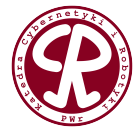
# Selected topics in Artificial Intelligence
## Convolutional Neural Networks

Wojciech Domski

Chair of Cybernetics and Robotics,
Wrocław University of Science and Technology

**Presentation compiled for taking notes during lecture**

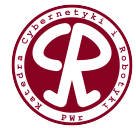
Wrocław University
of Science and Technology



# Outline



Wrocław University
of Science and Technology



# Cat experiment

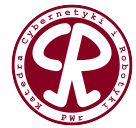
## The Visual cortex

In late 50s of XX$^{th}$ century a series of experiments were conducted including cats. The most informative were observations of the visual cortex where it was noticed that some of the neurons react to only limited range of nearby neurons.

## Dense layer and CNN

In comparison to dense layer neurons in convolutional layer are not connected to each other. In fact, they are connected only to nearby neurons inspired by the biological phenomenon.

Wrocław University of Science and Technology



# Applications

Convolutional Neural Networks are used mainly with 2D data, thus it can be used for image processing.
CNNs can be used for:

- OCR,
- image recognition,
- object detection,
- face recognition,
- image classification,
- image analysis e.g. in medicine.

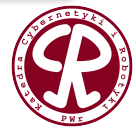
# Input of CNN

CNN layers work with 2D input signals, thus images. In case of multiple convolutional layers each pixel is only connected with the surrounding pixels of previous layer.
This stacking of layers allows to resemble chain of more and more complex feature detectors.

## Car detector

Let's assume that we implement a CNN for detecting cars in the image. The first layer would focus on detecting primitive shapes like lines or circles. The second layer would focus on detecting an outline of a car provided the input form the first layer while the last layer would detect cars themselves.

Wrocław University
of Science and Technology

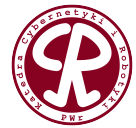


# Building blocks

Each convolutional layer consists of three intermediate layers (given in order from input to output):

1. Convolution
2. Activation
3. Pooling

Wrocław University of Science and Technology



# Convolution in CNN (1/2)

Let's consider two layers:

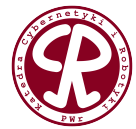
1. Input layer.
2. Convolution layer.

The data from input layer are passed to the next layer through convolution operation expressed as

$$S_{i,j} = (K * I)_{i,j} = \sum_m \sum_n I_{i-m,j-n} K_{m,n} \tag{1}$$

where $K$ is the kernel. Through training process when learning algorithm is applied the values of the kernel matrix are adjusted.

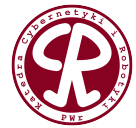
Wrocław University of Science and Technology



# Convolution in CNN (2/2)

During training of CNN convolution layer is used to create multiple filters at given stage. Thus, multiple convolution layers can be present where each one is feed with the same input, e.g. image.

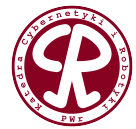
Wrocław University of Science and Technology



# Padding

When convolution operation is applied to an image the convoluted output image has a different (smaller) shape than the original image.
To prevent change of image (layer) size operation called padding has to be considered. The padding is done by artificially enlarging the surrounding input layer with new elements (a border).
In order not to impact the computation result the newly added border is filled with zero values. This technique is called zero padding.

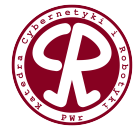
Wrocław University
of Science and Technology



# Stride (1/3)

Striding is another technique which impacts the size of the resulting layer. Also, it increases the performance, thus computation takes less time.
The striding step usually is equal to one. It describes the gap between elements from input data. It is equal to stride size minus one. Thus, for stride equal to one the extracted matrix has elements adjacent to their neighbours.

