Optimized LSTM neural networks for material damage predictions

Ivan Otic[1]

[1]Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract

We present a predictive model using Long Short-Term Memory (LSTM) neural networks to forecast component material conditions at timepoints (t + n) based on monitored historical sample data up to time t. LSTM architectures are renowned for their ability to capture long-term dependencies and are particularly well-suited for sequential data. In this work we introduce an improved and optimized LSTM model designed to enhance the performance and efficiency of sequential data predictions. Our approach includes a statistical evaluation that utilizes experimental process information extracted as time series data to develop multi-layer models for predicting the condition of component materials. We demonstrate the effectiveness of the new approach on a case study using NASA Ames milling data set by Agogino and Goebel [1]. Results suggest that the new method can precisely predict the component condition while significantly improving the efficiency. The proposed modeling approach can also be directly applied for predictions of other sequential data, as for example materials corrosion data, required for the design and safety evaluation of nuclear power plants.

1. Introduction

Nuclear power plant components must withstand severe mechanical and thermal loads throughout their life cycle. To prevent component and system failure one of the key challenges in damage tolerant design, manufacturing and maintenance is improving the accuracy of monitoring and prediction of possible component damage. However, monitoring and damage prediction of power plant components are a prototypical example for a highly non-linear system which is consequently very hard to model. In the frame of the Joint European Canadian Chinese Development of Small Modular Reactor Technology project ECC-SMART [2], Small Modular Supercritical Water Reactor (SM-SCWR) is being developed, see Schulenberg and Otic [3]. Material corrosion or component material damage for SM-SCWR are one of key chalanges in the development of new generation of small modular nuclear reactors with enhanced safety features. To predict material damage of machine components various methods have been developed over the past decades. Empirical and mechanical models have been developed to determine model coefficients from extensive experimental data, focusing on identifying machining characteristics over time. Empirical and mechanical models can approximate the relationship between tool wear and machining time but fail to explicitly identify critical wear status times, reflecting overall wear rate and real-time conditions. Additionally, these models lack generality due to their reliance on specific processing conditions. To address this, statistical and machine learning methods have been developed to correlate process parameters to component material condition. Various techniques have been applied in case of identifying component material condition as for example 3D surface texture approach by Stephenson and Ni [4], cutting force features identification method by Janić and Sortino [5], and acoustic emission features identification method by Yen et al. [6], among others. Wang et al. [7] and Castejón et al. [8] proposed computer vision and statistical

learning approaches to estimate and classify tool wear levels, using linear discriminant analysis to show that three image descriptors could effectively estimate tool wear. Attanasio et al. [9] compared artificial neural networks (ANN) with response surface methodology for predicting tool flank wear, finding that an empirical approximation function and trained ANN provided the best results. A probabilistic neural network was developed by da Silva et al. [10] to classify wear conditions based on acoustic emission and cutting power signal features. For CNC milling machines, Hesser et al. Hesser and Markert [11] used a supervised classification approach, training an ANN with acceleration data to classify tool state, supported by computational intelligence and big data analysis. More recently An et al. [12] established a data-driven model with convolutional and stacked long short-term memory (LSTM) networks, comparing it with support vector regression (SVR), random forests (RF), and feed-forward neural networks. The LSTM method proved effective in tracking tool wear evolution and predicting its remaining useful life. LSTMs are known for good handling of sequential data and are now used in industrial and manufacturing applications. Application of LSTM in various engineering domains, such as predicting the attitude and position of underground drilling machines, is a clear evidence to their adaptability and efficiency in handling sequential data, see e.g. Niu et al. [13]. Nguyen and Medjaher [14] used LSTMs to predict turbofan engine system failure probabilities, Zhang et al. [15] applied LSTMs to assess bearing degradation states, and Cai et al. [16] proposed a hybrid information model based on LSTM for tool wear prediction, validated with the NASA Ames milling data set Agogino and Goebel [1], among others. The aim of this paper is to develop a machine learning model which can predict the material condition of the components not only for SM-SCWR. The model here developed is based on the data features which are widely shared among various processes as for example cutter material abbreviation in milling process or material corrosion layer development under high pressure conditions. Therefore, this approach could also be directly applied for predictions of other sequential data, as for example materials corrosion data, required for the design and safety evaluation of SM-SCWR. To the author's knowledge, cyclic time series of corrosion-fatigue data in water under supercritical pressure conditions are not publicly available in the literature. Such data are also not yet available within the current stage of project ECC-SMART [2]. Therefore, our model will be applied and validated using the well defined NASA Ames milling data set, by Agogino and Goebel [1] which provides large amount of experimental data on milling machine tool wear material condition during operation.

