14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020)
Virtual Congress: 11—15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

ESTIMATION OF LOAD-TIME CURVES USING RECURRENT NEURAL NETWORKS BASED ON CAN BUS SIGNALS

Dominik Herz[1,2], Constantin Krauß[2], Clemens Zimmerling[2], Benedikt Grupp[1], Prof. Dr. rer. nat. Frank Gauterin[2]

[1] Dr. Ing. h. c. F. Porsche AG
R&D Center Weissach, 911 Porschestraße, 71287 Weissach
dominik.herz2@porsche.de

[2] Karlsruhe Institute of Technology (KIT)
Institute for Vehicle System Technology, 2 Rintheimer Querallee, 76131 Karlsruhe, Germany
fast.kit.edu

Key words: Machine Learning, Structural Durability, Deep Neural Network, Lifecycle Analysis

Abstract. Precise knowledge of the load history of safety-relevant structures is a central aspect within the fatigue strength design of modern vehicles. Since the experimental measurement of load variables is complex and therefore associated with high costs, vehicles require estimation of these variables in order to design even more customer-orientedly in the future and thus consistently pursue sustainable lightweight construction. Hence the data measured by sensors in today's standard production vehicles is based on vehicle bus system signals which can be permanently retrieved. Due to the increasing availability of large quantities of recorded vehicle data, machine learning methods are moving into the focus of application. In this work, the implementation of Recurrent Neural Networks for the estimation of load-time curves is investigated. In order to close existing gaps in the state of the art, successful concepts of machine learning for sequential data, such as speech processing, are to be transferred to this application case. Long Short-Term Memory cells [1] play a central role for this type of problem. In addition to the adaptation of the network architecture, the integration of engineering knowledge is pursued within the method development process in order to increase the quality of the model. Relevant input variables are specifically selected by feature engineering and new meaningful variables are generated by filtering. Statistical analysis is used to investigate the correlation of these input signals with the estimated quantities. The development of a robust load estimation takes place in the course of model development on the basis of the torque of the left-hand rear drive shaft. Results reveal that the Recurrent Neural Networks approach is justified in estimating the highly nonlinear load curve of a complexly loaded part – as a component of the dynamic system – by means of available sensor signals [2]. Subsequently, the model is validated using recorded measurement data for different chassis settings of the same vehicle. Finally, the transferability of the designed network configuration to other chassis components of the same vehicle is investigated and evaluated.

1 INTRODUCTION

The use of Neural Networks for modeling and predicting time series has gained tremendous importance. Artificial Neural Networks (ANN) have proven to be a viable competitor compared to various traditional time series models [3]. Hidden Markov Models (HMM) have a long history on the field of generative and discrete sequence modeling. [4] shows that such models are further capable of establishing temporal relationships in sequential data. The focus of this work is on computational models in the form of ANNs. A common example of Feedforward Neural Networks (FFNN) is the Multi Layer Perceptron (MLP), which is composed of several hidden layers. Such universal network architectures are suitable for a wide range of applications. Especially in the field of autonomous driving promising results could be achieved as pointed out by D. A. Pomerleau [6]. However, FFNNs also find application in sequential learning approaches. Within the scope of [7], D. Vengertsev investigates in his work the suitability of deep FFNNs in comparison to previous models for time series prediction. The author shows that MLPs are also capable of predicting time series. In sequence learning, mainly Recurrent Neural Networks (RNN) have led to promising results. For this reason, the application and qualification of RNNs in this field will be discussed in more detail.

Recurrent Neural Networks – analogous to Feedforward Neural Networks – yield far better results, especially in application areas such as image processing and pattern recognition. RNNs mainly show a large number of breakthroughs in Natural Language Processing (NLP) [8]. Those are focused in successes in the subareas of automatic question answering, speech recognition, and machine translation. In addition, RNNs have also shown promise in applications in the field of sequence learning. In the work of [9], Lipton Z. C. cunducts a comparison between RNNs and HMMs for problems with long temporal dependencies. This shows that the computationally intensive and complex mapping of long-term dependencies is a central disadvantage of HMM. A difficulty in the application of simple recurrent units is the vanishing gradient problem described by [10]. Thus, successful applications of RNNs for sequence learning to date are due to complex recurrent units, which include gates. Typical representatives of such units are Long Short-Term Memory (LSTM) cells or Gated Recurrent Units (GRU). Through a comprehensive analysis using different experiments, [11] shows that by using LSTM units, long-term dependencies can be learned significantly better compared to other variations of RNNs. [12] demonstrates, using handwriting recognition as an example, that LSTM networks have led to far better results than previous results using HMMs to generate complex sequences with long-range structure.

