Workshop: Introduction to Deep Learning

Lecture 2 : Neural Networks and Deep Learning

Petia Georgieva

Department of Electronics Telecommunications and Informatics (DETI)
Institute of Electronics and Informatics Engineering of Aveiro (IEETA)
Intelligent Robotics and Systems Lab
University of Aveiro, Portugal (https://www.ua.pt/)
Email: petia@ua.pt

European Union-NextGenerationEU, through the National Recovery and Resilience Plan of the Republic of Bulgaria, project N0 BG-RRP-2.004-0005.

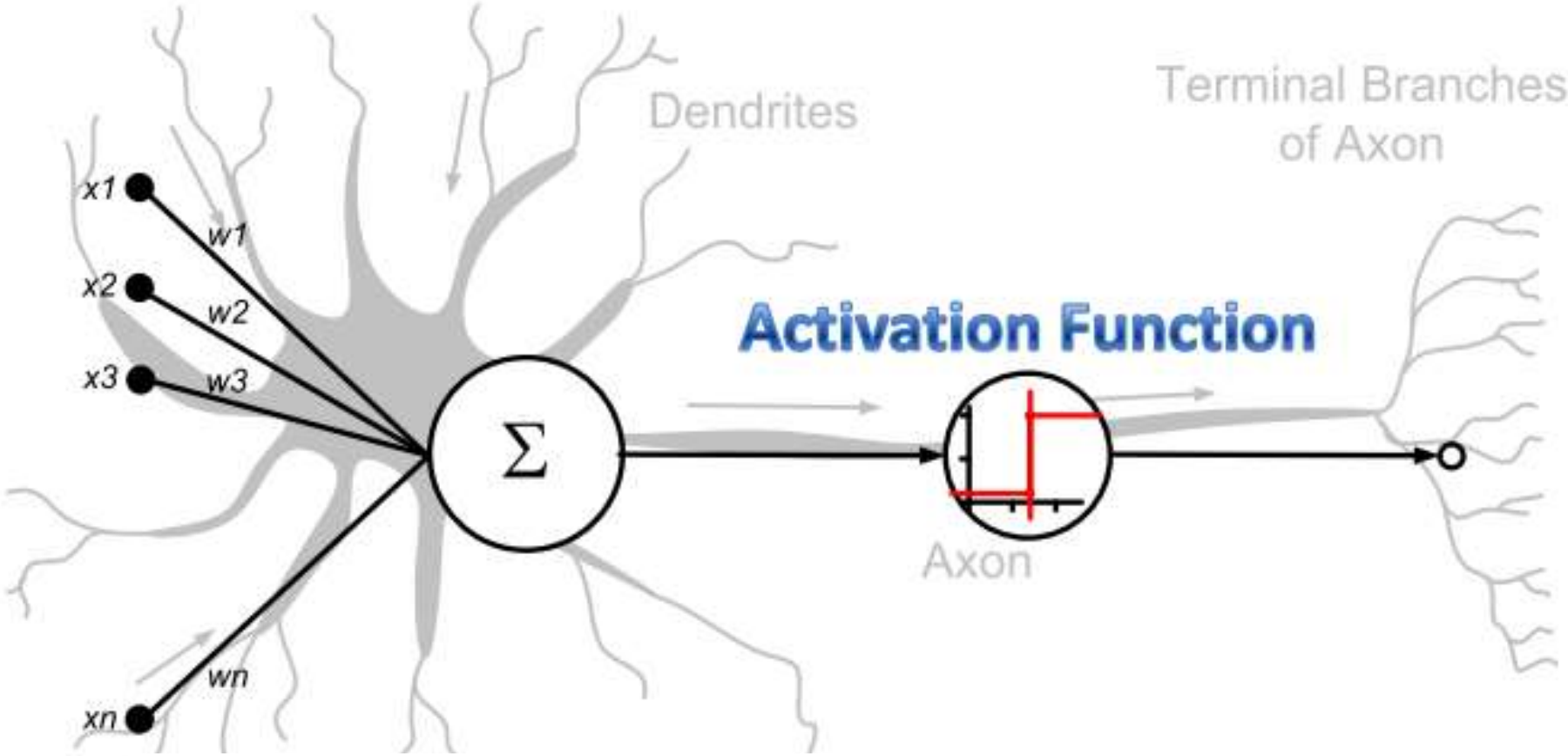
Outline

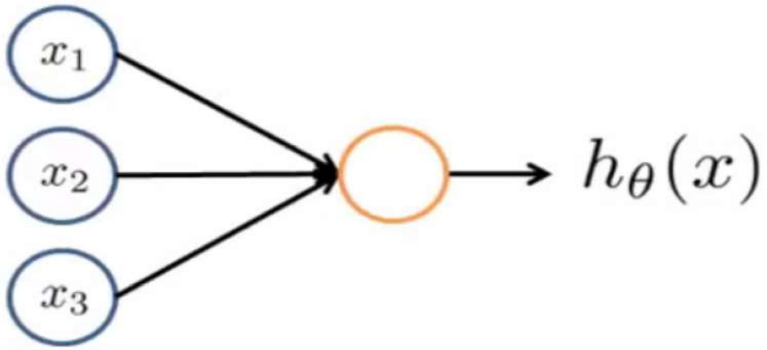
- Artificial Neural Networks (ANN) – classical architecture
- Machine Learning versus Deep Learning
- Convolutional Neural Networks (CNN)
- Image Segmentation - Unet

1. Artificial Neural Networks (ANN) – classical 3 layers architecture

Neuron model

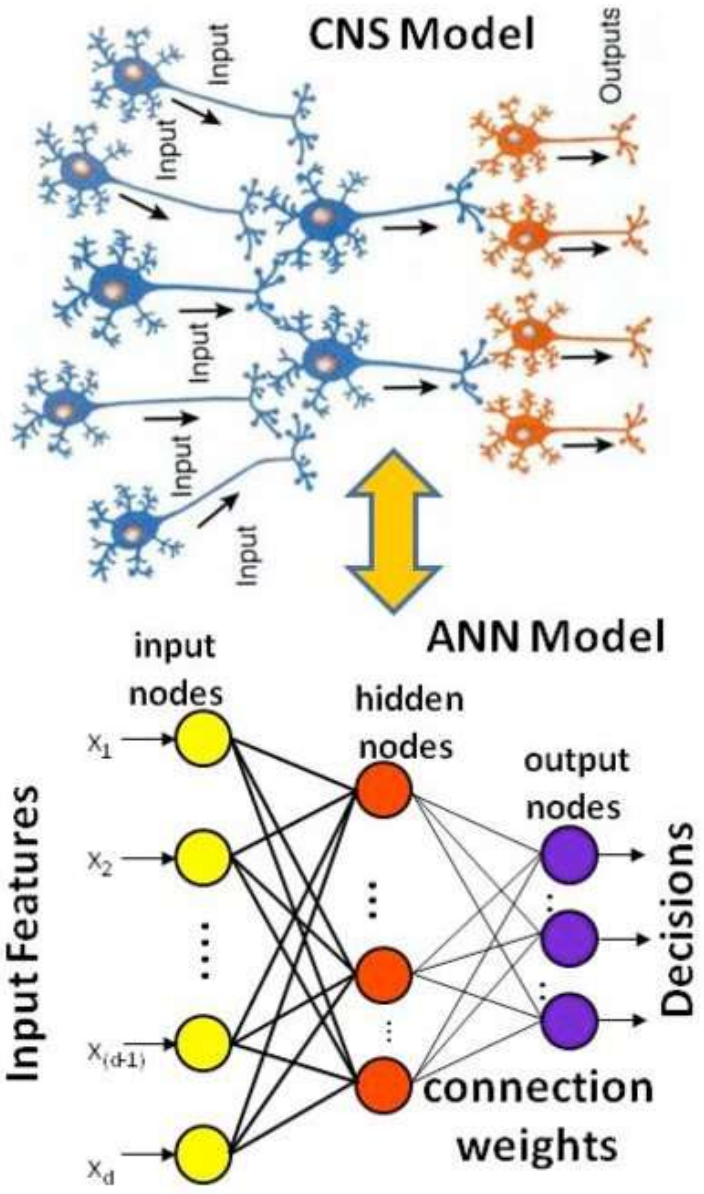
Origins: NN models inspired by biological neuron structures and computations.





Model of artificial neuron (unit, node)

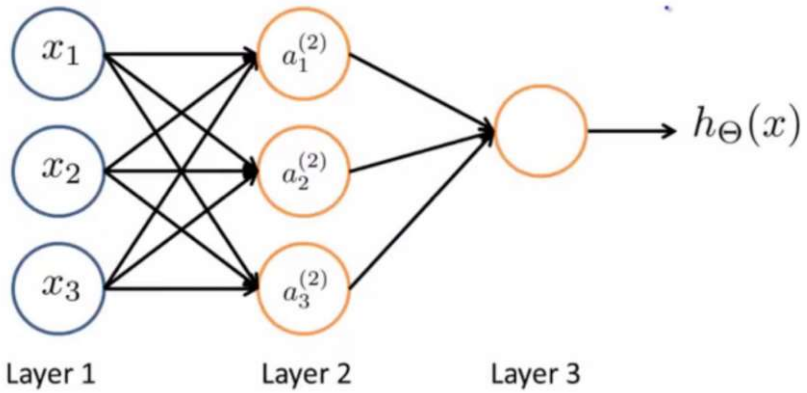
Analogy: natural learning



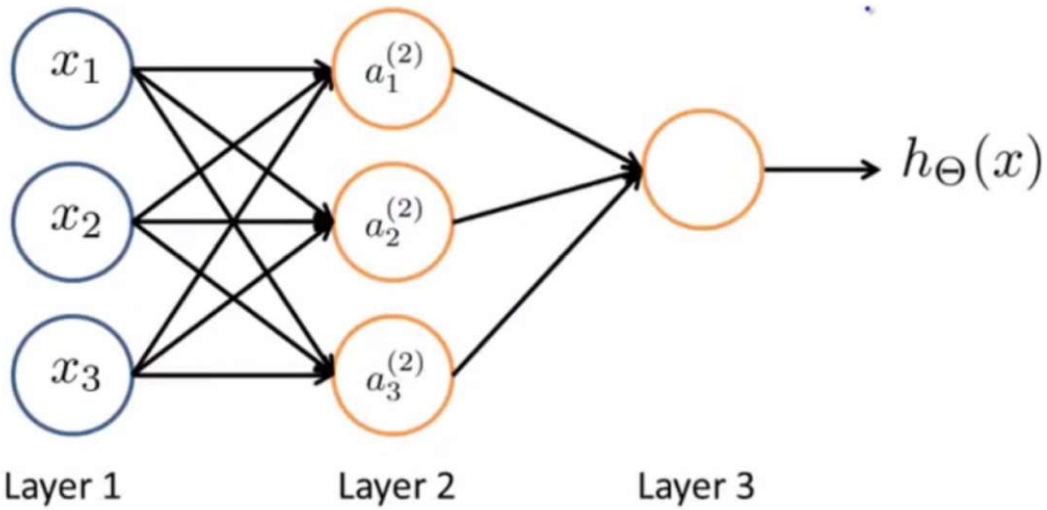
- Central Nervous System (CNS)
- The brain is a highly complex structure, **non linear** and highly **parallel;**
- It has an ability to organize its neurons, to perform complex tasks;
- A neuron is 5/6 times slower than a logical port;
- The brain overcomes this slowness through a parallel structure;
- The human cortex has 10 billion neurons and 60 trillion synapses!!

