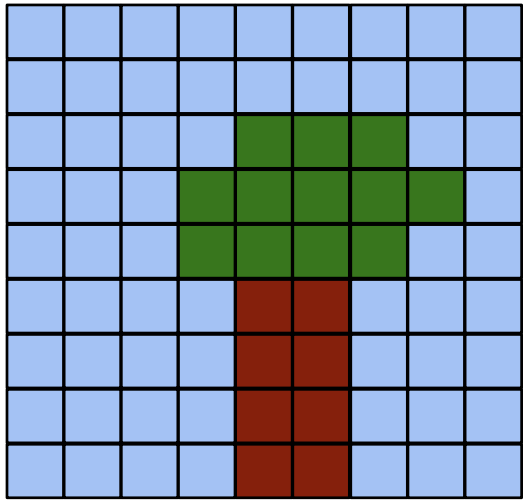
Convolutional Neural Networks – Building blocks

Deep Learning & Applications

Neural network with images

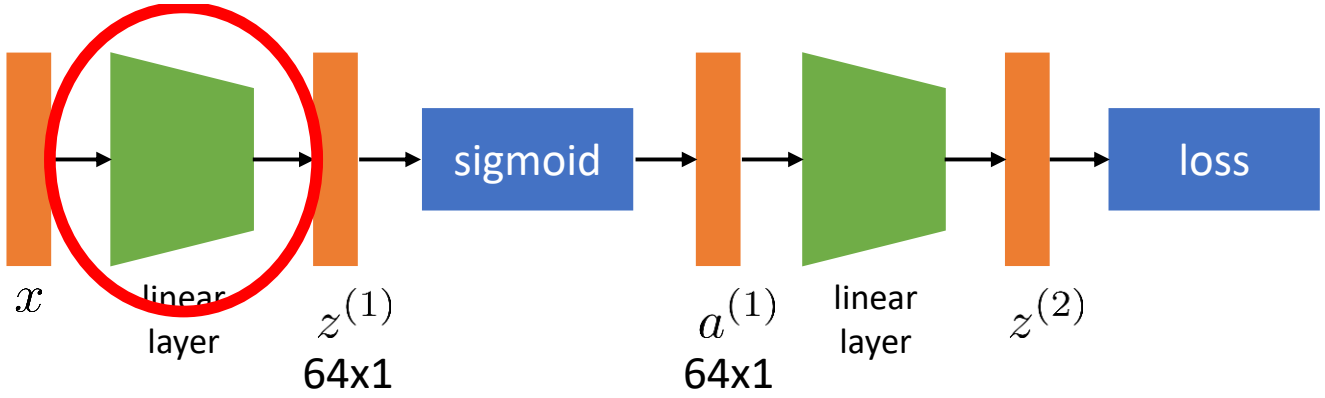
A digital image is a 2D **grid of pixels**.

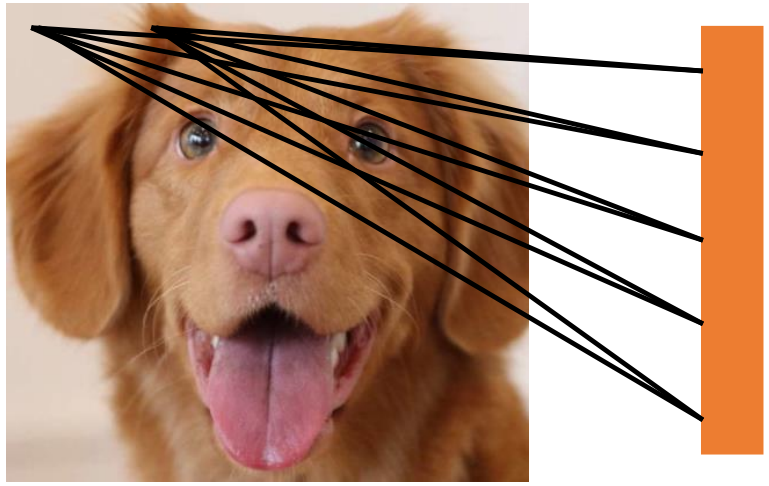


A neural network expects a **vector of numbers** as input.

Neural network with images

[object label]





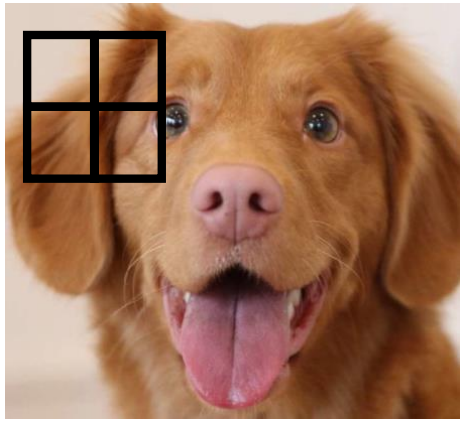
$$\text{image is } 128 \times 128 \times 3 = 49{,}152$$

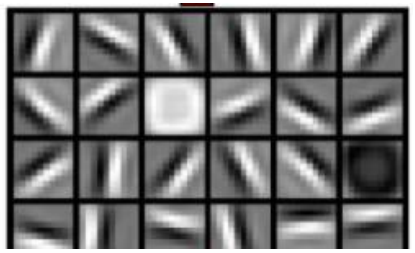
$$z^{(1)} \text{ is 64-dim}$$

$$64 \times 49,152 \approx 3,000,000$$

We need a better way!

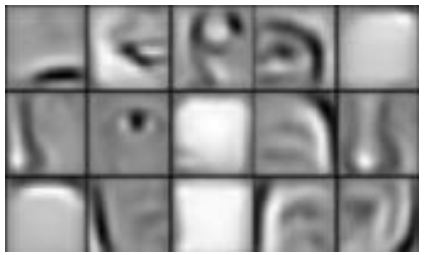
An idea…





Layer 1:

edge detectors?



Layer 2:

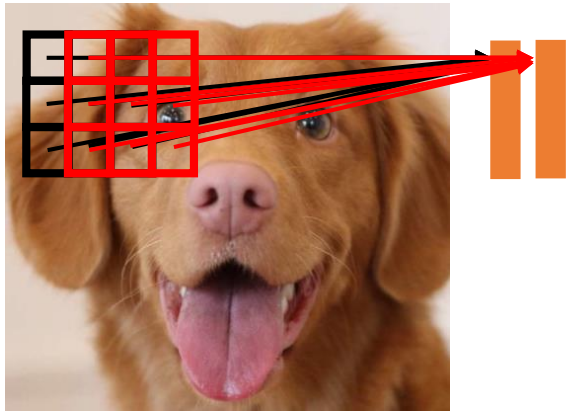
ears? noses?

Observation: many useful image features are **local**

to tell if a particular patch of image contains a feature, enough to look at the local patch

An idea…

Observation: many useful image features are **local**

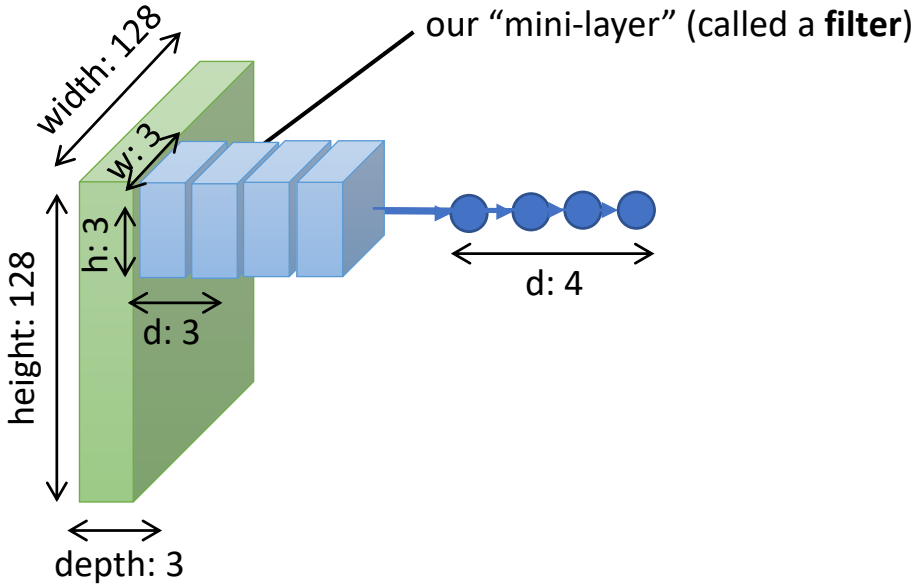


$$\text{patch is } 3 \times 3 \times 3 = 27$$

$$z^{(1)} \text{ is 64-dim}$$

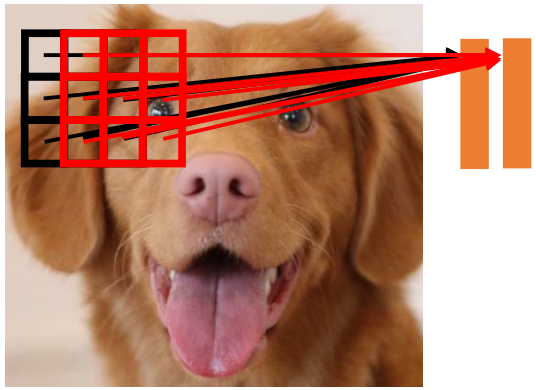
$$64 \times 27 = 1728$$

We get a **different** output at each image location!



An idea…

Observation: many useful image features are **local**

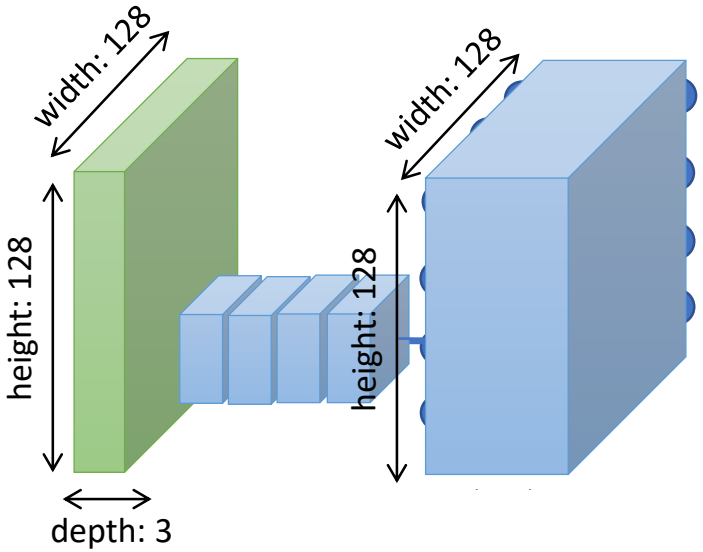


$$\text{patch is } 3 \times 3 \times 3 = 27$$

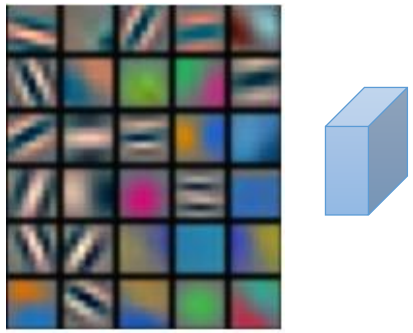
$$z^{(1)} \text{ is 64-dim}$$

$$64 \times 27 = 1728$$

We get a **different** output at each image location!

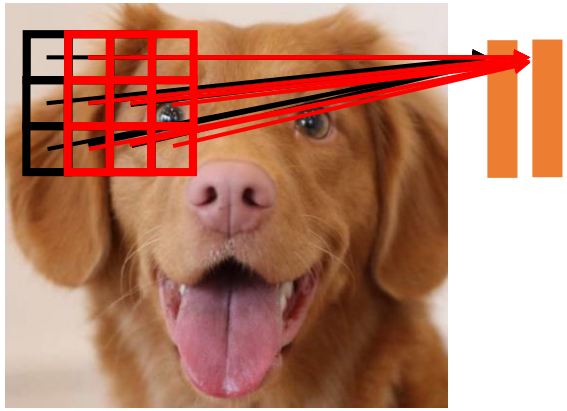


What do they look like?



An idea…

Observation: many useful image features are **local**

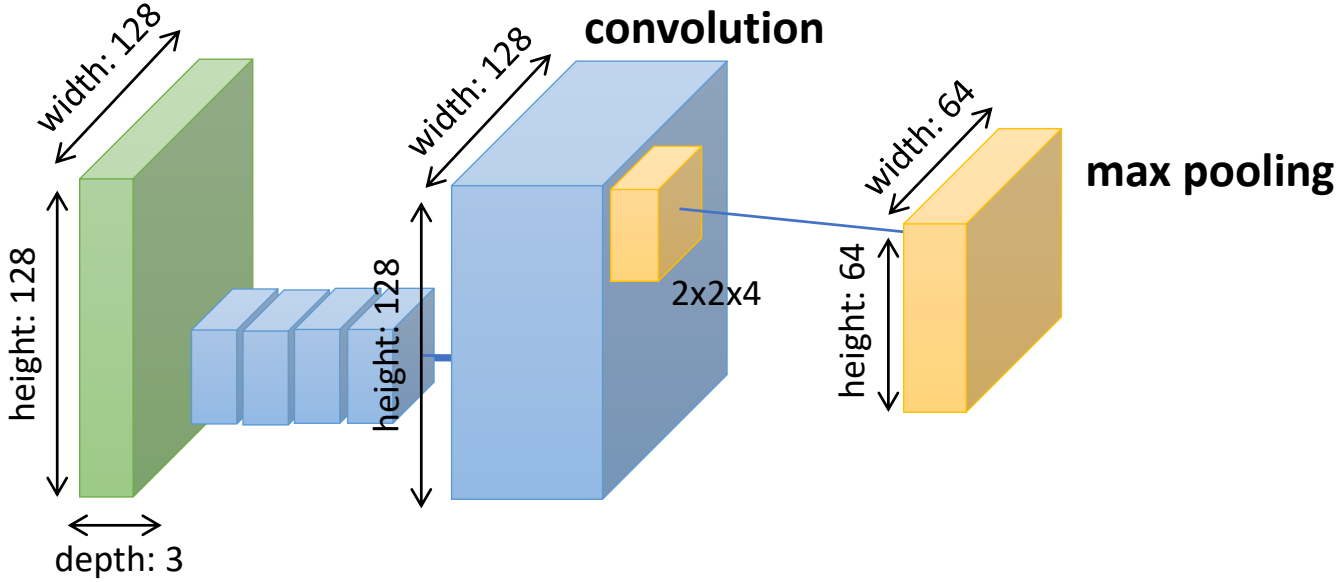


$$\text{patch is } 3 \times 3 \times 3 = 27$$

$$z^{(1)} \text{ is } 64\text{-dim}$$

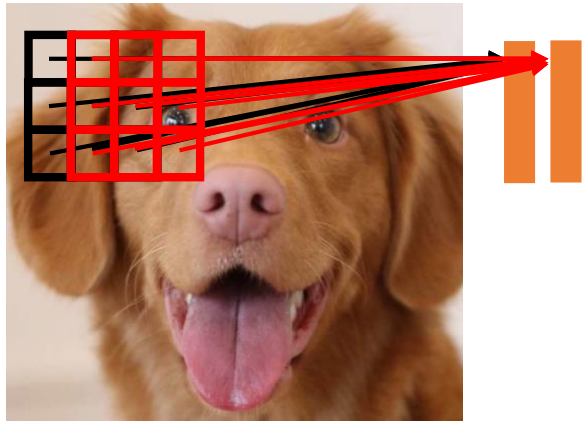
$$64 \times 27 = 1728$$

We get a **different** output at each image location!