As an alternative to the LSTM cell, the GRU proposed by [13] is also used in the field of sequential learning. Its structure differs from the LSTM unit only in the number of gates. Due to the absence of the output gate, this consequently has a smaller number of parameters to be learned. [14] evaluate the behavior of LSTM and GR units for the task of sequence modeling on a set of music and raw speech signal data compared to traditional recurrent units. The results show the clear superiority of both LSTM and GR units. Furthermore, [15] prove that GRUs provide similar or even slightly improved results as part of an extensive study for various sequential learning problems. In addition, using a torque-controlled robotic arm as an example, [2] show that the use of Long Short-Term Memory networks is quite suitable to describe a real, complex as well as dynamic system. In the context of a hyperparameter study, the best possible settings for datasets with over $100,000$ samples could be found to predict the highly nonlinear torque signal of the joint. For the present thesis, similar conditions apply in terms of data and estimation size. Nevertheless, the problem is differentiated in the context of this thesis. No simple equations of motion are solved, but the implementation aims to describe unknown stresses of a complex loaded component within a highly dynamic system by means of sensor data. The first orientation concerning the choice of hyperparameters should nevertheless be done according to the study investigated by [2]. Based on previous experience with RNNs, the use of complex recurrent units is a promising approach. Both Long Short-Term Memory cells and Gated Recurrent Units are suitable for the estimation of time series. In the further course of the work, only the use of LSTM cells is proposed.

2 RECURRENT NEURAL NETWORKS

For the current timestep, the prediction of FFNNs is based solely on the recognition of correlations for each independently and identically distributed sample [16]. In comparison, recurrent network architectures are dynamic systems that additionally include a dependency on points in time that lie in the past. RNNs are able to store past network states and thus learn temporal dependencies. By activating individual neurons ${}^{j}a$, this class of networks has an internal state h_t that affects future computations. Consequently, they have implicit character.

2.1 Network architecture

Characteristic of such network structures are both the links between neighboring layers and feedback edges. These connections end in the layer from which they originate. The prerequisite for training RNNs is the transformation of the implicit structure into an explicit one. To do this, the network is mentally rolled out along the time axis. The following figure 2.1 schematically illustrates how a RNN works.

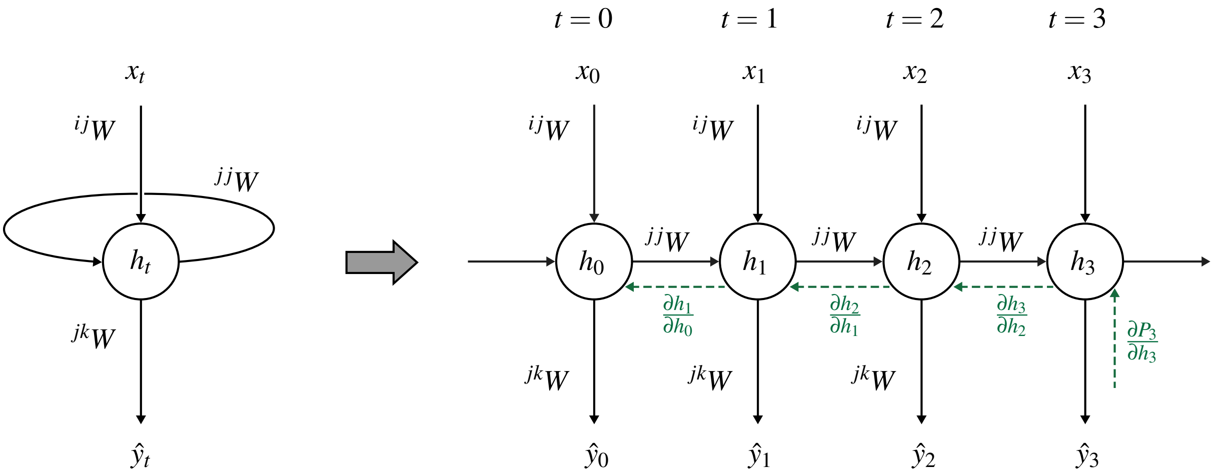


Figure 1: Illustrated explanation of a recurrent neural network in compact (left) and unrolled (right) representation form with input value x, approximated output value $\hat{y}$, input layer i, hidden layer j, output layer k, weighting matrices W as well as the internal states h. Own illustration based on [5].

The internal state at the current time t: $h_t = \phi({}^{j}z_t) = \phi({}^{ij}W^{\top}x_t + {}^{jj}W^{\top}h_{t-1} + {}^{i}b)$ with activation function $\phi(z)$ and bias b can be calculated as a function of the input x_t and the state feedback $h_{t-1} = {}^{j}a(t-1)$.

From this, in turn, the output of the current time can be determined: $\hat{y}_t = \phi({}^{k}z_t) = \phi({}^{jk}W^{\top}h_t + {}^{j}b)$. The weighting matrices W are identical for each time step. Consequently, they are valid over the entire length of the temporal sequence to be processed. Analogous to FFNN, RNN are considered as very deep networks with respect to their optimization.

