RESEARCH AND DEVELOPMENT OF AN ARTIFICIAL NEURAL NETWORK FOR SPECTRAL DATA

Theo Dedeken

Ghent University
Faculty of Sciences
Computer Science
Sint-Pietersnieuwstraat 25, 9000 Gent, Belgium
theo.dedeken@ugent.be

Abstract

There is a global challenge to determine the state of the inventory of transport and storage containers for high radioactive waste (spent fuel assemblies). One possible solution is to perform vibration analyses and evaluate vibration responses by using artificial neural networks. For this approach first investigations have been carried out. Vibration data is obtained from a testing setup modeling the nuclear storage fuel assemblies and is converted to the frequency domain via the Fourier transform. Raw spectral data is first prepared by normalization, data augmentation and limiting the frequency range. These measures are proven to have significant impact on the overall performance of the training of the neural networks. Using fully connected and convolutional neural networks, classification and regression is performed on the spectral data. Classification is shown to be possible with very high accuracy and regression has decent results with options for improvement in later stages. Convolutional neural networks are shown to be superior in both cases.

Keywords

Neural networks; Spectral data; Nuclear waste monitoring; Vibration analysis

Introduction

The storage of nuclear waste is a complex and worldwide problem. Currently there are little to none long term solutions available for storing spent fuel assemblies, so the waste is stored in intermediate transport and storage containers inside the nuclear power plants. These storage containers require active maintaining and monitoring to avoid dangerous and environmentally damaging situations.

To research and develop novel solutions to these problems a cooperative project was set up between Dresden Technical University (TUD) and Zittau/Görlitz University of Applied Sciences (HSZG). In light of this project different possible non-invasive measurement principles were proposed [1]. This paper focuses on the monitoring of containers using vibration analysis and training neural networks on the obtained spectral data to perform classification and assessment of the current state of the storage container inventory.

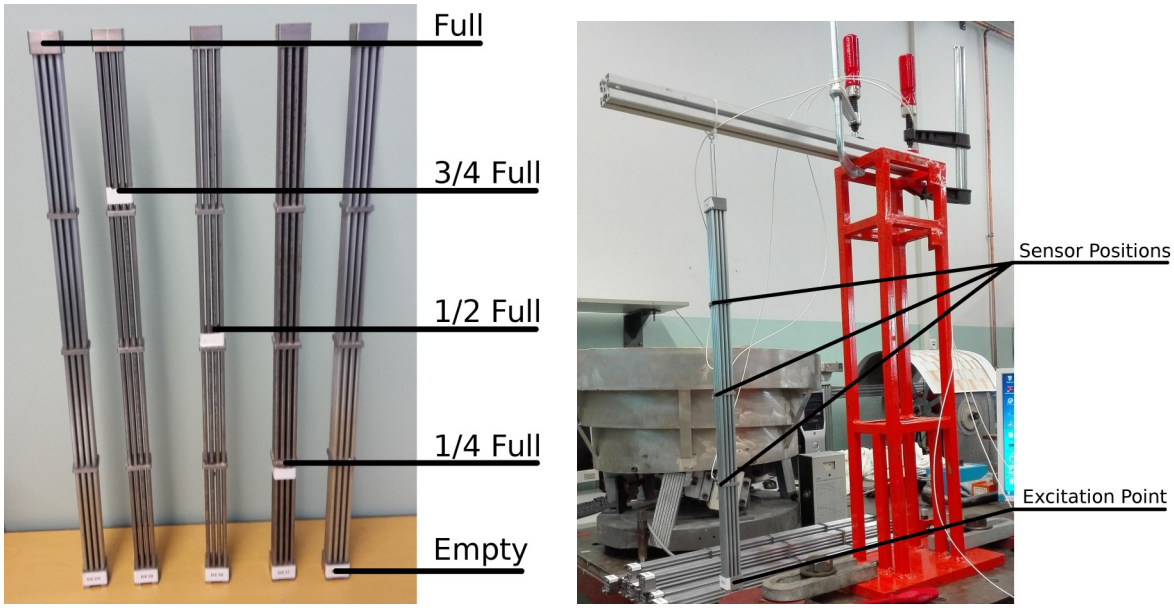
1 Methods

1.1 Data acquisition

To train a neural network, first a sufficient amount of data needs to be collected. To acquire the data necessary for further research, a testing setup was developed and measurements were performed.

Scale models representing the actual fuel assemblies contained in the storage containers were constructed. The model is assembled as a collection of 16 tubes arranged in a four by four pattern. It is possible to change the amount of filament in these tubes. This way a number of different states were constructed each with a different configuration.

Two main rounds of experiments were performed, the first round with five different states ranging from full to empty, in Fig 1 the configured states for this round are shown. Later the amount of states was increased to a total of 15. The results presented in section 2 use the data obtained from the last set of experiments.



