ARTIFICIAL INTELLIGENCE 2

Convolutional Neural Networks

Francesco Spinnato

* francesco.spinnato@unipi.it
University of Pisa

Human vision

We are constantly analysing the world around us. Without conscious effort, when we see something:

- we label every object
- based on what we have learned in the past
- and decide what to do

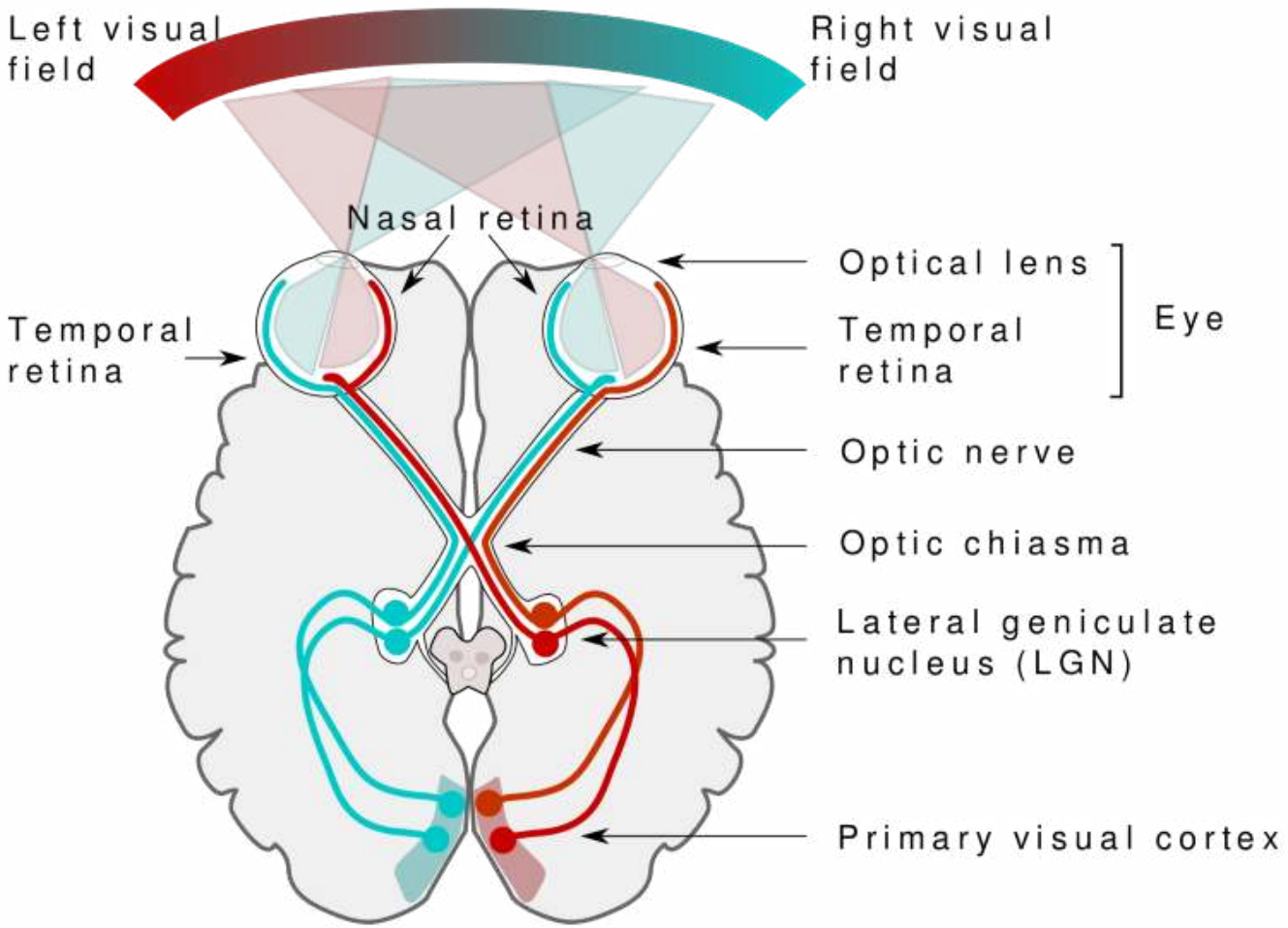
https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/

Human vision

We are constantly analysing the world around us. Without conscious effort, when we see something:

- we label every object
- based on what we have learned in the past
- and decide what to do

Human Vision



When you see an object:

- the light receptors in your eyes send signals via the optic nerve to the primary visual cortex, where the input is being processed.
- The primary visual cortex makes sense of what the eye sees.
- The deeply complex hierarchical structure of neurons and connections in the brain play a major role in this process of remembering and labelling objects.

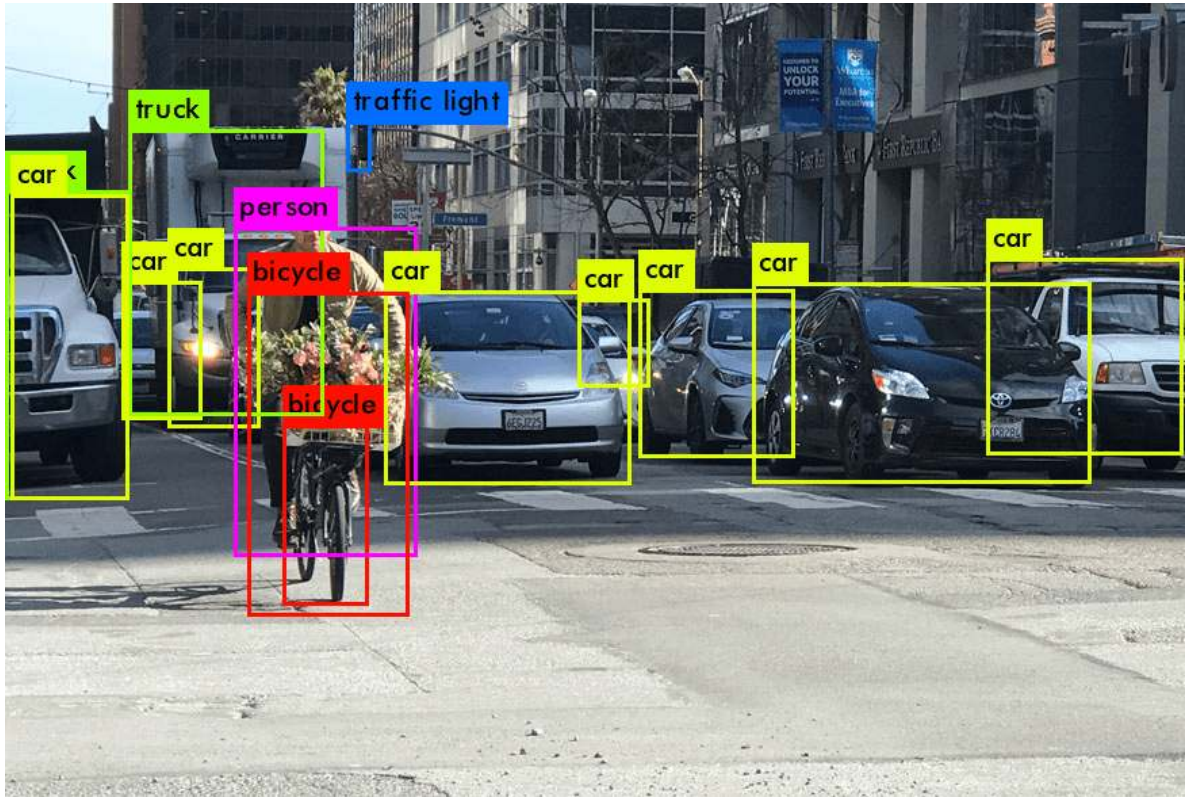
Convolutional Neural Networks

2D Convolution

Computer vision

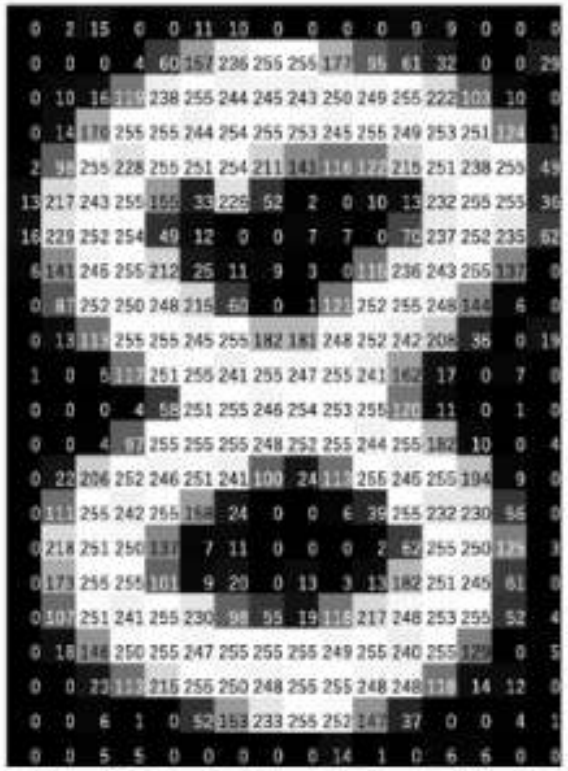
Computers ‘see’ in a different way than we do! Computer vision is the task of:

- **getting computers to handle and interpret visual data**
- for analysing visual images and videos
- **emulating human vision.**



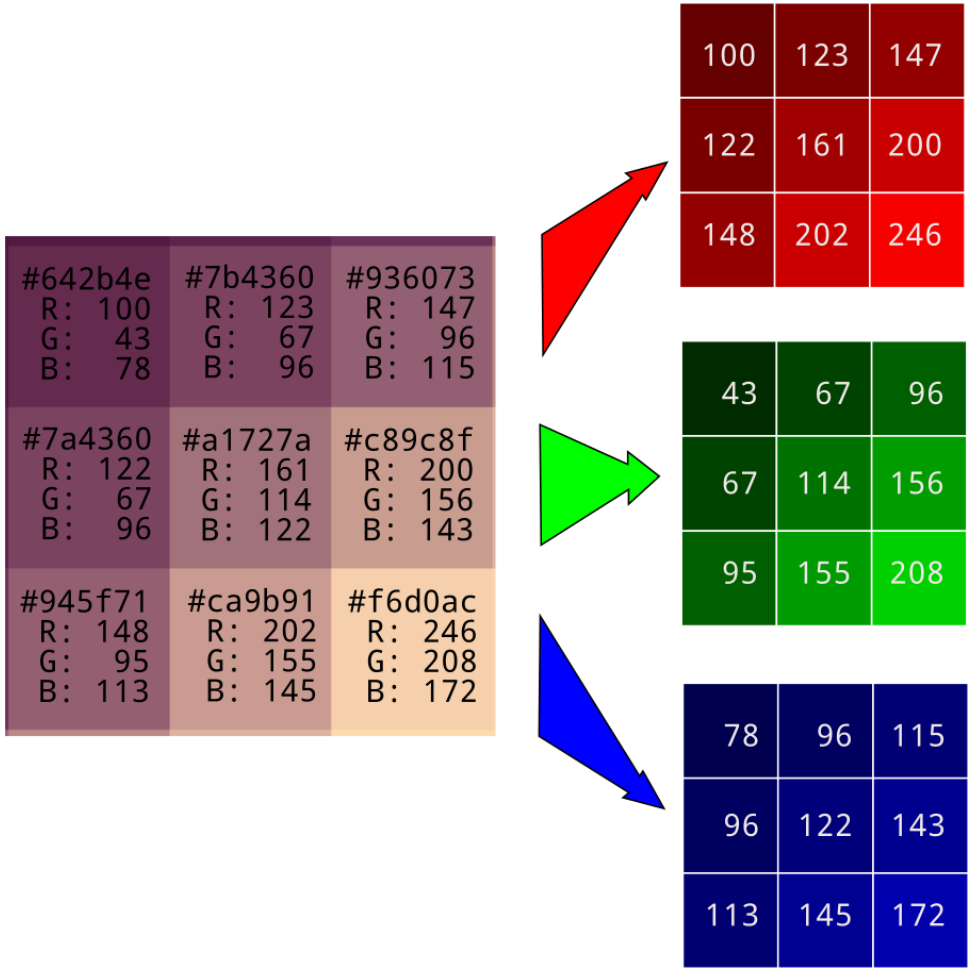
Images as matrices

Computers see images as matrices of pixel intensities.



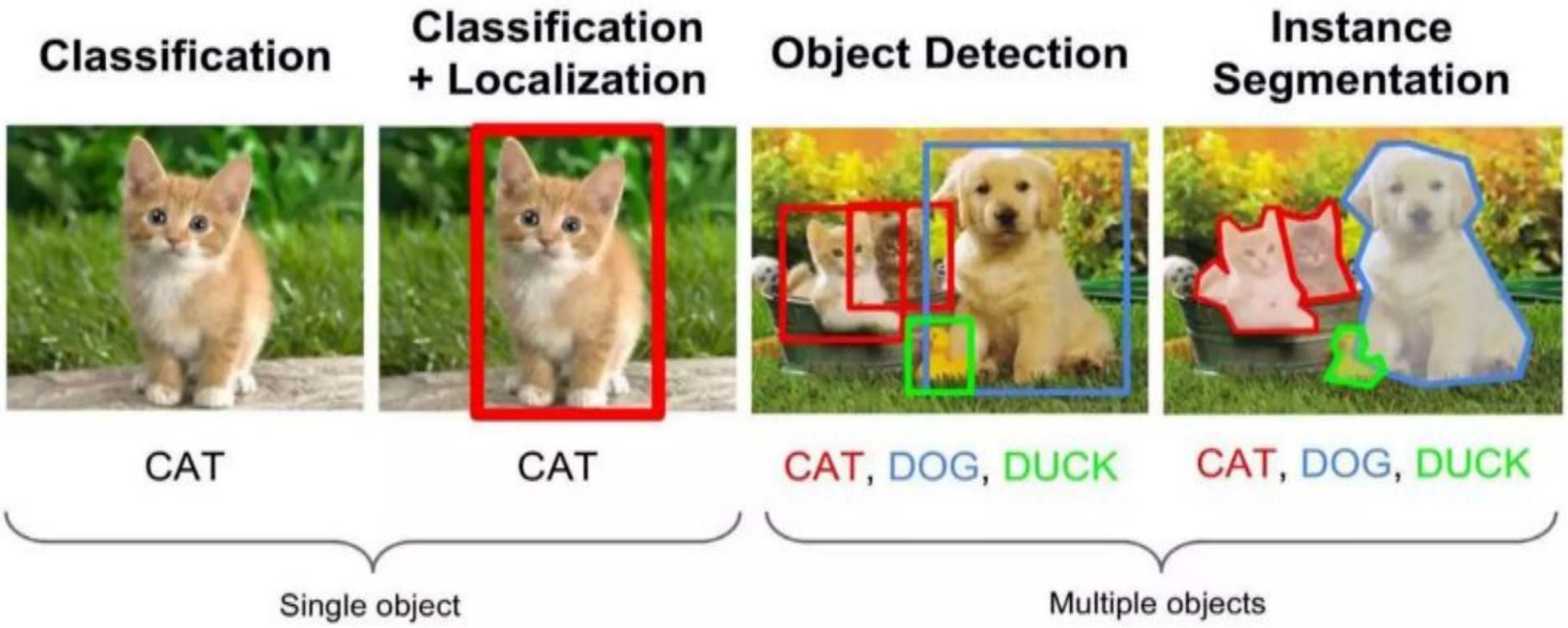
Images as matrices

Colors are also pixels, in particular a triplet of red, green, and blue intensities.



