

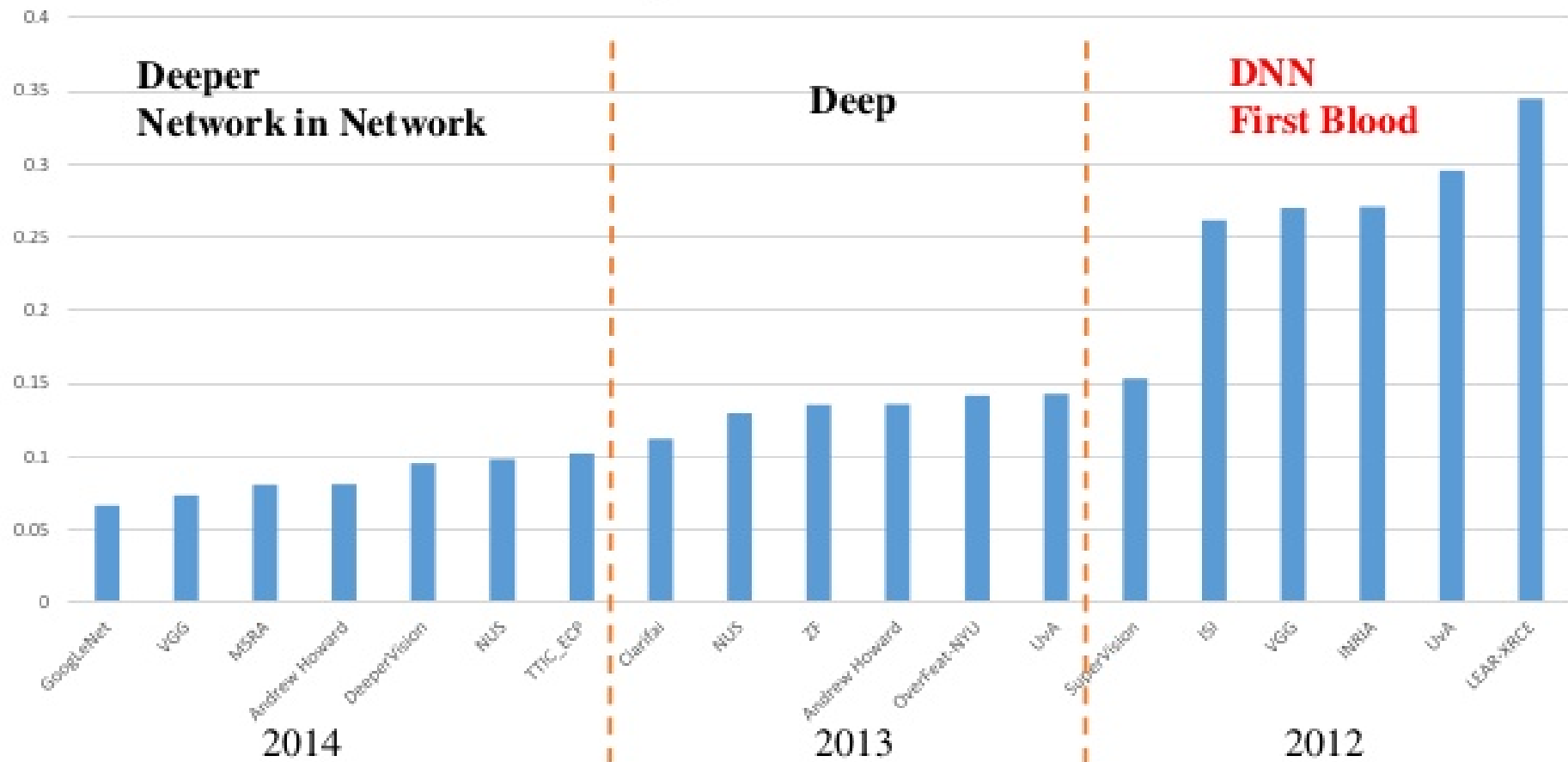
# Neural Networks

Jan Chorowski

# ImageNet Classification

- 1000 categories and 1.2 million training images

ImageNet Classification Error



Li Fei-Fei: ImageNet Large Scale Visual Recognition Challenge, 2014 <http://image-net.org/>



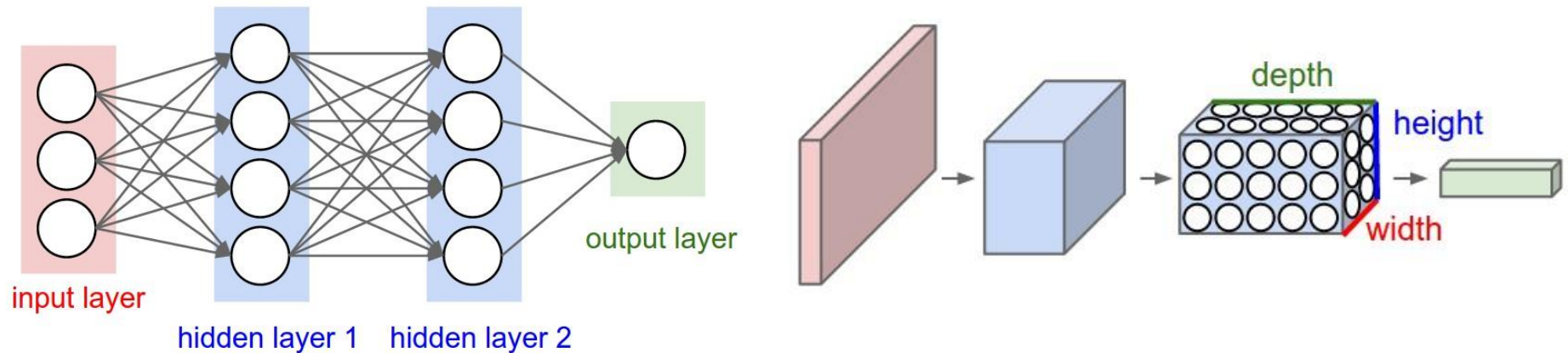
CONVNETS, CONVNETS  
EVERYWHERE

# Sharing neurons - convolutions

Note: material from <http://cs231n.github.io/convolutional-networks/>

In a conv net we use a different connection pattern between layers:

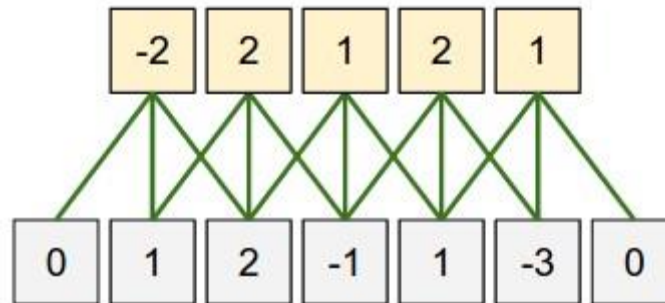
- Typically we use an all-to-all scheme
- In a conv-net we use local connectivity!



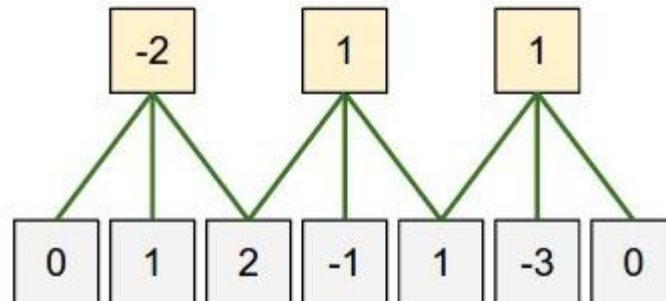
# 1D Convolution layer

- Small filter (neuron): 

1	0	-1
---	---	----
- Swipe the filter over the sequence

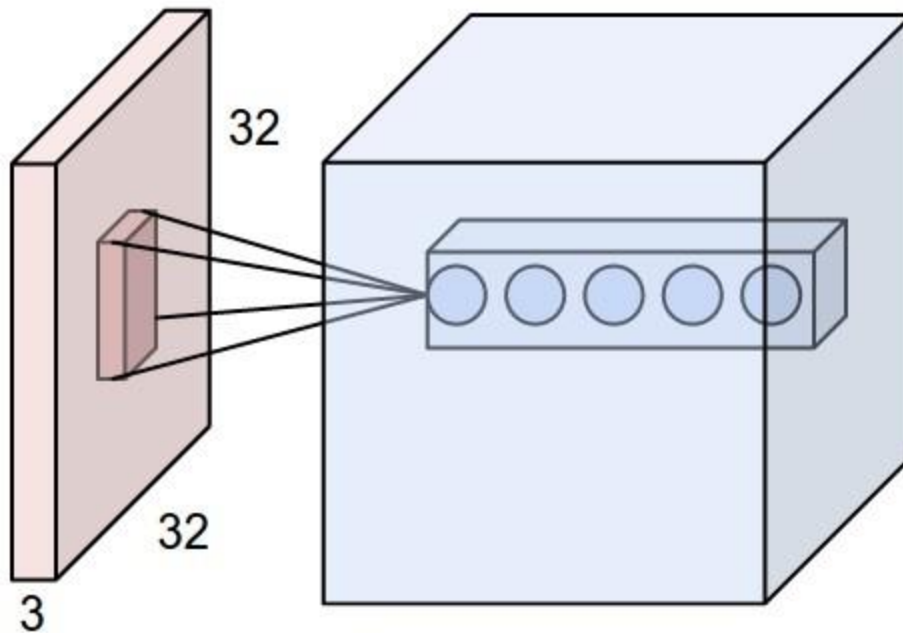


- Pool or stride (select only a few outputs)

