Neural networks: layers, activations, normalization

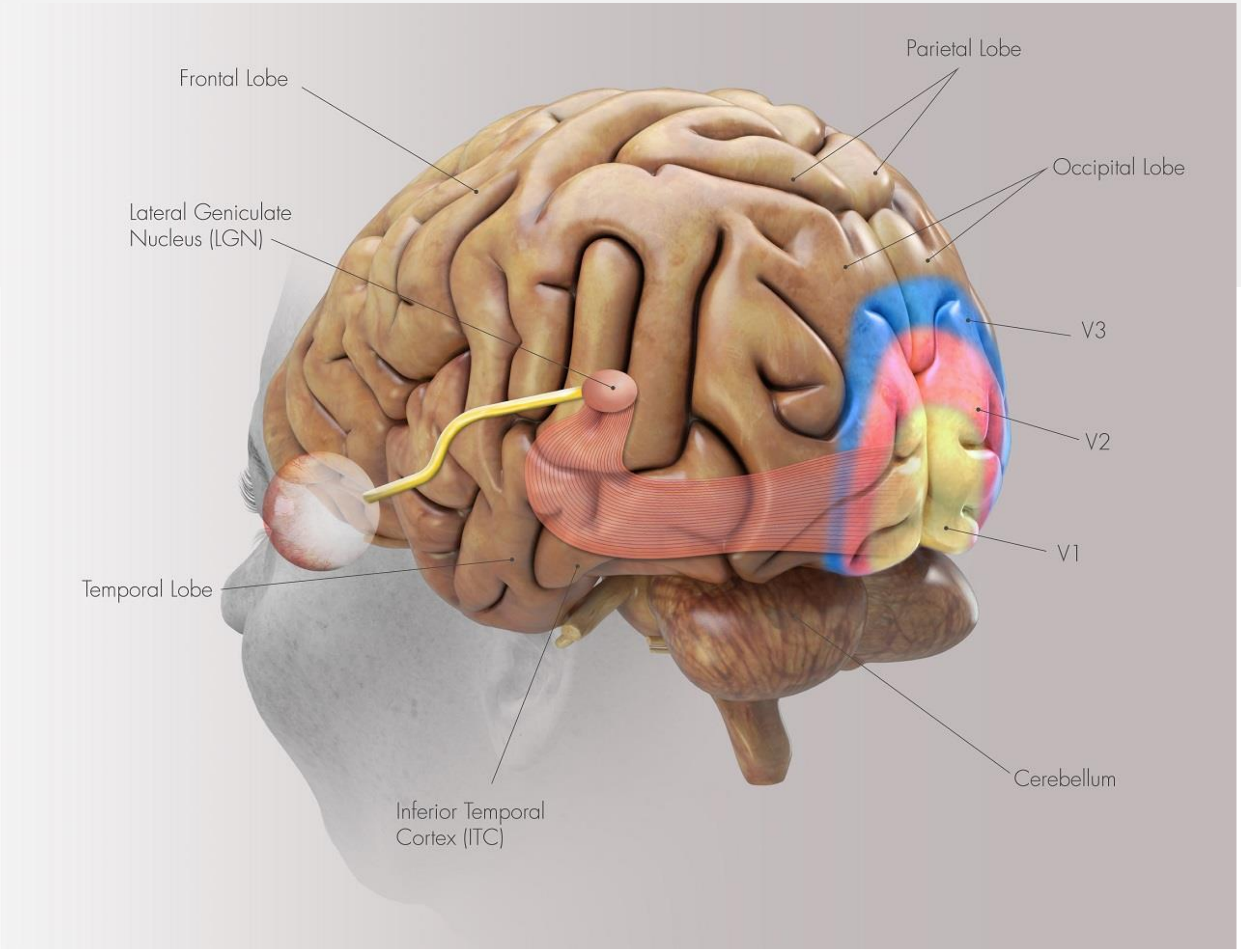
Prepared by Rytis Augustauskas

rytis.augustauskas@ktu.lt

Kaunas, 2024

What is neural network? [Vision inspiration]

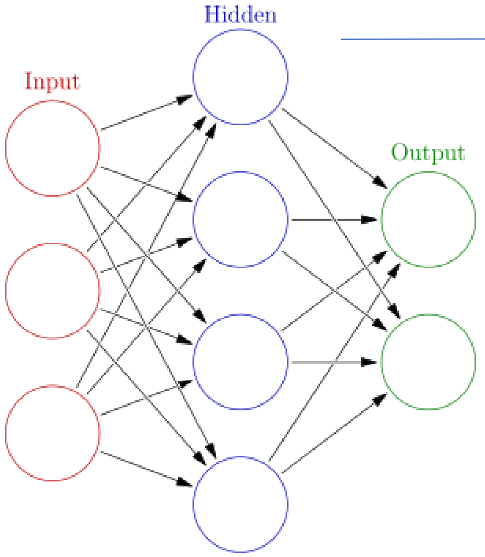
- **Artificial neural networks** (ANNs) or connectionist systems are computing systems inspired by the biological neural networks that constitute animal brains, mainly of visual cortex V1 that has 140 billion neurons with connection between them. Human vision involves not only V1, but V2, V3, V4 and V5 cortices doing more complex processing [1].



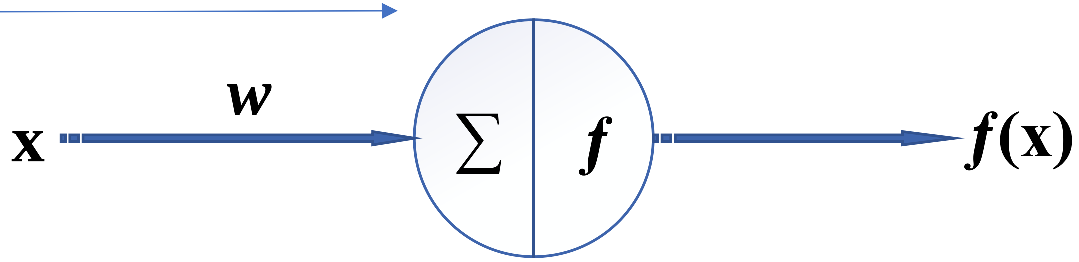
Visual cortices [2]

What is neural network? What is Neuron?

- Neural networks, a beautiful biologically-inspired programming paradigm which enables a computer to learn from observational data.
- A **neural network** is a network or circuit of neurons, or in a modern sense, an artificial neural network (ANN), composed of artificial neurons or nodes.



Basic structure of neural network with input, output and one hidden layer [3]



Single neuron structure: ***x*** – input, ***w*** – weight, ∑ - sum of all weights, ***f*** – activation function, ***f(x)*** – output

$$f\,(x) = \sigma(\sum w_n x_n + b),$$

where $\boldsymbol{\sigma}$ *– activation function,* $\boldsymbol{w_n}$*–weight,* $\boldsymbol{x_n}$ *– input value,* $\boldsymbol{b}$ *– bias*

How does a neural network learn from the data? (Supervised)

We have labelled data, which forms our dataset that we want to use to train the network.

Inputs: x1 and **x2**

Outputs: True Values (y) (ground truth)

1. **Forward propagation** is performed by feeding the input data (typically in small batches known as *minibatches*) through the network in a forward direction to generate predictions. Each layer applies weights, biases, and activation functions to the input data, producing outputs that flow to the next layer.
2. Error Calculation. The error between the predictions and the true values (ground truth) is calculated using a *loss function*. This function quantifies the difference between the model's predictions and the actual values, providing a measure of the model's performance.
3. Backpropagation. The network weights are adjusted using *backpropagation*, a process where the error is propagated backward through the network to update the weights and biases in a way that minimizes the error. The learning rate (or *training speed*) determines how much each weight is updated in response to the calculated gradients, controlling the pace of the learning process.
4. Iterative Process. This process is repeated for each minibatch in the training data over multiple *epochs*, meaning the entire dataset is passed through the network multiple times. With each iteration, the model's weights are adjusted further, progressively reducing the error and improving the model's accuracy.

