Computer Vision

CS152 DS153 Sp’24

Class 18: Convolutional Neural Networks

Convolutional Neural Nets

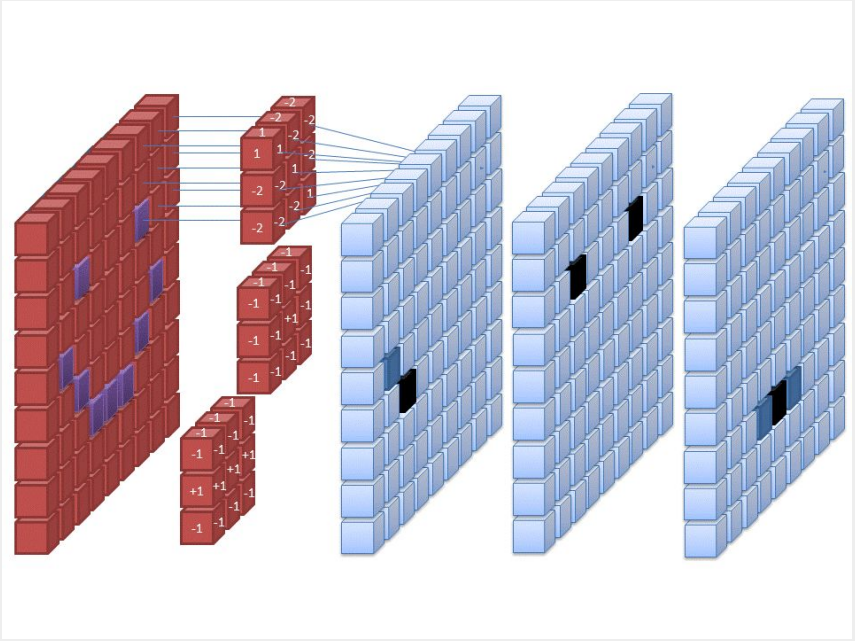
History, Intro, Human Vision

Convolutional Perceptrons

Pooling

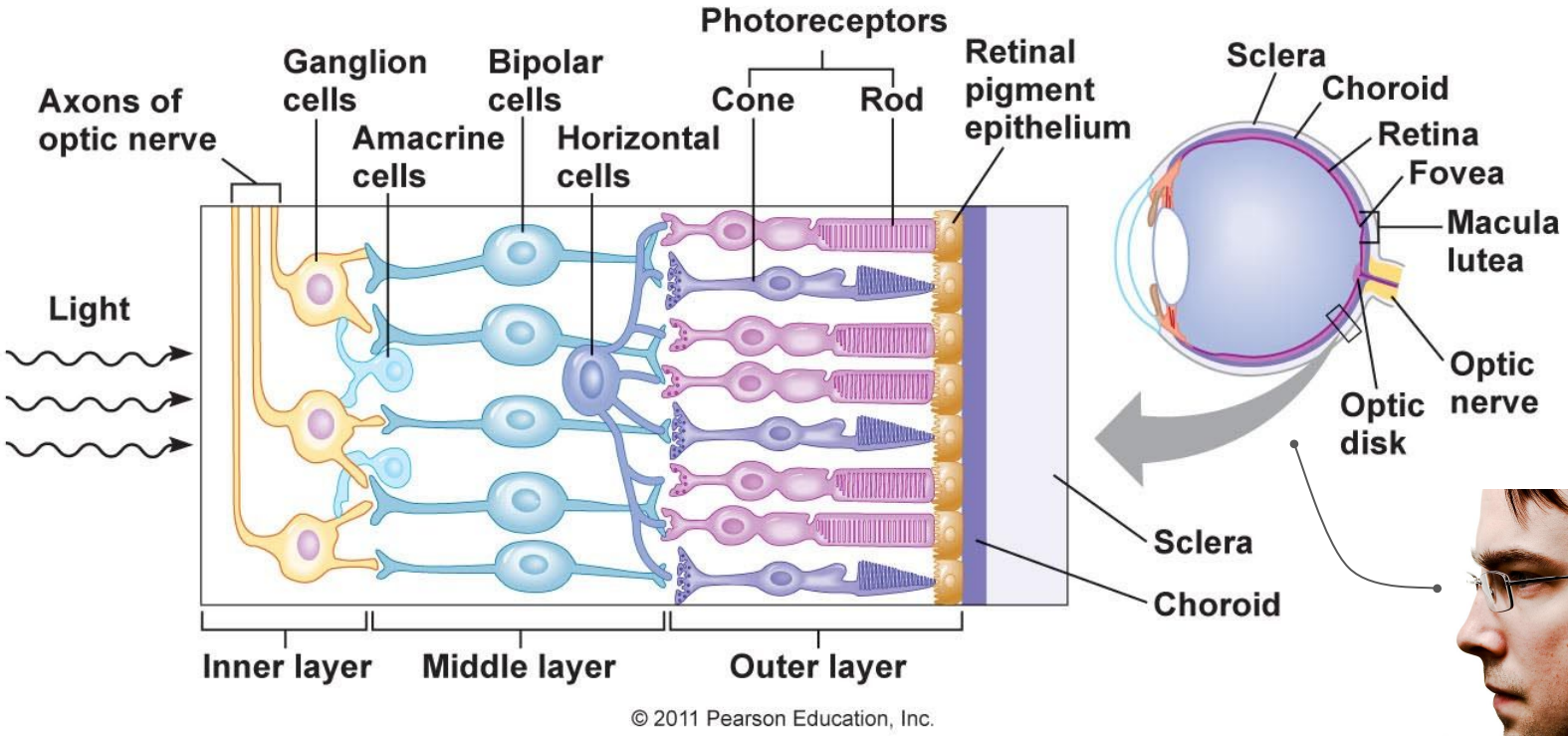
Activation maps

ConvNets changing the world



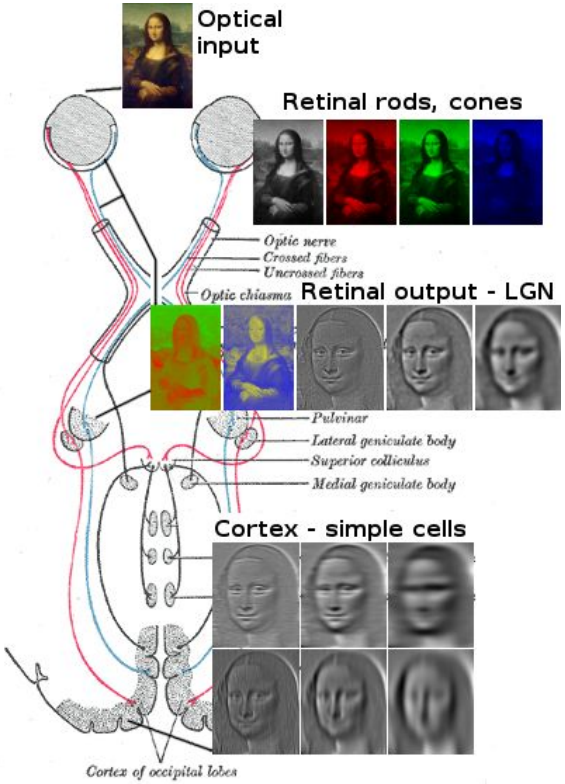
Retina | Layers, Rods and Cones

Recall



Visual Processing System | Hierarchy

Recall



V1
- Speed
- Direction
- Spatial frequency
- Temporal frequency
- Orientation
- Color

V2
(thick stripes)
- Speed
- Direction
- Spatial frequency
- Temporal frequency

MT
- Speed
- Direction
- Spatial frequency
- Temporal frequency
- Local/Global motion

MST
- Expansions
- Contractions
- Rotations
- Translations
- Spirals
- Optic Flow

Parietal Regions
- Heading
- Optic flow
- Self motion
- Multi-modal integration

V2
(thin stripes, interstripes)
-Edges
-Illusory edges
- Border ownership
- Color

V4
- Angles
- Curvature
- Perceived color
- Kinetic contours
- Motion

TEO/PIT
- Simple shapes

TE/AIT
- Complex shapes/ body parts
- Object recognition
- Object invariance

Visual Cortex

Huber & Wiesel, 1950s

Researched the levels of the visual cortex organization in cats.
(Warning: cat experiment images may be disturbing)

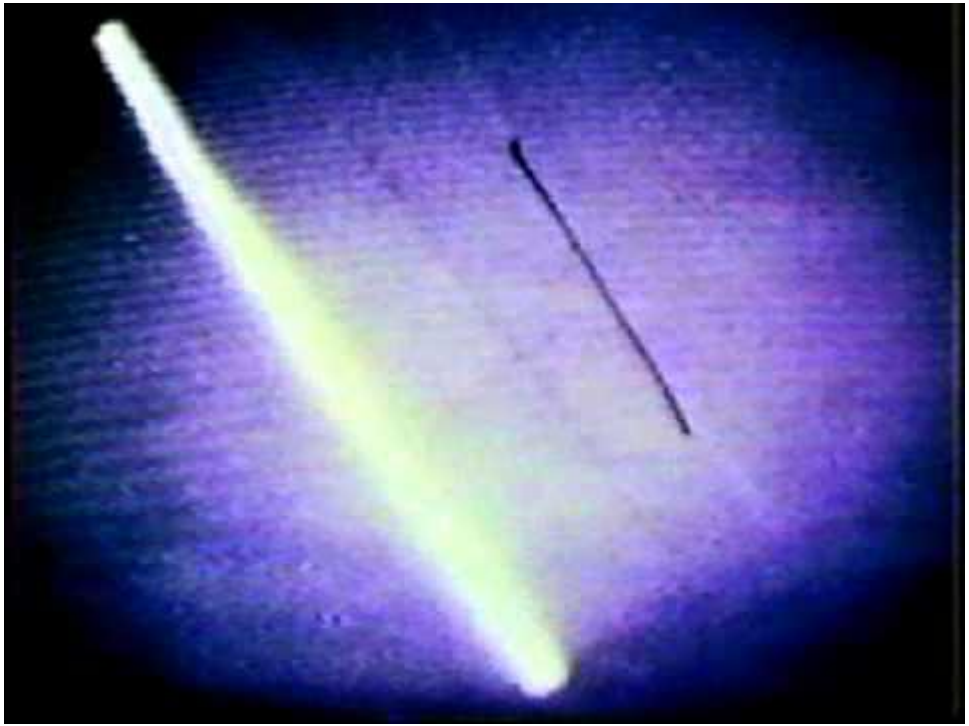
Found 3 types of cells:

- Simple cells: response to orientation
- Complex cells: Orientation and Motion
- Hypercomplex cells: Orientation, Motion and simple shape (circle, short line, etc.)

[Video]

Visual Cortex

They won a **nobel prize**!



[Video]

Visual Neural Networks

Y. LeCun and Y. Bengio, 1990s: researching the application of hierarchical organization of learnable convolutions in neural networks.

“LeNet-5”:

"Gradient-based learning applied to document recognition." [1998] (video next slide)

~>53,700 citations!!!!

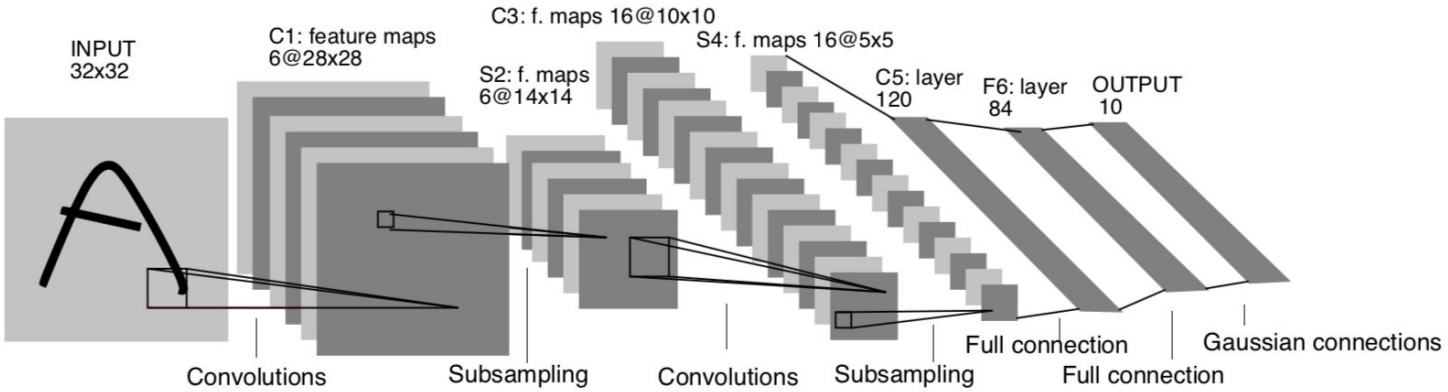


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Visual Neural Networks

Y. LeCun and Y. Bengio, 1990s: researching the application of hierarchical organization of learnable convolutions in neural networks.