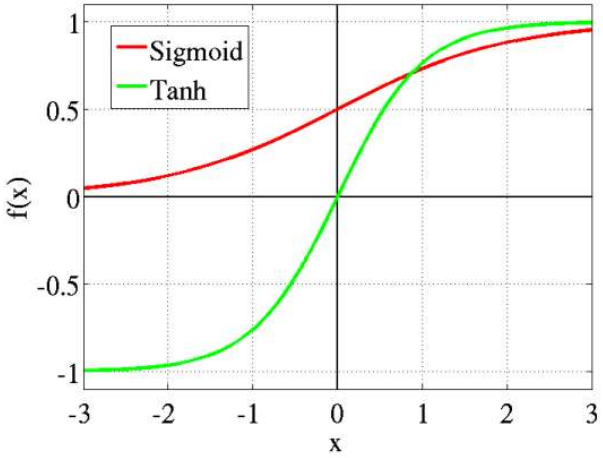
ANN

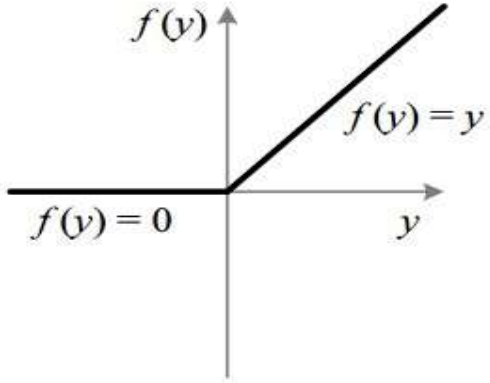
- Artificial Neural Networks (ANN) are models of machine learning that follow an analogy with the functioning of the human brain.
- **The relation between ANN and the brain is a very loose analogy !!!**
- A neural network is a parallel processor, consisting of simple processing (math) units (neurons).
- Knowledge is stored in the connections between the neurons.
- Knowledge is acquired from the data through a learning process (training algorithm) that adjusts the weights of the connections.

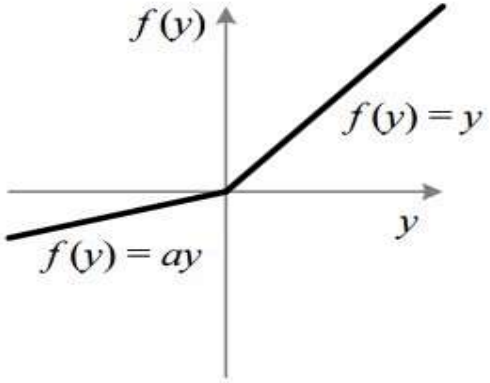


Typical Activation functions







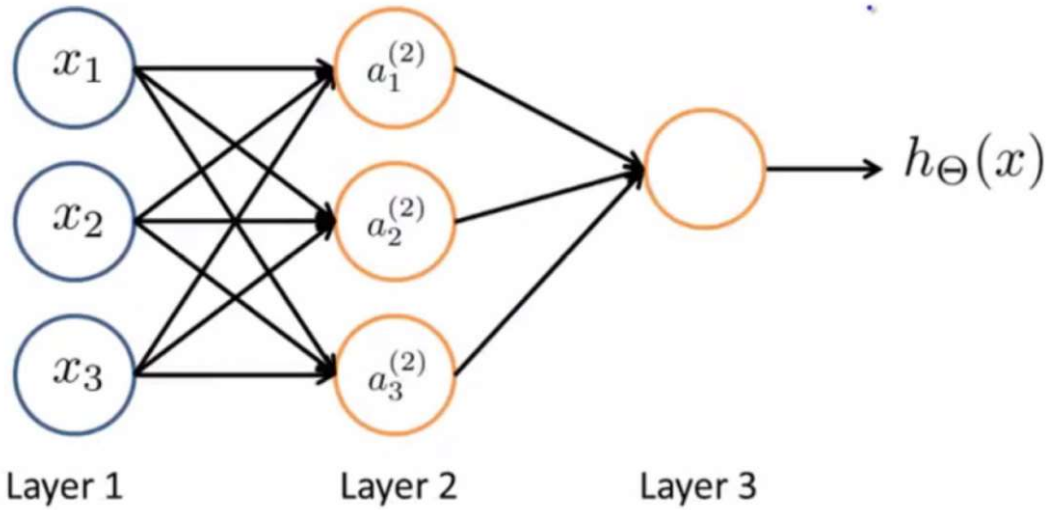


Sigmoid (logistic) vs. Hyperbolic tangent (Tanh)

ReLU (Rectified Linear Unit)

Leaky ReLU

NN - binary classification



2 classes { 0,1 } => one output unit (i.e. sigmoid function)

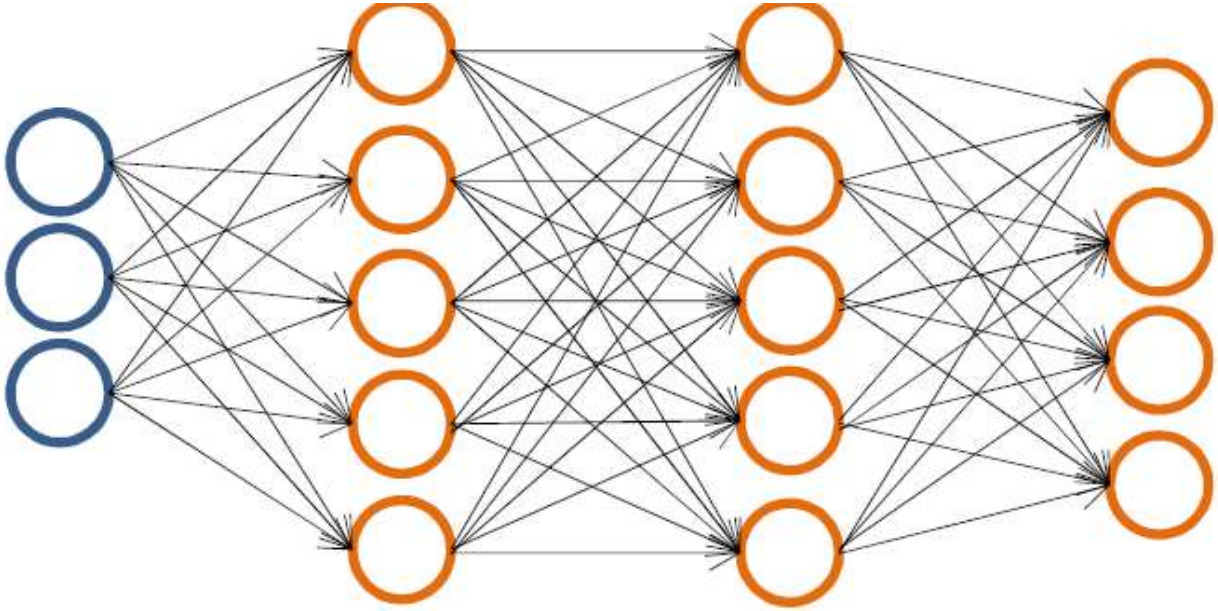
NN - multi-class classification

Pedestrian

Car

Motorcycle

Truck



$$h_\Theta(x) \in \mathbb{R}^4$$

K classes {1,2, K} => K output units

Model Training

Cost Functions:
Cross entropy (categorical, binary) – for classification
Mean Squared Error - for regression

Optimization methods:
- Gradient Descent
- Gradient Descent with momentum
- Backpropagation – the most used historically

- **RMSprop** (Root Mean Square propagation) - speed up gradient descent.

- **Adam** (Adaptive Moment Estimation) - combines Gradient Descent with momentum and RMSprop.

- Marquardt- Levenberg
- Rprop
- Quickprop

NN Parameters (weights) update

- **Start with some randomly initialized small values drawn from different distributions (Gaussian/Uniform/ Xavier/ Glorot's /LeCun)**

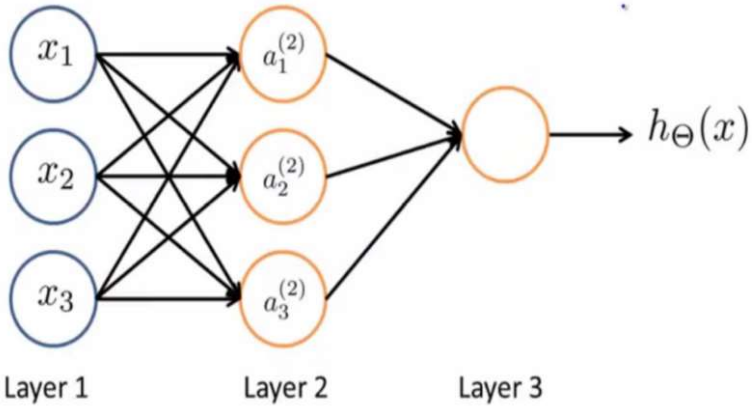
- **Then are iteratively updated:**

$$\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$$

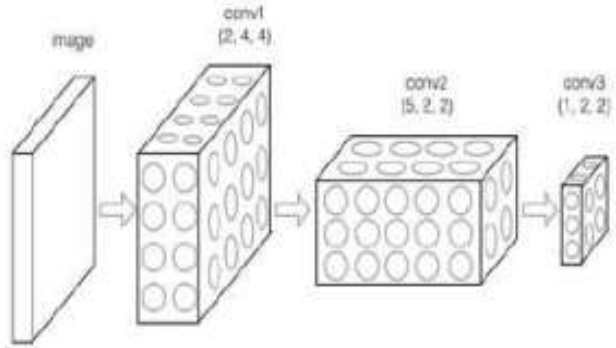
α **-Learning rate**

2. Machine Learning versus Deep Learning

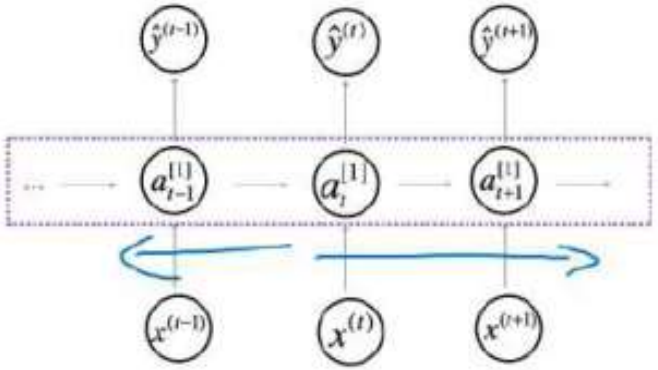
Machine Learning vs. Deep Learning



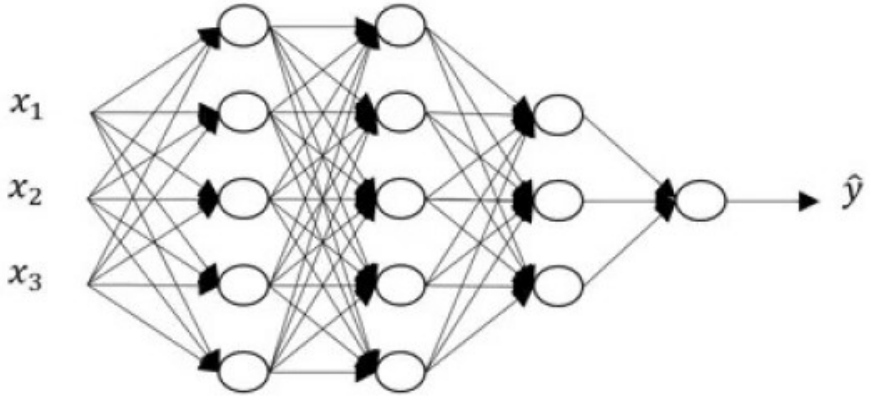
Standard NN



Convolutional NN



Recurrent NN



Fully Connected Feedforward Deep Neural Networks

Supervised learning with NN

Different types of neural networks for supervised learning:

- Standard (shallow, 3 Layers) NN - for Structured data

- CNN (conv nets) - in Computer vision (image processing)

- Recurrent neural networks (RNN) - in Speech recognition, machine transl.
  LSTM, GRU Natural Language Processing (NLP)

- Hybrid/custom NN/Mixture of NNs types- structured & unstructured data

Structured vs Unstructured data

Structured data (databases, tables)

Size	#bedrooms	...	Price (1000$s)
2104	3		400
1600	3		330
2400	3		369
⋮	⋮		⋮
3000	4		540

User Age	Ad Id	...	Click
41	93242		1
80	93287		0
18	87312		1
⋮	⋮		⋮
27	71244		1

Unstructured data (audio, image, video, text)

Audio

Image

Four scores and seven years ago...

