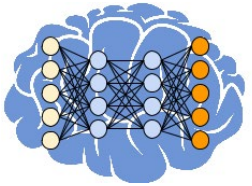


RWTH AACHEN UNIVERSITY

Recurrent Neural Networks: Short Introduction

ErUM-Data Community Meeting 01.07.21

Uwe Klemradt

RWTH Aachen

Starting point: standard network

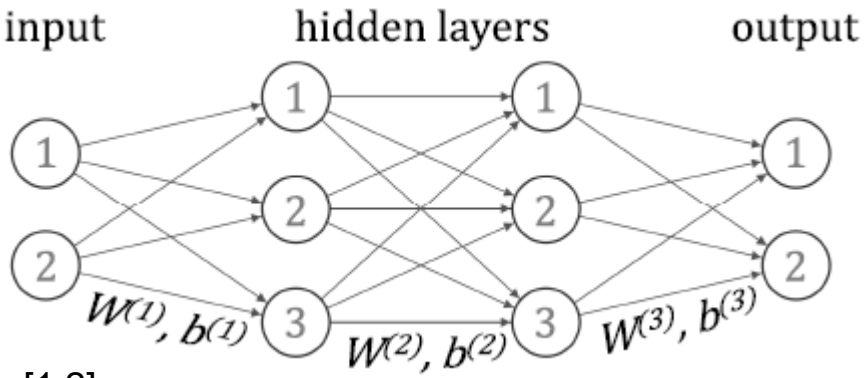
- Fully-connected networks:

  matrix multiplication between layers / bias vectors / nonlinear activation functions

  weights and biases are fixed by training

- Input data are processed from left to right (feedforward processing)

  network ready for new input after each output, results are not kept



[1,2]

Sequential data and time series

- Many forms of ordered data: time series, words, phrases, nucleobases in DNA…
- Time series important in physics: sensor readings (cf. data streams at CERN)

  everyday life: regularly updated weather forecasts

  financial data streams

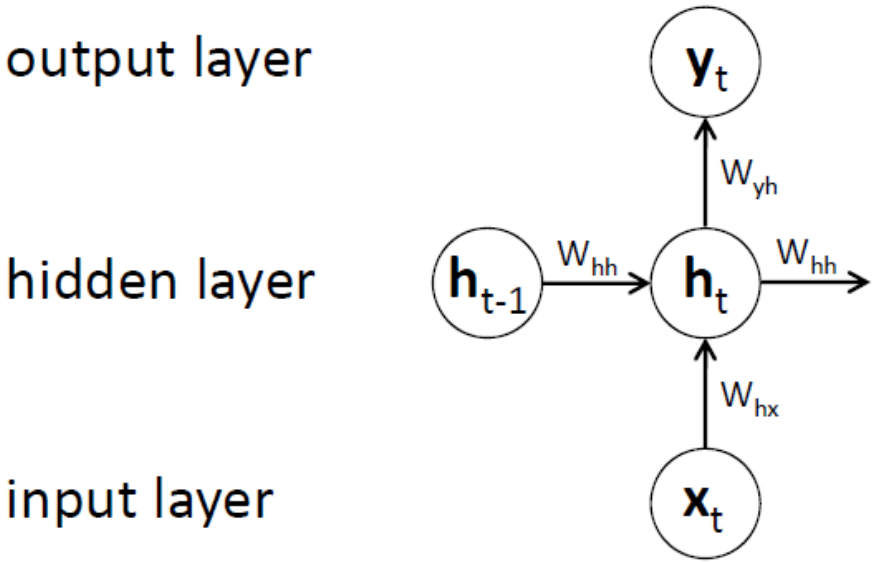
  audio + video streams

  voice assistants

  → time series are paradigm for physics
- Typical feature: variable input (= length of series not fixed, can vary during use)

No fixed input vector - standard matrix multiplication not well suited → New network design required when dealing with sequences: recurrent neural network

Basic unit of a recurrent network (RNN)