This was the igniting spark of the “deep learning revolution” of the 2010s.

Demo, 1989

Convolutions

Recall

An operation on a close neighborhood of the center pixel.

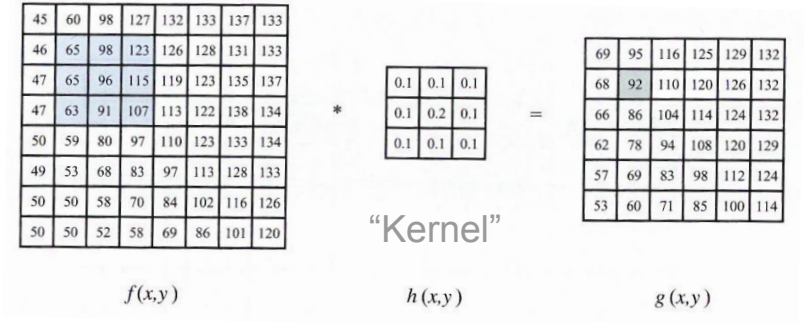
We'll start with linear operators:

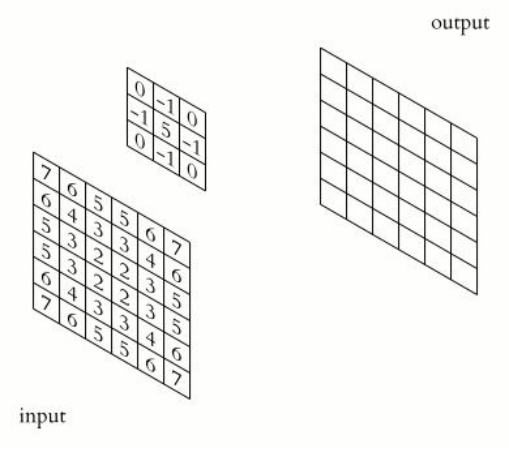
Correlation: $g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l)$

Convolution: $g = f * h$

$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l) = \sum_{k,l} f(k,l)h(i-k, j-l)$$

Linear Shift Invariant (LSI): Behave the same regardless of pixel location.





Example: Box Filter | Moving Average

Recall

*

1/9

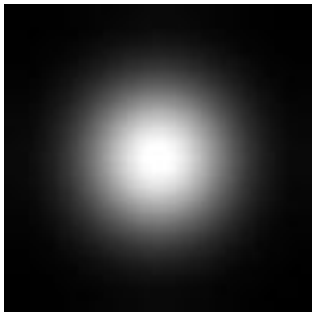
1	1	1
1	1	1
1	1	1

=

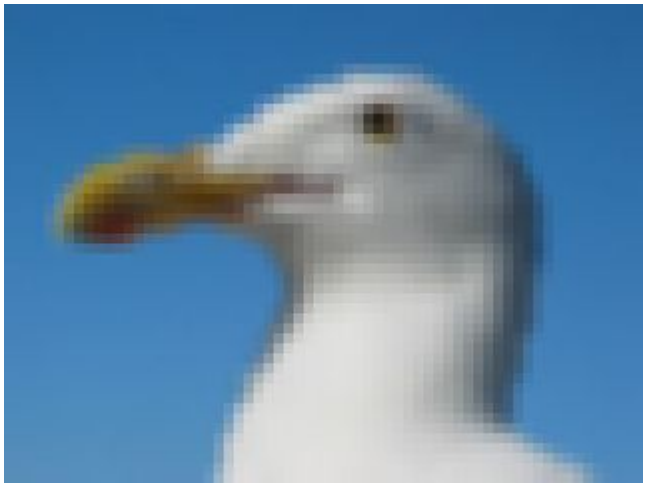
Recall

Example: Gaussian Filter | Blur

*



=



$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Example: Sharpening

Recall

*

-1/9	-1/9	-1/9
-1/9	2 - 1/9	-1/9
-1/9	-1/9	-1/9

=

Convolutional Neural Networks

The big ideas in [LeCun and Bengio 1998]:

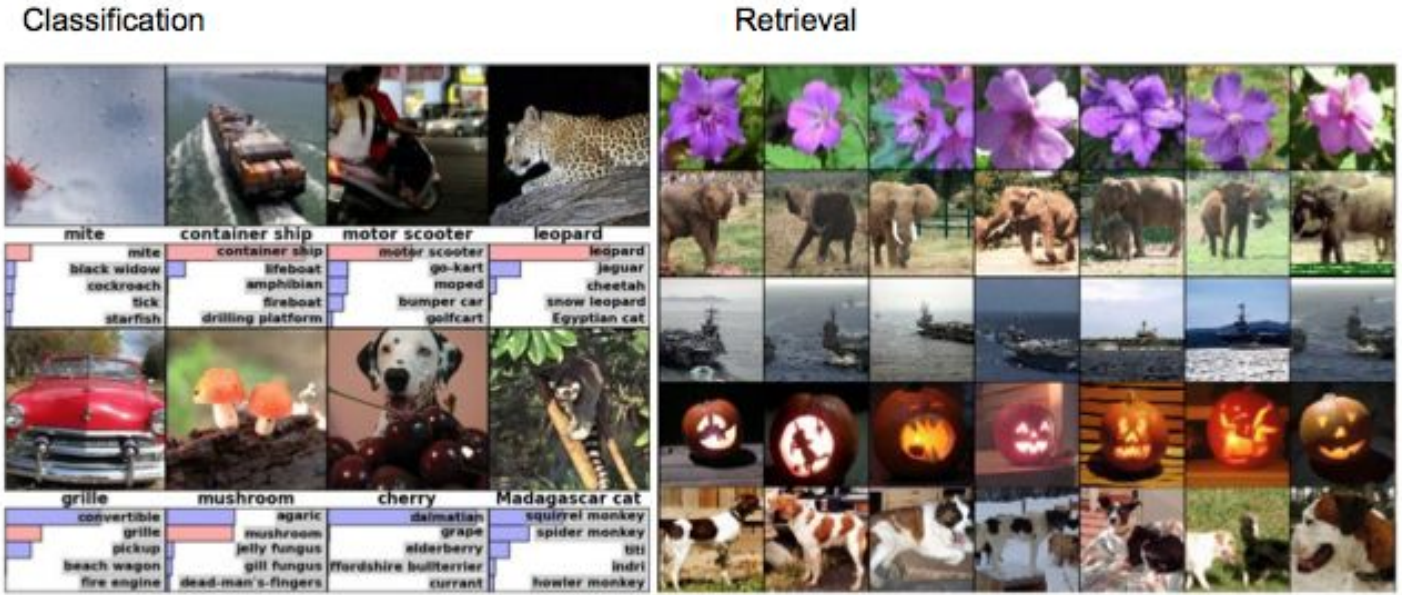
- From locally (fully) connected neurons to “shift invariant” convolutional neurons.
- “weight sharing”
- Spatial sub-sampling: “Pooling”

Similar initial ideas, e.g. in Fukushima’s “Neocognitron” [1980], predated backprop and gradient-based training was virtually impossible.

Convolutional Neural Networks

Fast-forward to today: ConvNets are everywhere

[Fei-Fei Li]



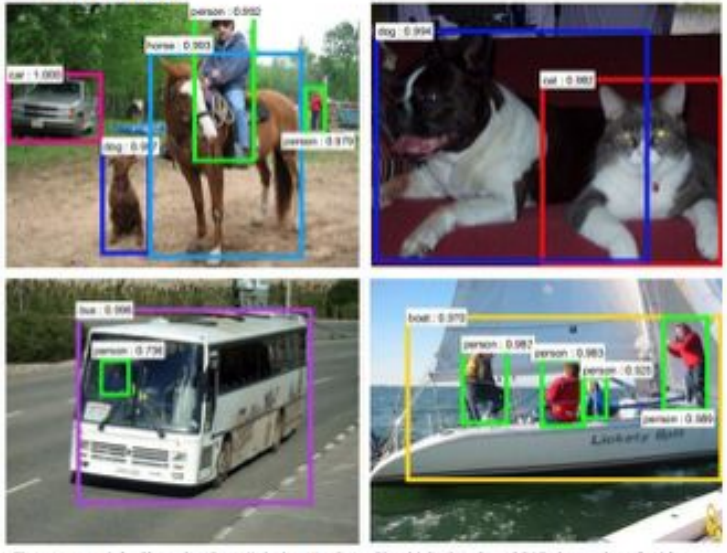
Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Convolutional Neural Networks

Fast-forward to today: ConvNets are everywhere

[Fei-Fei Li]

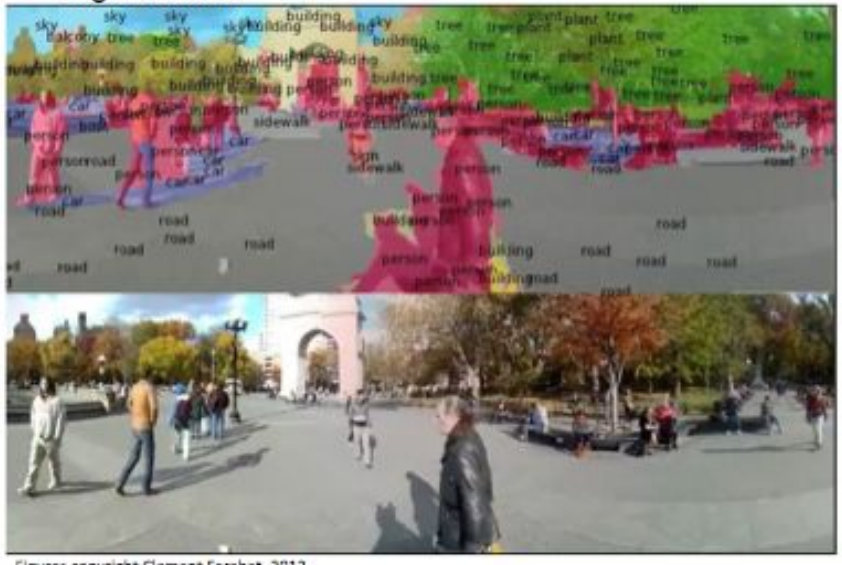
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



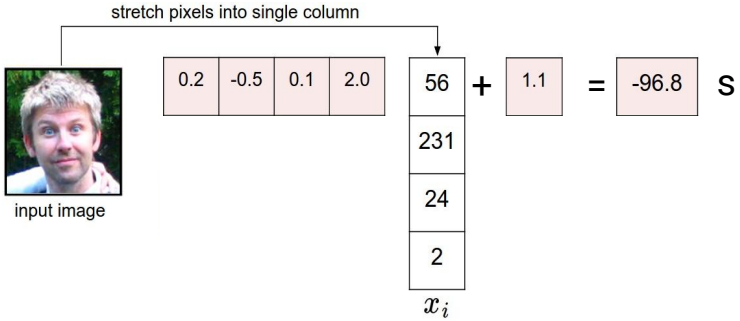
Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

Learning Convolutions

Learned Convolutions

So far we've seen "stretch pixels to a vector", and our weights were also the size of the vector.



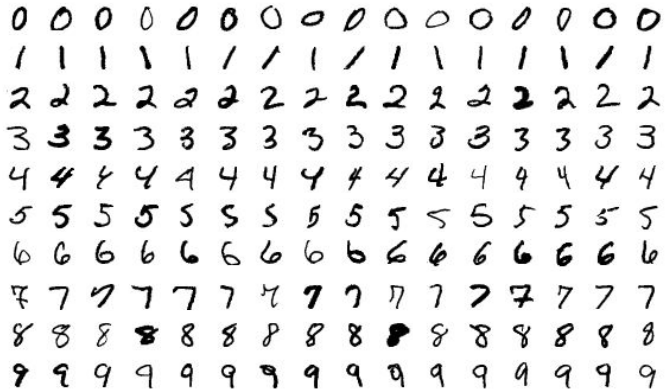
Perform dot product of input and weights.