Text

Supervised learning examples

95%-99% of the economic value criated by AI nowadays is one type of AI : Supervised Learning (x to y mapping)
x(input) => y(output)

Input(x)	Output (y)	Application	NN type
Home features	Price	Real estate	Standard
Ad + user info	Click on Ad (0/1)	On-line advertising	Standard
Image	Object (1,...1000)	Image analysis	ConvNet (CNN)
Audio	Text transcript	Speech recognition	RNN
Portuguese	English	Machine translation	RNN
Image+Radar info	Position of other cars	Autonomous driving	Custom/Hybrid

Conventional ML vs. Deep Learning

Conventional Machine Learning (ML)

- Needs feature engineering – hard, time-consuming task, requires expert knowledge.
- Don't work well with raw data (e.g. pixels/voxels of images).
- Better than DL if good features are found.

Deep Learning

- Representation-learning methods with multiple levels of representation.
- Layers that extract automatically features from raw data (latent structures not seen/difficult to be designed by humans)
- Needs large amounts of labelled training data
- Needs massive computational resources (e.g. GPUs, parallel solvers)

3. Convolutional Neural Networks (CNN)

Why Convolution Learning ?

Deep learning on large images

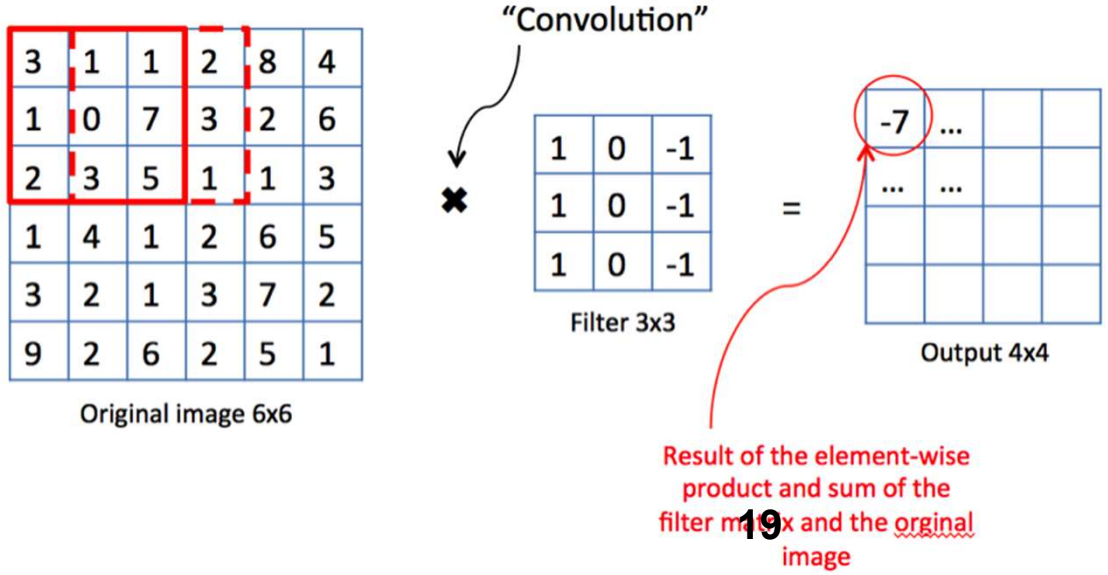
If we have a large image, e.g. 1000x1000x3 (RGB) pixels=>
3 million features (inputs)

If we take Fully Connected NN =>
we end up with billions of trainable parameters (weights)

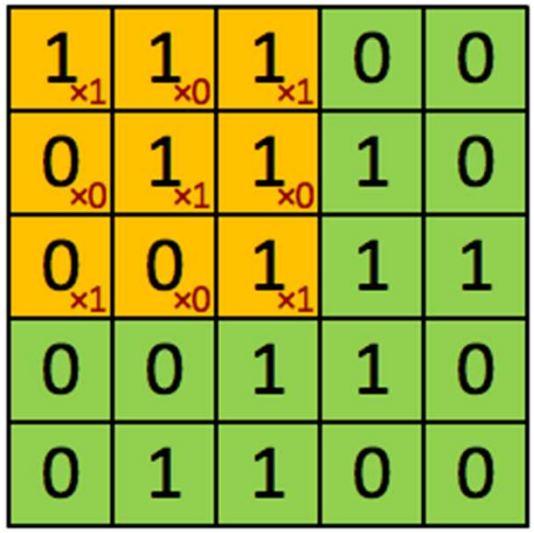
1st problem: Difficult to get enough labelled training data to prevent model overfitting.

2nd problem: Computational (memory) requirements to train such networks are not feasible.

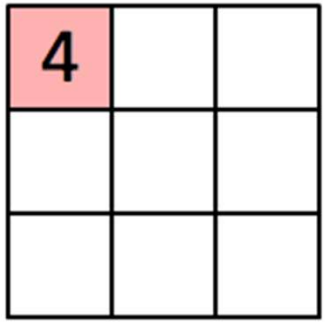
Solution: Convolution operations



OPERATION CONVOLUTION

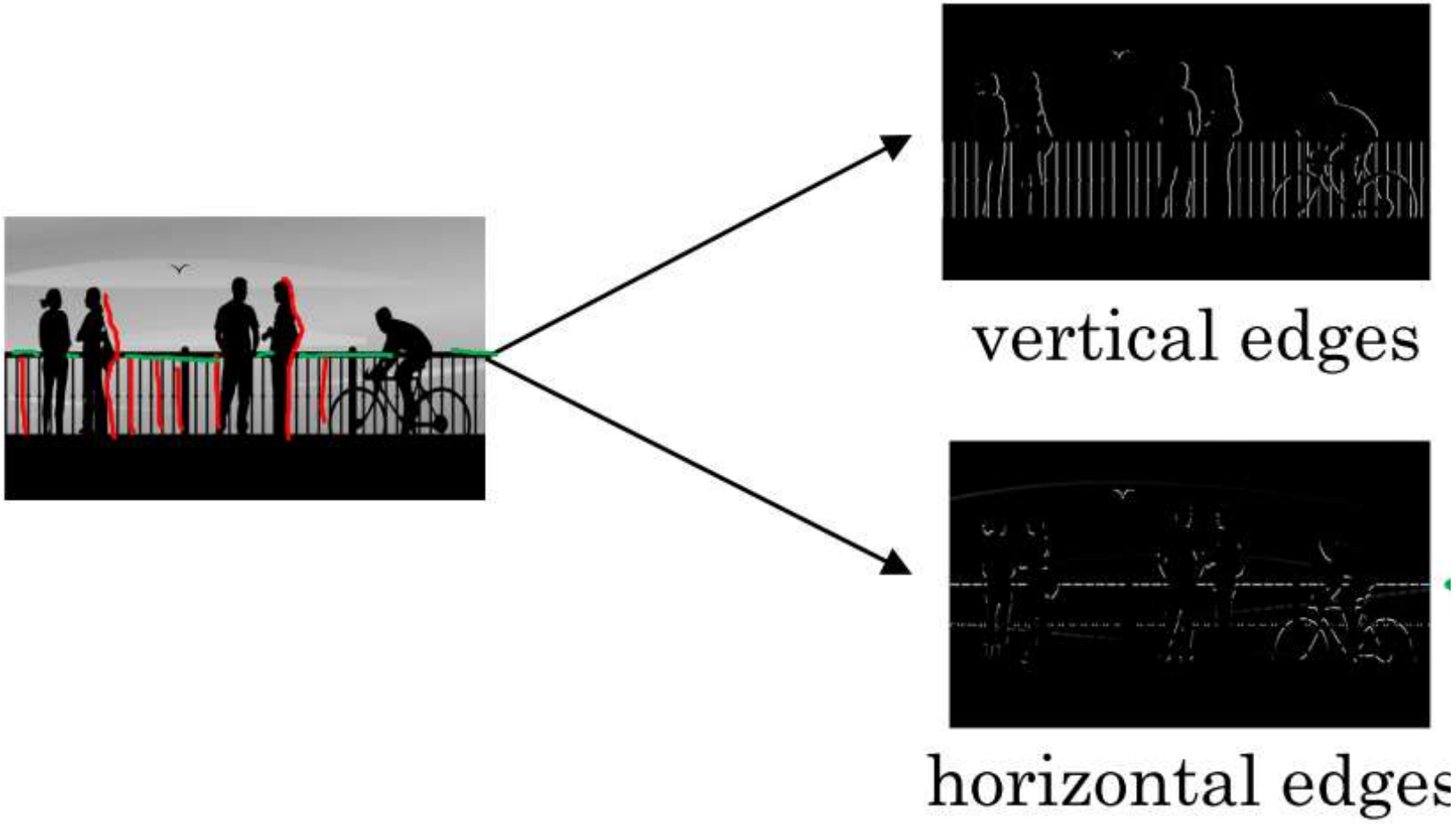


Image



Convolved Feature

Detect horizontal/vertical edges



VERTICAL EDGES DETECTOR

Ilustrative example:

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

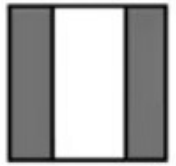
*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

*



Detection of bright to dark transition

Hand-picked convolutional filters (kernels)

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

=> Horizontal edge detector

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$\Rightarrow$ **Sobel filter**

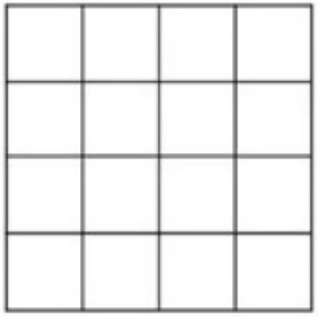
$$\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$$

$\Rightarrow$ **Sharr filter**

CONVOLUTIONAL FILTERS (KERNELS)

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



Hand-picking the filter values is difficult.
Why not let the computer to learn them?
Treat the filter values as the trainable parameters (w), and let the computer learn them automatically.
Other than vertical and horizontal edges, such computer-generated filter can learn information from different angles (e.g. 45°, 70°, 73°) and is more robust than hand-picking values.

By convention the conv filter is a square matrix with odd size (typically 3x3; 5x5; 7x7, also 1x1) .
It is nice to have a central pixel and it facilitates the padding.

PADDING

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -5 & -4 & 0 & 8 \\ -10 & -2 & 2 & 3 \\ 0 & -2 & -4 & -7 \\ -3 & -2 & -3 & -16 \end{bmatrix}$$

Ex. Take 6 × 6 input matrix, apply 3 × 3 conv filter, get 4 × 4 output matrix, because we shift the filter one row down or one column right and there are only 4 × 4 possible positions for this filter to convolve with the input matrix.