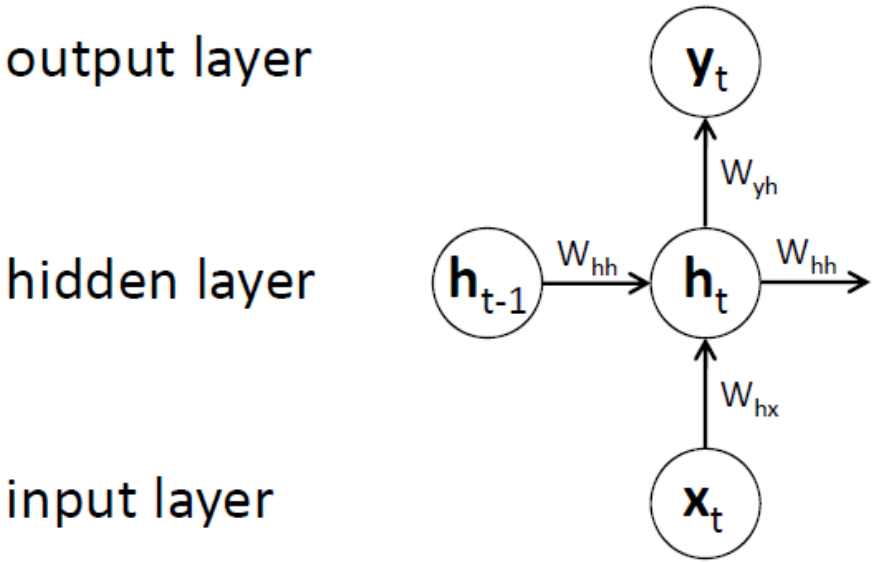
[2]

Output (e.g., forecast of the temperature at time step t+1)

Internal memory (to take into account previous input)

Input (e.g., reading of a temperature sensor at time step t)

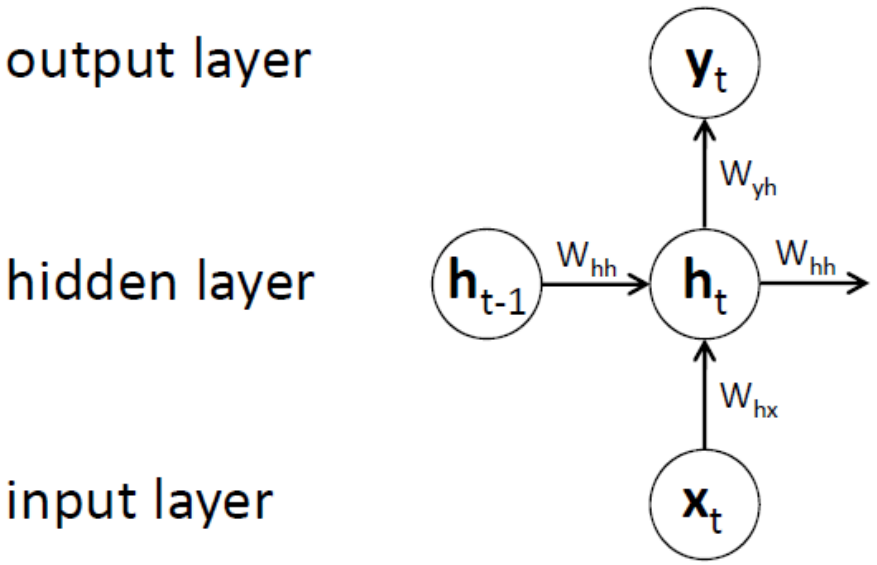
Basic unit of a recurrent network (RNN)



Direction of data processing from input to output

Timeline
(processing of previous input kept in internal memory)

Basic unit of a recurrent network (RNN)



Formal calculation:

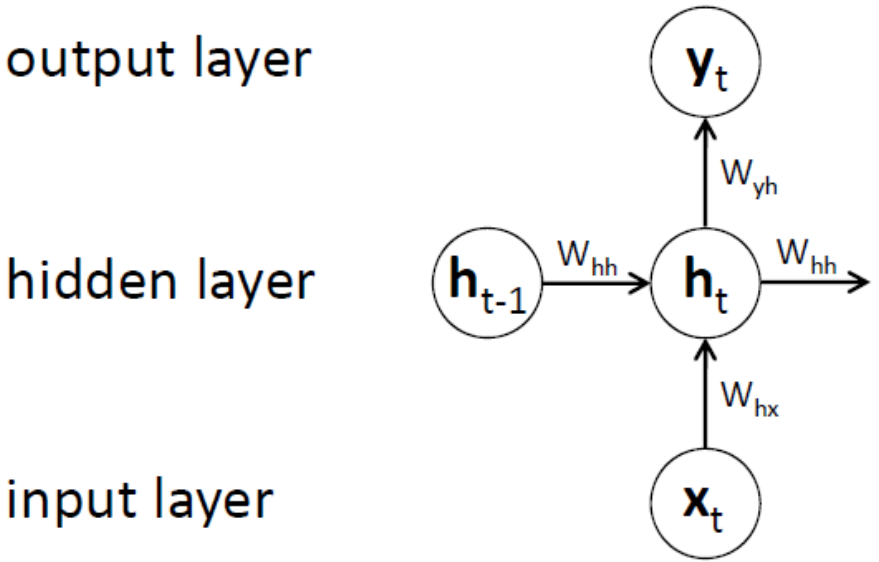
$$\vec{y}_t = g(\vec{h}_t)$$

$$\vec{h}_t = f(\vec{x}_t, \vec{h}_{t-1})$$

- Functions *f, g,* remain the same for the entire series
- Internal state at time t depends on *all* previous inputs:

$$\begin{aligned}\vec{h}_t &= f(\vec{x}_t, \vec{h}_{t-1}) \\ &= f(\vec{x}_t, f(\vec{x}_{t-1}, \vec{h}_{t-2})) \\ &= u(\vec{x}_t, \vec{x}_{t-1}, \vec{x}_{t-2}, \ldots)\end{aligned}$$

Basic unit of a recurrent network (RNN)

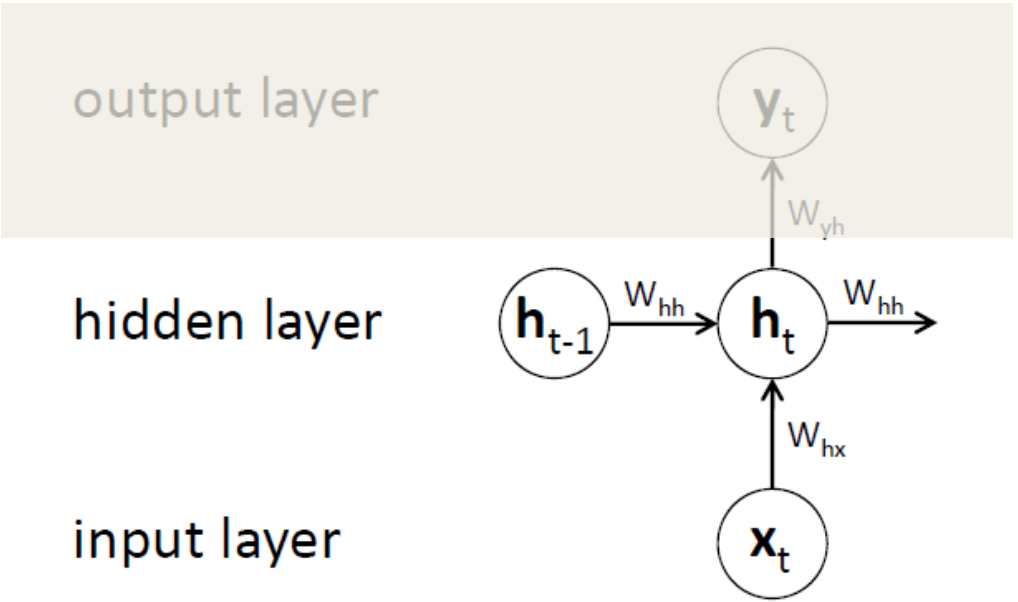


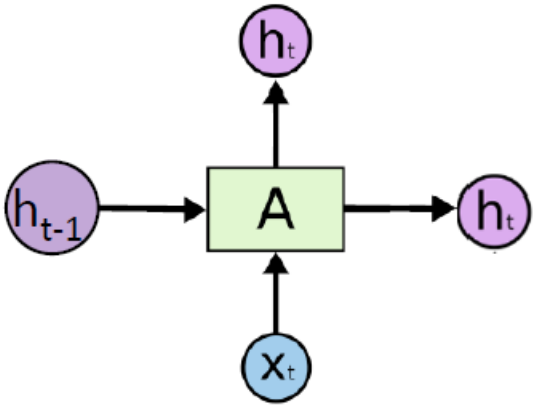
Example:

$$\vec{y}_t = \sigma(\mathbf{W}_{\mathrm{yh}}\,\vec{h}_t + \vec{b}_y)$$

$$\vec{h}_t = \tanh\left(\mathbf{W}_{\mathrm{hx}}\,\vec{x}_t + \mathbf{W}_{\mathrm{hh}}\,\vec{h}_{t-1} + \vec{b}_{\mathrm{h}}\right)$$