But that doesn't preserve spatial information well. In fact - it "mixes everything together".

This may not be such a bad thing!

We find relationships between all pixels in the image at once.

Works well for simple problems such as MNIST



Learned Convolutions

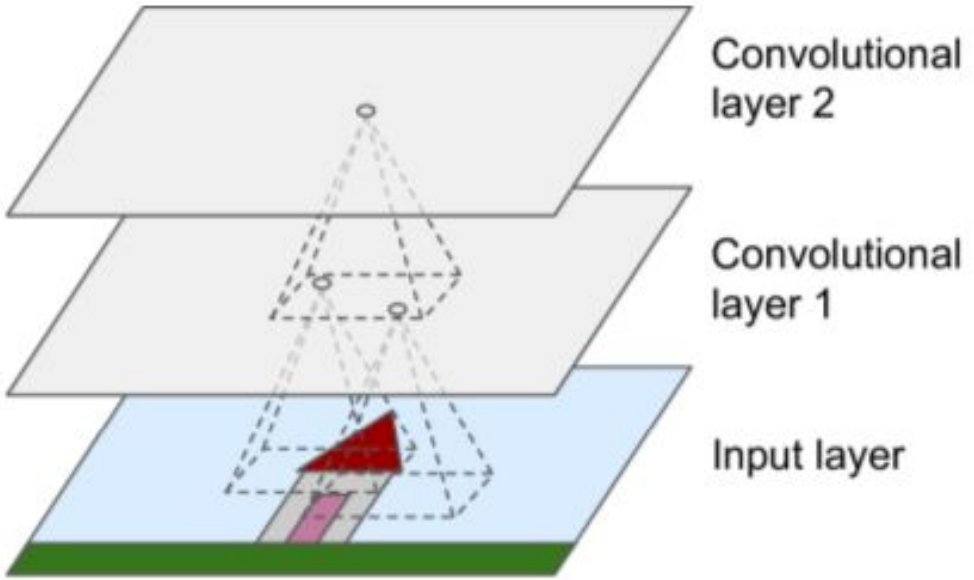
Learned Convolutions

"Fully Connected Layers": neurons are connected (Receptive Field) to all pixels in the input layer.

"Convolutional Layers": neurons have just a NxN receptive field, which is applied spatially across all the input pixels, **sharing the weights**.

"Shift invariance" of convolutions comes into play. Visual structures can be recognized *anywhere in the image*.

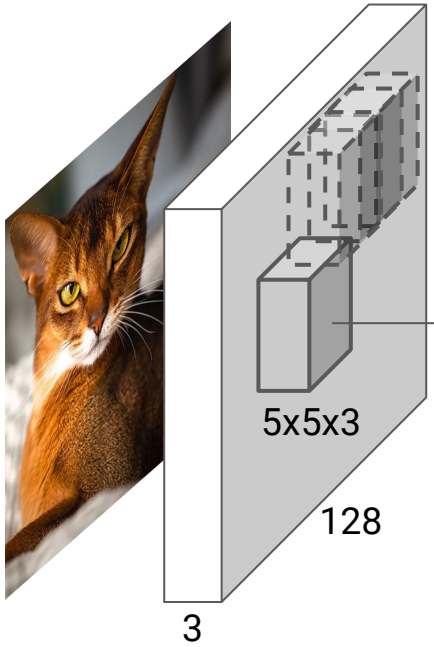
Conv layers reduce the number of trainable parameters by many orders of magnitude...



[Géron]

Convolutional Layer

128x128 Image, e.g. RGB (x3)



Convolve the image with the filter (multiply at every position)

“Filter”, size 5x5x3+1

Dot product

A single number

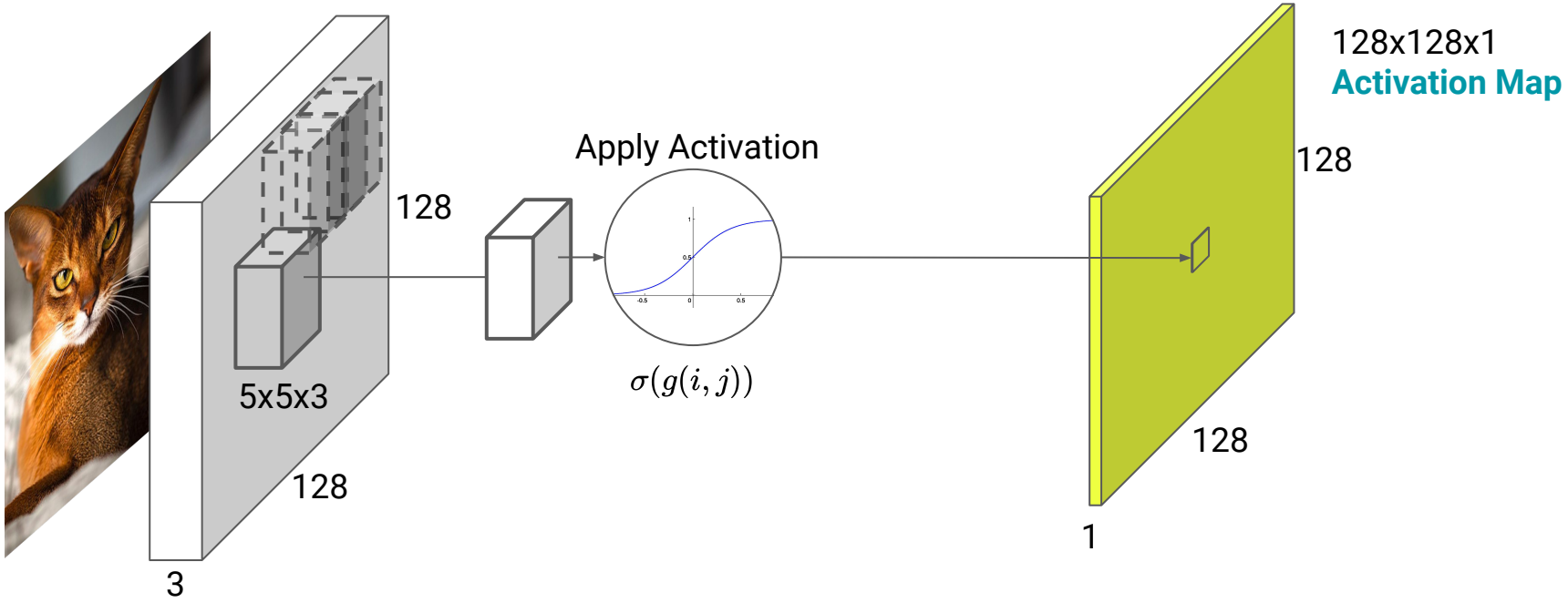
$$g(i,j) = \sum_{k,l,m} f(i-k, j-l, m)h(k,l,m) + b$$

$$g(i,j) = f[i:i+k, j:j+l] \cdot h + b = Wx + b$$

Learnable parameters: 5x5x3+1 = 76

Convolutional Layer

128x128 Image, e.g. RGB (x3)

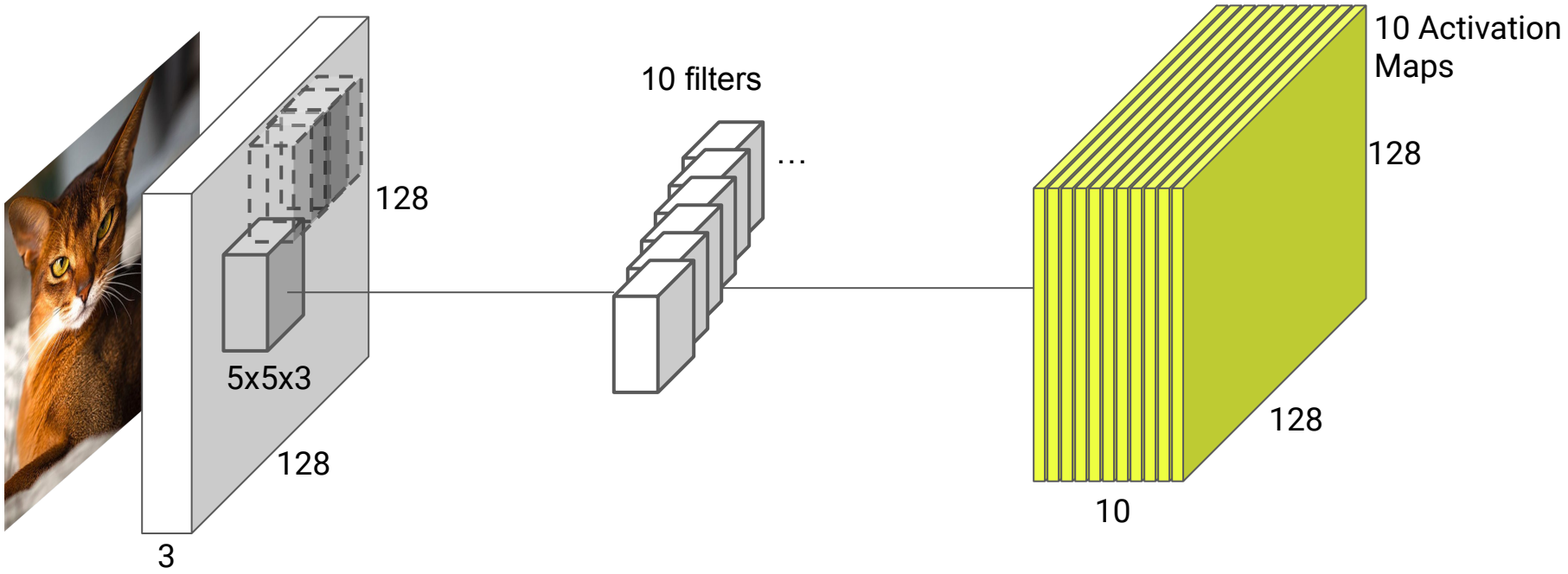


With “Same” padding, output size is maintained

Convolutional Layer

$$\text{Parameters} = (K \times K \times D_{\text{in}} + 1) \times N$$

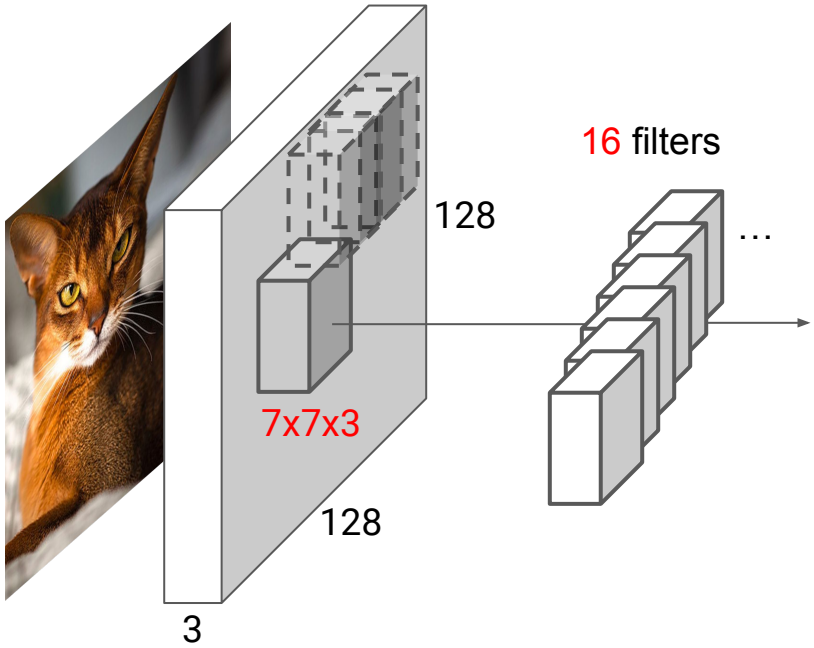
128x128 Image, e.g. RGB (x3)



Learnable parameters: (5x5x3+1)x10 = 760

Usually we learn many convolution kernels at each layer...