The main contribution of this paper is the optimized LSTM model which aims to enhance modeling performance and efficiency. Predictive performance of the standard LSTM model is compared with predictions by optimized LSTM modeling approach and validated using the NASA milling data set by Agogino and Goebel [1]. Results prove the capability of the optimized LSTM model to predict material condition of the milling tool while improving performance and efficiency.

The remainder of the paper is organized as follows: Section 2 describes recurrent neural networks and the standard LSTM approach. In Section 3 optimized LSTM method is presented. Section 4 describes the data preprocessing and statistical data evaluation required for the approach. In section 5 predictive performance of the standard and optimized LSTM models is presented. Finally, Section 6 provides a summary and conclusion.

2. Neural Network-Based Time Series Predictions

2.1 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed for processing sequential data. Unlike traditional feedforward neural networks, RNNs have connections that form directed cycles, allowing information to persist across time steps. This makes RNNs particularly

effective for tasks involving temporal sequences, such as time-series forecasting. The fundamental building block of an RNN is the recurrent cell, which takes as input the current data point x_t and the hidden state h_{t-1} from the previous time step. The hidden state is updated using the following equations:

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b_h) y_t = W_y h_t + b_y$$

where W_h, W_x, and W_y are weight matrices, b_h and b_y are biases, and tanh is the activation function. The hidden state h_t captures the information from previous time steps, enabling the network to learn temporal dependencies. RNNs have been widely applied in various fields, demonstrating their versatility and power in handling sequential data. Applications include language modeling see e.g. Mikolov et al. [17], machine translation see e.g. Sutskever et al. [18], and speech recognition see e.g. Graves et al. [19] among others. Despite their effectiveness, RNNs suffer from limitations such as the vanishing gradient problem, which makes learning of long-term dependencies difficult. Long Short-Term Memory (LSTM) networks are extension of RNNs which have been proposed to address these issues by introducing gating mechanisms that regulate the flow of information.

2.2 Long Short-Term Memory (LSTM) Method

LSTM networks, first introduced in Hochreiter and Schmidhuber [20], are a type of recurrent neural network (RNN). Traditional RNNs struggle with long-term dependencies due to the vanishing gradient problem, where gradients used for learning diminish exponentially as they are propagated back through time, making it difficult for the network to learn long-term dependencies. LSTMs address this issue with a unique structure composed of memory cells, each containing three gates: the input gate, the forget gate, and the output gate. In the following we describe standard LSTM approach for completeness.

Forget Gate: Decides how much of the previous cell state should be retained

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f).$$

Input Gate: Controls the extent to which new information flows into the cell state

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i).$$

Candidate Cell State: Computes the new candidate cell state based on the input and the previous hidden state

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C).$$

Cell State: The cell state acts as a conveyor belt that runs through the entire sequence

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t.$$

Output Gate: Determines the output of the cell based on the cell state and the input

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o).$$

Hidden State: The hidden state is the output of the LSTM unit, computed based on the cell state and the output gate

$$h_t = o_t * \tanh(C_t).$$

Where x_t is the input at time step t; $h_{t-1}$is the hidden state at the previous time step; C_{t-1} is the cell state at the previous time step; W_f, W_i, W_C, W_o are weight matrices for the forget, input, candidate

cell state, and output gates, respectively; b_f, b_i, b_C, b_o are bias vectors for the forget, input, candidate cell state, and output gates, respectively; σ is sigmoid activation function; tanh is hyperbolic tangent activation function and $*$ assign the element-wise multiplication. This allows LSTMs to maintain and update information over long sequences, making them highly effective for various tasks such as sequence prediction and time-series forecasting.

However, capturing the information over long sequences, LSTMs will require a large number of parameters and computational resources and increase also the risk of overfitting. To address these issues, we propose an optimized LSTM model in the following section.

3. Optimized LSTM

In this study, we implement an optimized Long Short-Term Memory (LSTM) model. Our optimized LSTM model simplifies the standard LSTM architecture while preserving its ability to capture long-term dependencies. The optimized model is designed to reduce the number of parameters and to optimize the computation graph while maintaining or enhancing predictive accuracy. For the optimization, the key requirement is the self-similarity of the statistical properties of the process data.