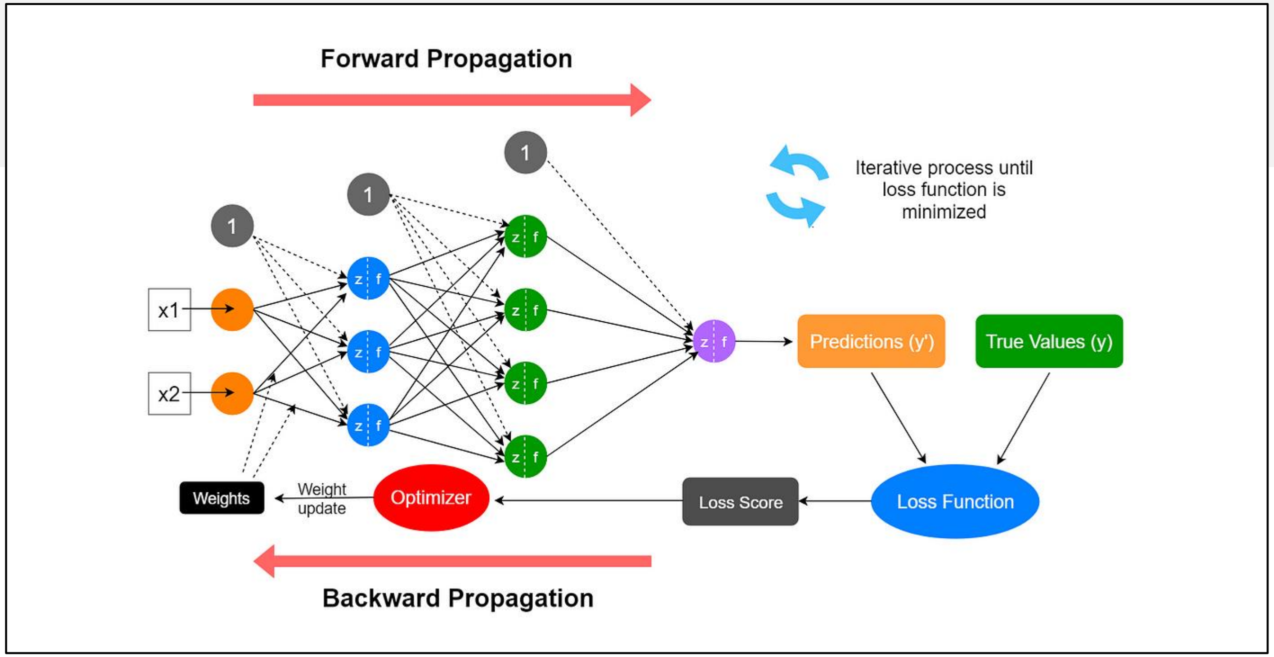


Illustration from: https://medium.com/data-science-365/overview-of-a-neural-networks-learning-process-61690a502fa

Important terminology in training neural network

Overfitting happens when a model learns the training data too well, capturing noise and small fluctuations rather than general patterns. As a result, the model performs excellently on the training data but poorly on new, unseen data (it fit too well to the seen [during training] data).

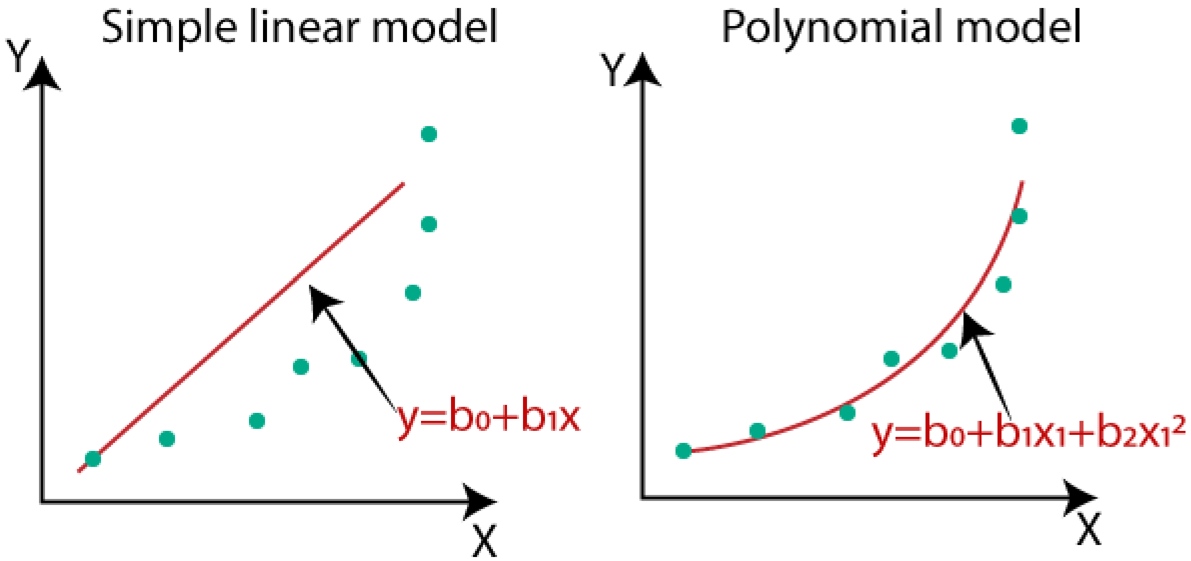
Overfitting example: A model trained with too many features or layers may memorize specific data points rather than generalizing patterns, which leads to high accuracy on the training set but low accuracy on the validation set.

Underfitting occurs when a model is too simple to capture the underlying patterns in the data, resulting in poor performance on both the training and test datasets.

Underfitting example: If a linear model is used to fit a highly complex dataset, it may fail to capture the data's nuances, leading to high error rates.

