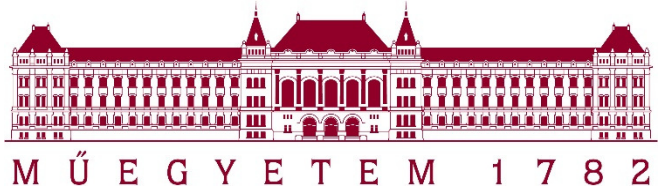


Gyires-Tóth Bálint

Deep Learning a gyakorlatban Python és LUA alapon RNN, LSTM

SmartLab
Intelligent Interactions

http://smartlab.tmit.bme.hu

NVIDIA
GPU
EDUCATION
CENTER

Jogi nyilatkozat

Jelen előadás diái a „*Deep Learning a gyakorlatban Python és LUA alapon*” című tantárgyhoz készültek és letölthetők a http://smartlab.tmit.bme.hu honlapról.

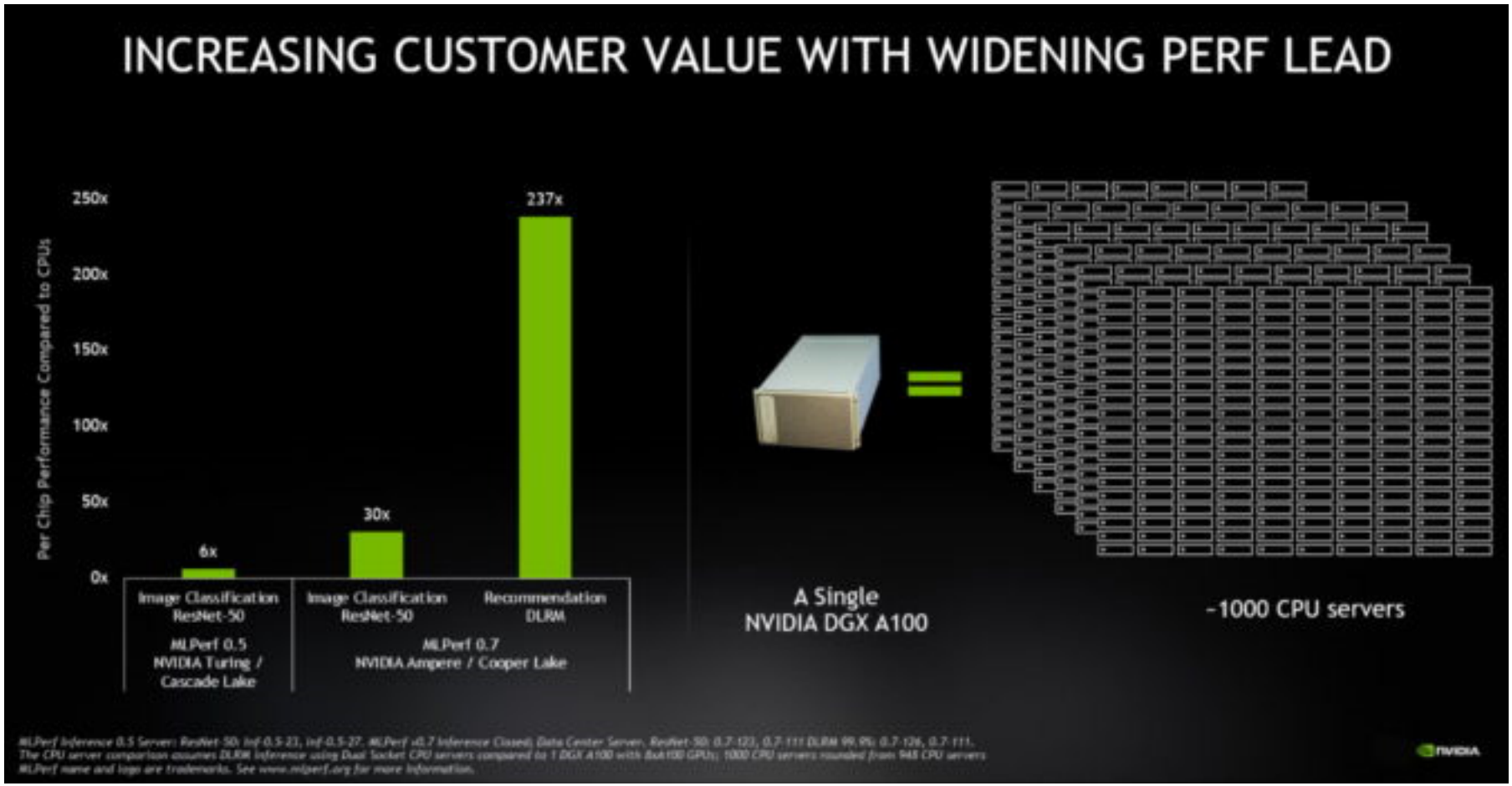
A diák nem helyettesítik az előadáson való részvételt, csupán emlékeztetőül szolgálnak.

Deep Learning Híradó

Hírek az elmúlt 48 órából

ML Perf új rekord

https://blogs.nvidia.com/blog/2020/10/21/inference-mlperf-benchmarks/



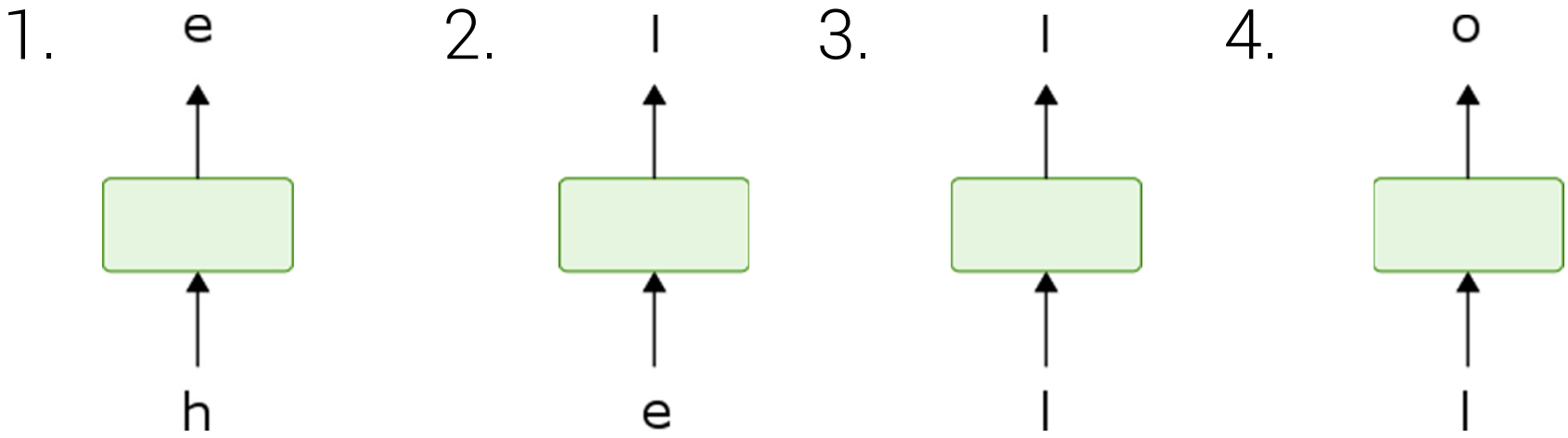
Nagy házi feladat

- Második mérföldkő (november 22. 23:59)
- Aláírás
  - Határidő: december 11. 23:59
  - Github: forráskód, readme.md, beszámoló PDF
  - Megajánlott jegy esetén is!!!
- Megajánlott jegy
  - Határidő: december 11. 23:59
  - I. és II. mérföldkő teljesítése határidőre kötelező feltétel
  - Angol nyelv használata kötelező! (forráskódban, beszámolóban)
  - Github-on az utolsó, határidő előtti commit-ot nézzük

http://smartlab.tmit.bme.hu/oktatas-deep-learning-nagy-hazi

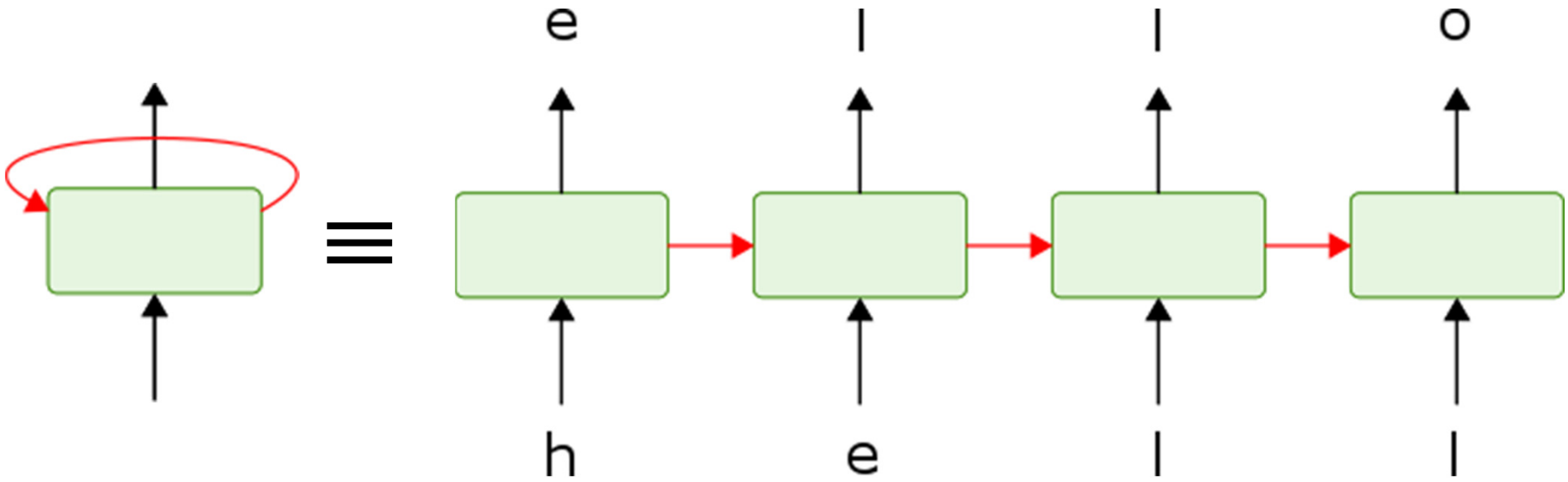
Ismétlés

Előrecsatolt hálók: a kimenet az aktuális bemenettől függ.

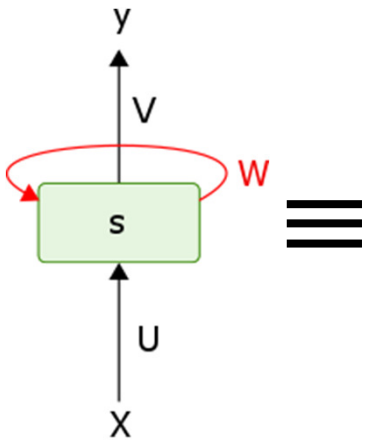


Rekurrens neurális hálózatok (RNN)

Recurrent Neural Network (RNN): a kimenet az aktuális bemenettől és a korábbi bemenetektől is függ.
Memóriája van a hálózatnak!