In general: given $n \times n$ input matrix and $f \times f$ *conv* filter, the convolution operation will compute $(n-f+1) \times (n-f+1)$ output matrix by applying one pixel shift rule.

1st problem: The input matrix will rapidly shrink (become very small) if we have many convolution layers.

2nd problem: Pixels on the corner of the image are used only once while pixels in the centre of the image are used many times. This is uneven, loose of inf.

Solution: Padding.

SYMMETRIC PADDING

Add p extra columns and rows at the image borders with 0 values (zero padding). Output matrix size;

$$(n+2p-f+1) \times (n+2p-f+1)$$

"valid" convolution =>

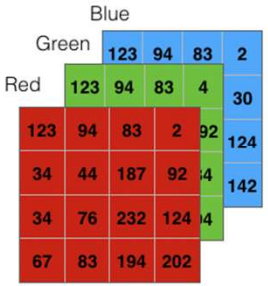
no padding

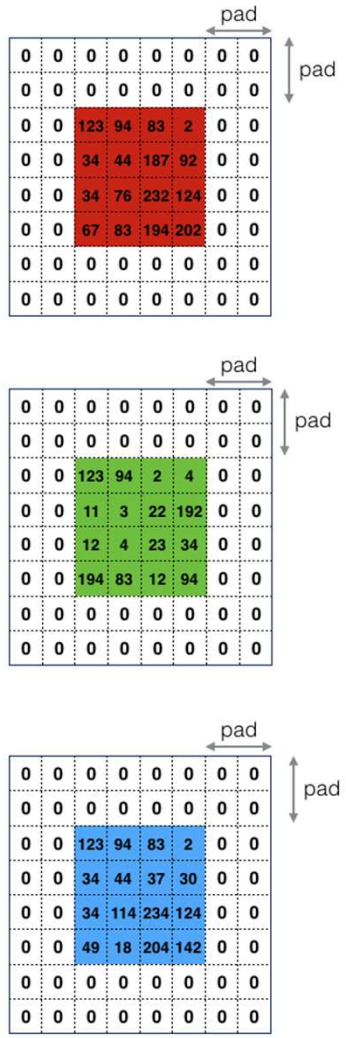
"same" covolution =>

Pad so that output size is the same as the input size. Formula for choosing p

(f is usually odd number!):

=





$$\begin{aligned} n + 2p - f + 1 &= n \\ 2p - f + 1 &= 0 \\ 2p &= f - 1 \\ p &= (f-1)/2 \end{aligned}$$

STRIDED CONVOLUTION

$$\begin{bmatrix} 2 & 3 & 7 & 4 & 6 & 2 & 9 \\ 6 & 6 & 9 & 8 & 7 & 4 & 3 \\ 3 & 4 & 8 & 3 & 8 & 9 & 7 \\ 7 & 8 & 3 & 6 & 6 & 3 & 4 \\ 4 & 2 & 1 & 8 & 3 & 4 & 6 \\ 3 & 2 & 4 & 1 & 9 & 8 & 3 \\ 0 & 1 & 3 & 9 & 2 & 1 & 4 \end{bmatrix} * \begin{bmatrix} 3 & 4 & 4 \\ 1 & 0 & 2 \\ -1 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 91 & 100 & 83 \\ 69 & 91 & 127 \\ 44 & 72 & 74 \end{bmatrix}$$

Stride: how many pixels (steps) we shift to the right or down after each convolution.

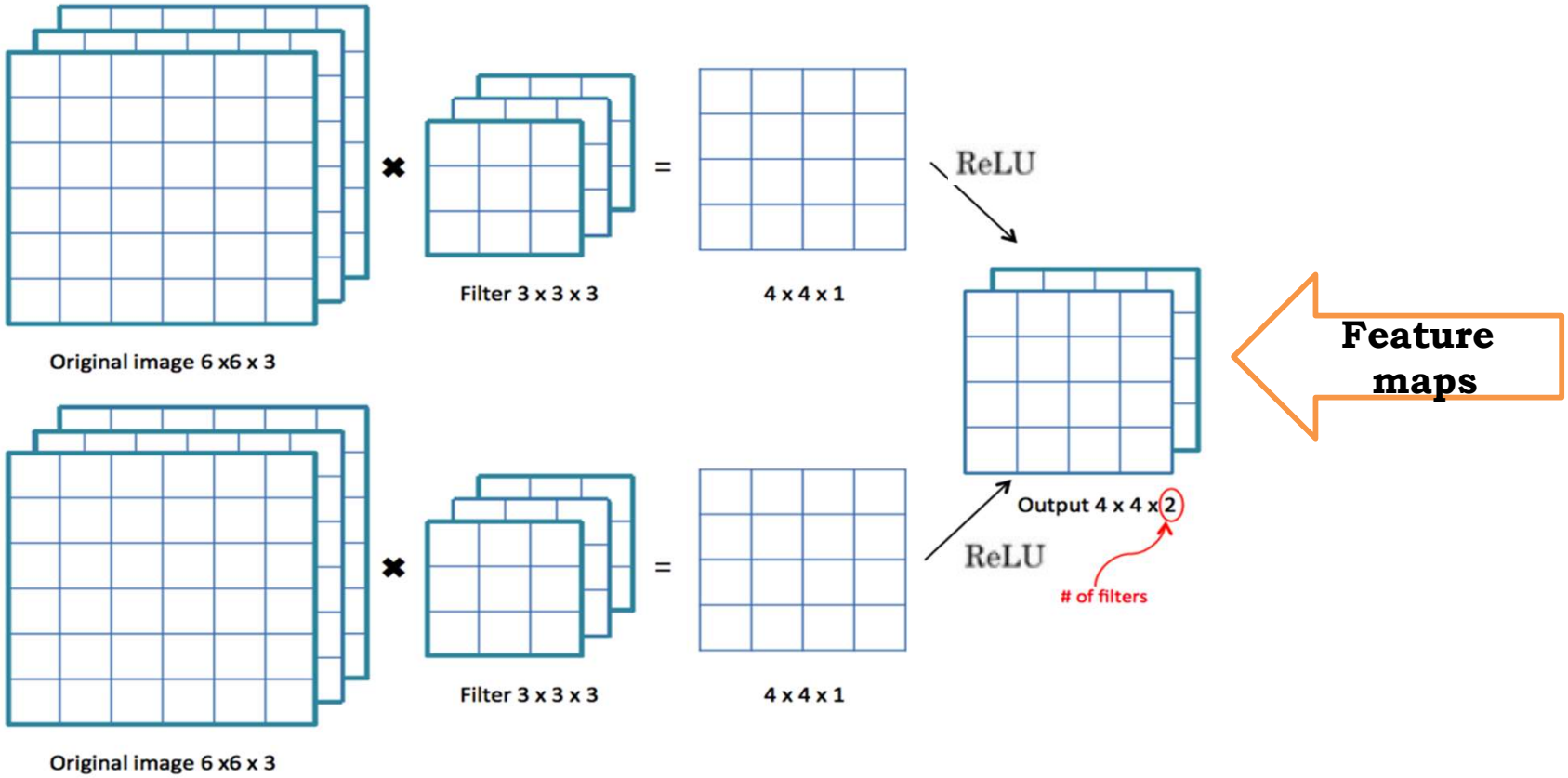
In general: given $n \times n$ input matrix and $f \times f$ filter with padding p and stride s, the convolution operation will compute output matrix with size:

$$\lfloor \frac{n+2p-f}{s} + 1 \rfloor \text{ by } \lfloor \frac{n+2p-f}{s} + 1 \rfloor$$

If it gives a non-integer value => the closest lower integer will be chosen.

Ex.: no padding (p=0) , stride s=2 => (7 + 0 − 3)/2 + 1 = 4/2 + 1 = 3 => 3 ×3 matrix

Multiple Conv Filters over Volumes (3D filters)

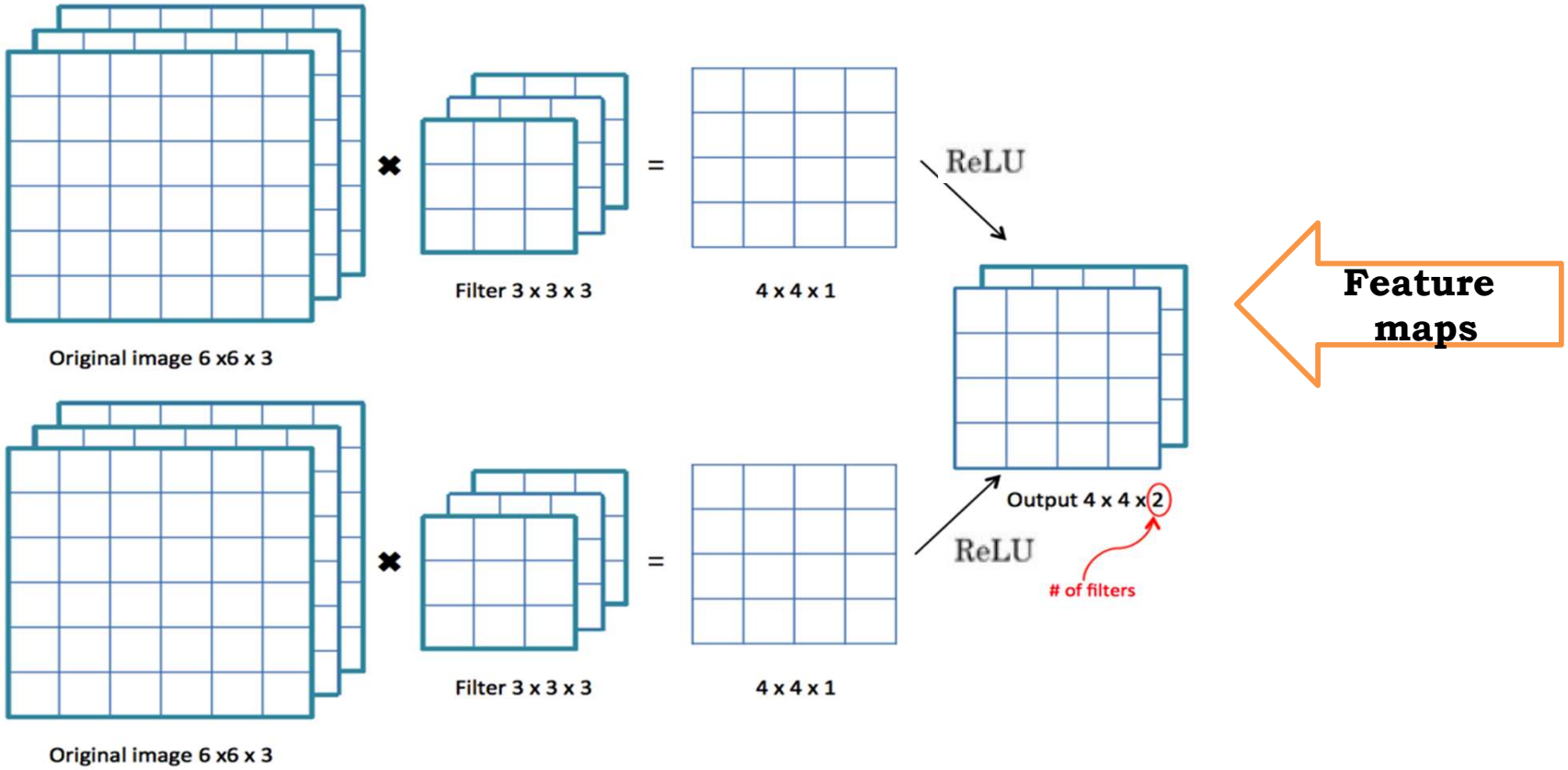


RGB images have 3 dimensions: height, width, and number of channels (3D volume).
The conv filter will be also a 3D volume : $f \times f \times 3$ (number of channels has to be equal in the image and the filter)