An idea…

Observation: many useful image features are **local**

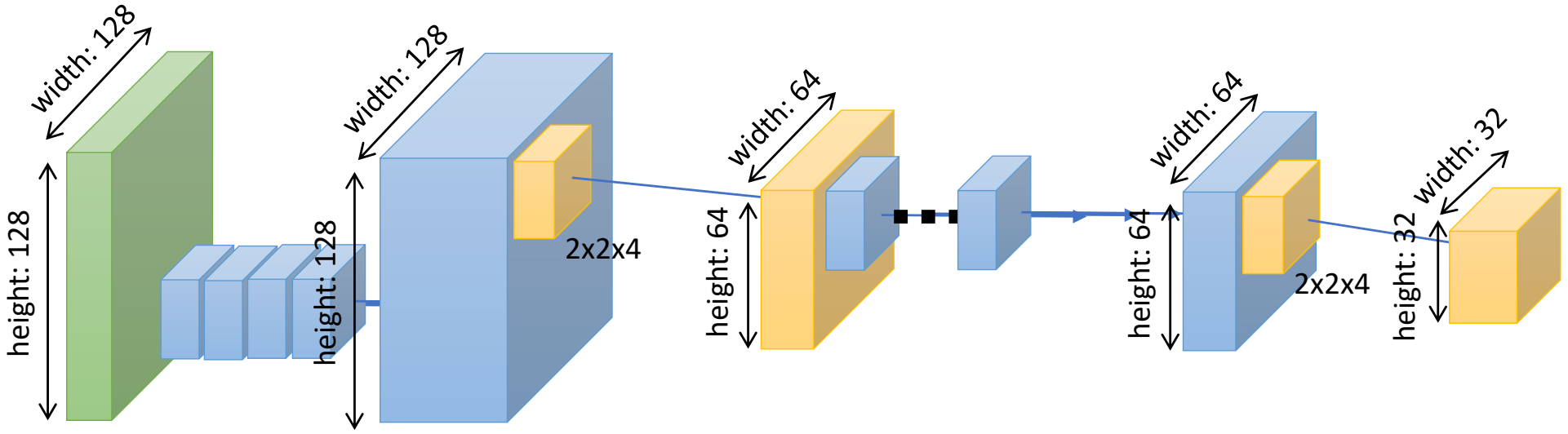


$$\text{patch is } 3 \times 3 \times 3 = 27$$

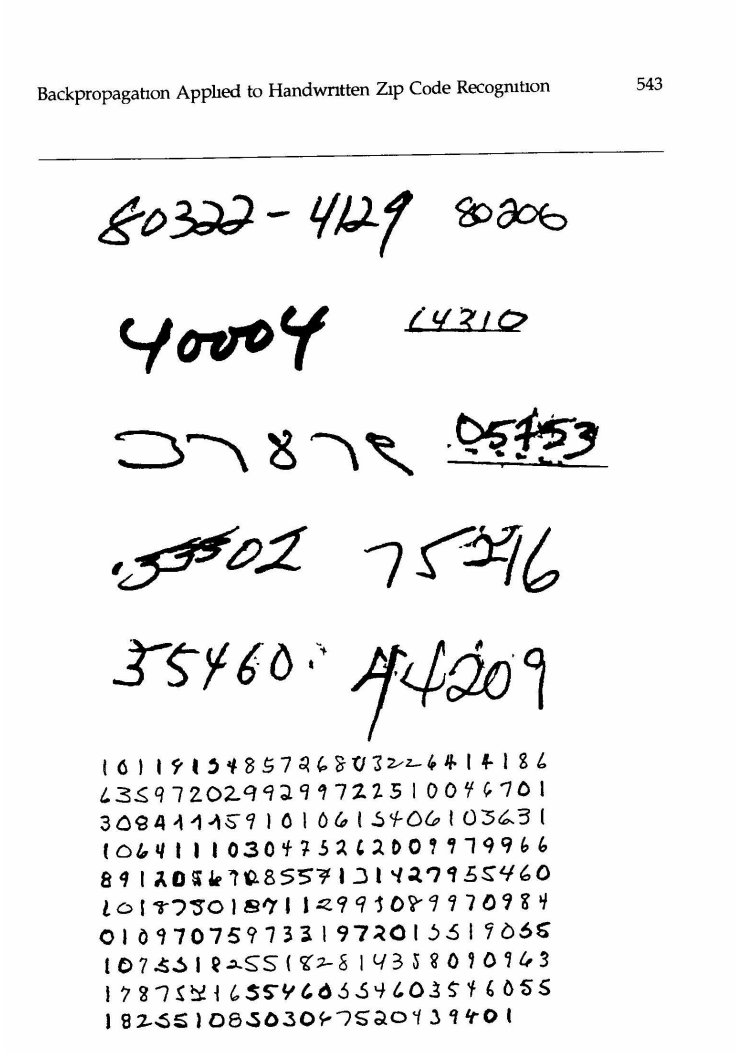
$$z^{(1)} \text{ is 64-dim}$$

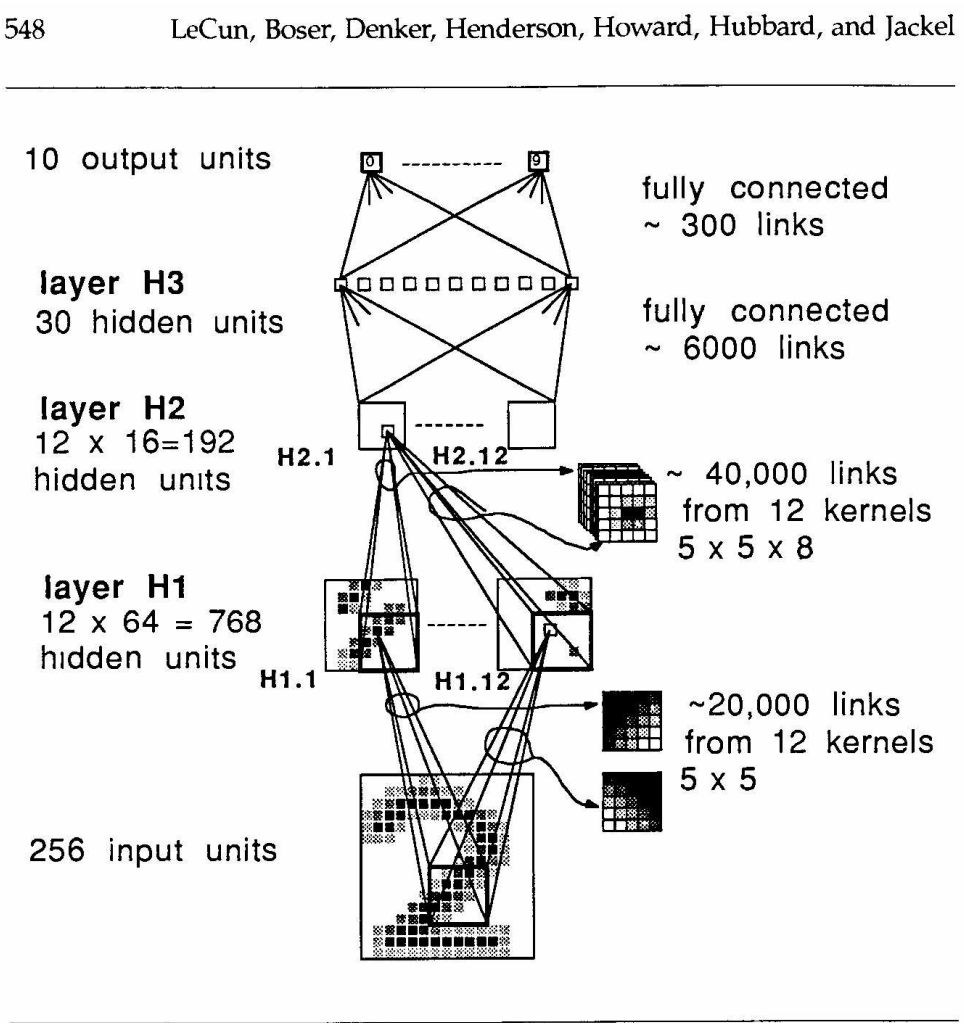
$$64 \times 27 = 1728$$

We get a **different** output at each image location!



Convolutional Neural Networks





Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

Convolutional Neural Networks

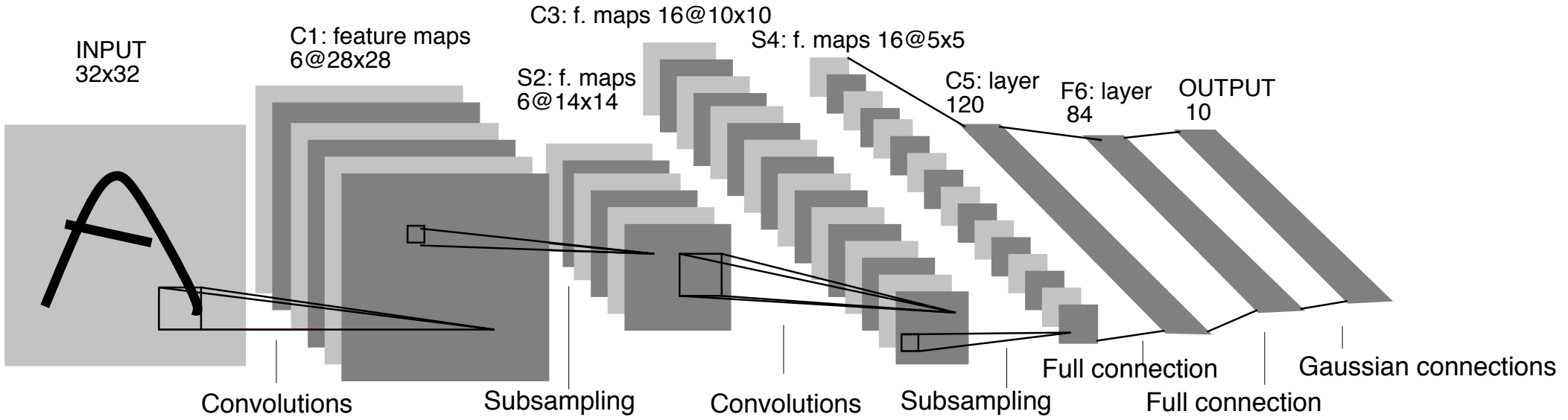
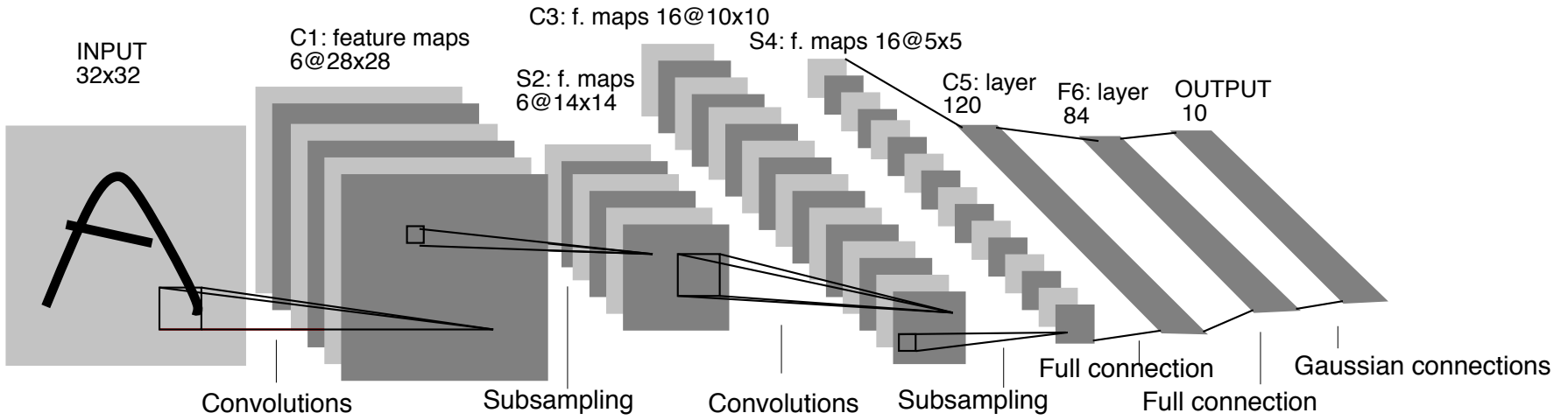
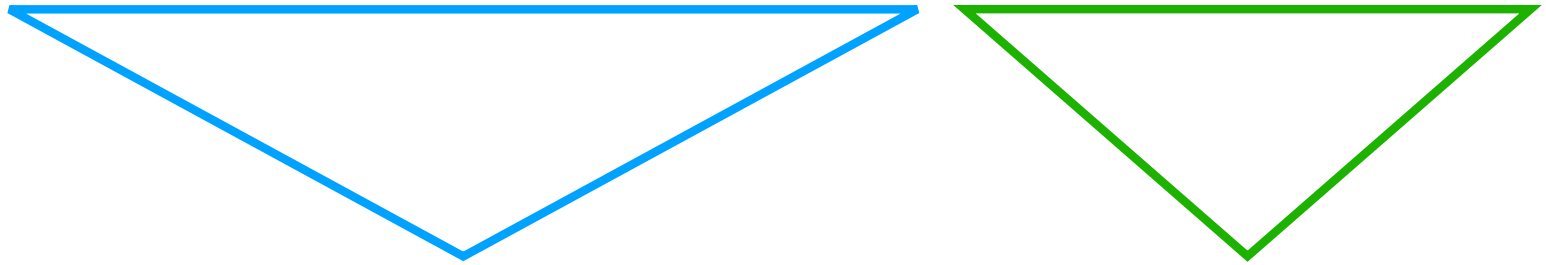


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner: Gradient Based Learning Applied to Document Recognition, Proceedings of IEEE, 86(11):2278–2324, 1998.

Hidden Layers





"Automatic feature extractor" "Regular classifier"

Counting the FC layers, this network has 5 layers

Convolutional Neural Networks

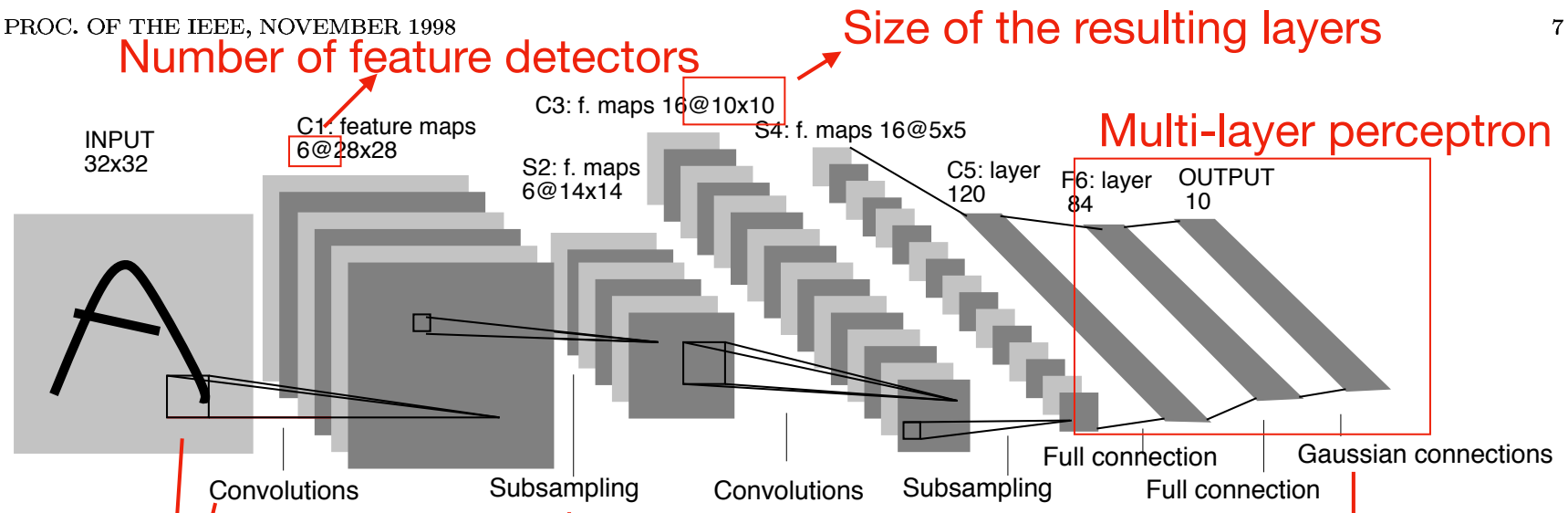


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

nowadays called "pooling"

basically a fully-connected layer + MSE loss
(nowadays common to use fc-layer + softmax + cross entropy)

"Feature detectors" (weight matrices) that are being reused ("weight sharing") => also called "kernel" or "filter"

Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner: Gradient Based Learning Applied to Document Recognition, Proceedings of IEEE, 86(11):2278–2324, 1998.