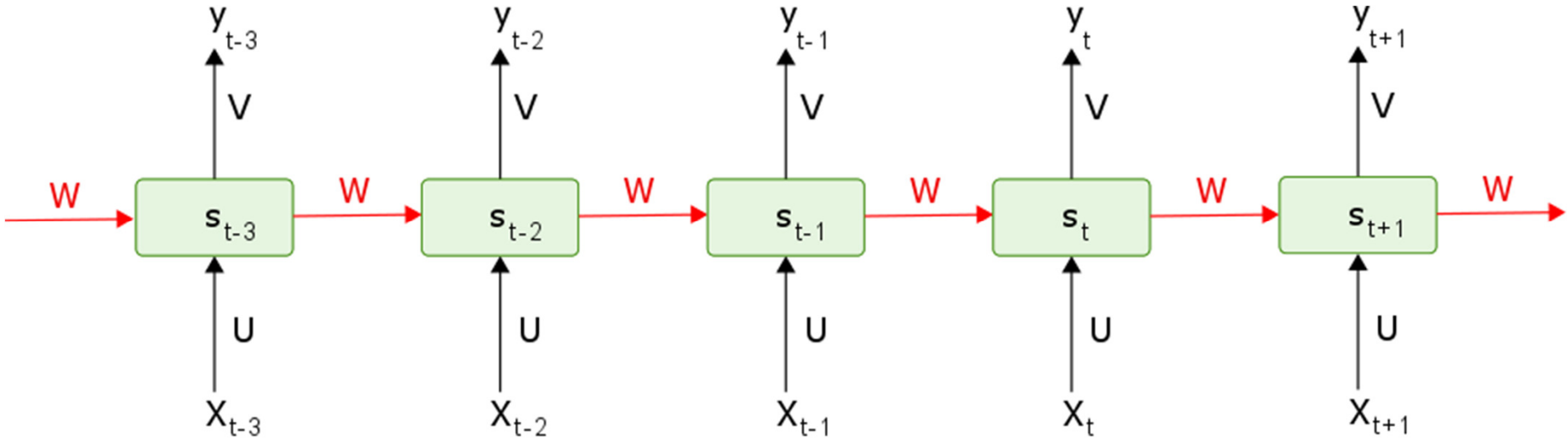


Megosztott súlyok az RNN-ben

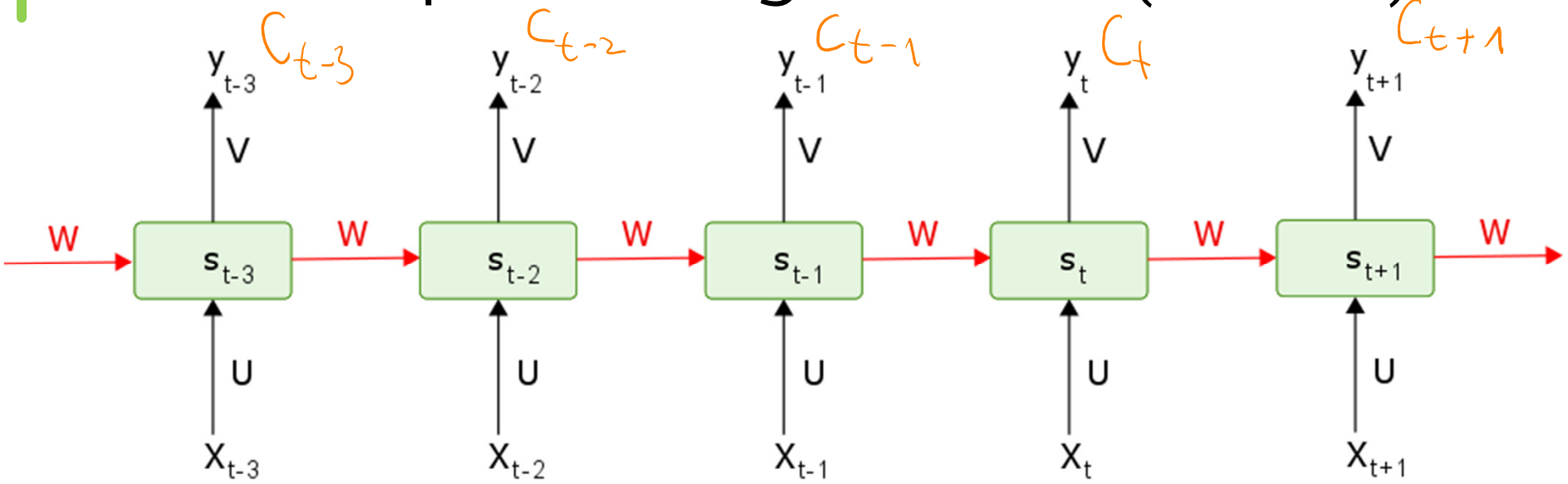


$$s_t = ReLU(Ux_t + Ws_{t-1}), \quad s_0 = \emptyset$$

$$y_t = softmax(Vs_t)$$



BackProp Through Time (BPTT) 1



$$C = -\sum_t y_t \log \hat{y}_t$$

$$\frac{\partial C}{\partial V} = \sum_t \frac{\partial C_t}{\partial V} = \sum_t \frac{\partial C_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial V} = \sum_t \frac{\partial C_t}{\partial \hat{y}_t}\cdot\frac{\partial \hat{y}_t}{\partial a_t}\cdot\frac{\partial a_t}{\partial V} =$$

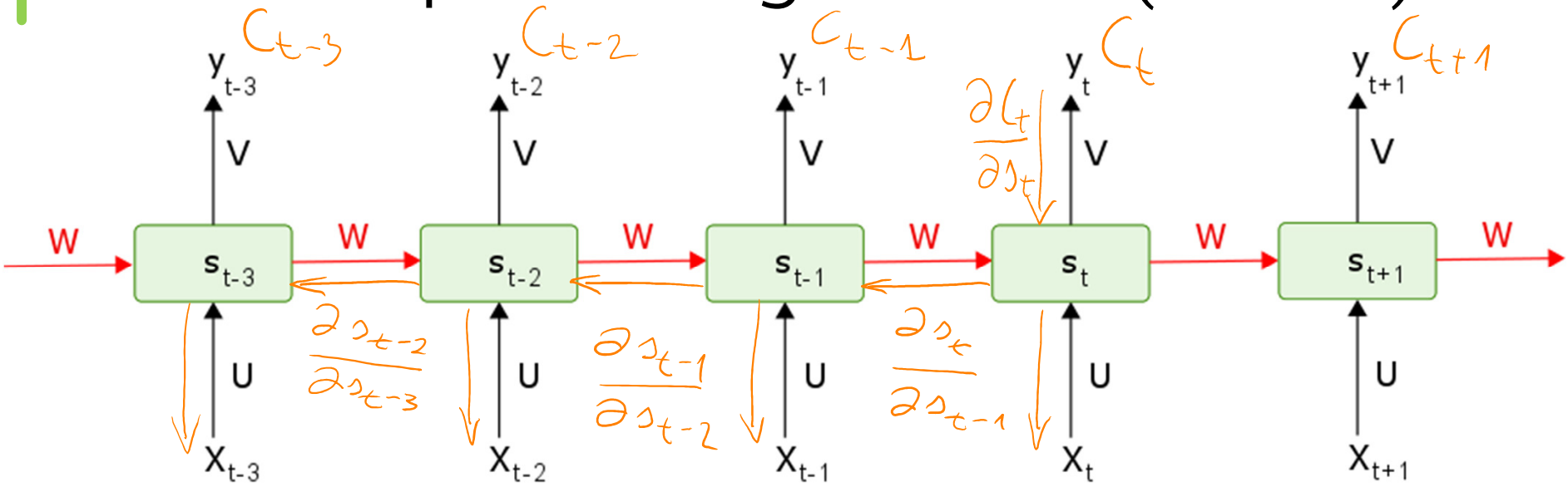
$$= \ldots = \underbrace{(\hat{y}_t - y_t)\cdot s_t}$$

CSAK t!

$$\hat{y}_t = \mathrm{softmax}(a_t)$$

$$a_t = V s_t$$

BackProp Through Time (BPTT) 2



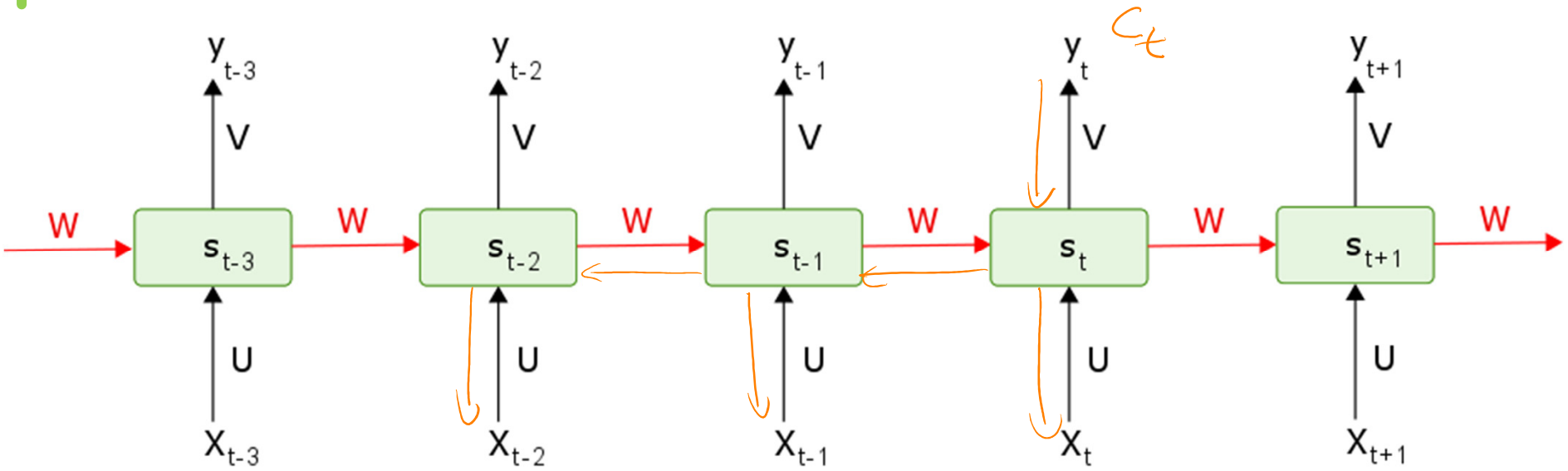
$$\frac{\partial C}{\partial W} = \sum_t \frac{\partial C_t}{\partial W} = \sum_t \frac{\partial C_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial W} =$$

$$= \sum_t \sum_k \frac{\partial C_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial s_k} \cdot \frac{\partial s_k}{\partial W}$$