Tasks in Computer Vision

Computer vision is used to solve a variety of tasks.



Why Image Classification is Hard

- Different lighting conditions
- Viewpoint changes
- Translation, scale, rotation
- Contrast variation

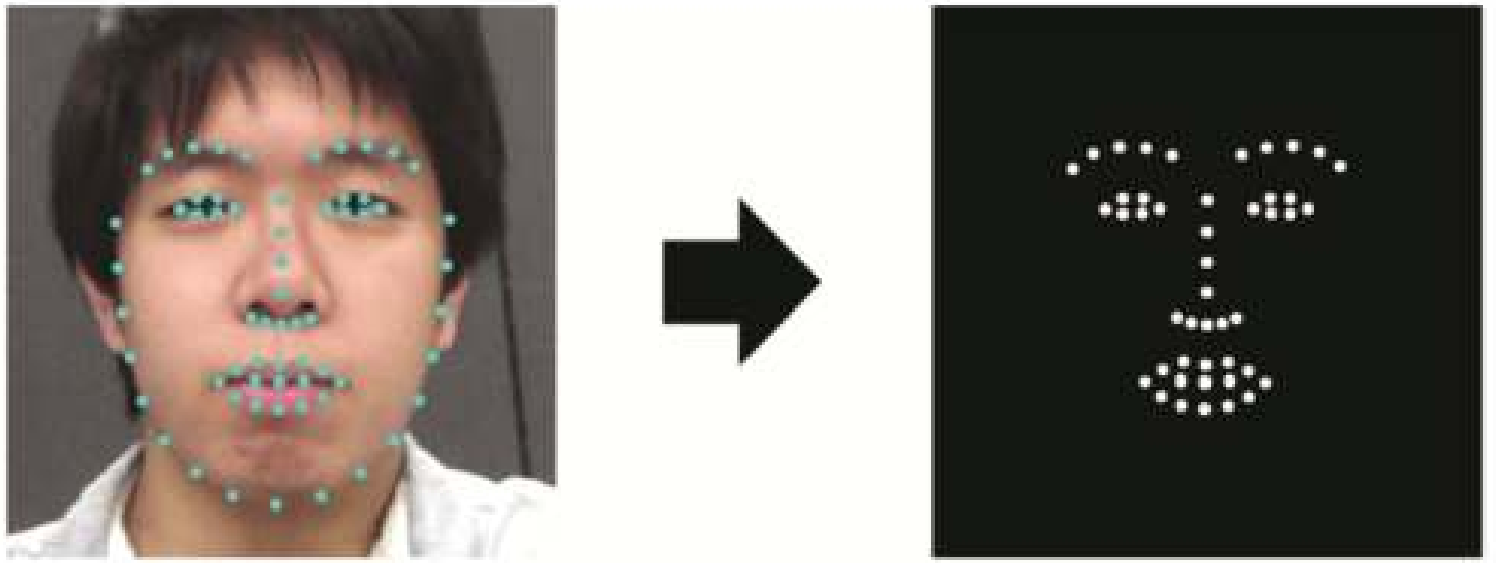
Image Source: twitter.com%2Fcats&psig=AOvVaw30_o-PCM-K21DiMAJQimQ4&ust=1553887775741551

Image Source: https://www.123rf.com/photo_76714328_side-view-of-tabby-cat-face-over-white.html

Traditional MLPs rely on **pixel-wise** sums, but objects are **local**

Traditional Approaches - Hand-Engineered Features

Handpick features based on expert knowledge



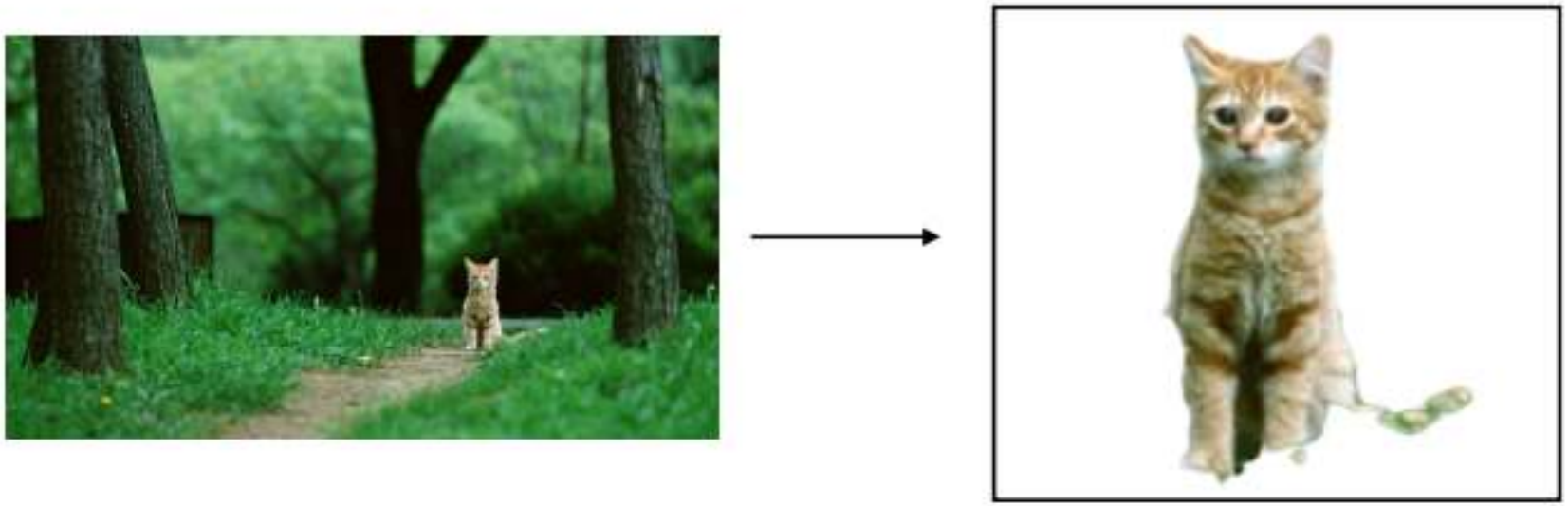
(a) Detected facial keypoints

(b) Facial organ keypoints

Sasaki, K., Hashimoto, M., & Nagata, N. (2016). Person Invariant Classification of Subtle Facial Expressions Using Coded Movement Direction of Keypoints. In *Video Analytics. Face and Facial Expression Recognition and Audience Measurement* (pp. 61-72). Springer, Cham.

Traditional Approaches - Preprocessing

Centering, cropping...



Convolutional Neural Networks

Convolution

Convolution in Math

- The basic operation under the hood in Convolutional Neural Networks (CNNs) is **convolution**.
- Classical convolution is a mathematical operation on two functions, $f * g$, as the integral of the product of the two functions after one is reflected about the y-axis and shifted.
- Machine learning uses the so-called **discrete convolution**, where convolutions are performed on the input data with the use of a **filter** or **kernel** to then produce a feature map.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

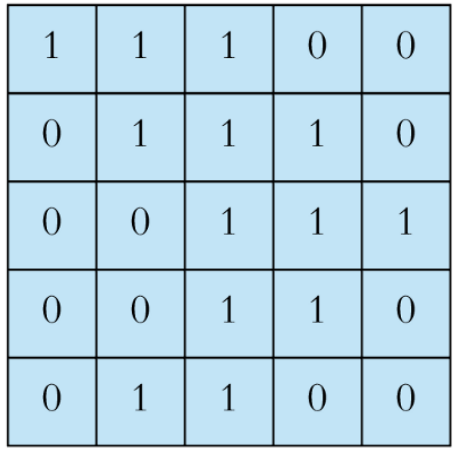
2D Convolution

We do 2D convolution by sliding the filter over the input.

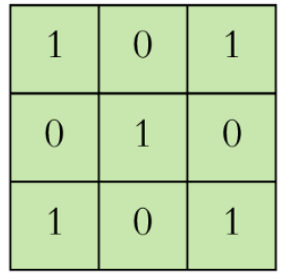
At every location, we do:

- element-wise matrix multiplication
- and then sum the result to get a single value.
- this value goes into the **feature map**.

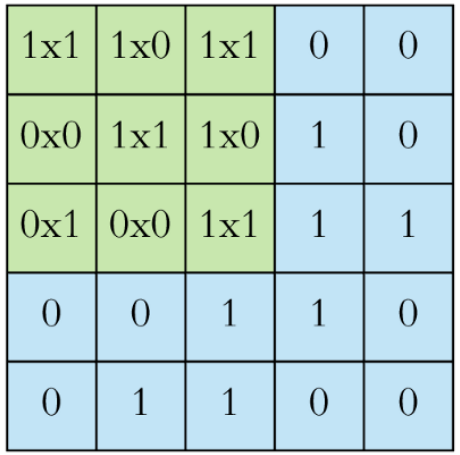
The green area where the convolution takes place is called the **receptive field**.



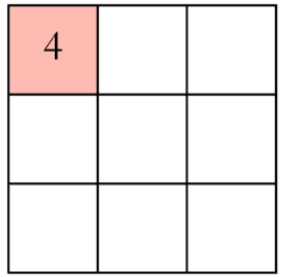
Input



Filter / Kernel



Input x Filter



Feature Map

2D Convolution

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Kernels

The kernels are the weights, i.e., what you learn in a CNN. Good kernels can process the image in such a way to highlight certain characteristics.

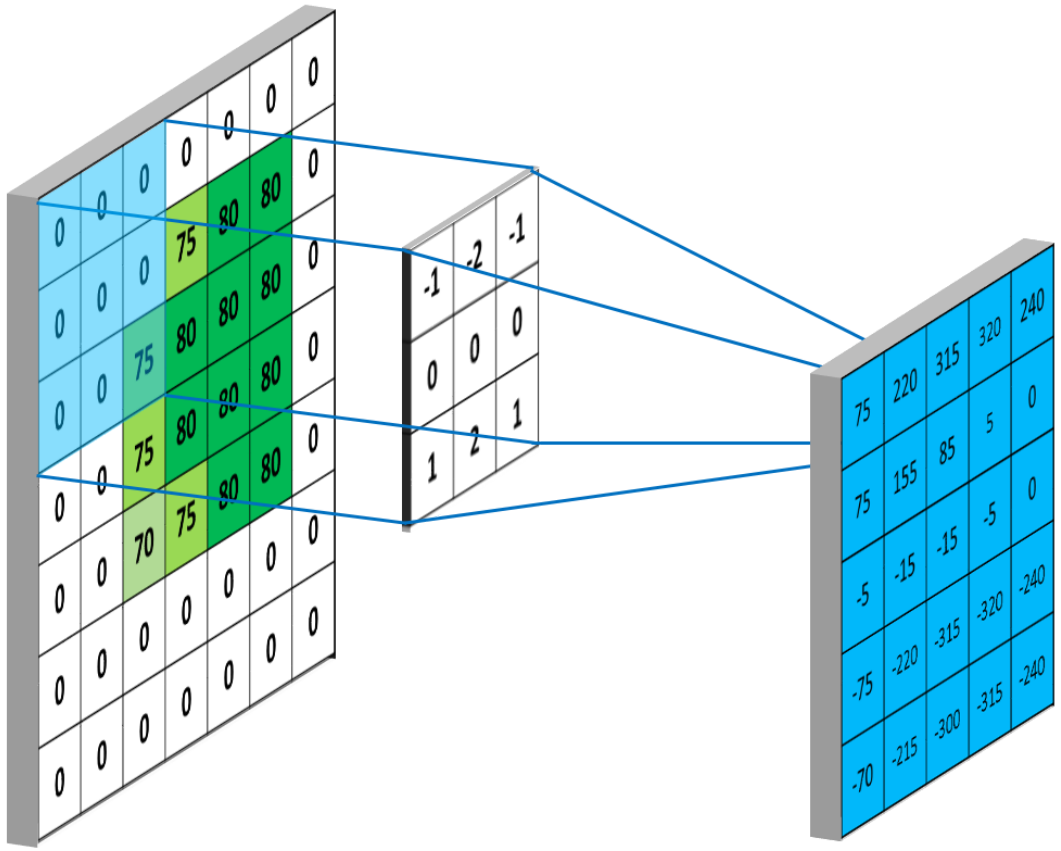
Original

Sharpen

Edge detect

Strong edge detect

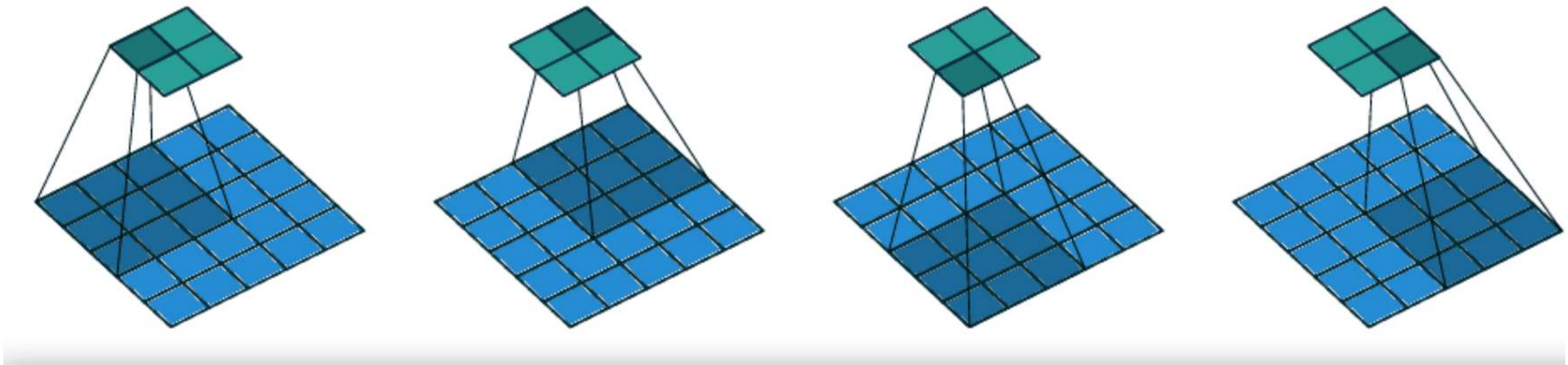
Padding



If you want the feature map to have the same shape as the input, **padding** is often used, i.e., adding a "border" of zeros to the image.

