Politechnika Warszawska

MSI

12. Sieci neuronowe głębokie (DNN)

Włodzimierz Kasprzak

Treść

1. Głębokie sieci neuronowe
2. Splotowe sieci neuronowe CNN
3. Rekurencyjna sieć neuronowa RNN
4. Warianty sieci RNN

1. Głębokie sieci neuronowe

Typowe architektury głębokich sieci neuronowych (ang. deep neural networks, DNN):

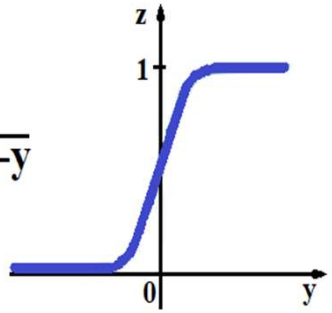
1. Klasyfikacja wzorców poprzez sieci wielowarstwowe („w pełni połączone”) z warstwą wyjściową typu „soft-max”;
2. Splotowe sieci neuronowe (ang. *convolutive neural networks*, CNN) do wyznaczania cech i klasyfikacji obrazów 2D/3D;
3. Sieci splotowe R-CNN i jej modyfikacje;
4. Rekurencyjne sieci neuronowe – RNN, LSTM - do modelowania sekwencji czasowych;
5. *Głębokie auto-enkodery.*
6. *Sieć antagonistyczna;*
7. *Sieci korelacyjne i syjamskie;*
8. *Grafowe sieci (grafowe sieci splotowe).*

Funkcje aktywacji

Typowe funkcje aktywacji (funkcja generująca wyjście neuronu na podstawie pobudzenia od wejść i wag połączeń)

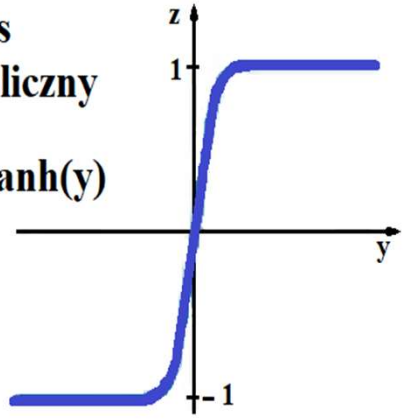
Funkcja sigmoidalna

$$z(y) = \frac{1}{1+ e^{-y}}$$



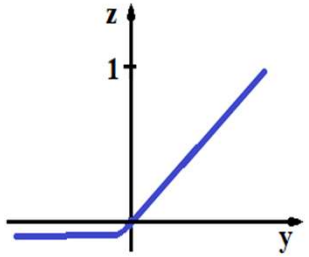
Tangens hiperboliczny

$$z(y) = \tanh(y)$$



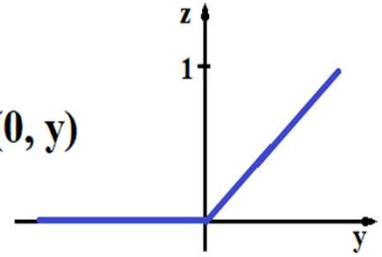
ELU

$$z(y) = \begin{cases} y \text{ , dla } y \geqslant 0 \\ \alpha(e^{y} - 1) \text{ , dla } y < 0 \end{cases}$$



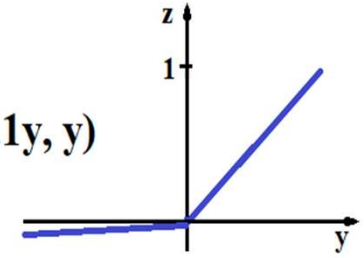
ReLU

$$z(y) = \max(0, y)$$



Leaky ReLU

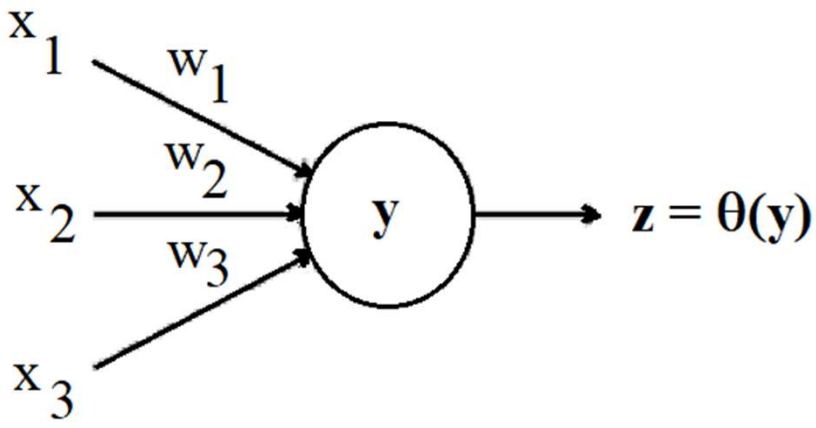
$$z(y) = \max(0.1y, y)$$



Sieci głębokie „w pełni połączone”

- Najprostsze sieci głębokie posiadają warstwy wyjściowe ”w pełni połączone” (ang. *Fully Connected Layer*, FC). Np. wielowarstwowy perceptron jest już przykładem sieci o w pełni połączonych warstwach.
- Wyjście każdego neuronu to wartość funkcji aktywacji dla sumy iloczynów wszystkich neuronów wejściowych i ich wag.

Np.



2. Splotowe sieci neuronowe CNN

Splotowe sieci neuronowe (ang. *convolutive neural networks*, CNN) to sieci jednokierunkowe, które posiadają specyficzną organizację neuronów w warstwy, dostosowaną do danych 2-wymiarowych (np. obrazów cyfrowych) posiadających topologię macierzy 2D (kraty).

Warstwa splotowa realizuje jednoczesną filtrację neuronów wejściowych za pomocą zbioru filtrów splotowych charakteryzowanych 2-wymiarowymi macierzami liczb, tzw. „jądrami" filtra. Jest to część liniowa (funkcja pobudzenia). Typowo stosowaną funkcją aktywacji jest ReLU.