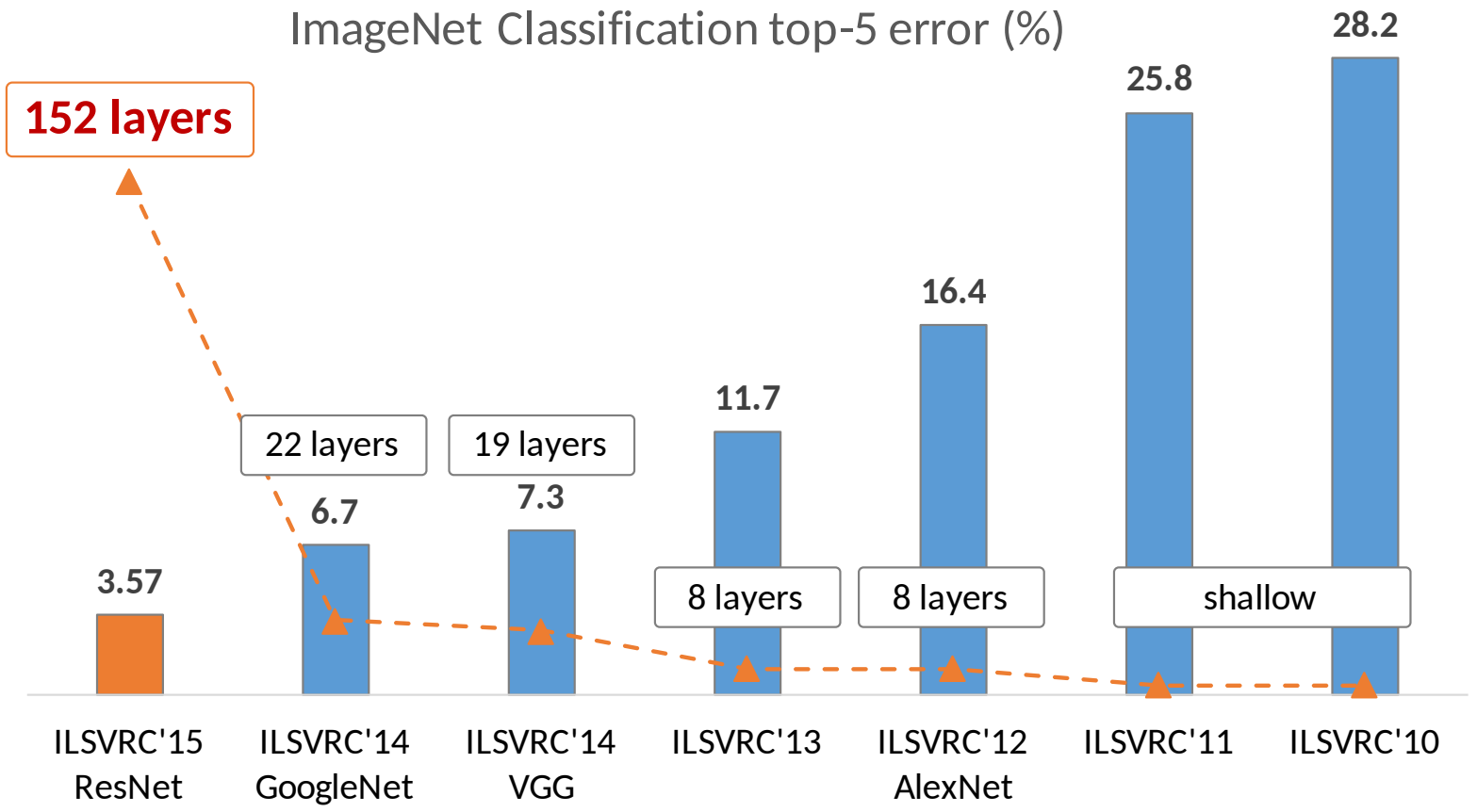
Main Concepts Behind Convolutional Neural Networks

- **Sparse-connectivity:** A single element in the feature map is connected to only a small patch of pixels. (This is very different from connecting to the whole input image, in the case of multi-layer perceptrons.)

- **Parameter-sharing:** The same weights are used for different patches of the input image.

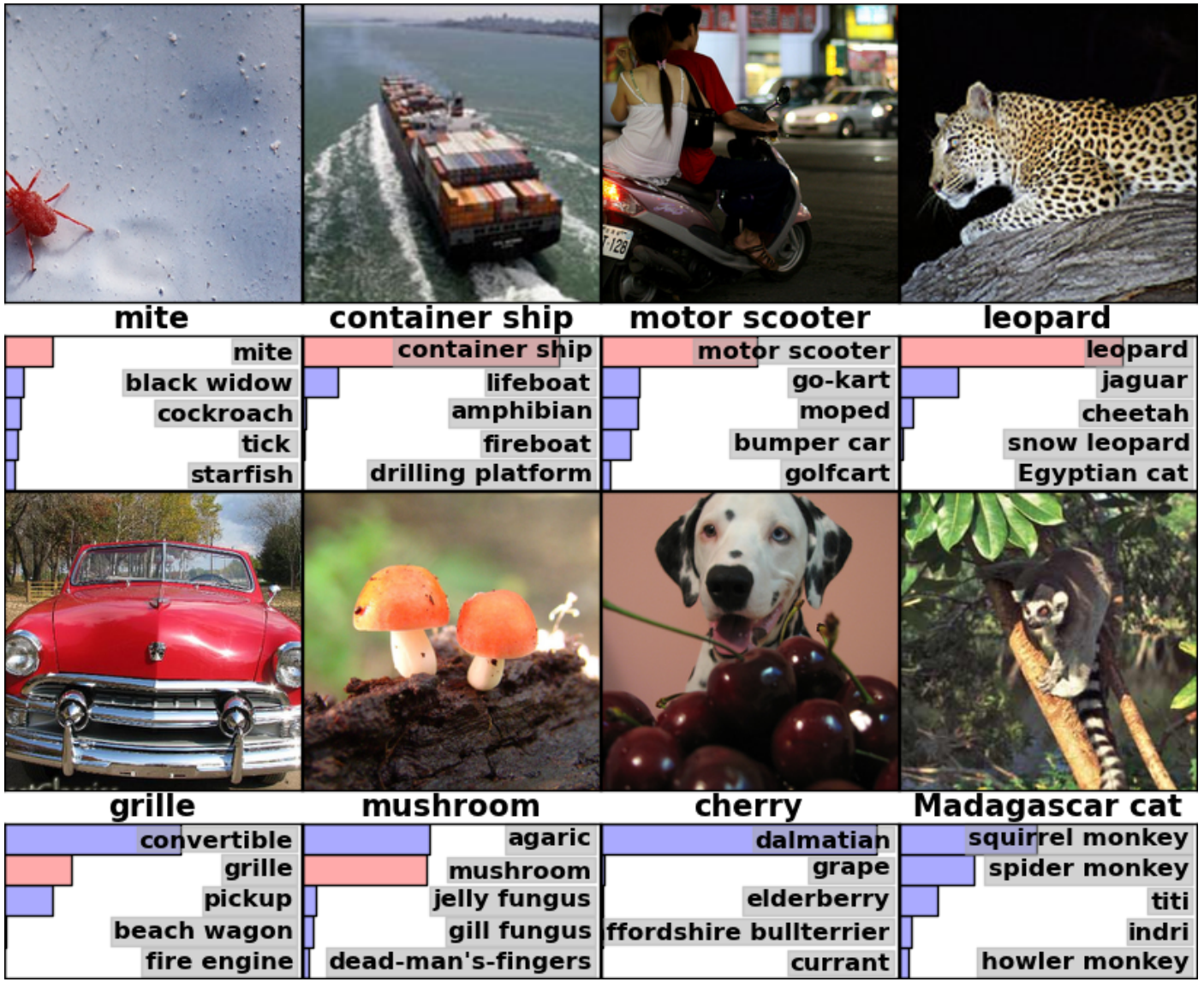
- **Many layers:** Combining extracted local patterns to global patterns

ImageNet Large Scale Visual Recognition Challenge



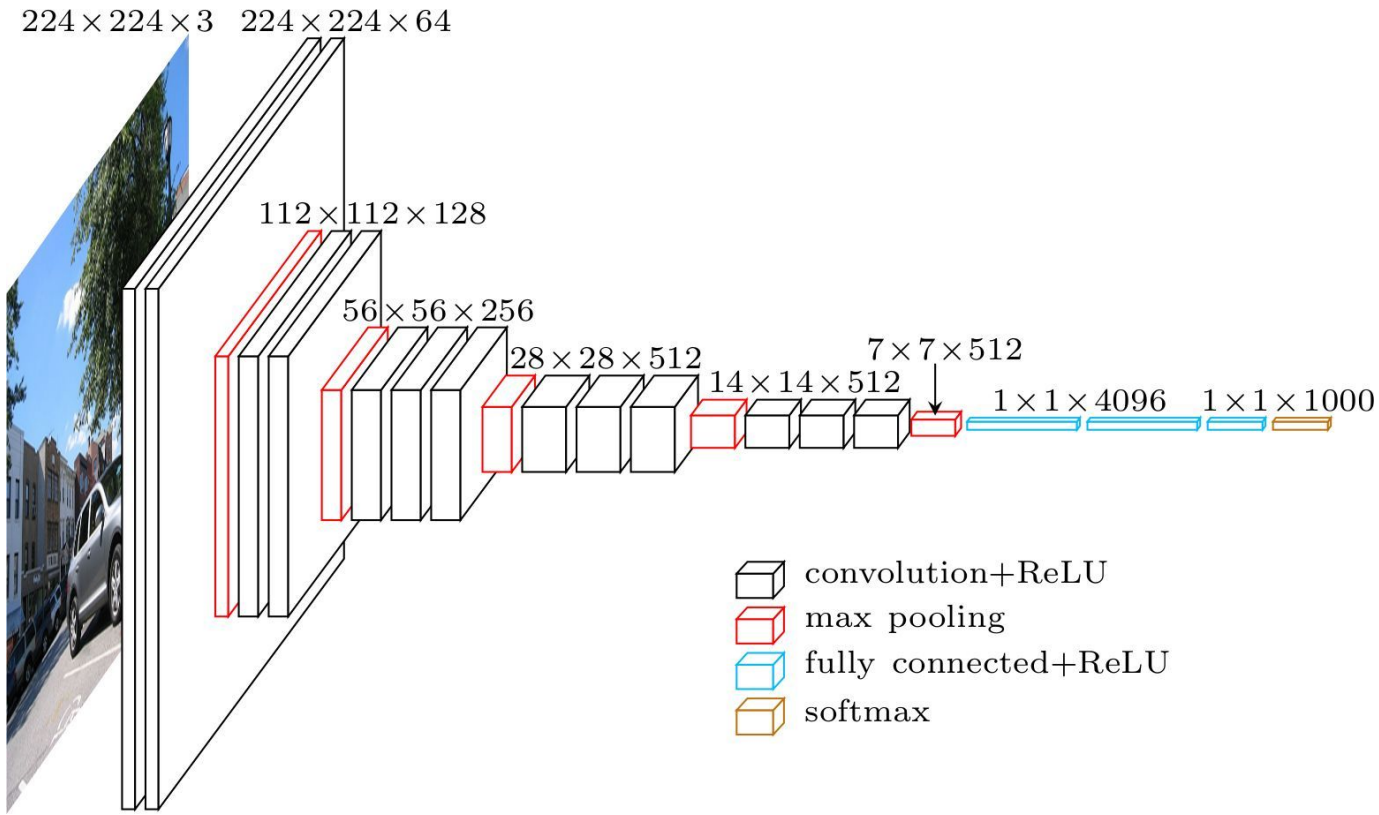
- Classification into 1000 object categories. Current state-of-the-art: 1.3 %

Russakovsky et al.: ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.



Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

VGG Network



3 Types of Layers:

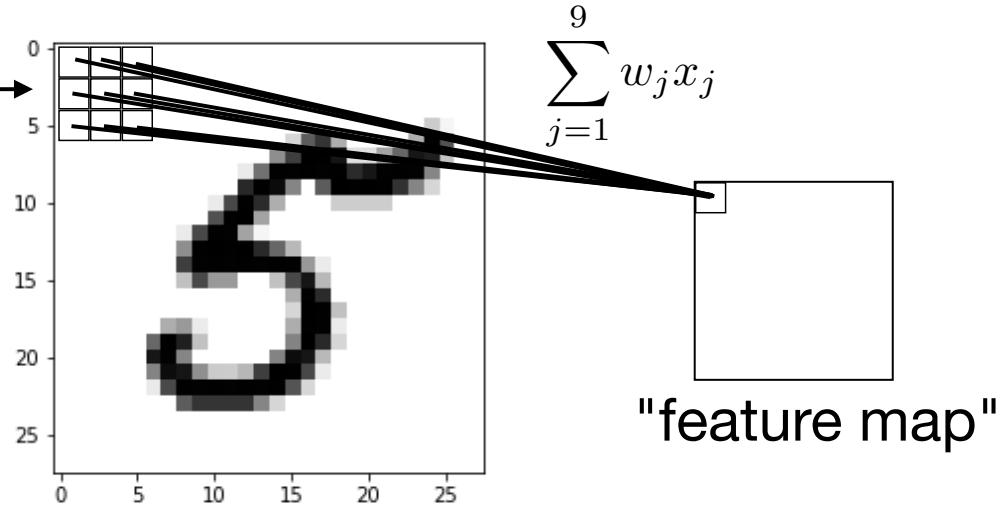
- Convolution layers, pooling layers and fully connected layers

Weight Sharing

A "feature detector" (filter, kernel) slides over the inputs to generate a feature map

Receptive field of a neuron is defined as all pixels in the input $\mathbf{X}$ which influence it

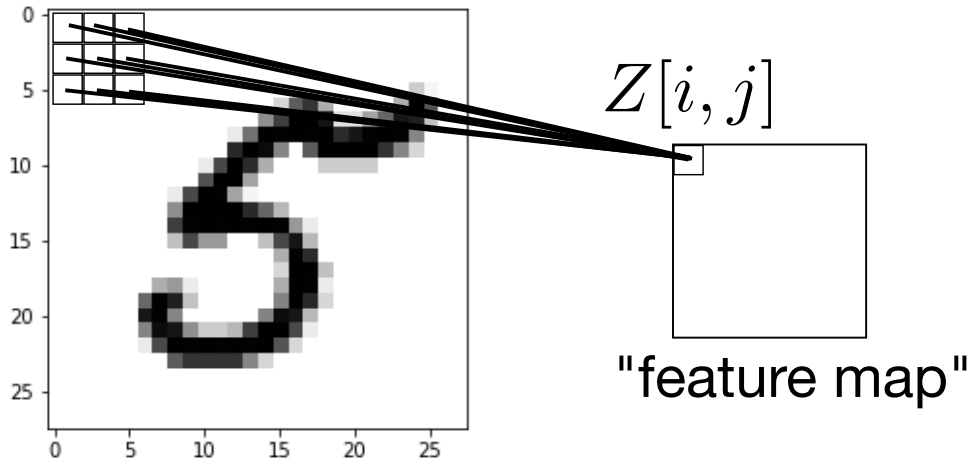
The pixels are referred to as "receptive field"



Rationale: A feature detector that works well in one region may also work well in another region