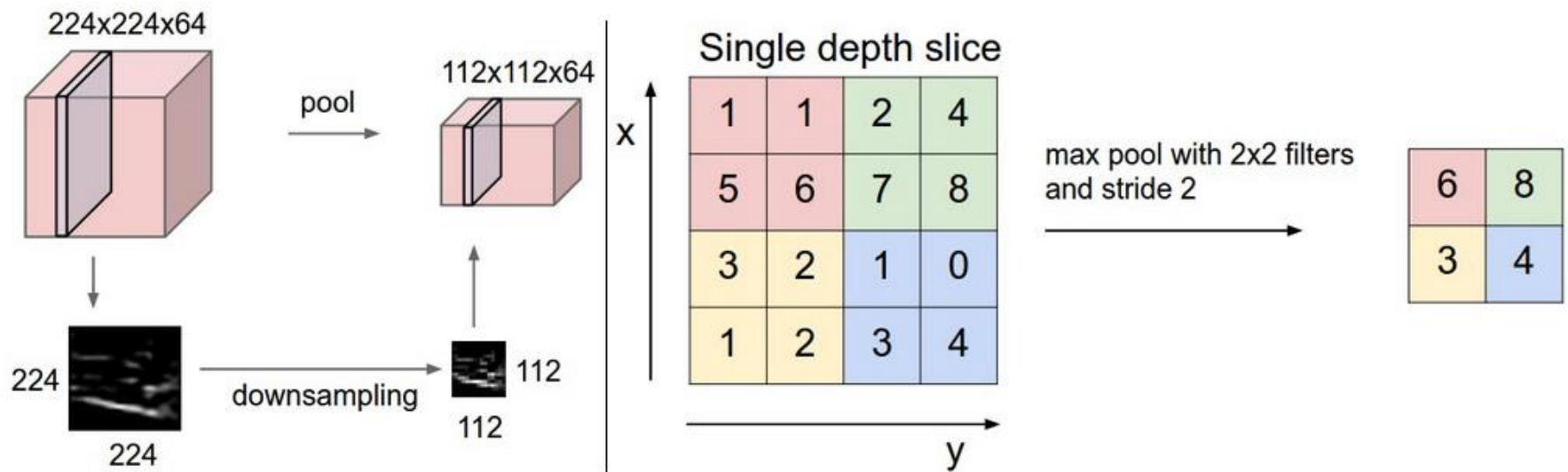


# 2D conv layer

- <http://cs231n.github.io/convolutional-networks/>



# Pooling

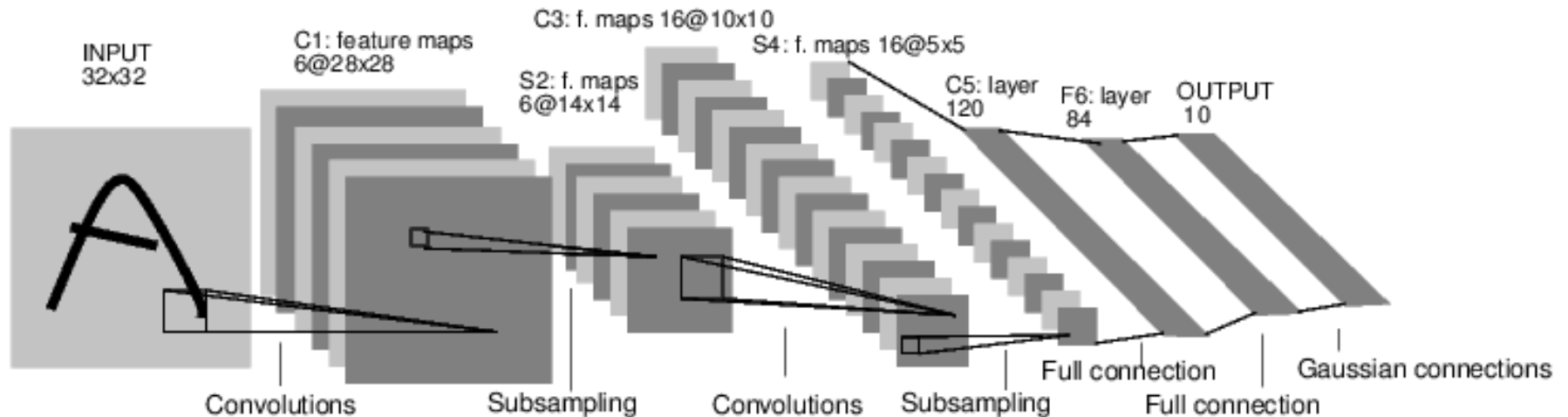


Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size  $[224 \times 224 \times 64]$  is pooled with filter size 2, stride 2 into output volume of size  $[112 \times 112 \times 64]$ . Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little  $2 \times 2$  square).

# Full network - LeNet

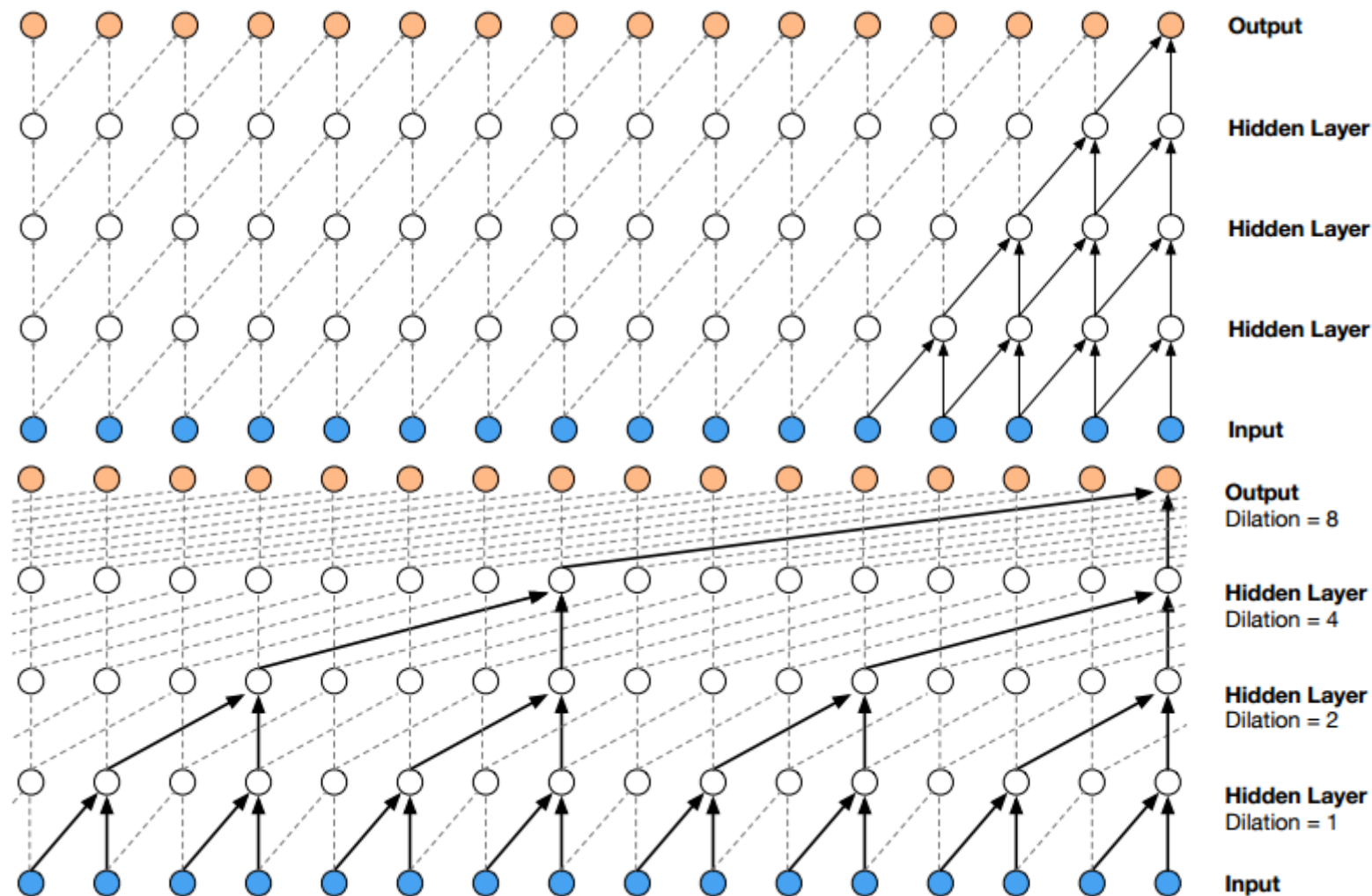
Y. LeCun et al. „Gradient-Based Learning Applied to Document Recognition”