Wrocław University
of Science and Technology



# Stride (2/3)

Stride = 1

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 |

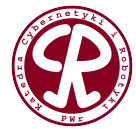
Wrocław University
of Science and Technology

# Stride (3/3)

Stride = 2

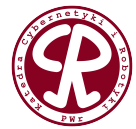
| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 |

Wrocław University
of Science and Technology



# Activation layers

Wrocław University
of Science and Technology



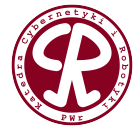
# Activation (1/10)

The purpose of activation layer is to introduce non linearity to multi-layer model.
A set of multiple layers where a single layer can be expressed as linear expression

$$Y = \sum wX + b \tag{2}$$

is linear.
Introducing non-linear function (activation function) allows us to introduce non-linearity to the model.

Wrocław University of Science and Technology



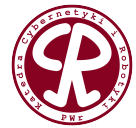
# Activation (2/10)

The activation function should be:

non-linear allows to introduce non-linearity to the model, thus the model can learn more complex relations between features and labels.

differentiable is result of how NN learns. The weight coefficients in each layer are adjusted in the process of back-propagation. When gradient based method is used to determine in which direction adjust the weights it relays on **gradients**, thus the activation function should be well defined.

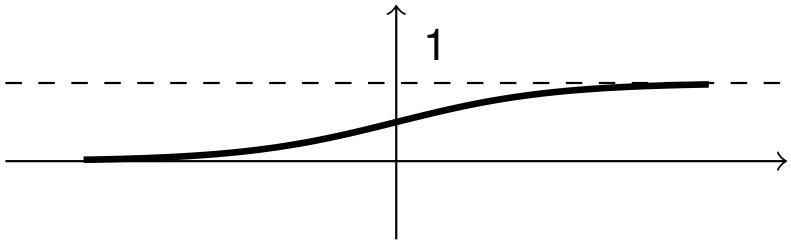
Wrocław University of Science and Technology



# Activation (3/10)

Sigmoid function (logistic function)

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



# Activation (4/10)

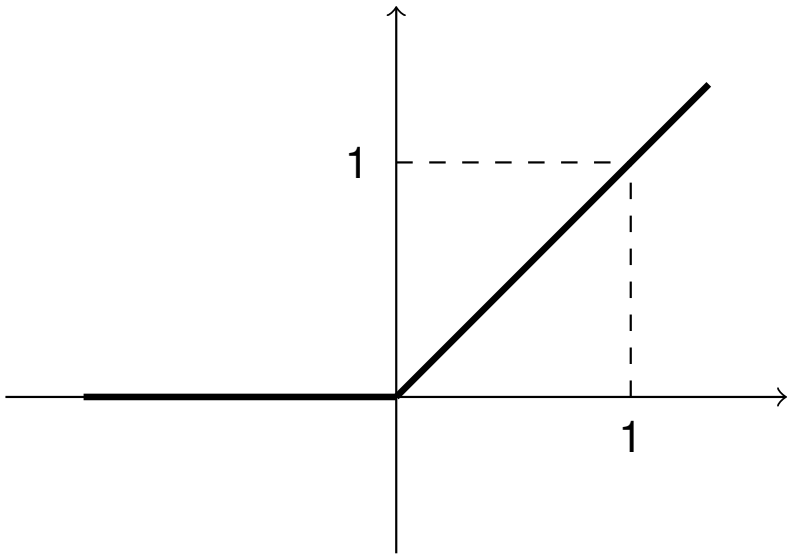
Sigmoid function properties:

- it is non-linear and differentiable,
- can be used for probability representation in binary classification due to limited output range between 0 and 1,
- computation resources demanding – learning process is longer,
- it is not advised to be used in hidden layers due to vanishing gradient problem – inefficiency when input values are too small or too large.

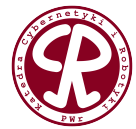
# Activation (5/10)

ReLU function

$$f(x) = \begin{cases} x, & \text{when } x \geq 0, \\ 0, & \text{when } x < 0. \end{cases} \tag{3}$$




Wrocław University of Science and Technology

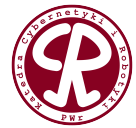


# Activation (6/10)

ReLu function properties:

- the function offers high performance gain due to its nature, training is more efficient,
- it is non-linear but is piecewise differentiable,
- can be used in multi layers due to vanishing gradient resistance,
- so-called dying neuron problem is present; negative input is always trimmed to zero causing undesirable result during back propagation – derivative can not be calculated.