Warstwa łącząca („pooling") zazwyczaj następuje po każdej warstwie splotowej. Odpowiednia funkcja próbkuje lokalne kraty wyników w warstwie splotowej zmniejszając rozdzielczość kraty wyników. Typowe funkcje próbkujące to: „*max pooling*", która zwraca maksymalną wartość lokalnej kraty, lub „*Lp-pooling*", która realizuje wygładzanie wyników lokalnej kraty.

Splotowe sieci neuronowe CNN

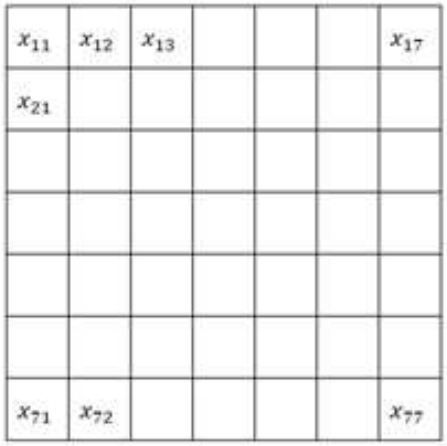
Część wyjściowa pełni rolę klasyfikatora i zazwyczaj ma postać dwuwarstwowego perceptronu (nazywanego siecią „w pełni połączoną” (*fully connected*) lub o ”gęstych” połączeniach (*dense*), przy czym warstwa wyjściowa jest zwykle typu „softmax” – określa rozkład prawdopodobieństwa *a posteriori* klas.

2-wymiarowa operacja splotu

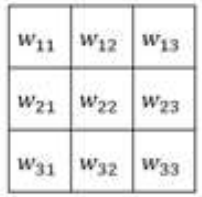
Dane: mapa wejściowa X,
maska filtra $\mathbf{f}$.
Wynik: mapa cech H
Krok przesunięcia maski
„stride” = $(\mathrm{S_w}, \mathrm{S_h})$

Np. dla filtra o rozmiarze
$(f_w, f_h) = (3, 3)$:

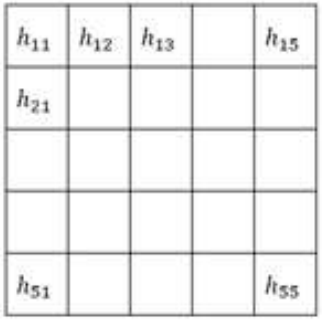
$$h_{13} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j+2} + b$$



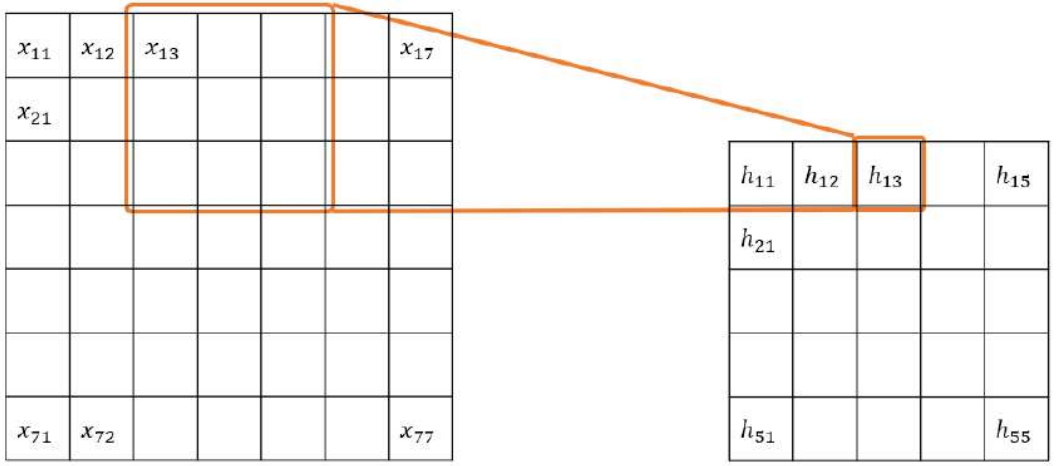
Mapa wejściowa X
o rozmiarze
$(\mathrm{X_{width}}, \mathrm{X_{height}})$



Maska
filtra f
o rozmiarze
$(\mathrm{f_w}, \mathrm{f_h})$



Mapa cech H
o rozmiarze
$(\mathrm{H_{width}}, \mathrm{H_{height}})$



Mapa X

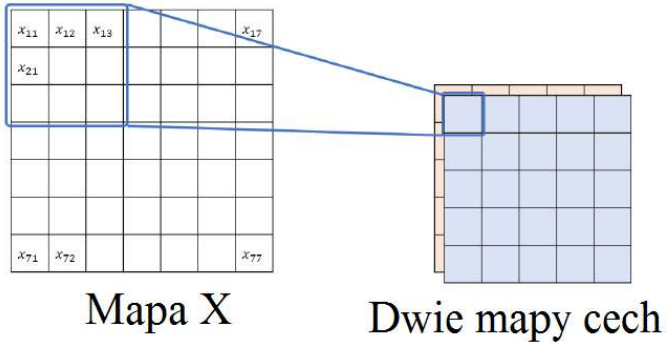
Mapa cech H

Warstwa splotowa

Kanały wyjściowe: liczba f_{num} masek splotowych (filtrów) stosowanych w warstwie sieci o X_{num} mapach wejściowych (tzw. kanałów) prowadzi do wielokrotnej liczby map wyjściowych H (tzw. kanałów) dla jednej mapy wejściowej:

$$Hnum_{wyjścia} = Xnum_{wejścia} \cdot f_{num}$$

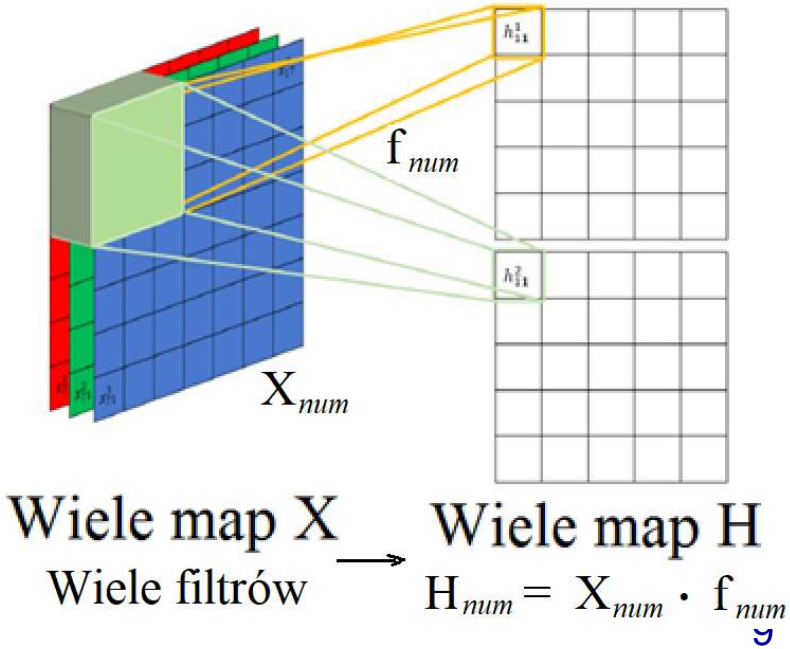
Np. dwa filtry w warstwie generują dwie mapy (kanały) wyjściowe dla każdej mapy wejściowej.



Wiele map wejściowych

i wielokrotne kanały wyjściowe:

$$Hnum_{wyjścia} = Xnum_{wejścia} \cdot f_{num}$$

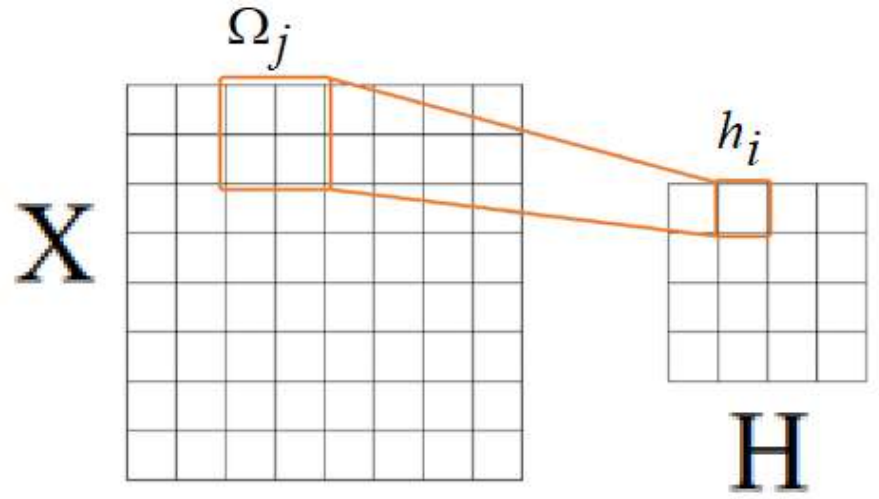


Padding (P_h, P_w) oznacza rozszerzenie mapy wejściowej o P_h wierszy i P_w kolumn dla pełnego wykorzystania brzegowych danych mapy.

Próbkowanie („pooling”)

„Pooling”: redukcja rozmiaru mapy wejściowei.

Np. $pool = 2\times2$: każdy podobszar Ω_j o rozmiarze 2×2 wejściowej mapy cech X odwzorowywany jest na jeden element mapy wyjściowej H.



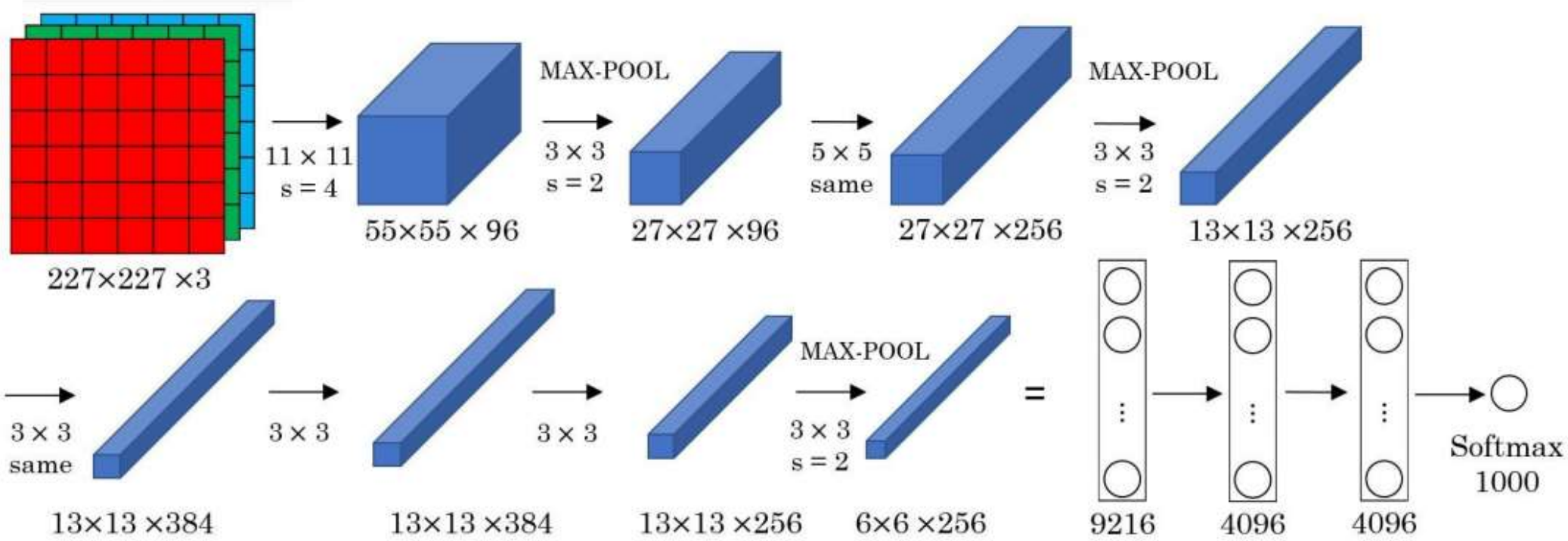
Funkcje stosowane w warstwie „pooling”.