*$(n \times n \times n_c)$ image * $(f \times f \times n_c)$ filter => (n-f+1) x (n-f+1) x n_filters (no padding)*

Play conv_kiank

ONE CONV LAYER OF CNN

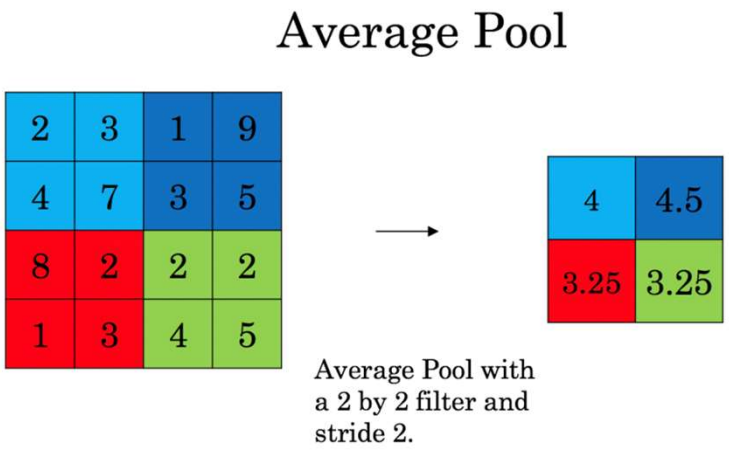


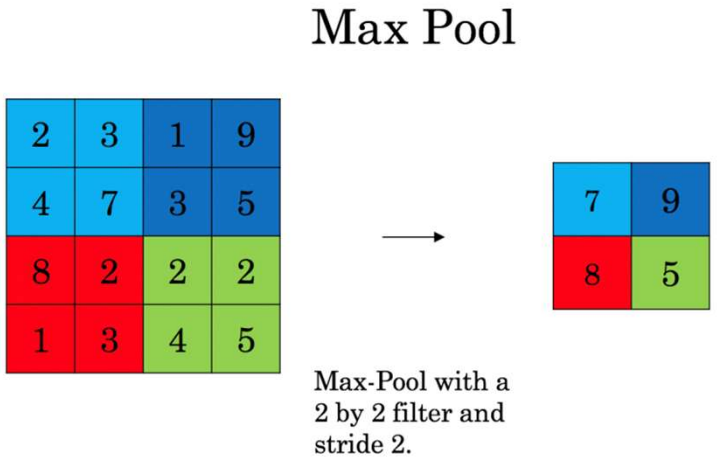
Different 3D filters (kernels) are applied to the 3D input image and the result matrices are stacked to form a 3D output volume.

After the convolution operation the result is passed through an activation function (e.g. ReLU, sigmoid, linear, etc.).

The outputs of the conv layers are known as feature maps.

POOLING (POOL)





Pooling operation reduces the size of the representation to speed up the computation and make the features more robust.

Ex. Divide the input in regions (e.g. 2 × 2 filter), choose stride (e.g. s=2), each output will be the max (**max pooling**) or the average (**average pooling**) from the corresponding regions.

Some intuition:
Large number means there is some strong feature (edge, eye) detected in this part of the image, which is not present in another part.
Max Pool: whenever this feature is detected it remains preserved in the output.

Softmax Layer

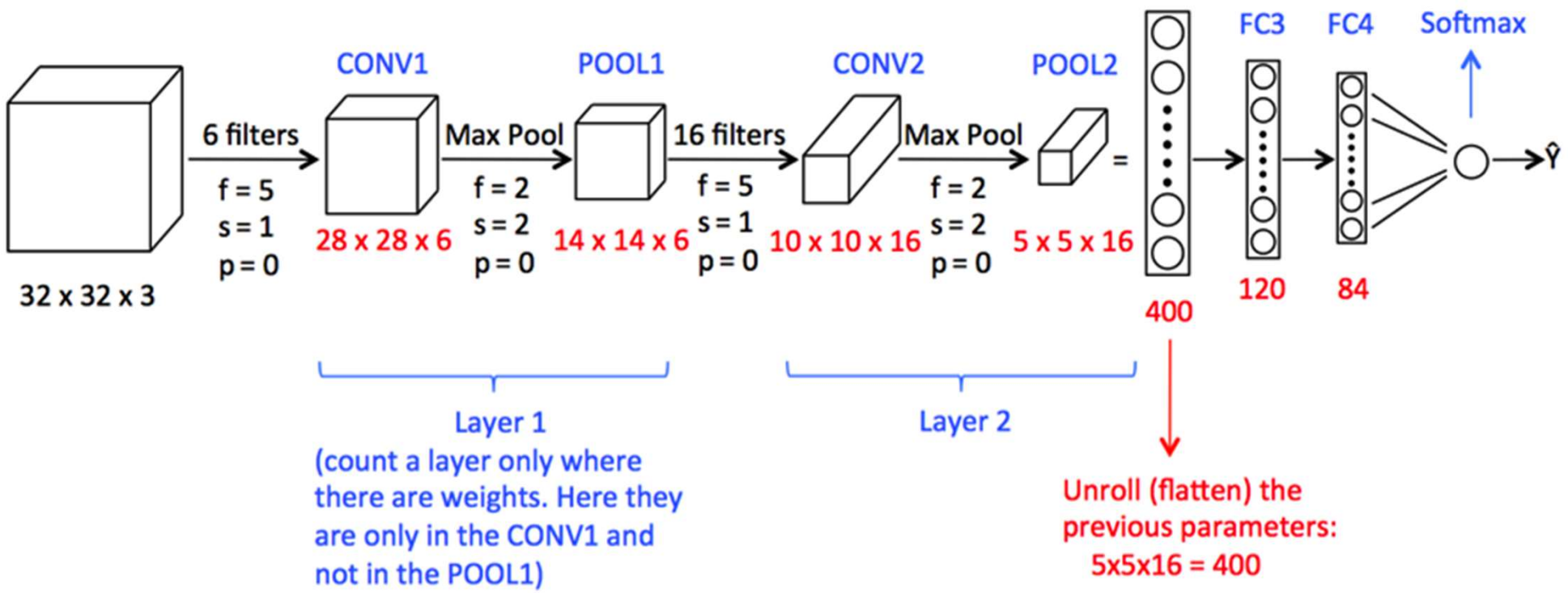
Softmax Layer (SL) estimates the probability that an example $x^{(i)}$ belongs to each of the k classes ((j=1,2,..k).

$$p(y^{(i)} = j|x^{(i)};\theta) = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}$$

SL outputs *k* dimensional vector with estimated probability for each class k:

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1|x^{(i)};\theta) \\ p(y^{(i)} = 2|x^{(i)};\theta) \\ \vdots \\ p(y^{(i)} = k|x^{(i)};\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

Classical CNN Example: LeNet5*

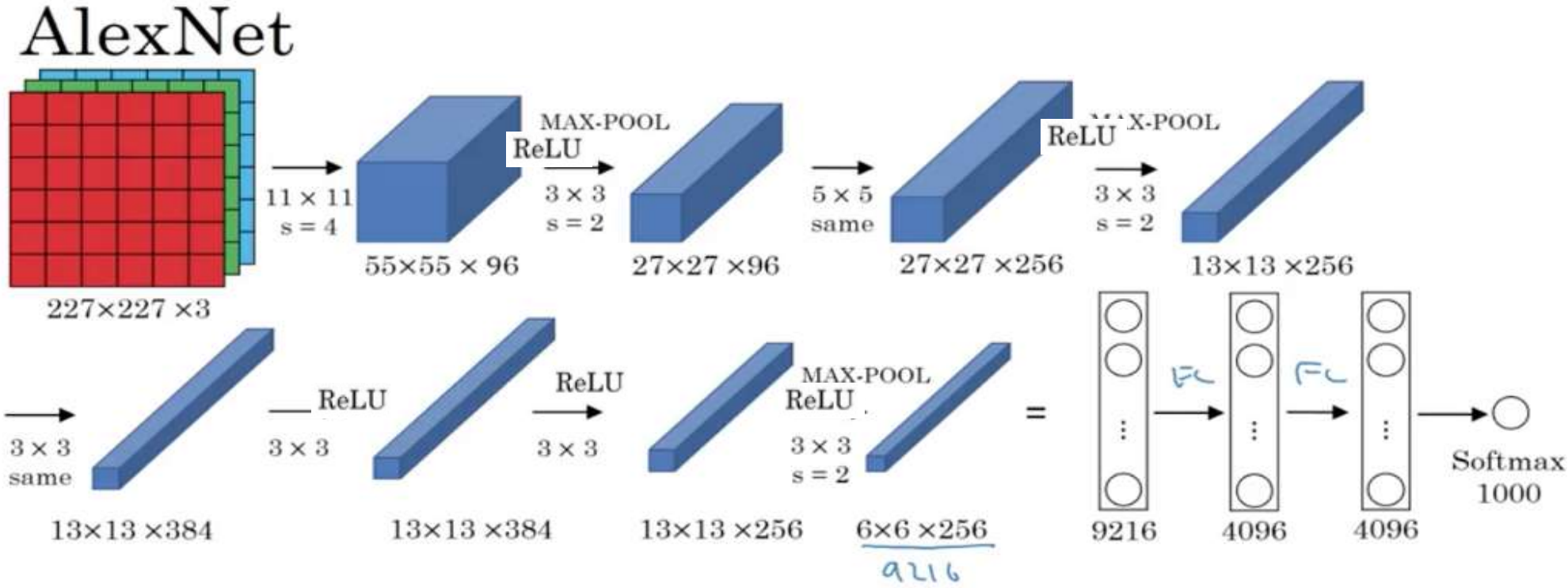


***LeCun et al., 1998, "Gradient-based learning applied to document recognition".** Original LeNet5 applied to handwritten digit recognition (grey scale images). Avg pooling, no padding, softmax classifier; ReLU and sigmoid/tanh neurons in the Fully Connected (FC) layers.

The activation function is always present after the convolution, even if it is not drawn on the CNN diagram.

General trend: CNNs start with large image, then height and width gradually decrease as it goes deeper in the network, where as the number of channels increase.

AlexNet*

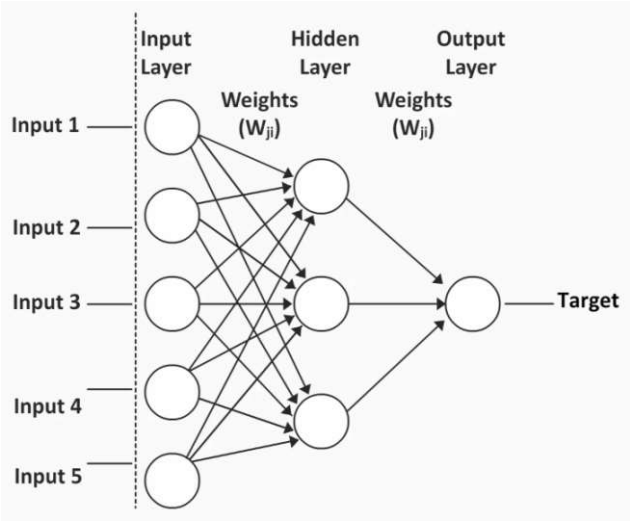


*** Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, 2012, ImageNet classification with deep convolutional neural networks.**
5 conv layers, 3 FC layers with softmax output, 60 million parameters in total.
AlexNet applied to ImageNet LSVRC-2010 dataset to recognize 1000 different classes.