$t = 0 \dots T$

$k = K \dots t$

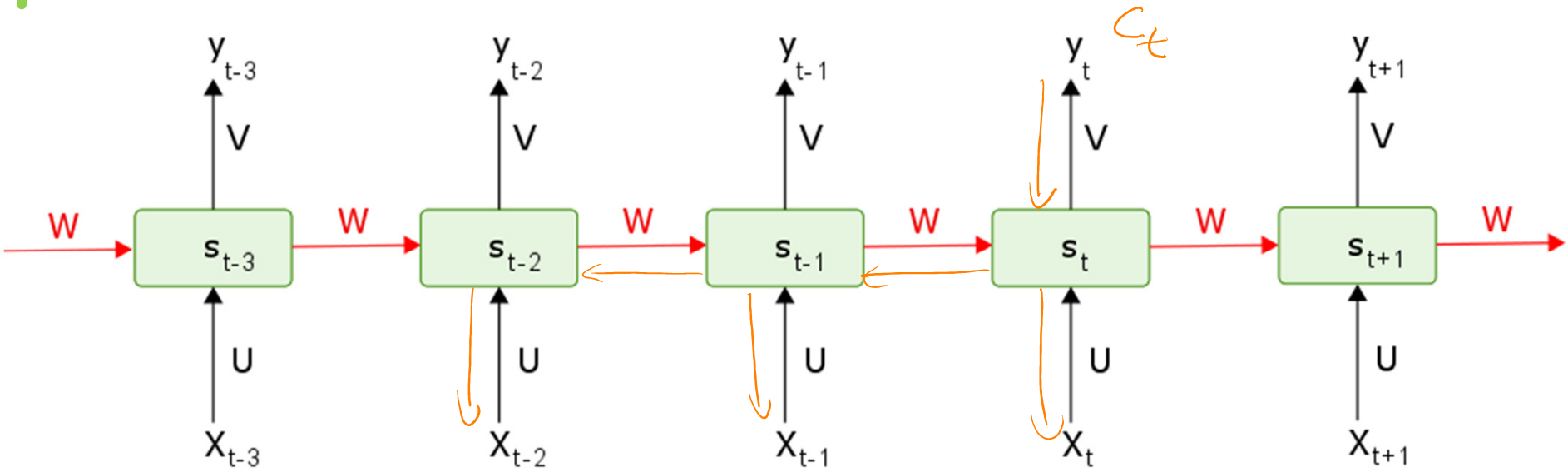
Truncated BPTT 1



$$\frac{\partial C_t}{\partial W} = \sum_{k=t-2}^{t} \frac{\partial C_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial s_k} \cdot \frac{\partial s_k}{\partial W} = \frac{\partial C_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \left(\frac{\partial s_t}{\partial s_{t-2}} \cdot \frac{\partial s_{t-2}}{\partial W} + \right.$$

$$\left. + \frac{\partial s_t}{\partial s_{t-1}} \cdot \frac{\partial s_{t-1}}{\partial W} \right) = \frac{\partial C_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \left(\frac{\partial s_t}{\partial s_{t-1}} \cdot \frac{\partial s_{t-1}}{\partial s_{t-2}} \cdot \frac{\partial s_{t-2}}{\partial W} + \frac{\partial s_t}{\partial s_{t-1}} \cdot \frac{\partial s_{t-1}}{\partial W} \right)$$

Truncated BPTT 2

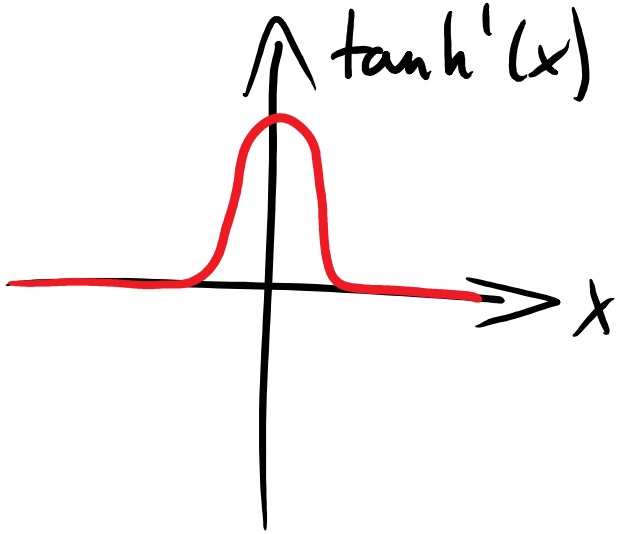


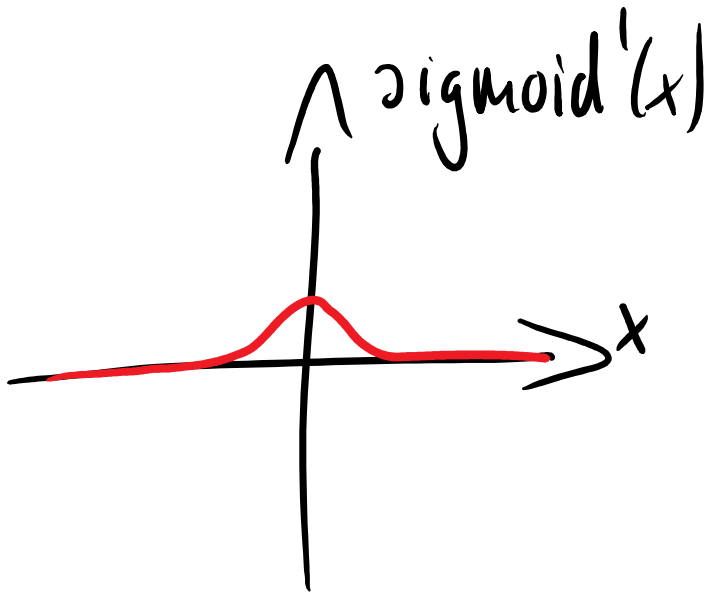
$$\frac{\partial C_t}{\partial w} = \frac{\partial C_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \left\{ \sum_{k=t-2}^{t} \left\{ \prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}} \right\} \cdot \frac{\partial s_k}{\partial w} \right\}$$

JACOBI-MÁTRIX: Vektor fv. első rendű parciális deriváltjait tartalmazó mátrix

Elenyésző (vanishing) gradiens

- Pascanu, Razvan; Mikolov, Tomas; Bengio, Yoshua. On the difficulty of training recurrent neural networks. *ICML (3)*, 2013, 28: 1310-1318.
- A cikk alapján: a 2-es normája a Jacobi-mátrixnak tanh/sigmoid aktiváció függvény esetén 1-ben korlátos.





Az aktivációs függvény deriváltja legtöbb helyen 0-t vagy nagyon kicsi értéket vesz fel → a gradients nullába tart, távolabbi időpillanatok *kinullázzák* a hozzájuk tartozó súlyokat, ami a megosztott súlyok miatt különösen gondot jelent.

Megoldás: ReLU használata (deriváltja 0 vagy 1), LSTM

Felrobbanó (exploding) gradiens

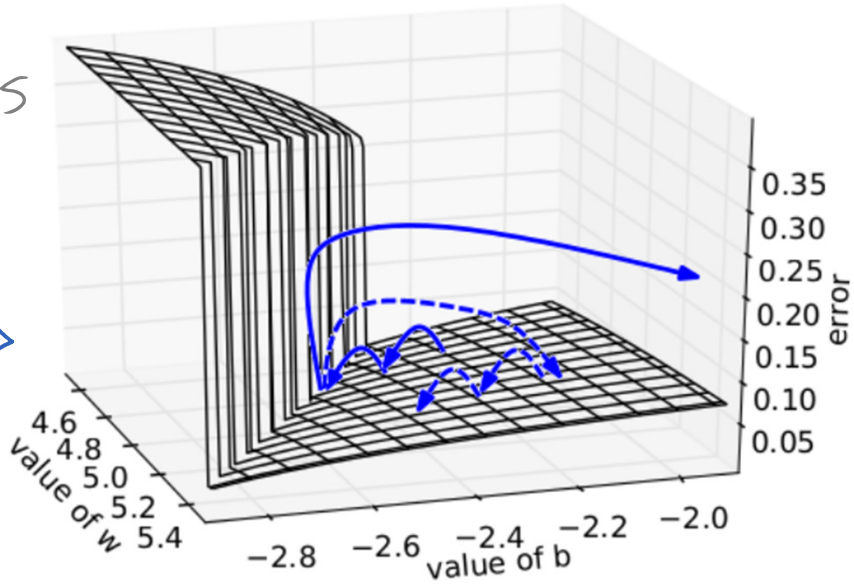
ReLU esetén 1-nél nagyobb értékek is lehetnek az egyes neuronok kimenetén, ami idővel túl nagy gradiens és így súlyértékekhez vezetnek.

Megoldás: gradiens vágás (gradient clipping)

$$s_t = w \cdot \mathrm{sgm}(s_{t-1}) + b \quad \leftarrow \text{BIAS}$$

$$E_{50} = (\mathrm{sgm}(s_{50}) - 0{,}7)^2 \Rightarrow$$

$$s_0 = 0{,}5$$



Ábra forrása: Pascanu et al., 2013

RNN implementáció

```
U = get_variable("W_z", [input_size, RNN_size])
W = get_variable("W_i", [RNN_size, RNN_size])
V = get_variable("W_f", [RNN_size, output_size])
b_U = get_variable("b_z", [1, RNN_size])
b_W = get_variable("b_i", [1, RNN_size])
b_V = get_variable("b_f", [1, output_size])

s = tf.relu(tf.mul(U,input)+tf.mul(W,s_prev))
y = tf.softmax(tf.mul(V,s))

s_t = activation("b_f", [1, output_size])
```