The new model is developed in two stages: first, based on the standard LSTM architecture we will explicitly compute the gate values and use a single weight matrix for all gates; and in the second stage we apply matrix factorization techniques.

Stage 1: Model architecture In the first stage, we optimize the standard LSTM by:

- Combining the input and hidden state transformations into a single operation for each gate;
- Using a single weight matrix for all gates and the candidate cell state, reducing the number of parameters.

Based on this, the optimized LSTM computes:

$$[i_t, f_t, o_t, \tilde{c}_t] = \sigma(W \cdot [x_t, h_{t-1}] + b) \tag{1}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3}$$

where W is a single weight matrix used for all gates and the candidate cell state, b is a bias vector, i_t, f_t, o_t are the input, forget, and output gates respectively, $\tilde{c}_t$ is the candidate cell state, c_t is the new cell state, h_t is the new hidden state and $\odot$ denotes element-wise multiplication.

Stage 2: Matrix Factorization Building upon the simplified architecture from Stage 1 we apply matrix factorization which is similar to the factorization introduced in [21]. In the Factored LSTM, we replace the possibly large matrix multiplication with two smaller matrix multiplications:

$$z_f = U_f(W_f x_t) \tag{4}$$

$$z_i = U_i(W_i h_{t-1}) \tag{5}$$

$$[i_t, f_t, o_t, \tilde{c}_t] = \sigma(z_f + z_i + b) \tag{6}$$

where $W_f \in \mathbb{R}^{d_x \times r}$, $U_f \in \mathbb{R}^{r \times 4d_h}$, $W_i \in \mathbb{R}^{d_h \times r}$, and $U_i \in \mathbb{R}^{r \times 4d_h}$. Here, d_x is the input dimension, d_h is the hidden state dimension and r is the rank of the factorization. The cell state and hidden state updates remain the same as in Stage 1.

By combining architectural simplification with matrix factorization, this LSTM variant seeks to enhance the model's ability to capture long-term dependencies in time series data while improving computational efficiency.

Key Optimizations

- **Unified Gate Computation**: A single operation computes all gates and the candidate cell state.
- **Reduced Parameter Count**: Using a single weight matrix in Stage 1 significantly reduces the number of parameters.
- **Matrix Factorization**: Further reduces parameters from $O(d_x d_h + d_h^2)$ to $O(r(d_x + d_h))$, where typically $r < \min(d_x, d_h)$.
- **Simplified Computational Graph**: The optimized architecture potentially leads to faster training and inference.

Main goal of the optimization is to reach the following advantages:

- Reduced Overfitting: Fewer parameters can lead to better generalization, especially on smaller datasets.
- Lower Memory Requirements: The optimized model requires less memory to store.
- Potential for Faster Computation: Both the simplified architecture and factored computations may lead to faster processing.
- Maintained Core Functionality: The essential gating mechanisms and information flow of LSTM are preserved.

The self-similarity property of the process data is the base for these optimizations. On the other hand, this may become a disadvantage of the approach in case the data do not have self-similarity property.

Also, the optimal configuration, including the rank r for factorization, is problem-dependent and may require tuning to balance model size and performance.

This task will be tackled by applying the statistical evaluation of the process data in the following section. We first preprocess the experimental data and perform statistical evaluation to identify some relevant features and patterns in the time series data. This step is crucial for the model setup.

4. Data Preprocessing and Statistical Analysis

4.1 Data Preprocessing

For validation of the LSTM and optimized LSTM model, we use the NASA Ames milling data set Agogino and Goebel [1]. The data set represents experiments conducted on a milling machine under various operating conditions. The experiments were performed with the Matsuura machining center MC-510 V. A 70 mm face mill with six KC710 inserts was chosen as a tool. In the data base different cases with different time interval length for each runs are provided. The number of runs depend on the degree of flank wear, which was measured at irregular intervals between runs up to (and sometimes beyond) a wear limit. Specific experimental conditions are described in Table 1. Some of the cases were repeated under similar conditions as can be seen in Table 1 (for example 1 and 9).