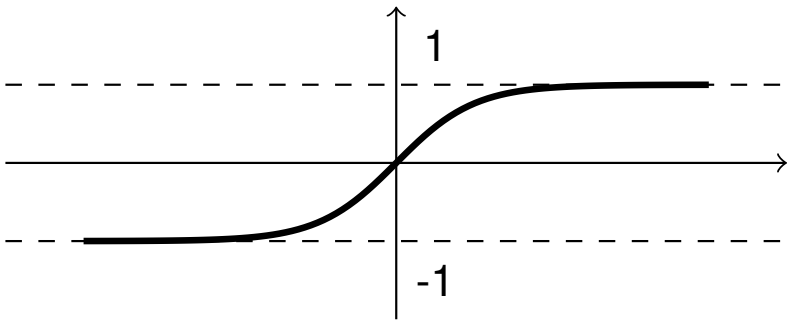
Wrocław University of Science and Technology



# Activation (7/10)

tanh function

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{4}$$

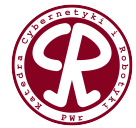


# Activation (8/10)

tanh function properties:

- like sigmoid function it is non-linear and differentiable,
- the output is range $x \in [-1, 1]$,
- can be used when negative output of a neuron is required,
- suffers from Vanishing gradient.

Wrocław University
of Science and Technology



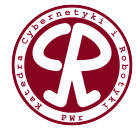
# Activation (9/10)

Softmax function

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{5}$$

| $x$ | $f(x)$ |
|---|---|
| 1 | 0.017 |
| 5 | 0.936 |
| 2 | 0.047 |

Wrocław University of Science and Technology

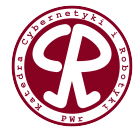


# Activation (10/10)

Softmax function properties:

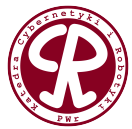
- outputs values $x \in [0, 1]$,
- elements of softmax vector sum to 1,
- outputs probability distribution,
- it is used as last layer in multiclass classification objectives.

Wrocław University of Science and Technology



# Pooling in CNNs

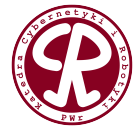
Wrocław University
of Science and Technology



# Pooling (1/2)

Pooling is the last layer in three staged convolutional layer. The purpose of pooling is to replace element of the input matrix (pixels) with statistic of it and is surrounding. For example, averaging can be used for pooling function.
This operation makes input invariant to small translations, thus small shift of input data does not affect the output. In other words, it relays more on feature presence than its location in the image.
Pooling layer downsamples the input. It uses the same approach of a window as the convolution layer but given statistic is used on the window.

Wrocław University
of Science and Technology



# Pooling (2/2)

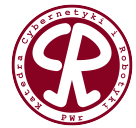
Max pooling – calculates maximum value.

Average pooling – returns average value for the window,

Global Max pooling – similar to Max pooling but returns max value over entire dimension,

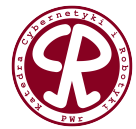
Global Average pooling – similar to Average pooling but the average is returned and as in case of Global Max pooling entire dimension is considered.