2.2 Long Short-Term Memory

Long short-term memory networks (short: LSTM) are able to prevent the disappearance of the gradient and to capture long-term dependencies [17]. Instead of ordinary neurons, the structure of the network is composed of complex LSTM cells. The cell is given a cell state C_t, which serves as an error memory. The flow of information within a cell takes place through so-called *gates*, through which the existence of the information is guaranteed over several time steps. They are divided into the *Input*-, *Forget*- and *Output*-Gate. The input gate $i_t = \sigma(W_i^{\top}[h_{t-1}, x_t] + b_f)$ decides which information to extract from both the input data x_t and the internal state $h_t = o_t \odot \tanh(C_t)$ into the memory $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$ should be included or prevented. The mathematical operator $\odot$ describes the Hadamard product, whose result matrix is obtained by multiplying the respective corresponding entries of two output matrices. The output gate o_t in turn learns which content of the memory is essential for the current computational step of the prediction and is read out. It can be mathematically expressed as follows: $o_t = \sigma(W_o^{\top}[h_{t-1}, x_t] + b_o)$. In order to decide which information of the error memory can be neglected or deleted for the further course of the estimation, it is useful to use the forget gate $f_t = \sigma(W_f^{\top}[h_{t-1}, x_t] + b_f)$ according to [1]. Finally, a new candidate $\tilde{C}_t$ is computed for the cell state C_t: $\tilde{C}_t = \tanh(W_C^{\top}[h_{t-1}, x_t] + b_C)$.

2.3 Database and Preprocessing

The design of the network architecture for estimating the load-time functions is based on measurement data sets from different vehicles. All measurement vehicles are all-wheel drive, purely electric sports cars, which are equipped with different measurement systems. Only signals that are measured by sensors as standard in today's series-production vehicles are to be used for estimating the load-time functions. Thus, these signals are already available on the Controller Area Network (CAN) and can be retrieved at any time. Furthermore, the corresponding stress-time functions must be provided as reference signals for the supervised learning approach. The signals are determined with the aid of additional electrical measuring systems. Special sensors, so-called strain gauges, enable the recording of force and torque curves on the basis of small, relative changes in length under the influence of force and/or temperature. In the following, the measures that are used for the selection of the measurement variables are shown. The input variables of the neural network are selected by adding a correlation analysis and by integrating physical knowledge. In the context of this work, the common normalization technique called min-max normalization is used [18]. Due to the nature of the data as well as the range of values of the estimated variable, the normalization of all features on the interval $r \in [-1, 1]$ is appropriate in the context of this work. The aforementioned correlation analysis is initially intended to provide an overview – without any incorporation of physical knowledge – of the relationships between internally measured sensor signals and external load signals. The evaluation of the correlation intensity between two variables is conducted by the Bravais-Pearson coefficient γ, which is defined on the following range of values: $-1 \leq \gamma \leq +1$. With the use of this satistical method, the model variables statistical influences can be pre-estimated.

Following the preprocessing of the data by means of correlation analysis, relevant features are to be separated from less relevant ones or even newly generated with the help of cross-domain knowledge through data modeling and investigation. In the scope of this work, this approach is implemented by signal analysis in conjunction with appropriate filtering. Power sepctral density filtering is applied on raw data to enhance quality and reduce noise of the underlying time series.

3 METHODOLOGY

The aim of this work is the design of different models which allow an approximation of highly nonlinear time series. The various CAN bus variables serve as input variables. The output variable to be estimated is the respective stress-time function. The derived network architecture should be capable of estimating intervals of low frequency. In addition to a high estimation quality, the model should exhibit a robust behavior for situations that are only rarely included in the training data. In principle, the design of a virtual sensor is spoken of.

3.1 Construction of the first network architecture

For a first estimation of the approximation quality by a RNN with LSTM cells, we firstly choose a simple model architecture. In [19], Pascanu et al. propose the extension of the input and output of recurrent units by additional forward layers. For a good representation of nonlinearities and learning of complex relationships, we add a feed-forward layer between the input layer and the recurrent layer. So we are able to achieve a high model quality already at the beginning of method development. This layer added to the network architecture will be referred to as projection layer in the following. The number of neurons is chosen to be 250 for this layer. Due to normalization of the data between $[-1,1]$, an activation function of the tangent hyperbolicus type is used. It is assumed that the network is capable of recognizing more abstract relationships in the input data. To prevent overfitting, the dropout method is used analogously to the recurrent layer. In addition, the capacity of the LSTM layer is chosen to contain 40 LSTM cells. Along with this, the L1 and L2 regularization must be used in addition to the dropout. Those measures are based on the assumption that the network can form abstract representations based on the input data by adding the projection layer. Accordingly, the subsequent recurrent layer should be able to learn temporal relationships more easily. With respect to optimization effort and computation time, the insertion of another forward layer does not represent a deficit worth mentioning.