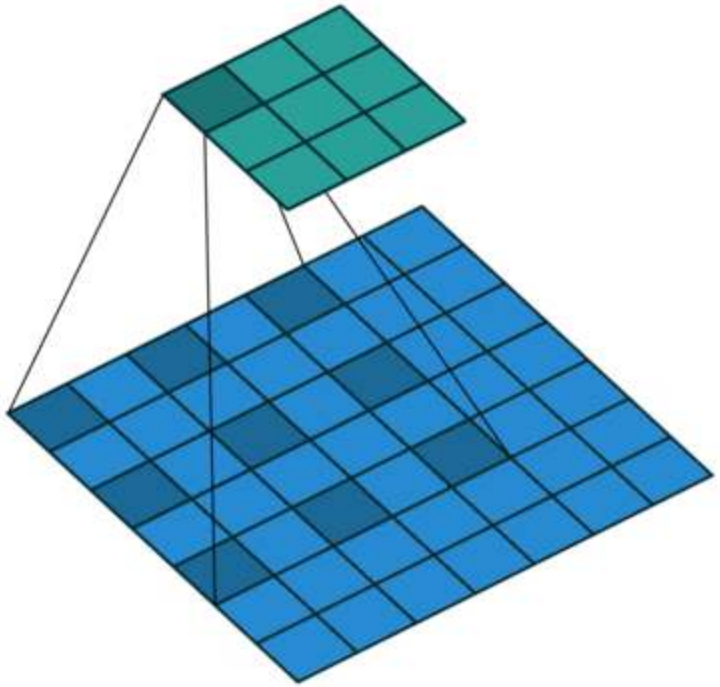
Stride

Strided kernels move more than one point at a time (e.g., stride=2 below)



Dilation

Dilated kernels cover non contiguous receptive fields (e.g., dilation=2 below)



Output Size

For a 2D conv with a square kernel, the output spatial dimensions are:

$$O = \left\lfloor \frac{I - K + 2P}{S} \right\rfloor + 1$$

Where:

- **I** = input size (height or width)
- **K** = kernel size
- **P** = padding
- **S** = stride

Without padding or stride, a $K \times K$ kernel on an $I \times I$ input gives $(I - K + 1)$ outputs per dimension

Exercise

Given:

Input matrix $\mathbf{X}$ (4×4):

$$\mathbf{X} = \begin{pmatrix} 1 & 2 & 3 & 0 \\ 4 & 5 & 6 & 1 \\ 7 & 8 & 9 & 2 \\ 0 & 3 & 4 & 5 \end{pmatrix}$$

Kernel $\mathbf{W}$ (3×3):

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

Settings: no padding, no dilation, stride = 1

Output size:

$$\left\lfloor \frac{4-3}{1} \right\rfloor + 1 = 2 \quad \Rightarrow \quad \mathbf{H} \in \mathbb{R}^{2\times 2}$$

Task:

Compute $h_{0,0}$, $h_{0,1}$, $h_{1,0}$, $h_{1,1}$

Position (0,0) — receptive field: rows 0-2, cols 0-2

$$\mathbf{X}_{0,0} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

Element-wise products:

$$\begin{pmatrix} 1{\cdot}1 & 2{\cdot}0 & 3{\cdot}(-1) \\ 4{\cdot}1 & 5{\cdot}0 & 6{\cdot}(-1) \\ 7{\cdot}1 & 8{\cdot}0 & 9{\cdot}(-1) \end{pmatrix} = \begin{pmatrix} 1 & 0 & -3 \\ 4 & 0 & -6 \\ 7 & 0 & -9 \end{pmatrix}$$

$$h_{0,0} = 1 + 0 - 3 + 4 + 0 - 6 + 7 + 0 - 9 = \mathbf{-6}$$

Position (0,1) — receptive field: rows 0-2, cols 1-3

$$\mathbf{X}_{0,1} = \begin{pmatrix} 2 & 3 & 0 \\ 5 & 6 & 1 \\ 8 & 9 & 2 \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

Element-wise products:

$$\begin{pmatrix} 2{\cdot}1 & 3{\cdot}0 & 0{\cdot}(-1) \\ 5{\cdot}1 & 6{\cdot}0 & 1{\cdot}(-1) \\ 8{\cdot}1 & 9{\cdot}0 & 2{\cdot}(-1) \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 5 & 0 & -1 \\ 8 & 0 & -2 \end{pmatrix}$$

$$h_{0,1} = 2 + 0 + 0 + 5 + 0 - 1 + 8 + 0 - 2 = \mathbf{12}$$

Position (1,0) — receptive field: rows 1-3, cols 0-2

$$\mathbf{X}_{1,0} = \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 0 & 3 & 4 \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

Element-wise products:

$$\begin{pmatrix} 4{\cdot}1 & 5{\cdot}0 & 6{\cdot}(-1) \\ 7{\cdot}1 & 8{\cdot}0 & 9{\cdot}(-1) \\ 0{\cdot}1 & 3{\cdot}0 & 4{\cdot}(-1) \end{pmatrix} = \begin{pmatrix} 4 & 0 & -6 \\ 7 & 0 & -9 \\ 0 & 0 & -4 \end{pmatrix}$$

$$h_{1,0} = 4 + 0 - 6 + 7 + 0 - 9 + 0 + 0 - 4 = \mathbf{-8}$$

Position (1,1) — receptive field: rows 1-3, cols 1-3

$$\mathbf{X}_{1,1} = \begin{pmatrix} 5 & 6 & 1 \\ 8 & 9 & 2 \\ 3 & 4 & 5 \end{pmatrix}$$

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

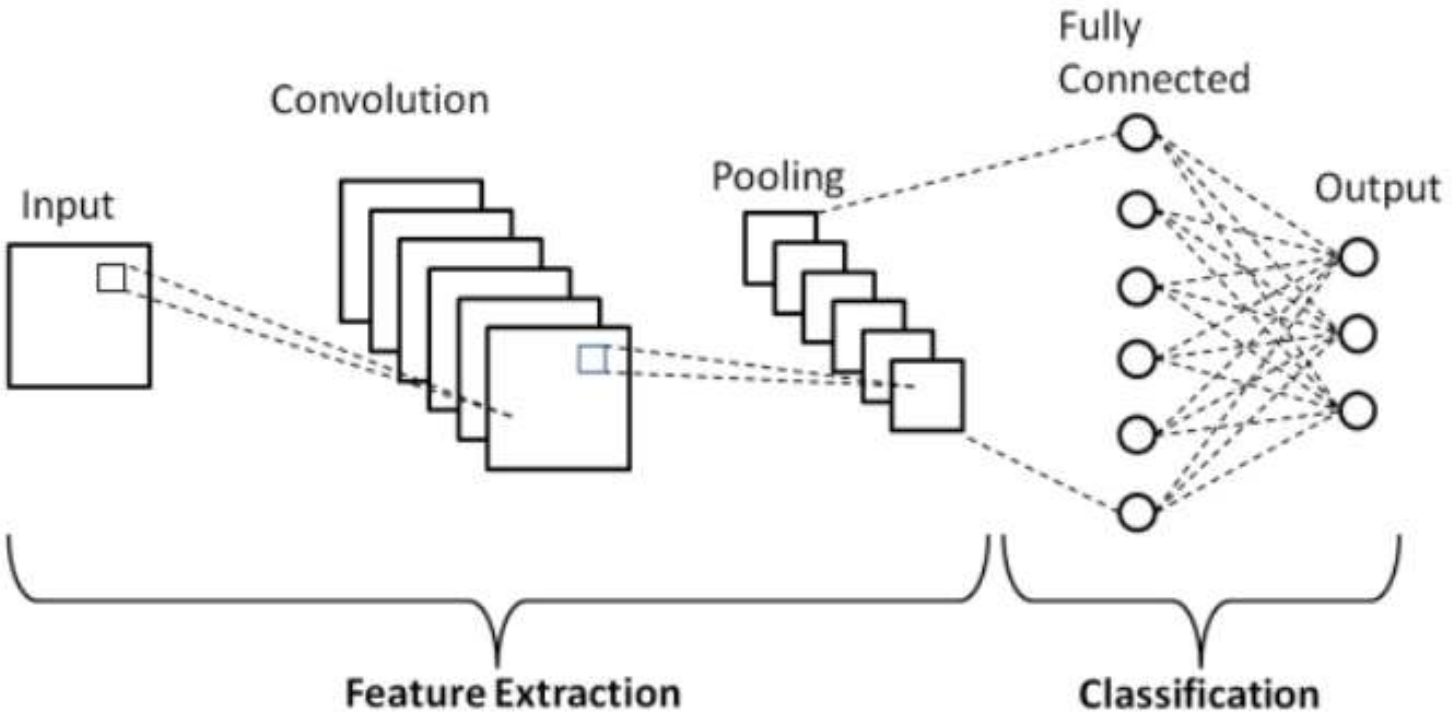
Element-wise products:

$$\begin{pmatrix} 5{\cdot}1 & 6{\cdot}0 & 1{\cdot}(-1) \\ 8{\cdot}1 & 9{\cdot}0 & 2{\cdot}(-1) \\ 3{\cdot}1 & 4{\cdot}0 & 5{\cdot}(-1) \end{pmatrix} = \begin{pmatrix} 5 & 0 & -1 \\ 8 & 0 & -2 \\ 3 & 0 & -5 \end{pmatrix}$$

$$h_{1,1} = 5 + 0 - 1 + 8 + 0 - 2 + 3 + 0 - 5 = \mathbf{8}$$

Convolutional Neural Networks

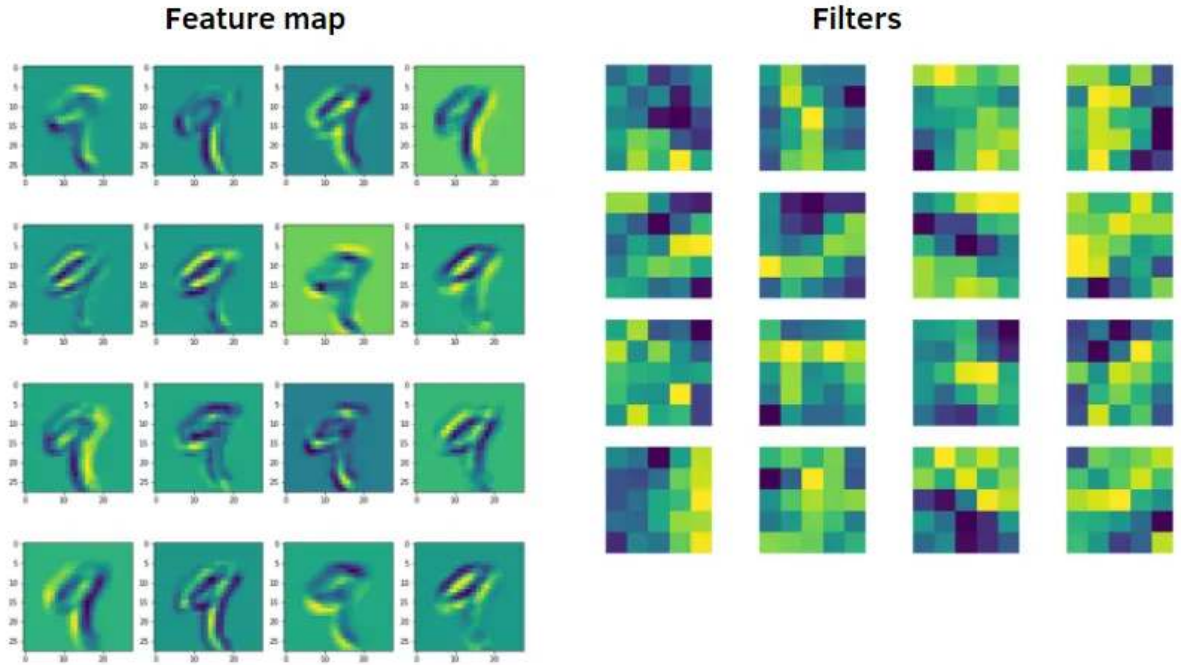
Convolutional Neural Networks (CNNs) automate learning good filters for a task!



LeCun et al. (1989, 1998): Backpropagation applied to handwritten digit recognition.

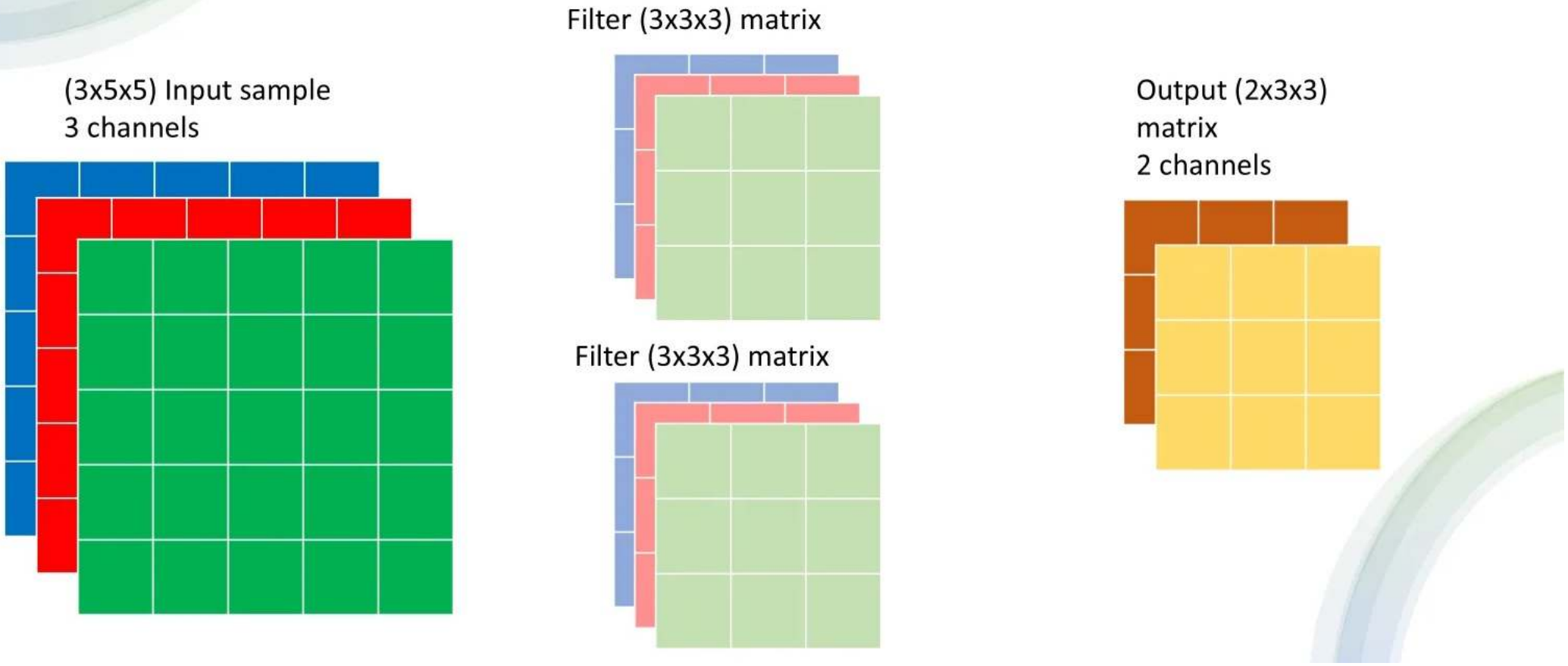
Feature Extraction: Multiple Kernels

Given that you want to extract multiple features from you image, you usually need more kernels.