Convolutional Layer



$$\text{Parameters} = (K \times K \times D_{\text{in}} + 1) \times N$$

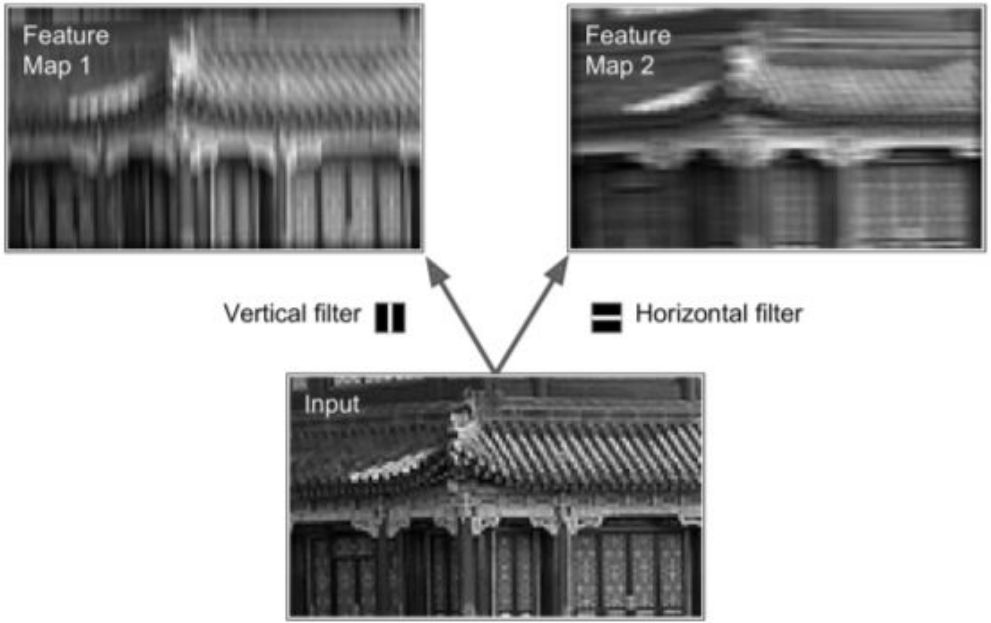
How many learnable parameters? Don’t forget the bias!

Learned Convolutions

Each Conv layer learns a set of weights - (multiple) convolution kernels.

The convolution operation is same as the linear perceptrons: Wx_i

Backpropagation adjusts the weights, and the layer learns the best convolution to fit the loss function minimization.



[Géron]

Feature Maps

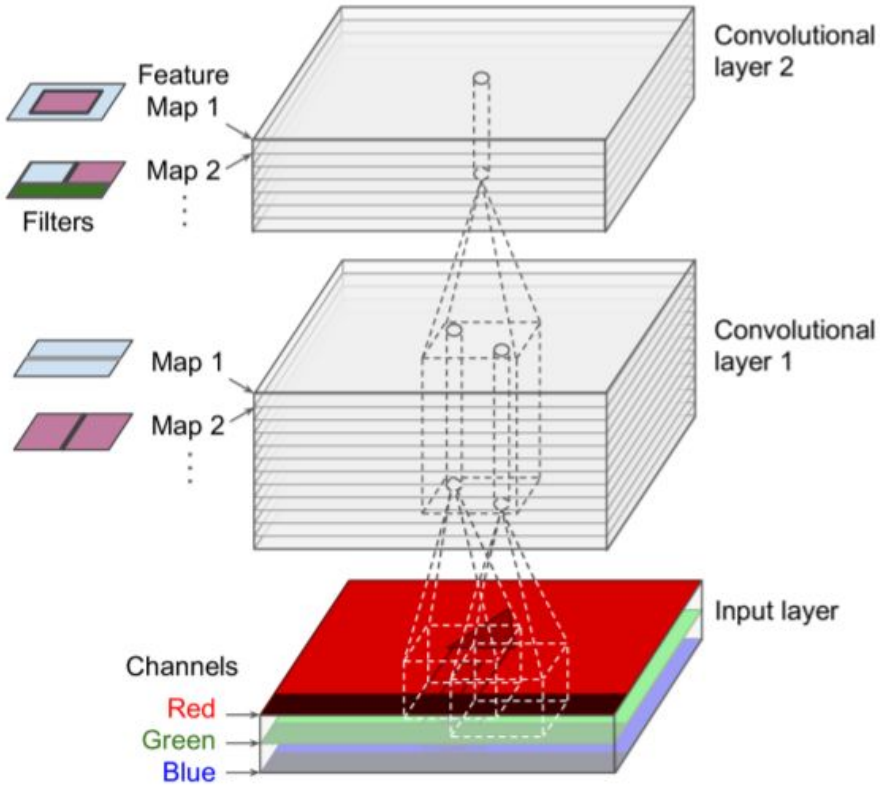
Each neuron contributes a convolved output image (local feature extraction).

The result of a convolutional layer is a stack of all those maps.

Result: A visual structure (e.g. “face”) can be detected anywhere in the image.

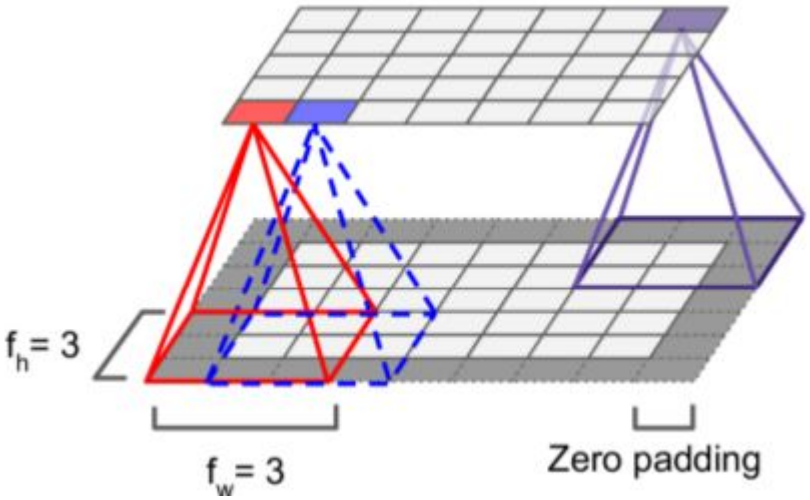
Successive layers apply to *all lower layers*. This can quickly blow up in dimension!

“Activation map”: Applying activation (sigmoid, tanh, ReLU,...) on the feature map.

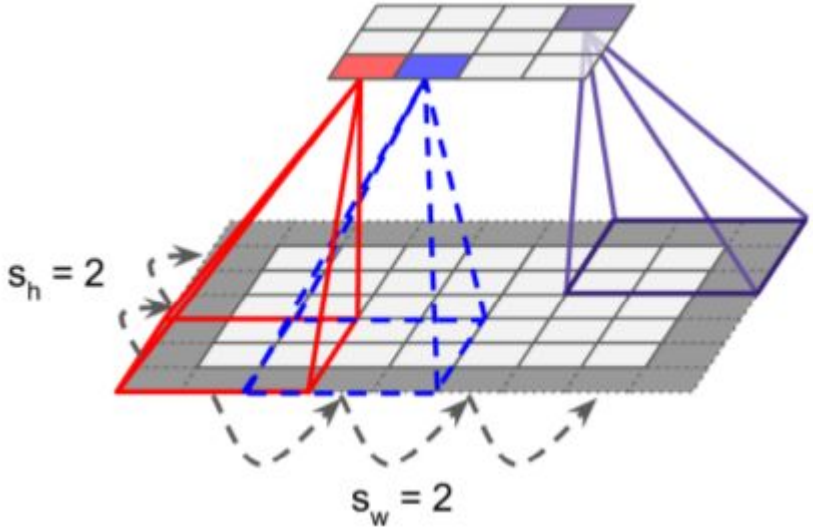


[Géron]

Padding and Stride



Border pixels are missing values for receptive field of 3.
Output layer will be smaller, or we add padding.



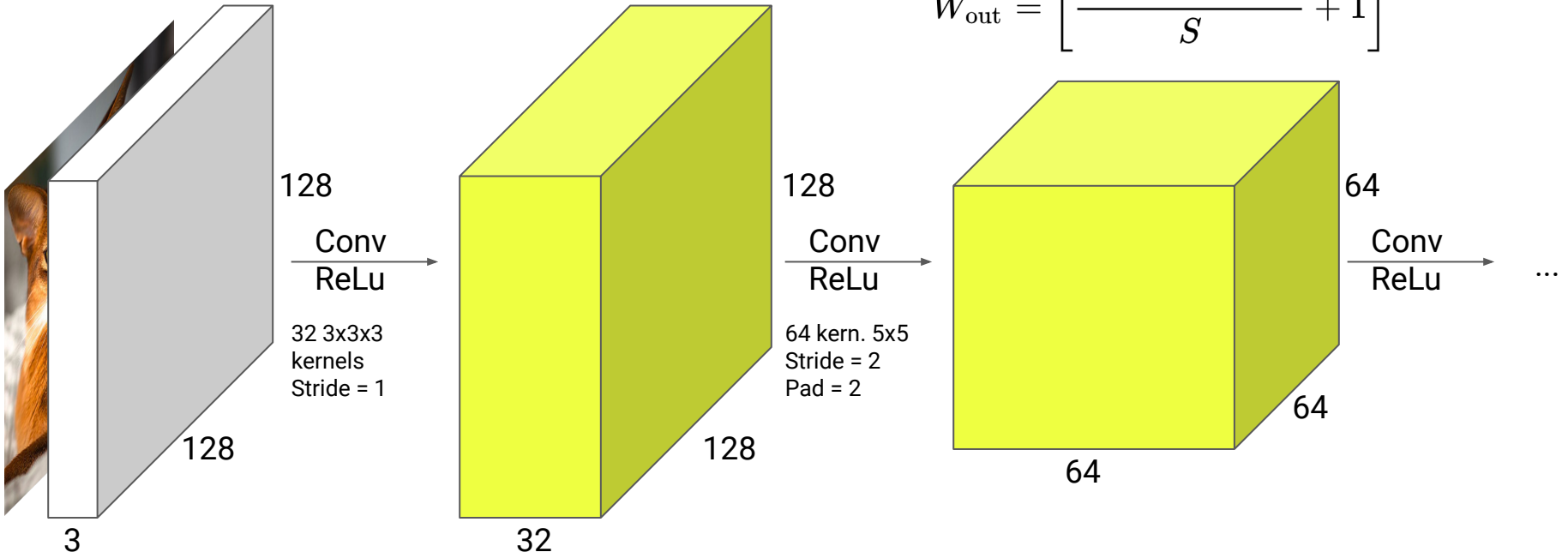
Stride can be used to reduce the size of the output of convolutional layers.

[Géron]

Fully Convolutional Networks

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - K + 2P}{S} + 1 \right\rfloor$$

$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - K + 2P}{S} + 1 \right\rfloor$$



Fully Convolutional Networks

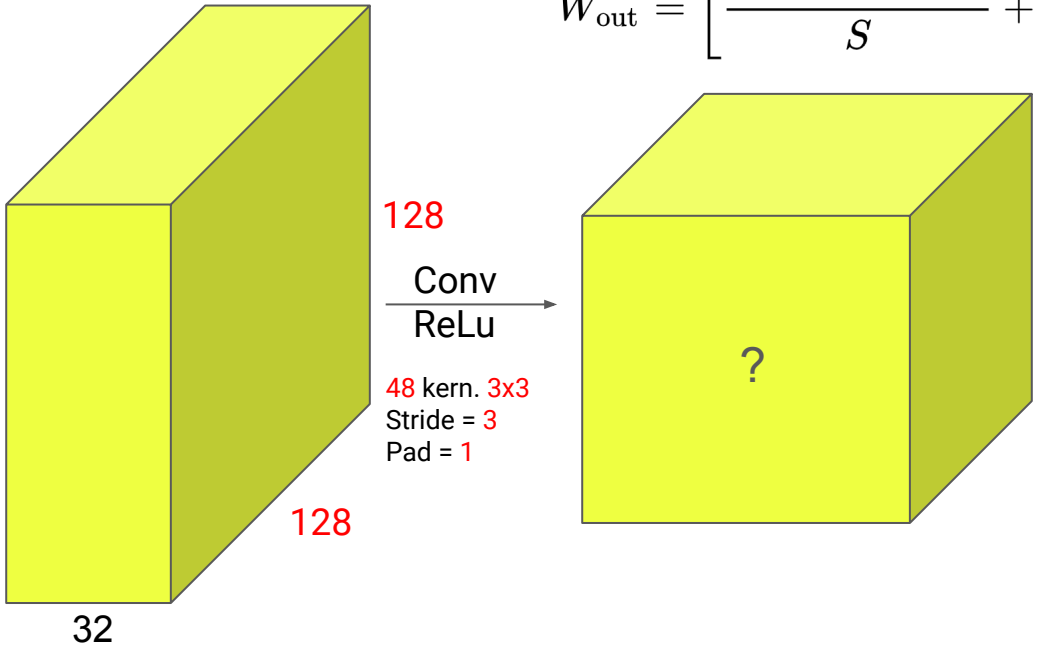
Asking

You

What's going to be the output shape of this Conv layer?