<http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>





# Dilated convolutions: wide receptive field at high resolutions



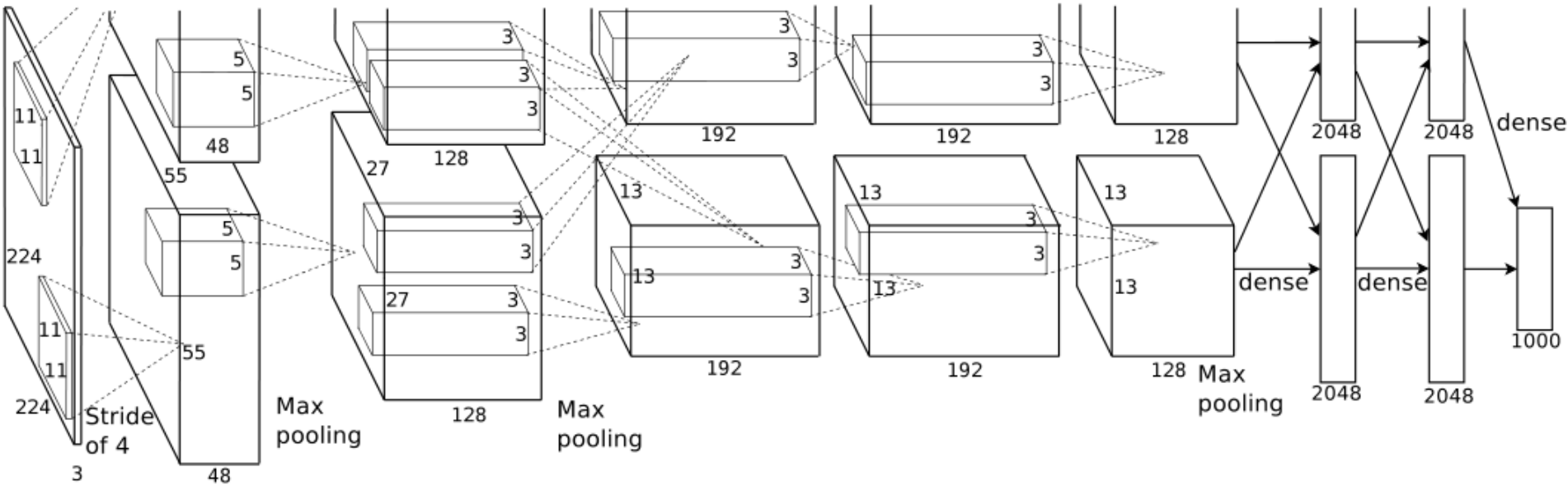
Wavenet: <https://arxiv.org/pdf/1609.03499.pdf>

<https://deepmind.com/blog/wavenet-generative-model-raw-audio/>

# Technicality: Edge padding

Convs can mimic FC layers

# AlexNet (2012)



# 60M parameters

The last dense layers takes:  $4096 * 4096 + 4096 * 1000 = 20\text{M}$  params!

# VGGNet: Network Design

## Key design choices:

- 3x3 conv. kernels – very small
- conv. stride 1 – no loss of information

## Other details:

- Rectification (ReLU) non-linearity
- 5 max-pool layers (x2 reduction)
- no normalisation
- 3 fully-connected (FC) layers

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

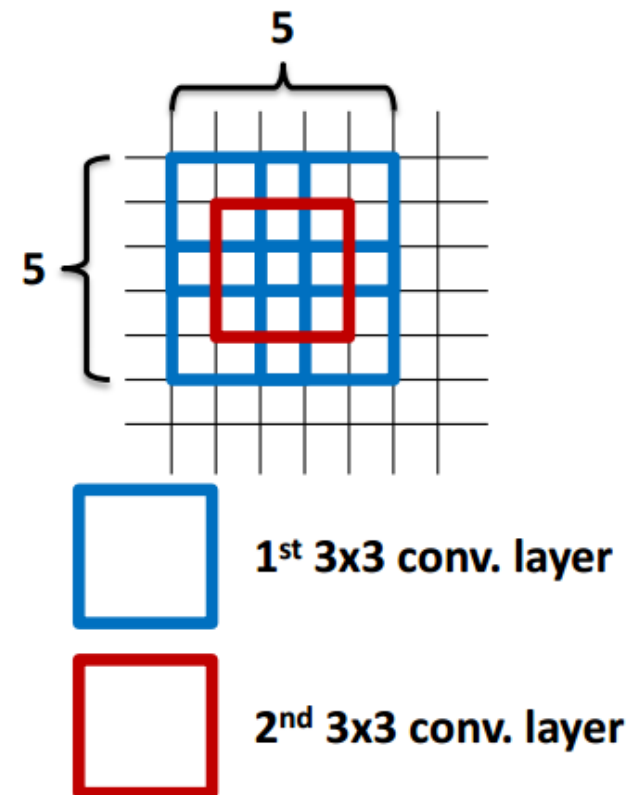
FC-1000

softmax

# VGGNet: Discussion

## Why 3x3 layers?

- Stacked conv. layers have a large receptive field
  - two 3x3 layers – 5x5 receptive field
  - three 3x3 layers – 7x7 receptive field
- More non-linearity
- Less parameters to learn
  - ~140M per net



# VGGNet (2014)

INPUT: [224x224x3] memory:  $224*224*3=150K$  weights: 0

CONV3-64: [224x224x64] memory:  $224*224*64=3.2M$  weights:  $(3*3*3)*64 = 1,728$

CONV3-64: [224x224x64] memory:  $224*224*64=3.2M$  weights:  $(3*3*64)*64 = 36,864$

POOL2: [112x112x64] memory:  $112*112*64=800K$  weights: 0

CONV3-128: [112x112x128] memory:  $112*112*128=1.6M$  weights:  $(3*3*64)*128 = 73,728$

CONV3-128: [112x112x128] memory:  $112*112*128=1.6M$  weights:  $(3*3*128)*128 = 147,456$

POOL2: [56x56x128] memory:  $56*56*128=400K$  weights: 0

CONV3-256: [56x56x256] memory:  $56*56*256=800K$  weights:  $(3*3*128)*256 = 294,912$

CONV3-256: [56x56x256] memory:  $56*56*256=800K$  weights:  $(3*3*256)*256 = 589,824$

CONV3-256: [56x56x256] memory:  $56*56*256=800K$  weights:  $(3*3*256)*256 = 589,824$

POOL2: [28x28x256] memory:  $28*28*256=200K$  weights: 0

CONV3-512: [28x28x512] memory:  $28*28*512=400K$  weights:  $(3*3*256)*512 = 1,179,648$

CONV3-512: [28x28x512] memory:  $28*28*512=400K$  weights:  $(3*3*512)*512 = 2,359,296$

CONV3-512: [28x28x512] memory:  $28*28*512=400K$  weights:  $(3*3*512)*512 = 2,359,296$

POOL2: [14x14x512] memory:  $14*14*512=100K$  weights: 0

CONV3-512: [14x14x512] memory:  $14*14*512=100K$  weights:  $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory:  $14*14*512=100K$  weights:  $(3*3*512)*512 = 2,359,296$

CONV3-512: [14x14x512] memory:  $14*14*512=100K$  weights:  $(3*3*512)*512 = 2,359,296$