- „Max pooling”: określa maksymalną wartość w każdym podobszarze Ω_j;
- „Avg pooling”: średnia (lub suma) - określa wartość średnią lub sumę elementów podobszaru Ω_j;
- „L_p-pooling”: dla każdego obszaru Ω_j określa normę rzędu p (L_p), czyli

$$h_j = \left(\sum_{i \in \Omega_j} x_i^p \right)^{1/p}$$

Przykład sieci CNN

Sieć „AlexNet”



https://harangdev.github.io/deep-learning/convolutional-neural-networks/25/

- Funkcja aktywacji ReLU.
- Trenowana na ImageNet
- Posiada ok. 60 mln parametrów

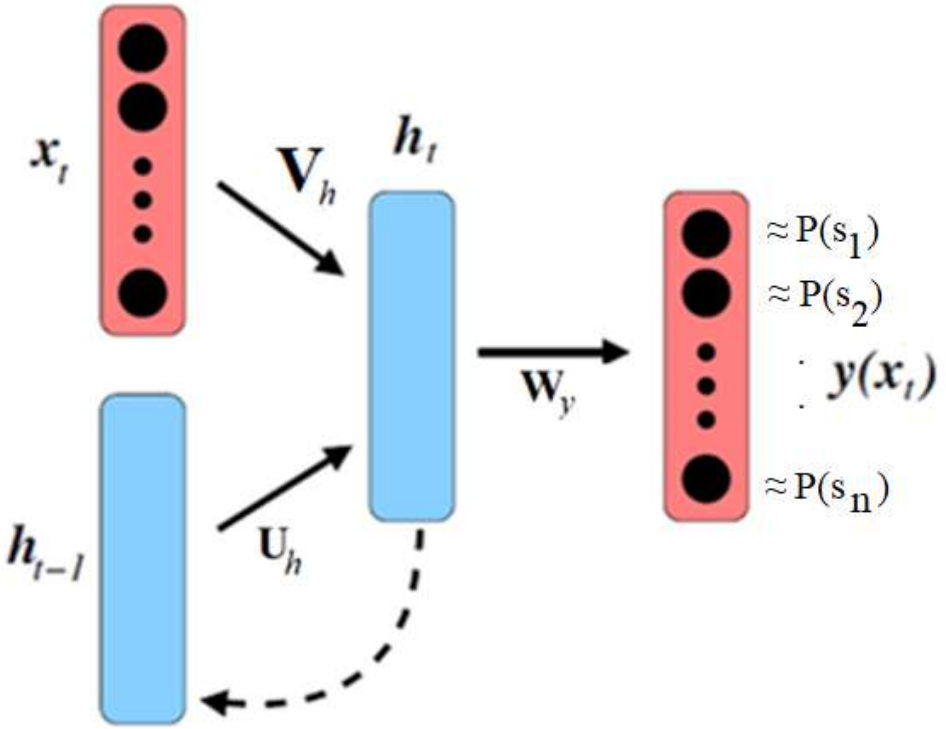
3. Rekurencyjna sieć neuronowa (RNN)

Dana jest sekwencja obserwacji w czasie: $\boldsymbol{x}_{1:t} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$.

Warstwa rekurencyjna sieci RNN

$$\boldsymbol{h}_t = \boldsymbol{f}_h(\mathbf{V}_h \boldsymbol{x}_t + \mathbf{U}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$$

$$\boldsymbol{y}_t = \boldsymbol{f}_y(\mathbf{W}_y \boldsymbol{h}_t + \boldsymbol{b}_y)$$



RNN realizuje aproksymację funkcji, $\boldsymbol{y}(\boldsymbol{x}_{1:t}) \cong \mathrm{F}(\boldsymbol{x}_{1:t})$,

przy założeniu:

$$\mathrm{F}(\boldsymbol{x}_{1:t}) = \{\, \mathrm{F}(\boldsymbol{x}_i, \boldsymbol{x}_{1:i-1}) \mid i=1,\ldots,t \,\}$$

$$\cong \{\, \mathrm{F}(\boldsymbol{x}_i, \boldsymbol{h}_{i-1}) \mid \ldots \}$$

$$\cong \{\, \mathrm{F}(\boldsymbol{h}_i) \mid \ldots \} = \{\, \boldsymbol{y}_i \mid i=1,\ldots,t \} = \boldsymbol{y}_{1:t}$$

Efektem jest sekwencja wyjściowa o identycznej długości co sekwencja wejściowa: $\boldsymbol{x}_{1:t} \rightarrow \boldsymbol{y}_{1:t}$

Sieć głęboka RNN

Zauważmy, że warstwa rekurencyjna może zostać wielokrotnie powielona, podobnie jak możliwych jest wiele warstw ukrytych w sieci jednokierunkowej MLP.

Wprowadźmy indeks górny (i) reprezentujący numer kolejnej warstwy ukrytej rekurencyjnej, $i = 1, 2, \ldots, \mathrm{L}$

$$\boldsymbol{h}_t^{(i)} = \boldsymbol{f}_h\left(\mathbf{W}_h^{(i)}\,\boldsymbol{h}_t^{(i\text{-}1)} + \mathbf{U}_h^{(i)}\,\boldsymbol{h}_{t-1}^{(i)} + \boldsymbol{b}_h^{(i)}\right), \text{ gdzie } \boldsymbol{h}_t^{(1)} = \boldsymbol{x}_t$$

Wyjście sieci:

$$\boldsymbol{y}_t = \boldsymbol{f}_y\left(\mathbf{W}_y\,\boldsymbol{h}_t^{(L)} + \boldsymbol{b}_y\right)$$

Daje to możliwość modelowania bardzo złożonych zależności pomiędzy danymi a także prowadzi do znaczącego oddzielenia poziomu zmiennych „stanu” od zmiennych obserwacji.