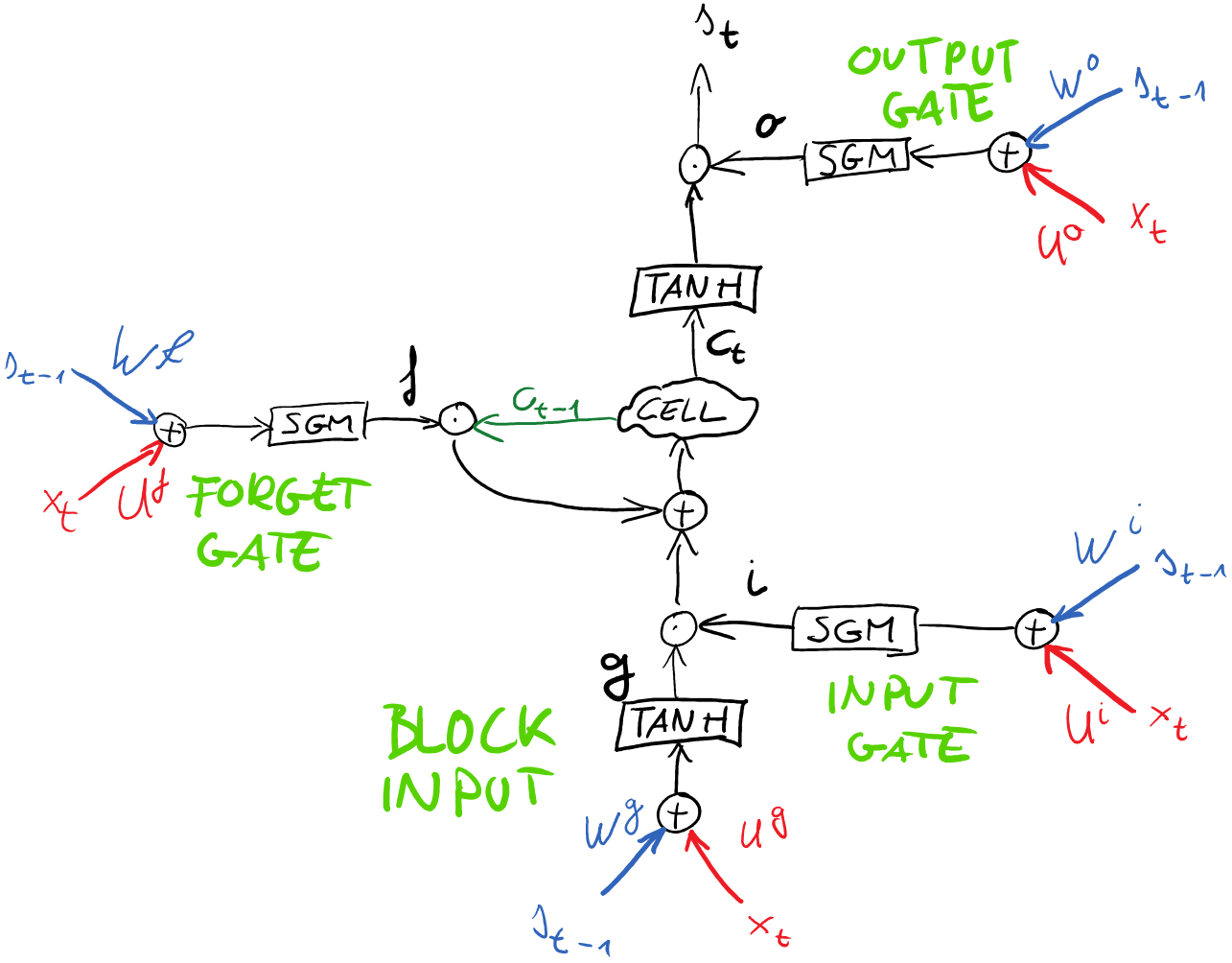
Long Short-Term Memory (LSTM)

Hochreiter, Sepp; Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 1997, 9.8: 1735-1780.

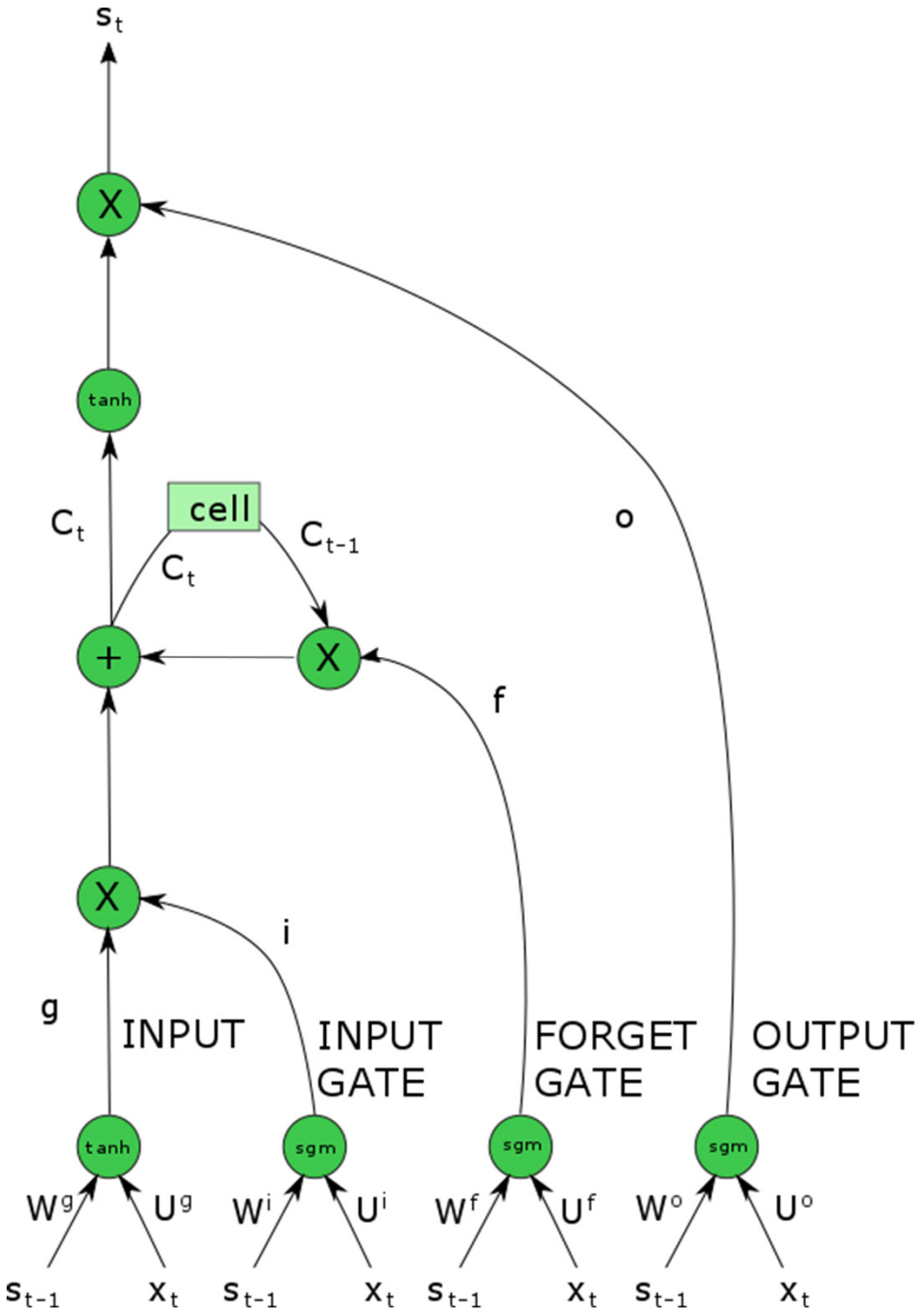
Deep learning korában újra fókuszban!

- Időben sokkal távolabbi események modellezésére képes, mint az alap RNN.
- Megoldás az elenyésző gradiens problémájára.
- Változó hosszú ki- és bemenetek. (Pl. mondatok)
- Jellemző optimizációs eljárás: RMSprop vagy ADAM.
- Új hiperparaméter: LSTM egységek száma („cells”).
- Ma ez az általánosan használt rekurrens architektúra.
- „Egyszerűbb” változat: Gated Recurrent Unit (GRU), 2014

LSTM



$$i = \mathrm{sgm}(x_t U^i + s_{t-1} W^i)$$
$$f = \mathrm{sgm}(x_t U^f + s_{t-1} W^f)$$
$$o = \mathrm{sgm}(x_t U^o + s_{t-1} W^o)$$
$$g = \tanh(x_t U^g + s_{t-1} W^g)$$
$$c_t = c_{t-1} \circ f + g \circ i$$
$$s_t = \tanh(c_t) \circ o$$



$$i = \mathrm{sgm}(x_t U^i + s_{t-1} W^i)$$

$$f = \mathrm{sgm}(x_t U^f + s_{t-1} W^f)$$

$$o = \mathrm{sgm}(x_t U^o + s_{t-1} W^o)$$

$$g = \tanh(x_t U^g + s_{t-1} W^g)$$

$$c_t = c_{t-1} \circ f + g \circ i$$

$$s_t = \tanh(c_t) \circ o$$

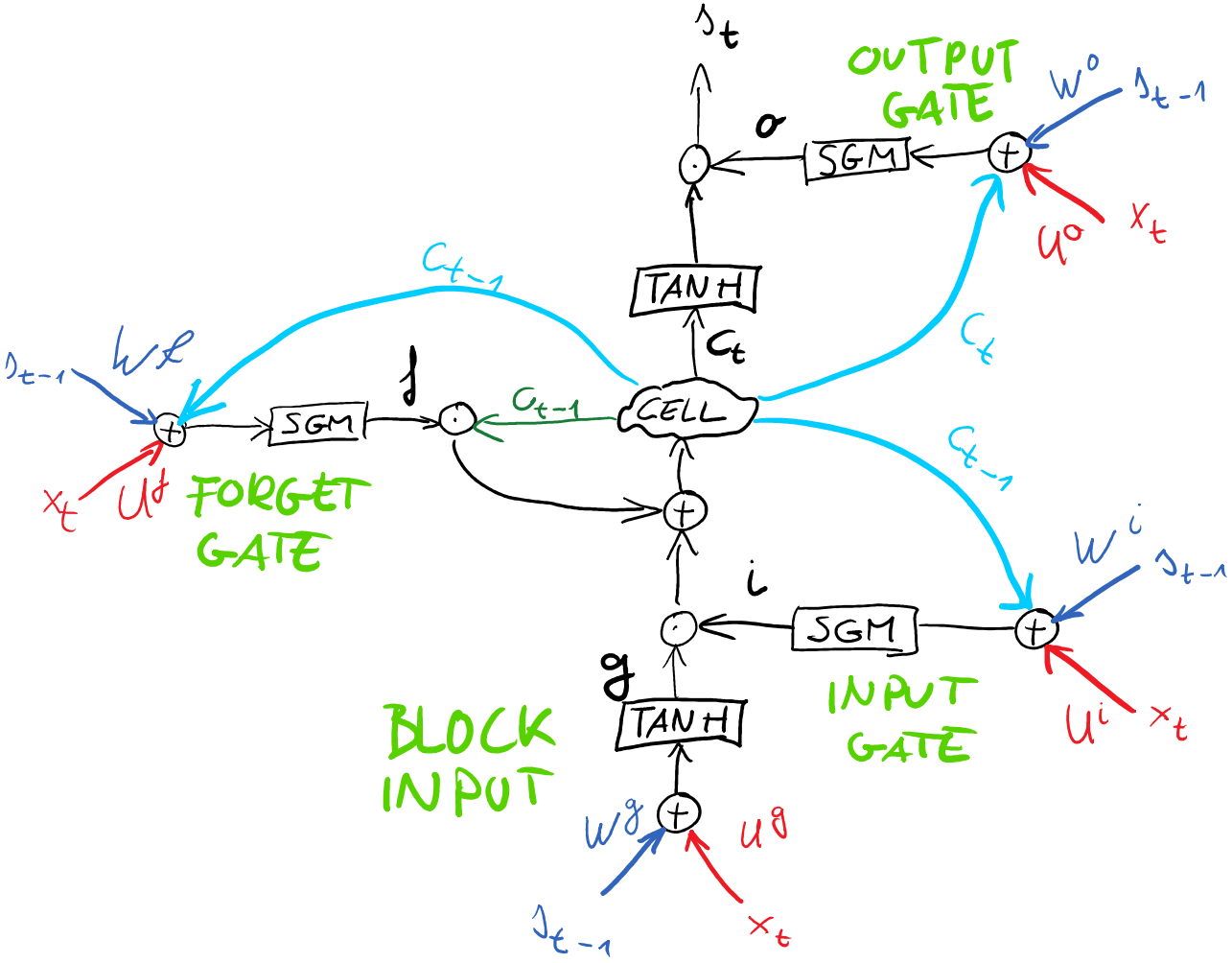
LSTM implementáció

```
W_g = get_variable("W_z", [input_size, LSTM_cells])
W_i = get_variable("W_i", [input_size, LSTM_cells])
W_f = get_variable("W_f", [input_size, LSTM_cells])
W_o = get_variable("W_o", [input_size, LSTM_cells])
U_g = get_variable("U_z", [LSTM_cells, LSTM_cells])
U_i = get_variable("U_i", [LSTM_cells, LSTM_cells])
U_f = get_variable("U_f", [LSTM_cells, LSTM_cells])
U_o = get_variable("U_o", [LSTM_cells, LSTM_cells])
b_g = get_variable("b_z", [1, LSTM_cells])
b_i = get_variable("b_i", [1, LSTM_cells])
b_f = get_variable("b_f", [1, LSTM_cells])
b_o = get_variable("b_o", [1, LSTM_cells])
```