POOL2: [7x7x512] memory:  $7*7*512=25K$  weights: 0

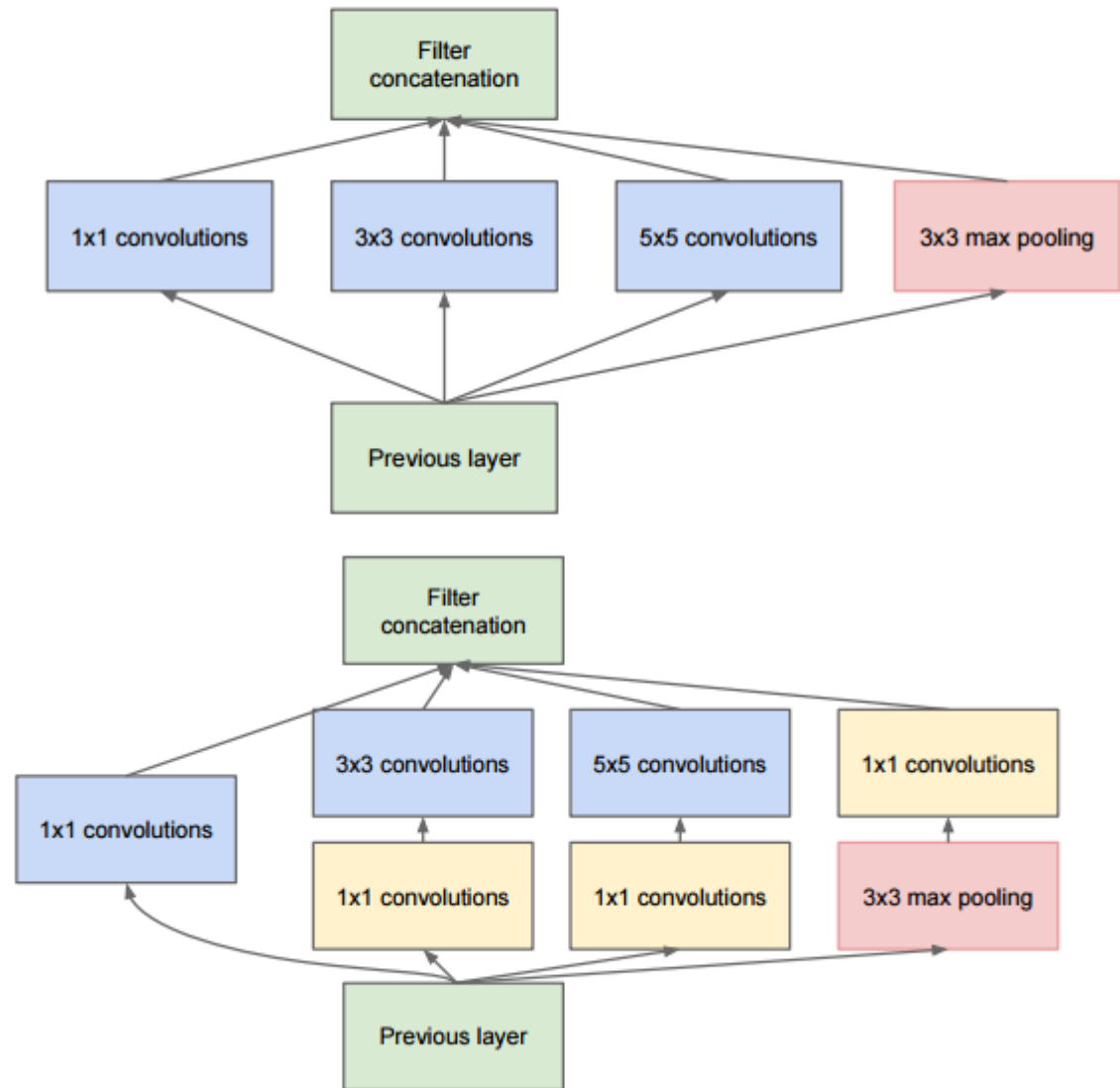
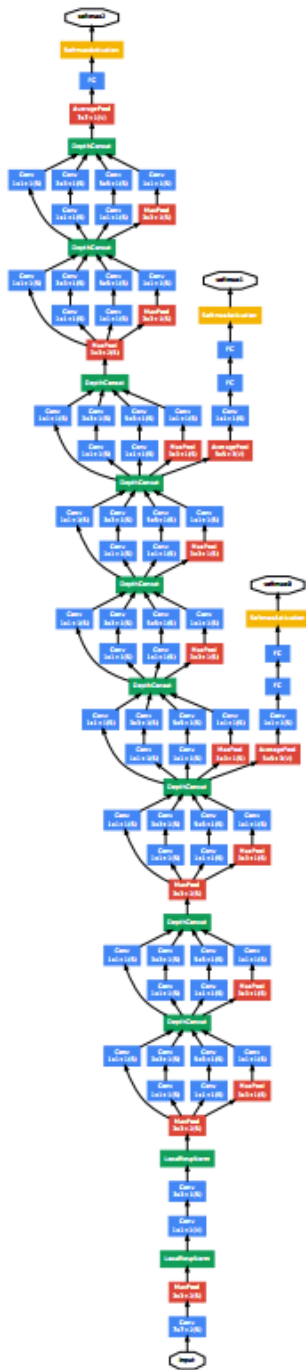
FC: [1x1x4096] memory: 4096 weights:  $7*7*512*4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 weights:  $4096*4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 weights:  $4096*1000 = 4,096,000$

TOTAL memory:  $24M * 4 \text{ bytes} \sim 93MB$  / image (only forward!  $\sim 2$  for bwd) TOTAL params: 138M parameters

# GoogLeNet (Inception) 2014





# Separable Filters

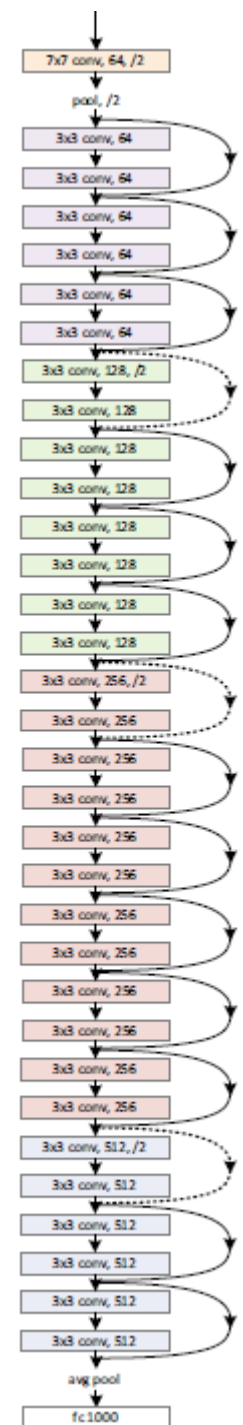
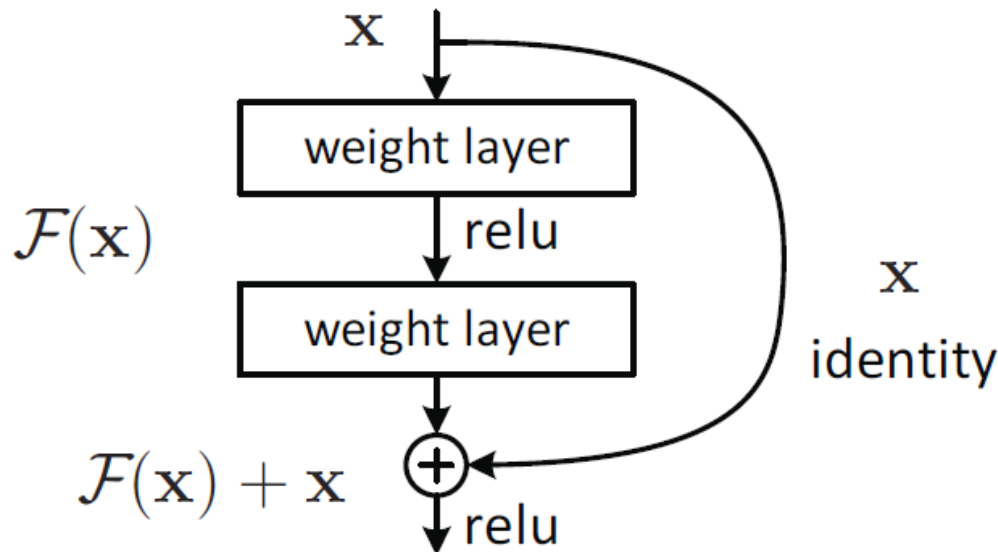
- Sometimes a 2D conv == 2\* 1D conv:

$$\frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * \frac{1}{3} [1 \quad 1 \quad 1] = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \mathbf{G}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * [+1 \quad 0 \quad -1] * A$$

