Deep Learning Based Fatigue and Vibration Analysis Using Sensor Data

Manuel Mühlhaus[1], Daniel Kreuter[2], Kian K. Sepahvand[3]

[1] *TUM student, 85748 Garching , Germany, Email: manuel.muehlhaus@gmail.com*

[2] *Bosch GmbH, 70465 Stuttgart, Germany, Email: DanielChristopher.Kreuter@de.bosch.com*

[3] *Chair of Vibroacoustics of Vehicles and Machines, 85748 Garching, Germany, Email: k.sepahvand@tum.de*

Abstract

Shaker tables are used for high-frequency fatigue testing of components. In piezo-shaker tests, the component is excited with input accelerations at different points, and the output acceleration is measured, e.g., at the circuit board of the component. However, piezo-shaker tests for various loads are expensive and laborious. Machine Learning (ML) algorithms promise to be a suitable method for simulation. The main task is using Neuronal Networks as a virtual sensor to derive a good approximation of the resulting vibration. Existing piezo tests can be used as the training data to get a good prediction of the output time series (TS) for different input loads and temperatures. The predicted time series allows us to determine an estimation of the pseudo damage inflicted by the vibration. Accordingly, the method results in less bench test time and experimental effort. The resulting output time series has matched statistics like Root Mean Square (energy) and also matched the spectrum of the true TS well. Nevertheless, when predicting output accelerations at untrained temperature areas, a time shift is noticed, making rating models with the help of validation loss impossible. Because of that, a new metric, the Mean Percentage Window Energy Error (MPWEE), is introduced, comparing energy deviations between small windows of size 1000 (equivalent to 0.02 second windows). The best ML models are verified with a classical nominal mean stress approach comparing the pseudo damage inflicted by the original and predicted time series from the ML models.

State of the Art

Convolutional Neural Networks (CNNs) have significantly contributed to increasing the accuracy of regression-based methods. It has been shown that CNNs can be used as virtual sensors to approximate vibration responses at different locations to reduce the number of sensors needed [1]. In general, methods that work well for time series forecasting seem to do well as virtual sensors. This makes sense since the signals are of high resolution in the time domain resulting in adjacent values being similar. Another approach is to wavelet transform the time series and use the frequency domain information as input. Time series data is driven by multiple latent patterns which occur at different frequencies. Many existing approaches to time series forecasting fail to identify these frequency-domain components [2]. This can be done with scalograms. They show a visual representation of the spectrum of frequencies of a signal as it varies with time. These images then can be analyzed by CNNs. Mainly for time series classification spectrograms have been used with great success as input data [3] [4]. Wang et al. [5] presented the use of wavelet scalogram images as an input into a CNN to detect faults within a set of vibration data. Zhao et al. [2] present time series forecasting with scalograms from wavelet transforms. This paper discusses different network architectures like 1D-CNN with different inputs or 2D-CNN getting scalograms as input and then compares the results.

Data preprocessing

We got data from performing different maneuvers with the piezo shaker table (see Fig. 1) on car components. The shaker table has excited the component with 10 input accelerations - Input_1 to Input_10 - at different locations in the direction of the x, y, z-axis. Then the resulting accelerations at a point of interest like at the support of the component were measured. The input excitations simulated different typical car maneuvers like curve cruise, emergency braking etc. The dataset contains 4 typical maneuvers repeated at 5 different temperatures, namely -5°C, 25°C, 50°C, 82°C, 115°C. Figure