3.2 Adjustment of the target function

To be able to evaluate the performance measure of the neural network, the Mean Squared Error (MSE) was used for the model building described previously. The results obtained so far show that high load margins are poorly represented by the model. From the perspective of strucural durability, it is crucial to be capable of mapping these load peaks, since they contribute most to the demage of the component. Therefore, the modification of the loss function of the network is now aimed at. The basic idea is to penalize deviations at higher absolute values, which represent load peaks. For this purpose, the weighting of the data point is selected depending on the magnitude of the actual time. To avoid neglecting the influence of values that are close to the mean, a lower bound that can be interpreted as a minimum weight can be additionally specified. It should be explicitly noted that this function is an extension of the MSE and should not be confused with the pre-implemented L^N norms, which penalize high deviations. In this case, deviations at high values are penalized. By adjusting the loss function as described, it is expected that high amplitudes within the time series could be better predicted. Conversely, this can be done at the expense of the estimation quality of small load cycles. The performance of the algorithm is further evaluated in the course of the work by the specially dedicated mean error measure

$$\mathrm{MPE} = \frac{1}{n}\sum_{i=1}^{n} |\hat{y} - y|_i \cdot \max(|\beta \cdot y_i|, G_{min}) \ , \tag{1}$$

for which the abbreviation MPE is used. The quantity β describes the scaling factor, which determines the strength of the weighting at absolute function values. This is selected on the basis of a series of experiments at $\beta = 500$. The size G_{min} represents a lower bound, so that values $< G_{min}$ are nevertheless considered with the fixed value of $G_{min} = 1$. The graphs in figure 2 are intended to illustrate different error metrices. Here – unlike the MPE metric – no dependence of the weighting G_i on the absolute function value y can be seen for the MAE and MSE metrics. Moreover, the strength of the weighting can be illustrated by projecting the graph area into the y-$(y-\hat{y})$-plane. Increasing saturation correlates with a higher weighting of the deviations.

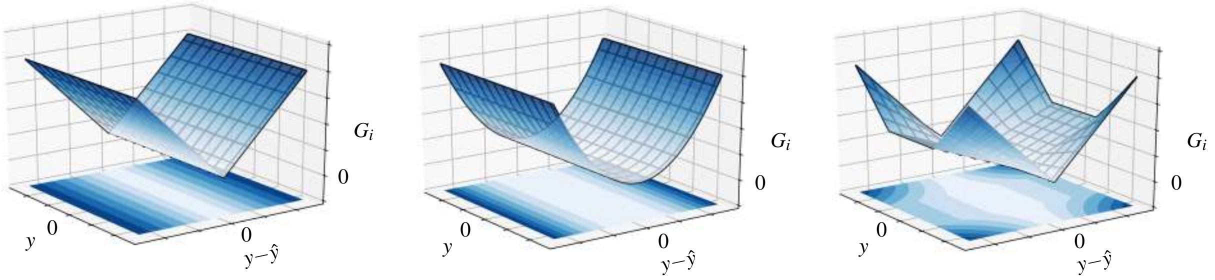


Figure 2: Graphical representation of different loss functions: MAE (left), MSE/RMSE (center), and MPE (right)

3.3 Consideration of frequency dependencies and influence of selected data

In order to additionally consider frequency dependencies of the dynamic system during the training phase, the influence of several past time steps on the performance of the predictive model is investigated in the further course of the method development process. This measure is associated with significant computational and storage effort, which limits the amount of backtracking. Within the scope of an elaborate study, the transfer of 20 additional past data points shall be targeted in the following and subsequently the most performant calculation shall be selected for the further method development process. The network configuration remains unchanged during the execution. In addition, the existing measurement data is analyzed in more detail to identify characteristic phenomena. Correctly interpreted, conclusions can be drawn about system-describing effects, which serve as a basis for meaningful input variables. The selection of information-rich signals is done on the one hand by engineering understanding, on the other hand by using the already described correlation investigation. For the second measure, only signals that satisfy the condition $|\gamma| > 0,1$ should be considered. The bandpass filtering in the natural frequency range of the drive shaft allowed for additional features to be generated for this approach. The wheel speeds filtered in the 10 – 15 Hz interval correlate strongly with the time history of Anti-Lock braking System (ABS) braking. By separating invalid features, the model should be able to specifically identify correlations in the data and increase its quality of fit accordingly. This methodological approach has now been implemented for the drive shaft torque, evaluated and used for a comparison with the previous results.

3.4 Final network architecture

Finally, this section is intended to provide information on the final choice of network configuration, thus complementing the method development by extending the previous model with two additional hidden layers. From a theoretical perspective, the nonlinear, dynamic overall system should lead to an even more complex or higher abstraction capability of the model by increasing the number of interactions between the input variables to be computed. In figure 3 the final network architecture is listed, which is described in the following. The continuous reduction of the number of neurons per layer with an increase of the network depth turns out to be a quite reasonable approach for the problem at hand. This applies to both the forward and the recurrent layer, up to the value-continuous output of the single neuron. The basic idea represents the continuous breaking down of the abstraction level to the scalar output value of the output neuron. Following on from the first recurrent LSTM layer, another feedforward layer is incorporated, for which the choice of neurons is made to 30.