LSTM + peephole implementáció

```
g = tf.tanh(tf.matmul(inputs, W_g) +
    tf.matmul(s_prev, U_g) + b_g)
i = tf.sigmoid(tf.matmul(inputs, W_i) +
    tf.matmul(s_prev, U_i) + b_i)
f = tf.sigmoid(tf.matmul(inputs, W_f) +
    tf.matmul(s_prev, U_f) b_f)
c = tf.mul(i, z) + tf.mul(f, c_prev)
o = tf.sigmoid(tf.matmul(inputs, W_o) +
    tf.matmul(s_prev, U_o) + b_o)
s = tf.mul(tf.tanh(c), o)
```

LSTM + peephole



$$i = \mathrm{sgm}(x_t U^i + s_{t-1} W^i + P^i \odot c_{t-1})$$

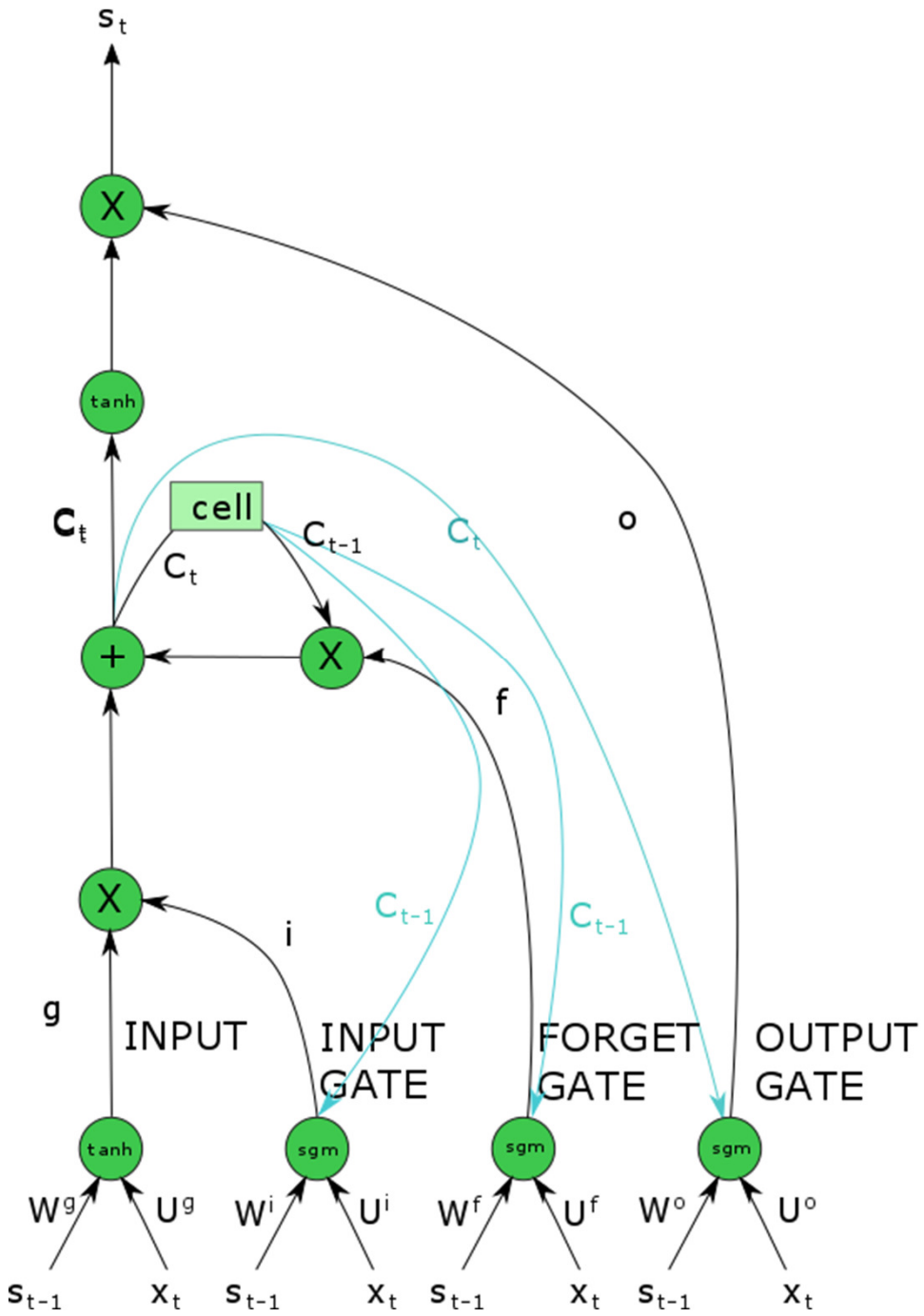
$$f = \mathrm{sgm}(x_t U^f + s_{t-1} W^f + P^f \odot c_{t-1})$$

$$o = \mathrm{sgm}(x_t U^o + s_{t-1} W^o + P^o \odot c_t)$$

$$g = \tanh(x_t U^g + s_{t-1} W^g)$$

$$c_t = c_{t-1} \odot f + g \odot i$$

$$s_t = \tanh(c_t) \odot o$$



$$i = \mathrm{sgm}(x_t U^i + s_{t-1} W^i + P^i \odot C_{t-1})$$

$$f = \mathrm{sgm}(x_t U^f + s_{t-1} W^f + P^f \odot C_{t-1})$$

$$o = \mathrm{sgm}(x_t U^o + s_{t-1} W^o + P^o \odot C_t)$$

$$g = \tanh(x_t U^g + s_{t-1} W^g)$$

$$C_t = C_{t-1} \odot f + g \odot i$$

$$s_t = \tanh(C_t) \odot o$$

Névnek a NEPTUN kódodat add meg!

https://kahoot.it/

LSTM + peephole implementáció

```
W_g = get_variable("W_z", [input_size, LSTM_cells])
W_i = get_variable("W_i", [input_size, LSTM_cells])
W_f = get_variable("W_f", [input_size, LSTM_cells])
W_o = get_variable("W_o", [input_size, LSTM_cells])
U_g = get_variable("U_z", [LSTM_cells, LSTM_cells])
U_i = get_variable("U_i", [LSTM_cells, LSTM_cells])
U_f = get_variable("U_f", [LSTM_cells, LSTM_cells])
U_o = get_variable("U_o", [LSTM_cells, LSTM_cells])
b_g = get_variable("b_z", [1, LSTM_cells])
b_i = get_variable("b_i", [1, LSTM_cells])
b_f = get_variable("b_f", [1, LSTM_cells])
b_o = get_variable("b_o", [1, LSTM_cells])
p_i = get_variable("p_i", [LSTM_cells])
p_f = get_variable("p_f", [LSTM_cells])
p_o = get_variable("p_o", [LSTM_cells])
```

LSTM + peephole implementáció

```
g = tf.tanh(tf.matmul(inputs, W_g) +
     tf.matmul(s_prev, U_g) + b_g)
i = tf.sigmoid(tf.matmul(inputs, W_i) +
     tf.matmul(s_prev, U_i) +
     tf.mul(c_prev, p_i) + b_i)
f = tf.sigmoid(tf.matmul(inputs, W_f) +
     tf.matmul(s_prev, U_f) +
     tf.mul(c_prev, p_f) + b_f)
c = tf.mul(i, z) + tf.mul(f, c_prev)
o = tf.sigmoid(tf.matmul(inputs, W_o) +
     tf.matmul(s_prev, U_o) +
     tf.mul(c,p_o) + b_o)
s = tf.mul(tf.tanh(c), o)
```

LSTM további információk

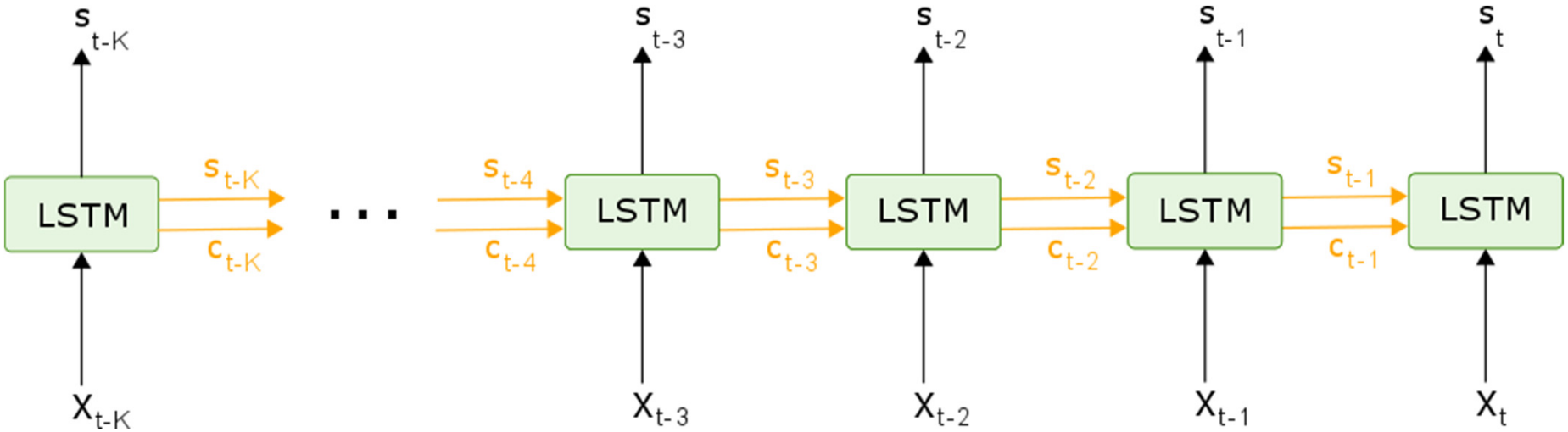
Nagyon ajánlott olvasmány:

- https://iamtrask.github.io/2015/11/15/anyone-can-code-lstm/

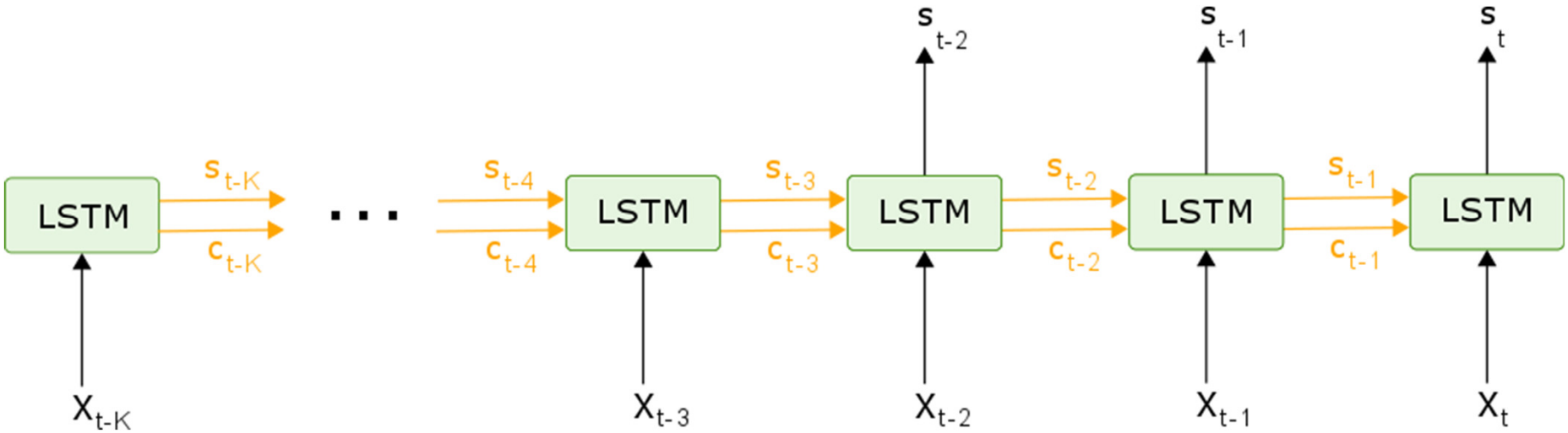
Kötelező olvasmány:

- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2015). LSTM: A search space odyssey. *arXiv preprint arXiv:1503.04069*.

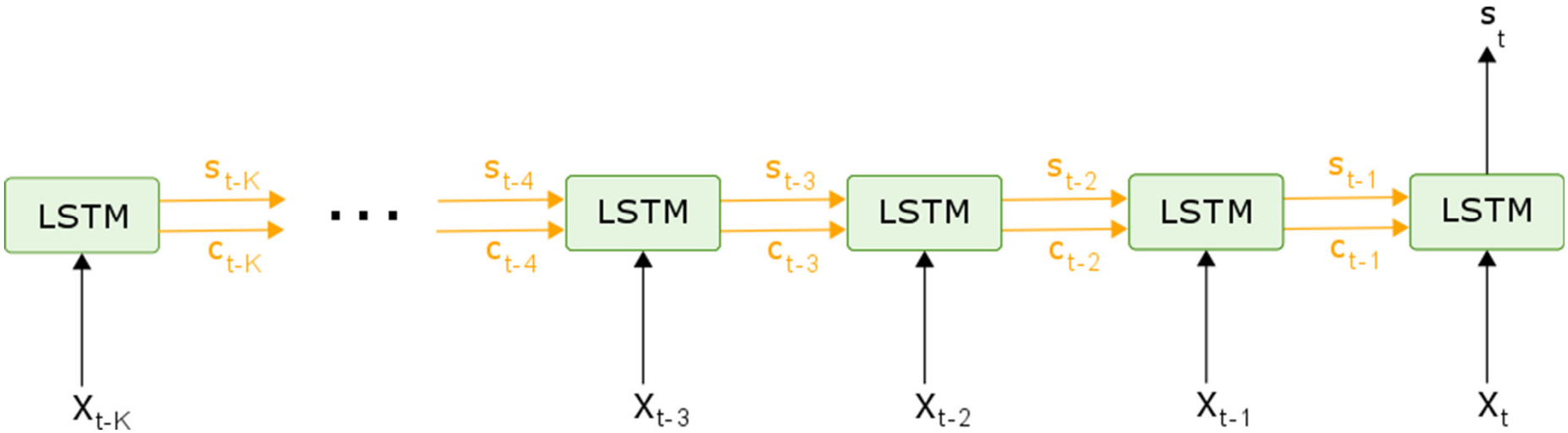
Egyirányú LSTM 1



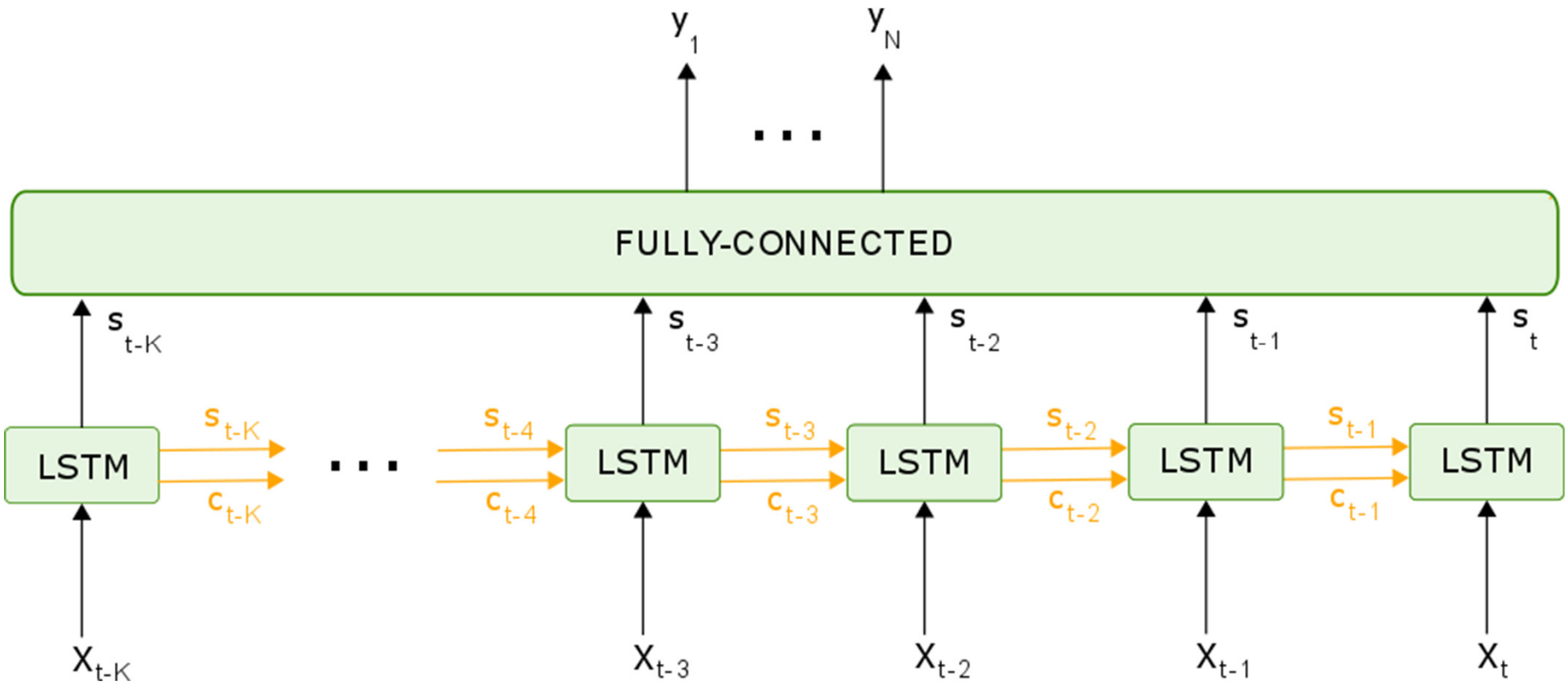
Egyirányú LSTM 2



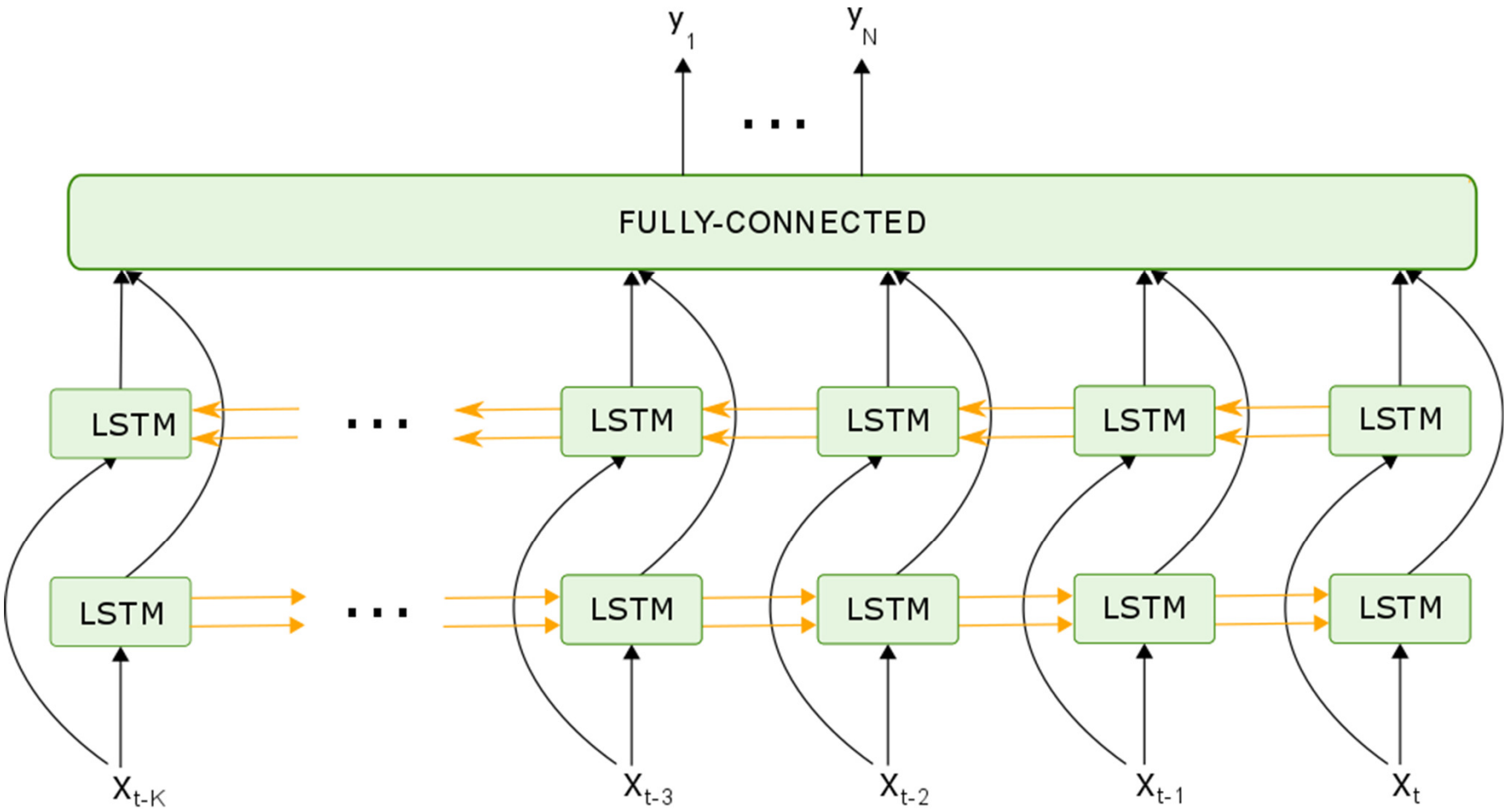
Egyirányú LSTM 3



Egyirányú LSTM + FC



Kétirányú LSTM (Bidirectional LSTM, BLSTM)