Table 1: Experimental conditions of milling data set

Case	Depth of Cut (mm)	Feed (mm/rev)	Material
1	1.5	0.5	1-cast iron
2	0.75	0.5	1-cast iron
3	0.75	0.25	1-cast iron
4	1.5	0.25	1-cast iron
5	1.5	0.5	2-steel
6	1.5	0.25	2-steel
7	0.75	0.25	2-steel
8	0.75	0.5	2-steel
9	1.5	0.5	1-cast iron
10	1.5	0.25	1-cast iron
11	0.75	0.25	1-cast iron
12	0.75	0.5	1-cast iron
13	0.75	0.25	2-steel
14	0.75	0.5	2-steel
15	1.5	0.25	2-steel
16	1.5	0.5	2-steel

In the data set [1] and in this paper, the flank wear data is assigned as VB. Figure 1 shows the experimental flank wear abbreviation data VB. Flank wear was not always measured and no entry was made during periods when no measurements were taken, resulting in gaps in the data sequence (Figure 1: experimental values VB, ●). To address this, we employ linear interpolation of VB values (Figure 1: interpolated experimental values InterVB, ●). Notice that the large jumps in the VB data, Figure 1, are due to a start of the next experimental run, where the VB is measured for a new cutting tool.

Interpolated data are then normalized to standardize the scale of the target variable. We used the z-score normalization

$$VB_n = \frac{VB - \mu}{\sigma},$$

where μ is the mean and σ the standard deviation. This normalization step ensures that the VB values are on a consistent scale. The normalized data are used for the statistical analysis and for the models training. In the data set [1] and in this paper, the time series over the repeating experiments is assigned as cases.

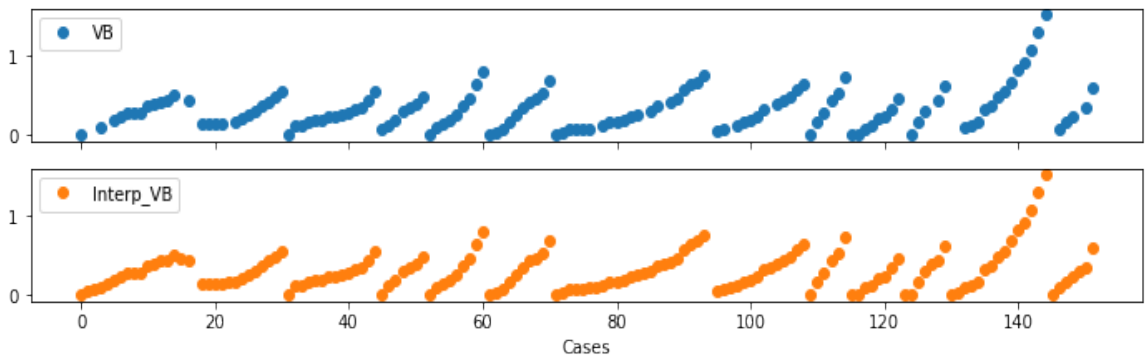


Figure 1: Flank wear abbreviation data VB from [1]

Based on the experimental data, we perform now data preprocessing and statistical analysis required for the model setup.

4.2 Statistical Analysis and Window Size

The Autocorrelation Function (ACF) measures the correlation between different points in a time series as a function of the lag between them. It is used to identify the degree of similarity between observations as a function of the time lag separating them. The ACF at lag k for a time series X_t is given by:

$$\rho_k = \frac{\sum_{t=1}^{n-k}(X_t - \bar{X})(X_{t+k} - \bar{X})}{\sum_{t=1}^{n}(X_t - \bar{X})^2}$$

where X_t is the value of the time series at time t, $\bar{X}$ is the mean of the series, and n is the number of observations. It is well known that the ACF helps in understanding the structure and pattern in the data, which is crucial for selecting the appropriate window size in LSTM models, see e.g. Box et al. [22].

The Partial Autocorrelation Function (PACF) measures the correlation between a time series and its lagged values, with the influence of intermediate lags removed. PACF function is particularly useful for identifying the direct effect of past values on the current value, excluding the indirect effects through other lags. The PACF at lag k is the coefficient ϕ_{kk} in the autoregressive model of order k

$$X_t = \alpha + \sum_{i=1}^{k} \phi_i X_{t-i} + \varepsilon_t$$

where X_t is the value of the time series at time t. α is the intercept term, representing the mean level of the time series if the series is stationary. ϕ_i (for $i = 1, 2, \ldots, k$) are the autoregressive coefficients. X_{t-i} are the lagged values of the time series. ε_t is the white noise error term at time t. The PACF helps in determining the order of the autoregressive process and is essential for understanding the direct influence of past values. When using LSTM and models for time-series forecasting, the choice of window size (i.e., the number of past observations considered) is critical. The ACF and PACF provide insights into the appropriate window size.