4. Warianty sieci RNN

4.1 Sekwencja czasowa sieci RNN

- jednokierunkowa
- dwukierunkowa

4.2 Modyfikacja warstwy RNN (dodanie „bramek”):

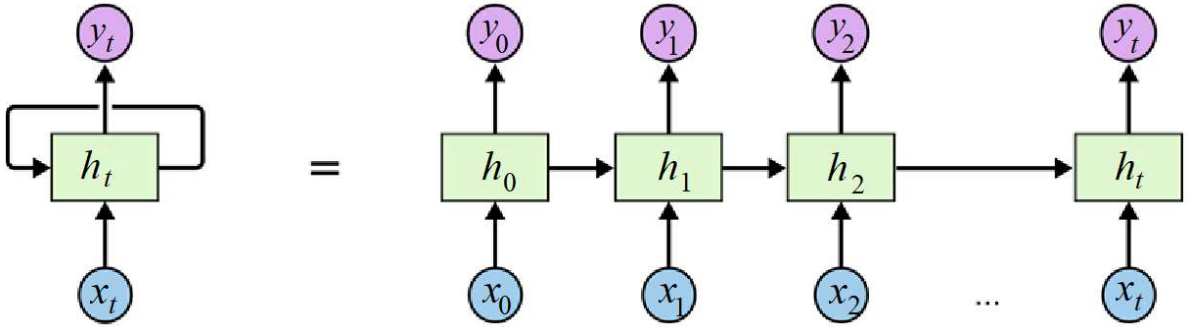
- bramkowane neuronów (GRU)

4.3 Bramkowanie sekwencji czasowej

- sieć „long-short term memory” (LSTM)

4.1 Sekwencja „czasowa” sieci RNN

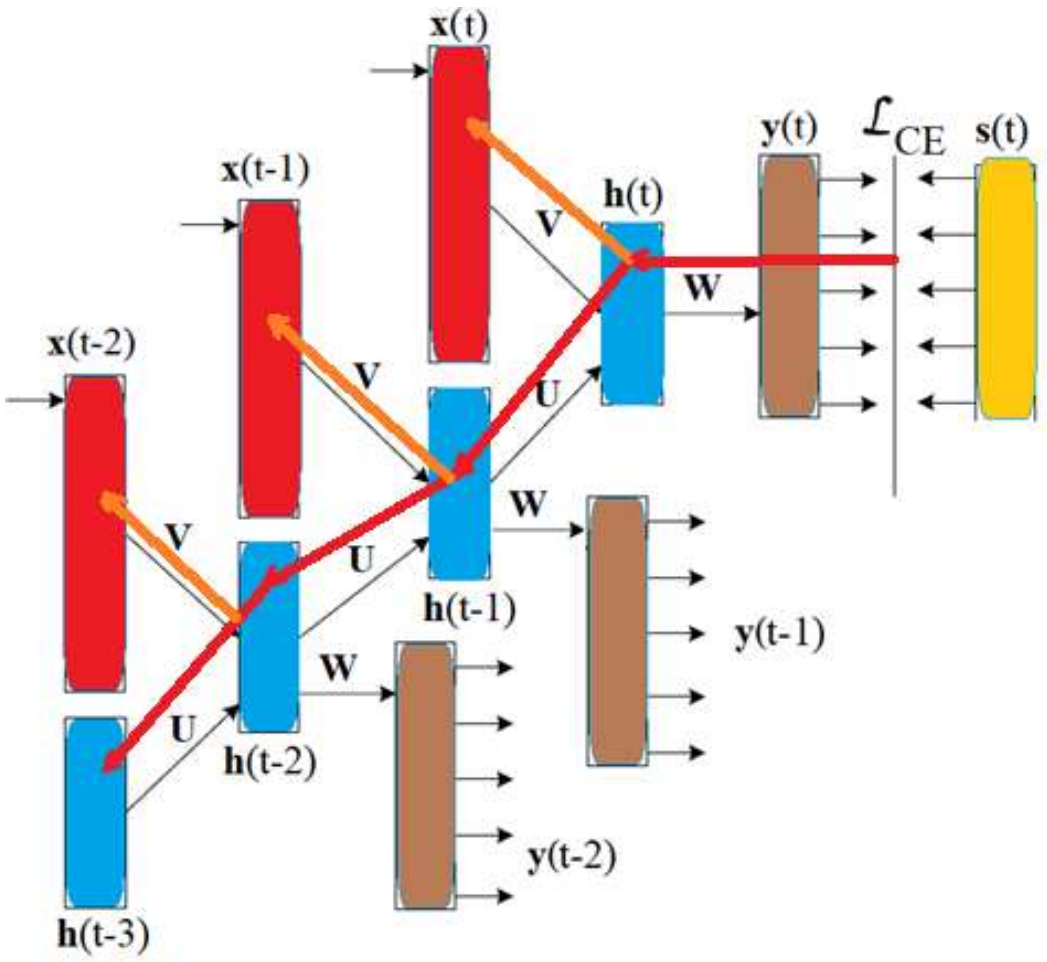
Rozwinięciem sieci RNN jest architektura **sekwencji czasowej takich** sieci, modelująca zależności czasowe dłuższe niż 2 kolejne chwile czasowe. Neurony z połączeniami rekurencyjnymi działają jak pamięć - potencjalnie są one w stanie modelować relacje występujące w dowolnie długim zakresie czasowym.