Feature Extraction: Multiple Kernels

With many input channels, for each output channel, you have a 3D filter bank



Feature Extraction: Pooling

One limitation of the feature map output of convolutional layers is that they capture the exact position of features in the input.

This means that small movements in the position of the feature in the input image will result in a different feature map. This can happen with

- re-cropping,
- rotation,
- shifting,
- and other minor changes to the input image.

To solve this problem we use **pooling**.

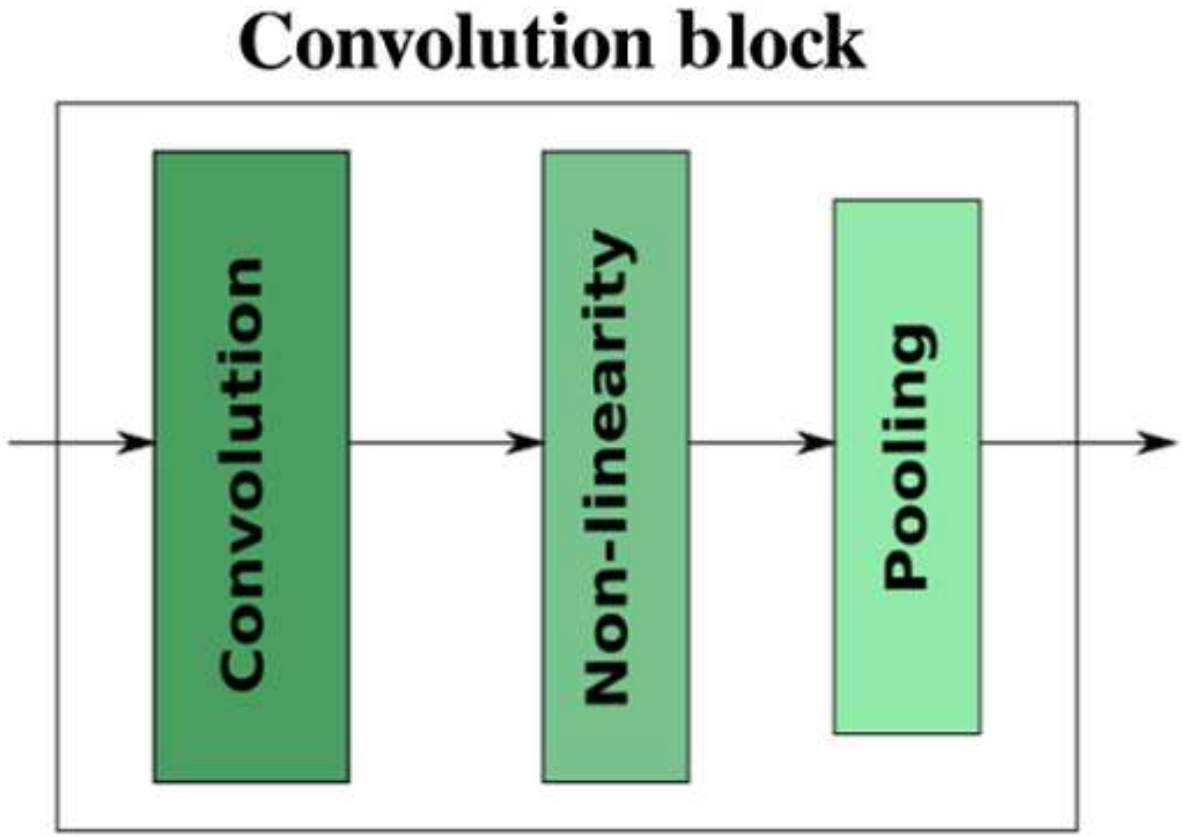
Pooling Layer

A pooling layer is a new layer added after the convolutional layer that segments the image and applies an aggregation.

This happens after a nonlinearity (e.g. ReLU) has been applied to the feature maps output by a convolutional layer.

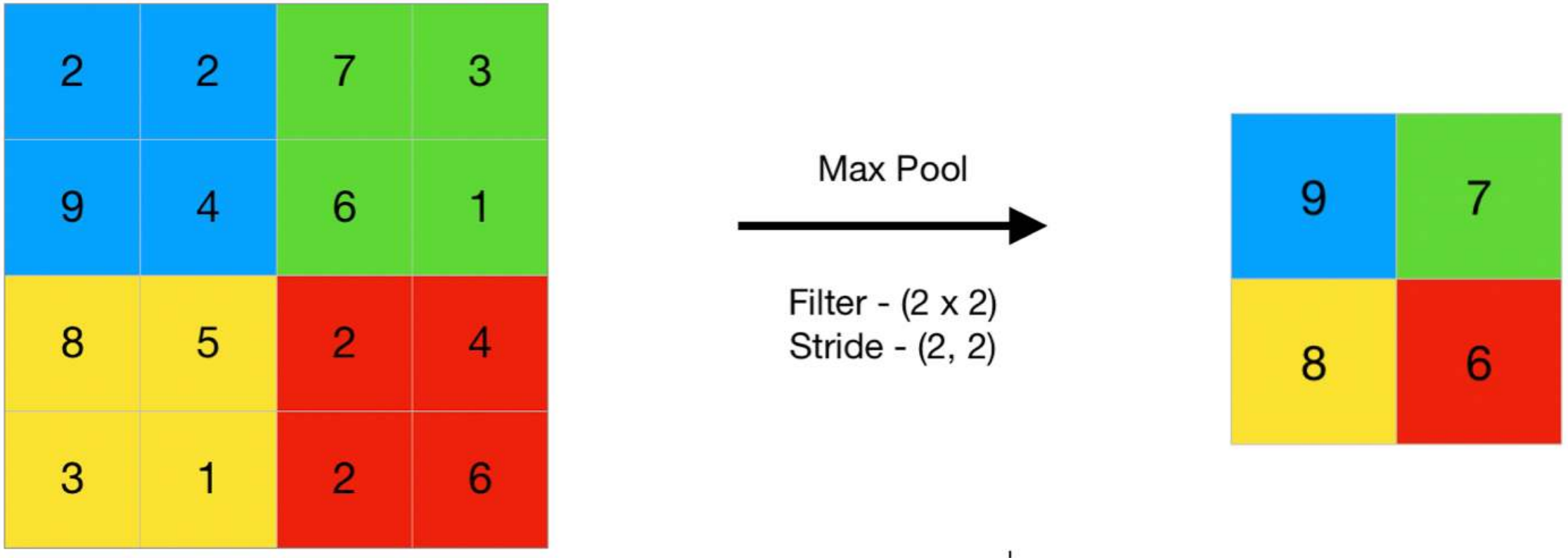
The usual convolutional block is composed by:

- Convolutional layer
- Nonlinearity
- Pooling Layer



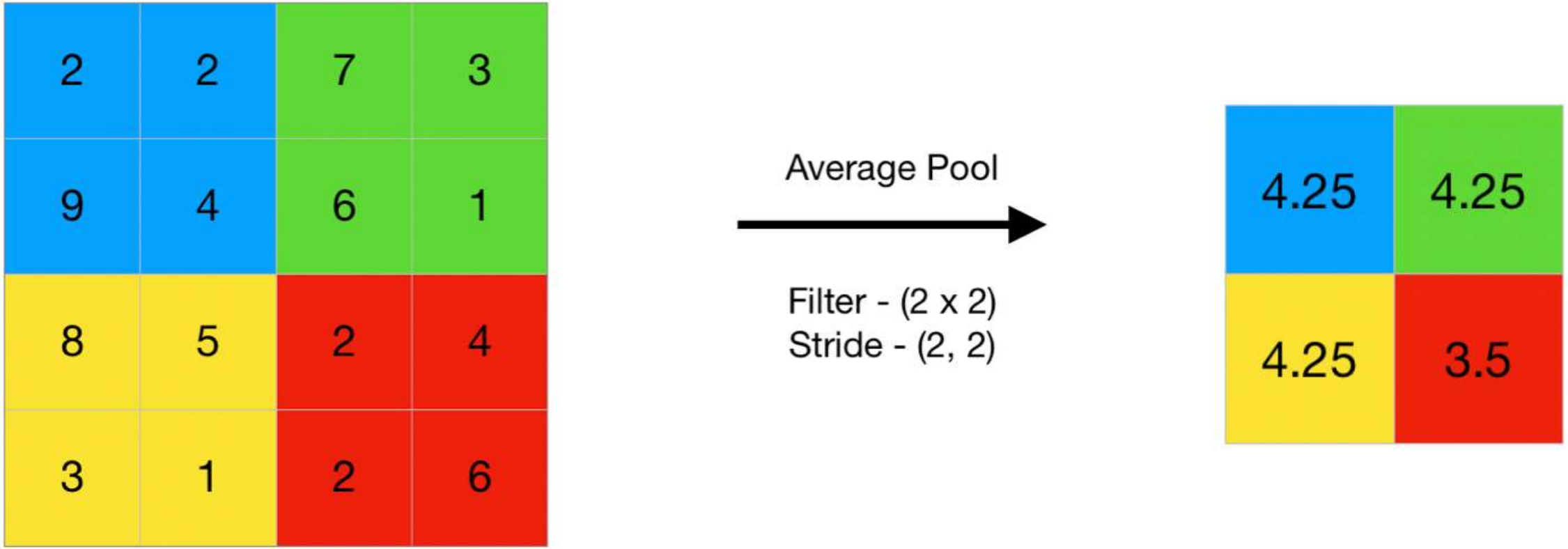
Max Pooling

Max pooling selects the maximum element from the region of the feature map covered by the filter.



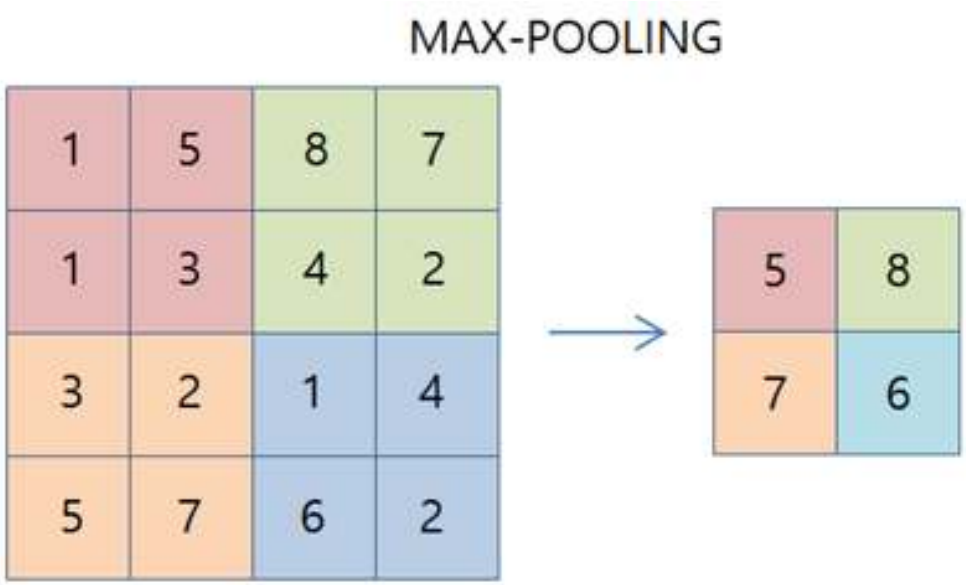
Average Pooling

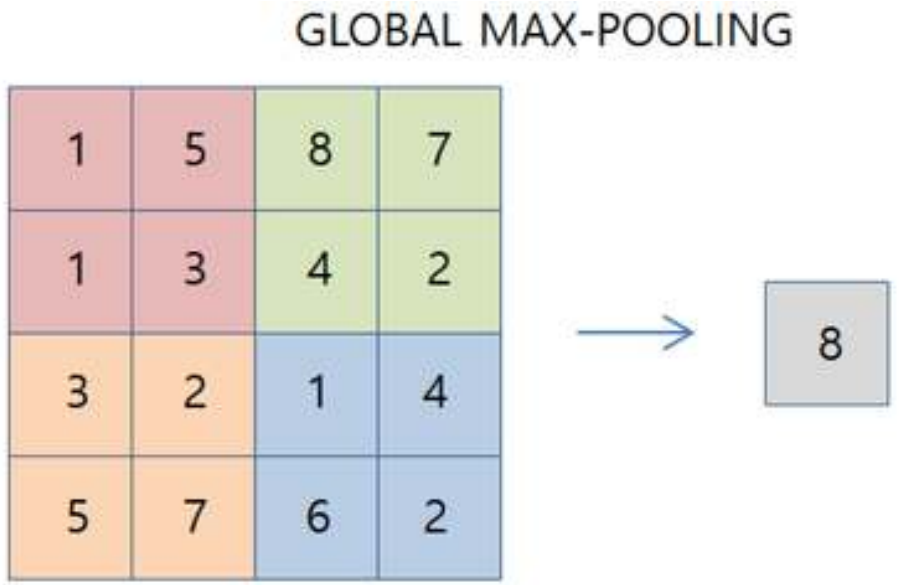
Average pooling computes the average of the elements present in the region of feature map covered by the filter.