- ACF can reveal the lags at which the data points are significantly correlated. The ACF show how the correlation between any two values of the series changes as their separation changes. Significant spikes in the ACF suggest important lags to consider for the window size. For LSTM models, the window size can be chosen based on the lag where the ACF significantly decreases, indicating the point beyond which past values have less influence on future values.

- PACF helps in identifying the most significant lag values that directly affect the current value. The PACF measures the correlation between a time series and its lagged values, controlling for the values of the time series at all shorter lags. Significant spikes in the PACF plot indicate important lags that directly influence the current value, without intermediate effects. This is useful for LSTM models to select a window size that captures the most important past values without including redundant information.

By combining insights from ACF and PACF, one can determine an optimal window size that balances capturing necessary information and maintaining computational efficiency. This practice ensures that LSTM models are trained on relevant past data, improving their predictive performance.

Figure 2 shows the ACF and PACF plots of the VB data. The ACF plot shows first significant but later slow decrease indicating long-range dependencies in the data. Similarly Partial Autocorrelation

Function (PACF) shows significant initial spike and a slow decrease later on. Since both ACF and PACF show a slow decrease after the initial lag, the window size should be long enough to include the decay range.

However, while a larger window size allows the model to learn from a broader history of the time series, it also increases computational complexity and the risk of overfitting.

This is the point where the optimized LSTM model can provide advantage. We can choose large window size to capture long-range dependencies in the data while significantly reducing the number of parameters. In the following we compare the performance of the optimized LSTM model against the standard LSTM counterpart to quantify the impact and performance of these optimizations.

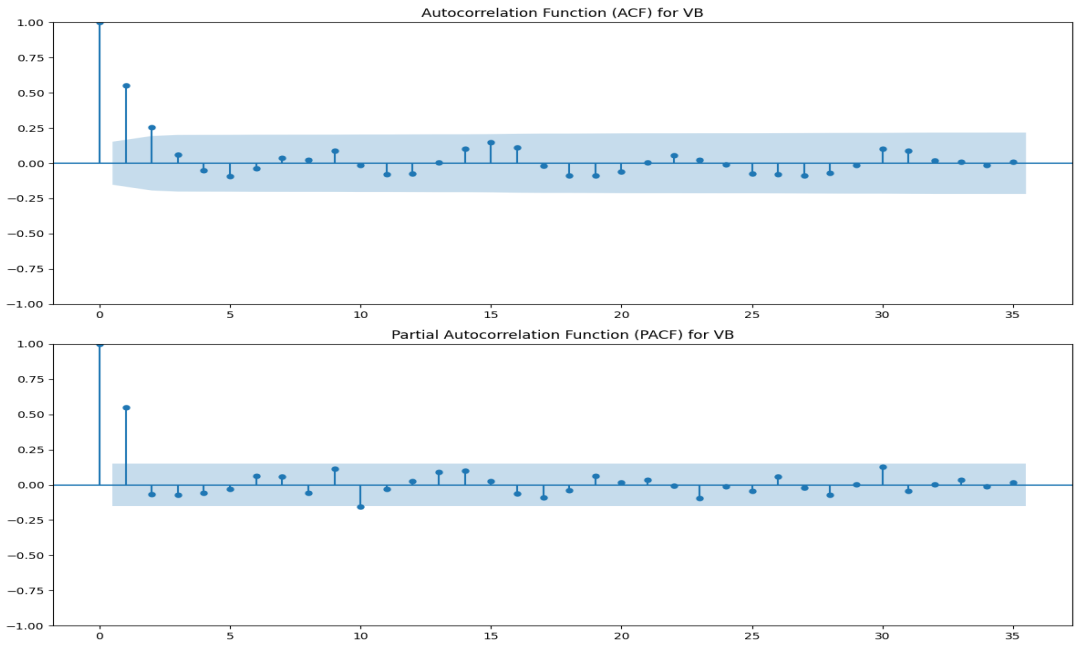


Figure 2: ACF, PACF for VB

5. Results

We configure both the standard deep LSTM model and its optimized variant as specified in the preceding sections. The architecture for both models is identical: First layer: 16 neurons; Second layer: 16 neurons; Output Dense layer: 1 neuron. Both models operate in autoregressive multistep prediction mode with a window size of 34 steps. Table 2 summarizes the standard deep LSTM model, while Table 3 presents the optimized deep LSTM model. Comparing the parameter count in tables 2 and 3 reveals that the optimized LSTM model has approximately 40 % fewer parameters compared to the standard LSTM model. This reduction in the number of parameters lead to faster training and inference times, as well as reduced memory requirements, see Figure 3. Standard and optimized LSTM models are trained on the preprocessed data as described previously. During the training of our models, we monitored two metrics: the Loss function and the Mean Absolute Error (MAE). The Loss function used in our models is the mean squared error defined as

$$\text{Loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where n is the number of samples, y_i is the true value, and $\hat{y}_i$ is the predicted value. The Mean Absolute Error measures the average absolute difference between the predicted and actual values:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MAE provides a more interpretable metric in the original scale of the target variable. The training history of the standard and optimized LSTM models are shown in Figure 3 to show the evolution of these metrics over epochs. Figure 3 indicate that the optimized LSTM model converges faster.