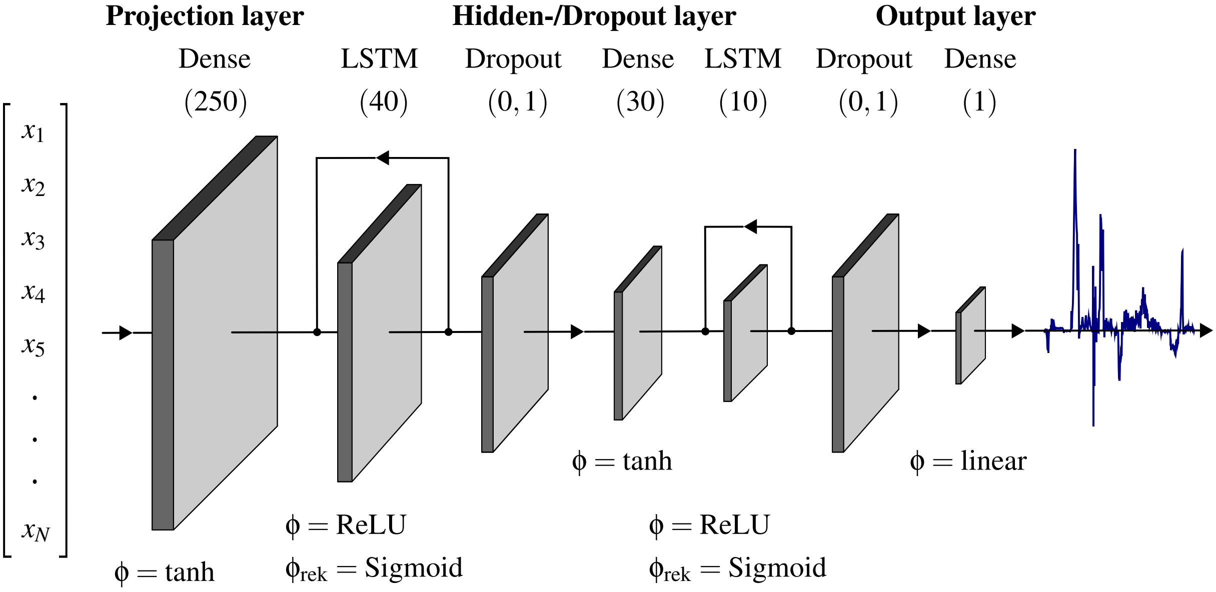
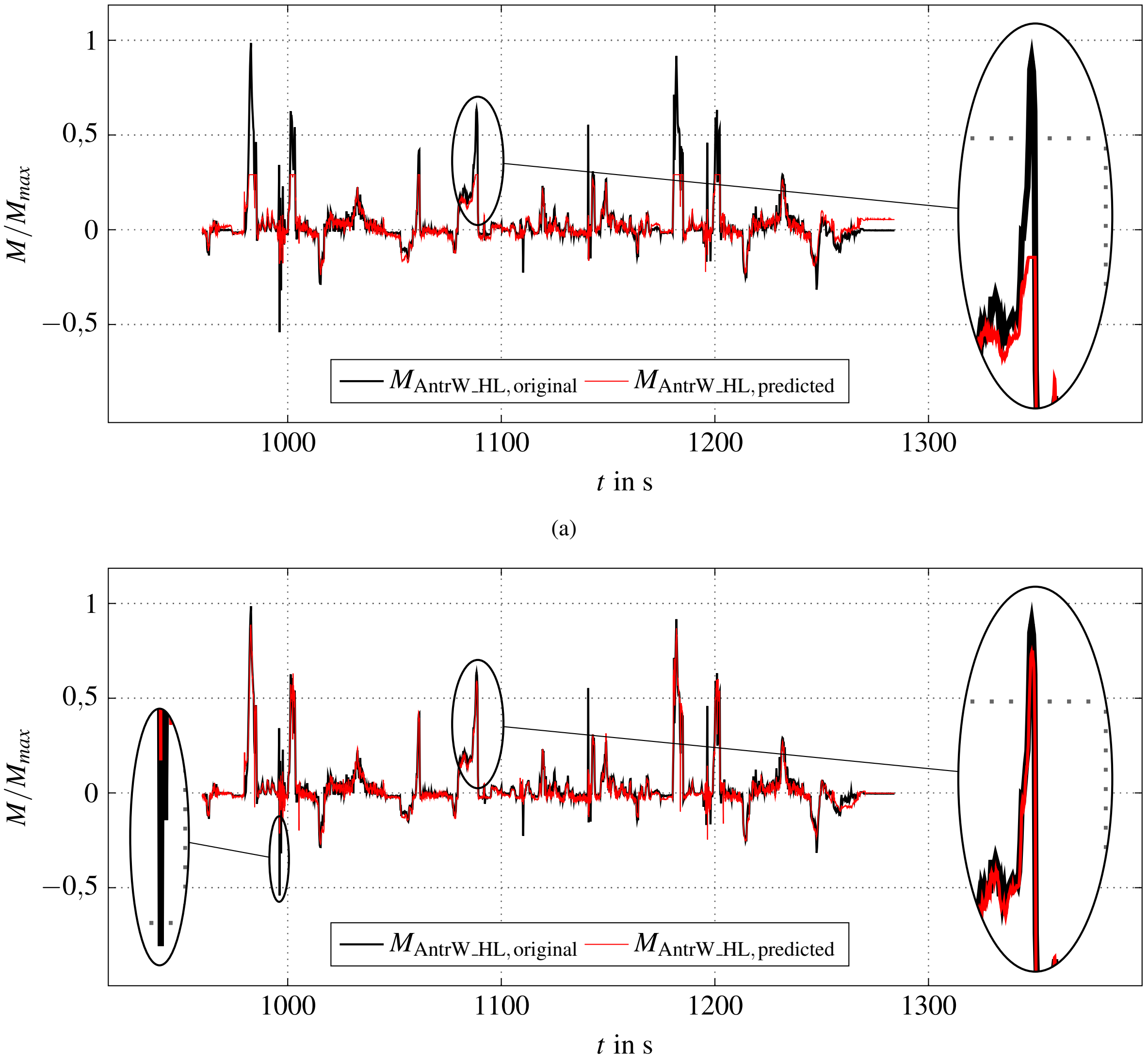