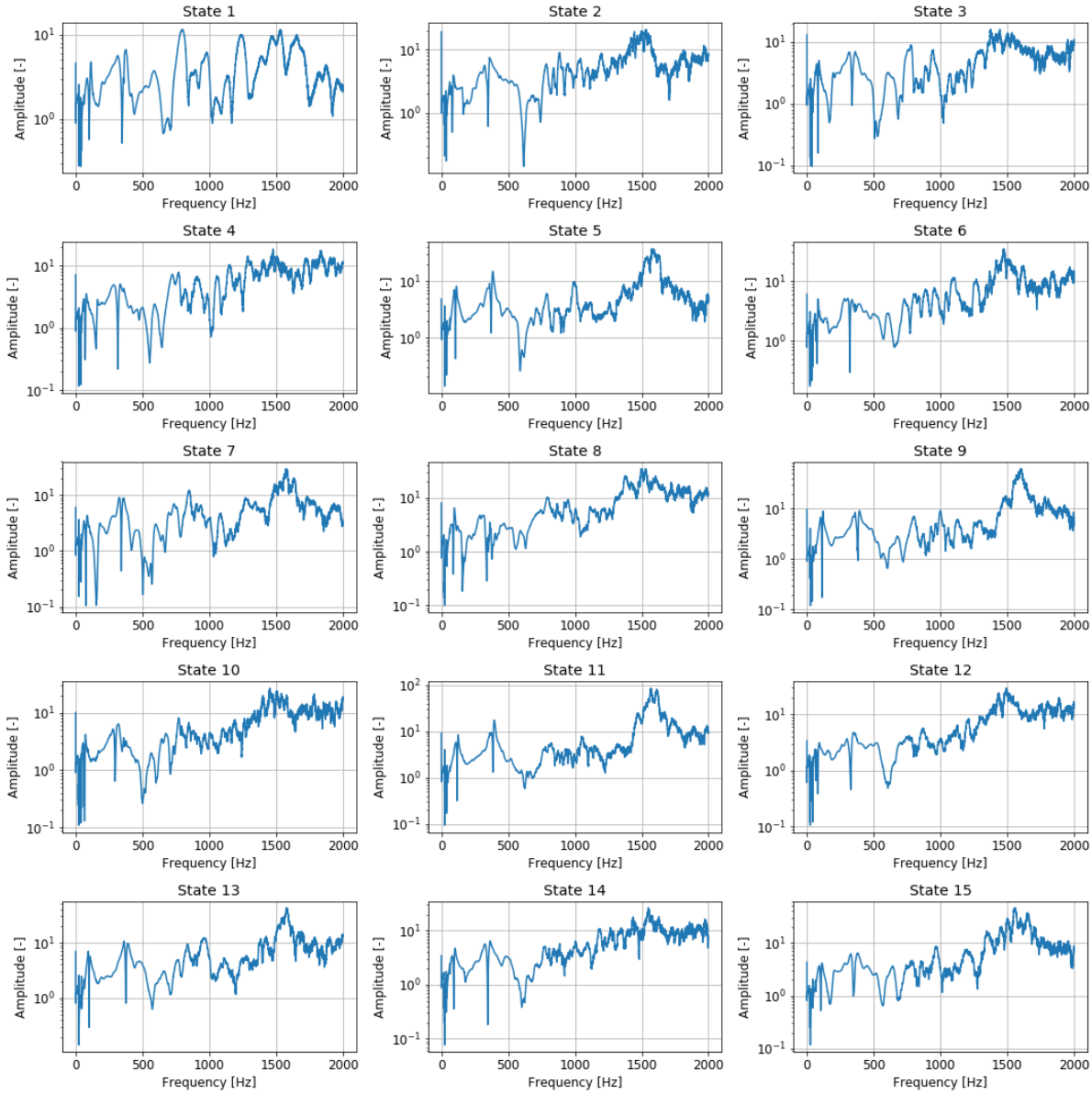
Source: Own

Fig 1: *The prepared states for the first round of experiments*

To measure the vibrations on the fuel assembly models, a testing rig was set up were the model could be suspended in the air, in Fig 1 this setup is presented. At three positions on the model sensors were attached to register the velocity of that position over time. The three sensors are positioned along the same axis as to where the impact is occurring. To perform a measurement an impact hammer was used to hit the assembly, the resulting vibrations were then registered and normalized according to the vibrations measured on the hammer. This process was repeated ten times for each state.

1.2 Data preparation

The raw data acquired from the sensors is not in a format that is useful for analysis and classification using neural networks. To better capture the unique properties of each signal we decompose the time series in the frequency domain using the Fourier transform. This leaves us with the frequency values in between 0 Hz and 2000 Hz (Fig 2).



