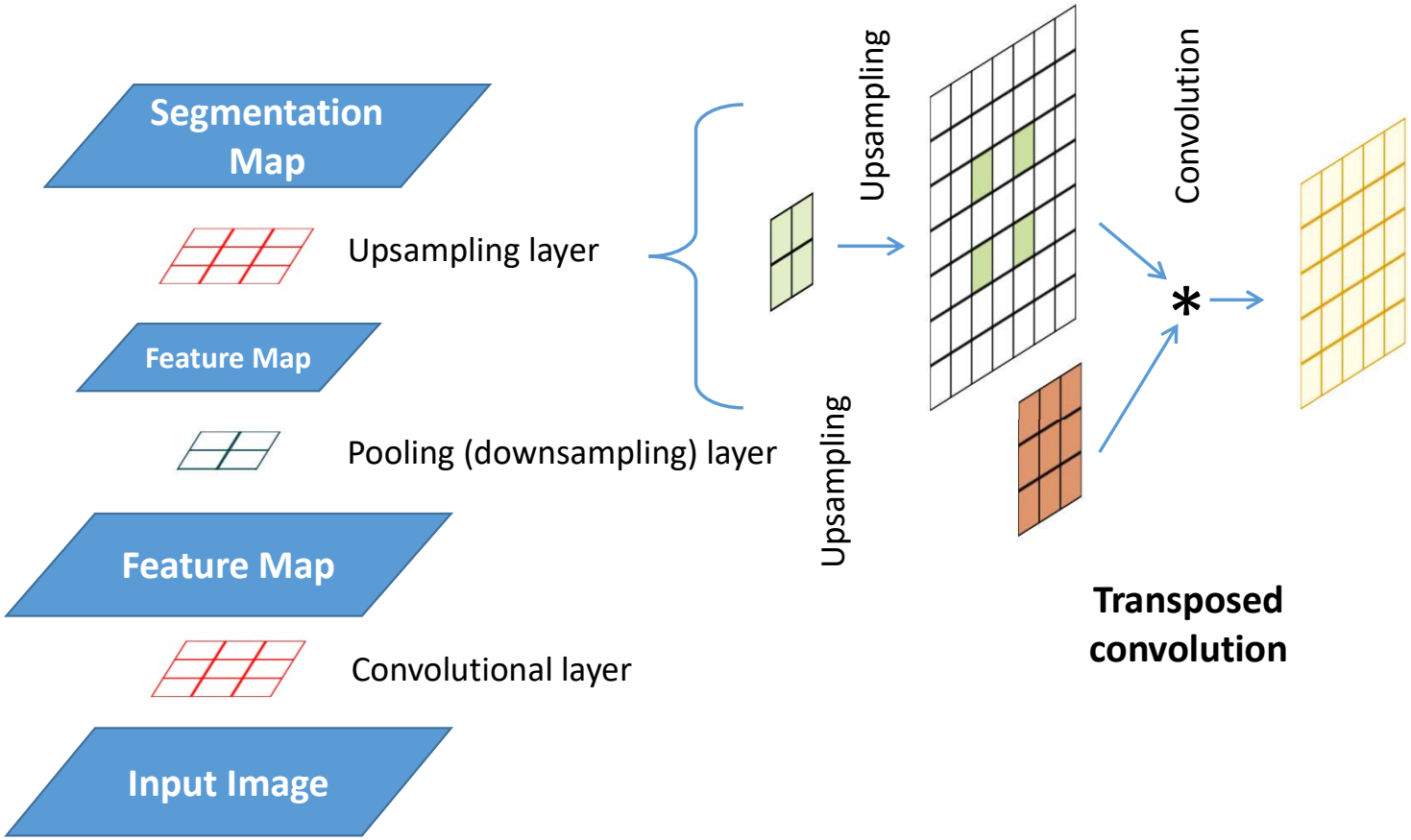
Image Source: "ICNet for Real-Time Semantic Segmentation on High-Resolution Images" Hengshuang Zhao1, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, Jiaya Jia, ECCV'18

To produce a segmentation map downsampling is followed by upsampling

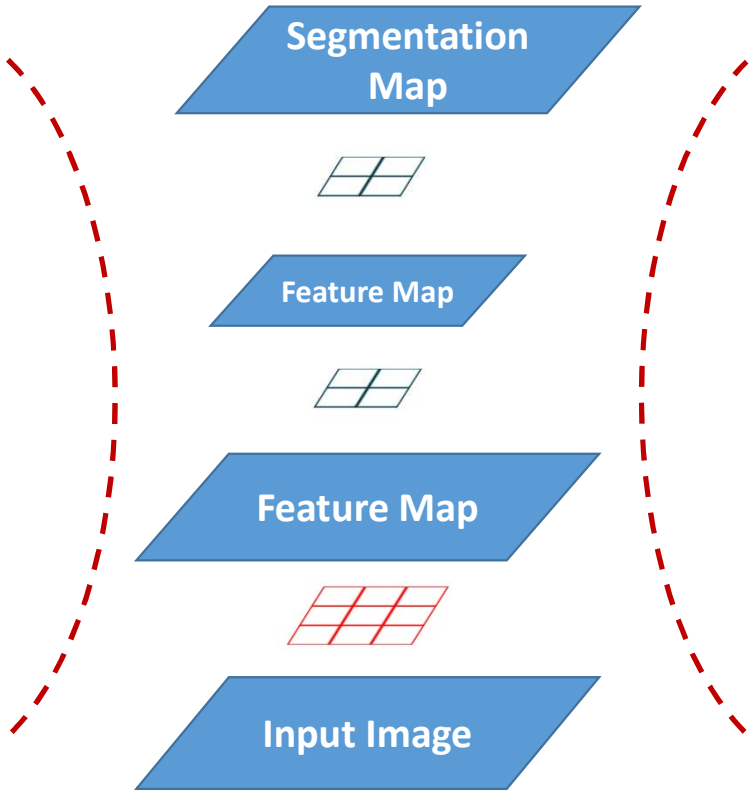


- Downsampling collects feature evidence from a larger area
- Upsampling distributes the information of a segment back to the original pixel domain