Plus, it is a nice reduction in parameters to fit

Cross-Correlation vs Convolution

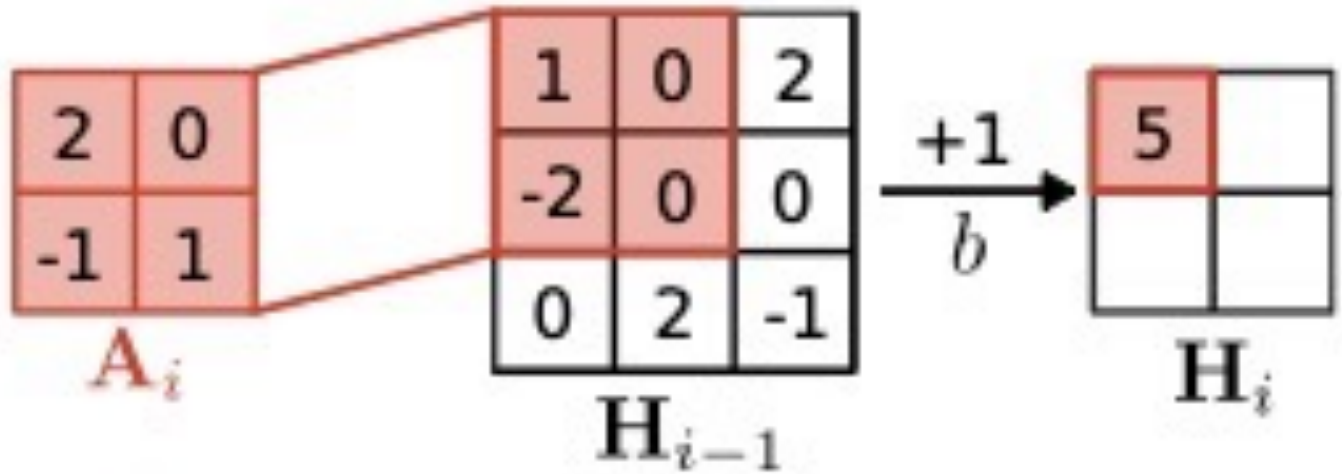


Cross-Correlation:

$$Z[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} K[u,v]A[i+u, j+v]$$

$$Z[i,j] = K \otimes A$$

Convolution Example

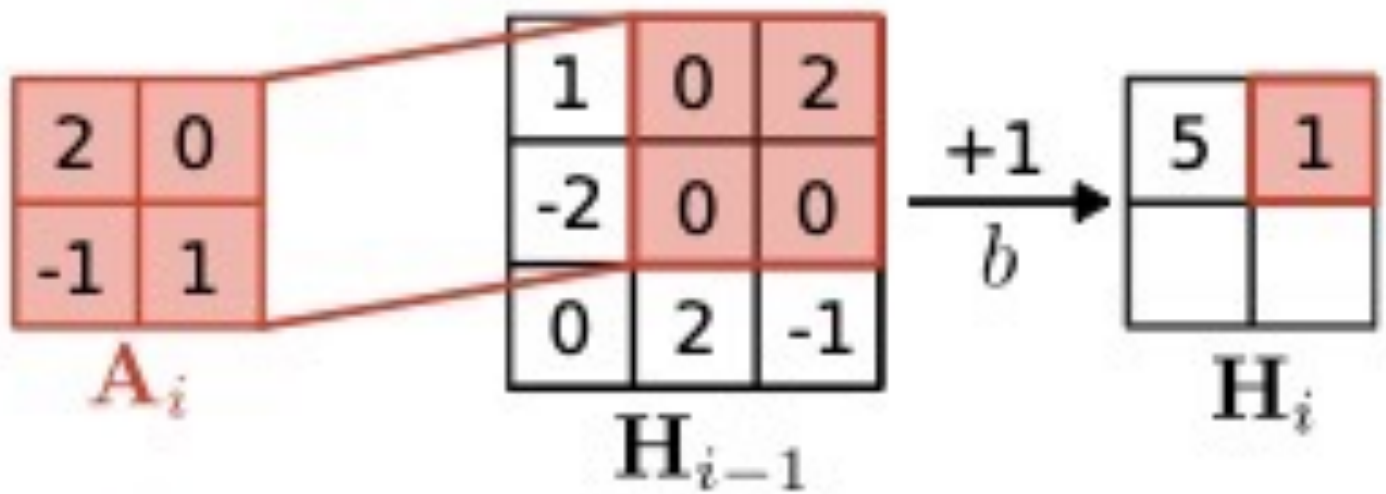


$$2 \cdot 1 + (-1) \cdot (-2) + 1 = 5$$

Remarks:

- Technically, ConvNets implement **correlation** (not convolution) operations

Convolution Example

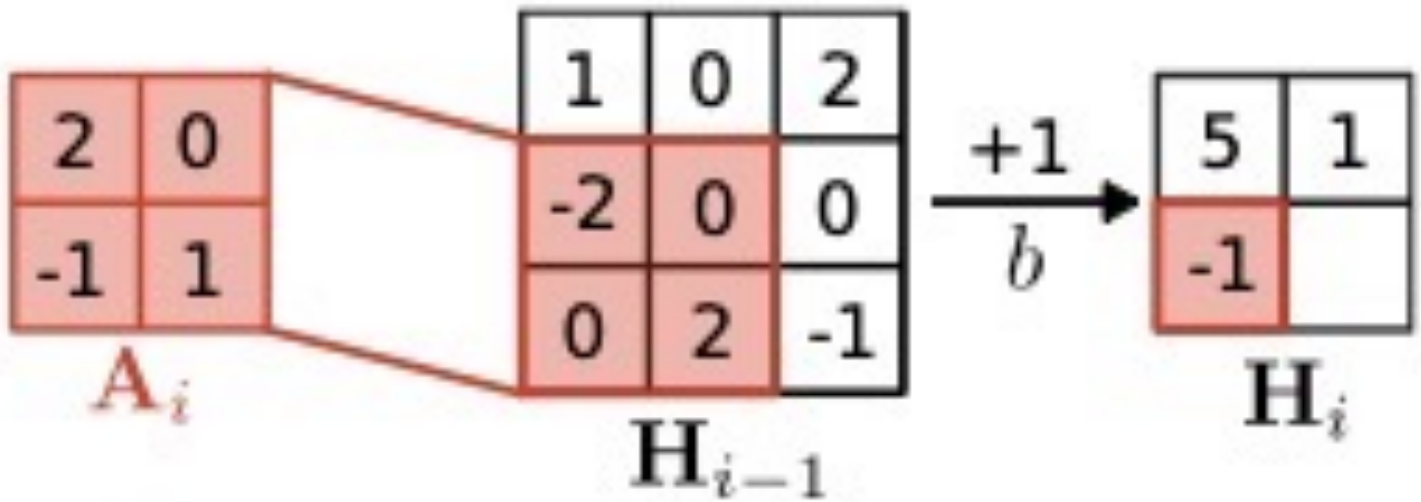


$$0 + 1 = 1$$

Remarks:

- Technically, ConvNets implement **correlation** (not convolution) operations

Convolution Example

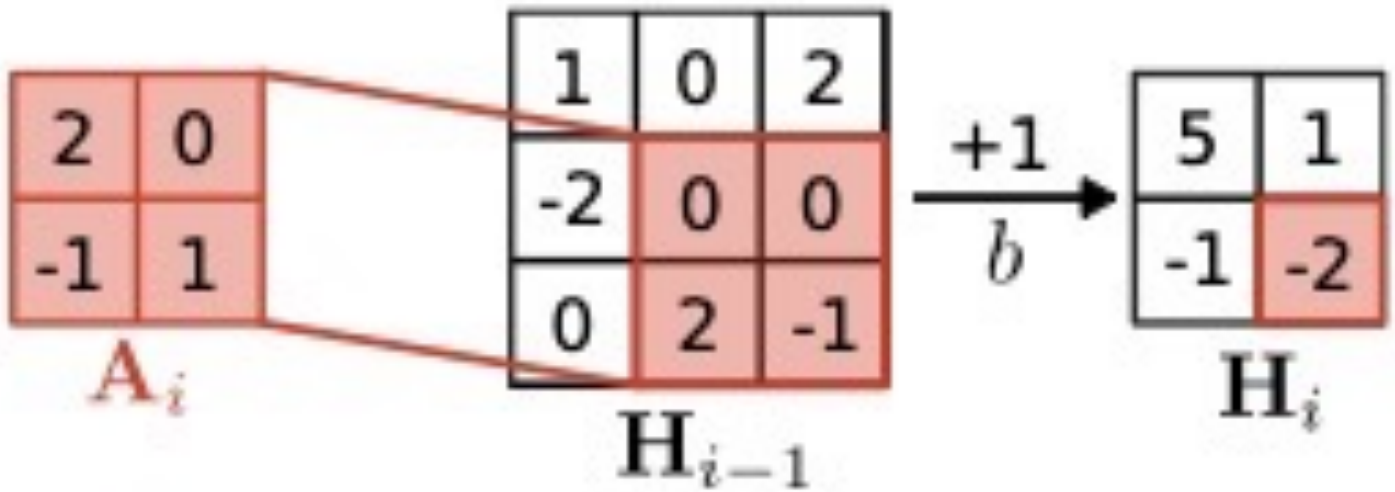


$$2 \cdot (-2) + 1 \cdot 2 + 1 = -1$$

Remarks:

- Technically, ConvNets implement **correlation** (not convolution) operations

Convolution Example



$$(-1) \cdot 2 + 1 \cdot (-1) + 1 = -2$$

Remarks:

- Technically, ConvNets implement **correlation** (not convolution) operations

Convolution Example

$*$

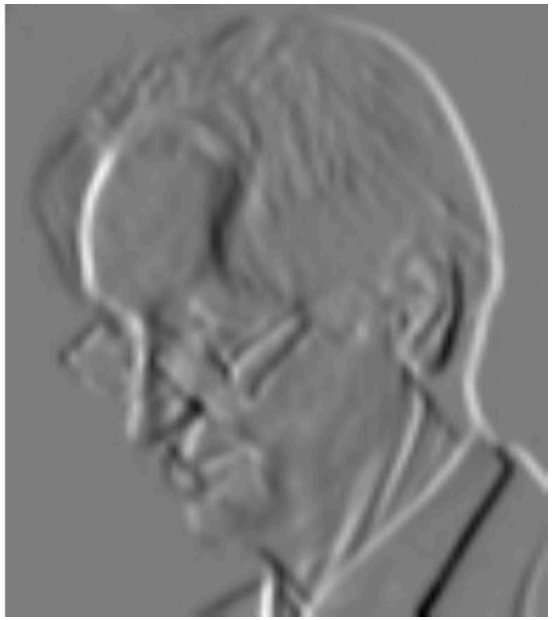
0	1	0
1	4	1
0	1	0

- Common filters. In deep learning our goal is to learn the filter kernels.

Convolution Example

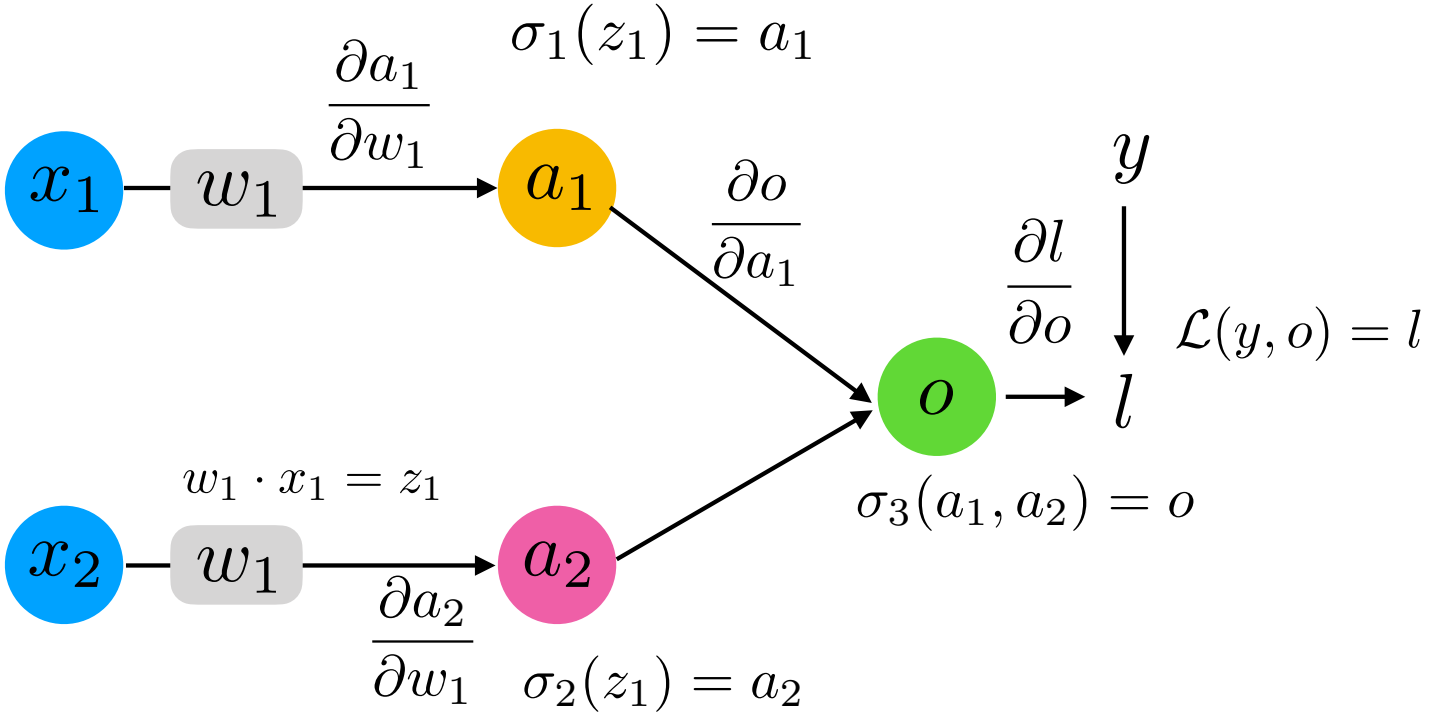
$*$

1	0	-1
2	0	-2
1	0	-1



- Common filters. In deep learning our goal is to learn the filter kernels.

Backpropagation in CNNs



Upper path

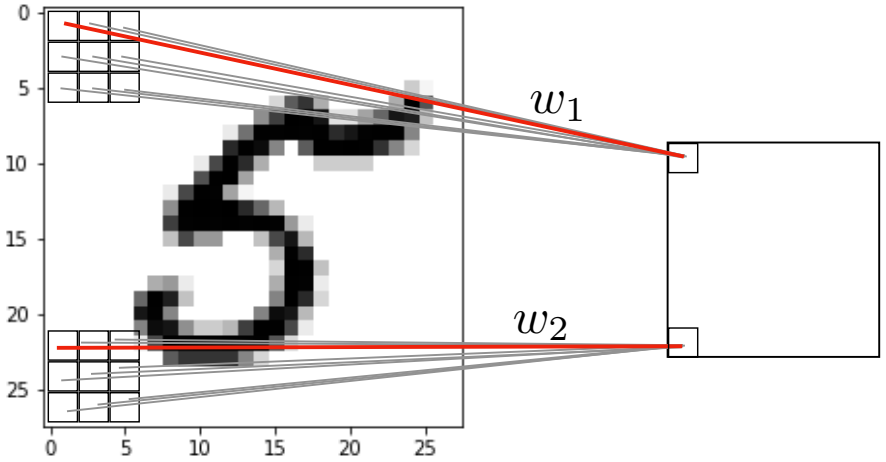
$$\frac{\partial l}{\partial w_1} = \frac{\partial l}{\partial o} \cdot \frac{\partial o}{\partial a_1} \cdot \frac{\partial a_1}{\partial w_1} + \frac{\partial l}{\partial o} \cdot \frac{\partial o}{\partial a_2} \cdot \frac{\partial a_2}{\partial w_1}$$ (multivariable chain rule)

Lower path

Backpropagation in CNNs

Same overall concept as before: Multivariable chain rule, but now with an additional weight sharing constraint