Umożliwia to lepsze modelowanie zależności czasowych a proces uczenia wag sieci oparty jest o dłuższe sekwencje czasowe danych niż o długości 2. Podstawowym algorytmem uczącym takiej sieci jest wsteczna propagacja względem czasu BPTT („*BackProgation Through Time*”), a w praktyce algorytm „przyciętego” BPTT („truncated BPTT).

Uczenie sekwencji sieci RNN (1)

Proces uczenia wag można przedstawić w postaci uczenia równoważnej sieci jednokierunkowej, rozwiniętej z sieci podstawowej RNN w sekwencję „czasową”, o pewnej liczbie K powtórzeń sieci RNN „indeksowanych czasem”, w których wagi każdej z sieci, **V**, **U** i **W** oraz wagi wejść hamujących (*bias*), są wspólne dla wszystkich warstw w „czasie”.



BPTT - uczenie sekwencji sieci RNN

Dane uczące dla sekwencji *K* sieci RNN to *N* par danych:

$\langle \mathbf{x}_1, \mathbf{s}_1 \rangle, \langle \mathbf{x}_2, \mathbf{s}_2 \rangle, \ldots, \langle \mathbf{x}_N, \mathbf{s}_N \rangle$, gdzie $\mathbf{x}_i$ to dane wejściowe a $\mathbf{s}_i$ odpowiadające im oczekiwane dane wyjściowe.

Nieznane początkowe wartości warstwy ukrytej $\boldsymbol{h}_0$ przyjmuje się za zerowe. Dla uczenia sieci neuronowych istnieją różne formy funkcji straty (optymalizowanej funkcji celu) ale dla RNN przyjmuje się zwykle entropię krzyżową.

Równania sieci o indeksie czasu *t* :

$$y_k(t) = \text{softmax}\left(\sum_{r=1}^{n_H} w_{kr}\, h_r(t) + b_k\right)$$

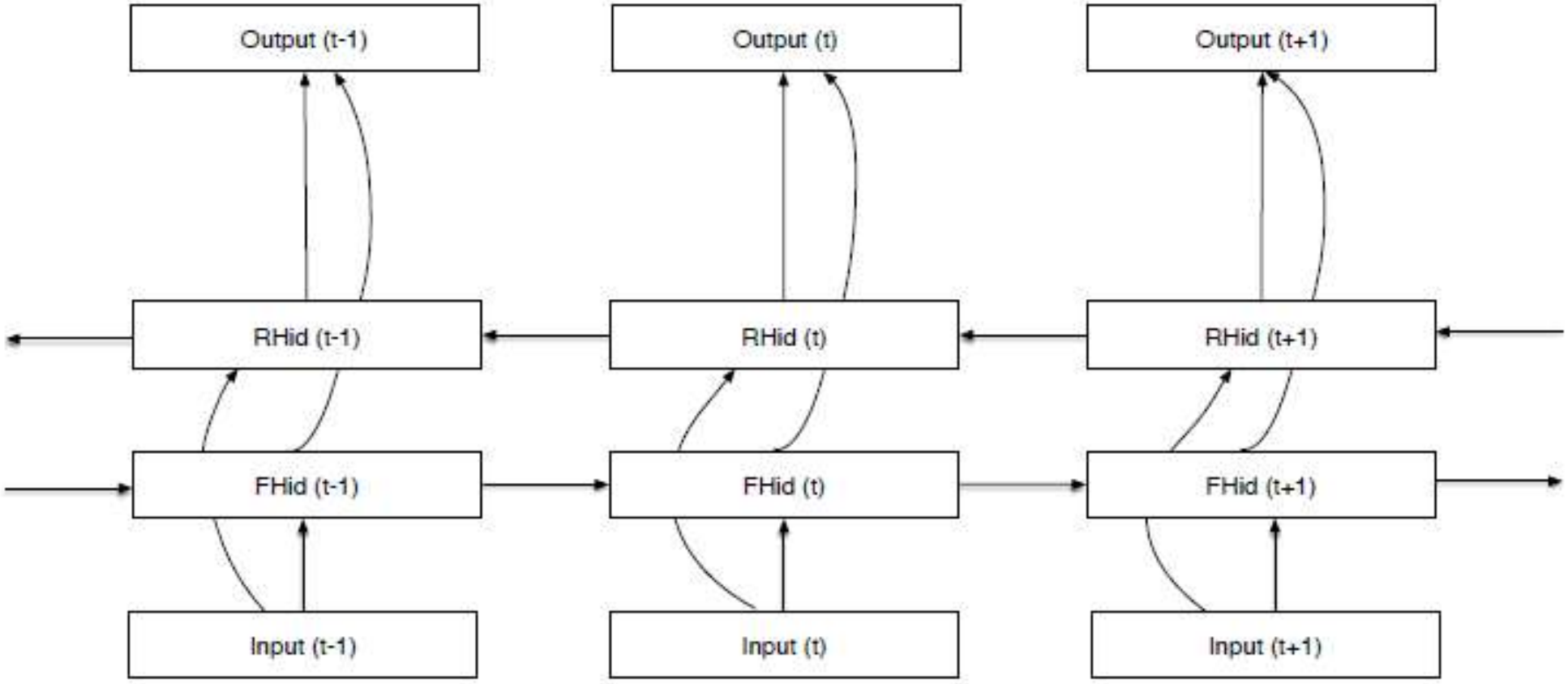
$$h_j(t) = \text{sigmoid}\left(\sum_{s=0}^{n_X} v_{js}\, x_s(t) + \sum_{r=0}^{n_H} u_{jr}\, h_r(t-1) + b_j\right)$$