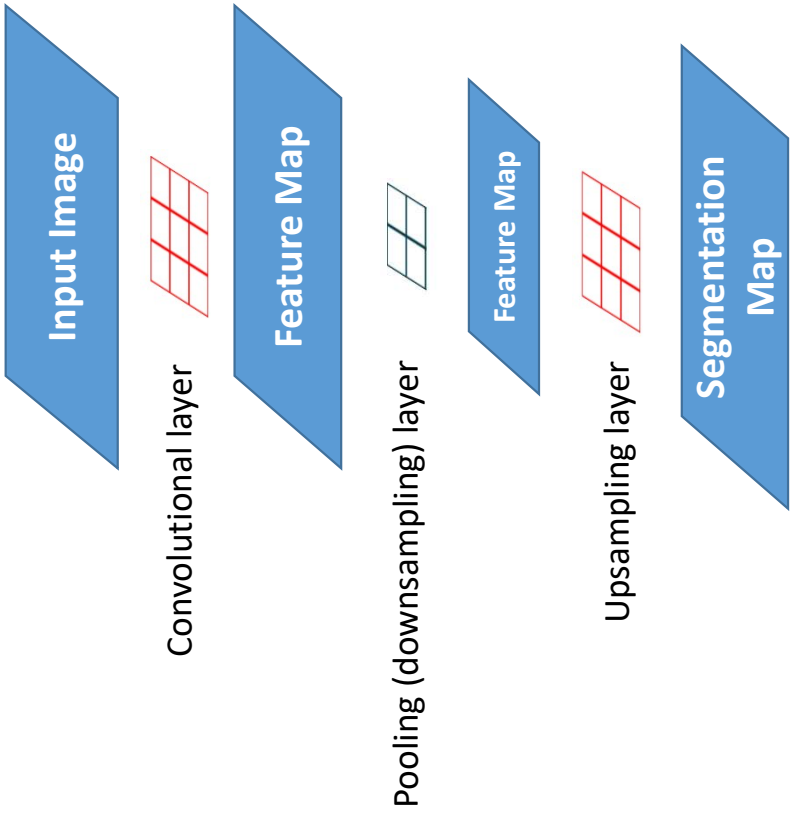
Upsampling can also be learned



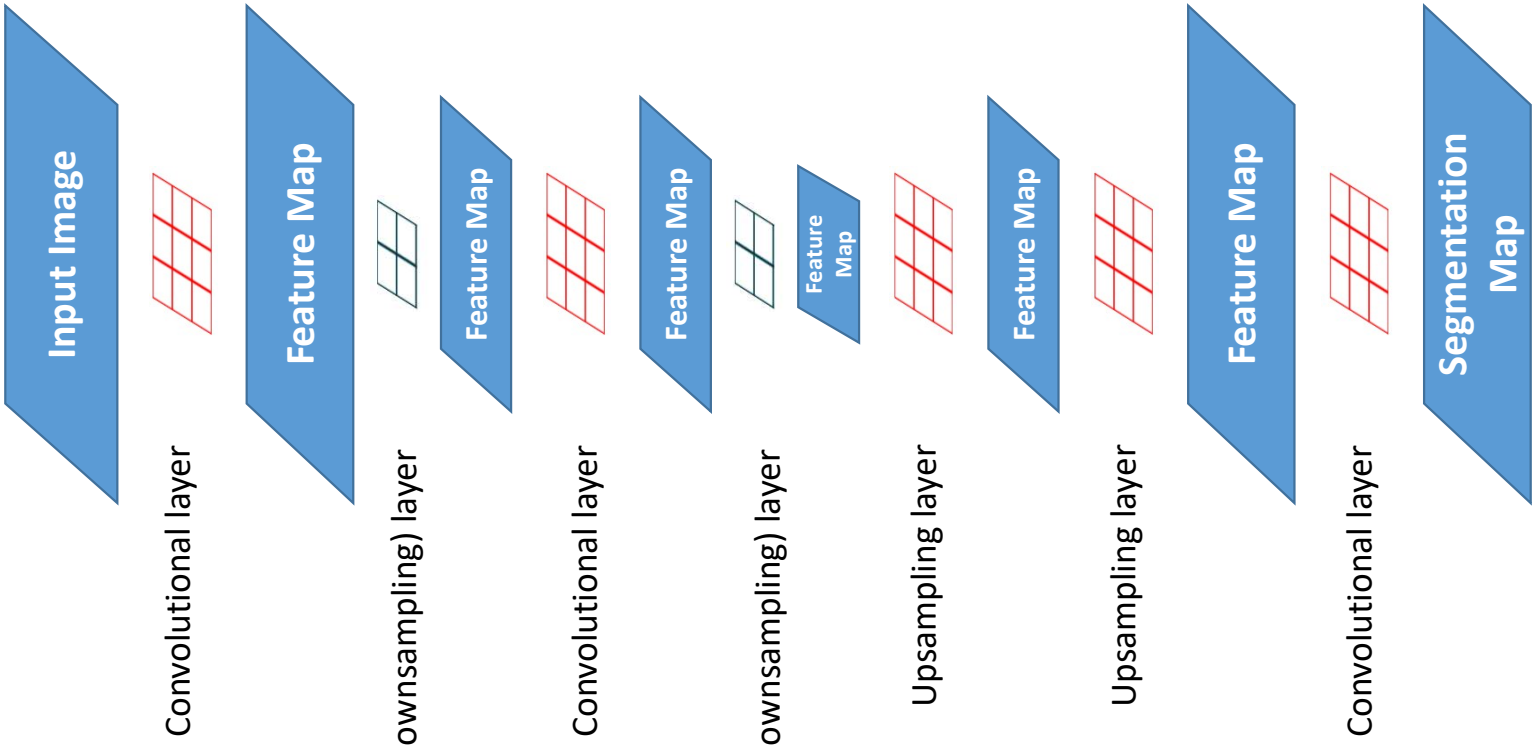
Downsampling and upsampling leads to an hour-glass structure