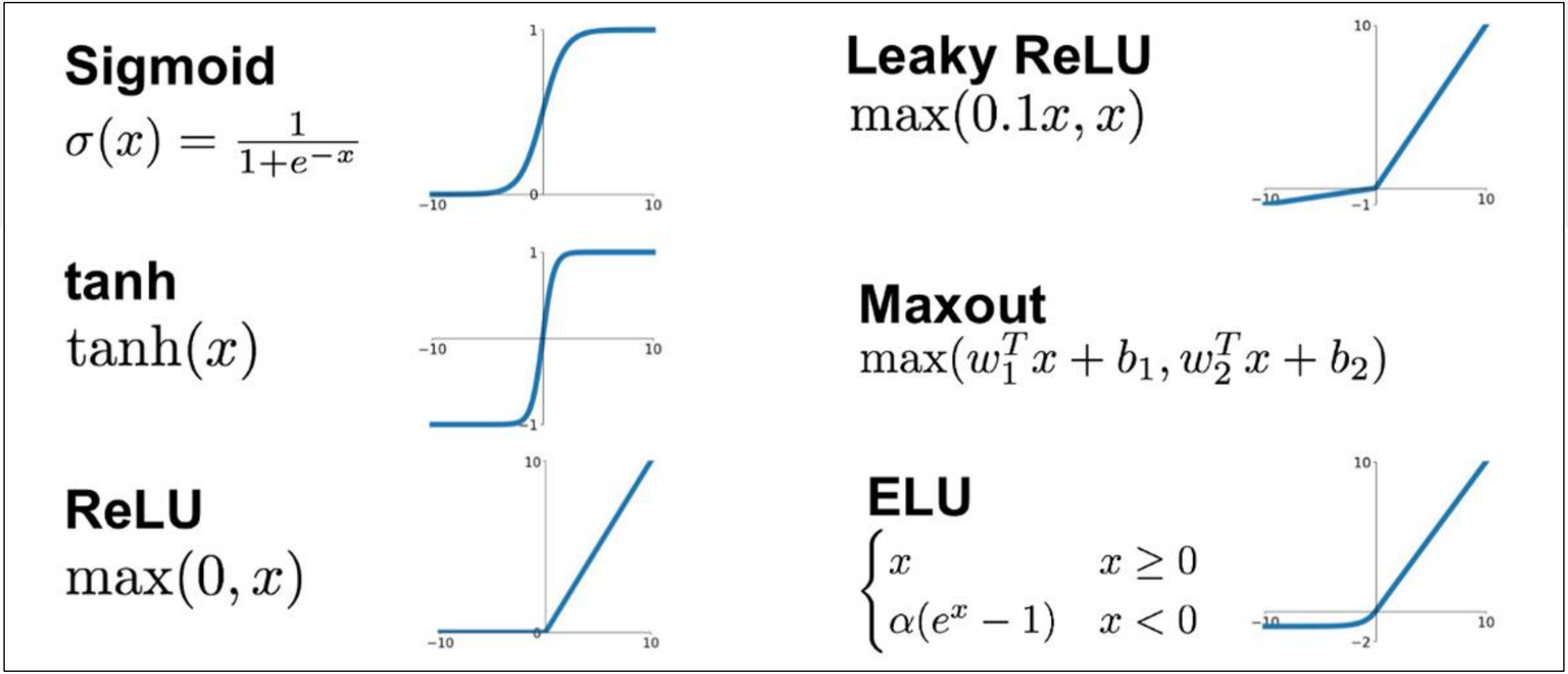
Why we need activation?

- To control the magnitude of output (example: ‘*sigmoid*’)
- Activation function might be considered as ‘decision-making’ part – ***how to activate neuron (give a signal) when it is present to stimulus***?
- Activation function usually brings non-linearity (exception -‘*linear*’ activation) to models that is important for the complex and versatile data. [**IMPORTANT FEATURE**]



Linear and non-linear (in this case polynomial) functions [4]

Activation functions



Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$

tanh

$\tanh(x)$

ReLU

$\max(0, x)$

Leaky ReLU

$\max(0.1x, x)$

Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

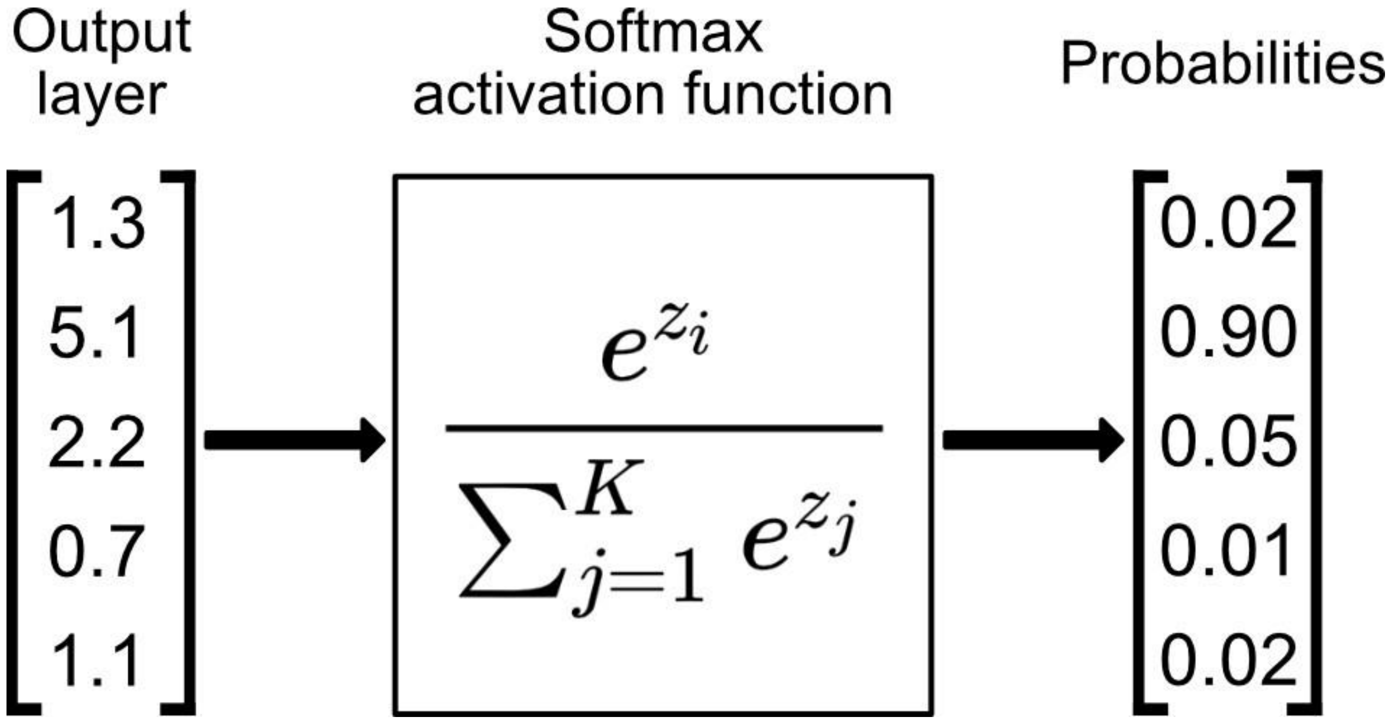
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

Common activation functions [5]

Activation functions. Softmax (multiclass)

$$\sigma(\overline{\mathbb{Z}})_i = \frac{\mathrm{e}^{z_i}}{\sum_{j=1}^{k} \mathrm{e}^{z_j}}$$