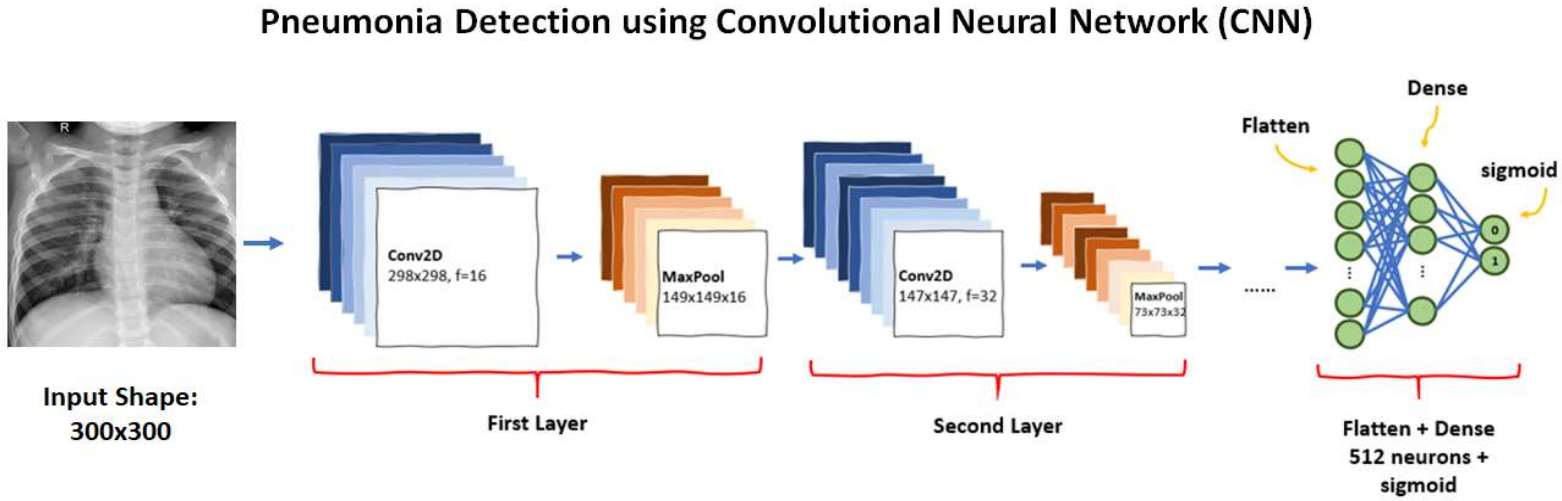
This paper convinced the Computer Vision (CV) community that DL really works and will have a huge impact not only in CV but also in speech/language processing.

Image Processing

FFNN: Transform the image into a pixel vector x
=> train NN model to detect a pathology (1, 0)



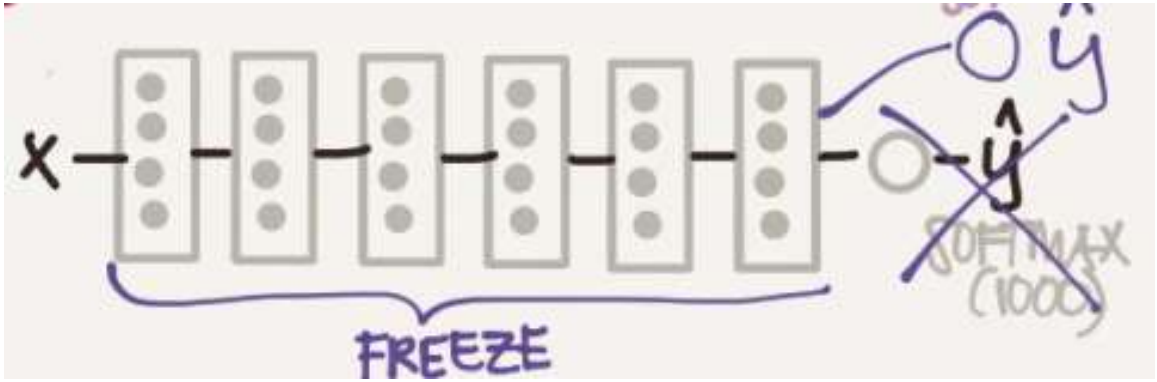
CNN: keep the 2D/3D image structure



Transfer Learning

- Training of deep NN may takes weeks/months, many GPUs, and is very expensive. Better use a pre-trained model (with pre-trained weights) => this we call Transfer Learning

Ex.: You have a small training set and a small number of classes (e.g. 3).
- Take a DNN trained for 1000 classes.
- Substitute the last classification layer (softmax with 1000 outputs)) with a softmax with 3 outputs.
- Freeze the parameters in all other layers, train only the last layer.



Data Normalization

1) Transforms features/data by scaling it to a given range.

```
from sklearn.preprocessing import MinMaxScaler

mms = MinMaxScaler(feature_range=(0, 1), copy=True)

mms.fit(data)
data_transformed = mms.transform(data)
```

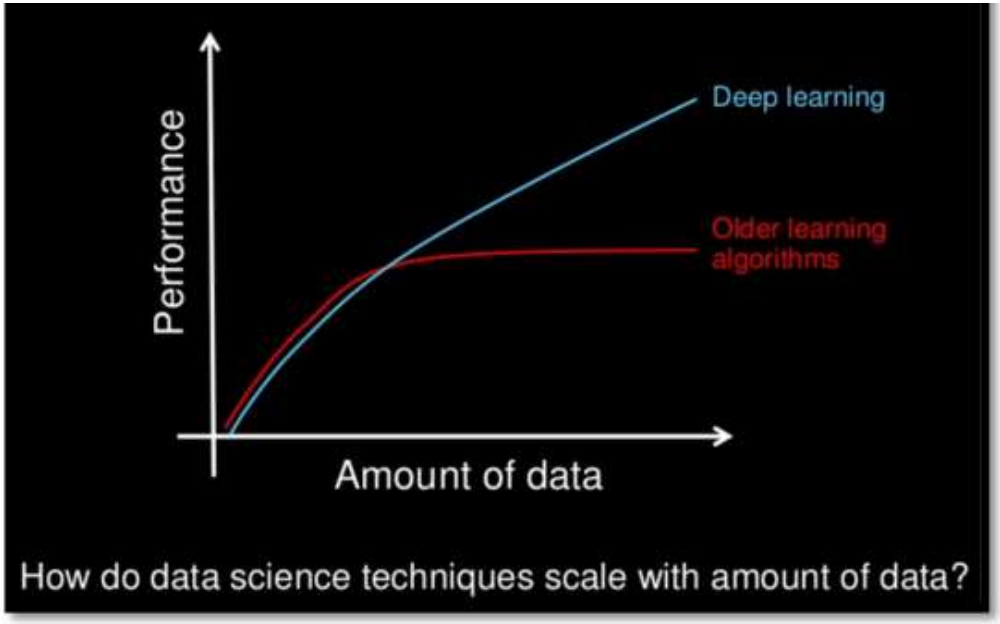
2) Standardize features by removing the mean and scaling to unit variance.

```
from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

sc.fit(data)

data_transformed =sc.transform(data)
```

$$x' = \frac{x - \mu}{\max(x) - \min(x)}$$

$$x' = \frac{x - \mu}{\sigma}$$

Why Deep Learning is successful (since 2015) ?

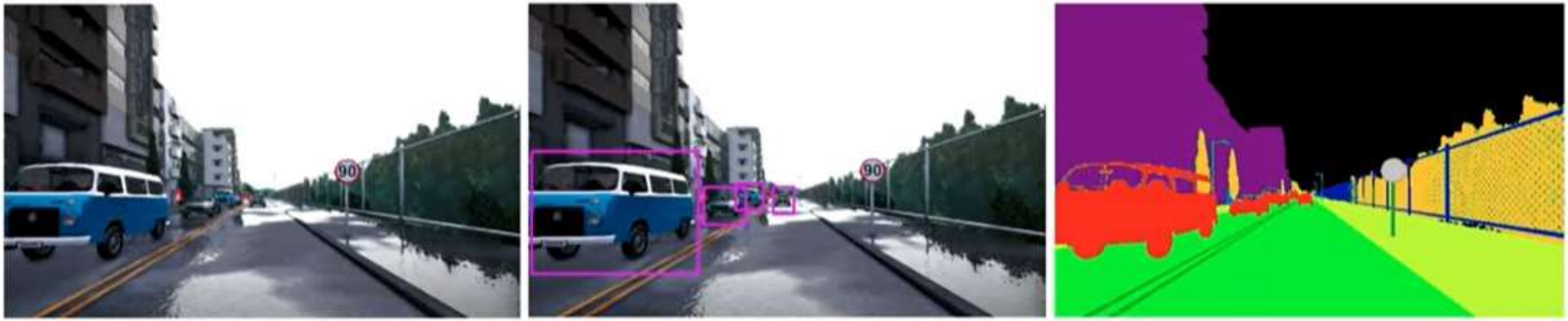


GPU (NVIDIA !!!) - workhorse of modern AI
Cloud Computing & Parallel graphic processing

AlexNet had beaten (by huge margin) the best handcrafted software written by computer vision experts. Trained with 1 million images that required trillions of math operations on NVIDIA GPUs.

PART 4 Image Segmentation

Object detection vs Semantic Segmentation

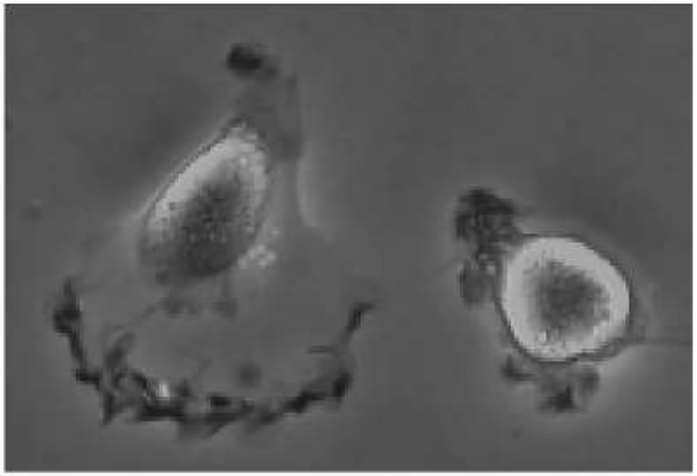


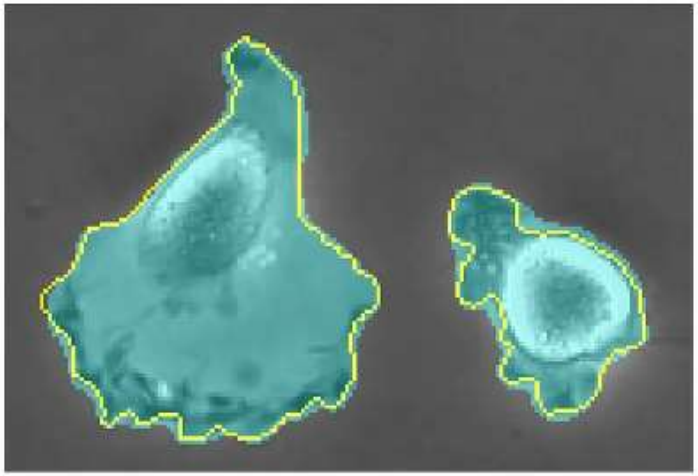
Input image | Object Detection | Semantic Segmentation

Object detection - the goal is to put a bounding box around a found object.