Figure 3: Abstract representation of the final network architecture with the derived network parameters

Due to the satisfactory result quality of the previous network structure, the choice of the activations of the neuronal units should fall on the tangent hyperbolicus function analogous to the forward projection layer by the taken normalization range. Moreover, the immediate embedding of another recurrent LSTM layer reveals promising for the handling of sequential data by the overall architecture. The activation functions of the LSTM cells are also selected in the same way as those from the initial network architecture. However, for this use case, the strict increase in network depth initially manifests itself in a slight overfitting on the training data. A countermeasure is implemented by adding a dropout regularization layer of 10 %. Thus, the ability of the model to generalize can still be guaranteed by the increased capacity. The renewed increase of the network depth is only expressed in an increase of the computation time as well as an increased tendency to overfit. The result quality cannot be positively influenced by this.

4 RESULTS AND DISCUSSION

The progression marked in black in figure 4 (a) shows the measured reference signal of the drive shaft torque for the last 20 % of the entire sequence. The red signal above represents the progression of the predicted signal through the ANN. For the sake of clear documentation, the reference values or estimates of the torque are introduced below by M and $\hat{M}$, respectively. The enlarged representation in figure 4 (a) shows the difficulty of estimating high torque excursions. The deviations on the test data reach maximum values of up to $|M-\hat{M}|/M_{max} = 0.62$. Although it could be stated that the recurrent neural network captures nonlinearities of the dynamical system with the derived model parameters, nevertheless a satisfactory quality of results is not achieved for this model. A new custom loss function has been introduced (3.2). The RNN is trained with the same network architecture. The effect of an improvement in the peak load estimate is much more pronounced compared to the previous calculations, which can be seen from the enlarged area of a representative point in figure 4 (b). This is due to the adjusted loss function and confirms the approach. Looking qualitatively at both trajectories, it can actually be seen that the discrepancy between $\hat{M}$ and M can be reduced significantly for values $M > 0$. For $M < 0$ this is not confirmable to the same extent. Especially for ABS braking, which can be identified in the range between $t \approx 990$ s and $t \approx 1190$ s, the neural network still shows weaknesses. A possible explanation can be found in the fact that the RNN lacks the ability to abstract basic relationships due to a lack of data for such high-frequency special events. The error on the training data is given by $\mathrm{MPE}_{\mathrm{training}} = 0.151$. This contrasts with a significantly improved goodness on the test data of $\mathrm{MPE}_{\mathrm{test}} = 0.091$. Although the aforementioned comparison of the investigations cannot be cunducted through using the error measure, by simply looking at the predicted progression in figure 4, a significant improvement in performance on the test data sequence can be observed. Accordingly, for the same network complexity, the third model resulted in significantly improved generalization and noticeably increased its performance on unseen data. In summary, this investigation has the significant tendency to underestimate the high load peaks can be reduced due to the introduced error function, which testifies to a more accurate representation of the nonlinear properties of the system. Equations of motion in dynamics, which unify the fields of kinematics and kinetics, are always second order differential equations in time. Since the signals to be estimated are kinetic quantities, it is natural to expect a dependence on the second order derivatives of the input quantities. An approximation by means of backward finite differences requires at least two support points from the past at the lowest order of convergence. For this reason, the following investigations are carried out with a minimum value of $n_{\mathrm{timesteps}} = 2$. The results of the parameter study are shown below. Figure 5 shows the course of the loss functions during the optimization as a set of curves for different values of $n_{\mathrm{timesteps}}$.