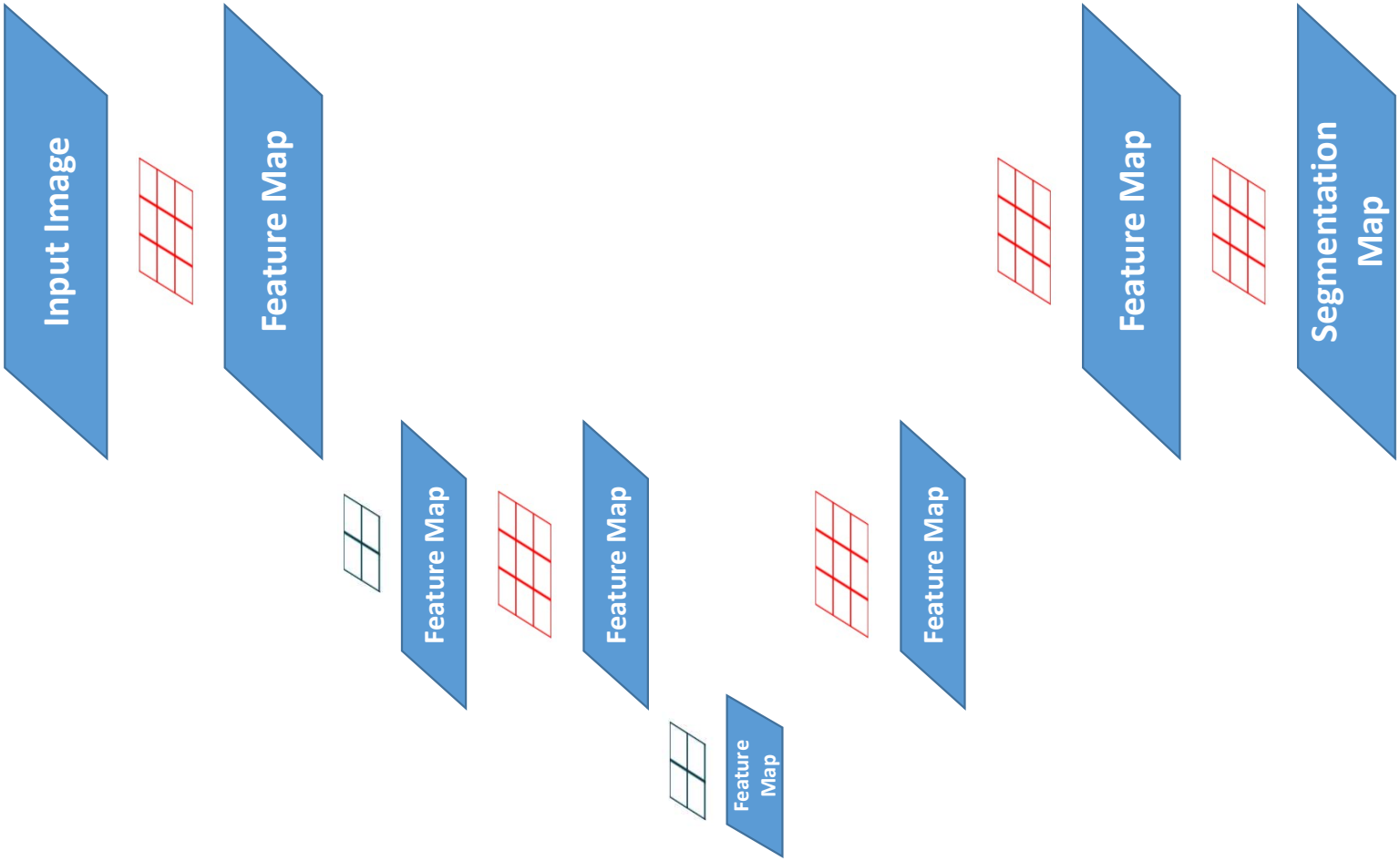
Let us rearrange the layers horizontally