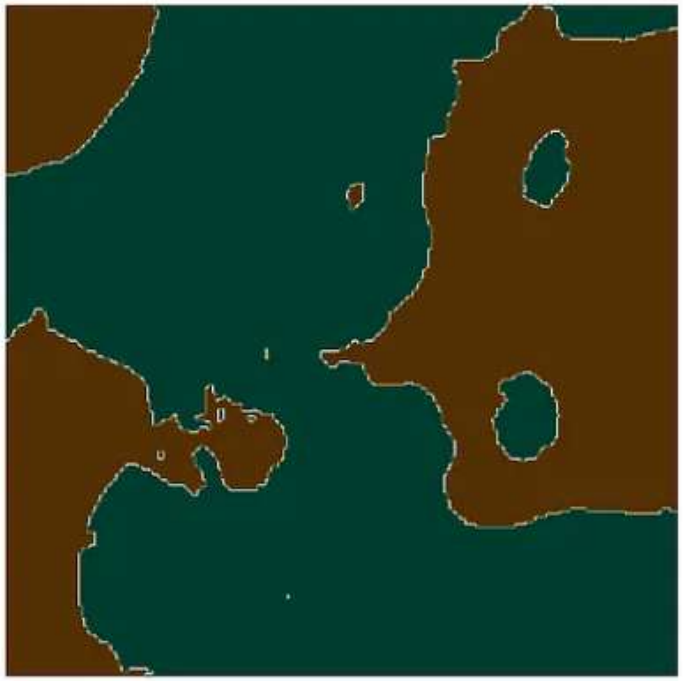
Semantic segmentation - more sophisticated learning algorithm, the goal is to know exactly which pixels belong to the object and which pixels don't, to know what is every single pixel in the image.

Semantic Segmentation



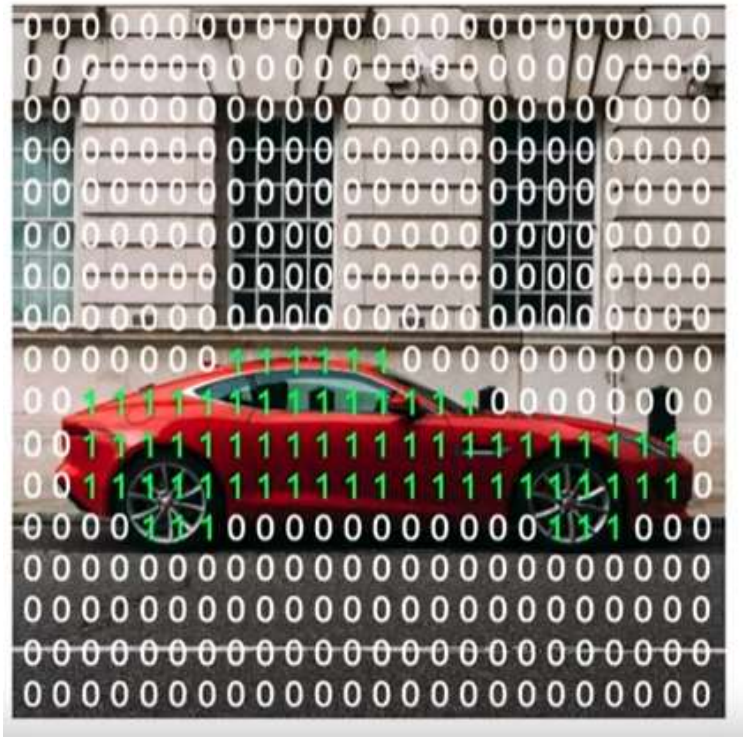


Segment the image into cell and non-cell



Segment images of coastlines into 2 classes — land and water

Per-pixel class label – example 1

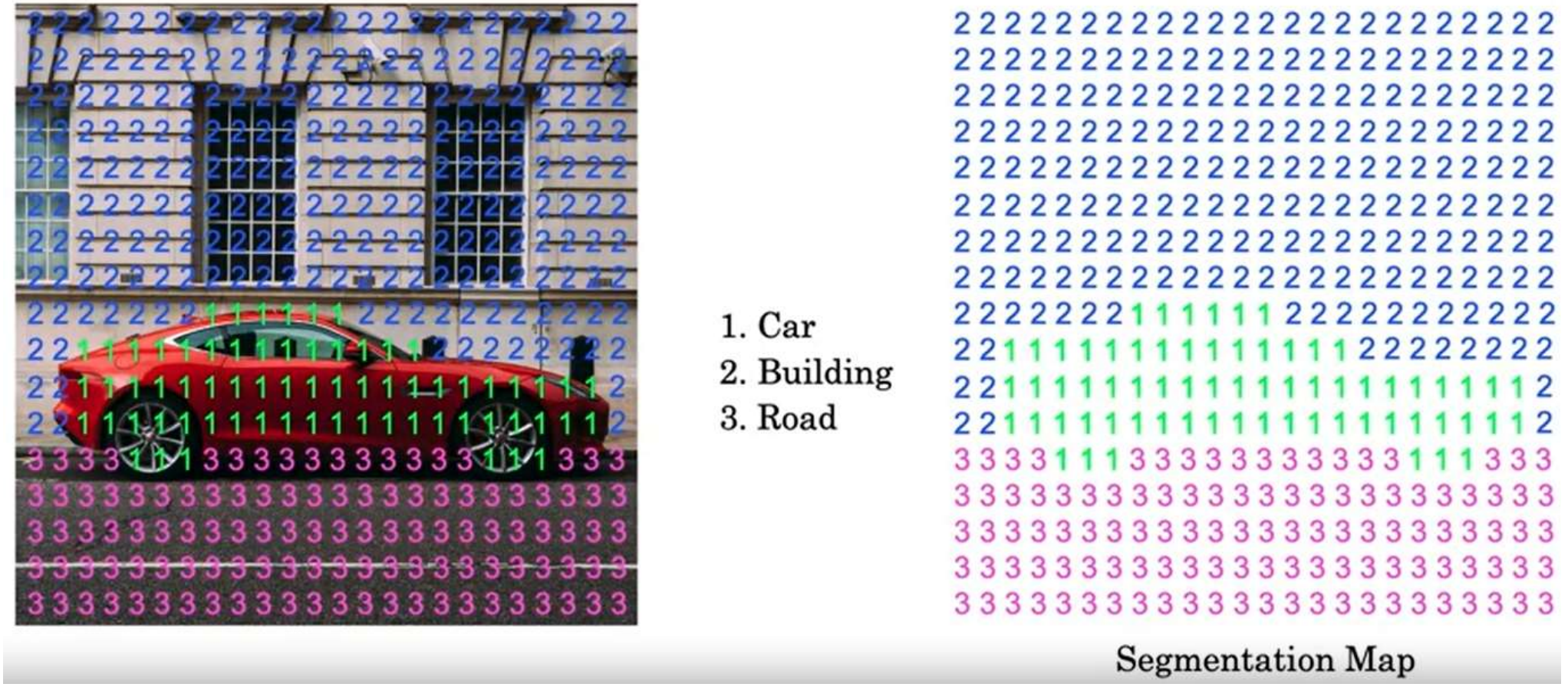


The task here is segmenting the car from the background.

Segmentation algorithm outputs, **1 or 0 for every pixel** in the image.

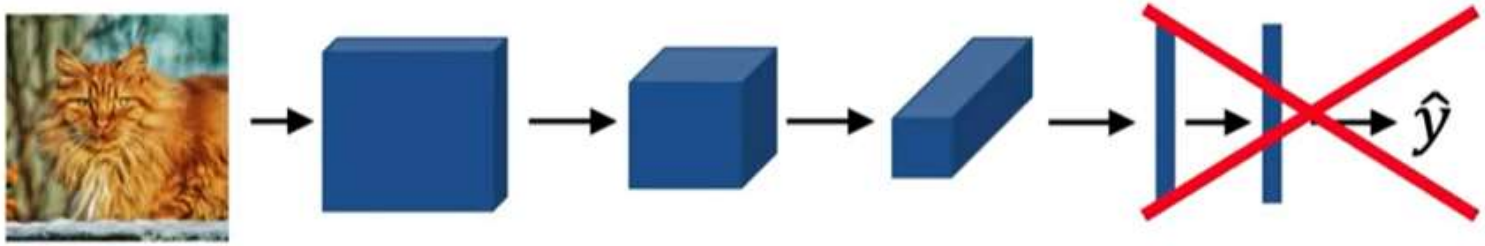
Pixel should be labeled 1, if it is part of the car; 0 if it's not part of the car.

Per-pixel class label- example 2

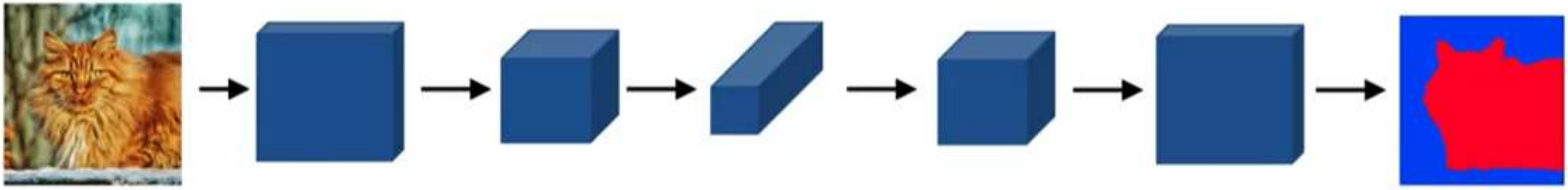


NN architecture suitable for such task is U-net

U-net architecture intuition



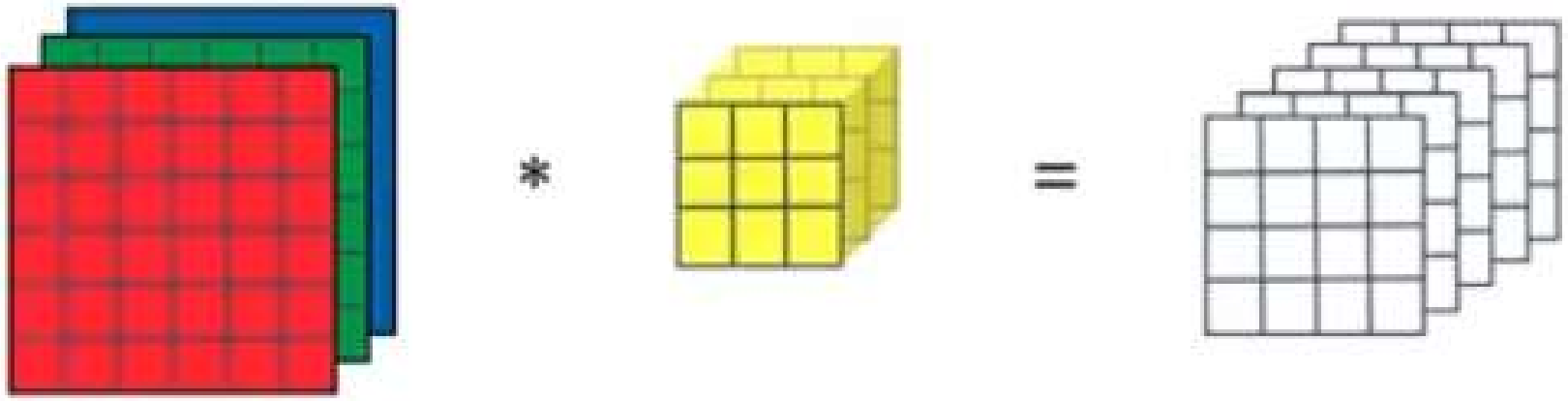
Standard CNN architecture: the input is an image, it goes through multiple layers in order to generate a class label.