Source: Own

Fig 2: *All obtained frequency domains for the second round of experiments*

1.2.1 Removing noise

When looking closer at the variance of the calculated amplitude of each frequency over measurements of the same state (Fig 3), it becomes clear that there are two zones in the spectrum with variance of different magnitude. The calculated amplitudes of the frequencies ranging from 0 Hz to 700 Hz are highly similar in over multiple measurements of the same state. The higher frequencies are exceptionally dissimilar over measurements of the same state. This increase in variance can mostly be attributed to noise caused both by measurement inaccuracies and outside factors.

When training a neural network on samples which are very different but still map to the same output, we run into problems where the network has difficulties to converge to a state that can handle each case of input. Keeping in mind the construction of the spectral domain we can assume that most of the original vibration data is defined by the values of the lower frequencies. To help the training performance of the used models we limit the frequency range to the interval [0 Hz, 800 Hz].

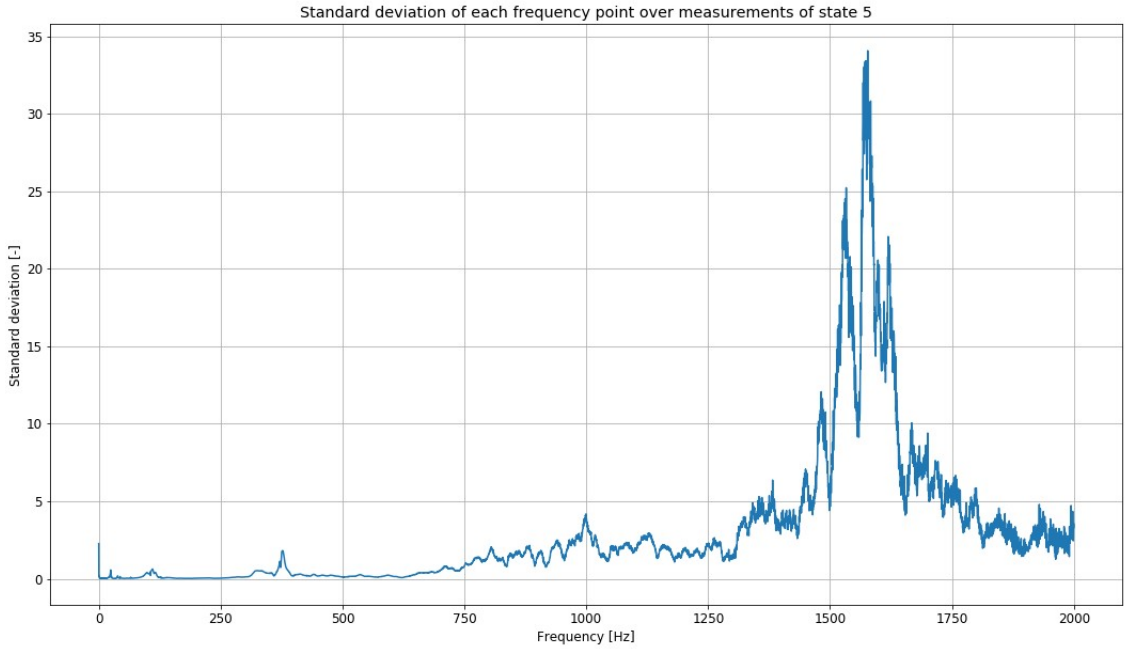


Fig 3: *Variance of the amplitude for each frequency over measurements of the same state*

1.2.2 Data normalization

Another technique used to improve training performance of the neural networks was to normalize the input data. This was achieved by subtracting the mean of the spectrum to each value and dividing by the standard deviation afterwards to obtain a data sample with mean zero and standard deviation of one.

1.2.3 Data augmentation techniques

As there were only ten measurements for each state, the amount of data to train the neural networks on was very limited. To improve the performance and robustness of the network data augmentation inspired by previous work in [2] was added. The techniques employed in this case were adding an offset to the original spectrum and multiplication of the spectrum. Offset was varied over ±0.1 times the standard deviation of the original sample and multiplication was varied over 1±0.1 times the standard deviation. For each sample nine extra samples were constructed: three using the offset technique, three using multiplication and three using the combination. This effectively increased the size of our dataset with a factor of ten. It is important to note that the augmentation was executed after the normalization of the data, otherwise the normalization operation would convert the extra samples back to the original sample.