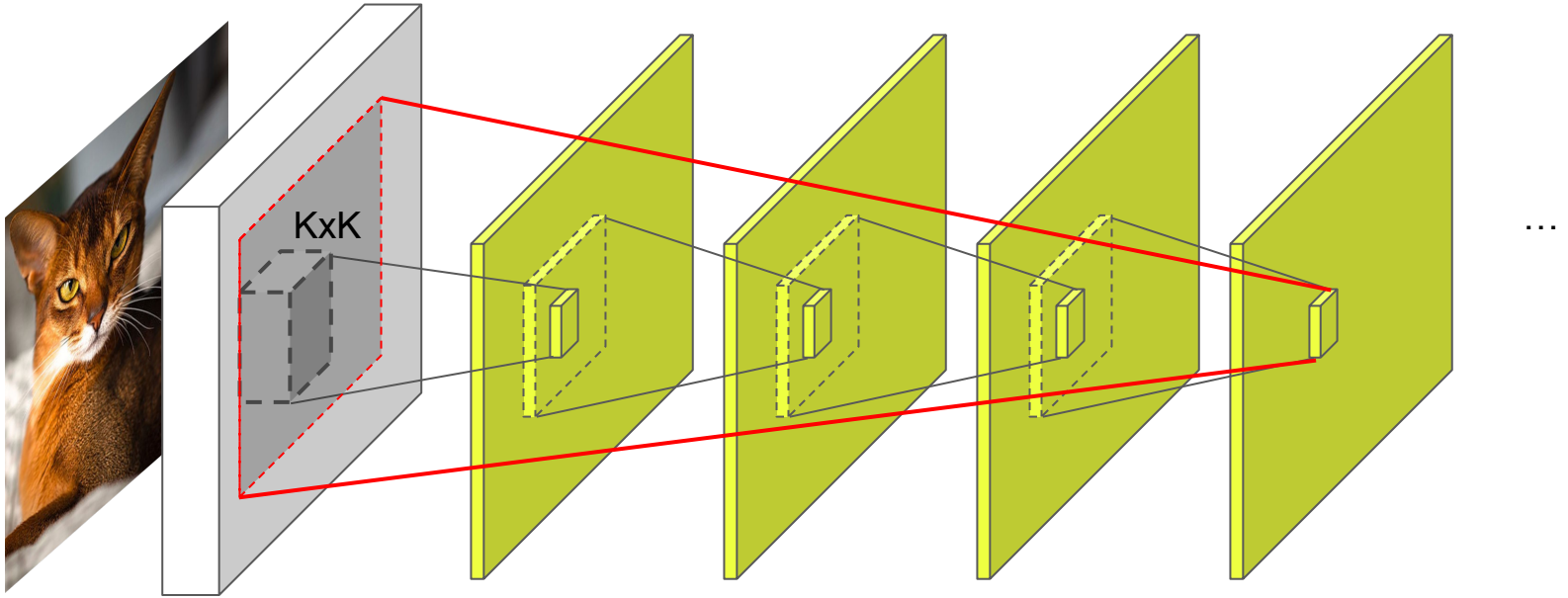
$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - K + 2P}{S} + 1 \right\rfloor$$

$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - K + 2P}{S} + 1 \right\rfloor$$



Receptive Field

Each conv layer adds K-1 to the receptive field of a “pixel” in the activation map.
Problem for very large images: We need a lot of layers to “see” the entire image…



Pooling Layers

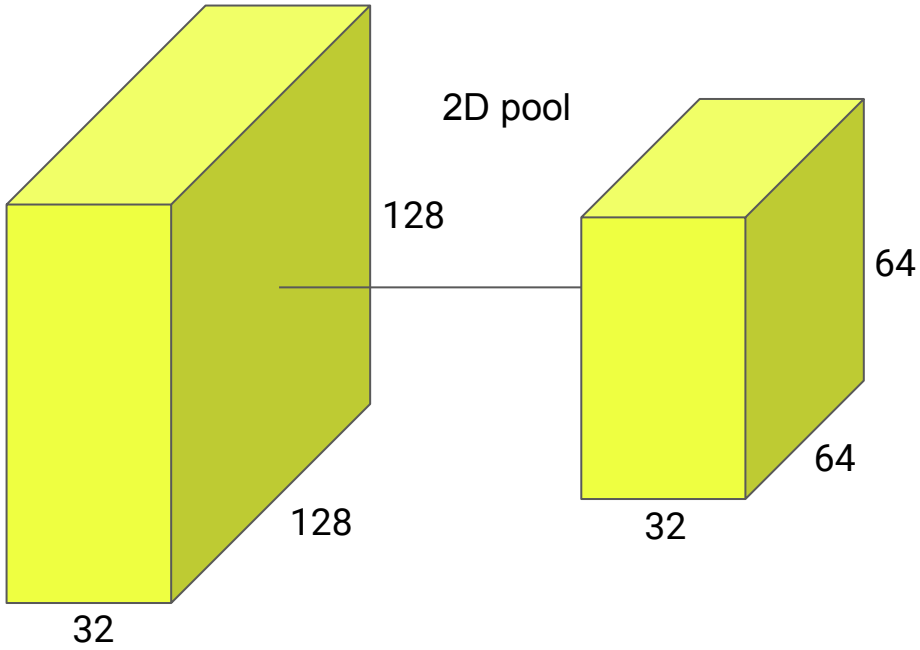
Pooling allows us to “downsample” inside the network and efficiently increase the receptive field of deep convolutional neurons.

Several (design) choices for pooling: (Any aggregation function works)

- Max/Min (most common)
- Average (very useful)
- Sum? Count?? Stdev???

No learnable parameters.

Back gradients are defined by the operation.



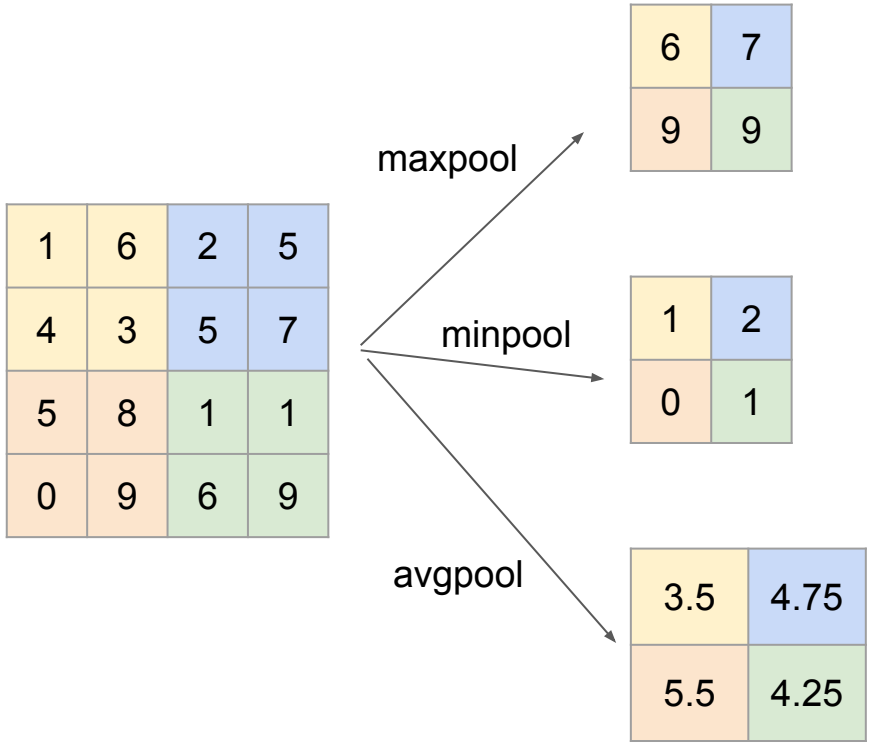
Pooling Layers

We can also control the pooling: (like conv's)

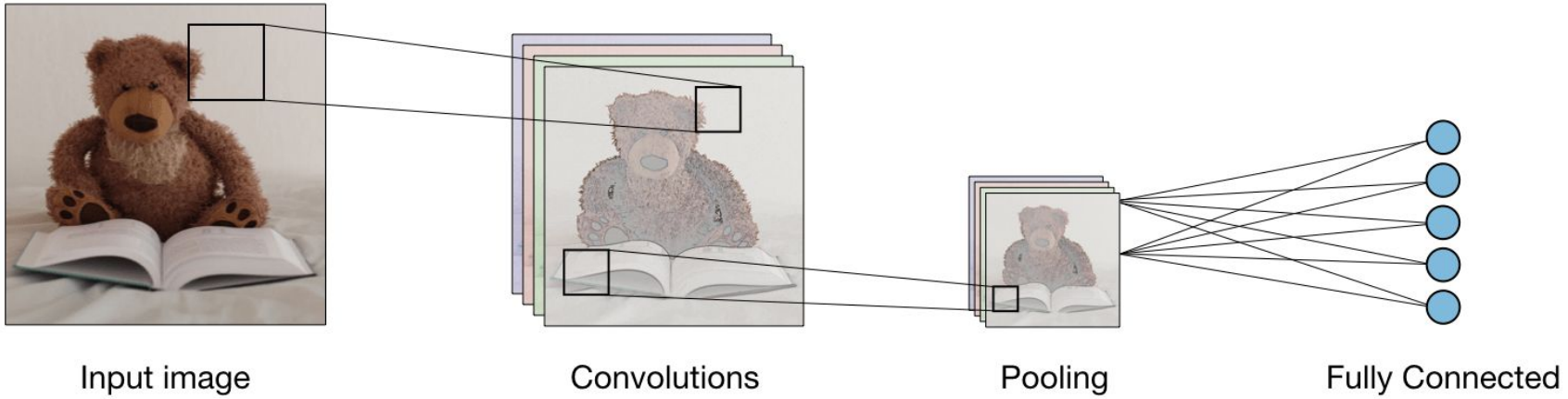
- Stride
- "Kernel" size

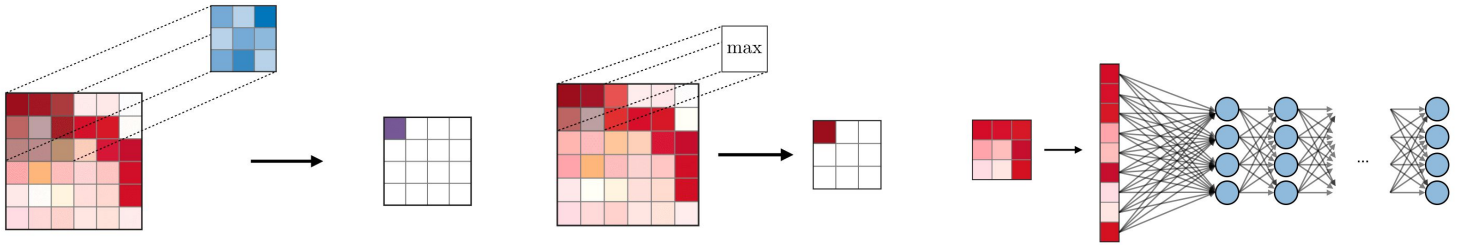
⚠️⚠️⚠️ Pooling (also strided conv.) does not respect the sampling theorem! And injects aliasing…