Due to weight sharing: $w_1 = w_2$

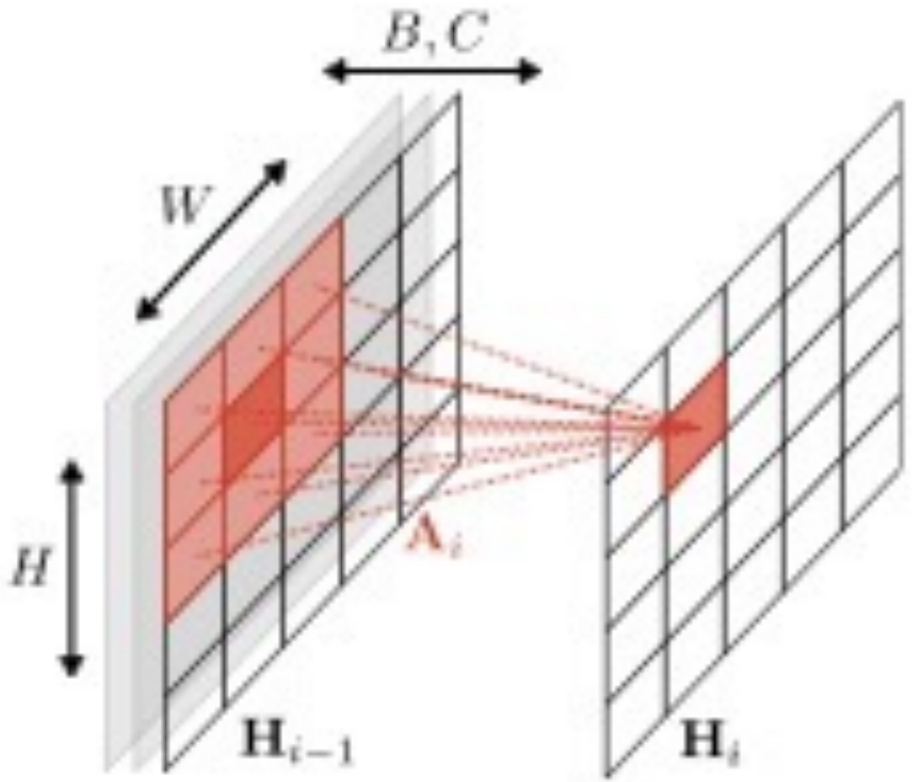


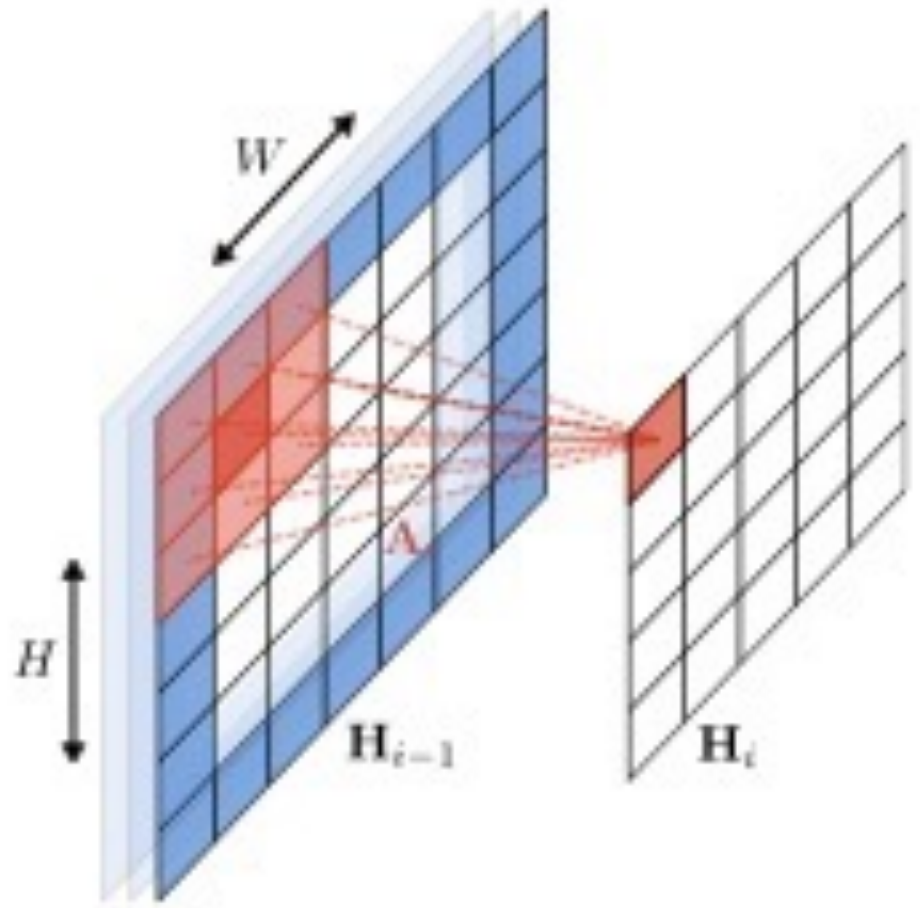
Optional averaging

weight update:

$$w_1 := w_2 := w_1 - \eta \cdot \frac{1}{2}\left(\frac{\partial \mathcal{L}}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial w_2}\right)$$

Padding

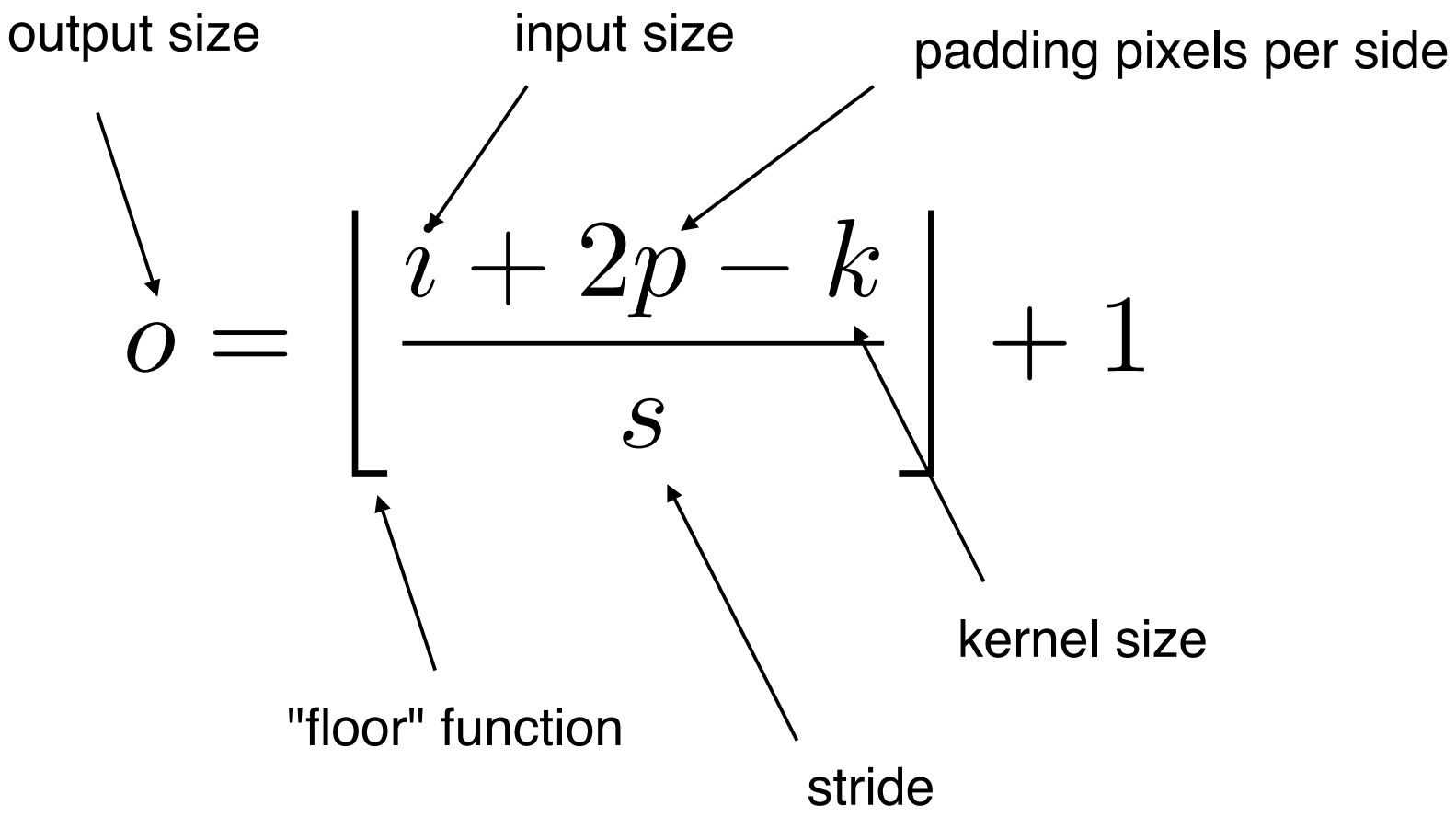




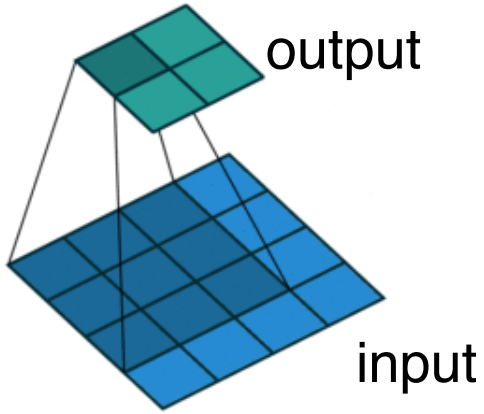
Idea of Padding:

- Add boundary of appropriate size with zeros (blue) around input tensor

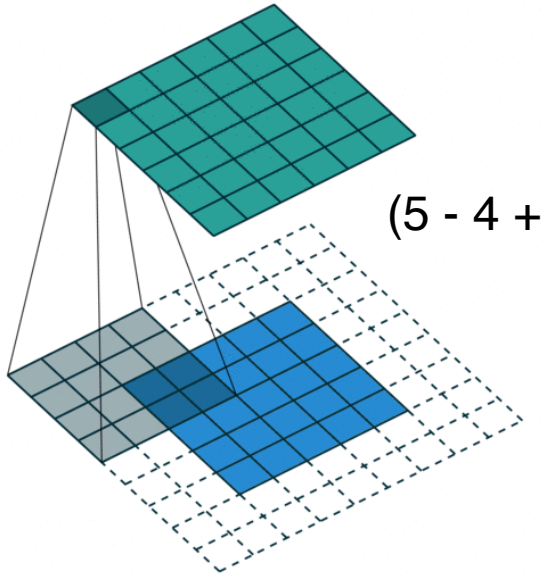
Padding


output size
input size
padding pixels per side
$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1$
"floor" function
stride
kernel size

(4 - 3 + 2*0)/1 + 1 = 2

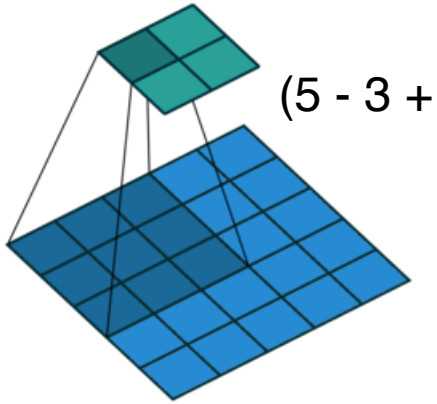


No padding, stride=1



(5 - 4 + 2*2)/1 + 1 = 6

padding=2, stride=1



(5 - 3 + 2*0)/2 + 1 = 2

No padding, stride=2

Highly recommended:

Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

Padding

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1$$

Assume you want to use a convolutional operation with
stride 1 and maintain the input dimensions in the output feature map:

How much padding do you need for "same" convolution?

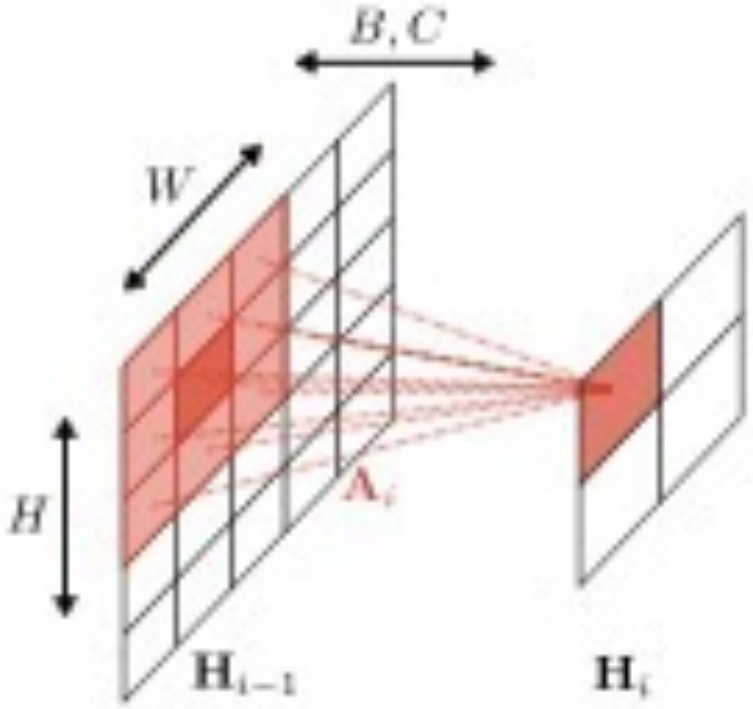
$$o = i + 2p - k + 1$$

$$\Leftrightarrow p = (o - i + k - 1)/2$$

$$\Leftrightarrow p = (k - 1)/2$$

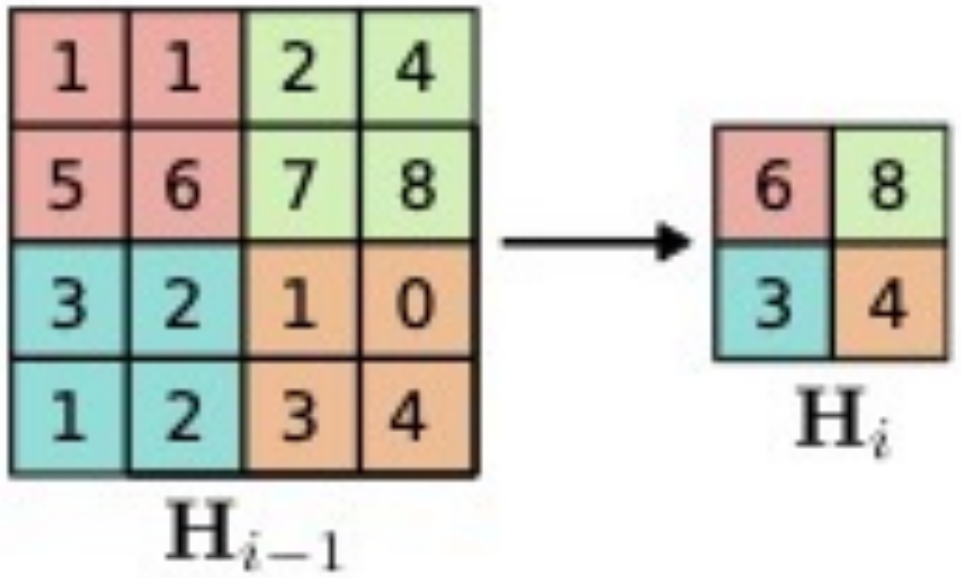
Probably explains why common kernel size conventions are
3x3, 5x5, 7x7 (sometimes 1x1 in later layers to reduce channels)

Pooling



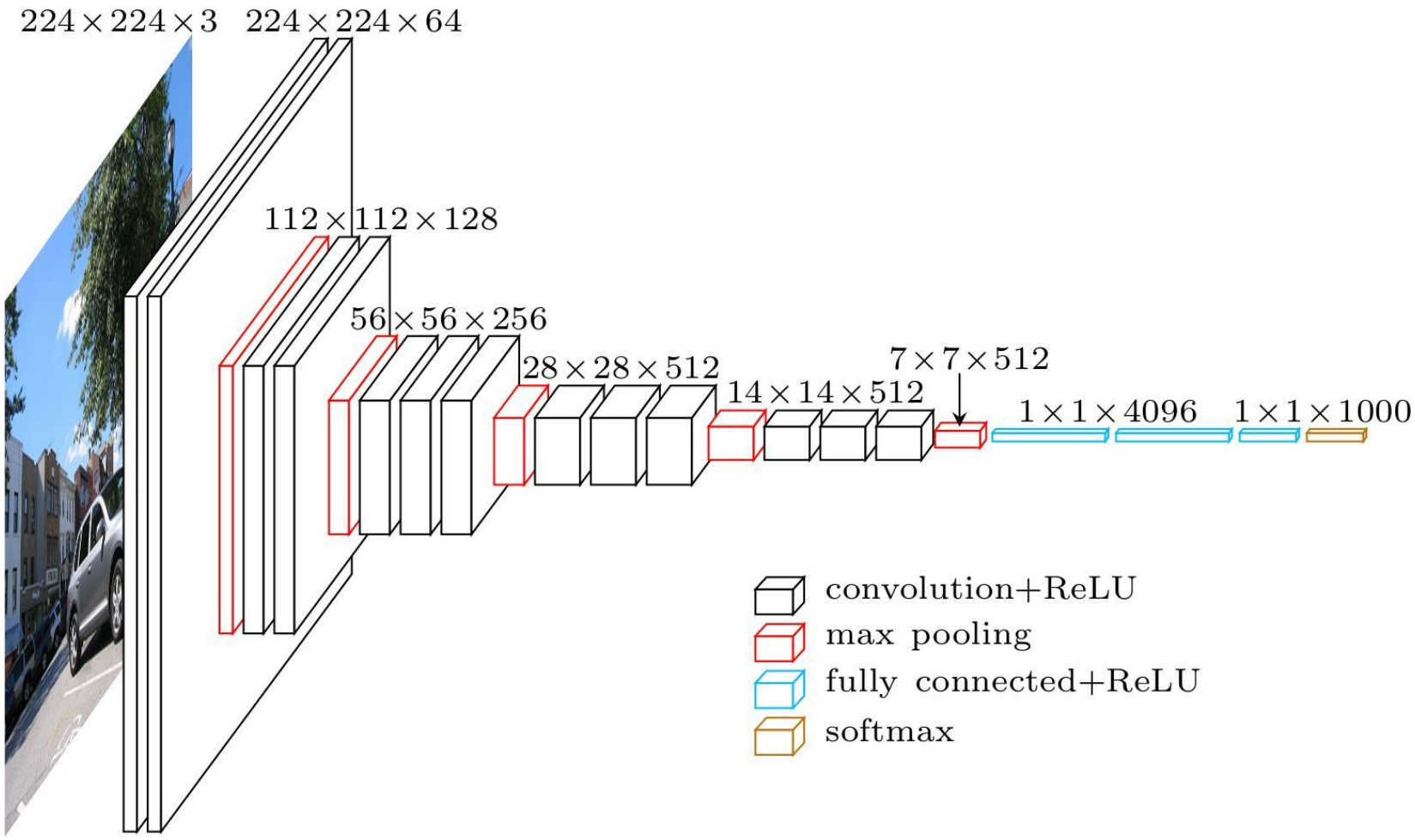
- Downsampling **reduces the spatial resolution** (e.g., for image level predictions)
- Typically, stride $s = 2$ and kernel size $2 \times 2 \Rightarrow$ **reduces spatial dimensions** by 2
- Pooling has **no parameters** (typical pooling operations: max, min, mean)
- Pooling is **applied to each channel separately** $\Rightarrow$ keeps number of channels