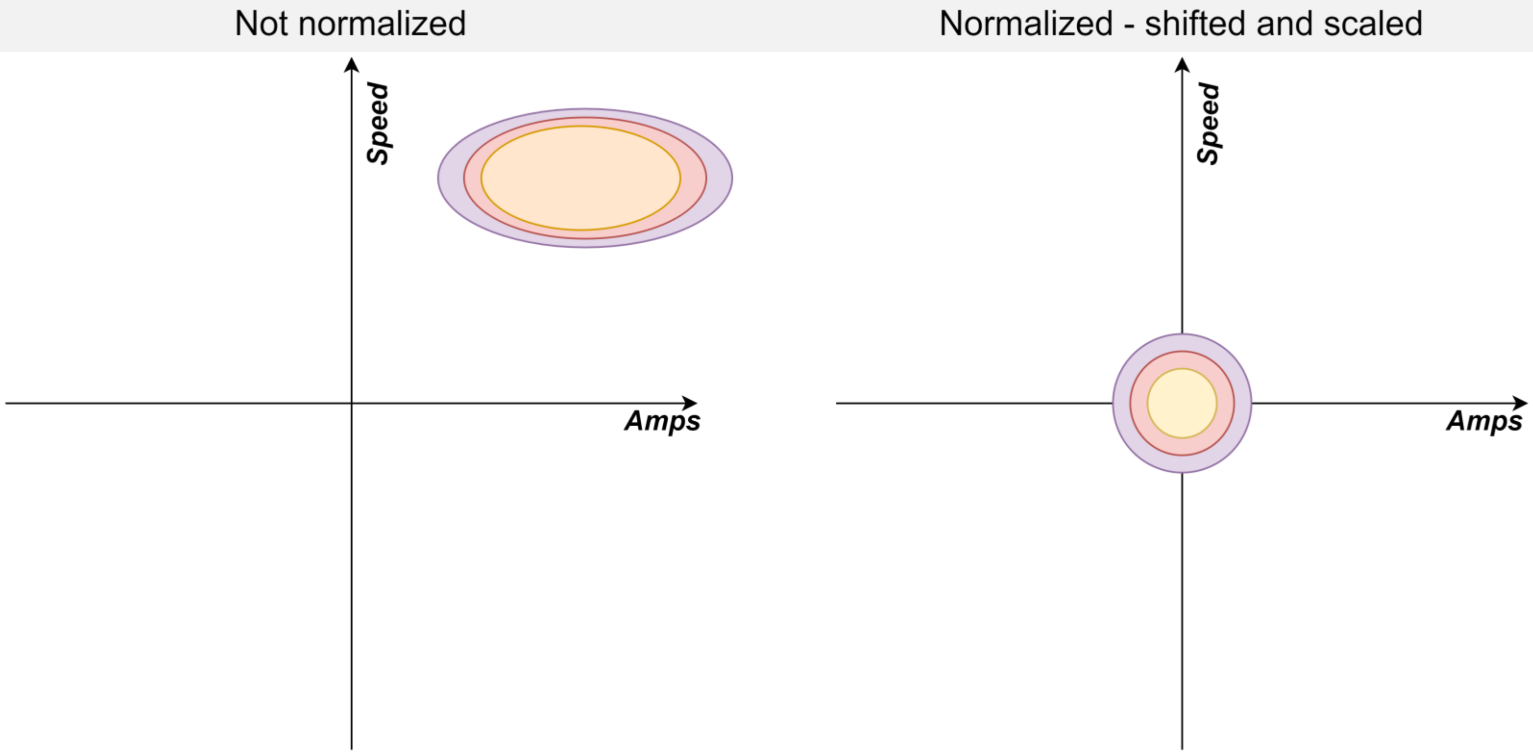
σ – *softmax*
$\overline{\mathbb{Z}}$ - input vector
k – number of classes
e^{z_i} - standard exponential function for input vector
e^{z_j} - standard exponential function for output vector


Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

Output layer activation with softmax [6]

Batch normalization

- Batch normalization rescales and remaps data
- Gives regularization effect to model, by introducing small noise from normalization, that depends on mini-batch
- Allows training model with higher learning rate



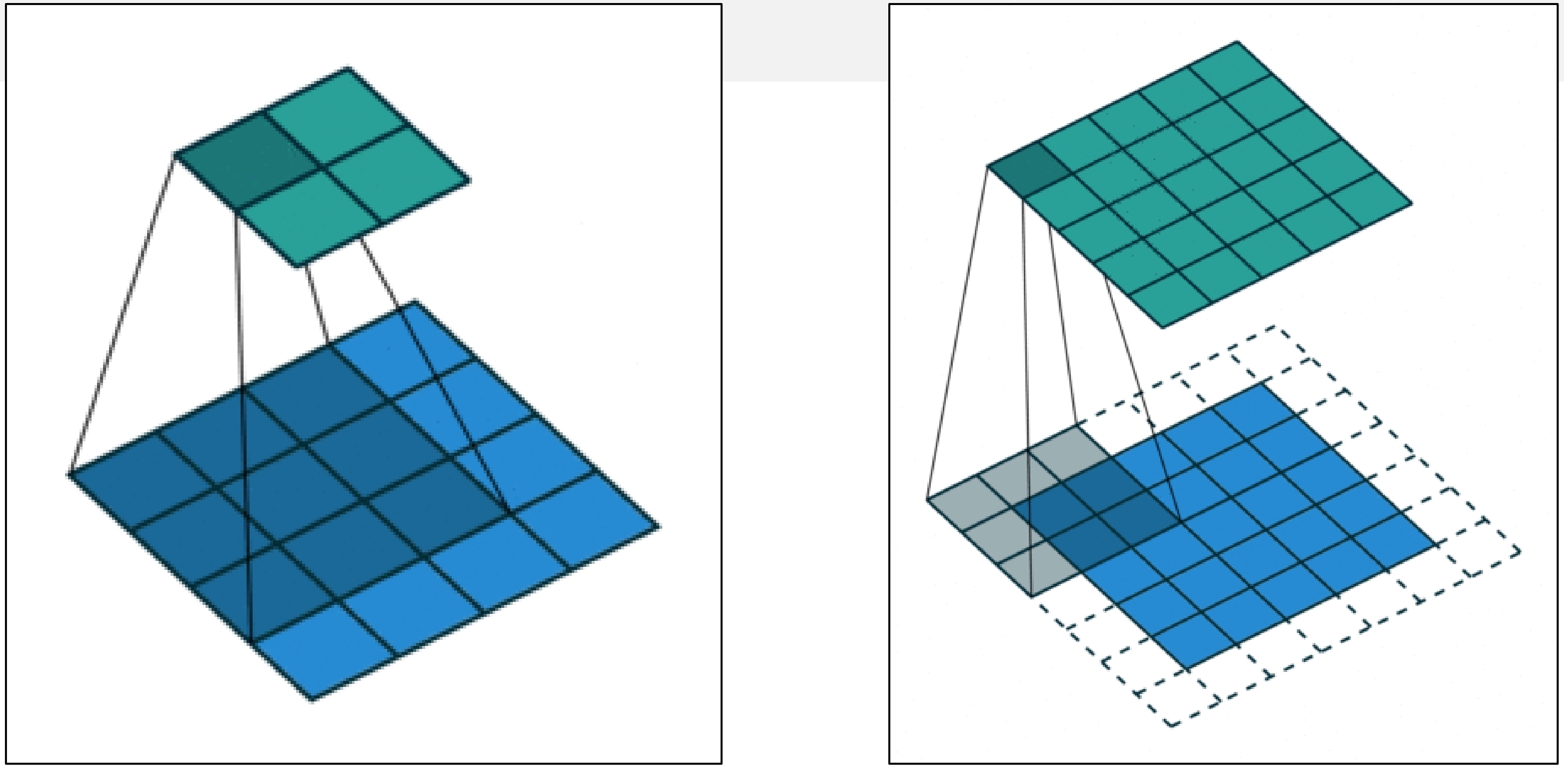
Not normalized

Normalized - shifted and scaled

Batch normalization example with motor data. Data on the left is scaled and shifted

Convolutional layer

- Higher-level neural network layers that are capable to capture features, such as lines, color changes, and so on
- Different from densely connected layers, convolutional layers have share weights and biases
- Convolutional kernel is ‚sliding‘ (convolving) through the image with defined stride