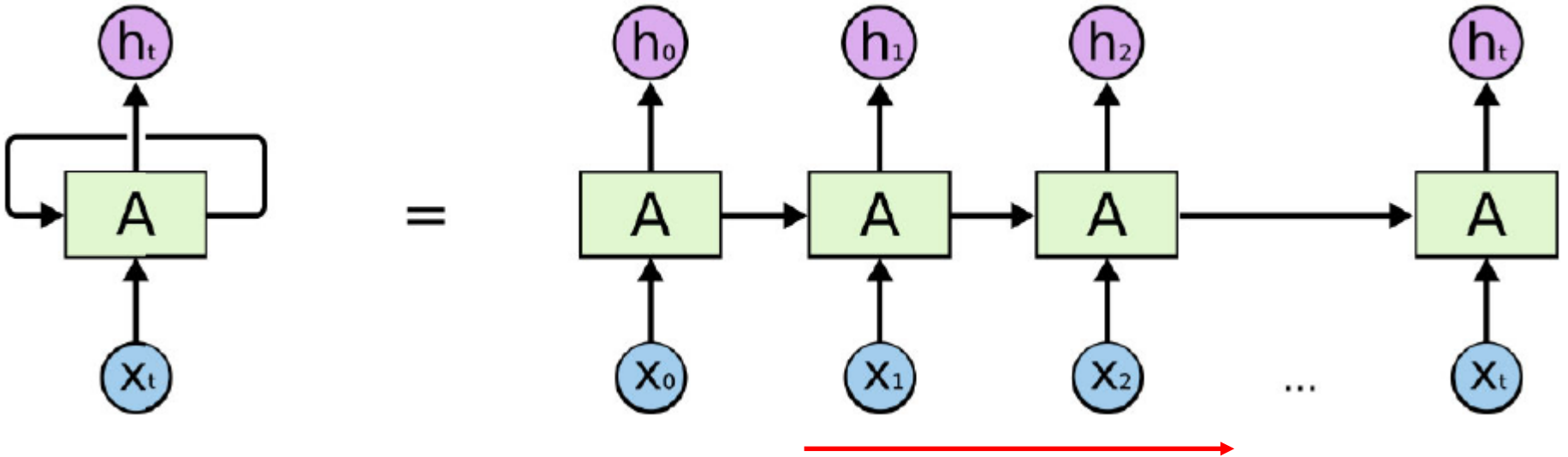
Core building block of a RNN





[2,3]

Unrolling of a RNN



[3]

- Unrolled RNN with similar appearance as conventional feedforward network
- Training through backpropagation through time (BPTT) = usual backpropagation process for weights and biases adapted to RNN

Problem: short-term memory good, but long-term memory fails

Long short-term memory (LSTM)

- Introduced in 1997 by Hochreiter und Schmidhuber
- Eliminates sensitivity to gap length between important events of a series
  $\rightarrow$ training becomes possible for much more useful sequence lengths
- Game changer for RNNs: applicable in real-world situations
- Until today “gold standard“ for recurrent networks

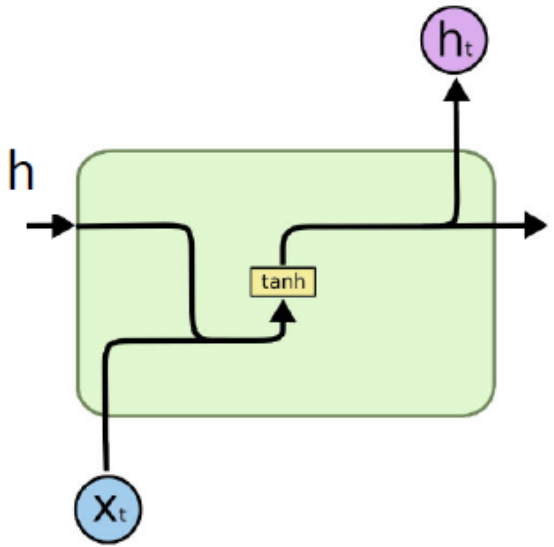
Key features:
- Additional internal memory called cell state
- Gates that control the cell state actively
- Closely controlled update procedure of the internal memories