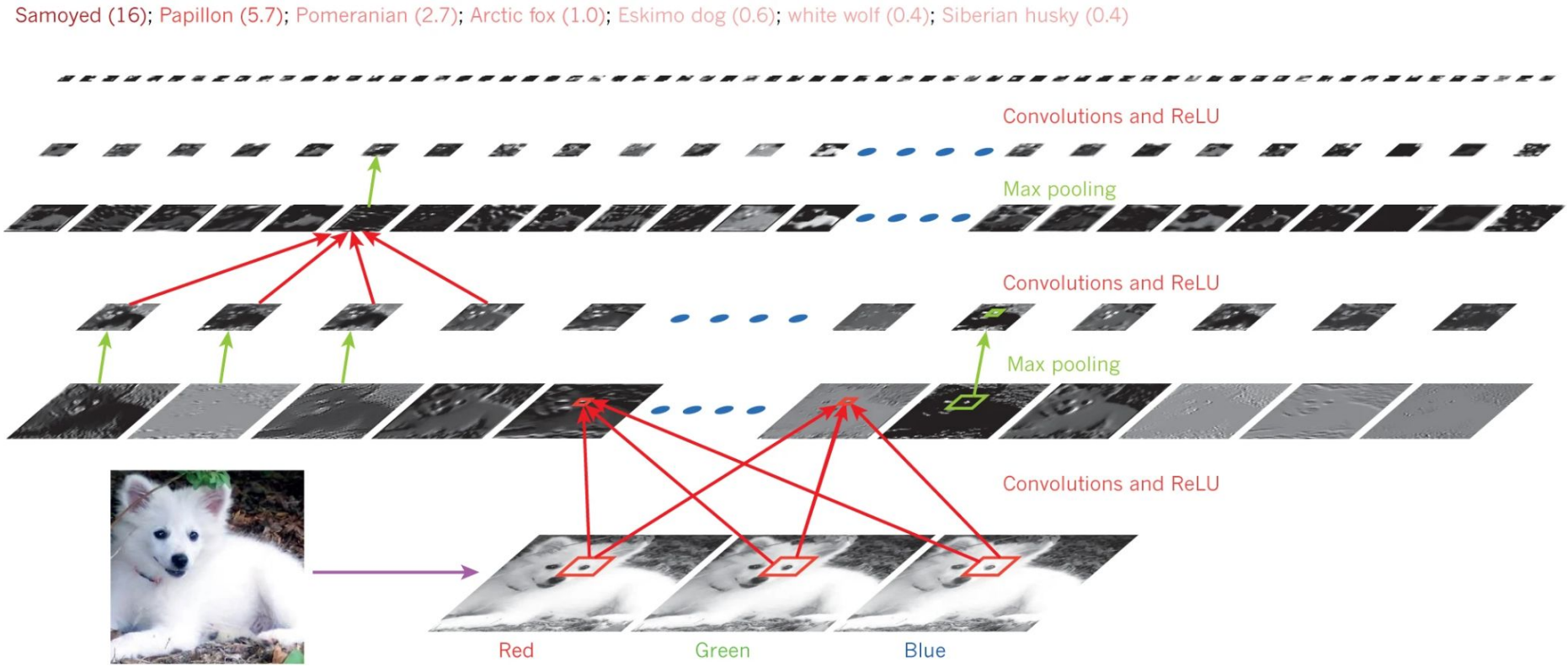
https://richzhang.github.io/antialiased-cnns/



Convolutional Networks







[LeCun, Bengio and Hinton 2015]

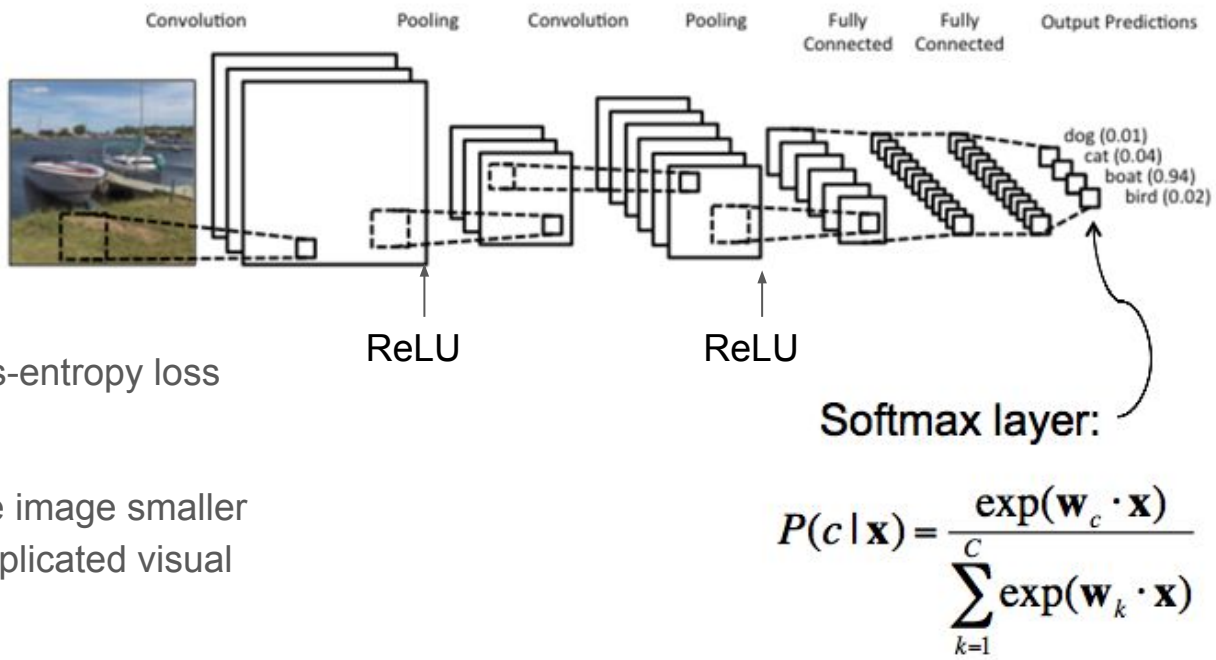
Convolutional Networks

Typical Classification CNN:

[Conv-Pool-ReLU] x N
+ FC x M
+ Decision layer

The last layer applies the cross-entropy loss (Softmax).

Each conv-pool pair makes the image smaller but deeper, learning more complicated visual constructs.



[Lazebnik]

Convolutional Networks

Creating a vanilla CNN is very simple:

```
model = keras.models.Sequential()
model.add(keras.layers.InputLayer(input_shape=(64, 64, 3)))
for i in range(3):
    model.add(keras.layers.Conv2D((i+1) * 16, (3, 3), activation='relu'))
    model.add(keras.layers.MaxPooling2D((2, 2)))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(3, activation='softmax'))
```

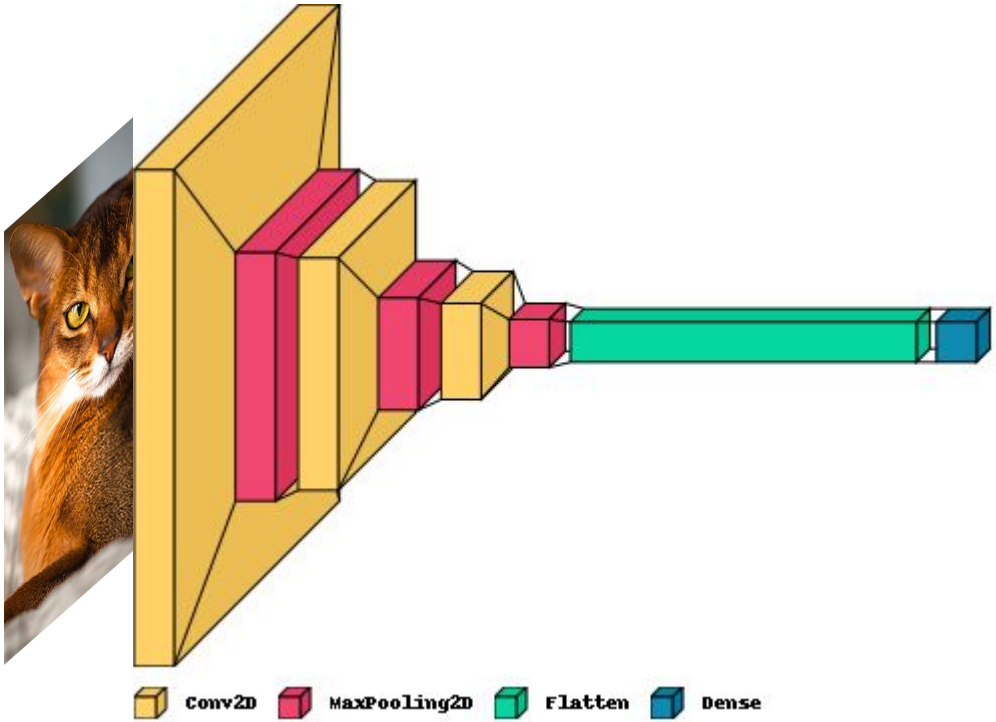
Convolutional Networks

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 62, 62, 16)	448
pool0 (MaxPooling2D)	(None, 31, 31, 16)	0
conv2d_21 (Conv2D)	(None, 29, 29, 32)	4640
pool1 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_22 (Conv2D)	(None, 12, 12, 48)	13872
pool2 (MaxPooling2D)	(None, 6, 6, 48)	0
flatten_6 (Flatten)	(None, 1728)	0
dense_6 (Dense)	(None, 3)	5187

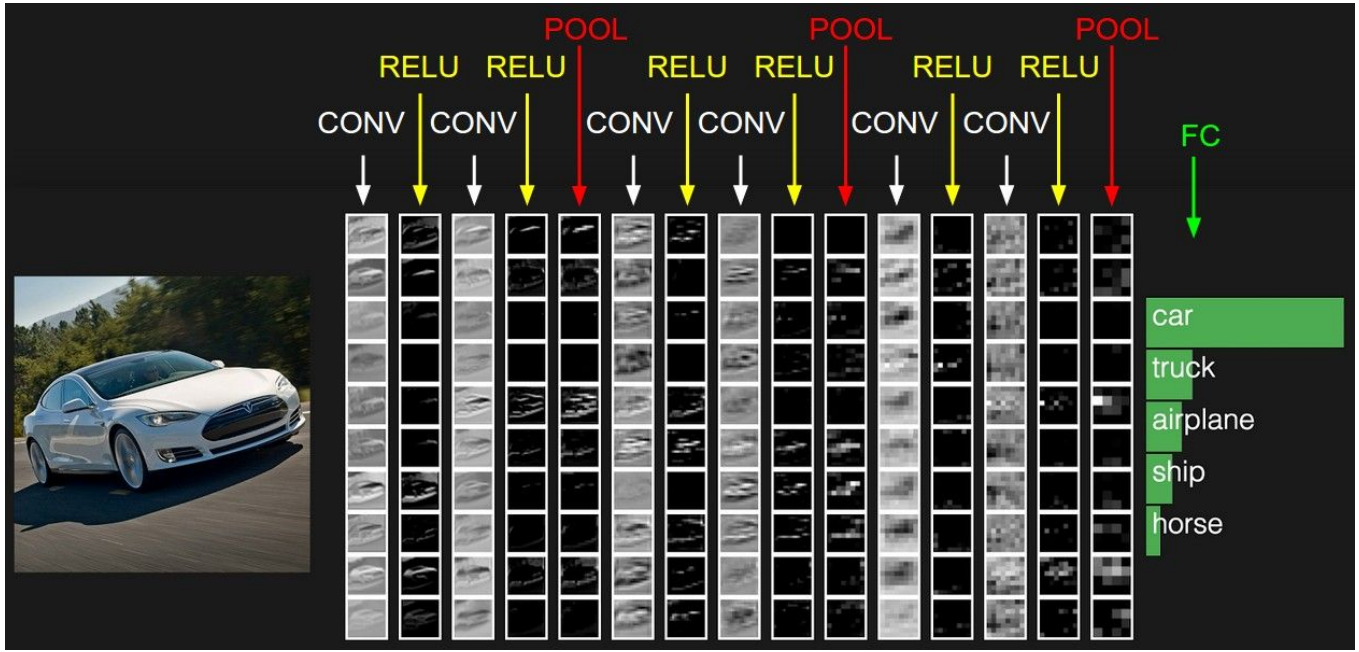
Total params: 24,147

Trainable params: 24,147

Non-trainable params: 0



What Does A CNN “See”?