Global Pooling

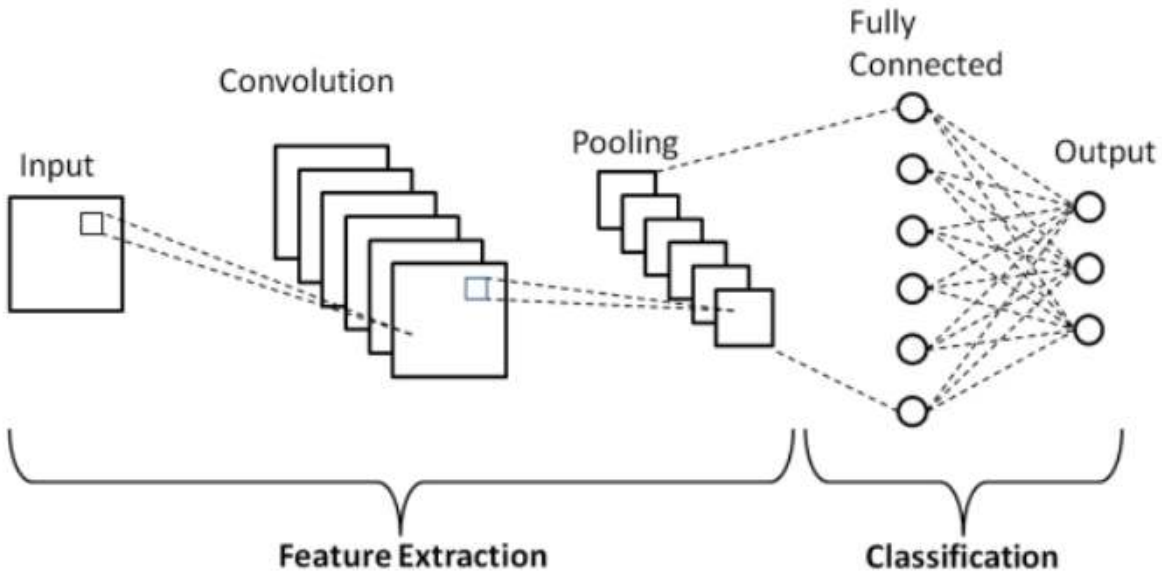
Instead of segmenting the image, take the avg or max of the whole feature map.





Dense Head

To perform a classification class, the output is flattened and passed to an fully connected (MLP-like) network to predict the labels.



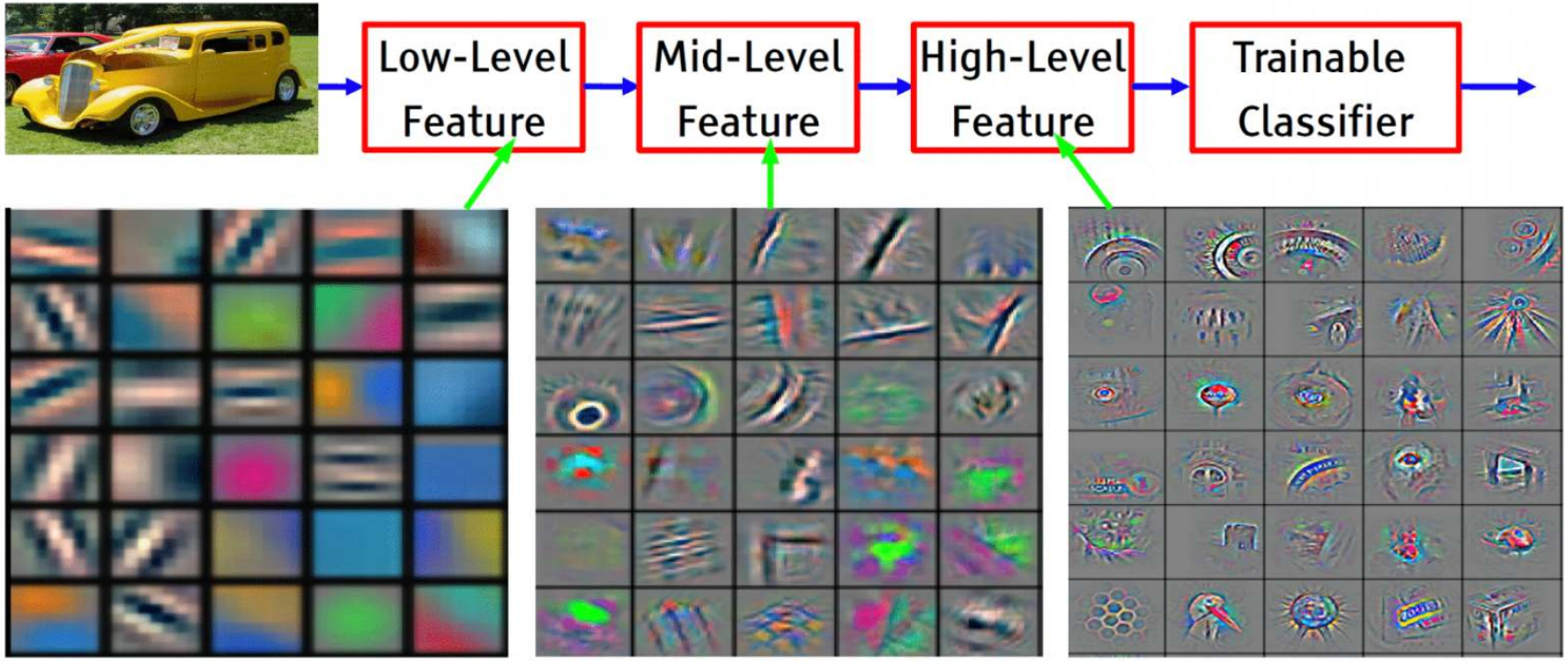
CNNs are trained very similarly to MLPs: same multivariable chain rule with additional constraint, i.e., weight sharing.

Strengths

The main strengths of CNNs are:

- **Sparse connectivity**
  - Each neuron is connected to only a small region of the input
- **Parameter sharing**
  - A filter slides over the input to create a feature map.
  - Same weights reused across spatial locations
- **Hierarchical feature learning**
  - Early layers: simple features like edges, corners, color gradients
  - Middle layers: more complex shapes, textures and patterns
  - Deep layers: abstract concepts, e.g, eyes, nose, fur

Strengths


Low-Level Feature
Mid-Level Feature
High-Level Feature
Trainable Classifier

Weaknesses

CNNs are **not strictly invariant** to:

- Translation
- Rotation
- Scale

Pooling helps with *local* invariance.

There can be vanishing gradient problems (but less than in RNNs)

LeNet

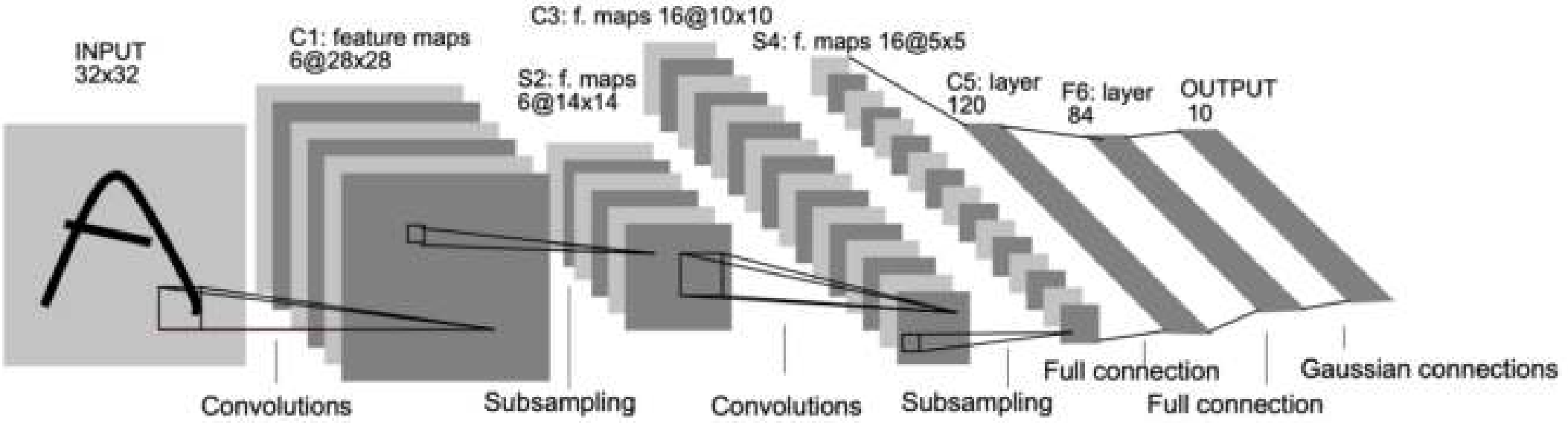


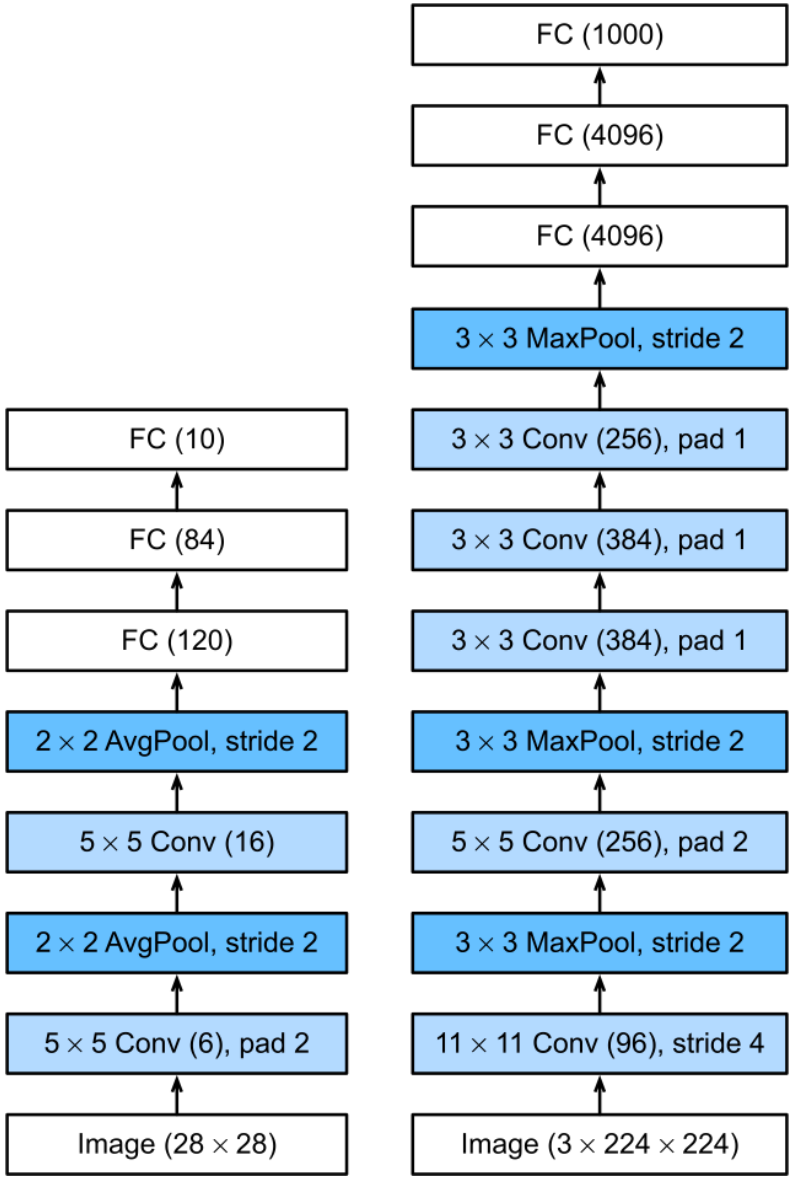
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeCun, Yann, et al. "Gradient-based learning applied to document recognition." (2002)

Lenet vs Alexnet

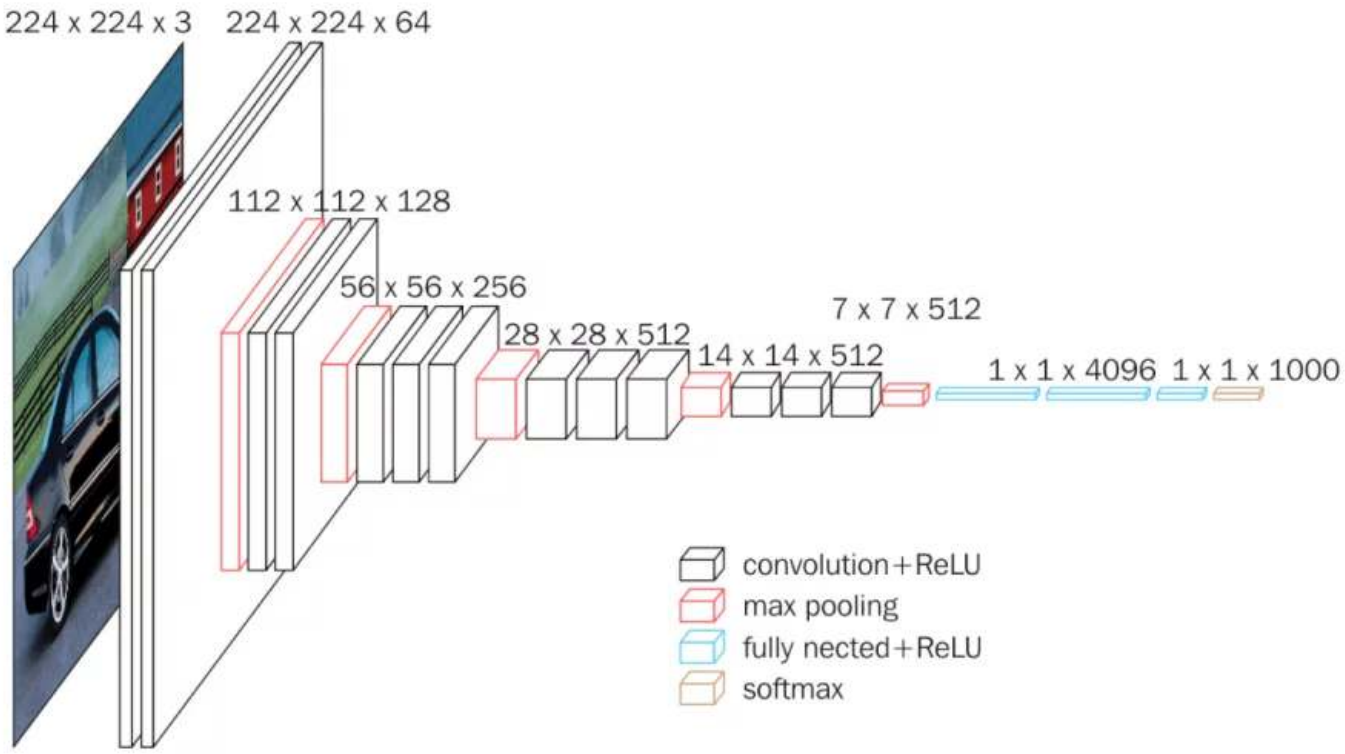
AlexNet (right) has a more complex architecture than LeNet (left).