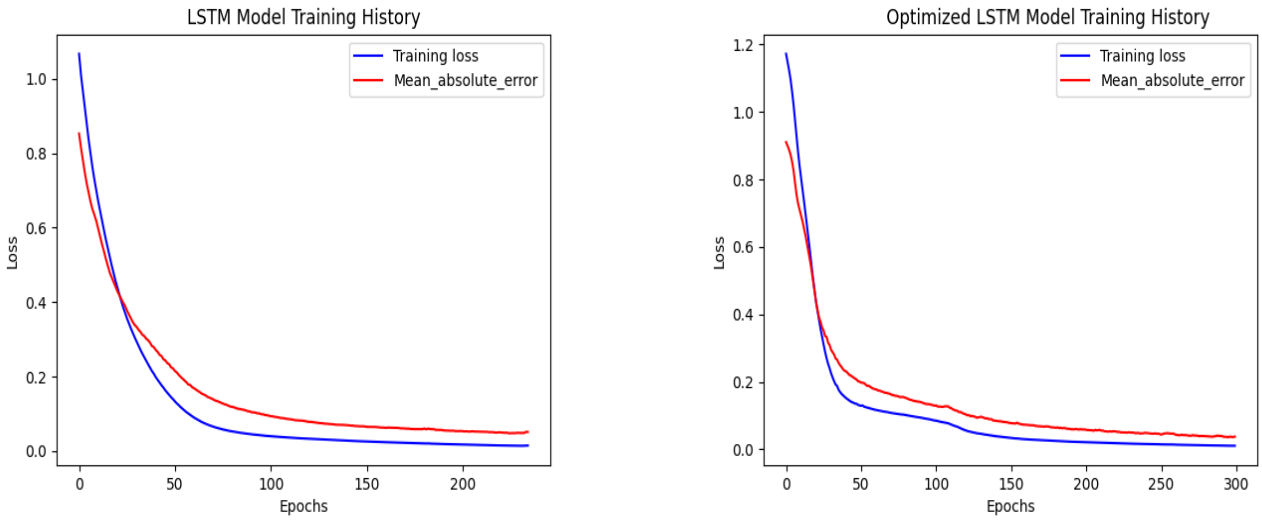
Table 2: LSTM Model Summary (Total number of parameters: 4305)

Layer	Type	Parameters
lstm	LSTM	2176
lstm_1	LSTM	2112
dense	Dense	17

Table 3: Optimized LSTM Model Summary (Total number of parameters: 2713)

Layer	Type	Parameters
Optimized_lstm	OptimizedLSTM	1352
Optimized_lstm_1	OptimizedLSTM	1344
dense	Dense	17

We aim to predict the component material condition at time t + n based on monitored sample data up to time t. The optimized LSTM model is expected to capture the long-range dependencies in the data while significantly reducing the number of parameters.



(a) LSTM training history

(b) Optimized LSTM training history

Figure 3: Training histories comparison

Predictions of flank wear abbreviation process using the LSTM and optimized LSTM methods are presented and compared with the reference experimental data VB in Figure 4. The experimental