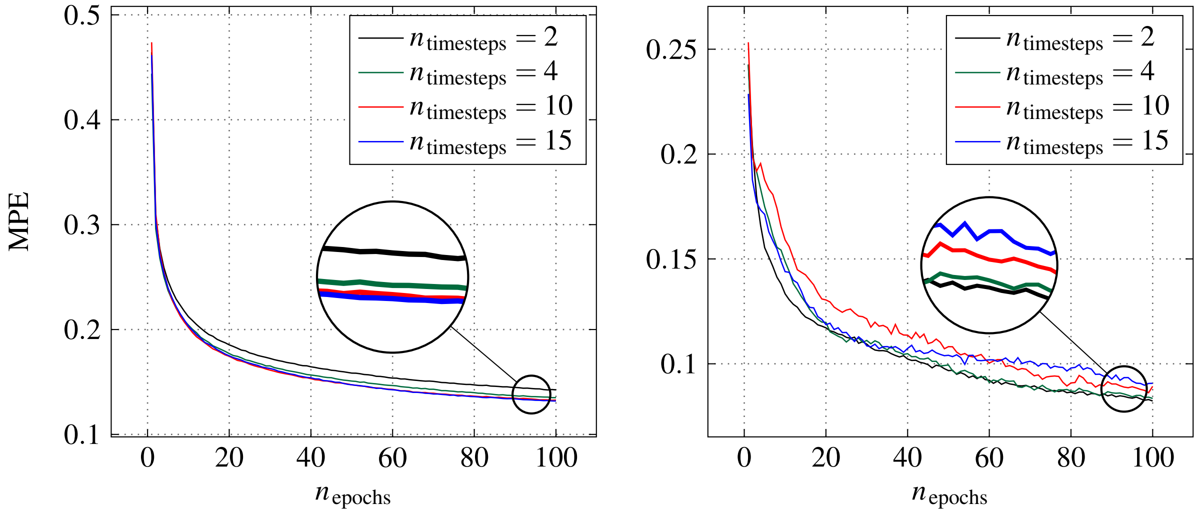


(a)

(b)

Figure 4: Estimated course of the drive shaft torque over time (red) compared to the reference value of the original measurement (black) on the test data for the second model: (a) without fitted objective function, (b) with MPE error metric

For clarity, only four progressions are displayed. Figure 5 (a) illustrates the differences in convergence behavior with varying numbers of considered past time steps for the training data. A positive correlation of the convergence rate and the distance of the absolute error value to the ideal value 0 as well as the successive increase of $n_{\text{timesteps}}$ is clearly detectable at first. However, for values > 10 the relation of the increase of the computational capacity and duration and the further, small improvement of the estimation quality has to be critically questioned. Furthermore, the course of the loss function on the validation data reveals that the statements made for the training set cannot be transferred. Rather, a reciprocal relationship can be observed. High values for $n_{\text{timesteps}}$ lead to a deterioration of the prediction of the validation time series. Therefore, it must be stated that an overfitting is favored by the increase of the model complexity. In addition, a slight improvement of the results could be observed by deliberately selected input signals. Thus, the error for the training data set can be reduced to $\text{MPE}_{\text{training}} = 0.092$. In addition, the test data error is simultaneously reduced to $\text{MPE}_{\text{test}} = 0.068$. This suggests that the model learned contained relationships better despite the reduced database, supporting the thesis that the model's performance is enhanced by feature engineering.



(a)

(b)

Figure 5: Course of the learning curves for the training data (a) and the validation data (b) related to a different number of past time points

By increasing the model depth, a further improvement of the results could be achieved in the last step of the model building process. The training performed and the subsequent application to the last 20 % of the total sequence reveal that the values of the training error are $\text{MPE}_{\text{training}} = 0.078$ and the test data error is $\text{MPE}_{\text{test}} = 0.052$. Both torque time series, shown in figure 6, do not reveal a completely congruent course. Nevertheless, the reduction of the error measure can be attributed to the deeper mesh structure. It becomes obvious that the choice of two integrated LSTM layers is permissible to identify and learn more abstract, temporal relationships in the data. With regard to ABS braking, a slight improvement can be observed by this approach. It is possible to reduce the difference of the moment peaks for the high-frequency sequences by approximately 23 % to $|M - \hat{M}|/M_{max} \approx 0.05$. However, even the much more complex model has weaknesses in these areas. Possible explanations can be found repeatedly to the theses from the previous investigation. Globally, the neural network delivers satisfactory results over the length of the test time series. However, the less accurate prediction in the area of braking suggests that the information content of the data is not sufficient to learn regularities for special loads of this kind. Thus, the limitations of machine learning methods can be clearly shown.