- it can classify images into 1,000 distinct object categories
- is regarded as the first widely recognized application of deep convolutional networks in large-scale visual recognition.



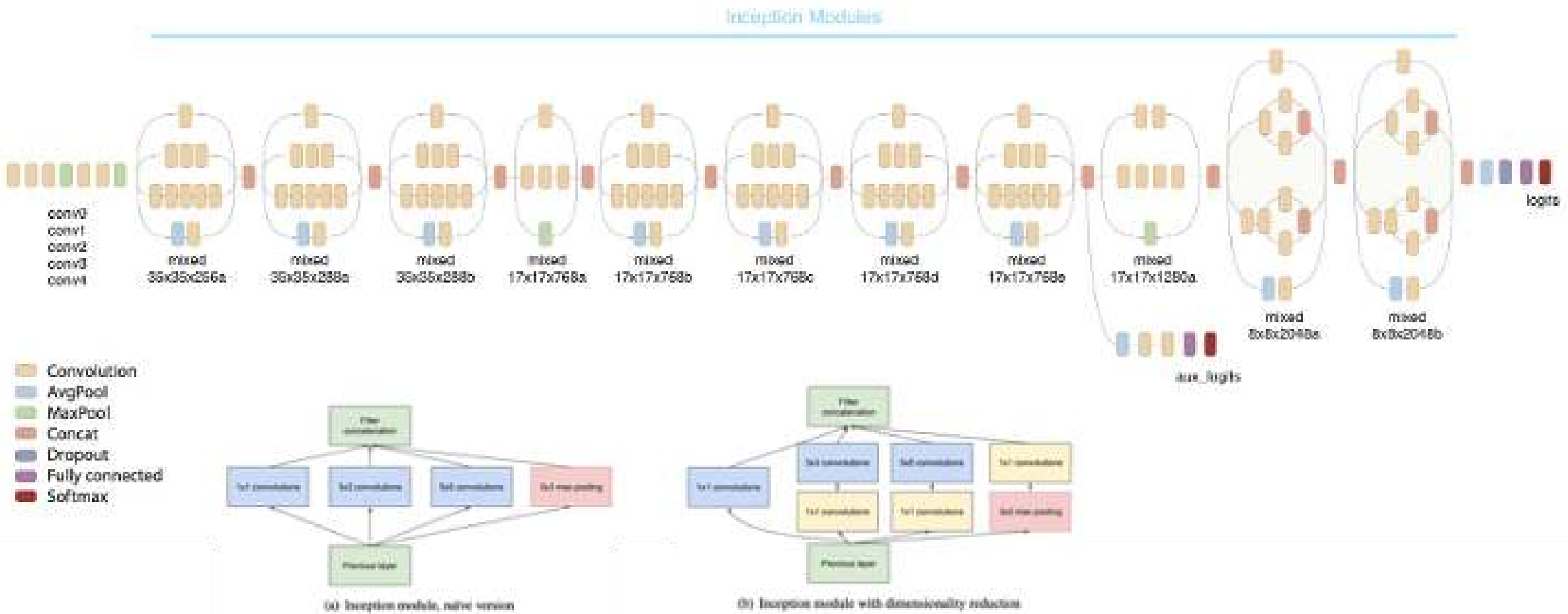
VGG16

Deeper network than AlexNet, smaller filter size



Inception

The inception module processes the input in parallel with multiple convolutions

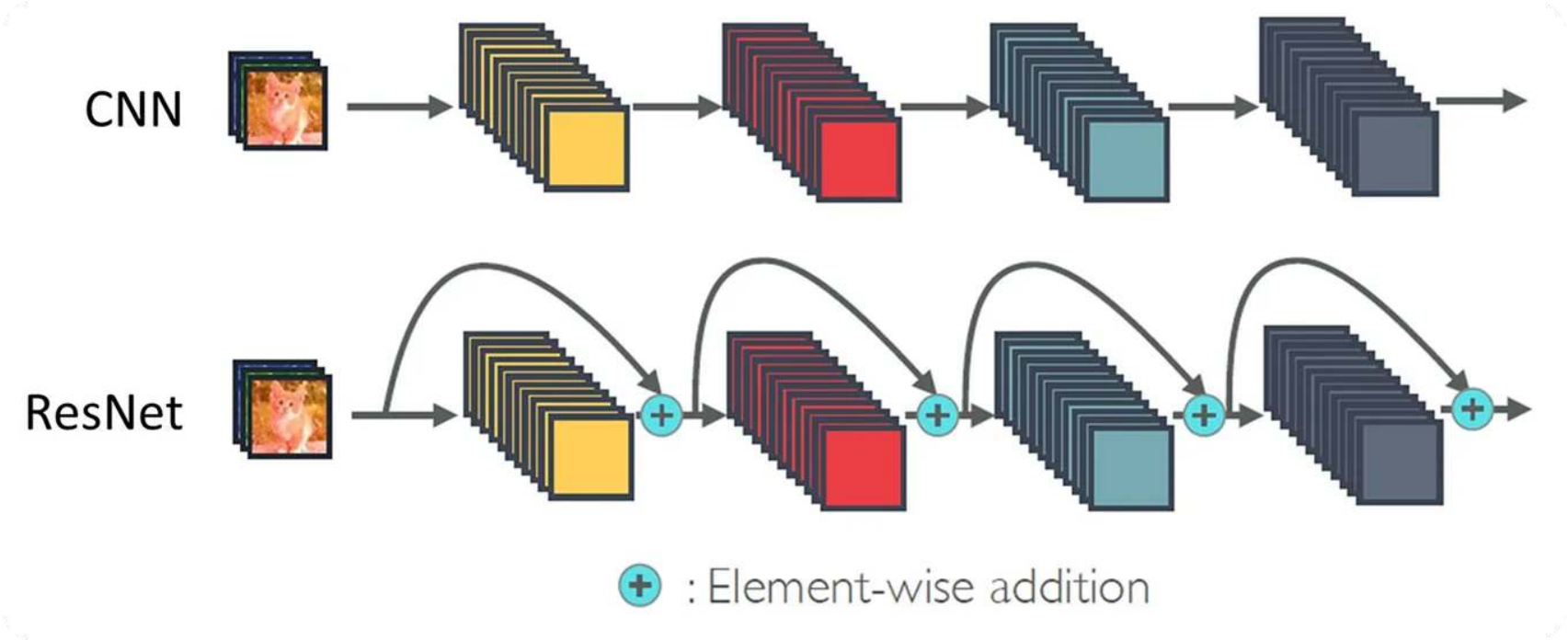


Christian Szegedy, et al. "Going deeper with convolutions.", CVPR 2015

Resnet

Uses shortcut connections to:

- alleviate vanishing gradient problem
- combine shallow and deep



Kaiming He et al. "Deep residual learning for image recognition", CVPR 2016

Evolution of Architectures

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

Convolutional Neural Networks

1D Convolution

1D Convolution

1D convolution is used when the input is **one-dimensional**, such as:

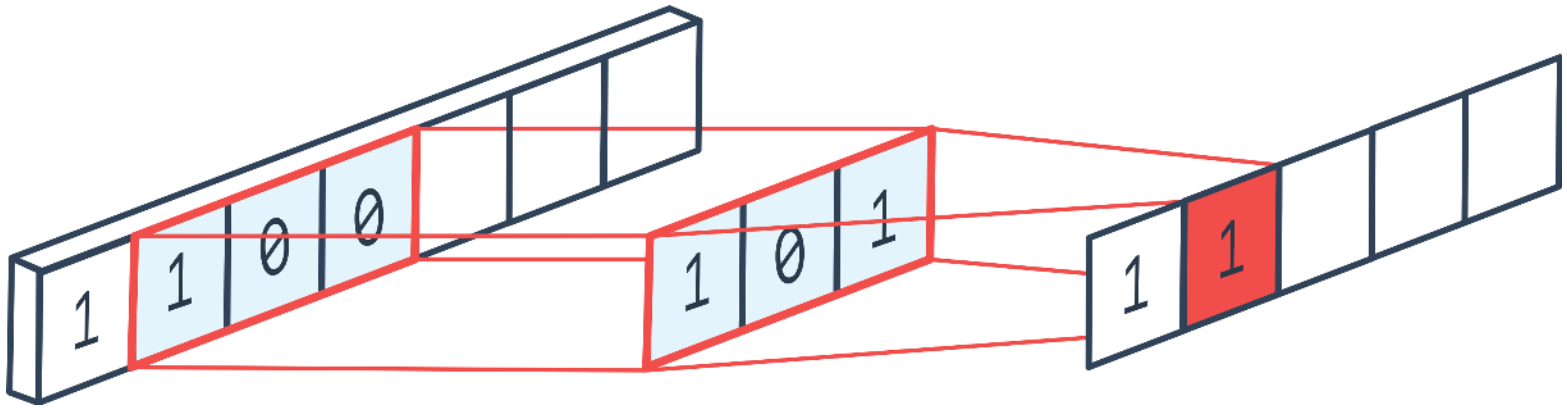
- time series
- audio signals
- text sequences
- sensor measurements

The input is a vector instead of a matrix.

1D Convolution

In 1D convolution:

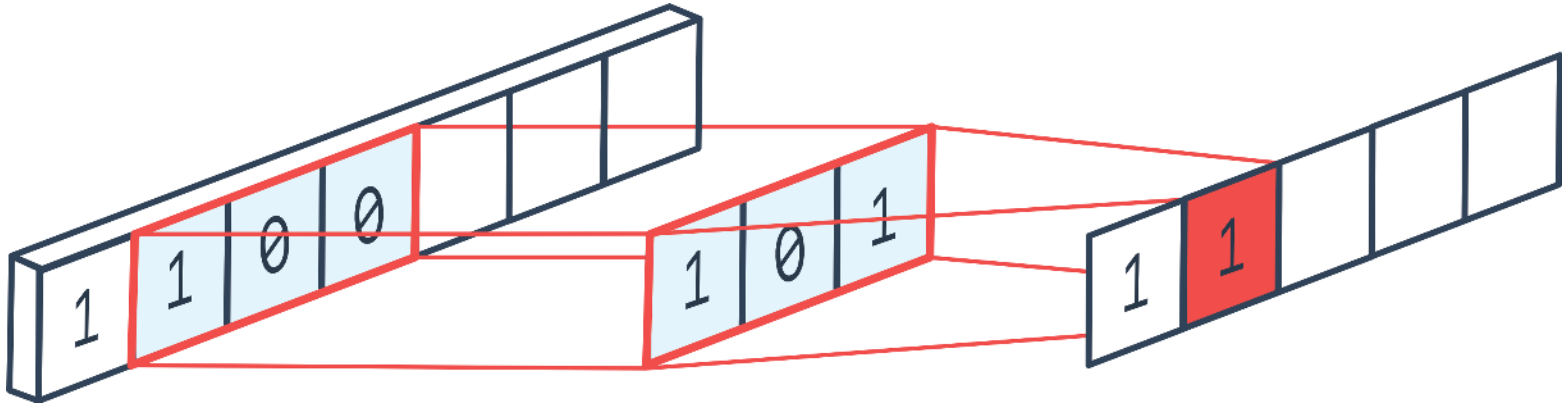
- the kernel is a **1D filter**
- it slides along a single spatial (or temporal) dimension
- at each position, we compute a dot product between the kernel and the input segment
- the result forms a **1D feature map**



1D Convolution

The **receptive field** in 1D convolution is the segment of the input covered by the kernel.

- kernel size $k \rightarrow$ receptive field of length k
- deeper layers increase the effective receptive field



1D Convolution

Given an input signal $\mathbf{x} \in \mathbb{R}^n$ and a kernel $\mathbf{w} \in \mathbb{R}^k$

At position i, define the receptive field as:

$$\mathbf{x}_i = \begin{bmatrix} x_i & x_{i+1} & \cdots & x_{i+k-1} \end{bmatrix}$$

The 1D convolution output at position i is the dot product:

$$h_i = \mathbf{w} \cdot \mathbf{x}_i$$

This operation is repeated by sliding the kernel along the input signal.

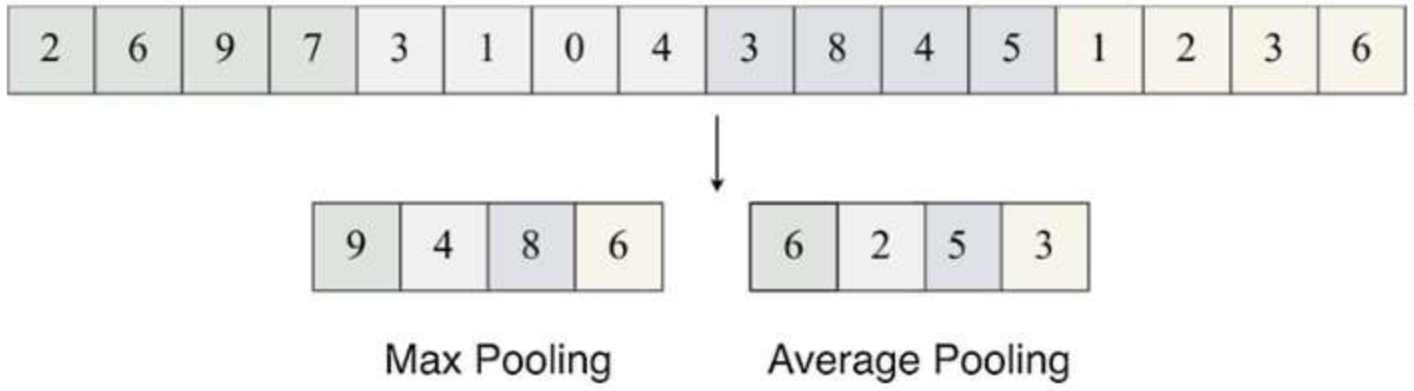
Stride, Padding, Pooling

As in 2D convolution:

- **stride** controls how much the kernel shifts at each step
- **padding** adds zeros at the beginning and end of the signal