1.3 Used models

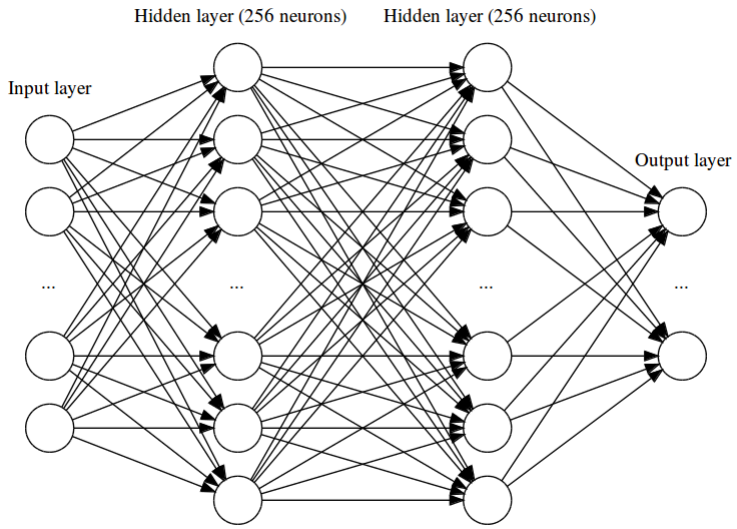
Training was performed on the two most used variants of feed-forward neural networks, the standard fully connected and convolutional neural networks. An architecture of comparable complexity for each one of the variants was constructed and trained on the prepared spectral data. See section 2.1 for the comparison between the different models.

The networks were constructed in the Keras [3] framework using the Tensorflow [4] backend. As for the optimizer, an instance of the Adam optimizer was used, configured with a learning rate of 0.0001.

1.3.1 Fully connected neural network

The fully connected network shown in Fig 4 consists of an input layer, two hidden layers and the output layer. The hidden layers have 256 neurons and are combined with the Exponential Linear Unit (ELU) activation function and a dropout measure dropping out 25% of the inputs.

The depth of the neural network was kept small as increases in layers did not cause any major improvements in performance of the network. The ELU activation function has good training performance as it does not have the same problems with vanishing gradient compared to the sigmoid activation function. Dropout is added as a measure to make the resulting network more robust and prevent overfitting of the data.

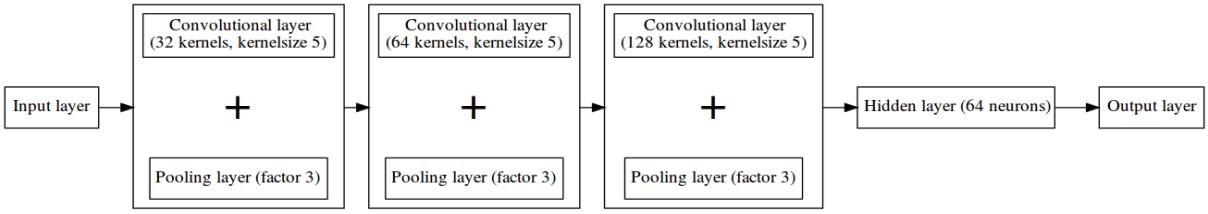


Source: Own

Fig 4: *Fully connected neural network architecture*

1.3.2 Convolutional neural network

The convolutional neural network shown in Fig 5 consists of an input layer, three convolutional layers, one hidden layer and the output layer. The three convolutional layers have 32, 64 and 128 kernels respectively. After each of the convolutional layers a max pooling operation is performed with pool size of three. At the end of these layers the input space is reduced sufficiently to be fully connected to a single hidden layer with ELU activation and a dropout measure with dropout rate of 25%.



Source: Own

Fig 5: *Convolutional neural network architecture*

1.4 Training input

In the case of the fully connected network the input of the network is constructed by concatenating the spectra after data preparation of the three positions. Each amplitude value in the resulting array then serves as an input node to the neural network.

As for the convolutional neural network, a different approach can be used. Since convolutional neural networks work on volumes rather than single input nodes we can construct the input as a two dimensional array by stacking the spectra of each position. This resembles the way convolutional neural networks are used in the case of colour images where each image pixel has three components, one for each of the colour channels.

To make sure the model was generalizing well, a validation split of 0.5 was chosen. This meant that only half of the samples were used in the training of the network. With the addition of data augmentation, the models still had sufficient data to be trained on.

1.5 Output encoding

Two different types of encoding was used for the expected output for the networks, each of them more appropriate in different approaches of solving this problem.

1.5.1 One-hot encoding

To convert the state number to this encoding a n-dimensional vector was constructed (n being the number of states: 5 or 15). In this vector all elements are zero except for the element at the position of the state being encoded.

This encoding was used to train the networks to do a classification of the input sample. A property of this encoding is that we can interpret the output as a distribution of how confident the network is about its prediction, the more confident, the more the input will resemble the one-hot encoding. If multiple positions in the output are high, we can presume that the network recognizes properties from both states in the input. To get the output in the desired format, a softmax activation layer was added as the last layer of the models. With this output format the categorical cross entropy between the expected output and the actual output was used as the loss measure while training.

1.5.2 Continuous value

Another way to encode the state of a particular sample is to use a single value. In this case the state number is converted to a number in the range [0,1]. This is done by linearly interpolating the states number to the new range.

