Date of publication MAR-29, 2022, date of current version MAR-07, 2022
www.computingonline.net/ computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.19.3.2513

On strange memory effects in long–term forecasts using regularised Recurrent Neural Networks

ARTHUR LERKE[1], HERMANN HEßLING[2]
[1]University of Applied Sciences (HTW) Berlin, 12459 Berlin (e-mail: Arthur.Lerke@Student.HTW-Berlin.de)
[2]University of Applied Sciences (HTW) Berlin, 12459 Berlin (e-mail: Hermann.Hessling@HTW-Berlin.de)
Corresponding author: Arthur Lerke (e-mail: Arthur.Lerke@Student.HTW-Berlin.de).

ABSTRACT Recurrent neural networks (RNN) based on a long short-term memory (LSTM) are used for predicting the future out of a given set of time series data. Usually, only one future time step is predicted. In this article, the capability of LSTM networks for a wide look into the future is explored. The time series data are taken from the evolution of share prices from stock trading. As expected, the longer the view into the future the stronger the deviations between prediction and reality. However, strange memory effects are observed. They range from periodic predictions (with time periods of the order of one month) to predictions that are an exact copy of a long-term sequence from far previous data. The trigger mechanisms for recalling memory in LSTM networks seem to be rather independent of the behaviour of the time-series data within the last “sliding window" or “batch". Similar periodic predictions are also observed for GRU networks and if the trainable parameters are reduced drastically. A better understanding of the influence of regularisations details of RNNs may be helpful for improving their predictive power.

KEYWORDS Time series; Recurrent Neural Network (RNN); Long Short-Term Memory (LSTM); Gated Recurrent Unit (GRU); long–term forecasting; autoregressive model; stock market;

I. INTRODUCTION

In a Recurrent Neural Network (RNN), the output of a neural network is fed back into the input layer where, at each time step, it is combined with time series data. Theoretically, the weights of the neural network should be trainable in such a way that a historical evolution can be stored. However, this construction has to cope the vanishing or exploding gradients when the weights are trained using the Back-Propagation Through Time (BPTT) technique, e.g. [1], which makes it practically impossible to apply RNNs for longer time intervals.

Long Short-Term Memory (LSTM) networks are an extension of RNNs [2] and are able to store historic information over quite long time intervals [3].

Mathematically, the gradient problem of RNNs when using BPTT are due to Jacobian matrices that are multiplied at least as many times as the number of relevant time steps.[1]

[1]If the eigenvalues of a matrix are smaller/larger than one, products of this matrix with itself vanish/increase quite rapidly ($0.9^{100} \approx 2.7 \cdot 10^{-5}$, $1.1^{100} \approx 1.4 \cdot 10^{4}$).

By replacing "simple neurons" with so-called "memory cells" with sufficiently complex substructures, long product sequences are avoided and effectively replaced by sums. In other words, memory cells can be considered as a regularisation method in order to avoid singularities.

In physics, regularisation is a common method when an observable quantity turns out to be singular: it is then modified by introducing additional parameters, which, however, must not have any influence on the measurable values of the observable (but on values which, e.g., are beyond the measurement range or below the measurement accuracy). Applied to RNNs, this means that a solution to the gradient problems should be such that, on the one hand, the essential properties of a time series can be stored and, on the other hand, reliable temporal predictions are possible.

There is considerable freedom in defining the substructure of a memory cell and, accordingly, there are several variants of LSTM networks [4]. GRU networks, for example, have a comparatively simple memory cell structure [5].

Despite their success, LSTM networks have faced funda-

mental difficulties. The memory cells strongly increase the number of weights that need to be trained. Consequently, it is hard to extract causal reasonings why specific predictions are made. Furthermore, the stability of predictions against small changes of the input values of time series data is unclear.

In Ref. [6], LSTM networks are applied to weather time series data of 14 physical parameters (e.g. temperature and atmospheric pressure) collected in periods of 1 hour. It turns out that a single time–step forecast of the temperature is pretty good. However, multiple time–step forecasts (of all parameters in parallel via autoregressive feedback loops) turn out to be considerably less successful and, from this behaviour, it is concluded that LSTM networks are not the means of choice for long–term forecasts of weather data.