BPTT - uczenie sekwencji sieci RNN

Back_Propagation_Through_Time(x[ ], s[ ], *K, N*)
// x[*t*] to wektor wejściowy w chwili t ; y[*t*] to oczekiwany wektor wyjściowy
// w chwili *t* ; K to długość sekwencji sieci RNN; N to długość sekwencji
// danych uczących
POWTARZAJ do spełnienia warunku końca (np. liczba epok)
 h[0:*K*-1] := 0;
 FOR *t* = 1, …, *N* - *K* - 1 DO //
   wejście_sieci ← (h[0:*K*-1], x[t], x[t+1], …, x[t+K])
   y[*t*+*K*] := propagacja_wprzód(x[ ], h [ ])
   e := L_{CE}(s[*t*+*K*], y[t+K]); // funkcja straty
   (∆W,∆V[K],∆V[K-1], …, ∆V[1], ∆U[K],∆V[K-1], …, ∆U[1]) =
                    propagacja_wstecz_błędu(e);
   ∆V = ∆V[K] + ∆V[K-1] + … + ∆V[1]; // suma zmian wag
   ∆U = ∆U[K] + ∆U[K-1] + … + ∆u[1]; // suma zmian wag
   (W, V, U) = modyfikuj_wagi (∆W, ∆V, ∆U)
   h[0:*K*-1] := h[1:K]; // pamięć dla następnej iteracji
 END
END

Dwukierunkowa sekwencja sieci RNN

Wzajemne połączenie dwóch kolejnych warstw ukrytych sieci RNN – daje możliwość uwzględnienia zależności dwukierunkowych **W** czasie.



4.2 GRU RNN

Gated recurrent unit (GRU)

Dodatkowo występują komórki pamięci przeznaczone do sterowania zapamiętywaniem informacji przez elementy rekurencyjne. Są to:

- bramka aktualizacji (*update* gate) z_t ,
- bramka „zerowania” (*reset* gate) r_t

Aktualizacja neuronu w warstwie ukrytej (rekurencyjnej) ma postać:

$$\boldsymbol{h}_t = (1 - z_t) \odot \boldsymbol{h}_{t-1} + z_t \odot \widetilde{\boldsymbol{h}_t}$$

gdzie $\mathbf{z}_t = \sigma(\mathbf{W}_\mathrm{z} \boldsymbol{x}_t + \mathbf{U}_\mathrm{z} \boldsymbol{h}_{t-1} + \boldsymbol{b}_\mathrm{z})$,

$$\widetilde{\boldsymbol{h}_t} = \tanh(\mathbf{W}\boldsymbol{x}_t + \mathbf{U}(\mathrm{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h)$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_\mathrm{r} \boldsymbol{x}_t + \mathbf{U}_\mathrm{r} \boldsymbol{h}_{t-1} + \boldsymbol{b}_\mathrm{r})$$

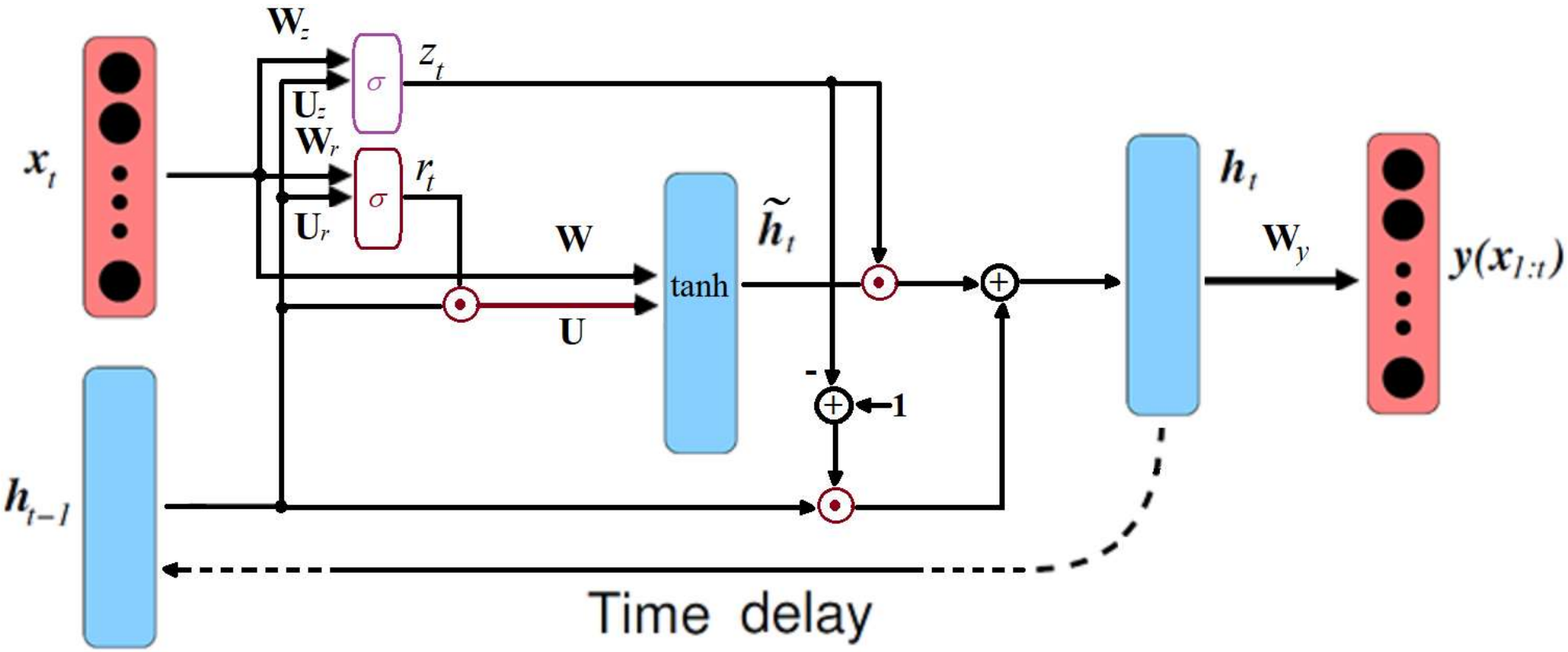
$\odot$ oznacza mnożenie *Hadamarda* (mnożenie punkt-po-punkcie),

$\sigma()$ jest zwykle funkcją sigmoidalną.