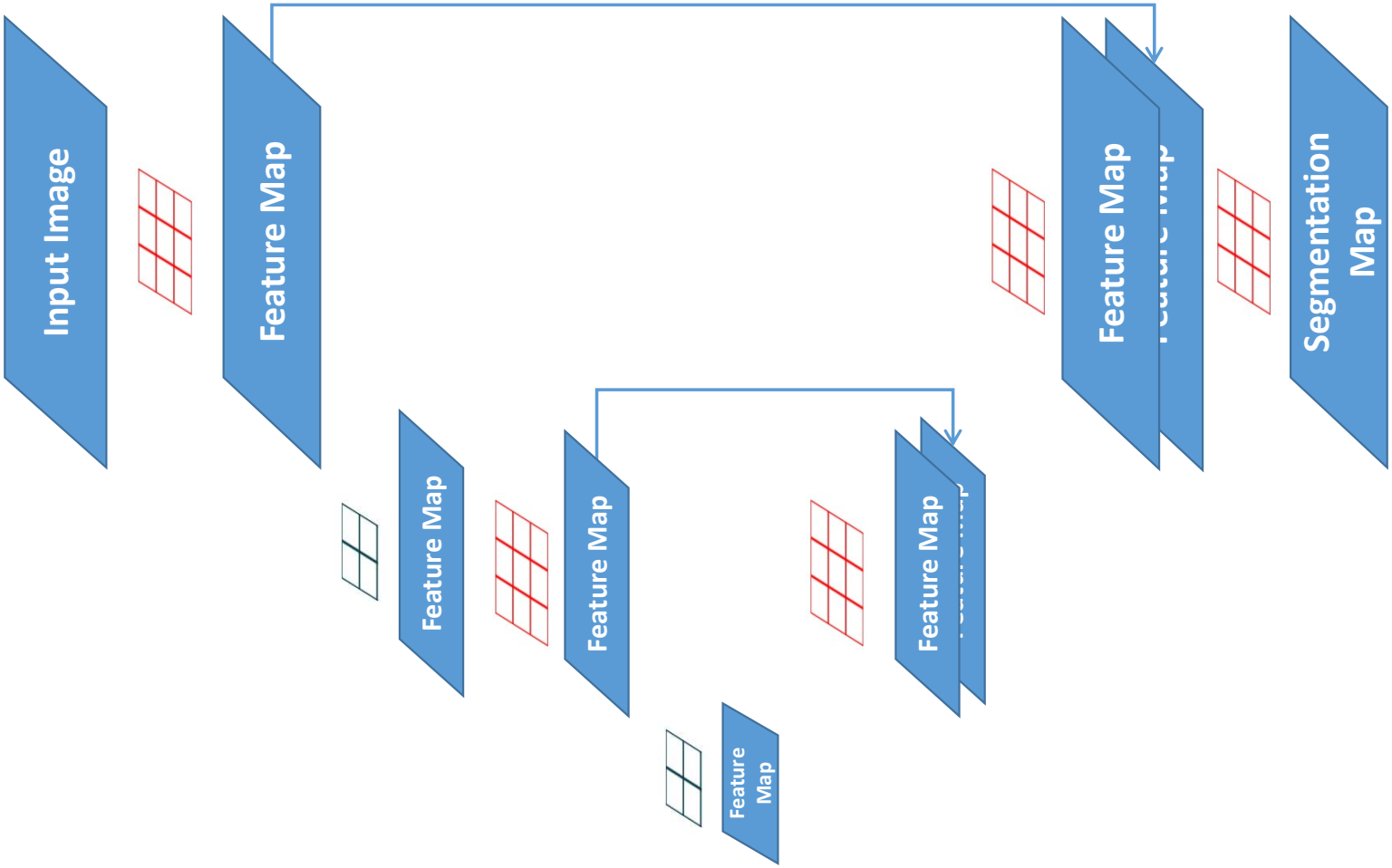
More layers can be added



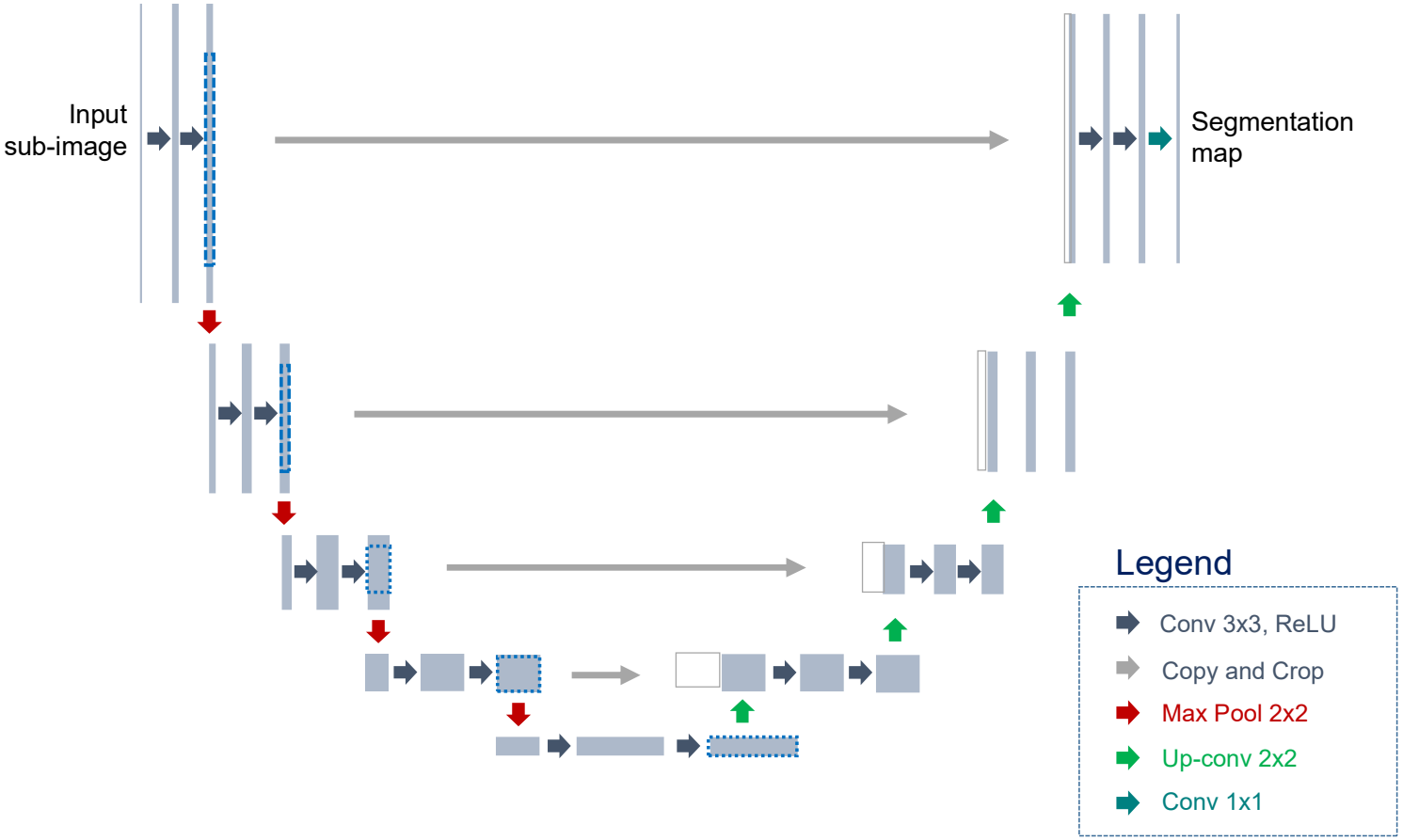
Visually rearrange layers in a big U



Concatenate previous feature maps for finer spatial context

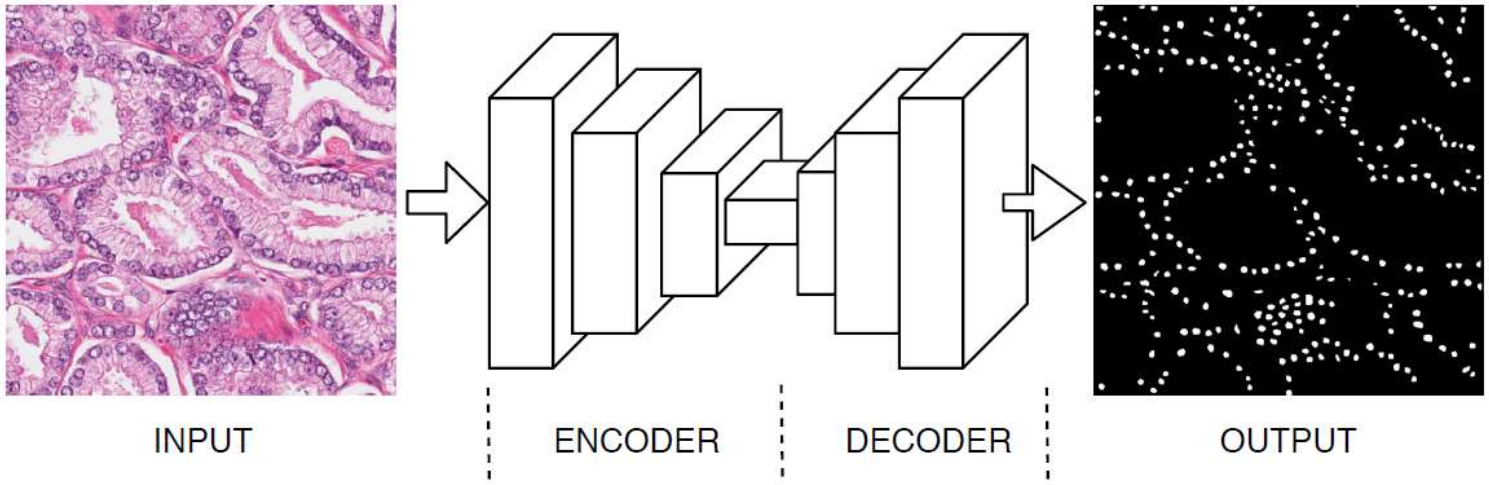