This encoding was used when solving this problem as a regression problem. As the states are numbered 1-15 going from empty to full, we can interpret the output as a value representing the amount of filling in the assembly. To limit the range of the output of the neural network model, a sigmoid activation layer was added as the last layer of the model. With this output format the mean squared error between the expected output and the actual output was used as the loss measure while training.

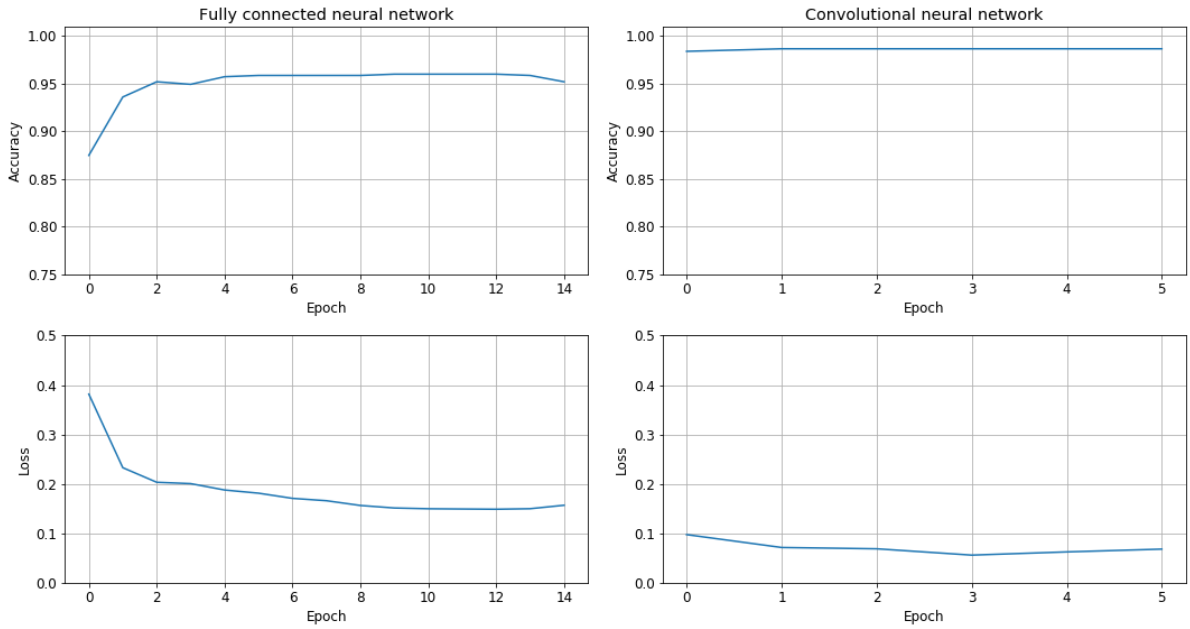
2 Results

2.1 Comparison of used models

2.1.1 Classification

In Fig 6, the evolution of the accuracy and loss function during training for each to the models is presented. It is clear to see the convolutional model achieves the best performance in both categories. However it is important to note that the training of the convolutional generally requires double the time per training iteration in comparison to the fully connected model. A possible explanation for this result is that the convolution kernels are able to extract features

from the input like slopes, peaks and valleys while a fully connected model is less likely to detect these local dependencies between values.