The motivation for this work resulted from the question, whether a time horizon can be specified up to where forecasts of an LSTM network are reliable. According to the "weak efficient market hypothesis", the current value of a stock price depends on its history but there is no effective method for predicting its future development [7]. In view of this hypothesis, the period of a reliable stock market forecast should effectively be zero time steps. However, as it will turn out, some previously unobserved properties of LSTM networks are now emerging quite clearly.

Related Work. Using LSTM networks to predict financial values is quite popular. The intention, in general, is to develop methods that allow the evaluation or improvement of the quality of forecasts. The following works reflect this approach.

Zanc et al. [8] try to predict intraday prices of the digital currency bitcoin. They trained an LSTM network with 100 epochs and a window with a length of 22 time steps. At first glance, the predicted price for one time step into the future (10 sec.) looks quite accurate. However, the prediction turns out to be nothing but a shifted version of the actual data by one time step.

To find optimal values for the hyper–parameters of LSTM networks, Chung et al. [9] use a genetic algorithm. They explore the influence of the length of the window and the number of LSTM units on predicting one day of the Korea Stock Price Index. Their hybrid model of an LSTM network and a genetic algorithm performs better than a simple model that predicts no day–to–day change.

II. MODEL DESIGN

In this section, the preparation of the data is is described and the details of LSTM networks are specified.

A. DATA PROCESSING

The time series data in this work are taken from “adjusted closing prices“[2] of a stock [10]. The raw data need to be preprocessed to use them in LSTM networks.

Firstly, the data are scaled to the range 0 to 1 (min-max normalisation), i.e. the price p_t of the day t obeys $0 \leq p_t \leq 1$. Secondly, the data set is split into a “training set" (for adjusting the weights of the LSTM network) and a “validation set" (for checking the forecast capabilities of the network). In a further step, “sequences" of N_w consecutive prices are generated from the training data (N_w is the length of a "sliding window"). From the sequences, two arrays are constructed: a 2–dimensional array x of input values and a 1–dimensional array y of output values [11]. The underlying idea is to consider the historical evolution in the sense that a price p_t at time t is influenced by the prices p_τ at the N_w previous times $\tau \in \{t-1, t-2, \ldots, t-N_w\}$. For example, for a training sample of $N_T = 8$ prices $[p_1, p_2, \ldots, p_8]$ and for a window of size $N_w = 5$, the input array consists of $N_T - N_w = 3$ sequences

$$x = [[p_1, p_2, p_3, p_4, p_5], [p_2, p_3, p_4, p_5, p_6], [p_3, p_4, p_5, p_6, p_7]]$$

that are obtained by “sliding" a window over the training data, and the output array

$$y = [p_6, p_7, p_8].$$

is given by the last 3 prices.

During the training phase, the array x is forwarded through an LSTM network, whose weights are adjusted such that the outcome $\widehat{p}_i$ of the i–th window of the array x becomes a good approximation of the i–th element of the array y, specifically the Mean Squared Error (MSE)

$$MSE = \frac{1}{N_T - N_w} \sum_{t=N_w+1}^{N_T} (p_t - \widehat{p}_{t-N_w})^2.$$

is minimised. The weights are determined using the backpropagation method. A common procedure for approximating the MSE is based on so–called (mini–)batches: the set of sequences is evenly divided into N_b disjoint subsets, where each subset consists of consecutive elements of the arrays x and y. The number N_b is called “batch size". The gradients of the backpropagation method are updated after the sequences of a batch have passed through the LSTM network.

An “epoch" characterises an iteration over the whole training set. Considering only one epoch is usually not sufficient as, for example, the “causal connections" during the transitions between the batch samples may not well be represented. To optimise an LSTM network, iterations are performed over N_e epochs, where N_e is large enough to make the MSE sufficiently small. Here, the weights at the end of one epoch are used as the initial weights of the next epoch. For more details see e.g. [12].

B. AUTOREGRESSIVE MODEL

Autoregressive models can be used for n–step–ahead forecasting. For a given window, a price for the next time step is predicted. The prediction is used to create a new window

[2]The closing raw price of a day is modified by taking relevant actions into account that influenced its value after the close of the stock market, e.g. dividend payments or stock splits

by “sliding" the given window by one time step towards the future, where the predicted price becomes the most recent price and the oldest price of the given window is removed. The new window is put again into the model to generate a prediction for the day after next. This process can be repeated until the desired number of days is generated.