Gate overview:
- *Forget gate (= keep gate):* decides whether to forget or keep elements stored in the cell
- *Input gate:* decides whether a new value flows into the cell
- *Output gate:* controls whether the updated cell value contributes to the hidden state

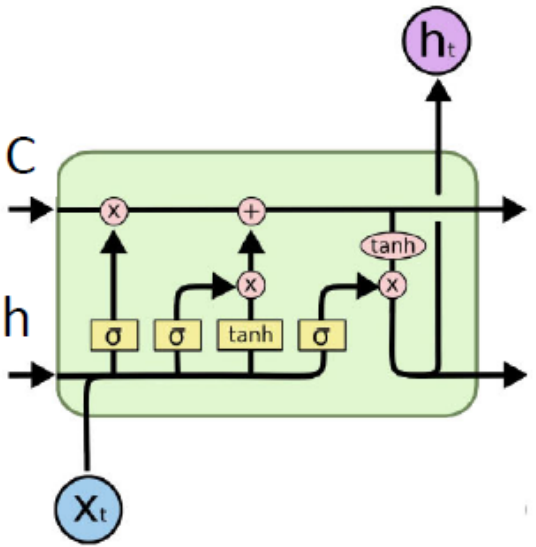
Long short-term memory (LSTM)

Graphical representation of the update process:

Standard RNN cell



LSTM cell



[3]

$$\vec{h}_t = \tanh\left(\mathbf{W}_{\mathrm{hx}}\,\vec{x}_t + \mathbf{W}_{\mathrm{hh}}\,\vec{h}_{t-1} + \vec{b}_{\mathrm{h}}\right)$$

Gated recurrent unit (GRU)

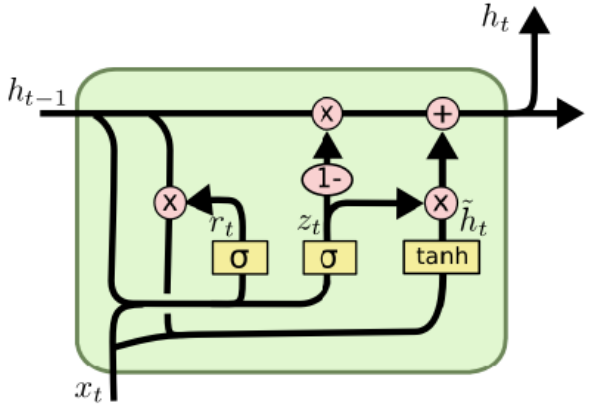
- Introduced in 2014 by Cho et al.
- Aim: simplification of the complicated LSTM update procedure
- Growing popularity in applications, needs less resources

Key features:
- No cell state
- Only two gates

Gate overview:
- *Reset gate:* decides whether to reset the hidden state
- *Update gate:* decides whether to update the hidden state

Gated recurrent unit (GRU)

GRU hidden state update



[3]

$$\text{reset layer}: \quad r_t = \sigma\left(W_r\, x_t + U_r\, h_{t-1} + b_r\right)$$

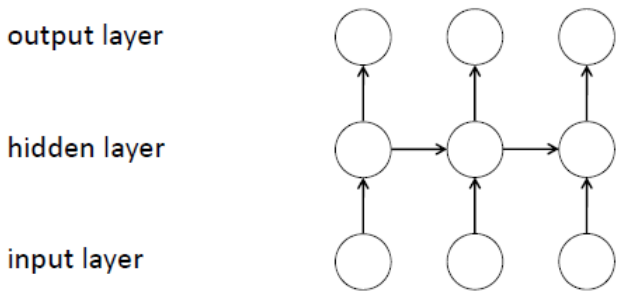
$$\text{update layer}: \quad z_t = \sigma\left(W_z\, x_t + U_z\, h_{t-1} + b_z\right)$$

$$\text{candidate layer}: \quad \tilde{h}_t = \tanh\left(W_h\, x_t + U_h\left(r_t \cdot h_{t-1}\right) + b_h\right)$$