Source: Own

Fig 6: *Accuracy and loss metric for the two models*

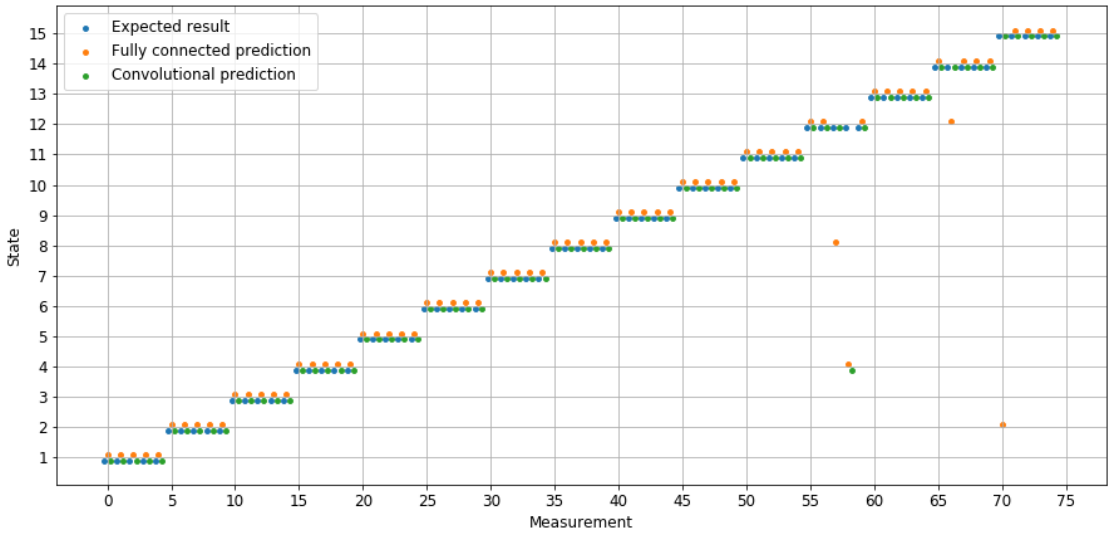


Fig 7: *Predictions of the different models for the samples in the test set*

In Fig 7 a visual representation of the accuracy of the predictions of both networks is given. As both networks have high accuracy most points match with the expected values. We can see that the convolutional network only wrongly classifies one sample of the testing set, while the fully connected network has worse performance.

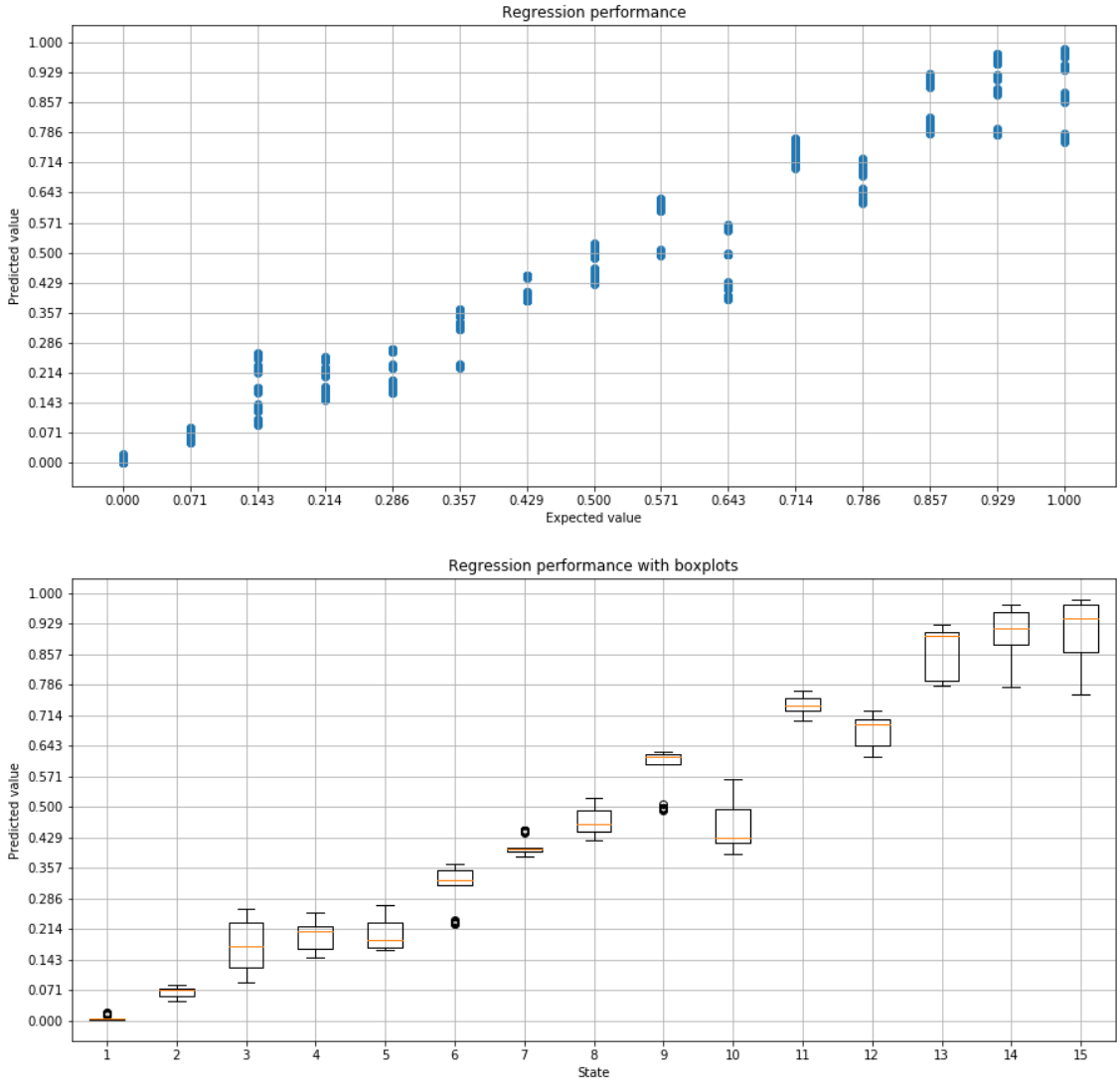
Another interesting observation is that the wrong predictions are not close to the actual result. This could indicate that subsequent states do not show a straightforward evolution, meaning that states are highly unique and not closely related. This could prove to be a problem when deploying these models in real world environments as the states in that case would lie in

between the learned states and the drastic changes in spectral data might provoke erroneous predictions.