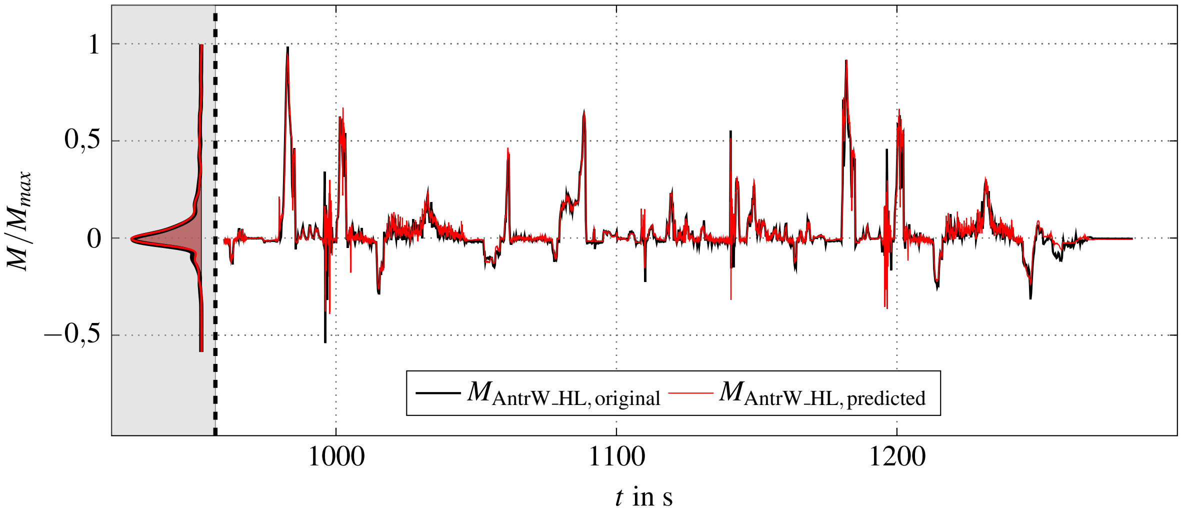


Figure 6: Course of the measured reference signal of the drive shaft (black) and the prediction by the final network architecture (red) for the selected input variables on the test data. In the gray colored area, a frequency distribution of the predicted (red) and the actual values (black) is given

5 CONCLUSION

The present work shows that recurrent neural networks as a possible form of machine learning methods allow in principle the precise approximation of highly nonlinear stress-time functions based on CAN bus signals in dynamic systems. Taking the left rear axle torque of the drive shaft of an all-electric vehicle as an example, a network architecture adapted to the complexity of the estimation task is designed as part of an extensive model development process. Due to insufficient results obtained by a simplified approach, in the form of a Long Short-Term Memory cell as a recurrent unit, the increase of the mesh depth is aimed at. The results show that increasing the model complexity results in a significant increase in the performance of the network. However, it can be listed that the prediction of high load peaks, especially for high frequency excitations, is a difficulty. In order to obtain the most realistic representation of the load-time signals, the adjustment of the loss function of the network to be minimized is performed. Due to the stronger weighting of deviations at high absolute function values, a clearly improved approximation quality can thus be observed. Subsequently, the influence of several past timesteps is examined so that frequency dependencies can be learned by the neural network. In addition, irrelevant features are sorted out by feature engineering on the one hand and new, more meaningful signals are generated by filtering operations on the other hand. The implementation of both measures shows that the algorithm is able to predict the load-time curve with small estimation deviations. Finally, the renewed increase in network depth is implemented to learn more abstract relationships in the data. The again improved results of the found network architecture show that the suitability of RNNs with LSTM cells for estimating the load function can be well confirmed.

REFERENCES

[1] Gers, F. A. and Schraudolph, N. N. and Schmidhuber, J. Learning precise timing with lstm recurrent networks. *J. Mach. Learn. Res.* (2003) **3**: 115-143.

[2] Rueckert, E. and Nakatenus, M. and Tosatto, S. and Peters, J. Learning inverse dynamics models in o(n) time with lstm networks. *Humanoids* (2017): 811–816.

[3] Zahng, G. and Patuwo, B.E. and Hu, M.Y. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting.* (1998) **14**: 35-62

[4] Hanif, M. and Sami, F. and Iqbal, M. Hidden markov model for time series prediction. *Journal of Asian Scientific Research.* (2017) **7**: 196-205.

[5] Pascanu, R. and Mikolov, T. and Bengio, Y. On the difficulty of training recurrent neural networks. *In Proceedings of the 30th International Conference on Machine Learning.* (2013): 1310-1318.

[6] Pomerleau, D. A. An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems 1* (1989): 305-313

[7] Vengertsev, D. Deep learning architecture for univariate time series forecasting. *Technical Report Stanford University* (2014)

[8] LeChun, Y. and Bengio, Y. and Hinton, G. E. Deep learning. *Nature* (2015) **521**: 436-444

[9] Lipton, Z. C. A critical review of recurrent neural networks for sequence learning. *CoRR* (2015)

[10] Bengio, Y. and Simard, P. and Frasconi P. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.* (1994): 157-166

[11] Hochreiter, S. and Bengio, Y. and Frasconi P. and Schmidhuber J. Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. *IEEE Press* (2001)

[12] Graves, A. Generating sequences with recurrent neural networks. *CoRR* (2013)

[13] Cho, K. and van Merriënboer, B. and Gülçehre, Ç. and Bahdanau, D. and Bougares F. and Schwenk, H. and Bengio, Y. Learning phrase representati- ons using rnn encoder–decoder for statistical machine translation. *In Proceedings of EMNLP* (2014): 1724-1734

[14] Chung, J. and Gülçehre, Ç. and Cho, K. and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* (2014)

[15] Jozefowicz, R. and Zaremba, W. and Sutskever, I. An empirical exploration of recurrent network architectures. *In Proceedings of International Conference on Machine Learning* (2015): 2342-2350

[16] Hastie, T. and Tibshirani, R. and Friedman, J. H. The elements of statistical learning: Data mining, inference and prediction. *Springer series in statistics, New York* (2009)

[17] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.* (1997): 1735-1780

[18] Kotsiantis, S. B. and Kanellopoulos, D. Data preprocessing for supervised learning. *International Journal of Computer, Electrical, Automation, Control and Information Engineering* (2006)

[19] Pascanu, R. and Gülçehre, Ç. and Cho, K. and Bengio, Y. How to construct deep recurrent neural networks. *CoRR* (2013)