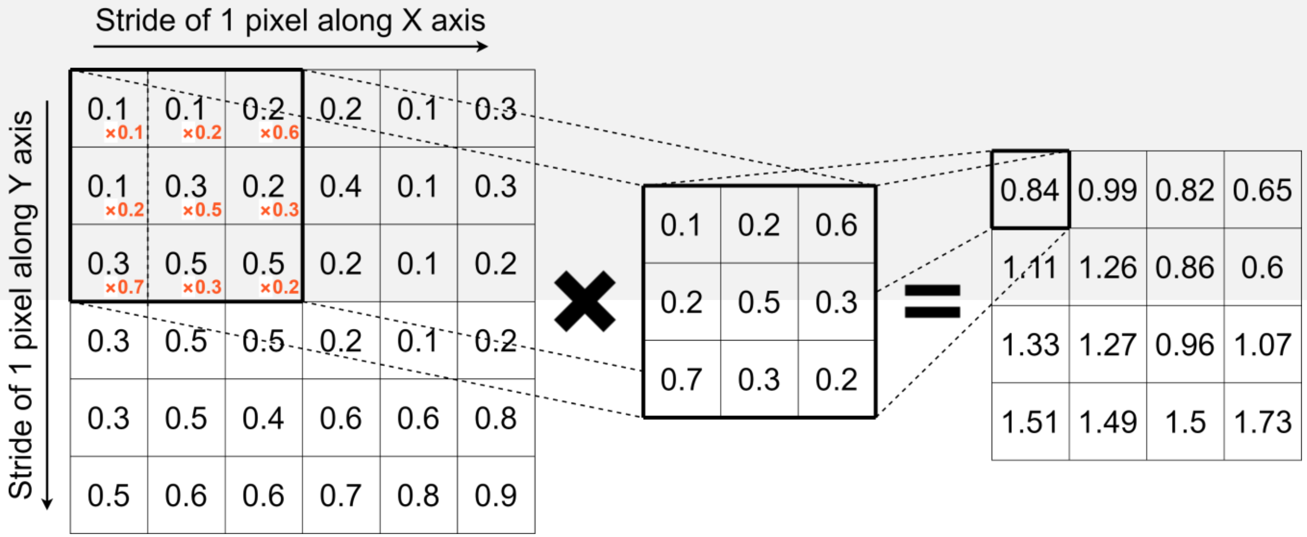


a)

b)

Convolutional operations: blue - input, green - output. **a** – no padding (smaller output), **b** – with padding [7]

How convolutional layers work?



Stride of 1 pixel along X axis
Stride of 1 pixel along Y axis

0.1 ×0.1	0.1 ×0.2	0.2 ×0.6	0.2	0.1	0.3
0.1 ×0.2	0.3 ×0.5	0.2 ×0.3	0.4	0.1	0.3
0.3 ×0.7	0.5 ×0.3	0.5 ×0.2	0.2	0.1	0.2
0.3	0.5	0.5	0.2	0.1	0.2
0.3	0.5	0.4	0.6	0.6	0.8
0.5	0.6	0.6	0.7	0.8	0.9

×

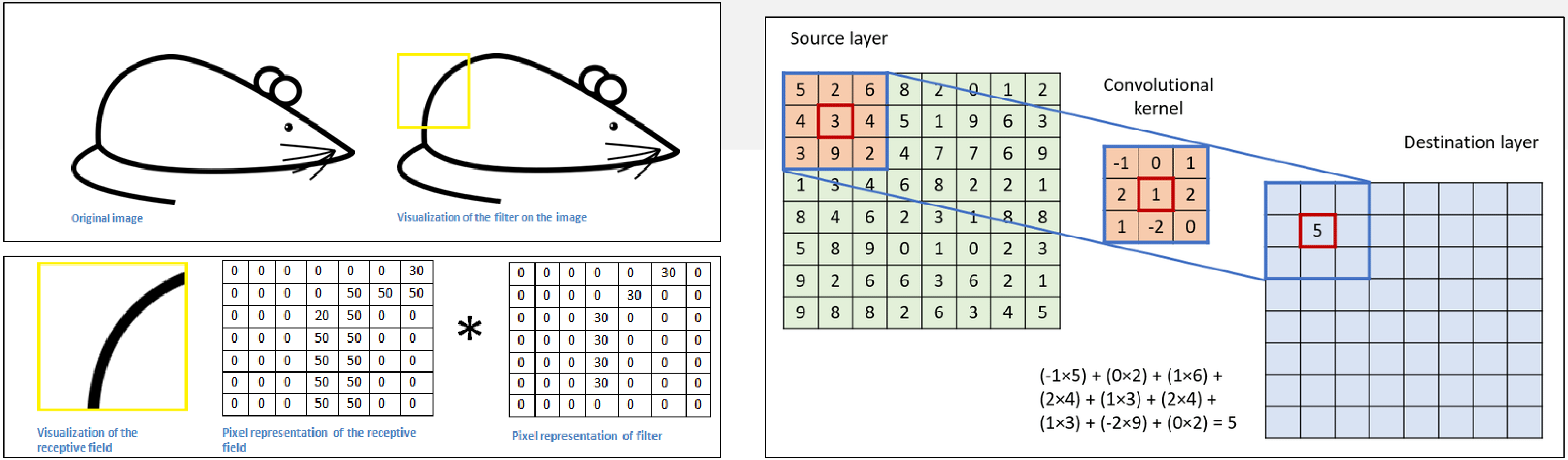
0.1	0.2	0.6
0.2	0.5	0.3
0.7	0.3	0.2

=

0.84	0.99	0.82	0.65
1.11	1.26	0.86	0.6
1.33	1.27	0.96	1.07
1.51	1.49	1.5	1.73

Mathematical convolution principle; 6x6 2D data map multiplied by 3x3 feature kernel; this filter is 'slid' through the image with a stride of 1 pixel along the x and y axes; the output is 4x4 data map, a result of the convolutional operation

How convolutional layers work?


Original image
Visualization of the filter on the image
Visualization of the receptive field
Pixel representation of the receptive field
*
Pixel representation of filter
Source layer
Convolutional kernel
Destination layer

Pixel representation of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of filter