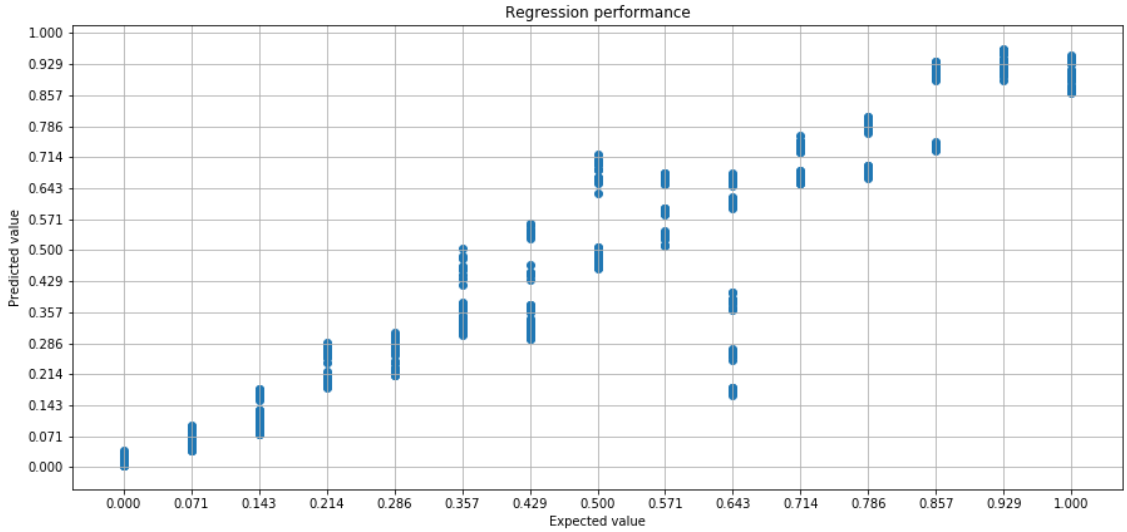
2.1.2 Regression

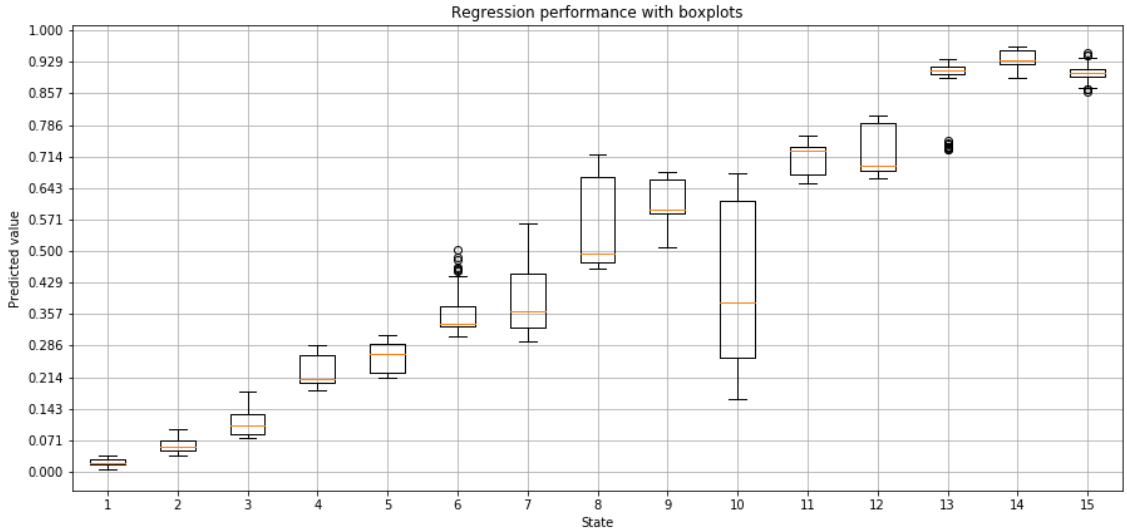
In Fig 8 and Fig 9 the performance of the test set for both models is presented. The first graph shows the scatter plot mapping the expected value to the predicted value, the second graph draws the boxplot of the predicted values for each one of the states.