Warstwa wyjściowa nie zmienia się: $\boldsymbol{y}(\boldsymbol{x}_{1:t}) = \boldsymbol{f}^f(\mathbf{W}_\mathrm{y} \boldsymbol{h}_t + \boldsymbol{b}_\mathrm{y})$

GRU RNN

Gated recurrent unit (GRU)



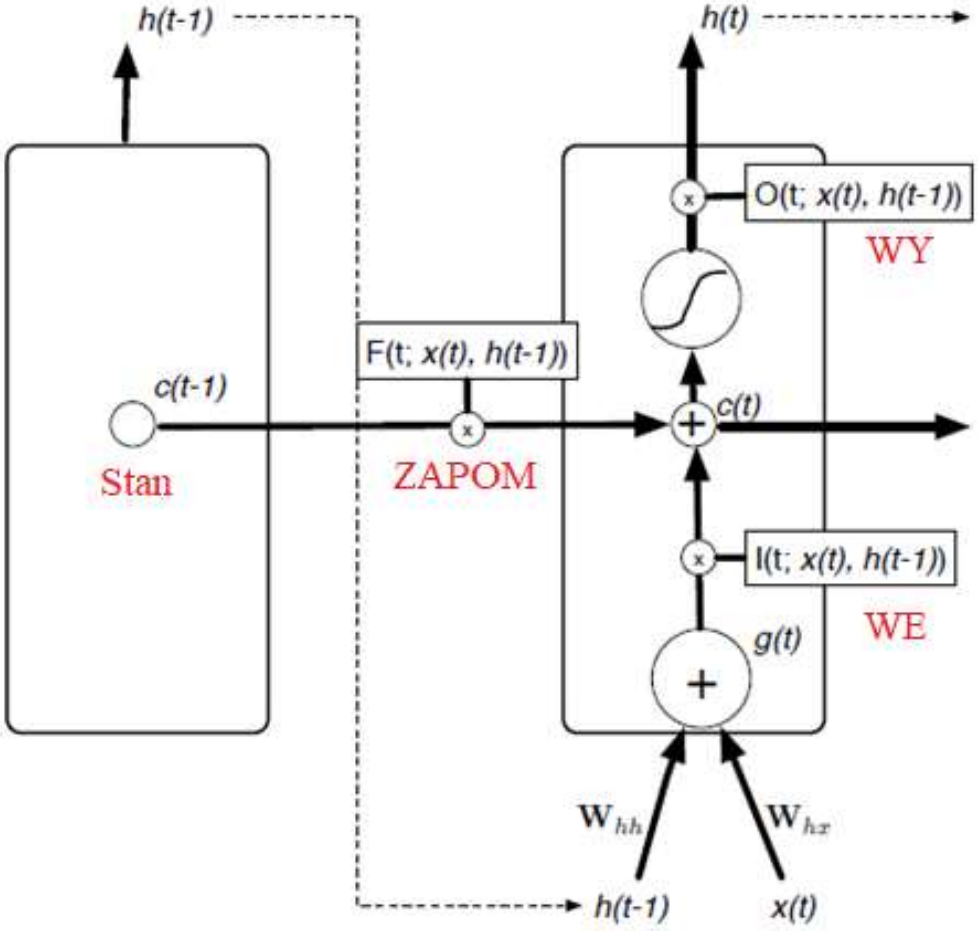
4.3 Sieć LSTM

Specyficznym rozwiązaniem sekwencji sieci rekurencyjnych stosowanym w modelowaniu sekwencji czasowych jest **sieć LSTM** (ang. *Long Short-Term Memory*):

- Każdy element warstwy ukrytej posiada wewnętrzną pamięć (stan C) - steruje ona usuwa problem zanikania gradientu przy długich opóźnieniach, umożliwia podtrzymywanie odległych informacji.
- Wprowadza bramki dla selekcji informacji. Wyróżnia się w niej bramkę wejściową, bramkę wyjściową i bramkę zapominania:
  - **Bramka wejściowa I** – określa to, która *nowa* wartość wpływa na wynik,
  - **Bramka wyjściowa O** – określa to, które wartości są używane do obliczeń wartości funkcji aktywacji,
  - **Bramka zapominania F –** określa to, które wartości są zapominane.

Funkcje sieci LSTM w warstwie ukrytej

Stan, wagi połączeń, bramki i ich funkcje:



$$\boldsymbol{I}(t) = \sigma\left(\boldsymbol{W}_{ix}\boldsymbol{x}(t) + \boldsymbol{W}_{ih}\boldsymbol{h}(t-1) + \boldsymbol{b}_i\right)$$
$$\boldsymbol{F}(t) = \sigma\left(\boldsymbol{W}_{fx}\boldsymbol{x}(t) + \boldsymbol{W}_{fh}\boldsymbol{h}t - 1) + \boldsymbol{b}_f\right)$$
$$\boldsymbol{O}(t) = \sigma\left(\boldsymbol{W}_{ox}\boldsymbol{x}(t) + \boldsymbol{W}_{oh}\boldsymbol{h}(t-1) + \boldsymbol{b}_o\right)$$
$$\boldsymbol{g}(t) = \boldsymbol{W}_{hx}\boldsymbol{x}(t) + \boldsymbol{W}_{hh}\boldsymbol{h}(t-1) + \boldsymbol{b}_h$$
$$\boldsymbol{c}(t) = \boldsymbol{F}(t) \circ \boldsymbol{c}(t-1) + \boldsymbol{I}(t) \circ \boldsymbol{g}(t)$$
$$\boldsymbol{h}(t) = \boldsymbol{O}(t) \circ \tanh\left(\boldsymbol{c}(t)\right)$$

σ - funkcja sigmoidalna

$\circ$ - mnożenie punktowe

Pytania

1. Przedstawić funkcje aktywacji stosowane w głębokich sieciach neuronowych.
2. Omówić budowę sieci CNN.
3. Przedstawić warianty sieci rekurencyjnej RNN