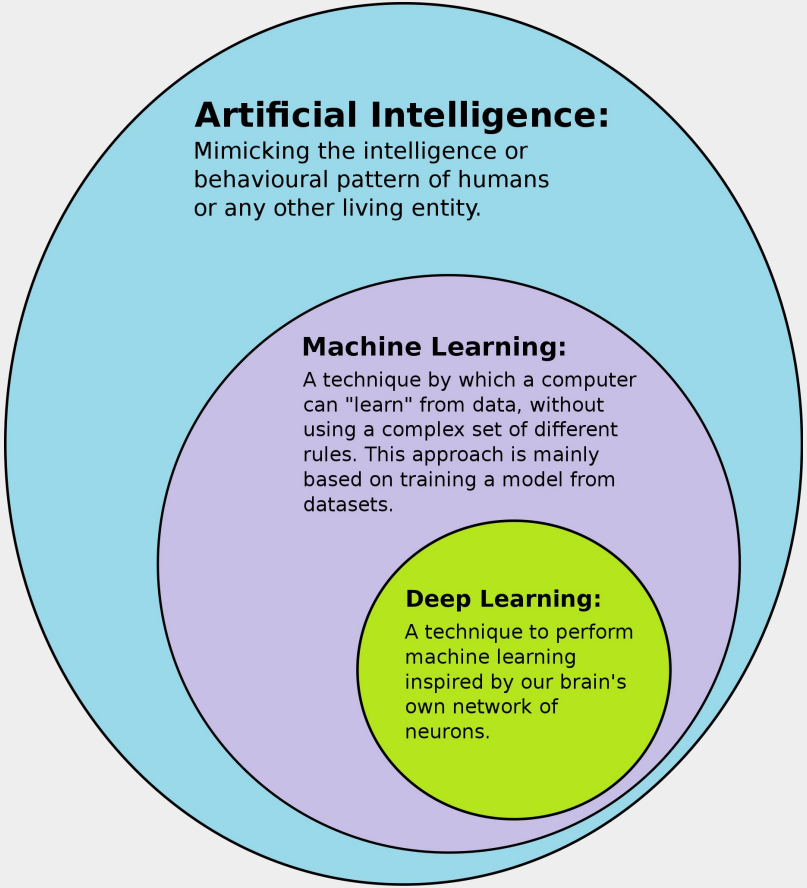


[DEMO]

[Visualization]

[Activation Atlas]

Deep Learning



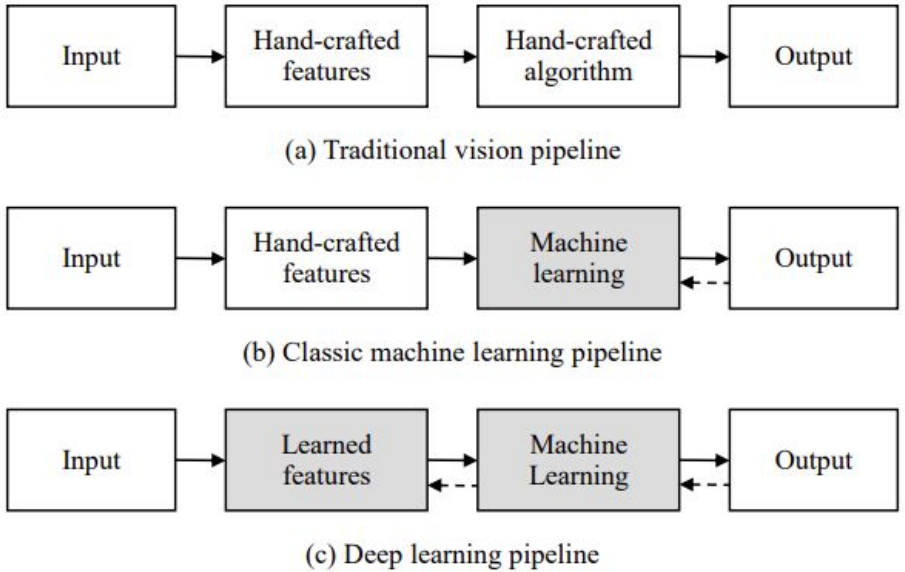
Deep Learning

Deep convnets have ushered in the era of Deep Learning.

Instead of extracting features with e.g. feature extractors (e.g. SIFT) or relying on intuition - the network learns what it needs to learn to describe the visual phenomena.

Easier as a consumer. Shifts the focus to data.

“Deep Learning for AI” [LeCun, Bengio, Hinton 2021]



Famous Architectures

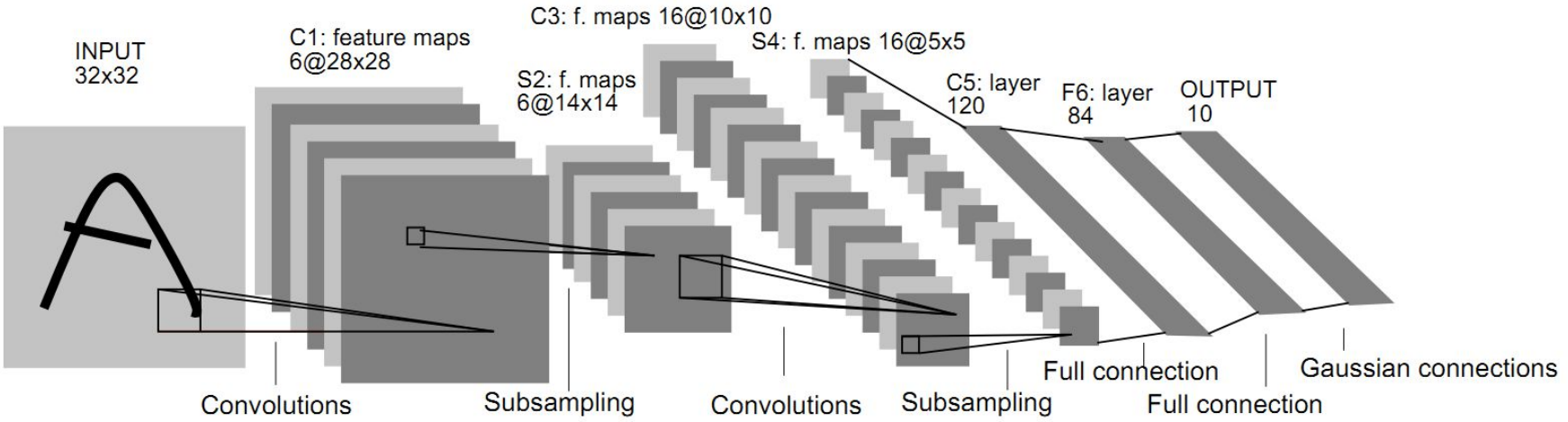
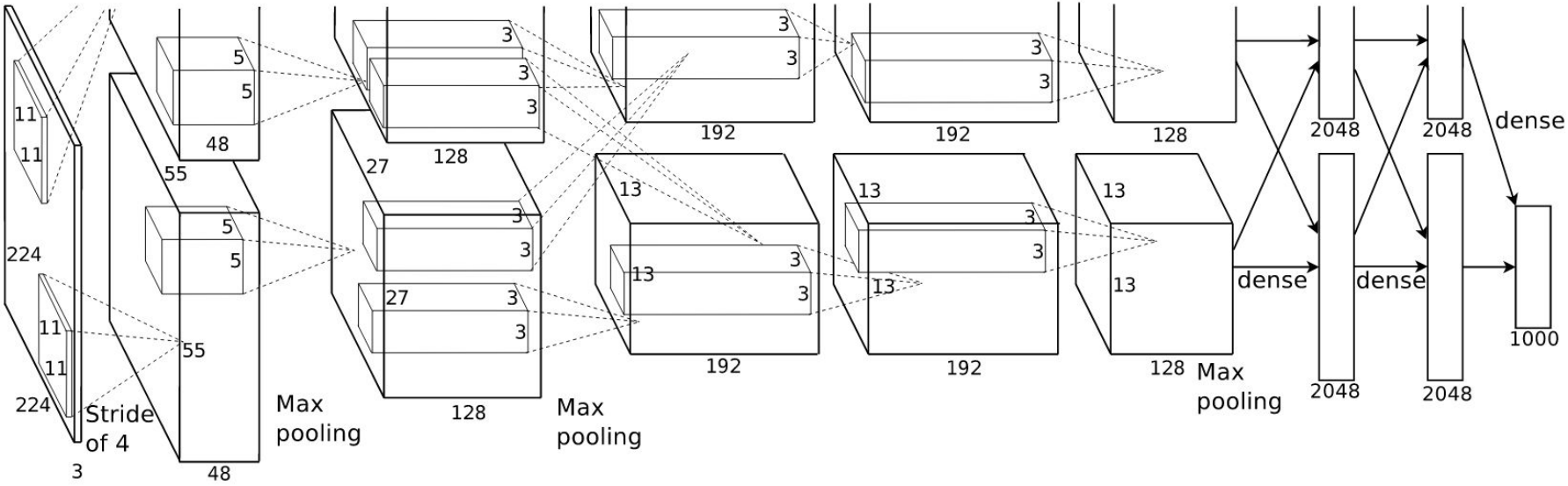


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

"LeNet", LeCun 1998. Achieved 98% accuracy on MNIST (handwritten digit classification)

Famous Architectures



"AlexNet", Krizhevsky + Hinton 2010. Achieved ~15% top-5 error rate on ImageNet Large Scale Visual Recognition challenge (ILSVRC). Beat the 2nd runner up by more than 10%!

ILSVRC Image Classification (CLS) Task

Steel drum

[ILSVRC 2017]

1000 object classes 1,431,167 images CLS-LOC

ILSVRC Image Classification (CLS) Task

Steel drum

[ILSVRC 2017]

Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

✔

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

"Top-5" acc

$$\text{Error} = \frac{1}{100{,}000} \sum_{\substack{\text{100,000} \\ \text{images}}} 1[\text{incorrect on image i}]$$

Famous Architectures

[Simonyan et al 2014] VGG 16,19

“Very Deep Convolutional Networks For Large-Scale Image Recognition”

16-19 convolutional layers

Not a breakthrough in architecture, but served as a solid baseline for many years.

Has >100M parameters.

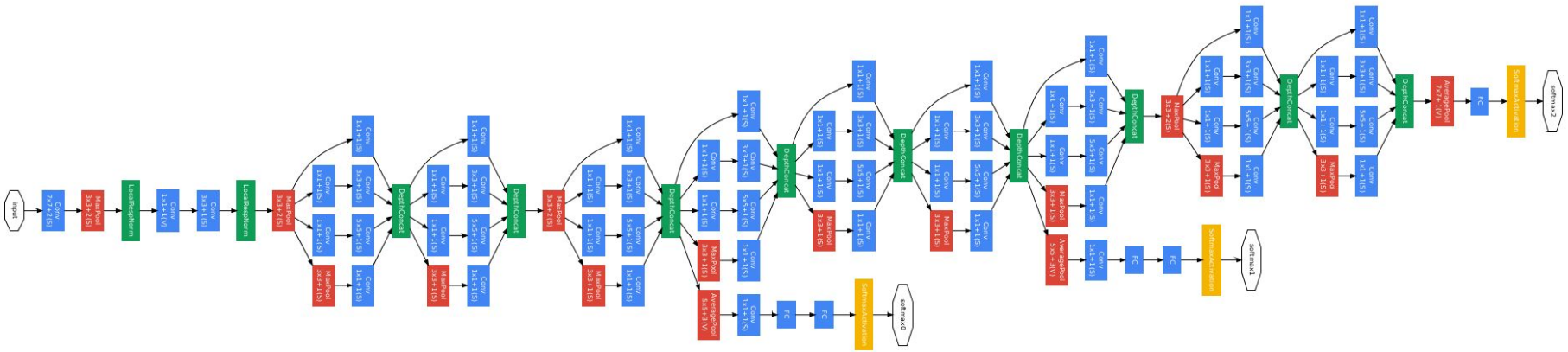
Achieved ~%7.5 top-5 error on ILSVRC 2012.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