- Inception and VGGnet use approximate separation, with more nonlinearity

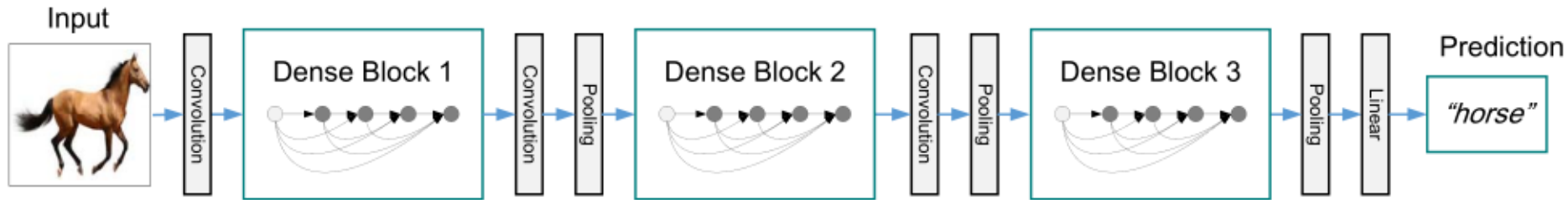
# ResNet (Residual Nets, 2015)

- Add bypasses to ease gradient flow
- Networks with more than 150 layers!
- 3x3 convs with number of feature maps doubling after every pooling

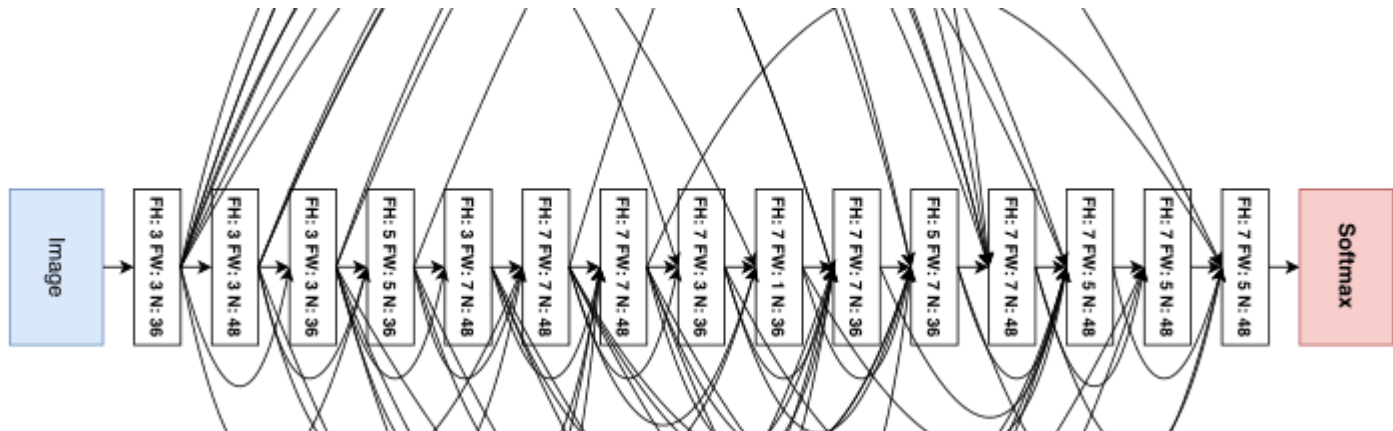


# Quest for models

- DenseNets (<https://arxiv.org/pdf/1608.06993>)



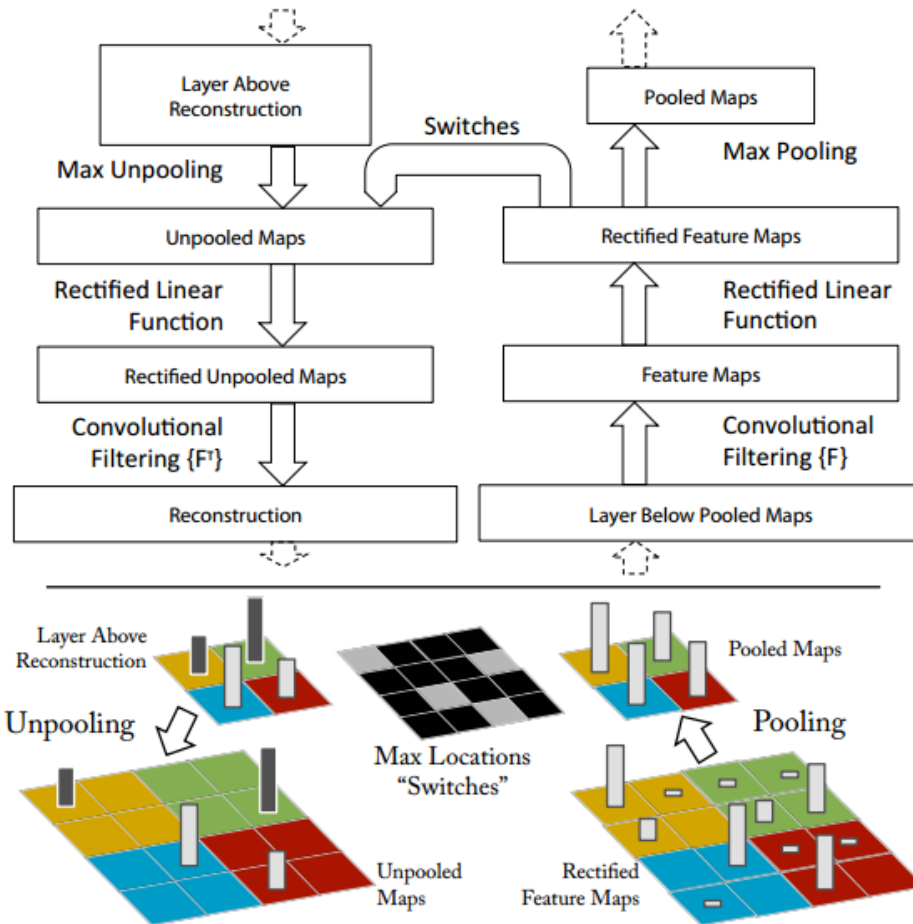
- Automatic architecture search (<https://arxiv.org/pdf/1611.01578>)



# Which ConvNet to use?

- Take current best on Imagenet
- Many networks are available for download (google tensorflow model and caffe model zoo)
- For your use – take a pretrained net and retrain only the last layers!
- On ImageNet (1M images) data augmentation and regularization is as important as the network!

# Visualizing CNN features

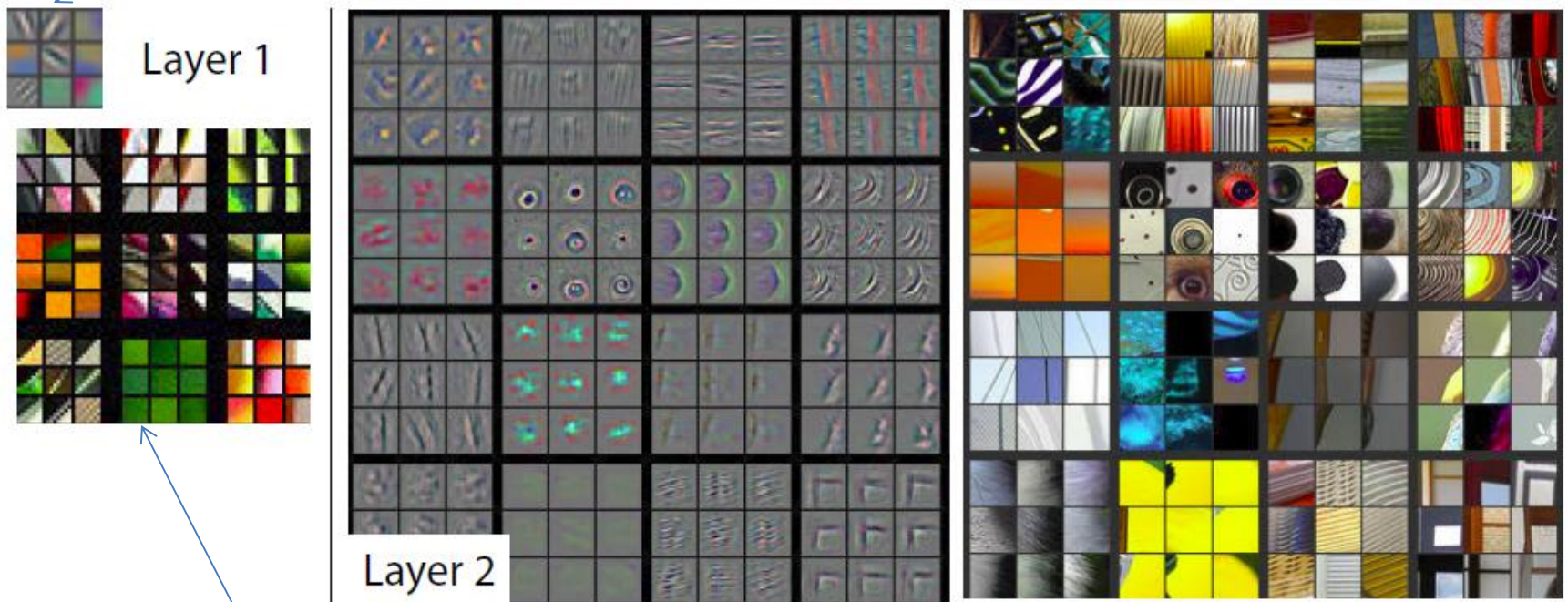


Main idea:

Do a forward propagation,  
Save state of nonlinearities,  
Do a full transposed convolution  
(deconvolution) – kind of like inverse  
convolution

# Low-level features

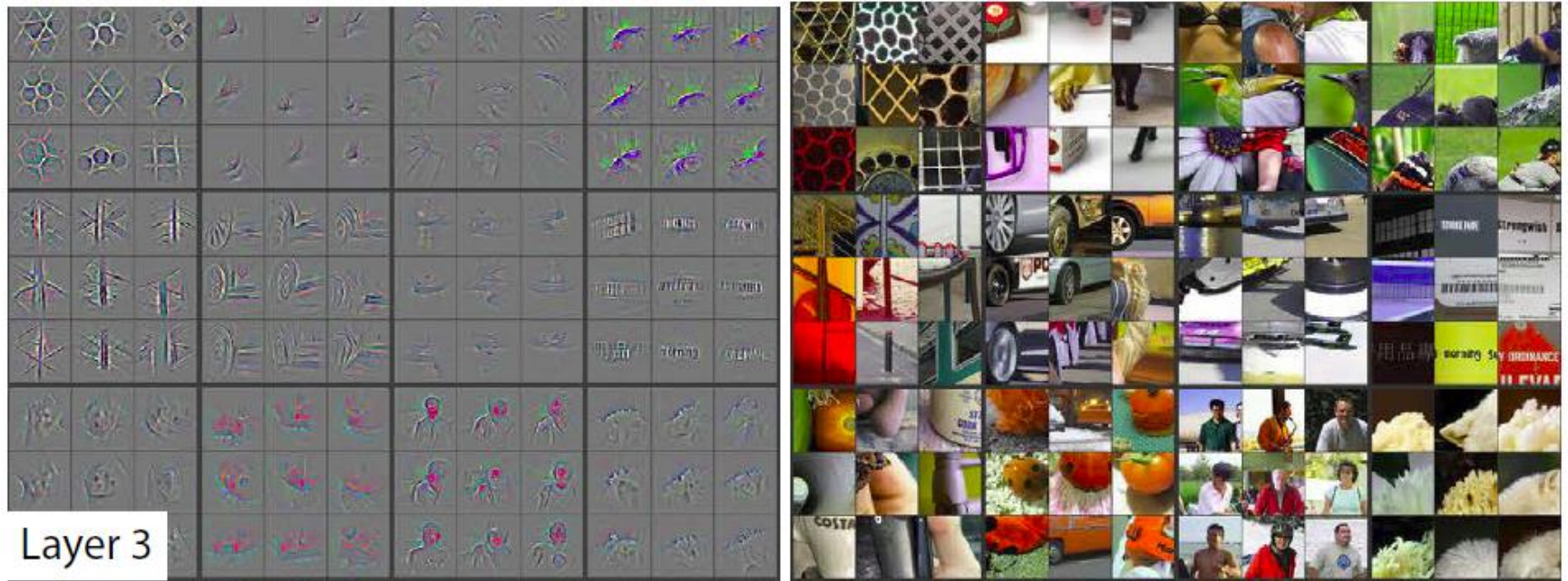
What the neuron (feature-detector looks for)