Wrocław University of Science and Technology



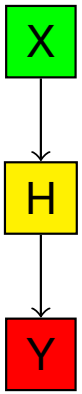
# Recurrent Neural Networks

Wrocław University
of Science and Technology

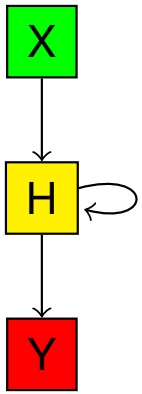


# Recurrence (1/2)

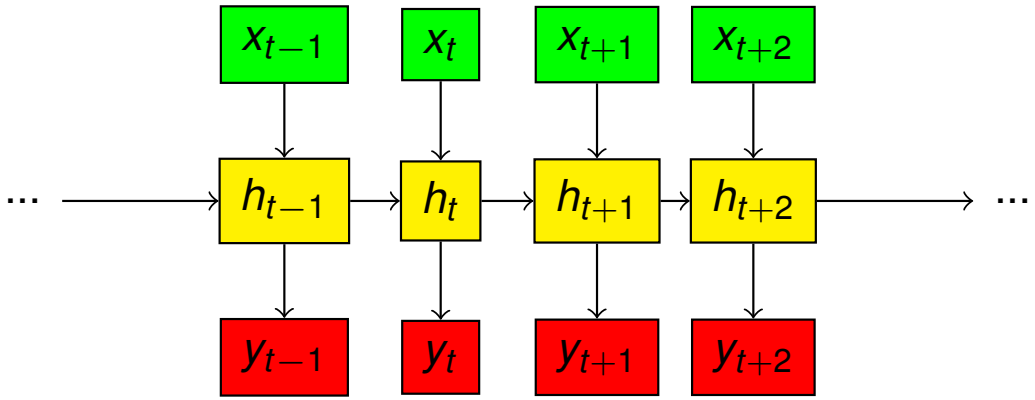
Conventional neural network



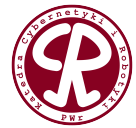

Recurrent neural network



# Recurrence (2/2)




Wrocław University
of Science and Technology



# Idea

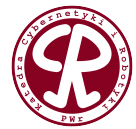
## Recurrent Neural Netowrk

RNN is a type of neural network which aims at processing sequences of values. Given a data series $x_1, x_2, \ldots, x_k$ the output can be a prediction or assessment of this time series. The length of the input vector does not necessary has to be fixed, thus variable length time vectors can be fed to the NN.

## Parameter sharing

Thanks to parameter sharing across different layers of NN we can benefit from variable length input vectors.

Wrocław University of Science and Technology

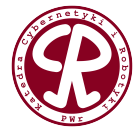


# Long term dependencies (1/2)

## Gradients

Due to depth of a neural network a phenomenon called vanishing gradient or exploding gradient can be observed. Vanishing gradient takes place when through the propagation of network's layers the gradients are getting smaller, thus they have less and less influence on the learning process. The learning process is inefficient.
On the other hand, exploding gradients tends too gain on magnitude when propagating through layers. This leads to optimization problems and relations are difficult to emerge.

Wrocław University of Science and Technology



# Long term dependencies (2/2)

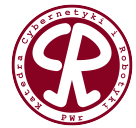
## Long term dependencies

Although short-term dependencies can be caught up easily noticing long-term dependency is difficult. It requires long input vectors while providing more data is crucial to find the relationship it can be lost due to previously mentioned issues.

## Structure

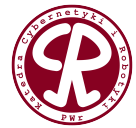
RNNs are sets of multiple (same) functions applied multiple times to the data (time series). This can lead to strong non-linearities.

Wrocław University of Science and Technology



# LSTM
# Long Short-Term Memory

Wrocław University
of Science and Technology



# Long Short-Term Memory (1/1)

## LSTM

LSTM is a representative of gated RNNs. They are based upon an idea that there exists connections across time which have derivatives but do not suffer from gradients issues such as vanishing and exploding ones.

Wrocław University of Science and Technology

# Gates

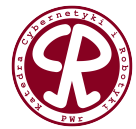
LSTM unit consists of three different types so-called gates:

**forget gate** controls cell state through memory cell. Forget gate can allow which information should be forwarded and what information should be suppressed. When activated it erases long-term state from the cell.