Pooling Example



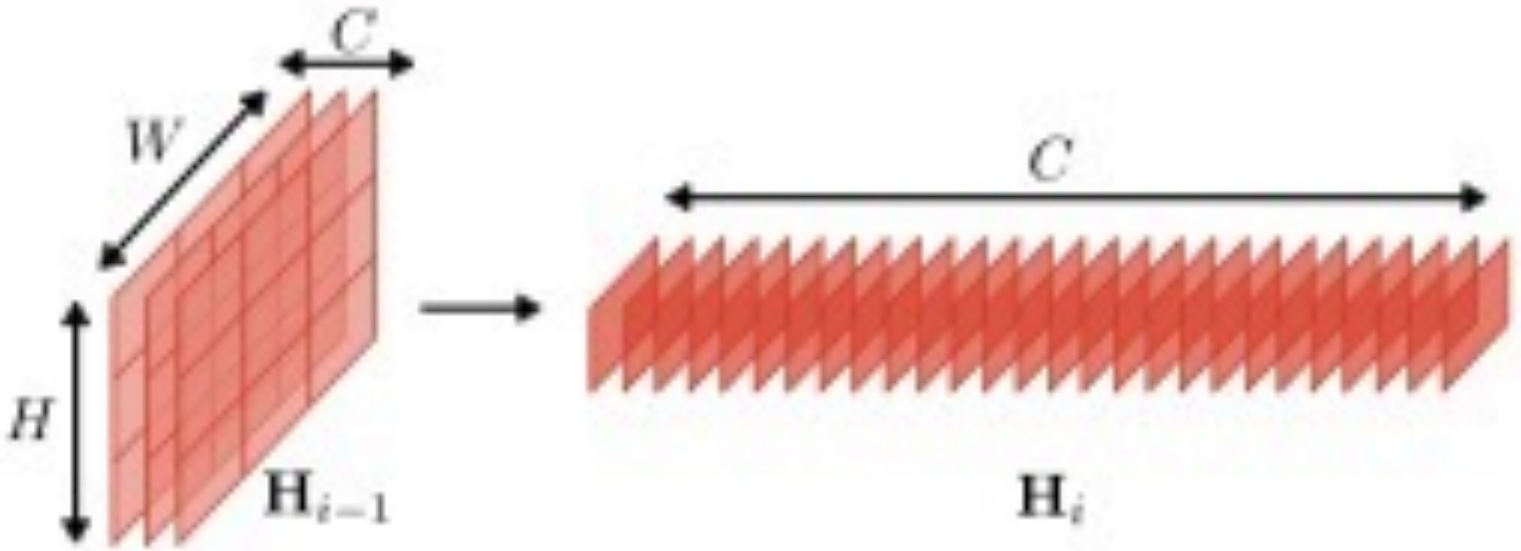
- Max pooling with stride $s = 2$ and kernel size 2×2 reduces spatial dimension by 2
- Remark: Max pooling provides **invariance** to small translations of the input

Fully Connected Layers



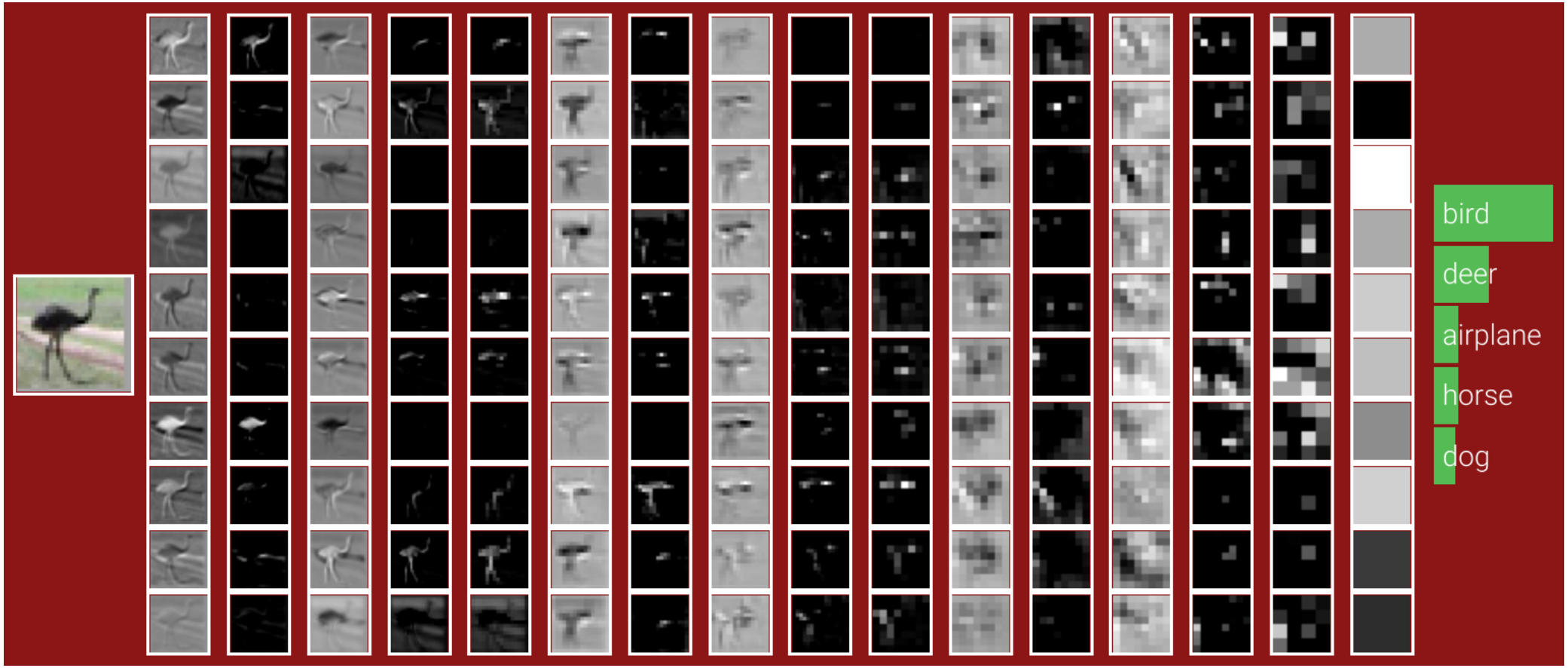
- Fully connected layers are most **memory intensive** part of VGG architecture

Fully Connected Layers



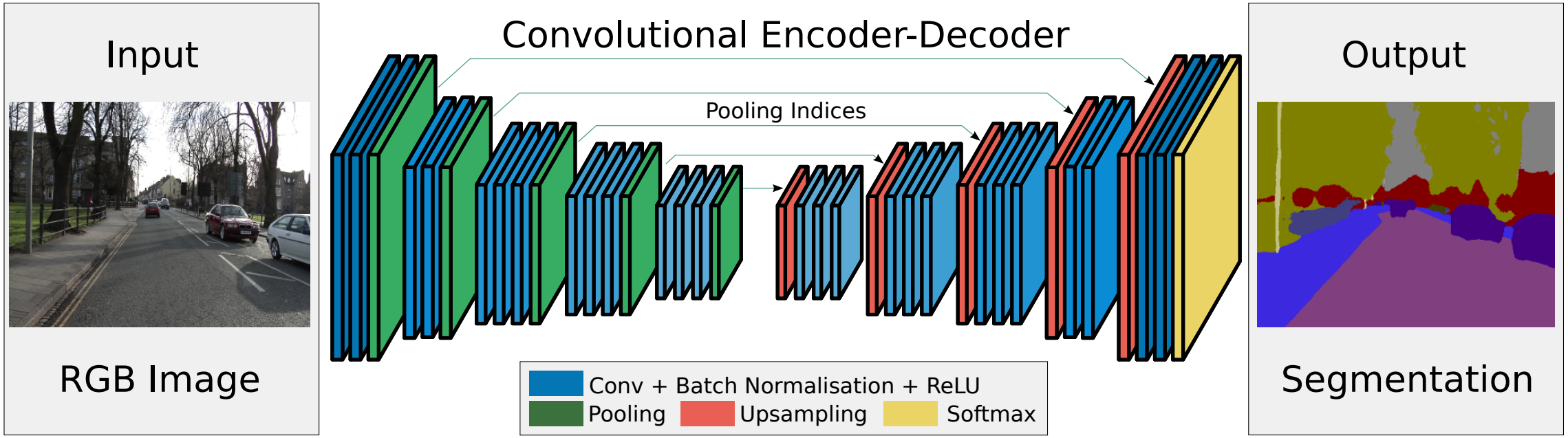
Now X and Y are **reshaped** into the feature channel dimension C

Convolution Network Example



http://cs231n.stanford.edu/2017/index.html

Upsampling

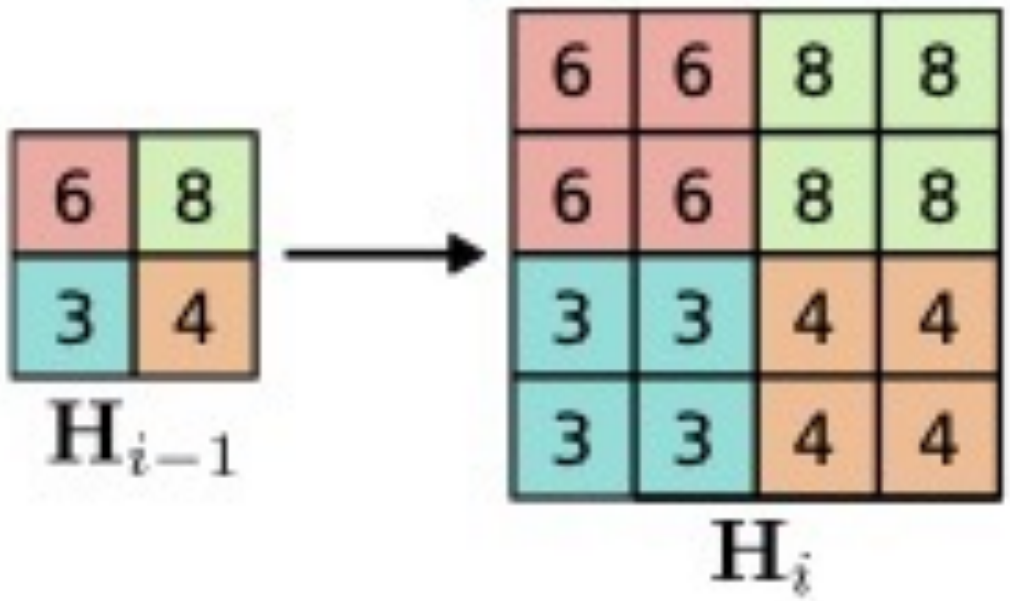


- If **pixel-level outputs** are desired, we have to **upsample** the features again
- Downsampling provides strong features with large receptive field
- Upsampling yields output at the same resolution as input

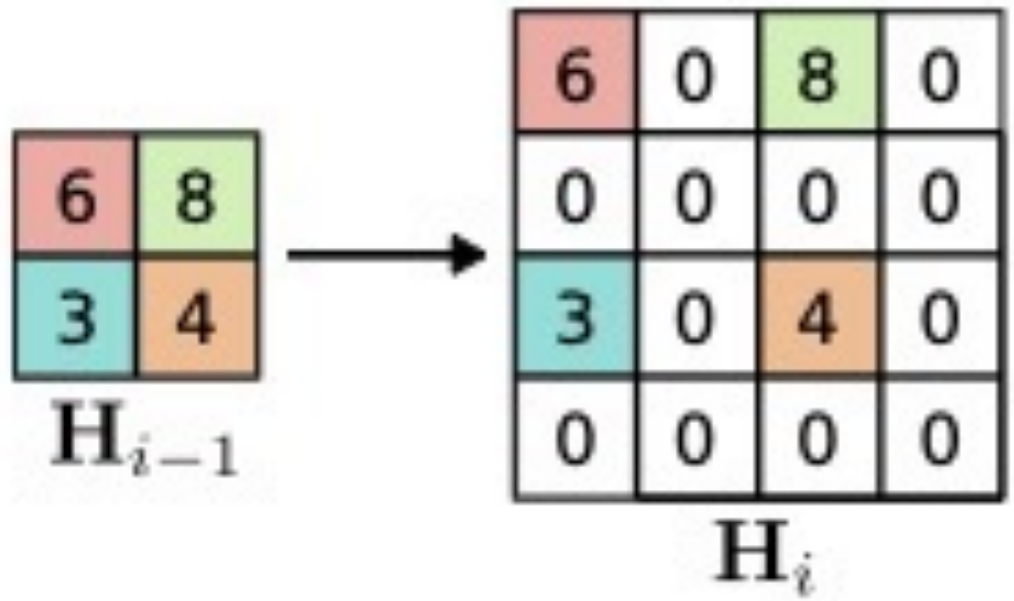
Badrinarayanan, Kendall and Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. TPAMI, 2017.