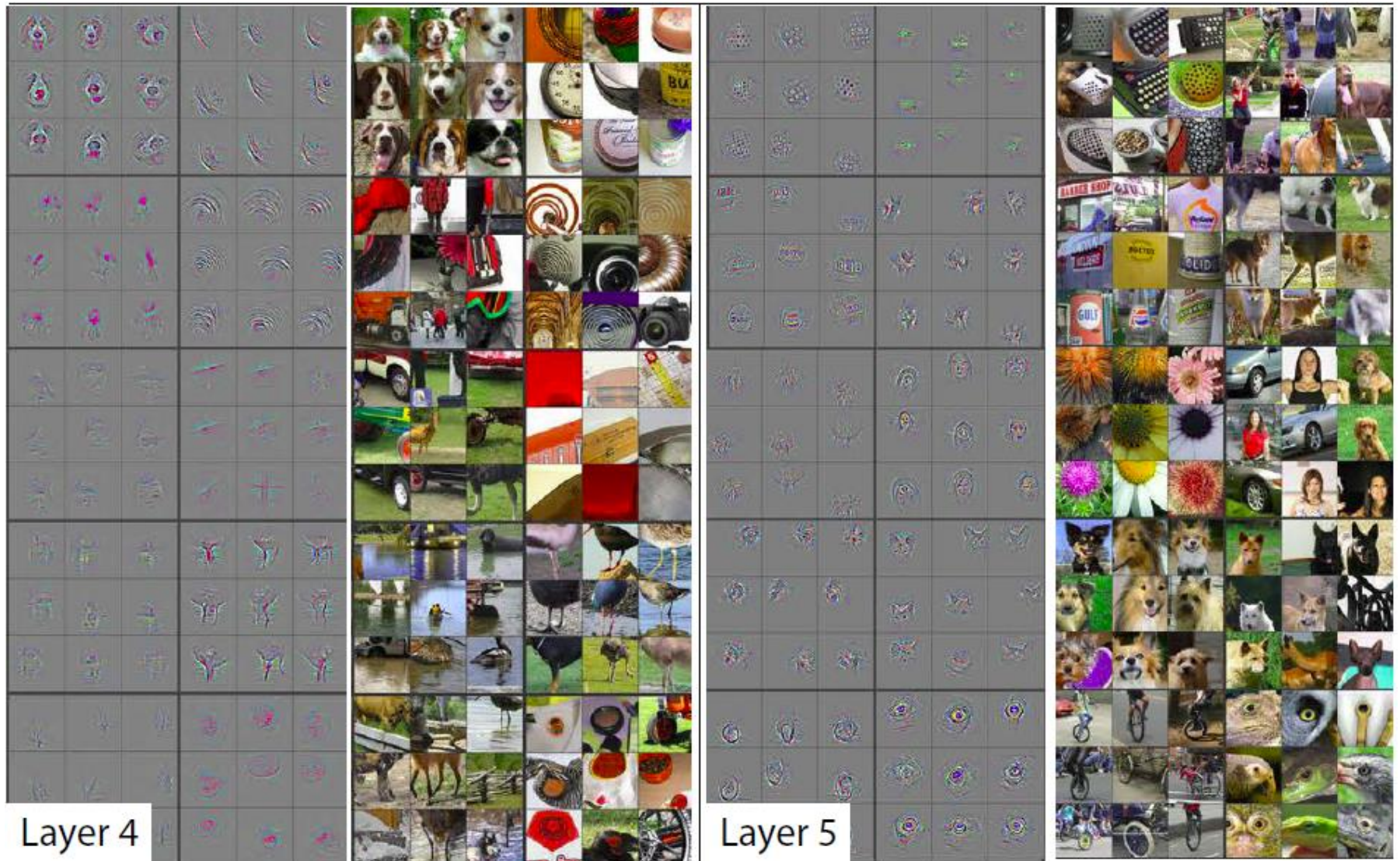
What images are selected by the neuron



# Mid-level features

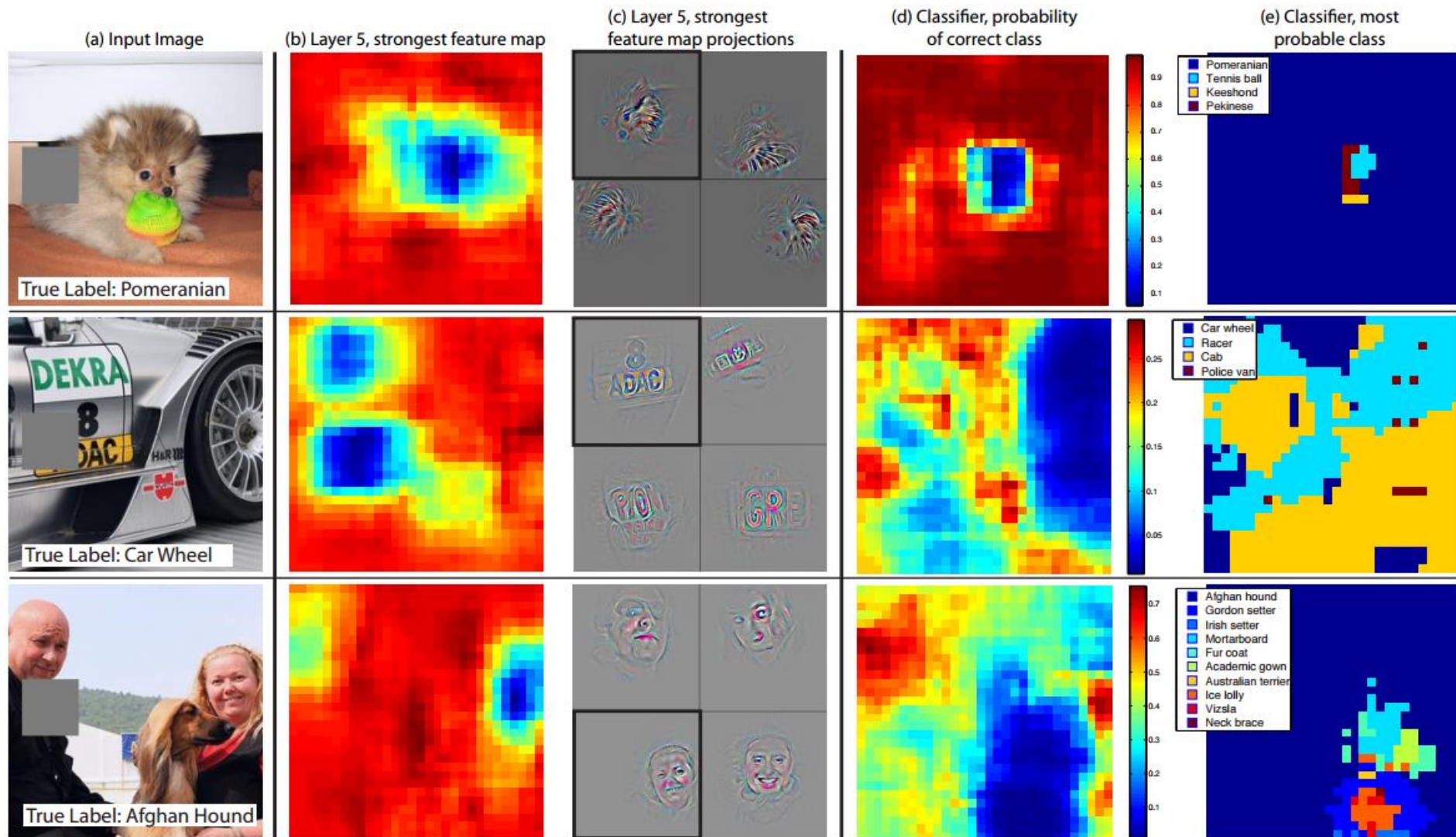


# High-level features





# Where Convnets look?



# Style Transfer



Find image (backpropagation toward pixels) minimizing:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_c + \mathcal{L}_s = \\ &= (F(C) - F(I))^2 + (G(C) - G(I))^2\end{aligned}$$

Where:

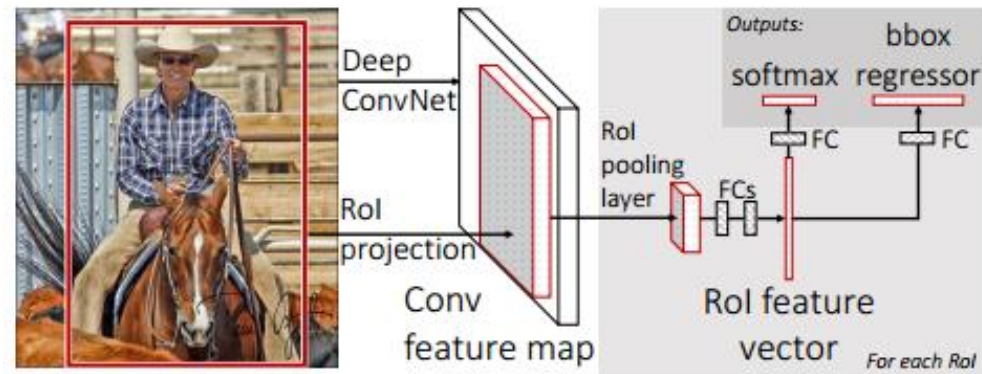
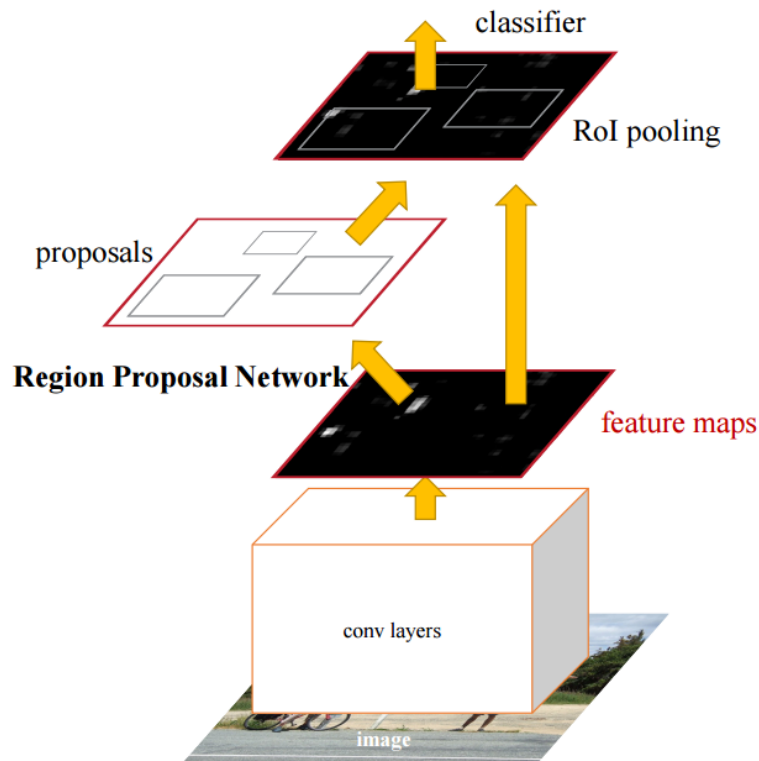
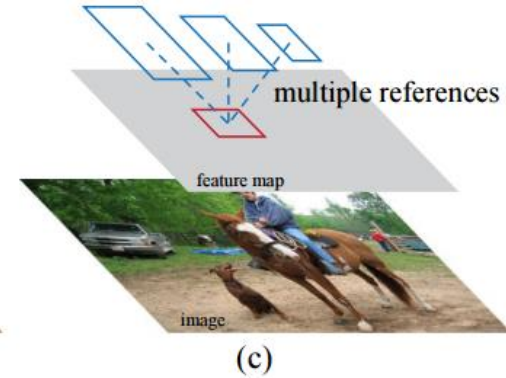
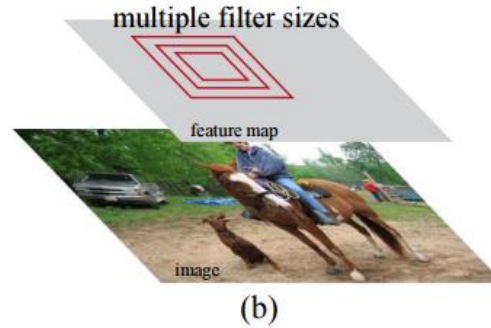
$F(x)$  features of a CNN layer on image  $x$

$G(x)$  matrix of correlations between a layer's features over pixels of image  $x$

# Object detection and segmentation

Fast R-CNN (<https://arxiv.org/pdf/1504.08083.pdf>)