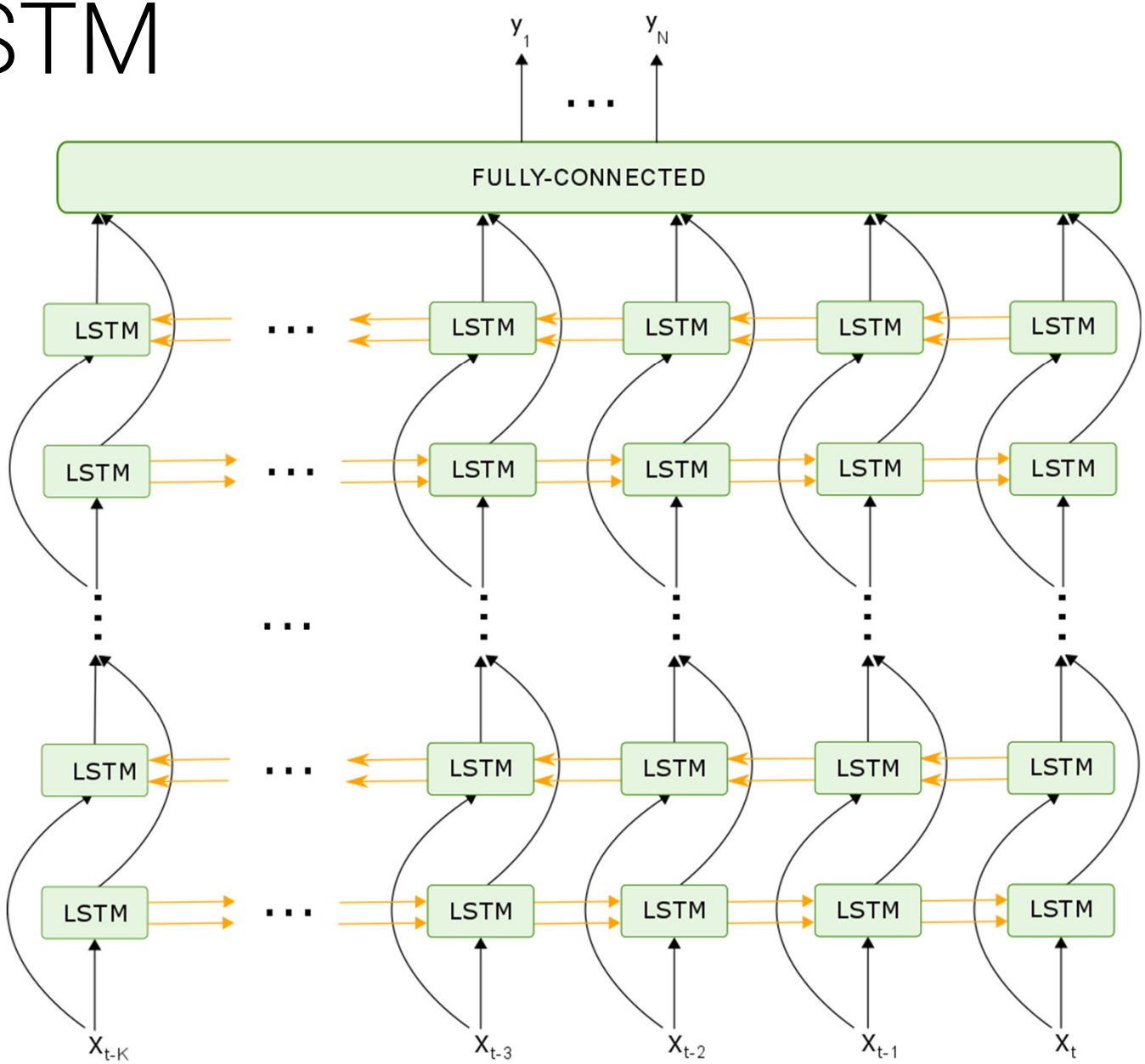
Minden LSTM réteghez külön osztott súlyok.

Mély BLSTM

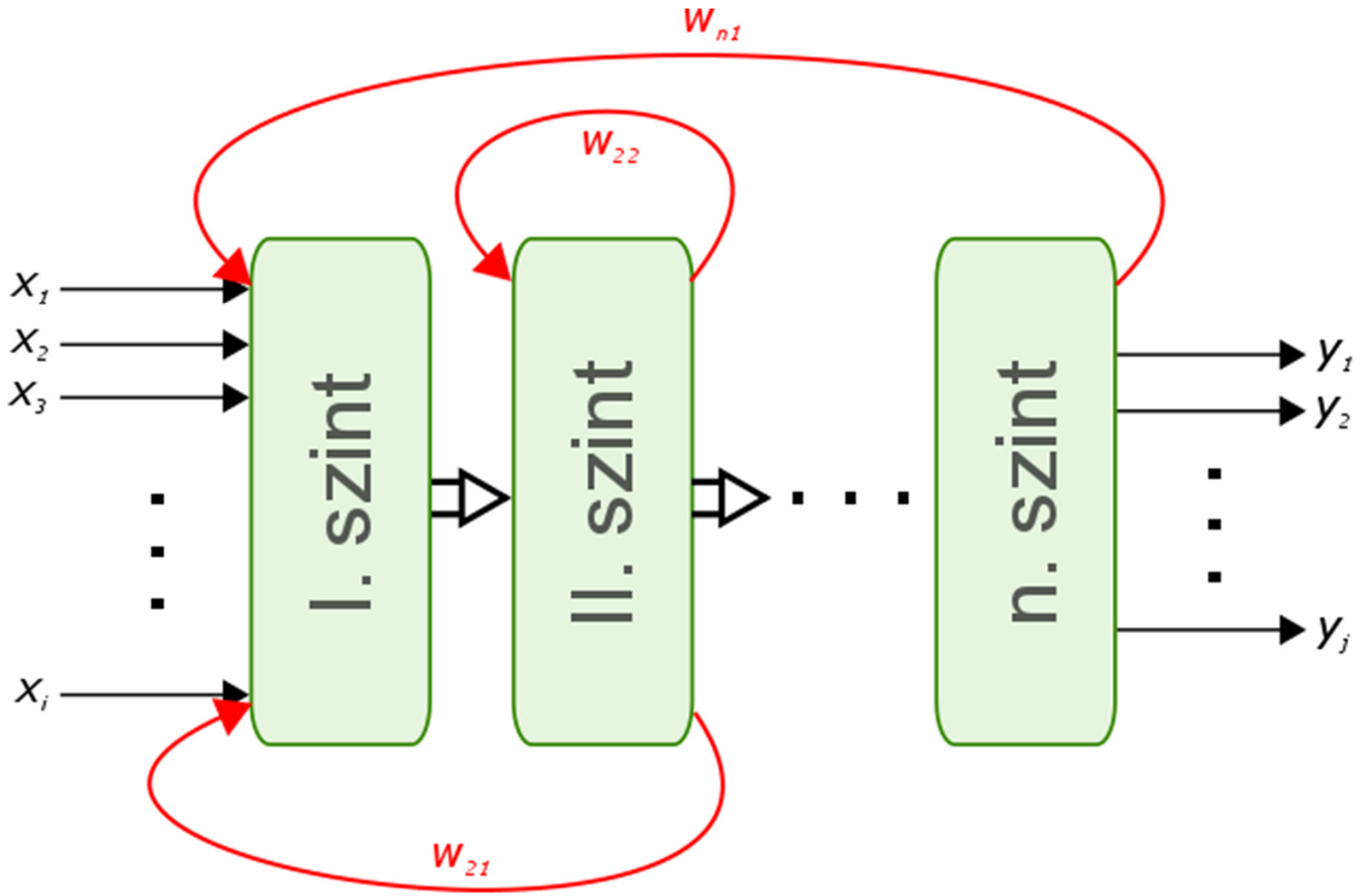
Minden LSTM réteghez külön osztott súlyok.

Hierarchikus adat modellezés

A sima LSTM/BLSTM elsősorban időbeli modellezést hajt végre!



Többszintű RNN/LSTM



RNN alkalmazások

- Idősorok modellezése
- Zajszűrés („azonos” ki és bemenet)

- Képek feliratozása tartalmuk alapján
- Videó tartalmának szöveges kivonata
- Kézírás felismerés és generálás

- Gépi fordítás
- Natural Language Processing (NLP)
- „Chat-bot”-ok
- Kérdezz-felelek
- Szentiment elemzés

LSTM implementáció TensorFlow

```
# n_hidden adja meg, hogy hány LSTM-et használunk
lstm_cell = rnn_cell.BasicLSTMCell(n_hidden,
            forget_bias=1.0)
# majd létrehozzuk az egyirányú LSTM-et, ehhez a bemenet
      formája:
# 'n_steps' db tensor egy listában, tensor alakja:
      (batch_size, n_input)
outputs, states = rnn.rnn(lstm_cell, x, dtype=tf.float32)
# az LSTM utolsó kimenetét egy lineáris rétegbe kötjük be
pred=tf.matmul(outputs[-1], weights['out'])
      + biases['out']
```

LSTM implementáció TensorFlow

```
# szokásos részek...
cost = tf.reduce_mean(
      tf.nn.softmax_cross_entropy_with_logits(pred, y))
optimizer = tf.train.AdamOptimizer(
      learning_rate=learning_rate).minimize(cost)
init = tf.initialize_all_variables()

with tf.Session() as sess:
    sess.run(init)
      # ... batch-ek összerakása, majd:
    sess.run(optimizer,
            feed_dict={x: batch_x, y: batch_y})
```

LSTM megvalósítás Keras

```
# adat bemenet: (nb_samples, timesteps, input_dim)
model = Sequential()
model.add(LSTM(128, input_shape=(timesteps,
      input_dim)))
model.add(Dense(len(chars)))
model.add(Activation('softmax'))

optimizer = RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy',
      optimizer=optimizer)
```

BLSTM megvalósítás TensorFlow

```
# külön definiálunk az előre és hátra irányt
lstm_fw_cell = rnn_cell.BasicLSTMCell(n_hidden,
                    forget_bias=1.0)
lstm_bw_cell = rnn_cell.BasicLSTMCell(n_hidden,
                    forget_bias=1.0)
outputs, _, _ = rnn.bidirectional_rnn(
                    lstm_fw_cell, lstm_bw_cell, x,
                    dtype=tf.float32)
pred=tf.matmul(outputs[-1], weights['out'])
                    + biases['out']
```

További paraméterek

- Return sequences
- Time Distributed
- Stateful LSTM

LSTM regularizáció

- Súly regularizáció
  - Bias
  - Bemenet és kimenet súlymátrix
  - LSTM cell súlymátrix
- DropOut
  - Bemenet és kimenet
  - LSTM cell

  Keras: LSTM(...., dropout=..., recurrent_dropout=...)
- ZoneOUt
  - Nem eldobja, hanem megtartja több korábbi állapothoz a kapcsolatokat.