Upsampling

Nearest Neighbor Upsampling

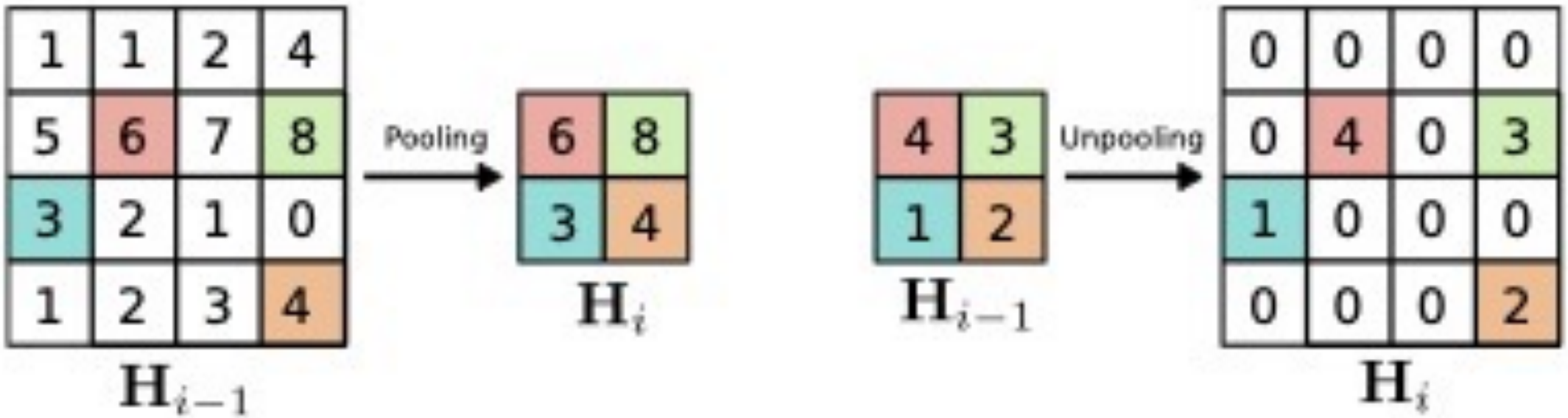


Bed of Nails Upsampling



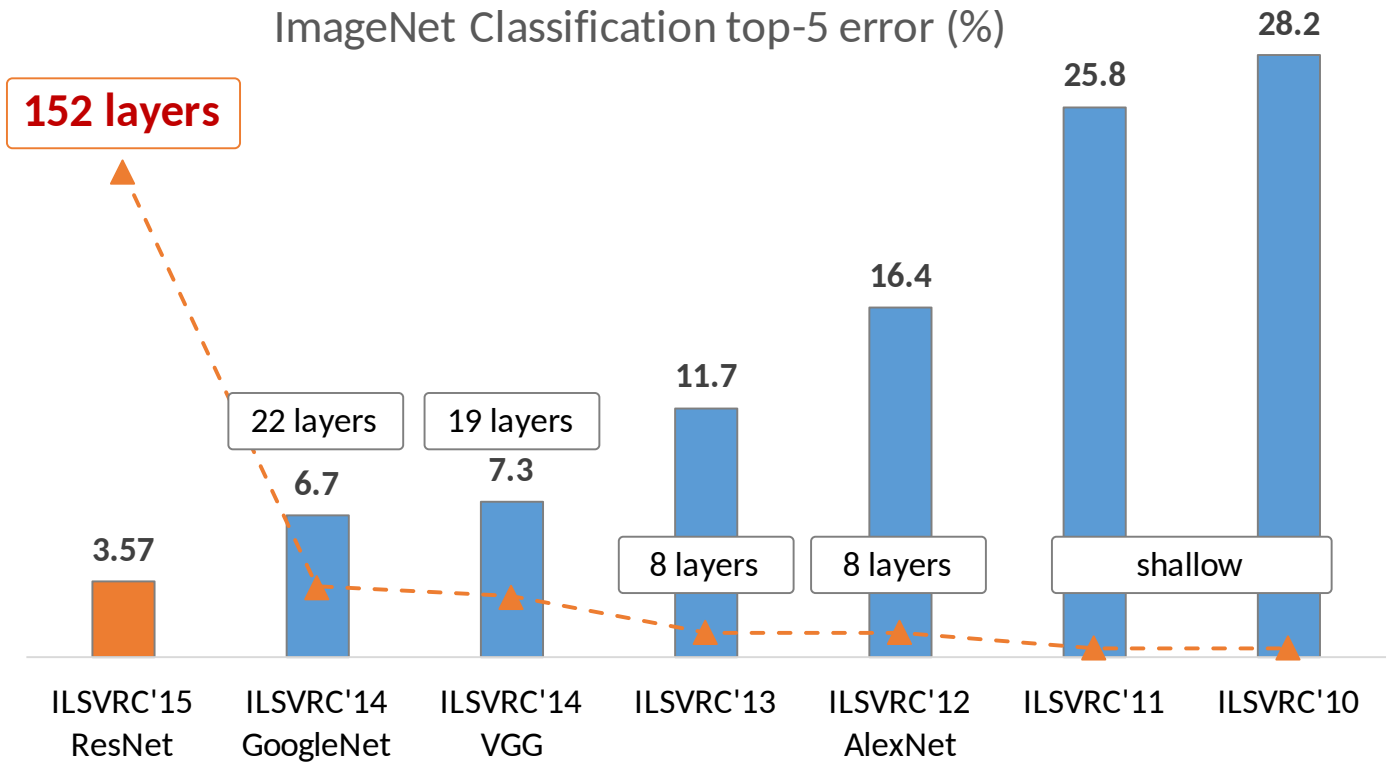
- **Nearest neighbor:** scale each channel using nearest neighbor interpolation
- **Bilinear:** scale each channel using bilinear neighbor interpolation
- **Bed of nails:** insert elements at sparse location (followed by convolution)

Max Unpooling



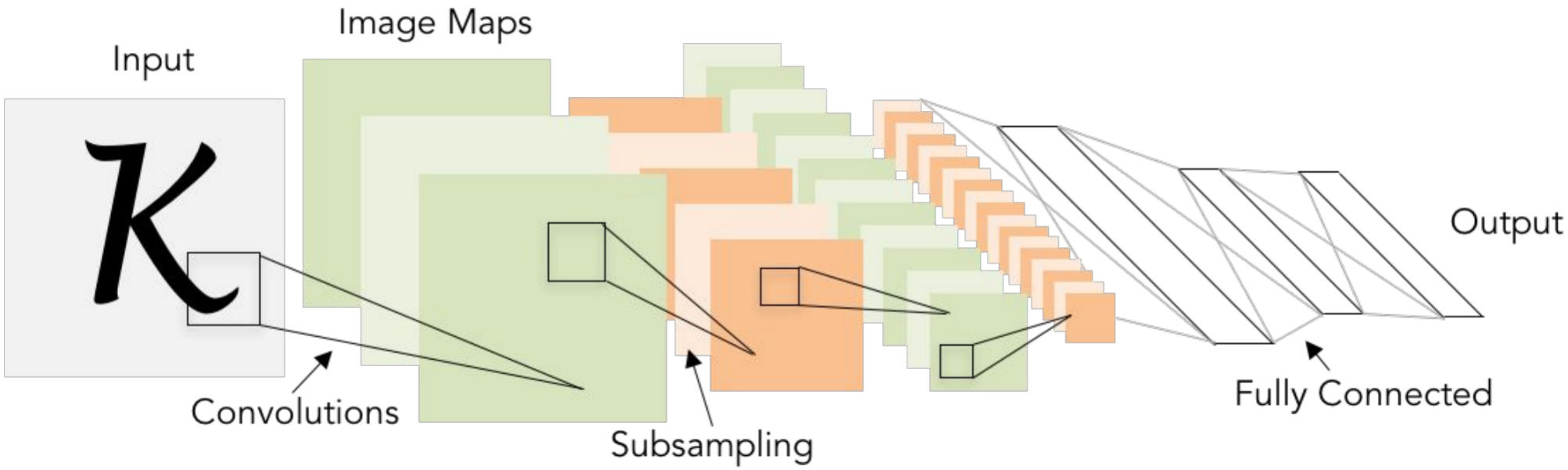
- For unpooling, **remember** which element was **maximum** during pooling
- Requires corresponding pairs of downsampling and upsampling layers
- This approach has been used in SegNet

ImageNet Large Scale Visual Recognition Challenge



- Deeper networks are better (provided enough training data and compute)
- Nearly all state-of-the-art networks for image applications use convolution layers

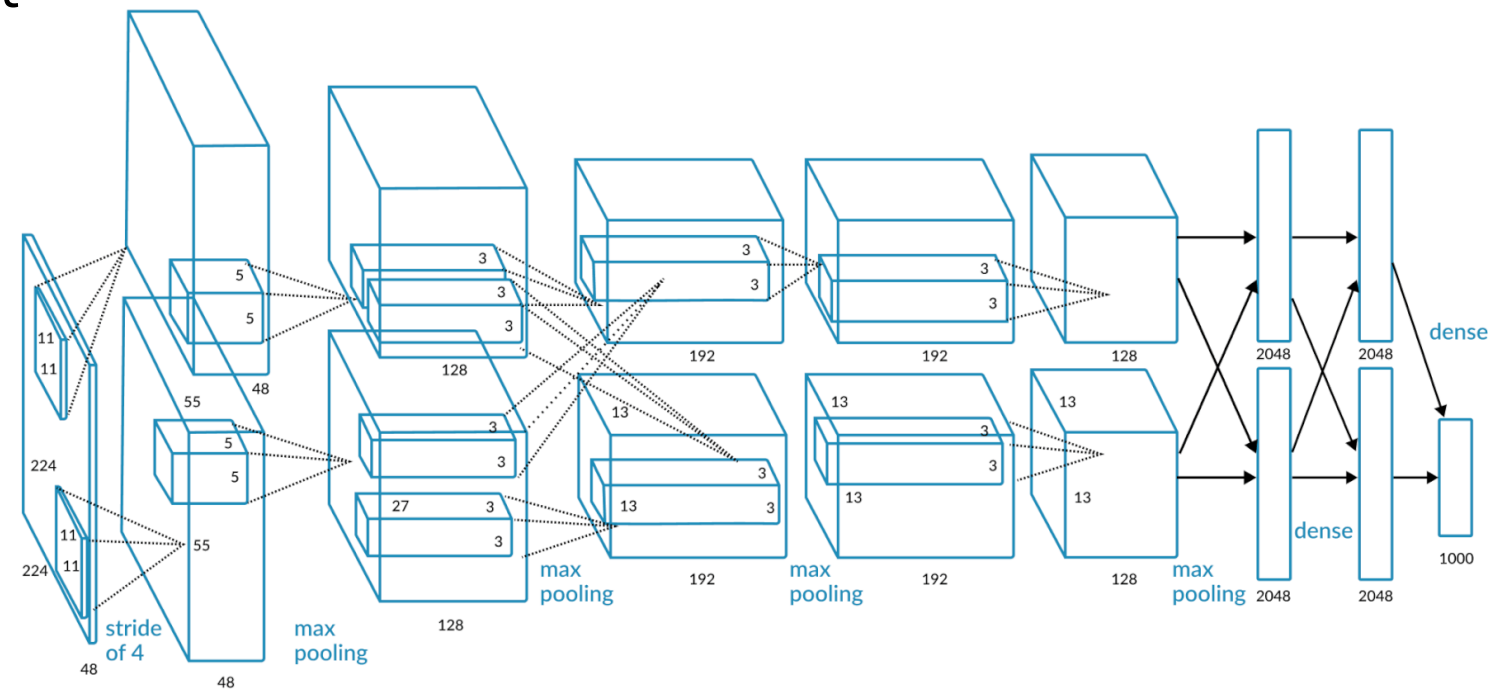
1998: LeNet-5



- 2 convolution layers (5×5), 2 pooling layers (2×2), 2 fully connected layers
- Achieved state-of-the-art accuracy on MNIST (prior to ImageNet)

LeCun et al. Gradient-based learning applied to document recognition. IEEE, 1998.

2012: AlexNet

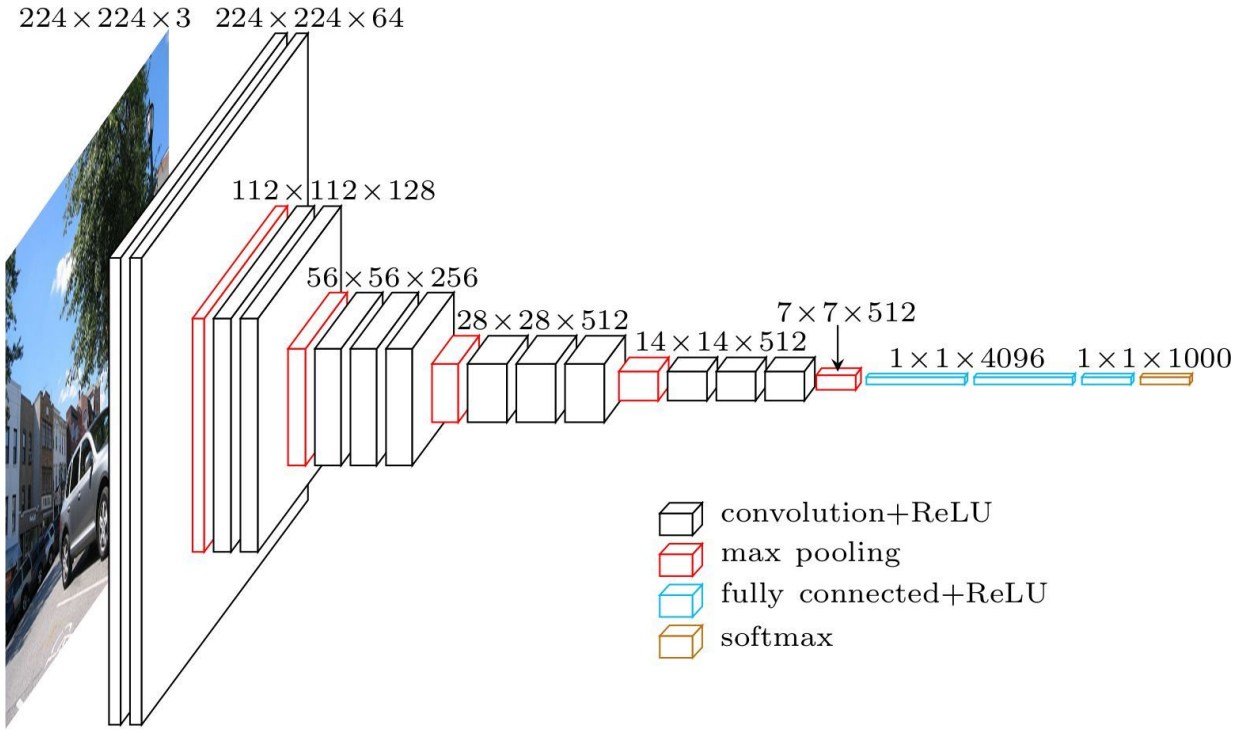


AlexNet:

- 8 layers, ReLUs, dropout, data augmentation, trained on 2 GTX 580 GPUs
- Number of feature channels increase with depth, spatial resolution decreases
- Triggered deep learning revolution, showed that CNNs work well in practice

Krizhevsky, Sutskever and Hinton: ImageNet Classification with Deep Convolutional Neural Networks. NIPS, 2012.

2015: VGG

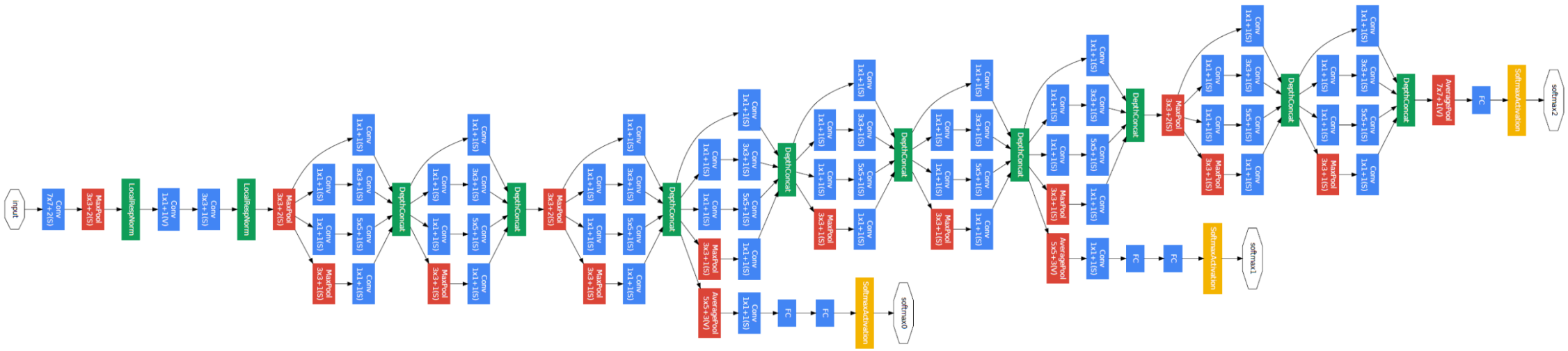


VGG:

- Uses 3×3 convolutions everywhere (same expressiveness, fewer parameters)
- Three 3×3 layers: same receptive field as one 7×7 layer, but less parameters
- 2 Variants: 16 and 19 Layers. Showed that deeper networks are better.

Simonyan and Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2015.