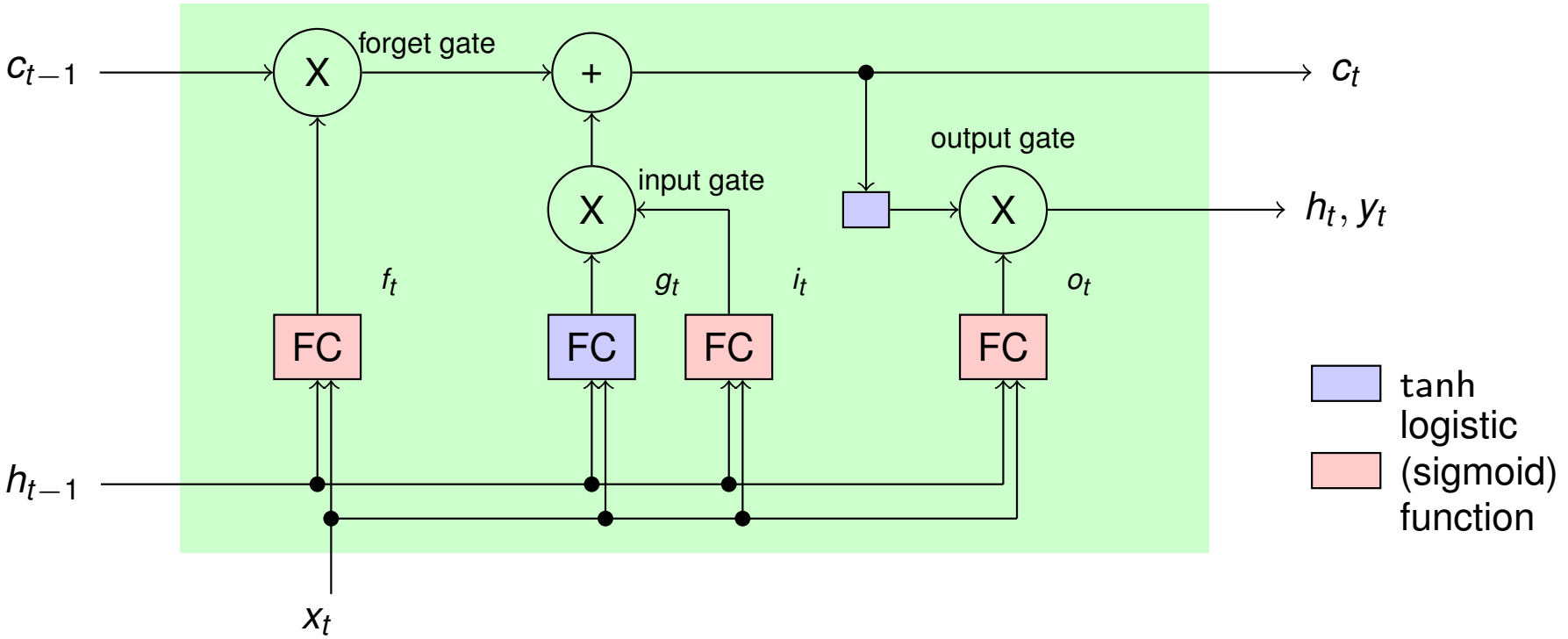
**input gate** watches over updating the cell's state. Is responsible for controlling what data ($g_t$) should be passed to long-term memory.

**output gate** is responsible of updating hidden values of the unit by controlling which parts of the state (long-term memory) should be passed forward.

Wrocław University of Science and Technology



# LSTM architecture (1/4)



# LSTM architecture (2/4)

- $c_t$ is long-term state. When this information passes through LSTM cell it is being suppressed by forget gate. After that, some new information is being added through the means of input gate control. After these operations the signal is passed to the next cell.
- $h_t$ is considered to be short-term memory. It is produced from $c_t$ which in turn is fed to activation layer (tanh). The output of the short-term memory is controlled by output gate which modifies current short-term memory state and the previous state $h_{t-1}$.
- $y_t$ output of the LSTM cell which is considered to be the same as given short-term memory state $h_t$.