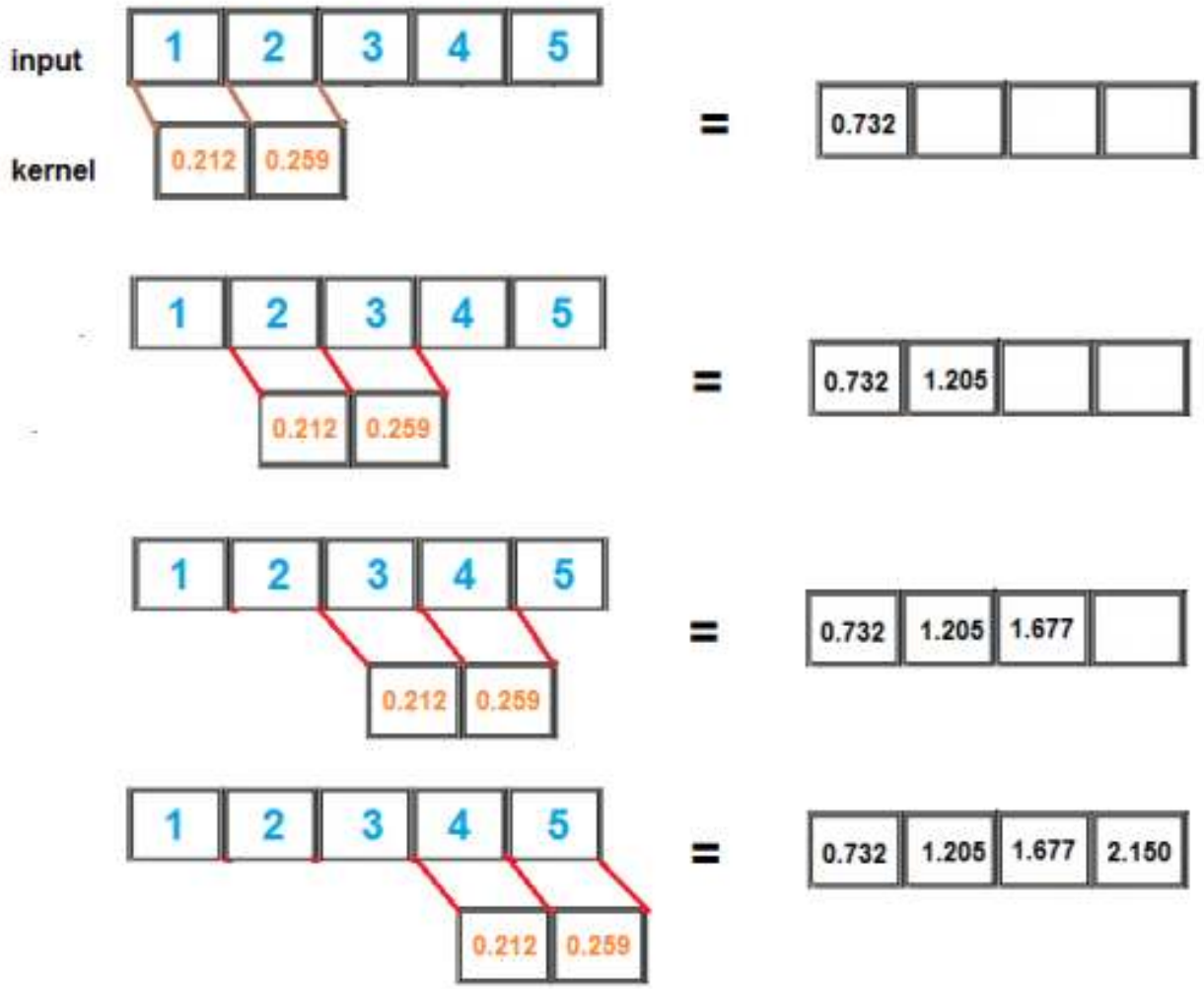
Pooling can also be applied in 1D:

- **max pooling** selects the maximum value in a window
- **average pooling** computes the mean



Example

Convolution with $k = 2$.



Pooling with $k = 2.$:

$$\text{maxpool}^{1\times 2} = [1.205, 2.150]$$

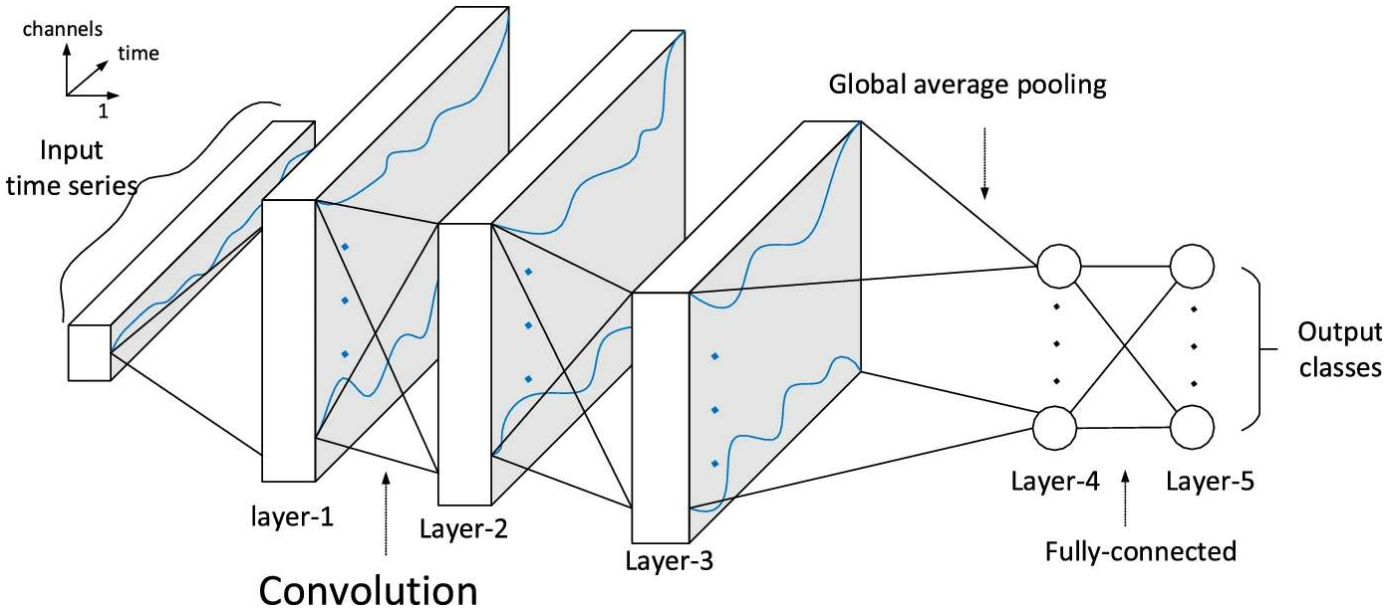
$$\text{avgpool}^{1\times 2} = [0.969, 1.914]$$

Multiple Kernels

Using multiple 1D kernels allows the network to:

- extract different temporal or sequential patterns
- produce multiple feature maps (channels)

Each kernel learns a different pattern.



InceptionTime

There are CNNs architecture developed specifically for time series.

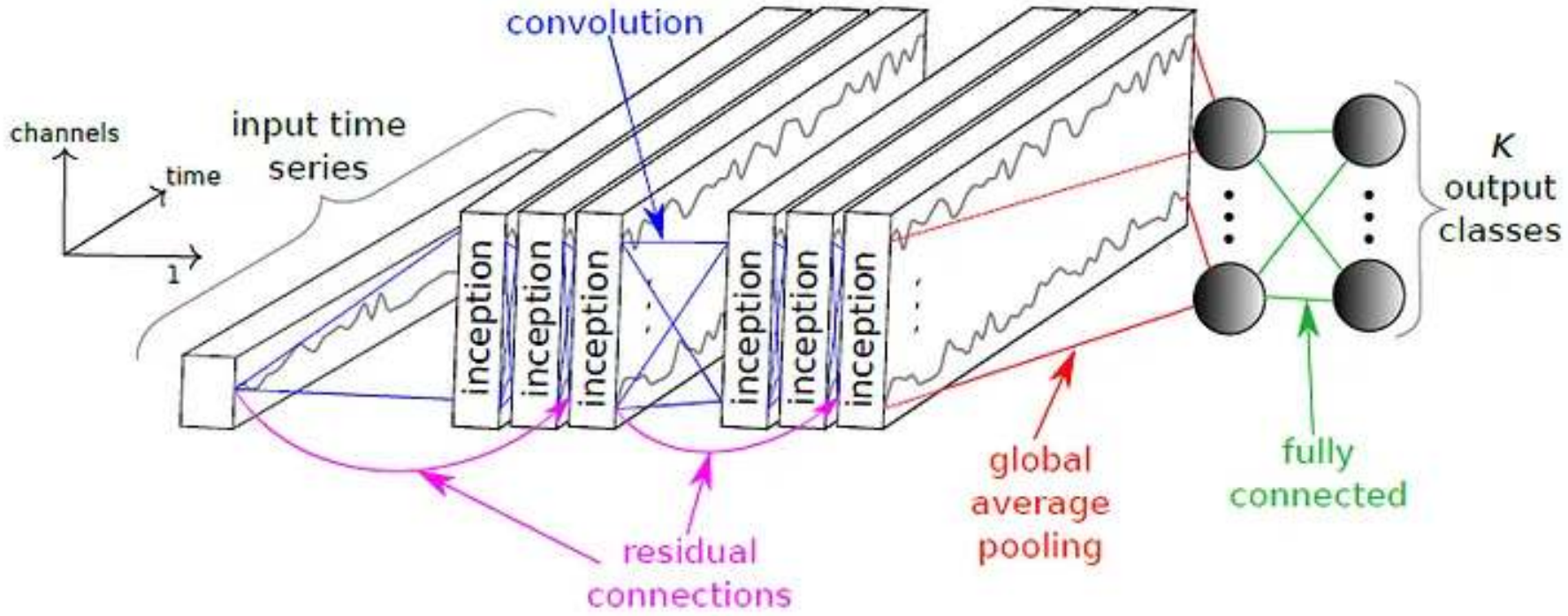
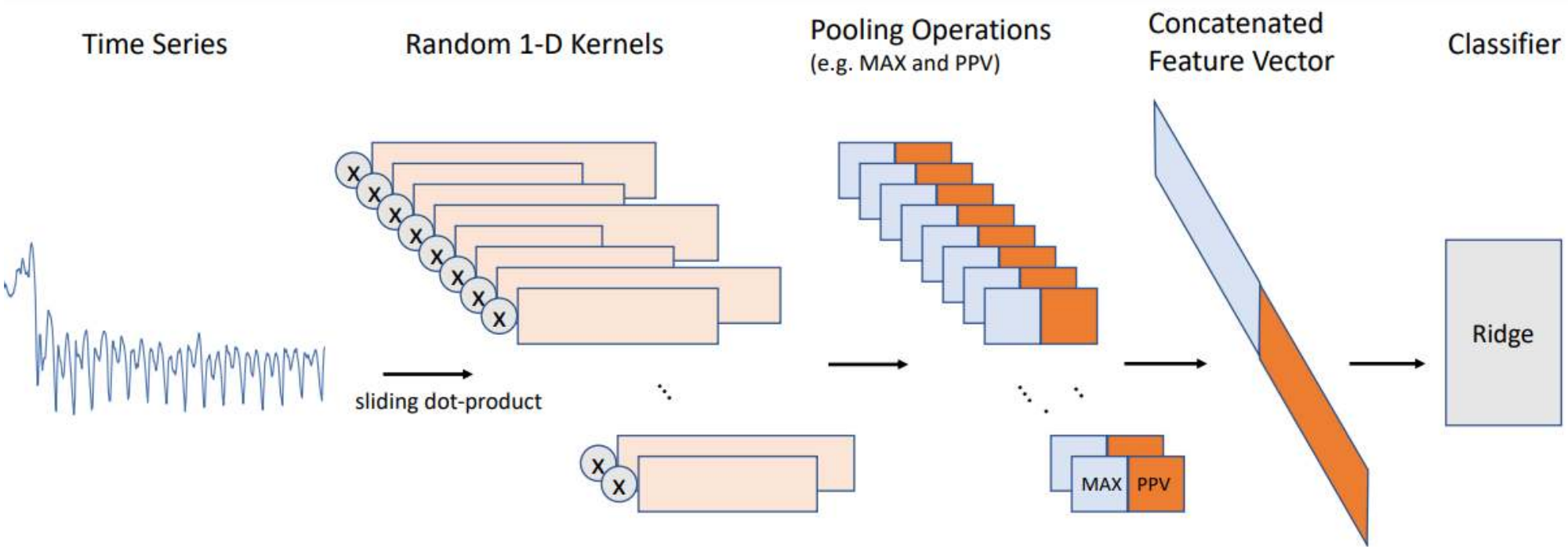


Fig. 1: Our Inception network for time series classification

Ismail Fawaz, Hassan, et al. "Inceptiontime: Finding alexnet for time series classification." (2020).

Rocket

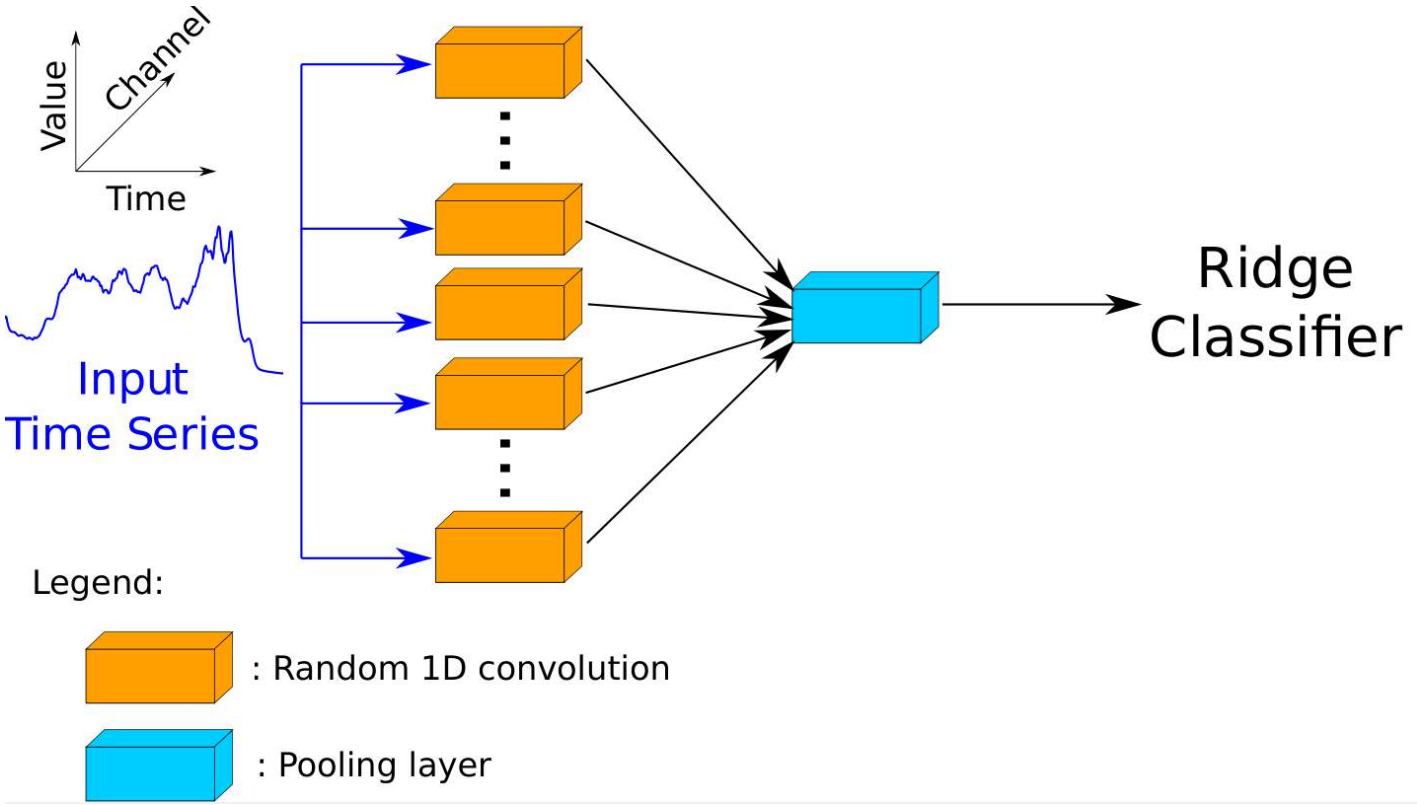
The state-of-the-art approach for time series classification is ROCKET



Rocket

Rocket is an interesting approach:

- it uses 10000 **random** convolutional kernels for convolution
- it uses max and ppv (percentage of positive values) global pooling
- trains only a regularized linear classifier at the end.