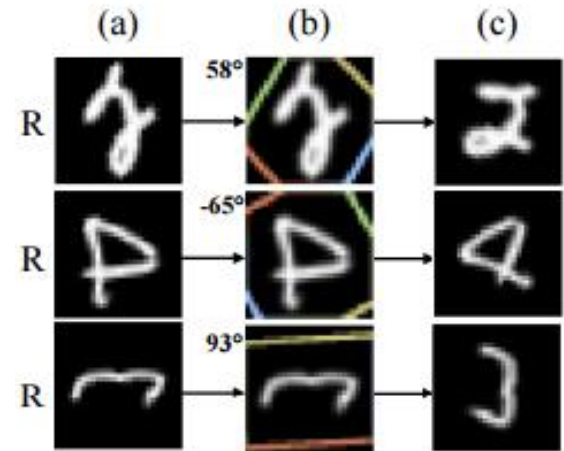
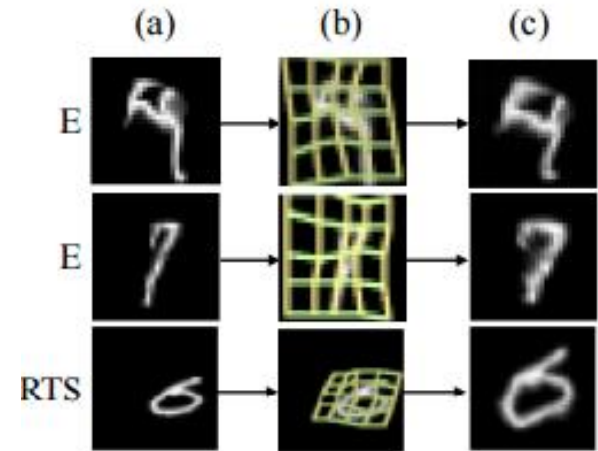
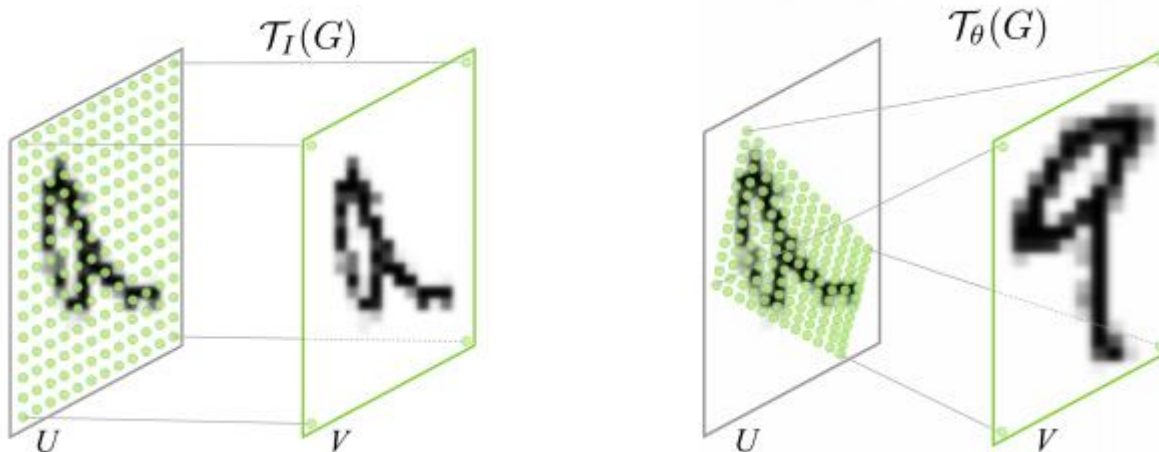
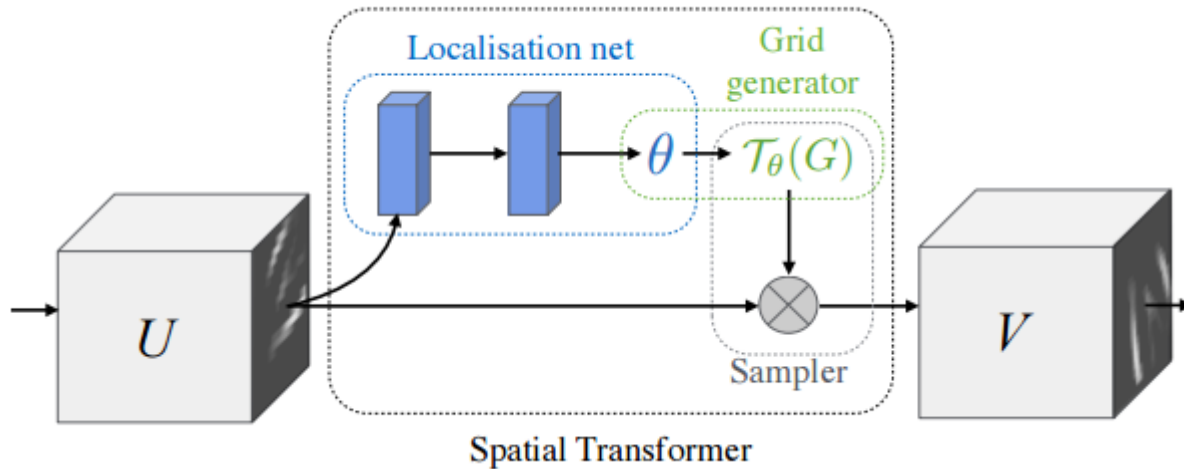
Faster R-CNN (<https://arxiv.org/pdf/1506.01497.pdf>)





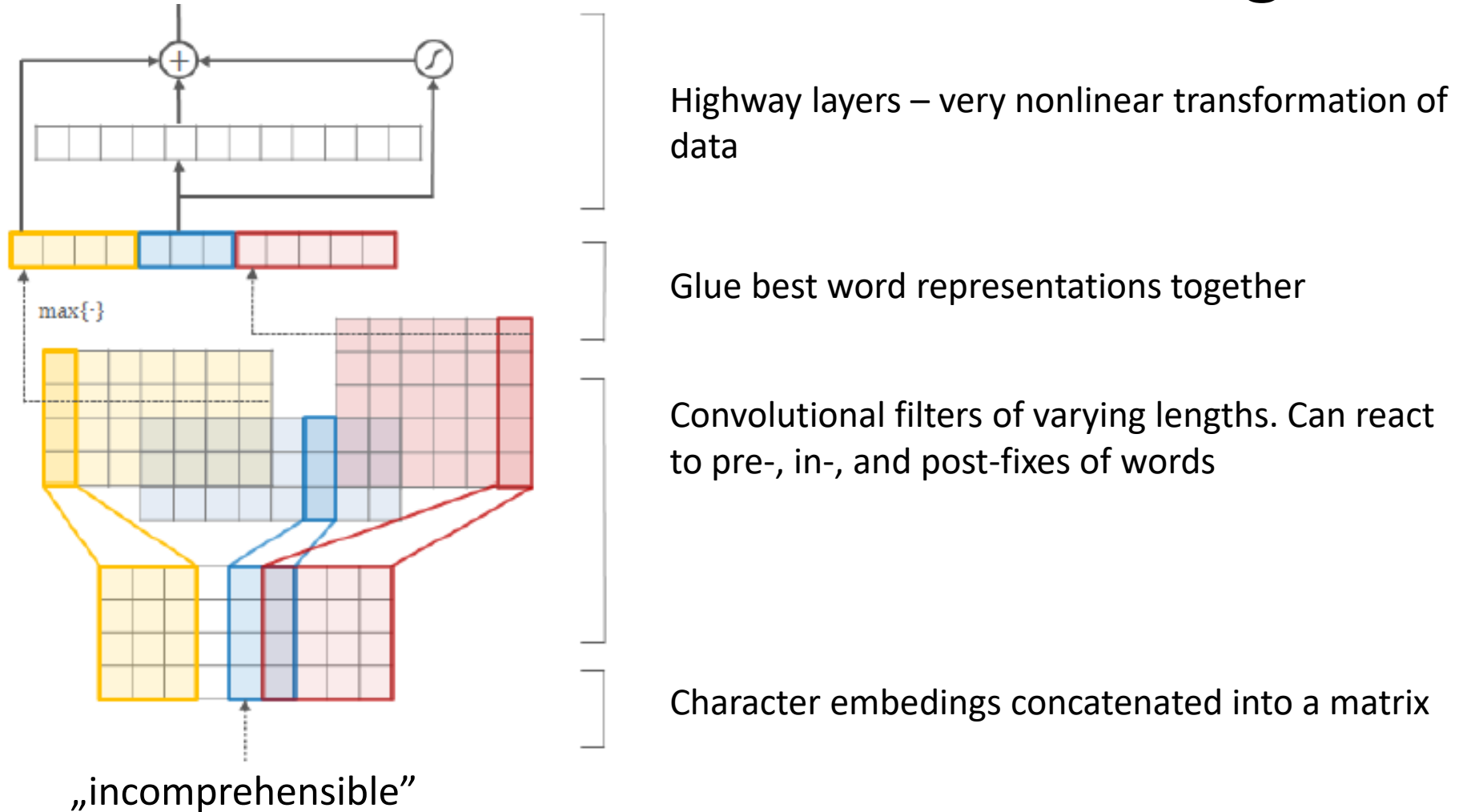
# Spatial Transformer Networks

<https://arxiv.org/pdf/1506.02025.pdf>



# CNN beyond images

## Or character-to-word embedding



Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, “Character-Aware Neural Language Models,” *arXiv:1508.06615 [cs, stat]*, Aug. 2015.

# Additional references

- <http://cs231n.github.io>
- [How transferable are features in deep neural networks?](#) studies the transfer learning performance in detail, including some unintuitive findings about layer co-adaptations.
- [CNN Features off-the-shelf: an Astounding Baseline for Recognition](#) trains SVMs on features from ImageNet-pretrained ConvNet and reports several state of the art results.