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

Source layer

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

Convolutional kernel

-1	0	1
2	1	2
1	-2	0

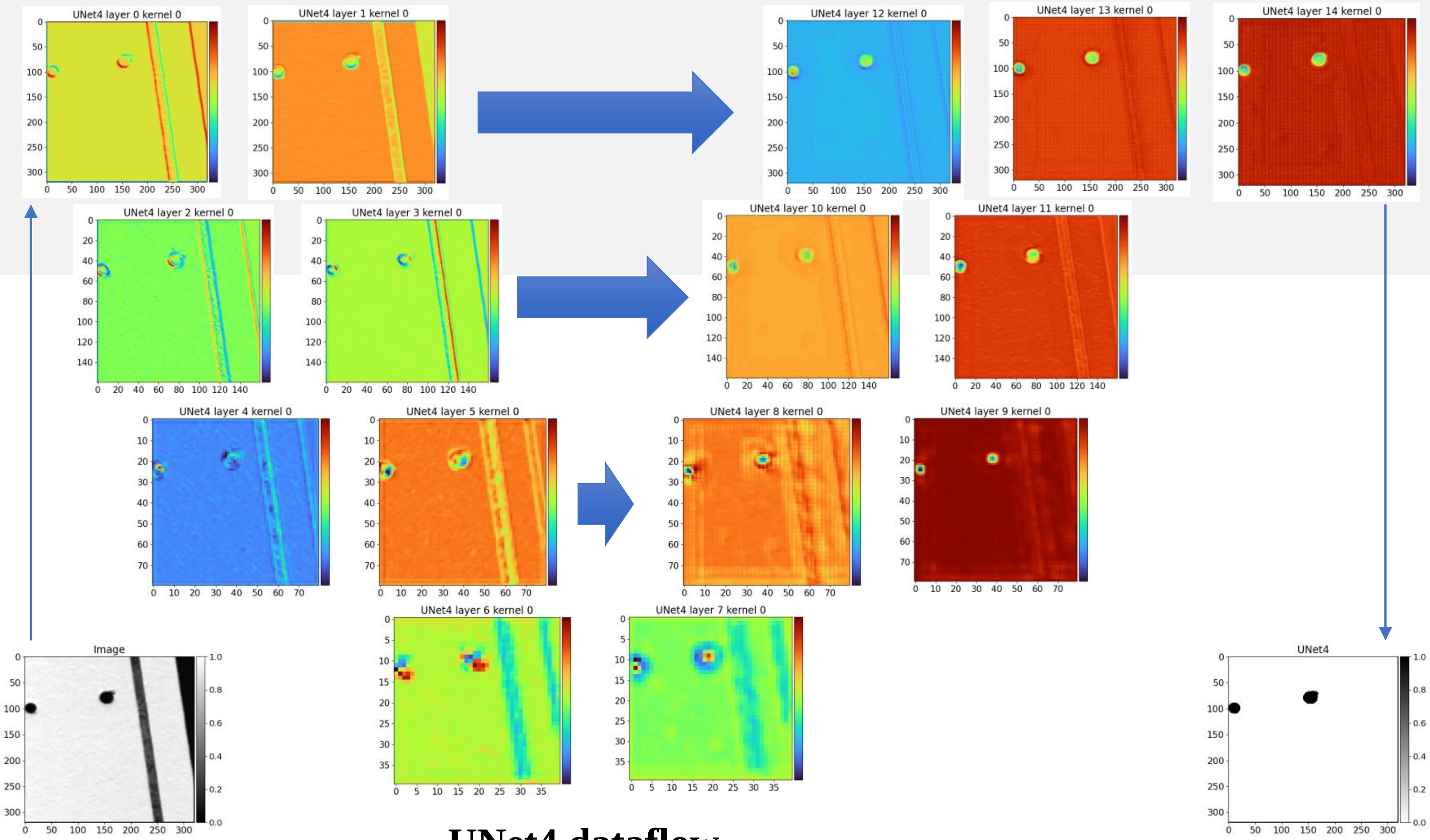
Destination layer

	5						

(-1×5) + (0×2) + (1×6) +
(2×4) + (1×3) + (2×4) +
(1×3) + (-2×9) + (0×2) = 5

a - Mouse contour extraction [8], b – convolutional multilplication example [9]

How convolutional layers work?



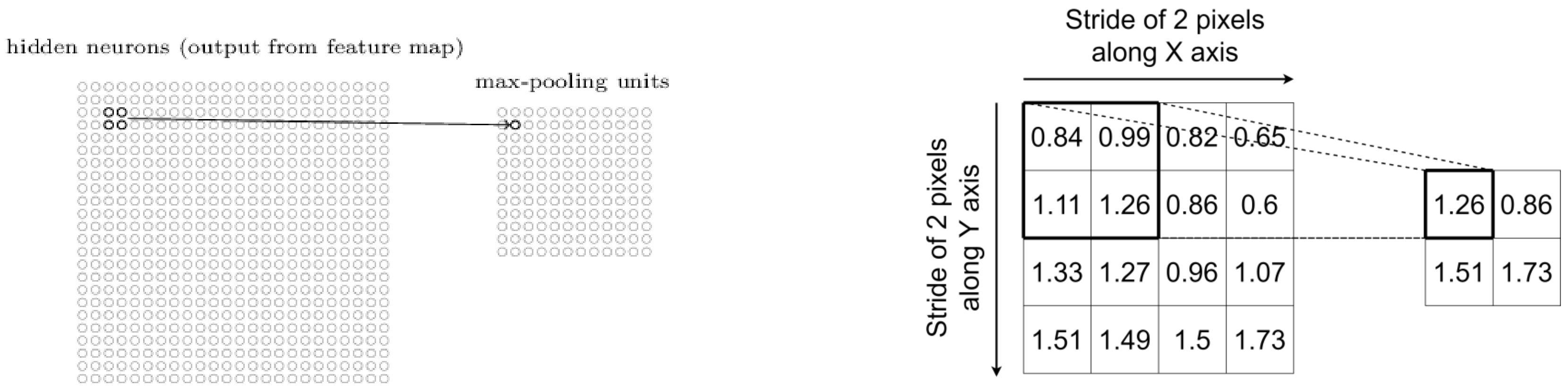
Input

UNet4 dataflow

Output

Pooling

- A different part from the classic neural network that CNN has is the pooling layer. This layer down-samples and simplifies output data from the convolutional layer (or any previous layer). **It summarizes the features of the specified region**
- Most popular approaches are *Max* and *Average* pooling
- *Global* pooling takes whole feature map into consideration


hidden neurons (output from feature map)
max-pooling units
Stride of 2 pixels along X axis
Stride of 2 pixels along Y axis

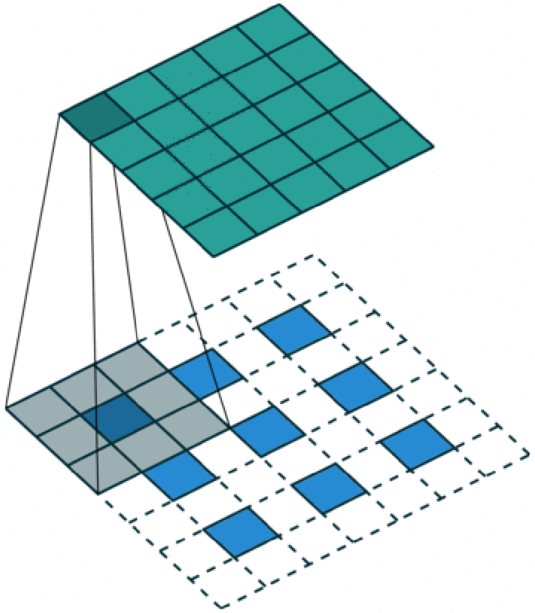
0.84	0.99	0.82	0.65
1.11	1.26	0.86	0.6
1.33	1.27	0.96	1.07
1.51	1.49	1.5	1.73

1.26	0.86
1.51	1.73

2x2 region pooling with stride of 2 [8]