Adatok újraformázása 1.

Egy paraméterfolyam esetén mini-batch összeállítása:

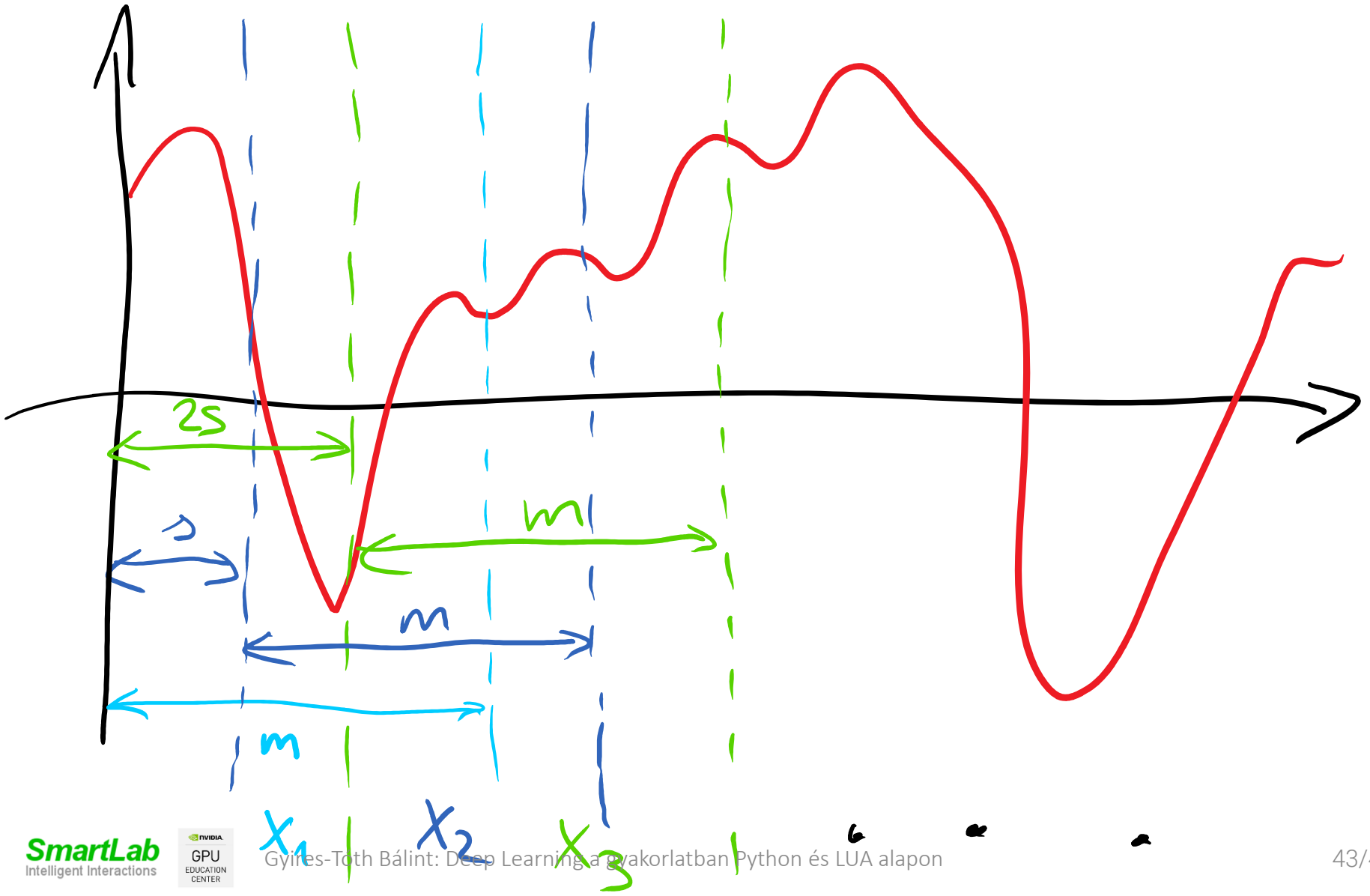
batch (b) x timesteps (m) x 1

s: lépésköz két minta között

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	.	.	.

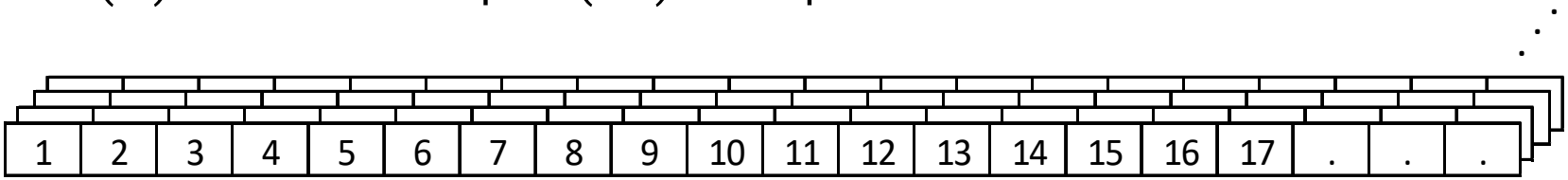
1	2	3	.	.	.	m
s	s+1	s+2	.	.	.	s+m
2s	2s+1	2s+2	.	.	.	2s+m
.	.	.	.	.	.	.
.	.	.	.	.	.	.
(b-1)*s	.	.	.	.	.	(b-1)*s+m

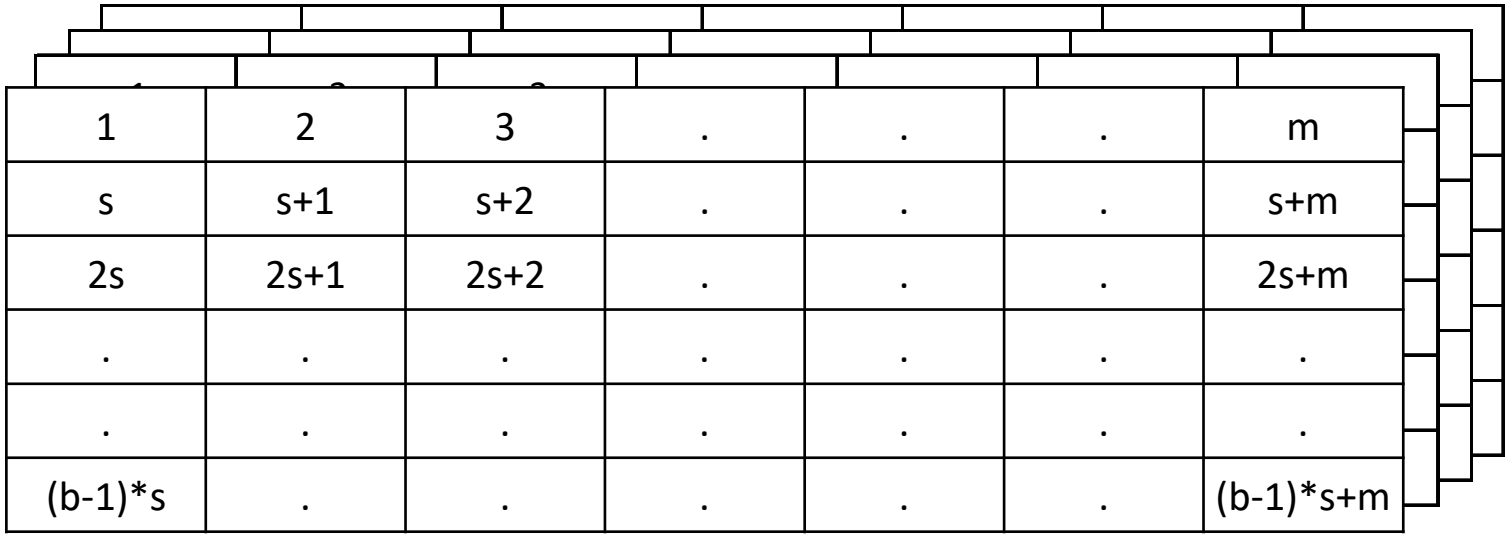
Ábrázolás



Adatok újraformázása 2.

Több paraméterfolyam esetén mini-batch összeállítása:
batch (b) x timesteps (m) x input_dim





4. kis házi feladat

A cél egy előtanított CNN háló tovább tanítása:

- Válassz ki két tetszőleges kategóriát a Google Open Image Dataset-ből (GOID: https://storage.googleapis.com/openimages/web/index.html) és tölts le 600-600 képet kategóriánként.
- Tölts be tetszőleges előtanított hálót (pl. Inception v3, VGG, ResNet) és tanítsd tovább (transfer learning) a kiválasztott két kategóriával:
  - Előbb csak a végső fullyconnectedrétegek tanuljanak.
  - Majd az így tovább tanított hálót tanítsátok még tovább úgy, hogy a konvolúciós rétegek felsőbb rétegei is tanulnak (az alsóbbak nem).
- Egy elkülönített teszt adatbázison, a kiválasztott két kategóriával értékeld ki a megoldás hatékonyságát.

4. kis házi feladat

Figyelj arra, hogy

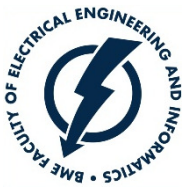
- a kép letöltés módszerét, scriptjét (ha saját kód) is mellékeld,
- a tanító adatbázisban 400-400, a validációsban 100-100, a tesztben 100-100 kép legyen,
- nagyjából kiegyenlített mennyiségűek legyenek az egyes kategóriákban a képek száma,
- azonos előfeldolgozó eljárást használj, mint amit az előtanított hálónál alkalmaztak.

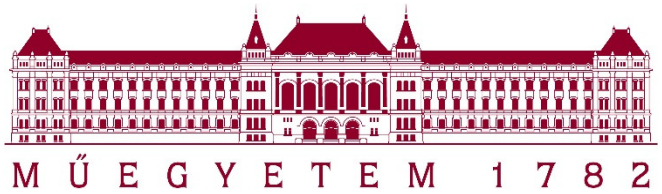
Tetszőleges deep learning keretrendszert használhatsz. A megoldást részletesen kommentezd és a tanítás kimenetét (log fájlját) is mellékeld. A leírás és kommentek angolul legyenek!

Segítségként javasoljuk az alábbi forrás felhasználását: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Beadási határido: 2020. november 10. 23:59

Beadás: http://smartlab.tmit.bme.hu/oktatas-deep-learning#kishazi





Köszönöm a figyelmet!

toth.b@tmit.bme.hu

Jelen kutatás az Emberi Erőforrások Minisztériuma által az Új Nemzeti Kiválóság Program keretében meghírdetett Bolyai+ Felsőoktatási Fiatal Oktatói, Kutatói Ösztöndíj támogatásával készült (ÚNKP-20-5-BME-210).

SmartLab
Intelligent Interactions

http://smartlab.tmit.bme.hu

NVIDIA
GPU
EDUCATION
CENTER