In the case of regression, there is almost no performance difference between the fully connected and the convolutional neural network. Both networks achieve similar results in



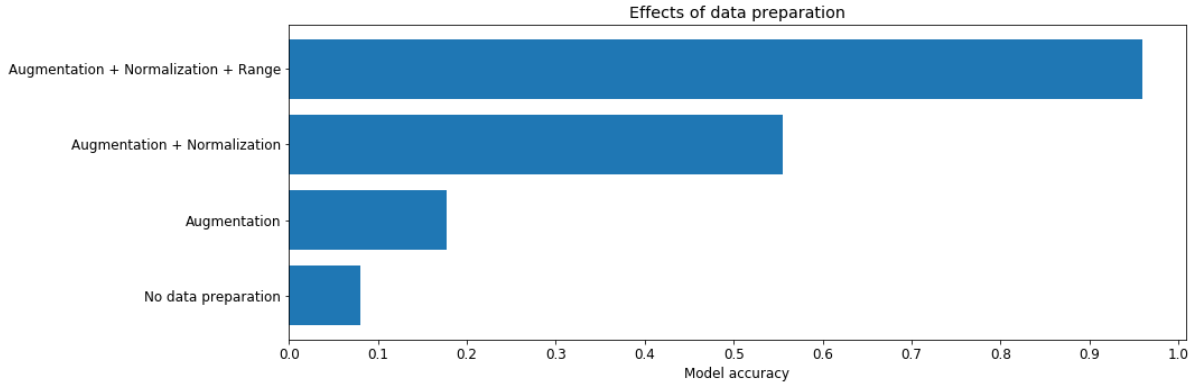
Regression performance of the test set for the fully connected neural network





2.2 Effects of data preparation

In Fig 10, the accuracy of the fully connected model is shown with different configurations of data preparation, the results for the convolutional model are similar. Four different configurations were tested, each one with the addition of one extra data preparation step.
Starting of without any data preparation, the accuracy of the network is equivalent as randomly guessing the state. With the addition of each of the data preparation measures the accuracy improves accordingly. It can be concluded that each of the data preparation measures explained in section 1.2 have a significant improvement on the performance of a particular model.



Source: Own

Fig 10: *Performance benefits gained by using data preparation*

3 Future work

As the classification spectral data is shown to have very high accuracy, there can be further experimentation with other fuel assembly states. For example the network can be adapted to classify the exact tube where there is an anomaly by changing the testing setup to have sensors on each axis and to prepare the necessary states.

There needs to be further research into the generalization ability of the obtained networks as they need to work in a real world environment where the container state will not exactly be one of the learned states.

Additional work is needed in the case of regression as it shows promise to be able to overcome the limitations with generalization. To accomplish this, the accuracy of the predicted values needs to be refined. Possibly by tuning the network parameters or by constructing hybrid models of different regression solutions.

Conclusion

The trained neural networks achieve very high accuracy when classifying spectral data and do not show signs of overfitting.

When using the models to perform regression, the results are less promising. While it is possible to predict a value that is rather close to the expected result, the presence of numerous outliers is worrying. It is possible when fine tuning some of the parameters and combining the network with other analysis methods, that an estimate with sufficient accuracy can be achieved.

Convolutional neural networks are best suited for the analysis of spectral data as they can be built to greater complexity and can recognize local patterns in the data, which certainly is an advantage when considering spectral data. Even though time complexity is greater for the convolutional architectures, the time needed for training is still within reasonable bounds.

When applying different data preparation procedures, the performance of all networks increase accordingly. With the addition of each of the procedures, the overall accuracy of the networks sees a significant improvement.

Acknowledgements

The investigations are funded by the German Federal Ministry for Economic Affairs and Energy on the basis of a decision by the German Bundestag. Grant identification number: 1501518A/B

Literature

[1] Schmidt S., Fiss D., Reinicke S., Wagner M., Rachamin R., Kratzsch A., Hampel U., *Development of a monitoring concept for Transport and Storage Containers for Spent Fuel and Heat-Generating High-Level Radioactive Waste on Prolonged Intermediate Storage.*

[2] Bjerrum E.J., Glahder M., Skov T., (2017, Oct 5) *Data Augmentation of Spectral Data for Convolutional Neural Network (CNN) Based Deep Chemometrics*

[3] Chollet F. and others, (2015) *Keras*. https://keras.io

[4] Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., … Zheng X., (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* https://tensorflow.org

---

Theo Dedeken

NADPIS ČLÁNKU V ČESKÉM NEBO SLOVENSKÉM JAZYCE

Abstrakt článku v českém nebo slovenském jazyce na 8 – 12 řádků.
Jestliže je článek v českém nebo slovenském jazyce, uveďte na toto místo nadpis článku a jeho abstrakt v anglickém jazyce.

TITEL DES ARTIKELS IN DEUTSCH

Abstrakt in der deutschen Sprache (8 – 12 Zeilen).
Ist der Artikel in deutscher Sprache verfasst, sind Titel und Abstrakt an dieser Stelle in englischer Sprache anzuführen.

NAGŁÓWEK ARTYKUŁU W JĘZYKU POLSKIM

Streszczenie artykułu w języku polskim na 8 – 12 linijek.
Jeżeli artykuł napisany jest w języku polskim, należy w tym miejscu umieścić tytuł artykułu i jego streszczenie w języku angielskim.