Transposed convolutional layer

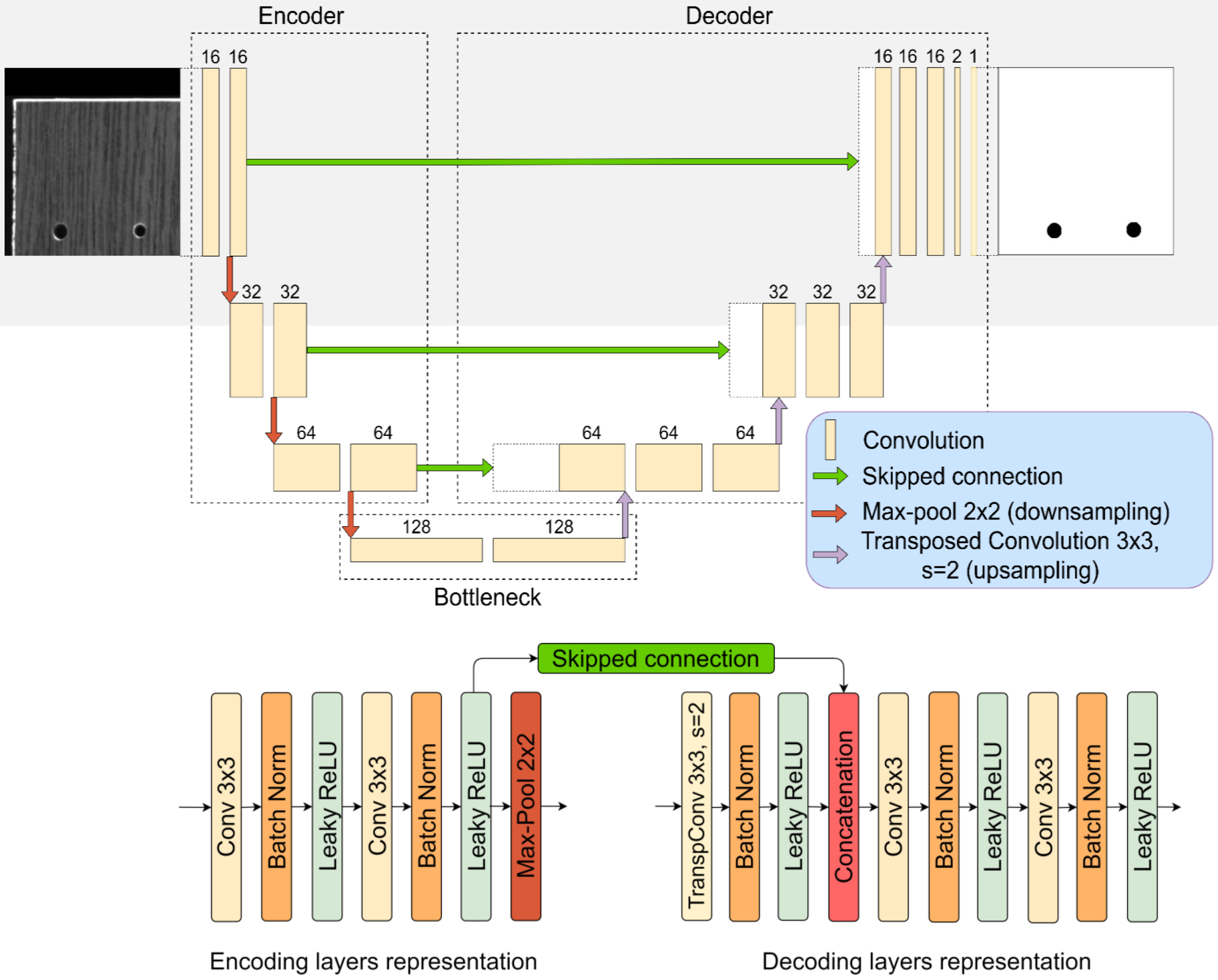
- Operation that learns how to upscale
- Can be used in super-resolution
- Used a lot in autoencoders (decoding part)
- Might be expressed as *Upscale + Conv*



Transposed convolutional operations 3x3, stride=2 : blue - input, green - output [7]

How to combine things?

- In general case, activation is needed after convolutional (or dense) layer.
- If we are using batch normalization, it should be inserted between convolution and activation



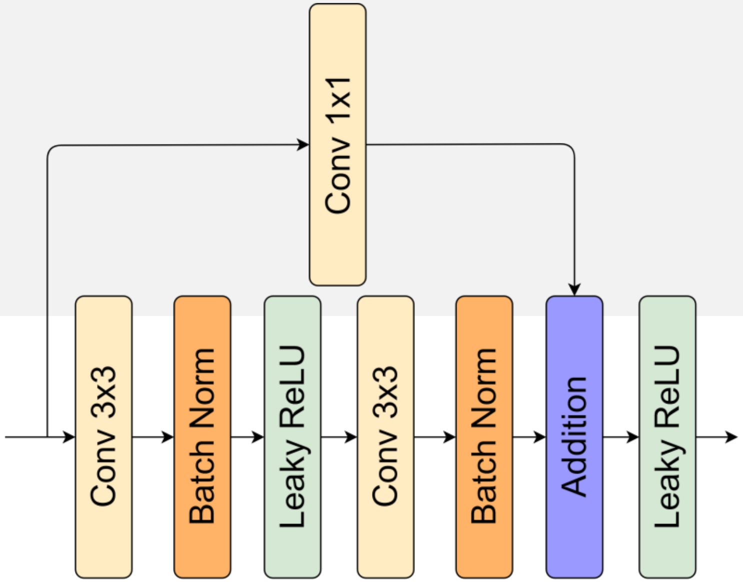
Encoding layers representation

Decoding layers representation

Layers representation in UNet

More advanced modules. Residual layers

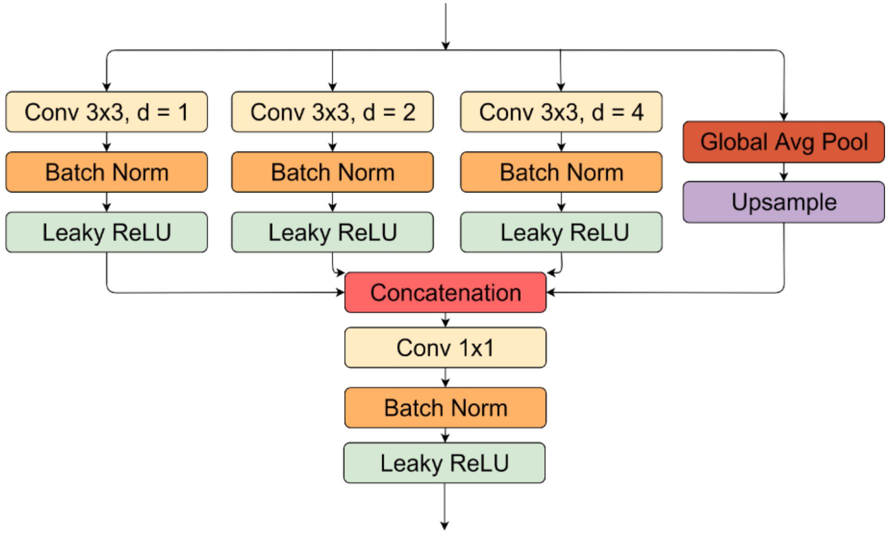
- Residual layers are proposed in ResNet.
- The branch connected in parallel skips convolutional operation. Residual connections help to maintain information flow through the whole network, without a possible degradation in series of operations conducted in a neural network.
- Moreover, this block increases model accuracy and might cope with the vanishing gradient problem.
- Residual layers are used in popular architectures, such as *SqueezeNext*, *DeepLab*, and *Inception*.
- We utilize 1x1 convolution to make the number of feature maps the same before the addition operation.



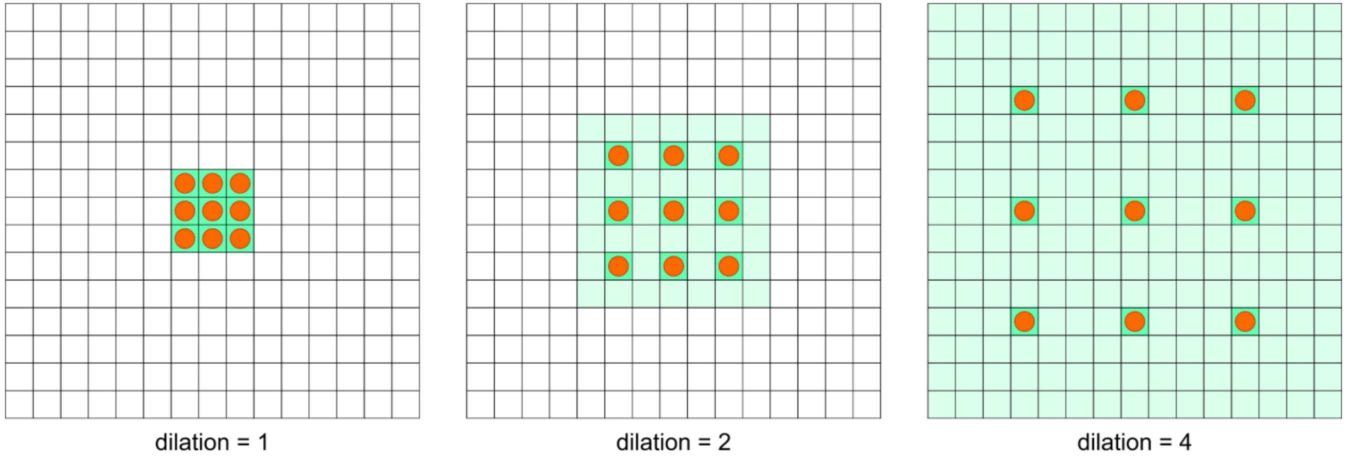
Residual connection with 1x1 convolution to equalize the number of feature maps

More advanced modules. Atrous spatial pyramid pooling

- Convolutional operations with different dilation rates might extract multi-scale contextual information better than regular convolutions (with a dilated rate equal to 1)
- Atrous or dilated convolutions in the parallel idea was proposed by Chen et al. [10].
- An expanded convolutional kernel can better respond to different resolution features.



