IMPACT SOURCE LOCALIZATION USING ARTIFICIAL NEURAL NETWORKS

Xiaoyang Liu[1] (410490), Shuteng Wang[1] (389883), Na Young Ahn[1] (392326), Jinyang Yu[1] (374390), Alejandro Delgadillo[1] (407829)

[1] Simulation Sciences M.Sc., RWTH Aachen University, Aachen, Germany

Accurate localization of impact is an important aspect of safe mechanical lightweight design. It is especially challenging in thin-walled structures due to dispersive wave modes and complexities that follow layered material decomposition given various environmental conditions. This study aims to accurately predict the impact location using a fully connected neural network (FCNN), convolutional neural network (CNN), recurrent neural network (RNN), and a hybrid neural network (CNN-GRU). Numerical and experimental data are preprocessed with a greedy approach for best alignment, while wavelet transform is additionally applied for inputs used for methods involving CNN. The study further identifies important features through heatmap and with reduced input, achieves a performance marginally better than the preconceived performance limit with all our methods combined.

KEYWORDS: source localization, artificial intelligence, wavelet transform, CNN-GRU

INTRODUCTION: Characterizing the health of mechanical components is crucial to ensure safety in modern lightweight designs. Many source localization estimation methods have been developed to locate active sources generated during the fracture process, e.g., crack initiation or internal friction, that propagates elastic waves. In thin-walled structures, however, the layered material decomposition in connection with environmental conditions and dispersive wave modes makes localization a challenging task. (Hesser et al., 2020) Such challenges can be overcome with the help of artificial intelligence, which excels at learning and recognizing unique and complex patterns. In this paper, we propose several deep learning methods to extract unique patterns, correlate them with the damage position, and ultimately determine coordinates of active sources.

For the initial data, we rely on 244 numerical and 24 experimental data points that lie on the first quadrant generated under the physical experiment and numerical simulation proposed by Hesser et al., 2020, where four piezoelectric transducers of dimension 10 x 10 x 0.5mm^3 are placed on a small aluminum plate of dimension 500 x 500 x 1mm^3, a size chosen to minimize any reflections off the edges. (Hesser et al., 2020)

In our previous work, we preprocessed the numerical and experimental data for best alignment and preliminarily optimized basis architecture and hyperparameters for basic neural network architectures i.e., Fully Connected Neural Network (FCNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), more specifically, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), with numerical data split to different parts used for training, validation, and test. Afterward, we trained on the augmented numerical data, validated with 24 labeled experiments, and tested with 18 experimental data provided specifically to the group to predict the final coordinates of the impact. Training on augmented data showed a mean absolute error (MAE) reduction of 59 percent for FCNN, 7 percent for CNN, 72 percent for LSTM, and 73 percent for GRU, with MAE of 4.48mm, as compared to having trained on non-augmented data. While such results aligned with our hypothesis of achieving good performance with a simple RNN, which generally is well suited for time-series data, we wanted to confirm how much improvement in prediction can be made by preprocessing the numerical data to be more suitable for CNN. One way to do this is through wavelet transform, which processes the numerical data on the frequency domain.

In this paper, we seek to better predict the impact location by additionally processing the data with Ricker wavelet transform and applying heatmaps to identify more meaningful parts of the data. A new design of FCNN, CNN, and CNN-GRU architecture is implemented to accommodate and tune input data with the additional dimensions resulting from the wavelet transform. The performances are then compared to see the impact of the wavelet transform and to confirm if a subsection of more relevant input data, identified via heatmaps, can yield a

better outcome. The method section contains the basic background theory, data preprocessing steps, neural network architecture as well as used hyperparameter values. The results section records the outcome of the tested hypotheses and parameter decisions while highlighting meaningful observations to be dealt with in-depth under the discussion section.

METHODS: The first part of this project is data preprocessing. Our initial dataset consists of numerical data generated from ABAQUS, a simulation program, for training and validation, and an experimental dataset for testing generated from real experiments. Each sample in the dataset has five channels: separate voltage signals from four Piezoelectric sensors allocated at different locations and one timeline. The trained model is used to predict the impact locations of an additional unlabeled experimental dataset.

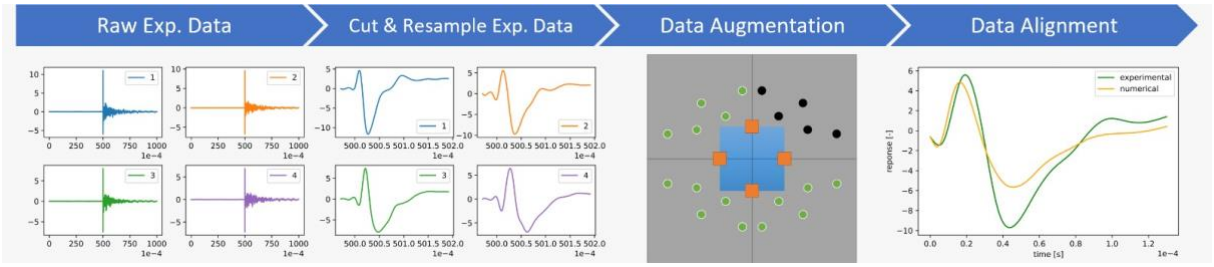

Figure 1 Data preprocessing steps

As seen in figure 1, data preprocessing is performed in four main steps. First, we cut and resample the experimental data, since the experimental data has different sampling lengths and sampling frequencies from the numerical data. Afterward, both datasets must be aligned. We develop a method to detect minor oscillations in the early stage, once the oscillation exceeds a certain threshold, a starting point will be determined. This operation also makes sure that the main shape of the data is cut out and saved for further processing. The third step is data augmentation. The initial numerical dataset is simulated and generated only from the first quadrant of the global coordinate system, which means more data from other quadrants are needed to predict impact location on the global plane. By flipping the initial data to the other three quadrants, a numerical dataset four times larger than the initial is obtained.