C. ARCHITECTURE OF LSTM NETWORKS

An LSTM network consists of a set of parameters. In Table 1, the values are specified that are used in this article for predicting the evolution of stock prices.

Hyper-parameter	Value
No. of hidden LSTM layers	1, 4, 6
No. of LSTM units per layer	3, 64, 128
Batch size N_b	256
Window size N_w	20, 30
No. of epochs N_e	4000, 8000, 64000

Table 1. Parameters of the LSTM networks used in this article.

After each LSTM layer there is one dropout layer, whose rate is set to 0.2, i.e. 20 % of the output of the previous layer is randomly set to zero. This acts as a regularisation mechanism for reducing overfitting effects. The training is done with the ADAM optimiser [13] with a learning rate of 0.001 and the MSE as a loss function.

The batch size has the fixed value $N_b = 256$. The initial value for the number of epochs is 4000 and increased to 8000 if the training data are not reproduced sufficiently well (see Fig. 3). Similarly, an architecture that consists of too few stacked

LSTM layers and too few units per layer may not be sufficiently complex. For example, for a network with only two layers and 64 units per layer “strongly damped" autoregressive predictions were observed that converge rapidly to a horizontal line. To get a handle on this effect, the numbers of epochs, or layers, or units per layer have been increased. The LSTM networks are trained with a window length of 20, or 30 days.

The LSTM networks are built using TensorFlow version 2.4 [14]. The framework provides built–in APIs for training, and prediction (such as Model.fit(), and Model.predict(), see [15]). The function predict() is applied to the training data and, for each window, a price is provided for the next time step (the orange curves in the next section represent the outcome of this forecast procedure). A multiple–day forecast is constructed using the autoregressive model, see Section 2.2 (the black curves in the next section represent the outcome of a 200–day forecast of the training data).

To speed up the training, CuDNNLSTM [16] is used. These cells were developed by Nvidia to run efficient on Nvidia GPUs. In all tests the Nvidia T4 GPU [17] is utilised.

III. RESULTS

The blue dots in the following figures show the evolution of the stocks selected for this article. The range of the gray curves correspond to the time period from which the training data are taken[3]. The orange curves represent the outcome if a fully trained LSTM network is used to make day–by–day predictions. The fact that the orange curves are nearly identical to the dotted gray curves indicates a success of the training.

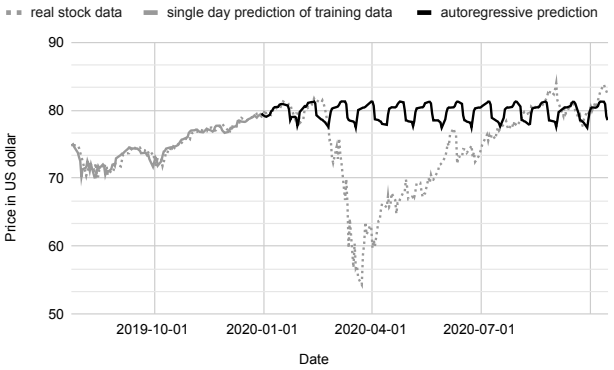

Figure 1. Evolution of the FTSE All World Index: training phase Apr. 2008 – Dec. 2019 (2971 days of training data, for reasons of clarity, only the last period of the training phase is shown, this applies also to the other figures), prediction period 200 days. Hyper–parameters: 6 LSTM layers with 64 units per layer, windows length 20 days, 4000 epochs.

Fig. 1 shows the outcome when an LSTM network is trained on the data of the FTSE All World Index [18], The black line shows the prediction of 200 days based on the autoregressive model: while the first two months after the training phase (at the end of December 2020) are quite well predicted, the collapse of the share due to the Corona pandemic is not reflected. Surprisingly, the black curve shows a repetition behaviour with a period of nearly one month. This strange property is not observed in the training data and must, nevertheless, be encoded in the LSTM network.