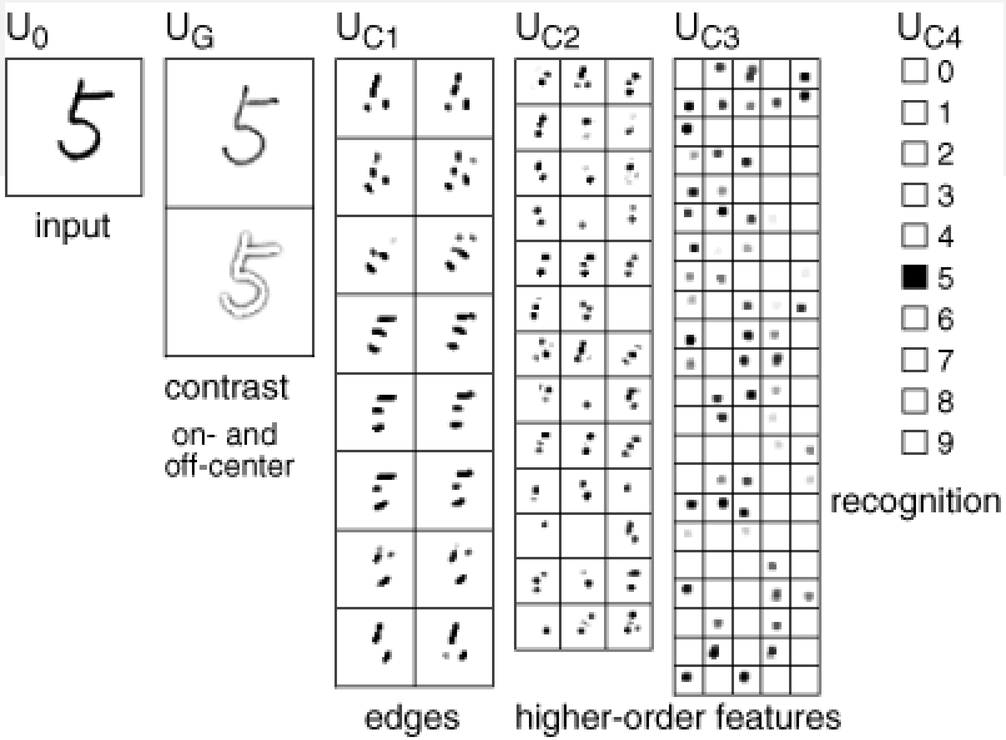
Atrous spatial pyramid pooling module



Convolution with various dilation rates

Why convolutional neural networks were not used before?

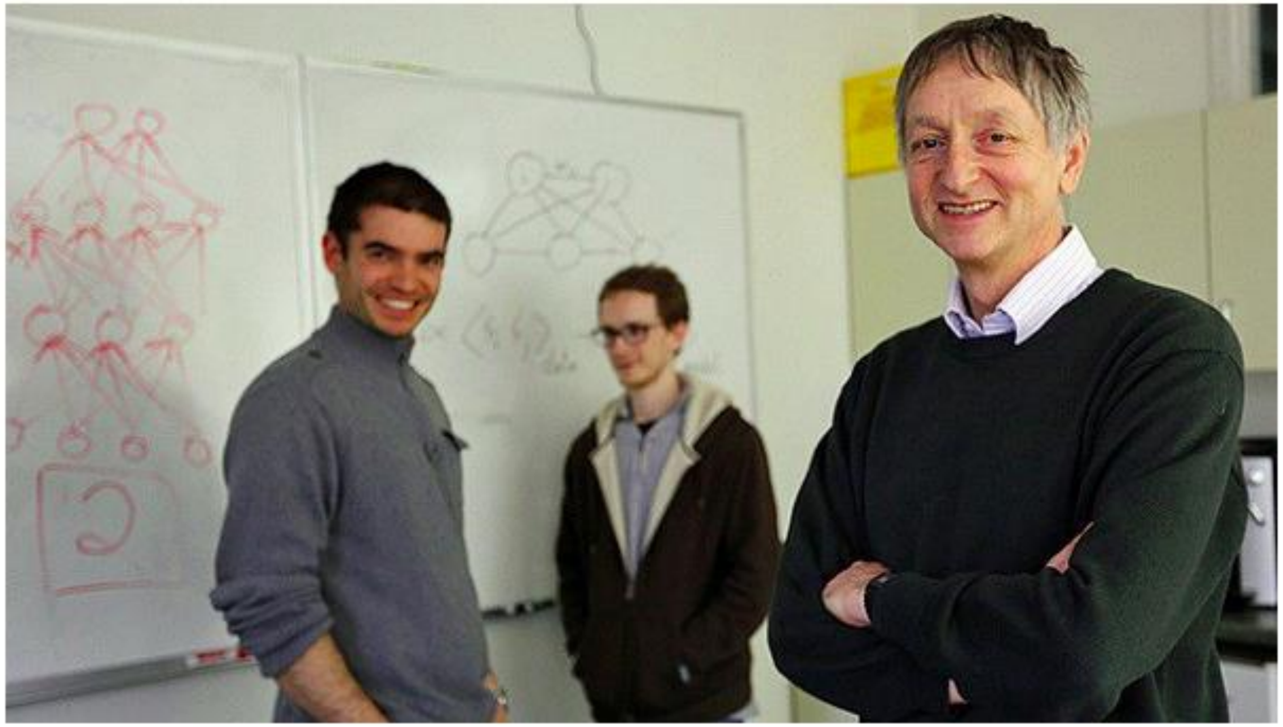
- ***Neurocognitron [11] – 1980***



LeNet [12] – 1993

CNN breakthrough

In 2012, in ImageNet Large Scale Visual Recognition Challenge (ILSVRC), convolutional neural network (CNN) approach by Alex Krizhevsky, Ilya Surskever, and Geoffrey E. Hinton was introduced to detect and classify the very big amount of data with a lot of classes from ImageNet database. The algorithm used a deep neural network with convolutional layers. This changes previous approaches and gave an idea to rethink the existing way of object detection and classification.



Learning Material

1. Extremely well-made short video series from 3Blue1Brown on Youtube about neural networks, gradient descent, and backpropagation *(Chapter 1-4)*
2. Extremely well-prepared book on Deep Learning: **Title, The Little Book of Deep Learning. Author, François. Publisher, Writers Republic LLC, 2023. ISBN, 9732346493, 9789732346495.** *Available (free) at https://fleuret.org/public/lbdl.pdf (Chapter 1, Chapter 3.1, 3.3, 3.4, 3.5, 3.6)*

Thank you for your attention! Questions?

email: rytis.augustauskas@ktu.lt