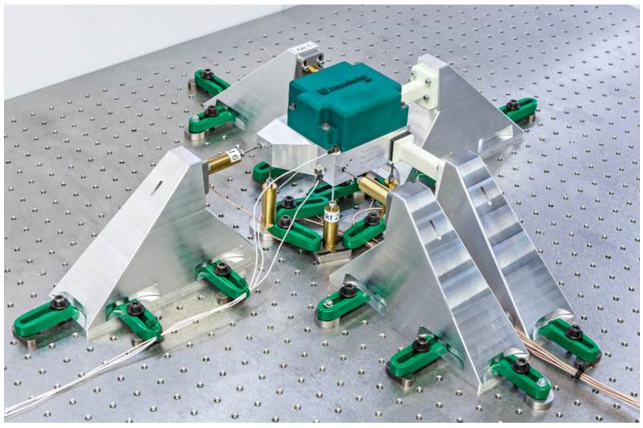

Figure 1: Piezo-shaker table exciting the component with 10 input accelerations and measuring the output accelerations.

2 shows one maneuver of raw data. The accelerations - Input _1 - Input _10 and Output - are plotted scaled between -1 and 1. Here, Temp is the temperature measured over time and stays rather constant in one maneuver. The output is the resulting acceleration in the z-axis at the circuit board and is also scaled between -1 and 1. To speed up the training process, we focus on the z-axis output acceleration and neglect the resulting acceleration in the x and y-axis. The data was bandpassed and idle time was removed. Afterwards, we reduced the number of inputs from 10 to 6 with the feature engineering method SelectKBest.

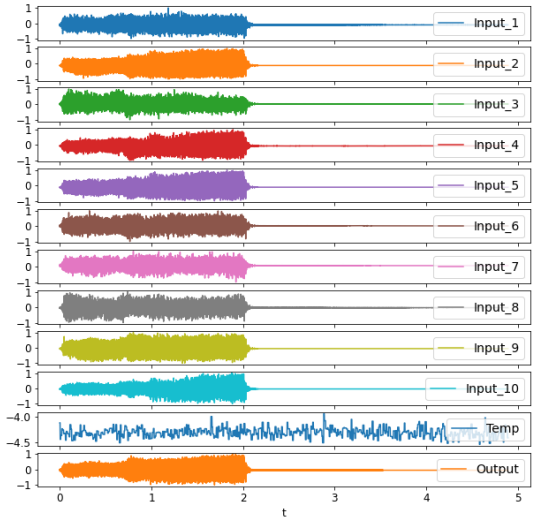

Figure 2: Starting point: Raw data from Bosch of one maneuver.

Before feeding the data to a CNN, the data is standardized. Next, the data needs to be rolled in windows for a 1D CNN. This is done by rolling the time series taking the 699 past values and the present value of all inputs to predict the current output. As we also wanted to test frequency domain information, scalograms have been created and saved as numpy arrays as inputs to the CNNs.

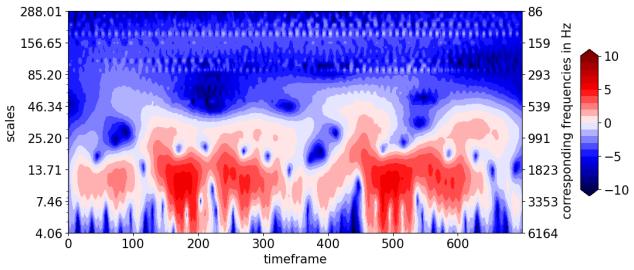

Figure 3: Contour plot of a scalogram showing the power values of the TS with a $\log_2$. The used time window is 700. The scales of the wavelets are chosen to depict the corresponding frequency band between 86-6164 Hz and are displayed logarithmically as the y-axis. Here a 3rd order complex Gauss wavelet was chosen.

Results

To create and optimize the CNNs Keras tuner was used. The inputs are fed into the nets starting with into a convolution block alternating CNN layers and pooling layers. The output is fed into a dense block and then it is decided if the temperature is added as an extra input. Another dense block then calculates the predicted output acceleration.