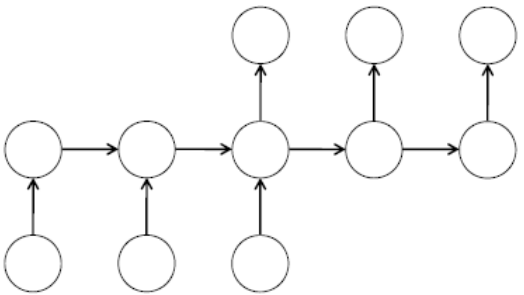
$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t$$

Applications in current technology

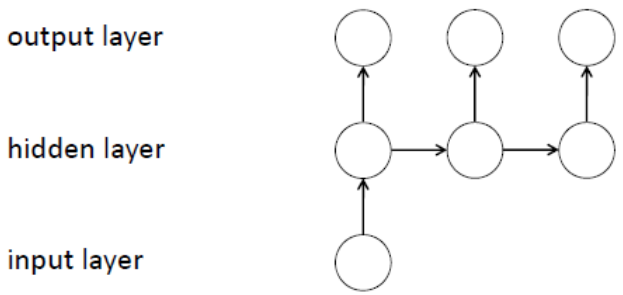
Variants of sequence processing



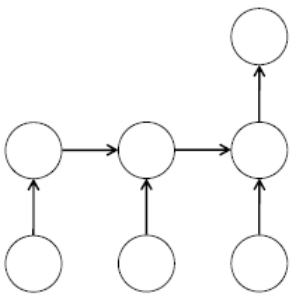
(a) Many-to-many (synced)



(b) Many-to-many (delayed)



(c) One-to-many



(d) Many-to-one

Application examples:

(a) Temperature forecast
(b) Automatic translation
(c) Image captioning
(d) Text classification

[2]

Application example in physics

Nuclear Inst. and Methods in Physics Research, A 867 (2017) 40–50

Contents lists available at ScienceDirect

Nuclear Inst. and Methods in Physics Research, A

journal homepage: www.elsevier.com/locate/nima

Using LSTM recurrent neural networks for monitoring the LHC superconducting magnets

Maciej Wielgosz [a,*], Andrzej Skoczeń [b,c], Matej Mertik [d]

[a] Faculty of Computer Science, Electronics and Telecommunications, AGH University of Science and Technology, Kraków, Poland
[b] Faculty of Physics and Applied Computer Science, AGH University of Science and Technology, Kraków, Poland
[c] The European Organization for Nuclear Research — CERN, CH-1211 Geneva 23, Switzerland
[d] Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia

ARTICLE INFO

Keywords:
LHC
Recurrent neural networks
LSTM
Deep learning
Modeling

ABSTRACT

The superconducting LHC magnets are coupled with an electronic monitoring system which records and analyzes voltage time series reflecting their performance. A currently used system is based on a range of preprogrammed triggers which launches protection procedures when a misbehavior of the magnets is detected. All the procedures used in the protection equipment were designed and implemented according to known working scenarios of the system and are updated and monitored by human operators.

This paper proposes a novel approach to monitoring and fault protection of the Large Hadron Collider (LHC) superconducting magnets which employs state-of-the-art Deep Learning algorithms. Consequently, the authors of the paper decided to examine the performance of LSTM recurrent neural networks for modeling of voltage time series of the magnets. In order to address this challenging task different network architectures and hyper-parameters were used to achieve the best possible performance of the solution. The regression results were measured in terms of RMSE for different number of future steps and history length taken into account for the prediction. The best result of RMSE = 0.00104 was obtained for a network of 128 LSTM cells within the internal layer and 16 steps history buffer.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, Chapter 10, MIT Press, 2016
www. deeplearningbook.org

[2] M. Erdmann, J. Glombitza, G. Kasieczka, U. Klemradt, *Deep Learning for Physics Research,* World Scientific, 2021

[3] C. Olah, *Understanding LSTM networks*, 2015
colah.github.io/posts/2015-08-Understanding-LSTMs