Dempster, A. et al. "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels." (2020)

Convolutional Neural Networks

Healthcare Applications

In General

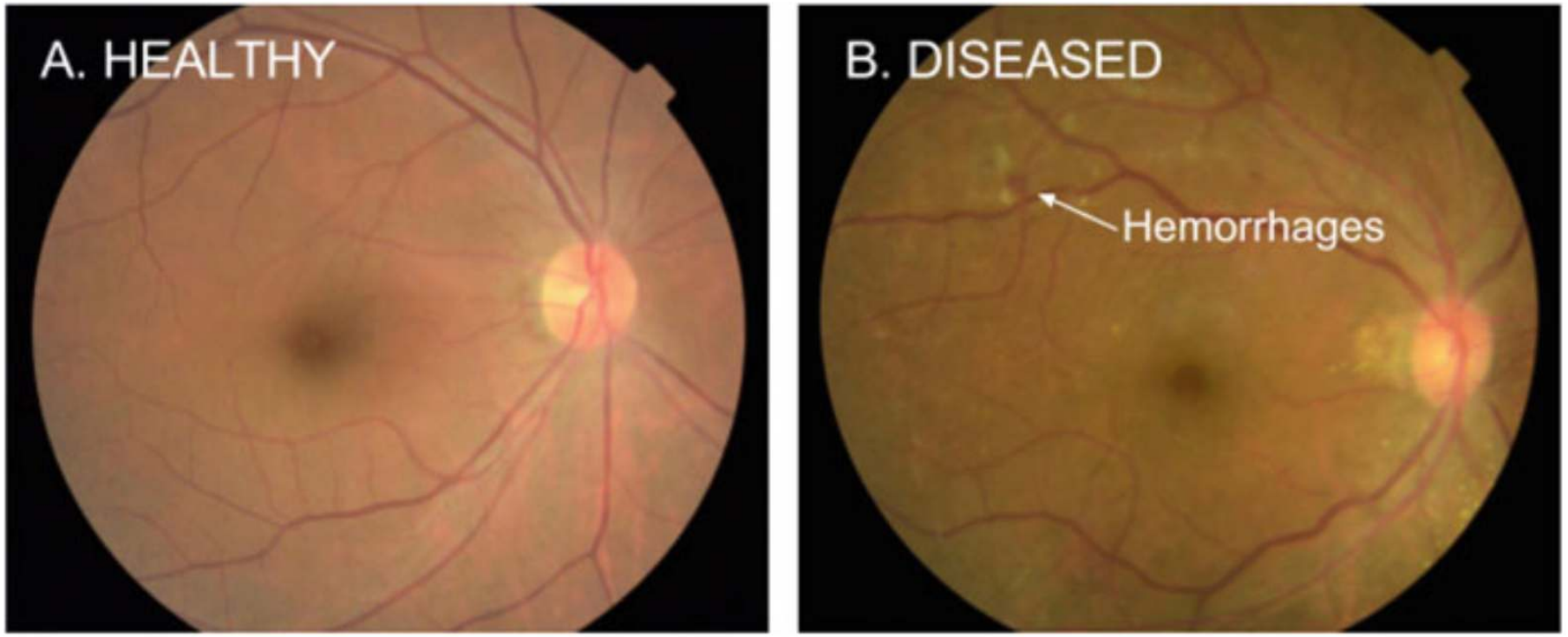
CNNs can tackle any problem where local patterns are relevant.

- For images: recognizing parts of the image that are relevant (e.g, skin lesions, cancer masses)
- For time series: recognize parts of vital signal that are important (e.g., abnormal heartbeat in ECGs, spikes in the blood pressure)

They are very good for classification problems.

Diabetic Retinopathy Detection

- Training data were macula-centered retinal fundus images
- Ophthalmologists graded images for the presence of diabetic retinopathy (none, mild, moderate, severe, or proliferative)



Gulshan, V., et al. 2016. “Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs.”

Detecting Skin Cancer

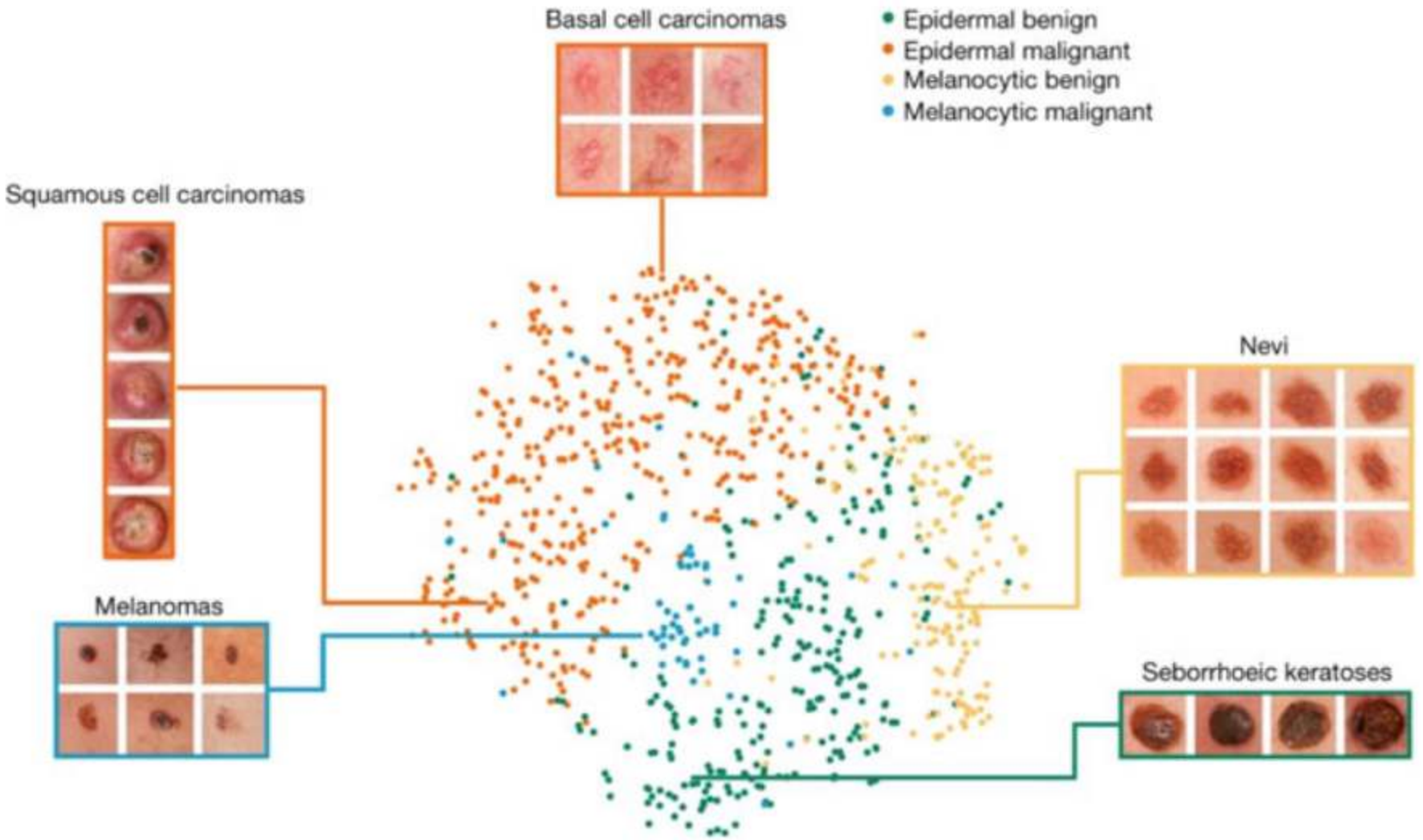
CNN using a dataset of 129,450 clinical skin images.

An Inception V3 model was pretrained on approximately 1.28 million images (1000 object categories) from the 2014 ImageNet Large Scale Visual Recognition Challenge.

To initialize with the pretrained model, the CNN model was further trained (i.e., finetuning) on the skin images.

The CNN model is trained using 757 disease classes.

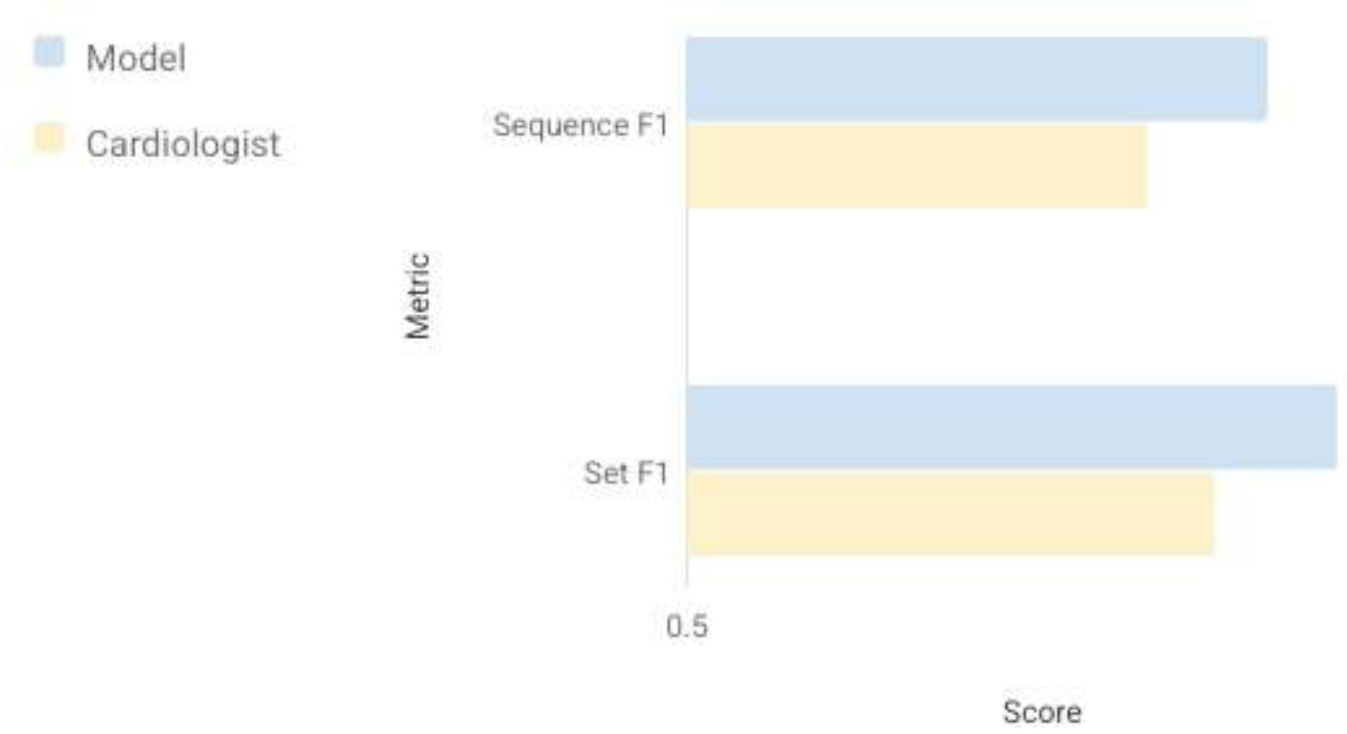
Detecting Skin Cancer

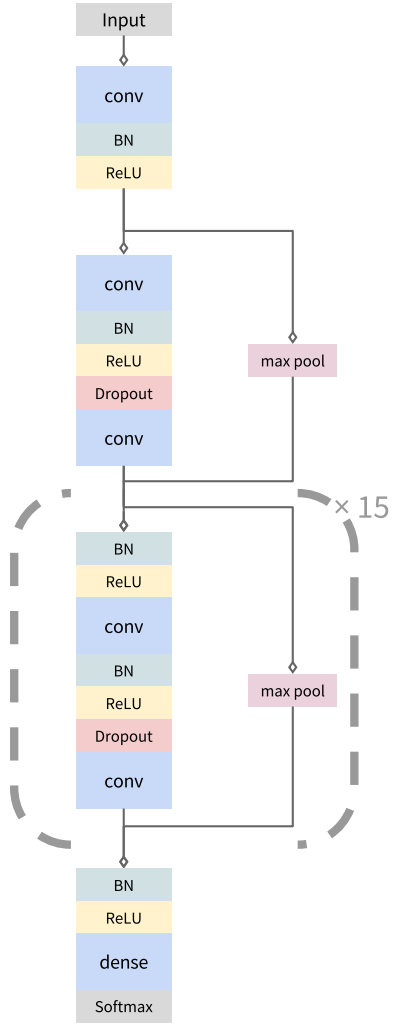


A. Esteva, et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature (2017)

Cardiologist-Level Arrhythmia Detection

They train a 34-layer convolutional neural network (CNN) to detect arrhythmias in ECG time-series.





Rajpurkar, Pranav, et al. "Cardiologist-level arrhythmia detection with convolutional neural networks." 2017