The subplot in the left of Fig. 4 shows the predicted and true validation acceleration time series. The true TS contains the 8 measured validation maneuvers appended behind each other. The y-axis of the plots is unit-less as the accelerations have been standardized. The predicted TS is overall very similar to the true TS. The subplot on the upper right compares the true and predicted spectrum. The amplitude spectrum is plotted over the frequency between 90 Hz and 5000 Hz as that is the interval of the bandpass filter. The spectrum of frequencies matches well except for the peak around 1000 Hz and the 2 peaks around 2000 Hz. The left subplot of Fig. 5 presents a

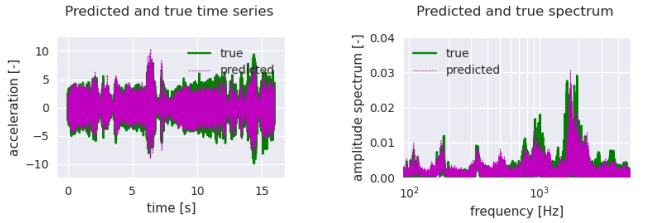

Figure 4: Comparing the predicted and true TS and spectrum of the best model from the grid search using 6 inputs without the temperature.

zoom into the time series and shows that the model is not able to predict the exact position of individual peaks of vibration, even so the RMS (Root Mean Square) and spectrum are depicted quite accurately. Trends in amplitude of acceleration are still depicted well. The data seems to be time-shifted by around 10 time steps. The time shift can be explained by the model predicting vibrations in unseen temperature areas as the temperature affects material properties.

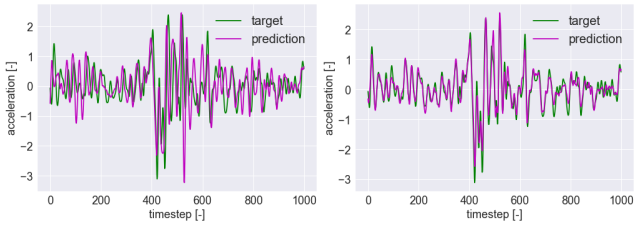

Figure 5: Comparing models predicting in trained/untrained temperature areas.

To confirm this statement, the maneuvers with temperature 25°C, 82°C were split in the ratio 70% to30% in training and validation data. Figure 5 compares the same window with the model training from data at 25°C, 82°C, and the model training from data at -5°C, 50°C and 115°C. The model training and testing with the same temperatures was able to accurately predict the phase of the vibration with the validation loss going towards zero, confirming our statement. However, the exact course of the vibration is not of interest as the goal is to predict the resulting damage statistics of the component. As long as the curve results in similar damage to the component, the model fulfills its job. That is why validation loss is a poor metric to decide on the best models as the time shift causes big errors but does not affect the damage done to the component. The Mean percentage window energy

error (MPWEE) was defined to rate model performance:

$$\mathrm{MPWEE} = \frac{1}{N}\sum_{i=1}^{N}\left|1 - \frac{\mathrm{RMS}_{\mathrm{WS},\,i}(\mathrm{true})}{\mathrm{RMS}_{\mathrm{WS},\,i}(\mathrm{pred})}\right| \qquad (1)$$

When using the scalograms as input we reduced the size of the dataset from 100.000 values to 20.000 values to keep scalogram creation and training time reasonable.

Table 1: Table of the best MPWEE values reached with different inputs. The number after TS (time series) stands for the number of input accelerations used, T refers to the temperature here and Scalo to the scalograms. Because of the long training time of network using scalograms there was no extensive grid search done for that one.

Inputs	large Dataset		
	TS(6)	TS(6)+T	TS(10)+T
MPWEE	26,86%	24,51%	21,00%
Inputs	reduced Dataset		
	TS(6)+T	TS(6)+T+Scalo	
MPWEE	32,54%	30,60%	

As the table shows using more data significantly outperformed the value of scalograms in this test.