Fig. 2 shows the evolution of the stock price of Nvidia. Over almost the whole period (2007–2008), the price development shows almost linear growth (until Oct. 2008) and is therefore similar to the growth behaviour of the All World Index in 2019 (see Fig. 1). This observation suggests that it is worth considering whether the LSTM network of one share can be used to predict the performance of another share. The 1–day prediction during the training phase (orange curve) is determined by applying the LSTM network that was trained with the All-World Index: the agreement is surprisingly good in the first half of the training period and deviates only by a few percent in the second half. The 200 days forecast (black curve) also shows a periodic behaviour, however the amplitude is decreasing (like a damped oscillator).

[3]The default version of the API predict(), which is used in the analyses of this article, does not provide values for the first N_w time steps of the training data.

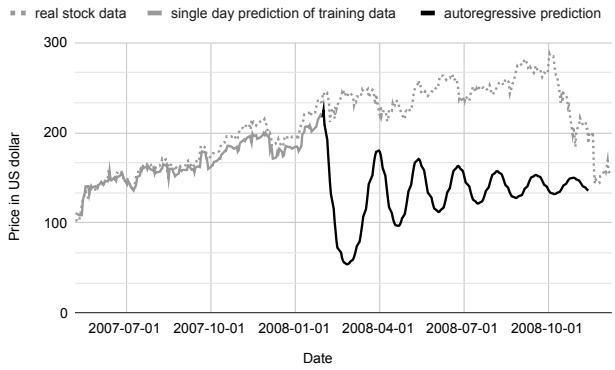

Figure 2. Stock price of Nvida: training data taken from the FTSE All-World Index (see Fig. 1), prediction period 200 days. Hyper–parameters: 6 LSTM layers with 64 units per layer, windows length 20 days, 4000 epochs.

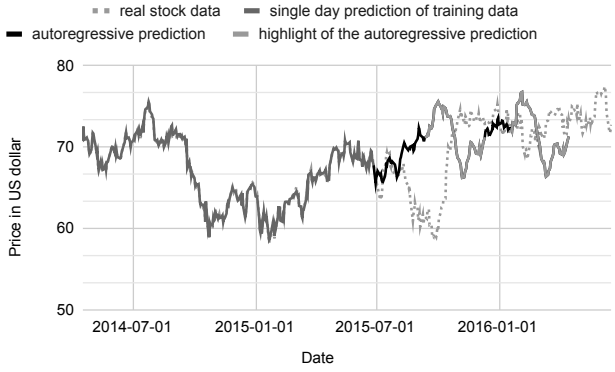

Figure 3. Stock price of SAP SE: training phase mid 2010 – mid 2015 (1238 days of training data), prediction period 200 days. Hyper–parameters: 4 LSTM layers with 64 units per layer, windows length 20 days, 8000 epochs.

A curious memory effect can be seen in Fig. 3. Within the forecast period after the training phase (black curve) a pattern is repeated after a certain time (highlighted by the gray curves in between the black). The shape of this repetition does not occur in the training period (mid 2010 - mid 2015). It should be noted that the period duration of the pattern is about a quarter of a year, which is considerably longer than the window size $N_w = 20$ days. The LSTM

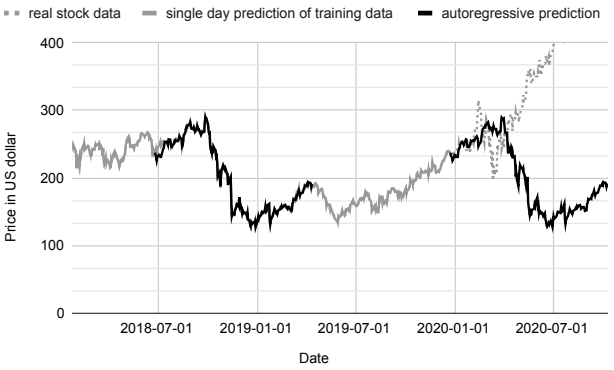

Figure 4. Stock price of Nvidia: training phase 1999 – 2019 (5237 days of training data), prediction period 200 days. Hyper–parameters: 6 LSTM layers with 128 units per layer, windows length 30 days, 4000 epochs.

network has in total 107,057 trainable parameters[4] and is seemingly able to remember predictions already made. At first glance, it is not apparent whether there is a triggering impulse for this flashback effect.