2015: Inception / GoogLeNet

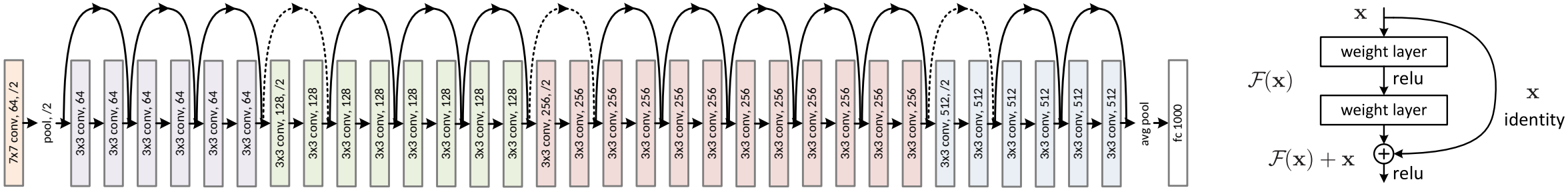


Inception:

- 22 layers. Inception modules utilize conv/pool operations with varying filter size
- Multiple intermediate classification heads to improve gradient flow
- Global avarage pooling (no FC layers), 27x less parameters (5 million) than VGG-16
- Uses 1×1 convolutions to reduce number of features $\Rightarrow$ higher efficiency

Szegedy et al.: Going deeper with convolutions. CVPR, 2015.

2016: ResNet

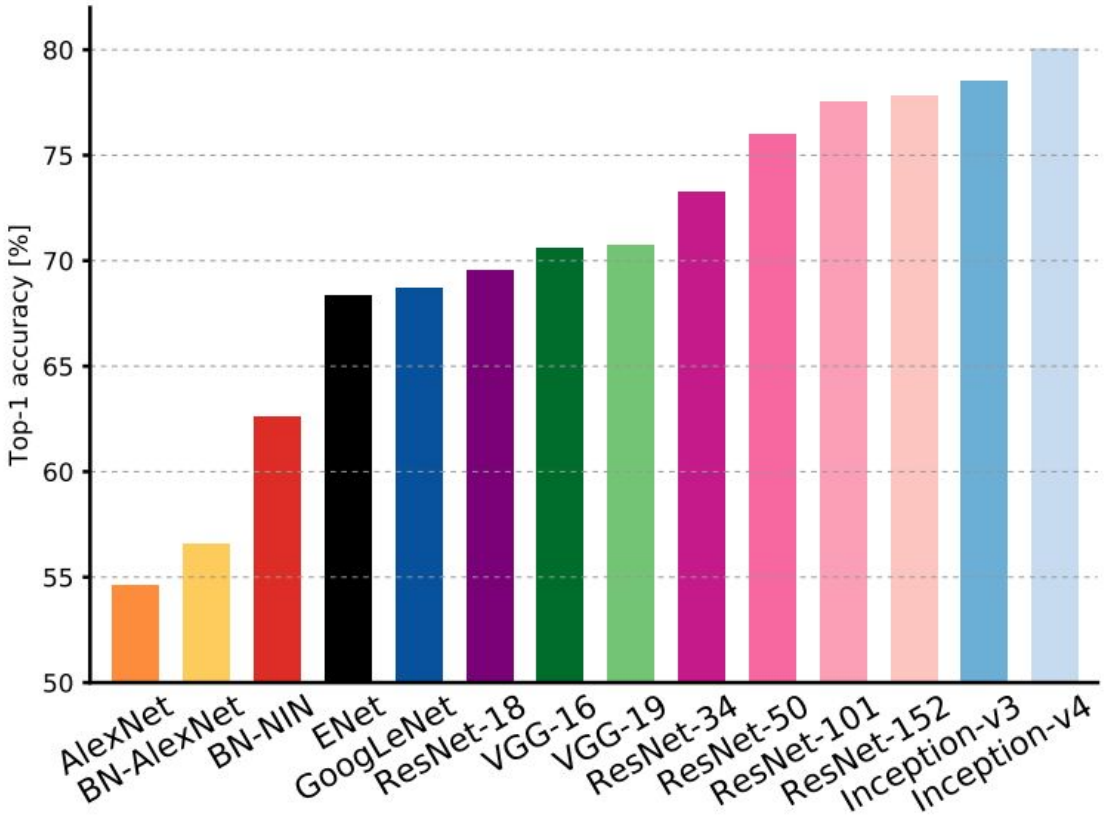


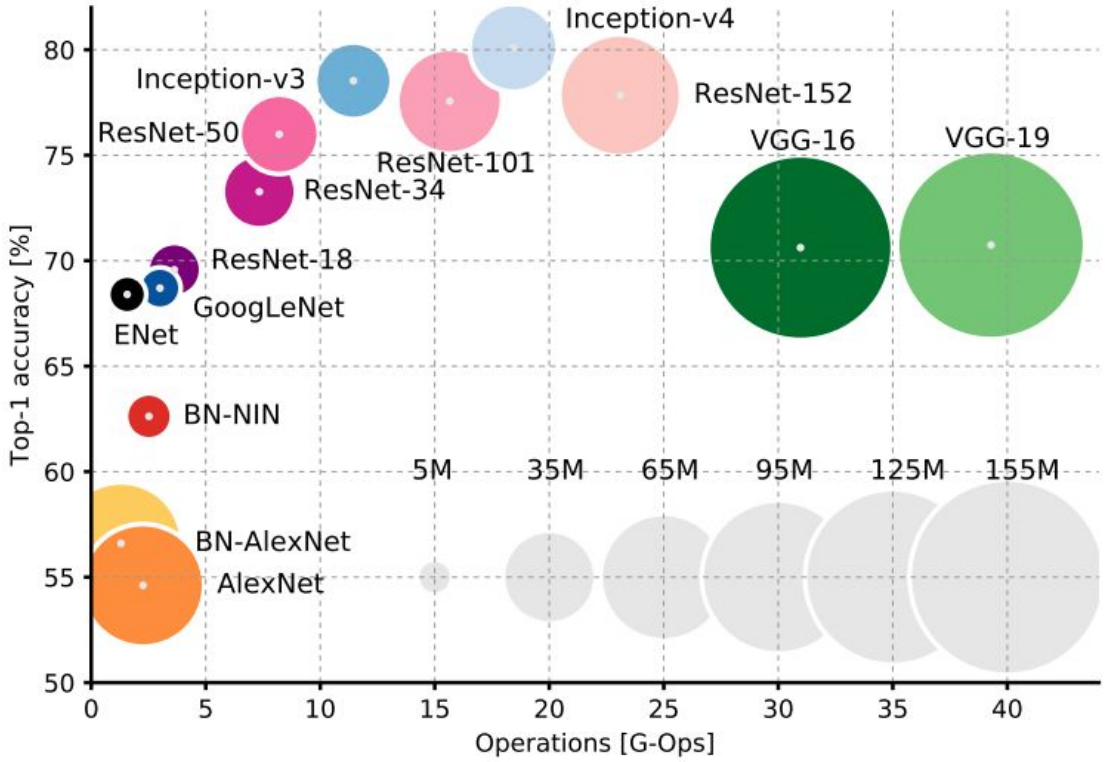
ResNet:

- Residual connections allow for training deeper networks (up to 152 layers)
- Very simple and regular network structure with 3×3 convolutions
- Uses strided convolutions for downsampling
- ResNet and ResNet-like architectures are dominating today

He, Zhang, Ren and Sun: Deep Residual Learning for Image Recognition. CVPR, 2016.

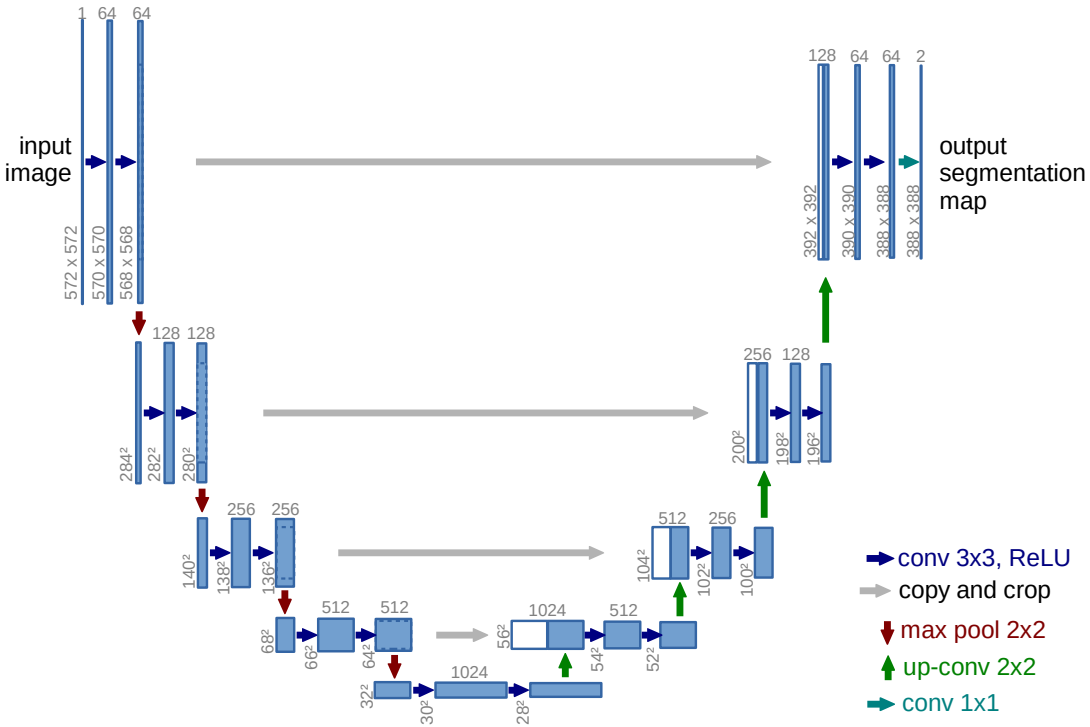
Complexity





- VGG has most parameters and is slowest
- ResNet/Inception/GoogLeNet are faster and have fewer parameters

U-Net



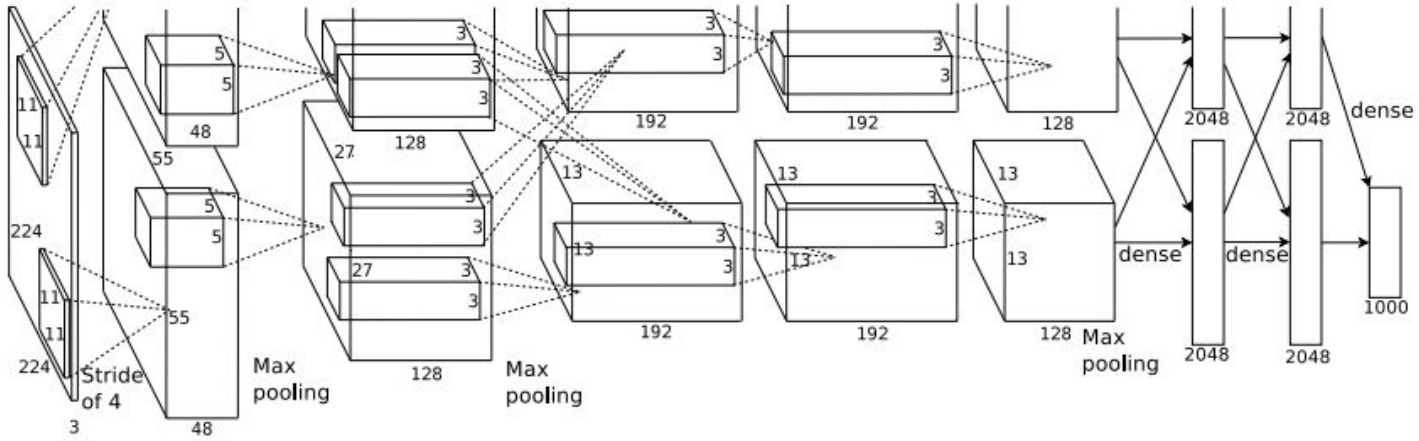
U-Net:

- Max-pooling, up-convolutions and skip-connections
- Defacto standard for many tasks with image output (e.g., depth, segmentation)

Ronneberger, Fischer and Brox: U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI, 2015.

Visualization

Input Image

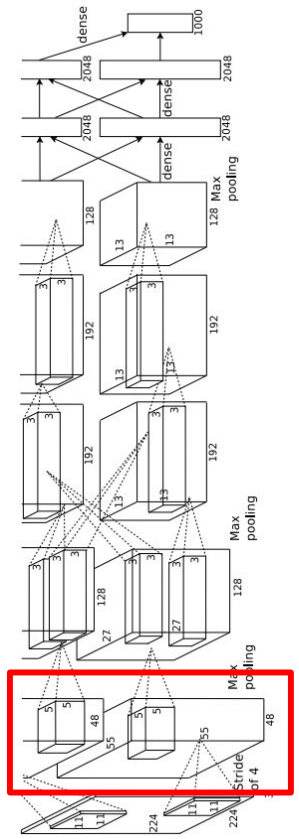


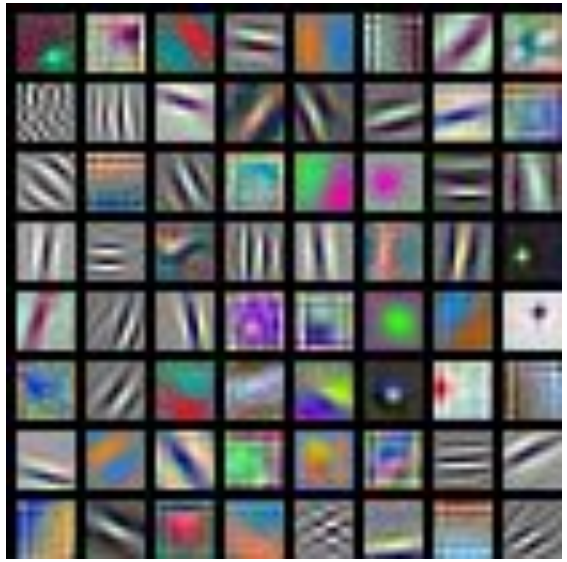
Class Scores

What are the intermediate features looking for?

Slide Credits: Fei-Fei Li, Ranjay Krishna, Danfei Xu. Lecture 13.

First Layers: Visualization of Features





AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

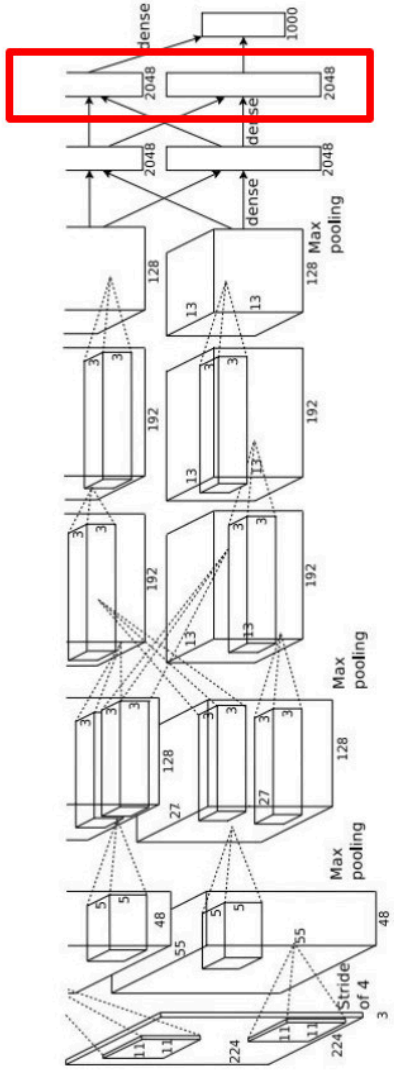
ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7

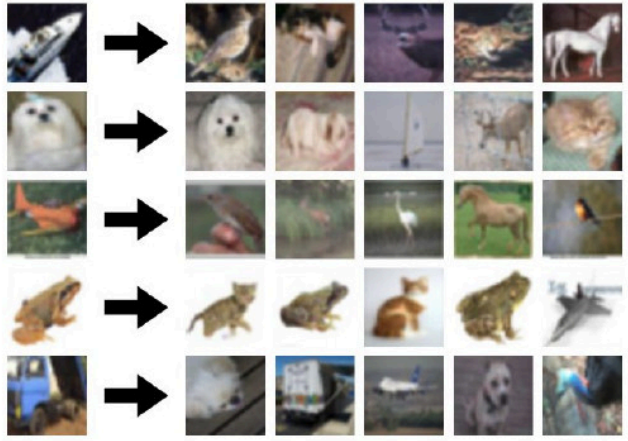
- First layer learns **simple gradient/Gabor filters** $\Rightarrow$ connection to V1 cells in brain
- Visualization less useful for later layers

Last Layer: Nearest Neighbors

4096-dim vector



Recall: Nearest neighbors in pixel space



Test image L2 Nearest neighbors in feature space

Krizhevsky et al.: ImageNet Classification with Deep Convolutional Neural Networks. NIPS, 2012.