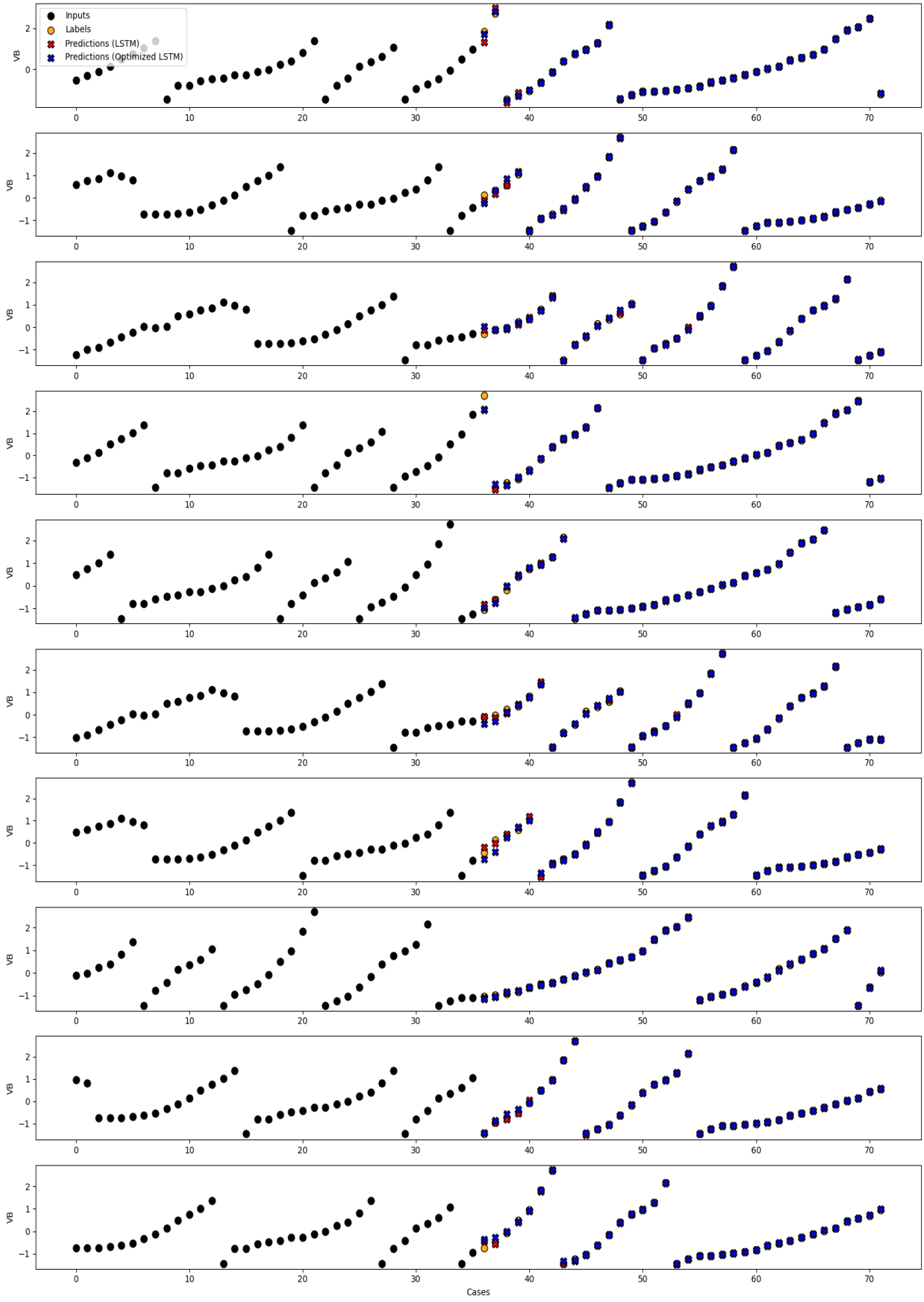


Figure 4: Predictions by LSTM and Optimized LSTM

data used for the network training are called Input and assigned with ■. The experimental data used for the validation are called Labels and assigned with ● in the Figure 4. The X assign predictions by the LSTM model, and X the optimized LSTM predictions in the Figure 4. We generate a figure containing 10 subplots. Each subplot in the Figure 4 represents a different window of the time series data, progressing through the test set. This visualization allows us to assess the model's prediction performance across various segments of the data. As we move from the first subplot to the last: Each subplot represents a later segment of the time series data; The input window (■) shifts forward in time; The prediction window (X and X) also shifts forward. This progression allows us to visualize how the model's prediction accuracy is performing across different parts of the time series.

Based on the performed statistical analysis and window size evaluation the results show that, apart of very few points, the optimized LSTM model provides same or improved performance in predictions of the data.

6. Summary and Conclusion

In this paper, deep LSTM neural network architecture is optimized to improve performance and efficiency. Based on the NASA Ames milling data set Agogino and Goebel [1] the performance of the new approach was compared with the standard LSTM method. Based on the performed statistical analysis and window size evaluation, results suggest that both models, LSTM and optimized LSTM, can precisely predict the material condition of the milling tool. Results also show that the optimized LSTM model can improve the performance of the standard LSTM model while significantly reducing the number of parameters and therefore increasing the efficiency. The modeling approach can be directly applied for predictions of other sequential data, as for example materials corrosion data, required for the design and safety evaluation of SM-SCWR.