U-Net is based on the ideas described in the previous slides



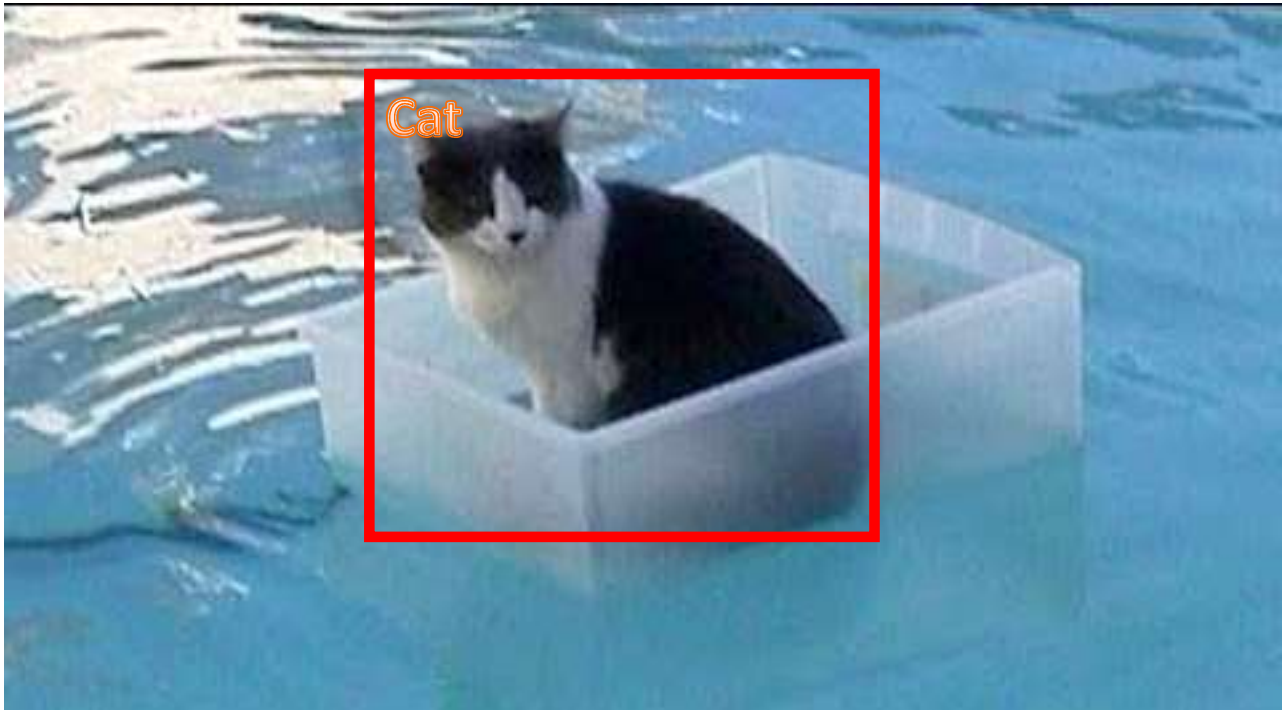
Source: "U-Net: Convolutional Networks for Biomedical Image Segmentation" Olaf Ronneberger, Philipp Fischer, Thomas Brox, 2015