Wrocław University of Science and Technology

# LSTM architecture (3/4)

$$i_t = \sigma(W_{xi}^T x_t + W_{hi}^T h_{t-1} + b_i) \quad (6)$$

$$f_t = \sigma(W_{xf}^T x_t + W_{hf}^T h_{t-1} + b_f) \quad (7)$$

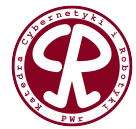
$$o_t = \sigma(W_{xo}^T x_t + W_{ho}^T h_{t-1} + b_o) \quad (8)$$

$$g_t = \tanh(W_{xg}^T x_t + W_{hg}^T h_{t-1} + b_g) \quad (9)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t \quad (10)$$

$$y_t = h_t = o_t \otimes \tanh(c_t) \quad (11)$$

Wrocław University
of Science and Technology



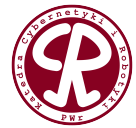
# LSTM architecture (4/4)

$W_{x\boxtimes}$ are four weight matrices for $i_t$, $f_t$, $o_t$ and $g_t$ layers, and theirs input vectors $x_t$.

$W_{h\boxtimes}$ are four weight matrices for $i_t$, $f_t$, $o_t$ and $g_t$ layers, and theirs previous short-term memory state $h_{t-1}$.

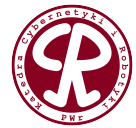
$b_{\boxtimes}$ are biases for $i_t$, $f_t$, $o_t$ and $g_t$ layers. The $b_f$ is initialized with ones instead of zeros to prevent instant forgetting at the very beginning of the training process.

Wrocław University of Science and Technology



# GRU
# Gated Recurrent Unit

Wrocław University
of Science and Technology

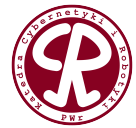


# GRU

## GRU

Gated Recurrent Units are another recurrent architecture implementation. GRU allow to dynamically adjust memory retention time and forgetting behaviour. The main difference between LSTM and GRU is that the later has a single gate unit which controls both forgetting factor and updates the unit's state.

Wrocław University of Science and Technology



# Gates

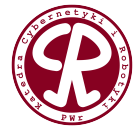
GRU unit consists of two (three) different types of gates:

**update gate** controls influence of previous memories on the current state of the unit. It is similar to output gate in LSTM architecture.