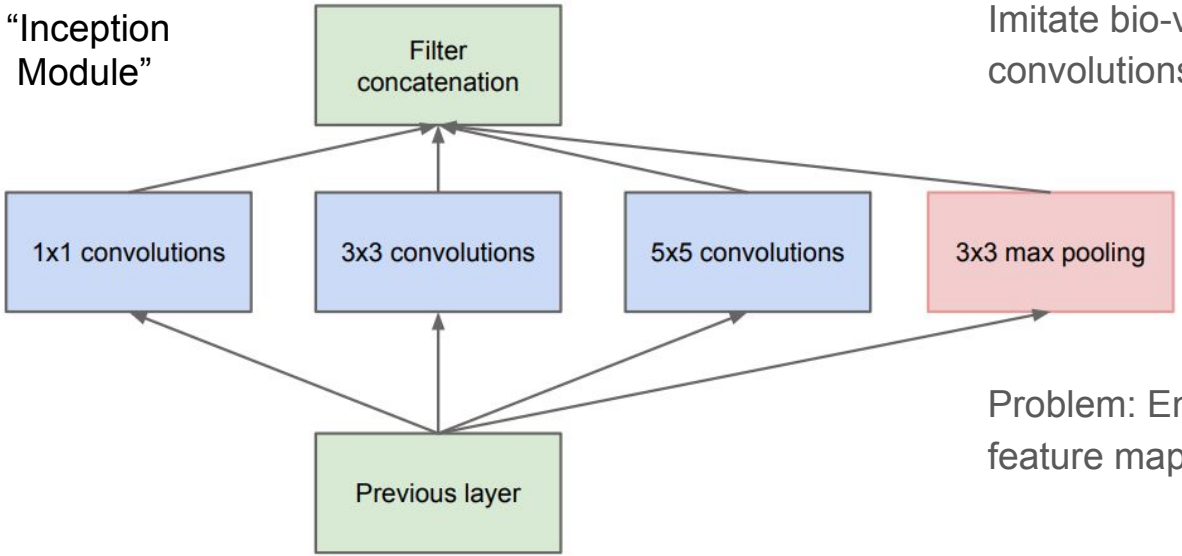
Famous Architectures



"GoogLeNet", Szegedy et al 2015. Has 22 layers. Won the ILSVRC 2014, achieved 6.67% top-5 err.

Famous Architectures

"Inception Module"



Imitate bio-vision by applying multiple sized convolutions to the same location.

Problem: Ends up with an enormous amount of feature maps.

"GoogLeNet", Szegedy et al 2015

Famous Architectures

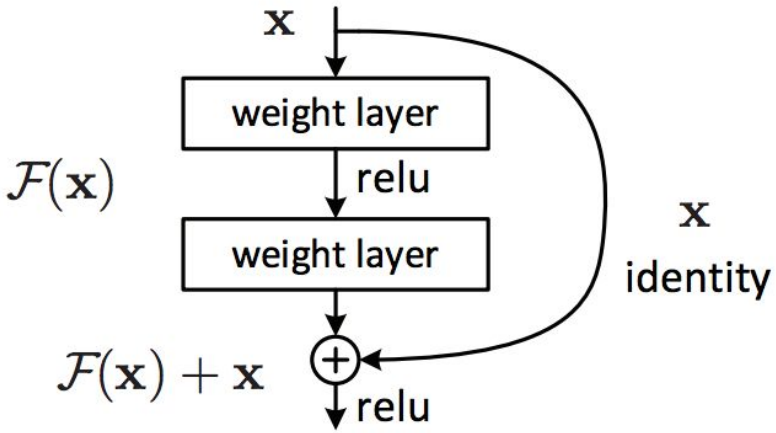


Figure 2. Residual learning: a building block.

"Residual"

Model: f(x) = h(x) - x, instead of f(x) = h(x)

Skip connections: send the signal across learning units, so even if they are stuck the signal still flows downwards.

"ResNet" (Residual Network), He et al. 2016. Achieved **3.57%** err on ILSVRC 2015 with 152 layers.

Famous Architectures

[Howard et al 2017] MobileNet

[Sandler et al 2018] MobileNet V2

[Howard et al 2019] MobileNet V3

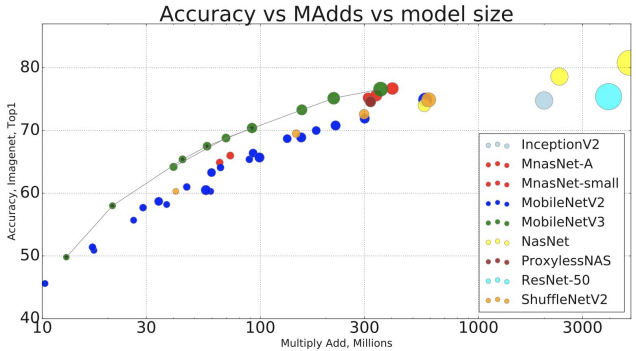
Very small (O(1m)) network. A lot of tricks to save on parameters. De facto backbone for resource constrained applications.

MobileNetV1 was able to replicate VGG16 acc on ImageNet, while 1/100s of its size.

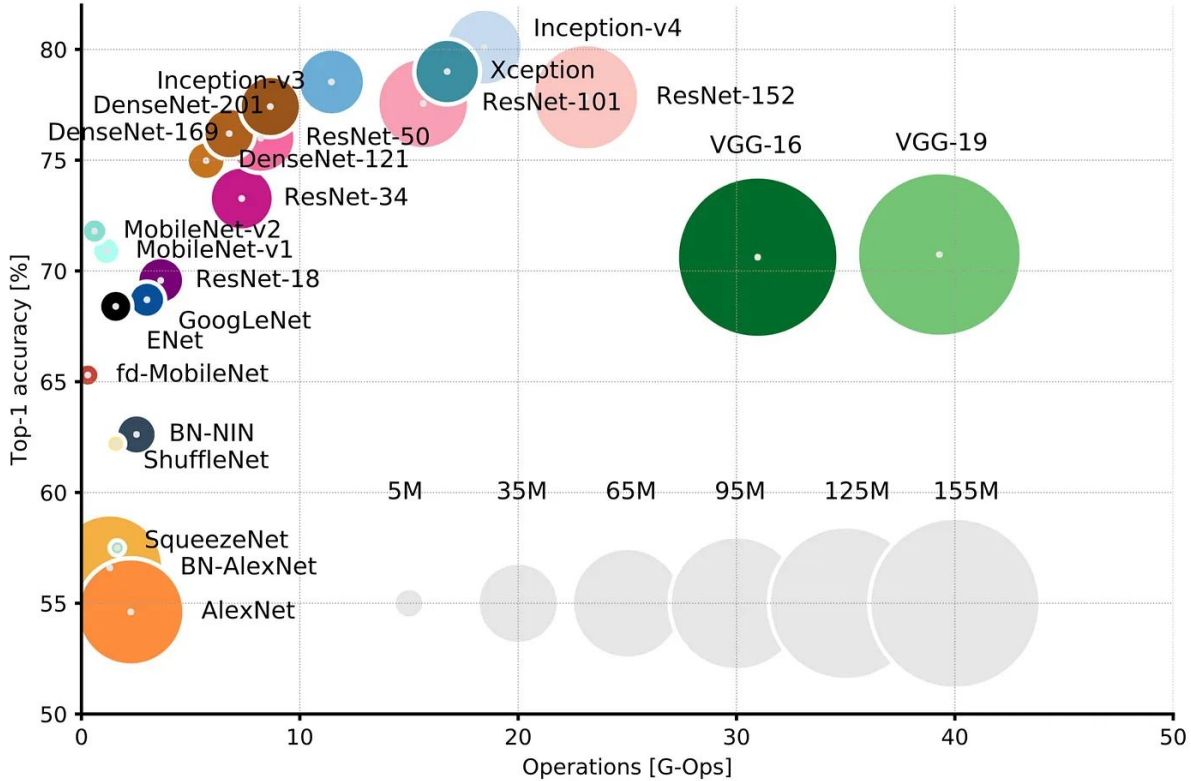
MobileNetV3 is able to achieve ResNet-50 acc, while 1/100s of its size.

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$



Famous Architectures



[Canziani et al 2016]

Famous Architectures

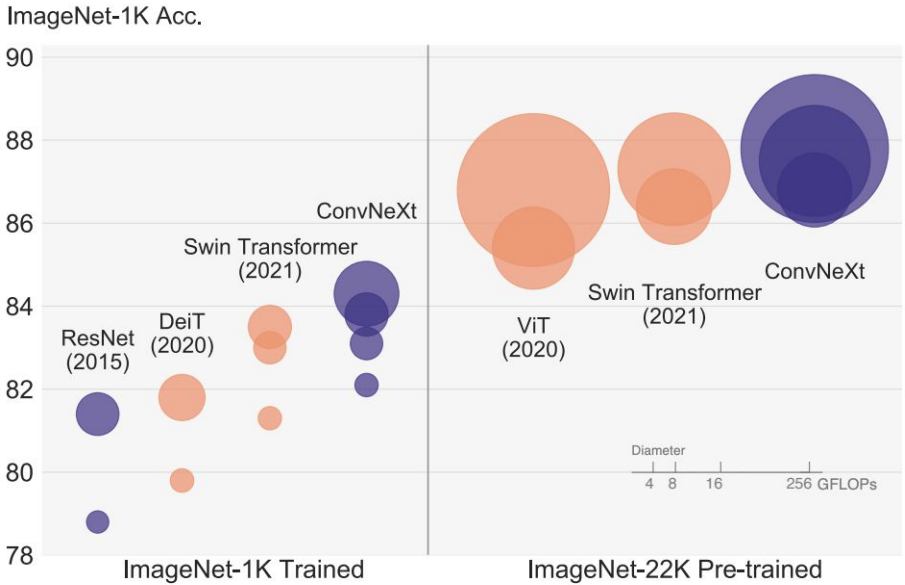
“A ConvNet for the 2020s”

[Liu et al 2022] ConvNeXt

CNNs were overshadowed by Visual Transformers.

The authors throw every trick in the CNN book at a ResNet - and create a better CNN than the leading ViT! Without creating a bigger model.

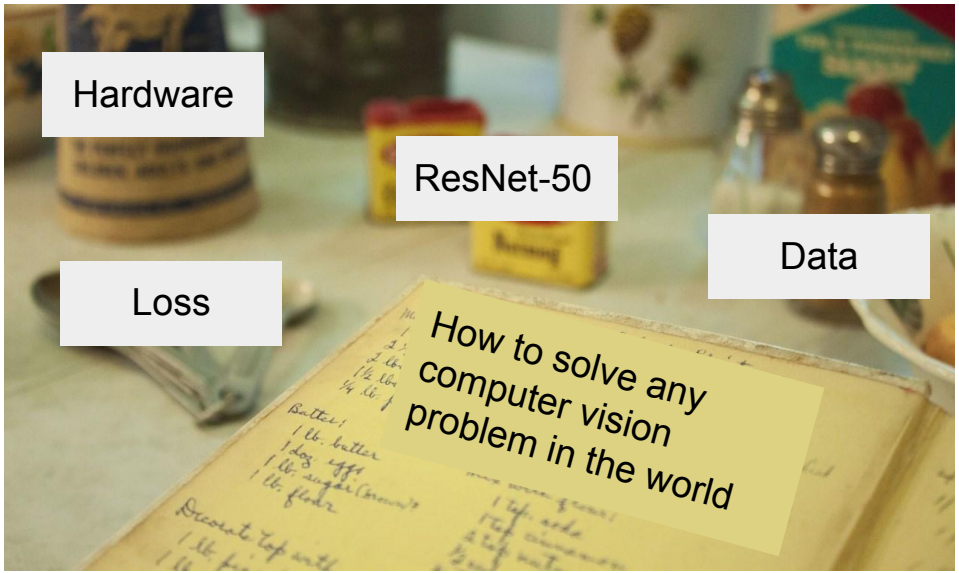
This may be “the last hurrah” for CNNs as they clear the stage.



Computer Vision: A Recipe

Given a visual detection problem at hand:

1. Don’t panic.
2. Pick a reasonable pre-trained backbone (ResNet, MobileNet)
3. Pick a valid “head” and the right loss. For classification: Flatten + Decision (e.g. softmax)
4. Create and track the right metrics.
5. Train on your data.
6. Iterate on architecture until you find a solution that satisfies the problem constraints (runtime, data, etc.)



Wrap Up

Vision Inspiration

Convolutional Layers

Network Architectures

Famous Works

Next:

Object Detection