A sample output for nucleus segmentation in pathology

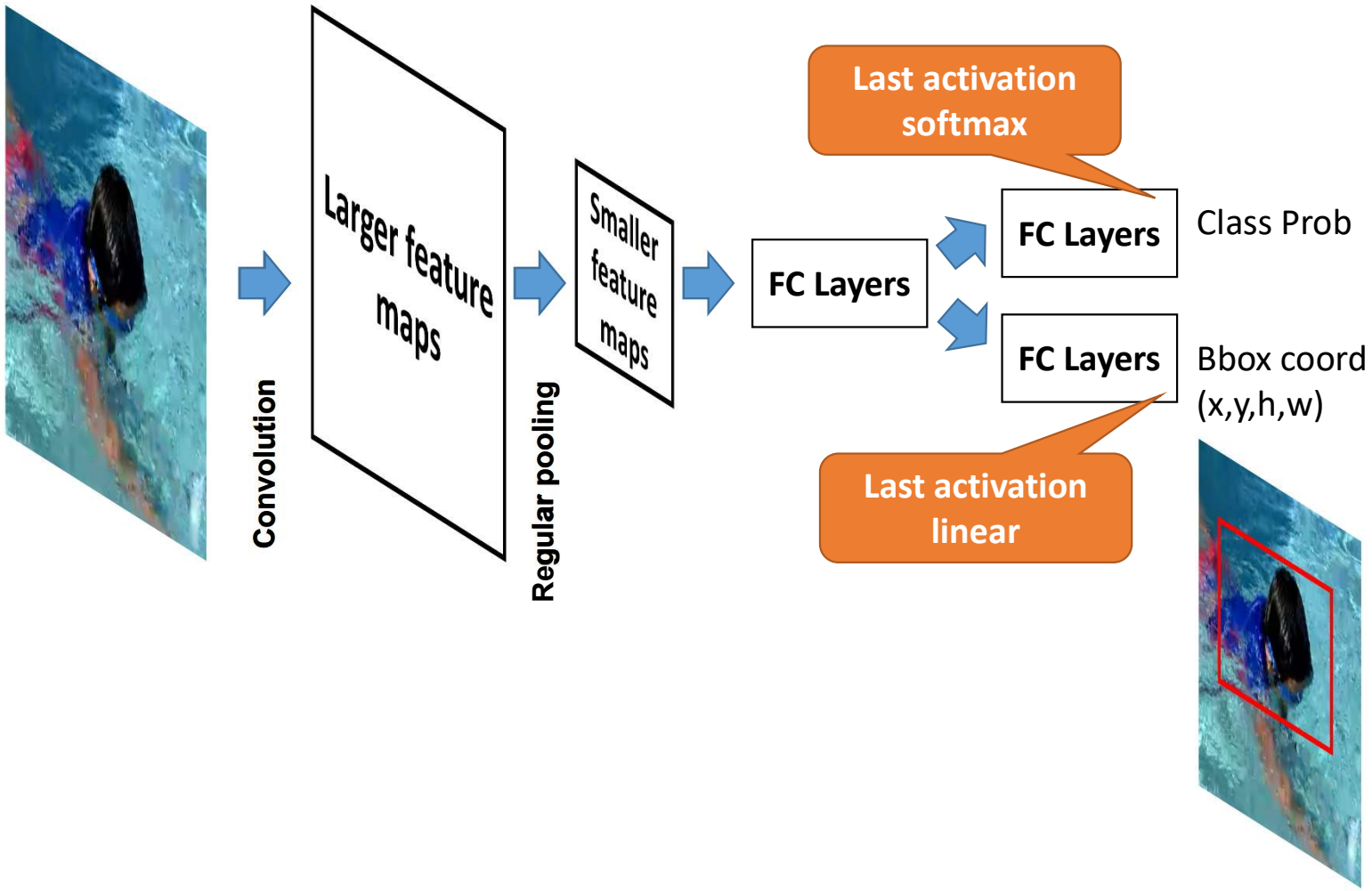


A general representation of fully convolutional networks. The encoder is composed of convolutional and pooling layers for downsampling and the decoder is composed of deconvolutional layers for upsampling.