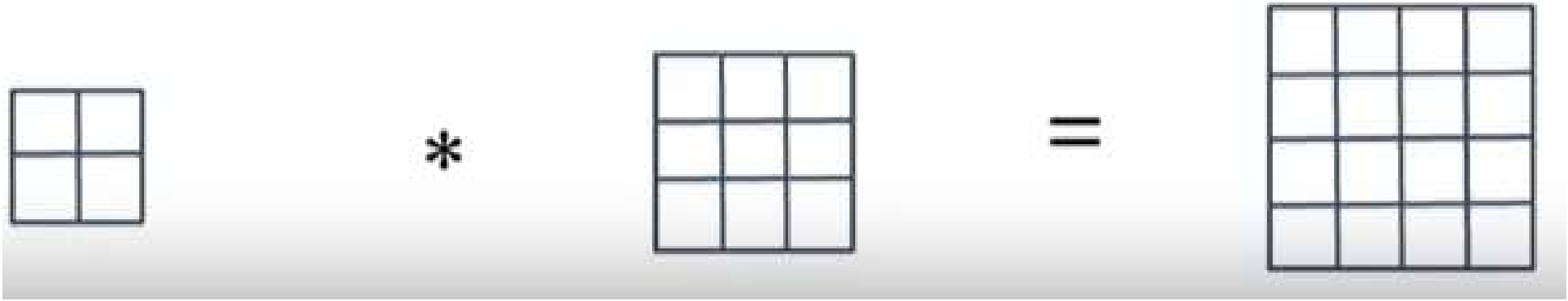
Semantic Segmentation CNN architecture: eliminate the last few (fully connected) layers.
Key semantic segmentation step: First the image size is getting smaller as it goes from left to right, then it gradually gets back to its full-size at the network output. We get the segmentation map of the input image.

Normal vs Transpose Convolution

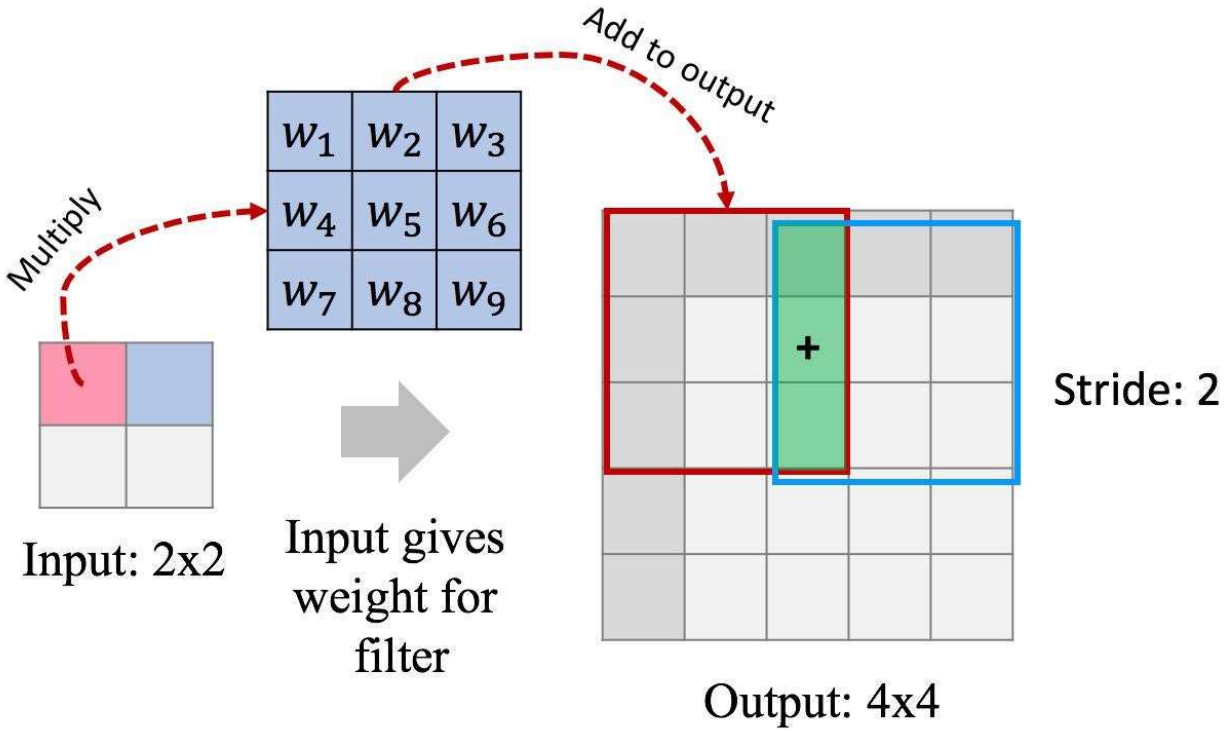
Normal Convolution



Transpose Convolution



Transpose Convolution - example

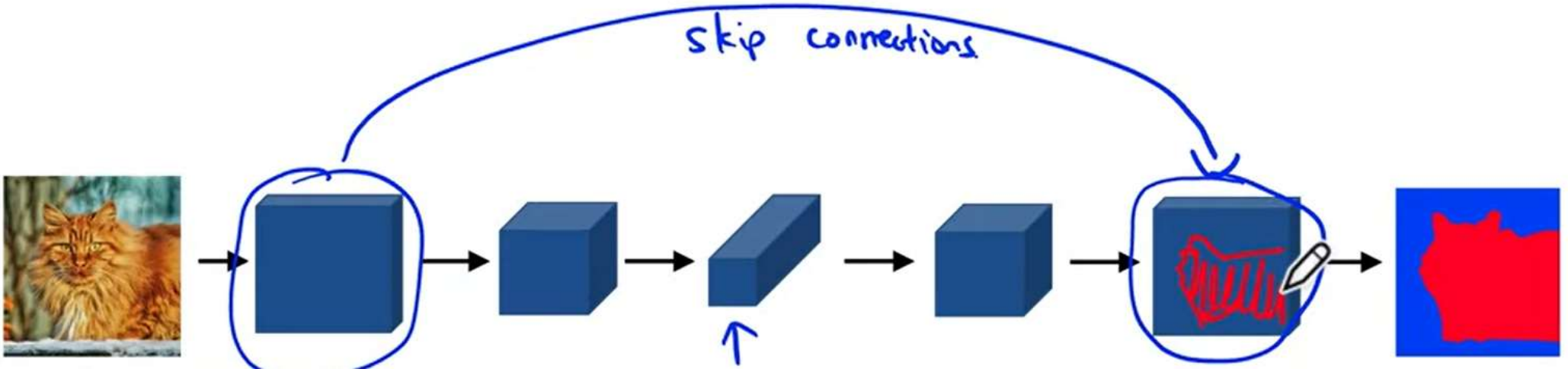


Ex. Take 2x2 input and blow it up into 4x4 output. If stride =2 => padding p=1.

There are many ways to take small inputs and turn into bigger outputs.
Transpose convolution (TC) is one of them.
TC is the key building block of U-net architecture.

Other names: deconvolution, upconvolution, backward strided convolution

U-net Architecture Intuition



Left half (Encoder) of U-net uses normal convolutions. The image will be compressed. From a large image go to volume with small height and width (detailed spatial information is lost) but it's much deeper.

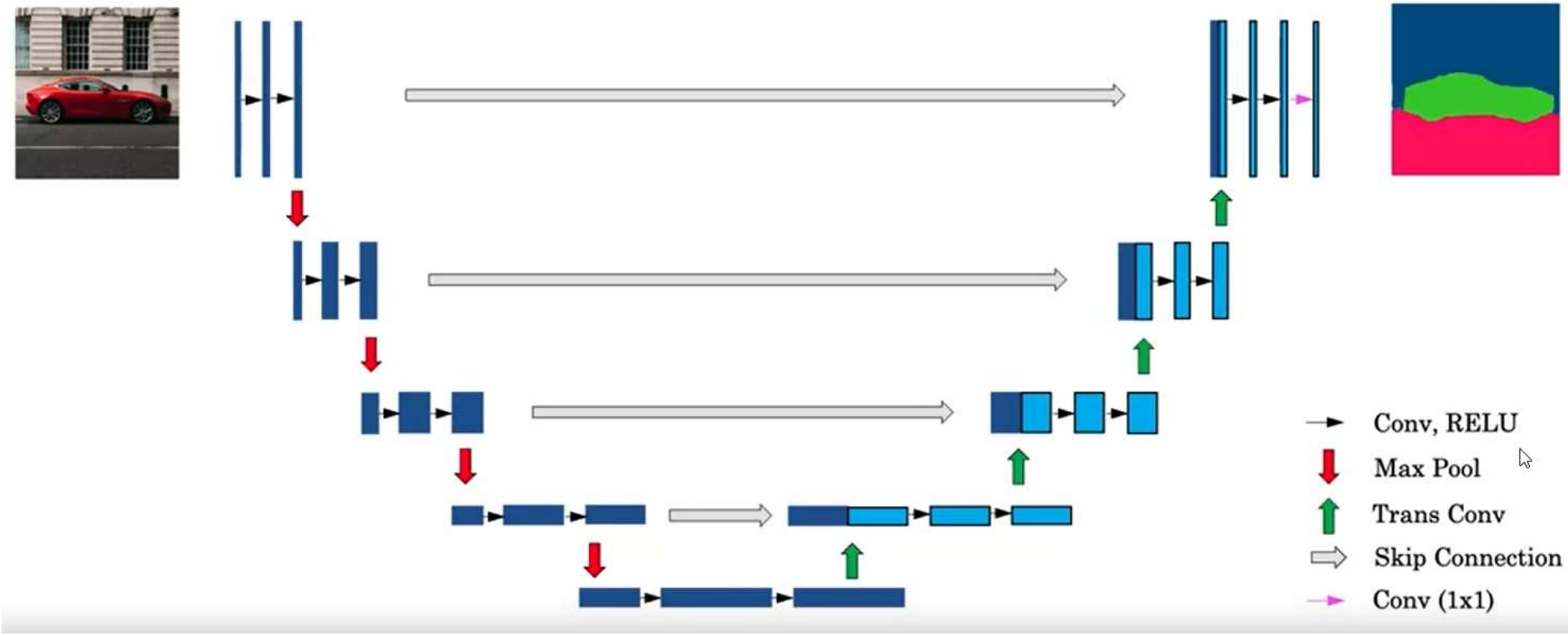
Right half (Decoder) uses the transpose convolution to blow the representation size up back to the size of the original input image.

With **skip connections** from earlier layers to later layers two types of info are got:

i) High level spatial /contextual information but with lower resolution from the previous layer;

ii) Low level feature information but with high resolution from the early layer.

U-Net



Encoder: sequence of normal feedforward (FF) conv layers (3D) + Relu act. functions & Max Pool to reduce height and width – each block has 3 conv layers.

Decoder: sequence of transpose convolution + skip connection+ normal conv layers. Increase the height and width, until it gets back to original height and width

Input image: height x width x 3

Output volume: height x width x Number of classes.