A further strange memory effect is shown in Fig. 4. The pattern of the forecast after the training phase is an (almost) exact copy of the stock price evolution a rather long time ago: see the highlighted black curve within the period July 2018 - March 2019. Apparently, the memory of an LSTM network seems to be active over long periods of time and can be triggered by a currently unknown mechanism.

When the training is limited to time series with short periods, the number of LSTM memory cells can be reduced strongly. Fig. 5 shows that only 76 trainable parameters are sufficient to train the performance of the SAP share (see Fig. 3) over a period of half a year, however, it should be noted that the number of epochs is significantly larger (64,000 instead of 8,000). Nevertheless, even for this rather small LSTM network, the prediction shows periodic behavior. This periodic property can even be observed, if the LSTM network is replaced by a GRU network with only 33 trainable parameters, see Fig. 6.

[4]The number is provided by the TensorFlow API summary(). Its high value indicates the complexity of the apparently simple LSTM network and is composed as follows. The dominant contribution is due the recurrent weight matrices R associated to the hidden state of each of the LSTM units: R increases quadratically with the number of LSTM units (the output of the hidden state of each LSTM unit is reconnected – in the next time step – to every LSTM unit within the layer). In addition, there are the input weights W, and the input bias parameters b_i of the time series data, and (for CuDNNLSTM units) the bias parameters b_r of the recurrent weights R. For the analysis shown in Figure 3, W, b_i, and b_r have 64 parameters each, whereas R has 64×64 parameters, i.e. in total there are $(64 + 64 + 64 \cdot 64 + 64) \cdot 4 = 17{,}152$ parameters, where the factor 4 takes into account that each LSTM unit consists of 4 sub–units (input gate, forget gate, cell state, output gate). Each of the second to fourth hidden LSTM layers contributes additional $64 \cdot 63 \cdot 4 = 16{,}128$ parameters, because each of the LSTM units of a hidden layer is “fully connected" to the every other LSTM output of the previous layer. The remaining 64+1 parameters belong to the final "dense layer".

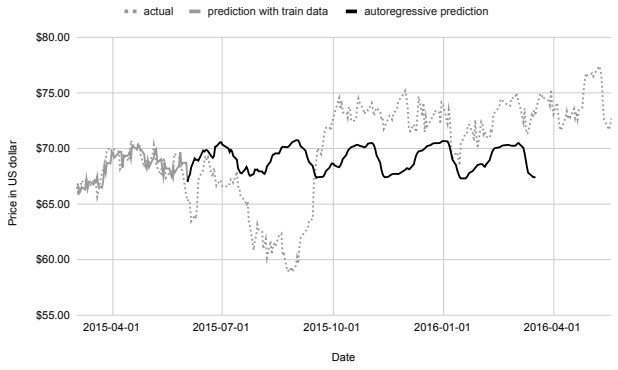

Figure 5. Stock price of SAP: training phase February 2015 – July 2015 (84 days of training data), prediction period 200 days. Hyper–parameters: 1 LSTM layer with 3 units, windows length 20 days, 64000 epochs, a total of 76 trainable parameters.

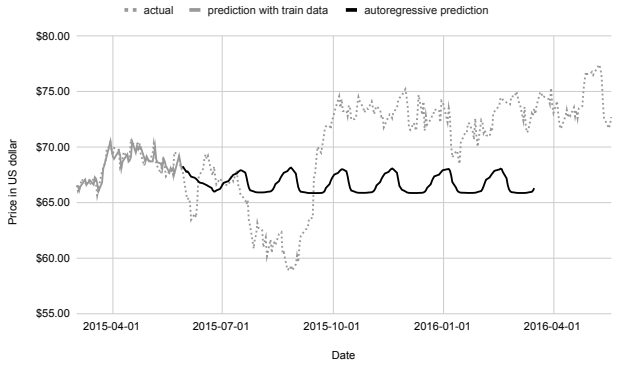

Figure 6. Stock price of SAP: training phase February 2015 – July 2015 (84 days of training data), prediction period 200 days. Hyper–parameters: 1 GRU layer with 2 units, windows length 20 days, 64000 epochs, a total of 33 trainable parameters.