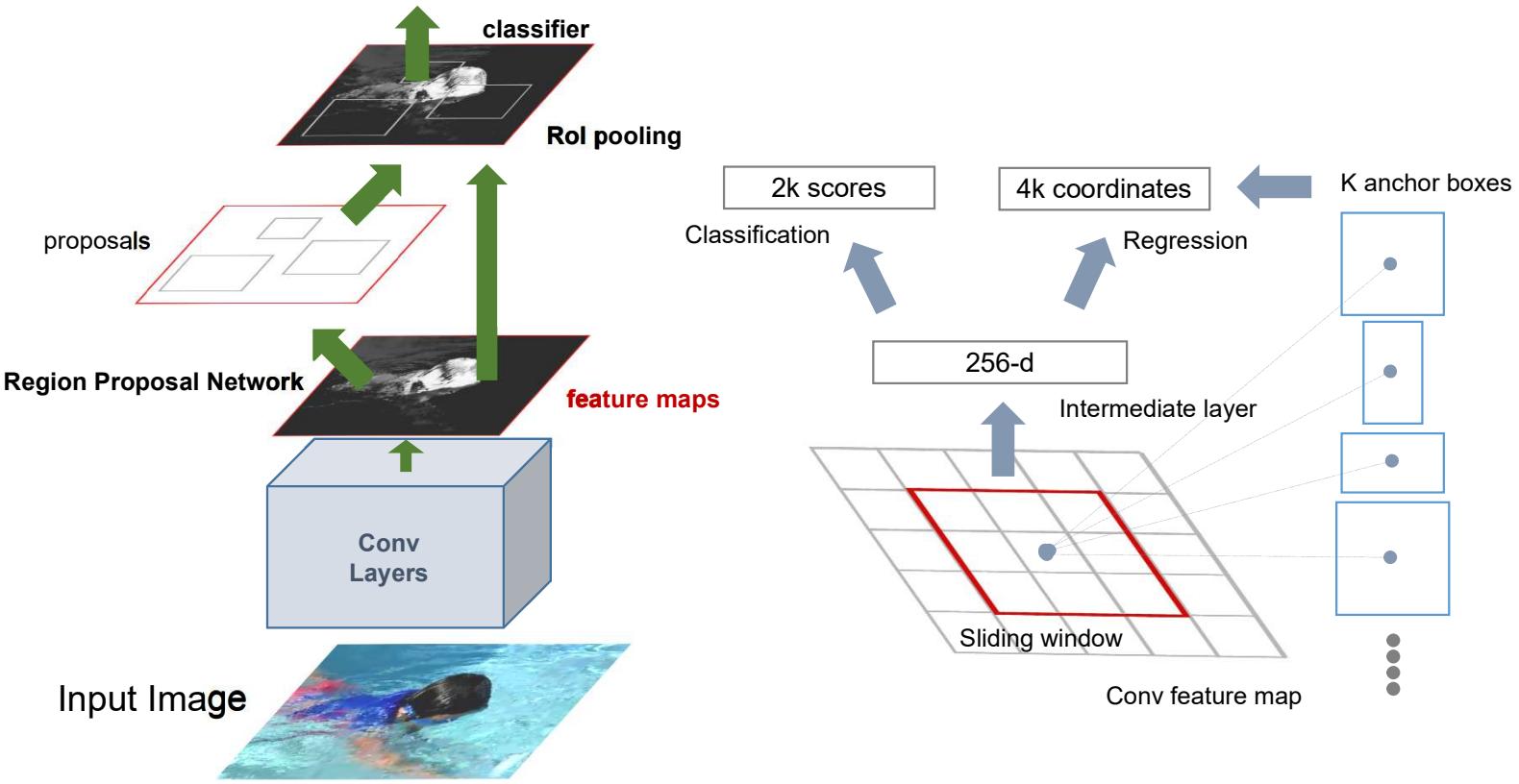
What is localization



We can train a regression network to give bounding box coordinates

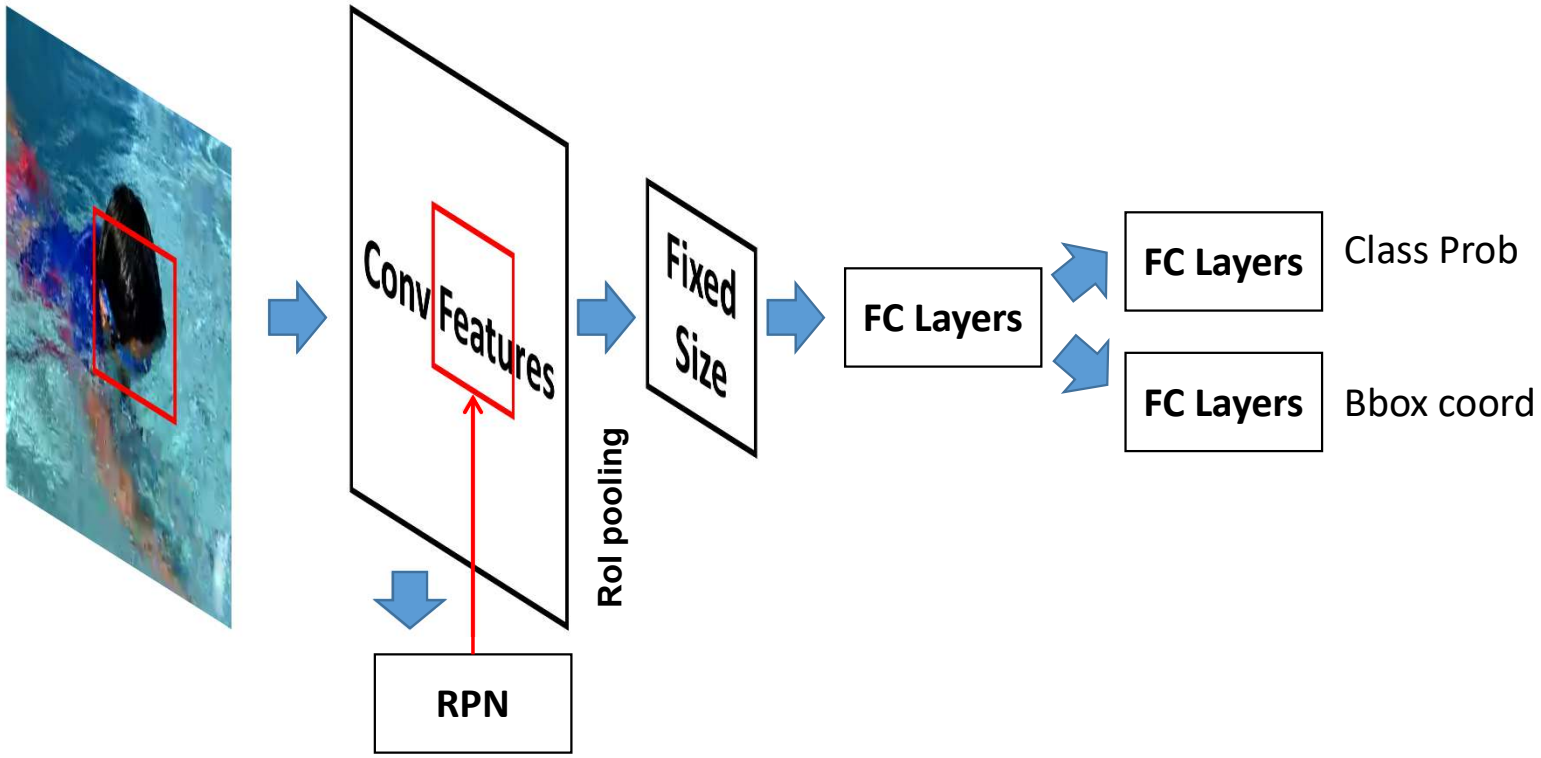


Faster R-CNN architecture



Source: "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, 2017

Classification and regression on region proposals



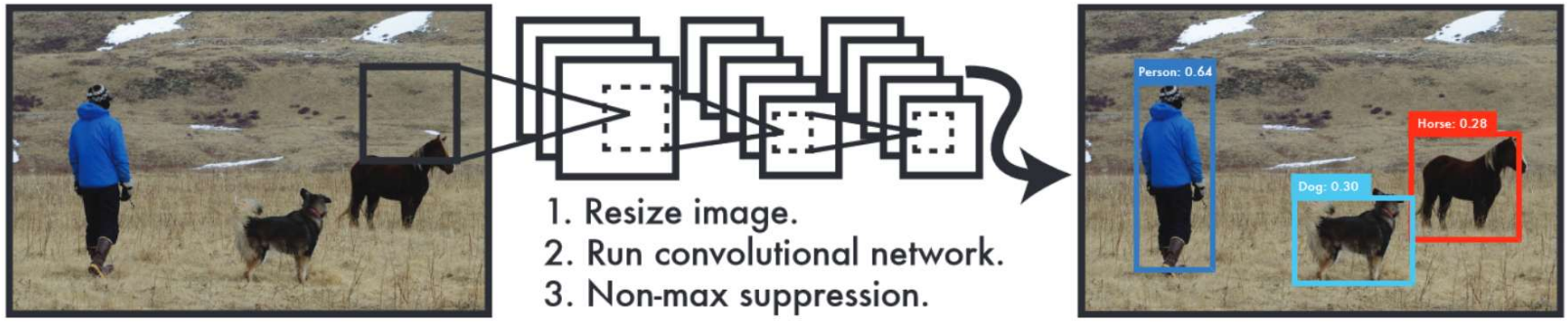
Case study: YOLO

- Problem: Simultaneous detection, localization, and classification of multiple objects in a fast manner

- Solution: Use one CNN to solve all tasks simultaneously

Source: “You Only Look Once:Unified, Real-Time Object Detection” Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi http://pjreddie.com/yolo/