7. Acknowledgements

This research is partially supported by the European Commission under the Grant Agreement n°945234 (ECC-SMART).

References

[1] A. Agogino and K. Goebel. Milling data set. best lab, uc berkeley. milling data set, nasa prognostics data repository, nasa ames research center, moffett field, ca. `http://ti.arc.nasa.gov/project/prognostic-data-repository`, 2007.

[2] ECC-SMART. Joint European Canadian Chinese development of Small Modular Supercritical Water Reactor (SM-SCWR) technology, supported by the European Commission under the grant agreement n°945234. `https://ecc-smart.eu/`, 2020-2025.

[3] Thomas Schulenberg and Ivan Otic. Concept of a small modular scwr with horizontal fuel assemblies. *Journal of Nuclear Engineering and Radiation Science*, 8(3):031104, 2022.

[4] David Stephenson and Jun Ni. A multifeature approach to tool wear estimation using 3d workpiece surface texture parameters. *Journal of Manufacturing Science and Engineering*, 132, 12 2010. doi: 10.1115/1.4002852.

[5] M. Janić and M. Sortino. Twem, a method based on cutting forces - monitoring tool wear in face milling. *Int. J. Mach. Tool. Manuf.*, 45(1):29–34, 2005.

[6] C.L. Yen, M.C. Lu, and J.L. Chen. Applying the self-organization feature map (som) algorithm to ae-based tool wear monitoring in micro-cutting. *Mech. Syst. Signal Process.*, 34(1–2):353–366, 2013.

[7] W.H. Wang, G.S. Hong, and Y.S. Wong. Flank wear measurement by a threshold independent method with sub-pixel accuracy. *Int. J. Mach. Tool. Manuf.*, 46(2):199–207, 2006.

[8] M. Castejón, E. Alegre, J. Barreiro, and L.K. Hernández. On-line tool wear monitoring using geometric descriptors from digital images. *Int. J. Mach. Tool. Manuf*, 47(12–13):1847–1853, 2007.

[9] A. Attanasio, E. Ceretti, C. Giardini, and C. Cappellini. Tool wear in cutting operations: experimental analysis and analytical models. *ASME J. Manuf. Sci. Eng.*, 135(5):051012, 2013.

[10] R.H.L. da Silva, M.B. da Silva, and A. Hassui. A probabilistic neural network applied in monitoring tool wear in the end milling operation via acoustic emission and cutting power signals. *Mach. Sci. Technol.*, 20(3):386–405, 2016.

[11] D.F. Hesser and B. Markert. Tool wear monitoring of a retrofitted cnc milling machine using artificial neural networks. *Mater. Lett.*, 19:1–4, 2019.

[12] Q. An, Z. Tao, X. Xu, M.E. Mansori, and M. Chen. A data-driven model for milling tool remaining useful life prediction with convolutional and stacked lstm network. *Measurement*, 154:107461, 2020.

[13] Dongxiao Niu, Lijie Sun, Min Yu, and Keke Wang. Point and interval forecasting of ultra-short-term wind power based on a data-driven method and hybrid deep learning model. *Energy*, 254: 124384, 2022.

[14] Khanh T.P. Nguyen and Kamal Medjaher. A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliability Engineering and System Safety*, 188: 251–262, 8 2019. ISSN 09518320. doi: 10.1016/j.ress.2019.03.018.

[15] Bin Zhang, Shaohui Zhang, and Weihua Li. Bearing performance degradation assessment using long short-term memory recurrent network. *Computers in Industry*, 106:14–29, 4 2019. ISSN 01663615. doi: 10.1016/j.compind.2018.12.016.

[16] Weili Cai, Wenjuan Zhang, Xiaofeng Hu, and Yingchao Liu. A hybrid information model based on long short-term memory network for tool condition monitoring. *Journal of Intelligent Manufacturing*, 31:1497–1510, 8 2020. ISSN 0956-5515. doi: 10.1007/s10845-019-01526-4.

[17] Tomas Mikolov, Martin Karafiát, Lukáš Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. *Interspeech*, 2:1045–1048, 2010.

[18] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[19] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649, 2013.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[21] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for lstm networks, 2018. URL https://arxiv.org/abs/1703.10722.

[22] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. Time series analysis: forecasting and control. 2015.