A classical nominal mean stress approach is used to approximate the fatigue damage of the component based on the true and predicted TS. We set the material parameters: slope of the Woehler curve to $k = 5$, edge load number of cycles ND_50 $= 1.5 \cdot 10^6$, the fatigue limit SD_50 $= 5$ and the mean stress parameters to M1 $= 0.2$ and M2 $= 0.3$ and the R-Ratio =-1 (fully reverse loading) and used the method "Miner elementary" [6].

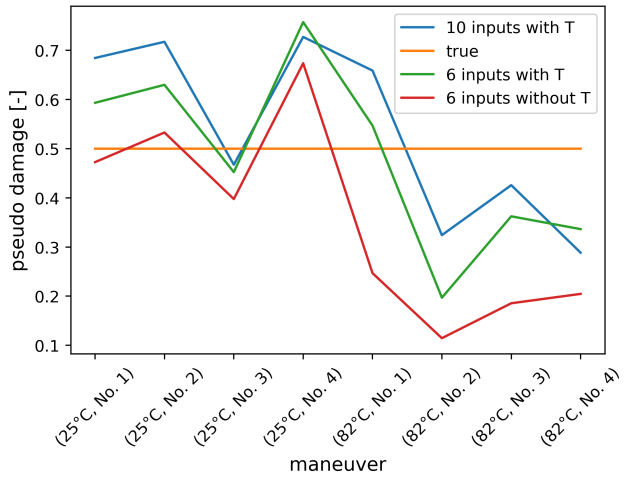

Figure 6: Pseudo damage of the TS predicted by the ML models compared to the original TS.

Figure 6 compares the pseudo damage of the original TS (nominated to 0.5) and the TS predicted by different ML models. The model with all 10 input accelerations - no SelectkBest - (10 inputs with T) performs best, followed by the model using 6 inputs and the temperature (6 inputs with T). As the pseudo damage of these maneuvers have different powers of tens the critical ones can be determined. Overall, the damage is well described by the model with 10 inputs and the temperature.

Conclusion

The output time series predicted by the ML models matches Root Mean Square and spectrum of true time series quite well but could not precisely locate the individual peaks of vibration in time. The time shift causes the validation loss to not decrease but does not affect the damage on the component. Because of that a new metric - Mean Percentage Window Energy Error (MPWEE) - was introduced to rate models.

Using scalograms to the model didn't make sense for the problem as the input dataset needed to be reduced. Nevertheless, it could be interesting for tasks, where only small amount of data is available or resources do not matter.

It was shown that ML models can approximate the acceleration time series and estimate the pseudo damage reasonably. As the pseudo damage of these maneuvers have different powers of tens the critical ones can be determined and bench-tested.

References

[1] Sun, Shan-Bin and He, Yuan-Yuan and Zhou, Si-Da and Yue, Zhen-Jiang: A Data-Driven Response Virtual Sensor Technique with Partial Vibration Measurements Using Convolutional Neural Network. Sensors 17(12) (2017)

[2] Zhao, Y. and Shen, Y. and Zhu, Y. and Yao, J.: Forecasting Wavelet Transformed Time Series with Attentive Neural Networks. IEEE International Conference on Data Mining (ICDM) (2018), 1452-1457

[3] Zeng, Zhengxin and Amin, Moeness G. and Shan, Tao: Arm Motion Classification Using Time-Series Analysis of the Spectrogram Frequency Envelopes. Remote Sensing 12(3) (2020)

[4] Verstraete, David and Ferrada, Andres and Droguett, Enrique and Meruane, V. and Modarres, Mohammad: Deep Learning Enabled Fault Diagnosis Using Time-Frequency Image Analysis of Rolling Element Bearings. Shock and Vibration 2017 (2017)

[5] Wang, J. and Zhuang, J. and Duan, L. and Cheng, W.: A multi-scale convolution neural network for featureless fault diagnosis. International Symposium on Flexible Automation (ISFA) (2016), 65-70

[6] Pylife Life time Calculation, URL: `https://pylife.readthedocs.io/en/latest/demos/lifetime_calc.html`