Case study: YOLO



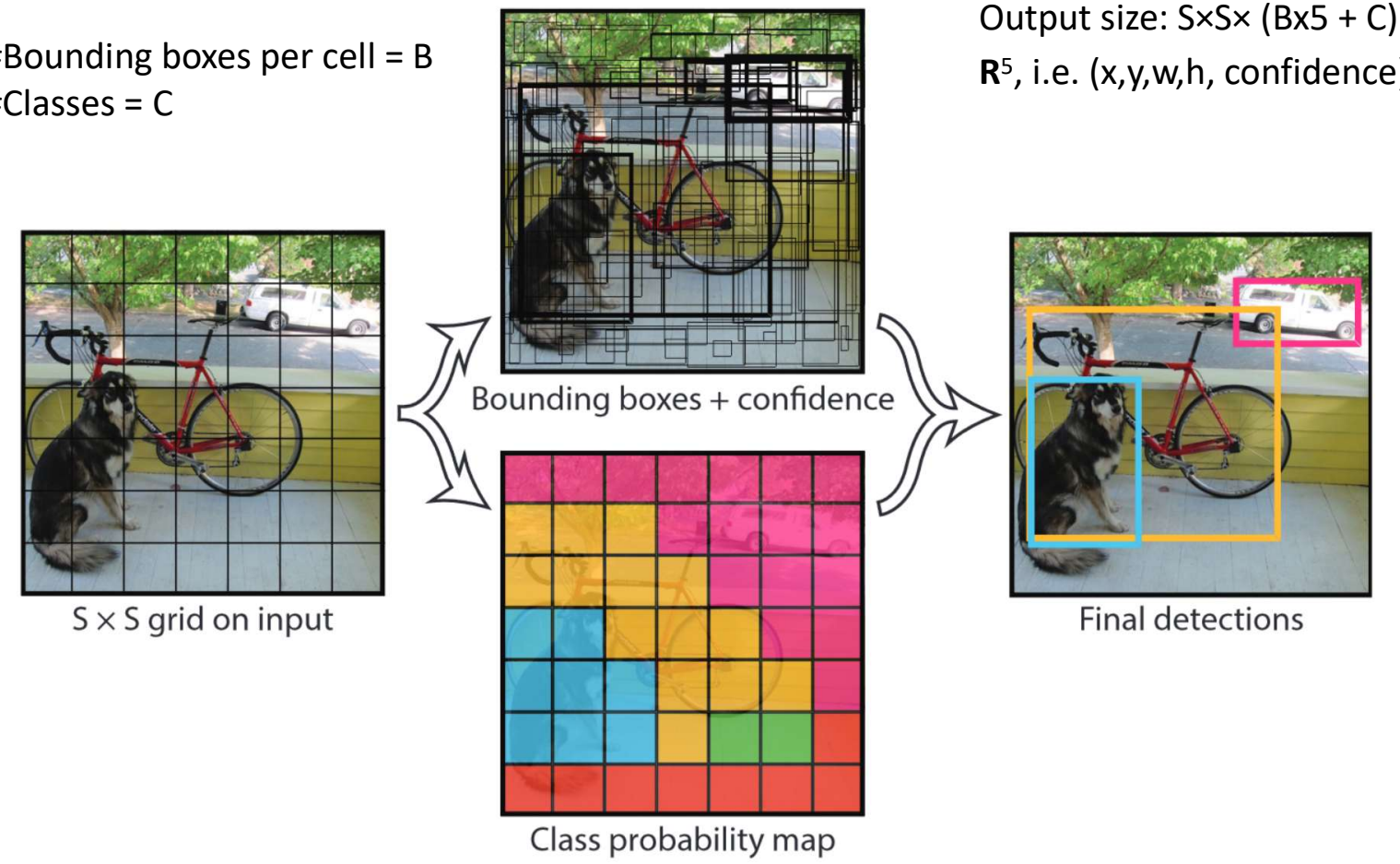
1. Resize the input image to 448×448,
2. Run a single convolutional network, and
3. Thresholds the resulting detections by the model's confidence.

Source: "You Only Look Once:Unified, Real-Time Object Detection" Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi http://pjreddie.com/yolo/

Case study: YOLO

#Bounding boxes per cell = B
#Classes = C

Output size: S×S× (Bx5 + C)
$\mathbf{R}^5$, i.e. (x,y,w,h, confidence)



Source: "You Only Look Once:Unified, Real-Time Object Detection" Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi http://pjreddie.com/yolo/

Case study: YOLO

loss function:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

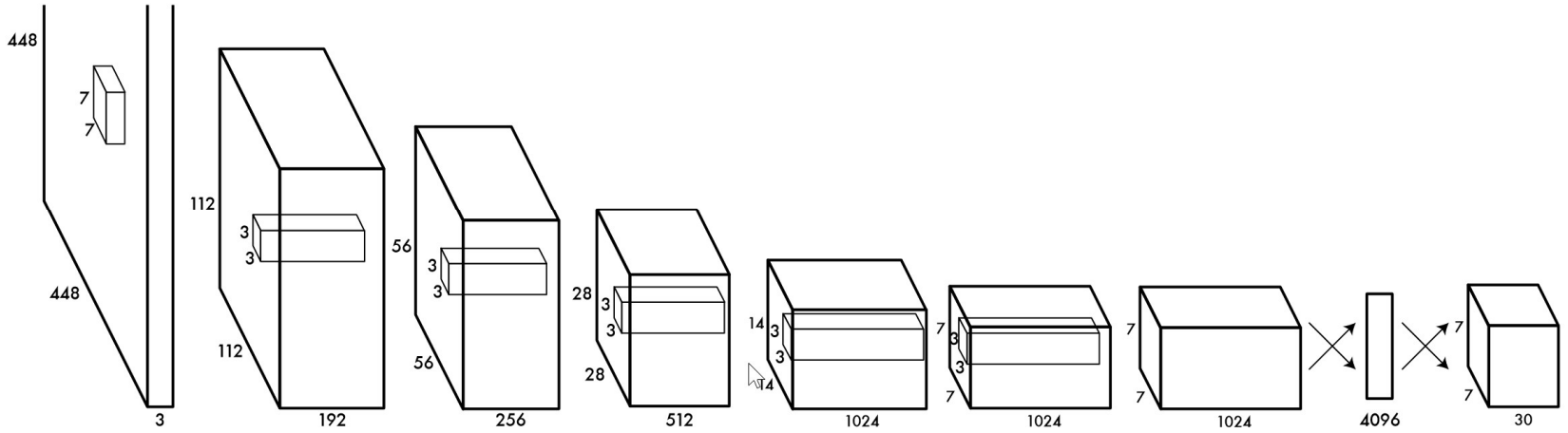
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Confidence C_i is Pr(Object) × IoU

Source: “You Only Look Once:Unified, Real-Time Object Detection” Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi http://pjreddie.com/yolo/

Case study: YOLO



Source: “You Only Look Once:Unified, Real-Time Object Detection” Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi http://pjreddie.com/yolo/