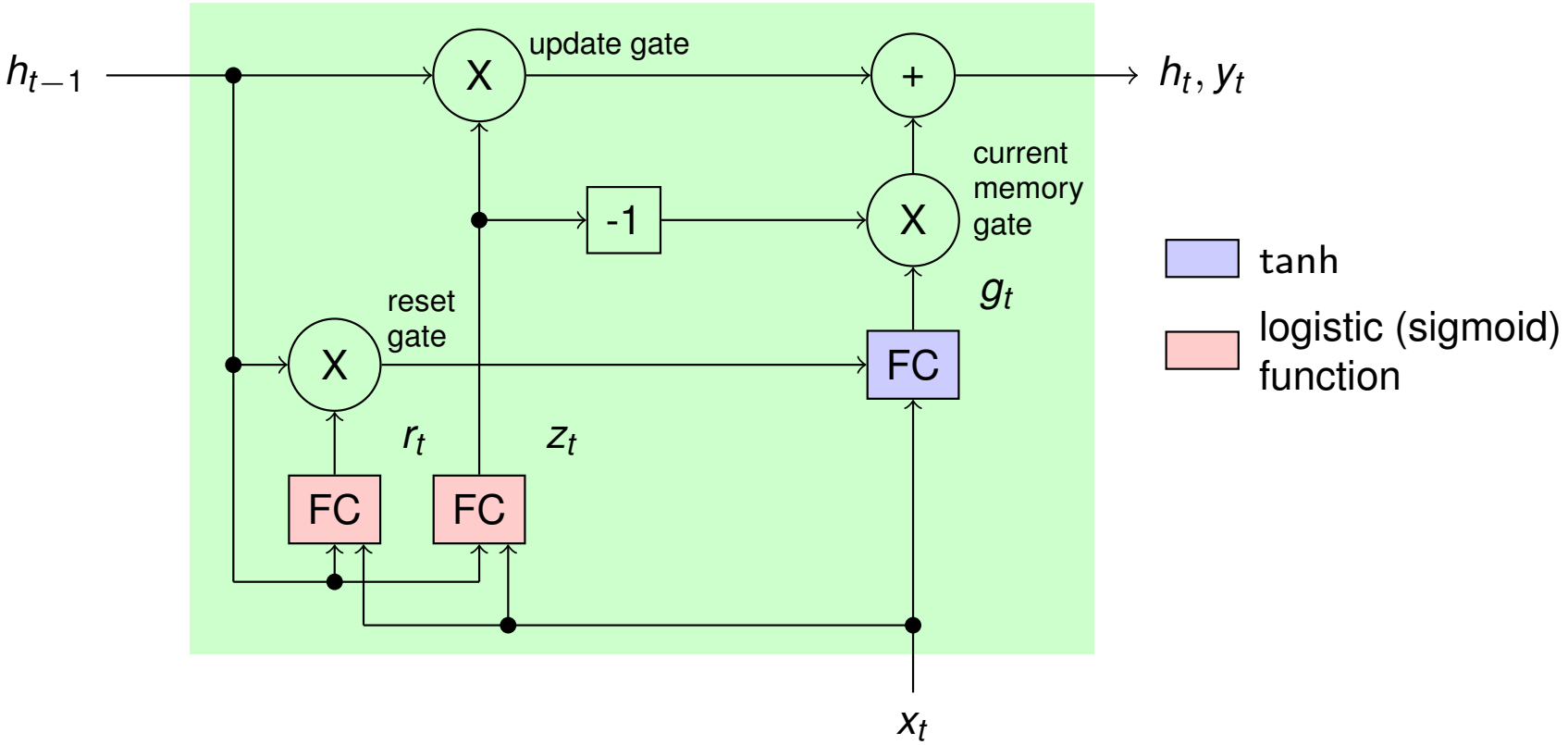
**reset gate** determines how much knowledge should be let go. Combines input and forget gate in LSTM architecture.

**current memory gate** can be considered as part of the reset gate. It allows to reduce impact of memories (previous state) on current state.

Wrocław University of Science and Technology



# GRU architecture (1/4)

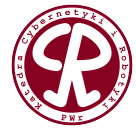


# GRU architecture (2/4)

$h_t$ is considered to be the state. It holds both short-term and long-term state.

$y_t$ output of the GRU cell which is considered to be the same as given state $h_t$.

Wrocław University of Science and Technology



# GRU architecture (3/4)

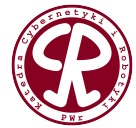
$$z_t = \sigma(W_{xz}^T x_t + W_{hz}^T h_{t-1} + b_z) \tag{12}$$

$$r_t = \sigma(W_{xr}^T x_t + W_{hr}^T h_{t-1} + b_r) \tag{13}$$

$$g_t = \tanh(W_{xg}^T x_t + W_{hg}^T (r_t \otimes h_{t-1}) + b_g) \tag{14}$$

$$y_t = h_t = z_t \otimes h_{t-1} + (1 - z_t) \otimes g_t \tag{15}$$

Wrocław University
of Science and Technology



# GRU architecture (4/4)

$W_{x\boxtimes}$ are three weight matrices for $z_t$, $r_t$ and $g_t$ layers, and theirs input vector $x_t$.

$W_{h\boxtimes}$ are three weight matrices for $z_t$, $r_t$ and $g_t$ layers, and theirs previous short-term memory state $h_{t-1}$.

$b_{\boxtimes}$ are biases for $z_t$, $r_t$ and $g_t$ layers.

Wrocław University of Science and Technology