IV. SUMMARY

LSTM networks are used to predict the evolution of stock prices over a rather long period of time. Given the highly dynamic development of stock prices, it is hardly surprising that the forecasts are not good. It is, however, remarkable that periodic patterns can be observed in the forecasts and that these patterns seem to be either unrelated to the training data or, on the contrary, are exact copies of rather long sub–periods of the training data. Whether there are relationships between the two effects or, more fundamentally, whether a common cause can be identified, is not known. In this context, it is worth mentioning that an increase of the number of LSTM units per layer seems to increase the probability for long–term predictions to show a periodic behaviour. Moreover, the observed “limit cycles" seem to be related to commonalities in the regularisation strategy of LSTM and GRU networks. Can LSTM and GRU networks even be associated with specific classes of feedback circuits? In this context, efforts to connect recurrent neural networks with neural ordinary differential equations are interesting, e.g. [19]. More detailed analyses are necessary for assured conclusions.

References

[1] P. J. Werbos, “Generalization of backpropagation with applications to a recurrent gas market model,” Neural Networks, vol. 1, pp. 339–356, 1988.
[2] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, pp. 1735–1780, 1997.
[3] F. A. Gers and J. Schmidhuber, “LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages,” IEEE Transactions on Neural Networks, vol. 12, no. 6, pp. 1333–1340, 2001.
[4] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A Search Space Odyssey,” p. 2222, 2017.
[5] K. C. B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning Phrase Representations using RNN Encoder- Decoder for Statistical Machine Translation.”
[6] (2021) Time series forecasting. [Online]. Available: https://www.tensorflow.org/tutorials/structured_data/time_series#advanced_autoregressive_model
[7] E. F. Fama, “Efficient capital markets: A review of theory and empirical work,” The Journal of Finance, vol. 25, no. 2, pp. 383–417, 1970. [Online]. Available: http://www.jstor.org/stable/2325486
[8] R. Zanc, T. Cioara, and I. Anghel, “Forecasting financial markets using deep learning,” pp. 459–466, 09 2019.
[9] H. Chung and K. Shin, “Genetic algorithm-optimized long short-term memory network for stock market prediction,” Sustainability, vol. 10, p. 3765, 2018.
[10] A. Ganti. (2020) Adjusted closing price. [Online]. Available: https://www.investopedia.com/terms/a/adjusted_closing_price.asp
[11] J. Brownlee. (2020) How to develop lstm models for time series forecasting. [Online]. Available: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
[12] R. C. Staudemeyer and E. R. Morris. (2019) Understanding lstm - a tutorial into long short-term memory recurrent neural networks. [Online]. Available: https://www.researchgate.net/publication/335975993_Understanding_LSTM_--_a_tutorial_into_Long_Short-Term_Memory_Recurrent_Neural_Networks
[13] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv:1412.6980, 2015.
[14] (2021) TensorFlow software. [Online]. Available: https://www.tensorflow.org/
[15] (2021) Training & evaluation with the built-in methods. [Online]. Available: https://keras.io/guides/training_with_built_in_methods/
[16] (2021) TensorFlow docs. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM
[17] (2021) Nvidia t4. [Online]. Available: https://www.nvidia.com/en-us/data-center/tesla-t4/
[18] (2021) Ftse all-world index. [Online]. Available: http://www.ftse.com/Analytics/FactSheets/Home/DownloadSingleIssue/GAE?issueName=AWORLDS
[19] M. Habiba and B. A. Pearlmutter, “Neural ordinary differential equation based recurrent neural network model,” IEEE: 31st Irish Signals and Systems Conference (ISSC), pp. 1–6, 2020.

ARTHUR LERKE has received a bachelor degree in Applied Informatics (2021) from the University of Applied Sciences HTW Berlin. He is Master of Applied Informatics degree candidate. Research interests: machine learning, in particular for predicting time series.

HERMANN HEßLING is Professor of Applied Informatics at the University of Applied Sciences HTW Berlin. After studying Physics at the Universities of Münster, Göttingen and Hamburg, he received his Ph.D. from the University of Hamburg and was a postdoctoral researcher at Deutsches Elektronen-Synchrotron (DESY). His interests include the analysis of Big Data in real–time.