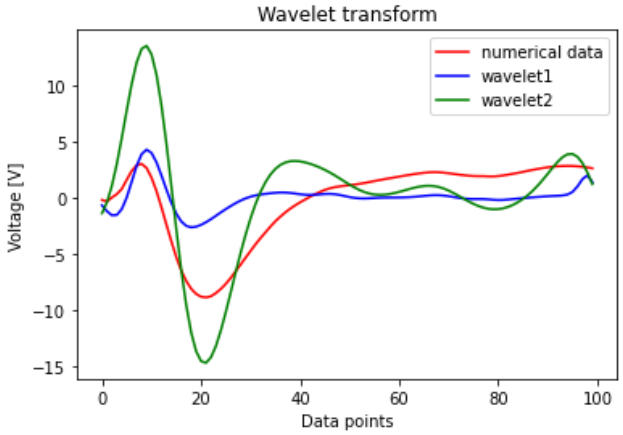

Figure 2 Transformed data points

The last method to be applied is the wavelet transform using the SciPy package. It can efficiently perform a continuous wavelet transform on data. Wavelet transform is a man-made feature extraction that is used to transform the input. It is an edge-detection technique that extracts high-frequency points that are thought to contain important information. We do not use the wavelet transformed input for the FCNN as wavelet transform yields too many parameters for FCNN to handle and the network automatically extracts useful features. Following these processes, we create 3 datasets. Dataset A includes data generated after the

first four preprocessing steps. Dataset B contains data resulting from the application of wavelet transform on dataset A. Finally, in dataset C we merge datasets A and B.
Neural networks are often used as universal function approximators, which can fit a mapping function between the sampled input and output data. Therefore, they are fit for regression tasks. FCNN has densely connected neurons and some non-linear activations units. The process of feed-forward computation of FCNN is doing matrix multiplication and non-linear activation, which for our network we use ReLU, layer by layer. Batch normalization (Ioffe et al., 2015) is added to center the activations. Mean square error (MSE) is used as the error function. The training process of the neural network uses the self-adapted back-propagation optimizer named Adam (Kingma et al., 2014). Dropout (Krizhevsky et al., 2012) and L2 regularizer (Cortes et al., 2012) are used to reduce overfitting. The parameters of FCNN we use in our project are shown in Table 1.

Table 1 Parameters for our FCNN model

Layers	Parameters
Input	Input shape = (100,) Batch Normalization (BN), Dropout = 0.1
Dense	128 units, ReLU activation, L2 regularizer, BN, Dropout = 0.1
Dense	64 units, ReLU activation, L2 regularizer, BN, Dropout = 0.1
Dense	16 units, ReLU activation, L2 regularizer, BN, Dropout = 0.1
Dense	8 units, ReLU activation, L2 regularizer, BN
Output	Output = (2,), dense output layer

When FCNN is used to address data like images, too many learnable parameters are sometimes needed. Hence, we need to use shared-parameter filters to detect the spatial correlations for the data with some inner structure. This is the idea of CNN. CNN architectures e.g., the classical LeNet (Lecun et al., 1995) consist of convolutional layers, pooling layers, and fully connected layers. There are also non-linear activations between layers. The parameters of our new CNN are shown in Table 2. We add more operations of batch normalization, L2 regularizer, and dropout. Moreover, in our new CNN model, we use the idea of skip connection with the outputs of the first convolutional layer and the third convolutional layer are added before going into the fully connected layer.

Table 2 Parameters for our CNN model

Layers	Parameters
Input	Input shape = (100, 4, 5), BN
Conv1	Kernel size = (2,4), depth = 16, BN, Dropout = 0.2, L2 regularizer
Conv2	Kernel size = (4,1), depth = 8, BN, Dropout = 0.2, L2 regularizer
Conv3	Kernel size = (8,1), depth = 16, BN, Dropout = 0.2, L2 regularizer
Add and Flatten	Add the output of Conv1 and Conv3 and do the flatten
Dense	32 units, ReLU activation, L2 regularizer, BN, Dropout = 0.2
Dense	16 units, ReLU activation, L2 regularizer, BN
Dense	8 units, L2 regularizer

Output	Output = (2,), L2 regularizer

The idea of shared-parameter filters is also well suited for the serial data, e.g., RNN. To learn the long-term dependency, we usually use some variations of RNN like LSTM (Hochreiter et al., 1997) or GRU (Cho et al., 2014). Because the data in our task is the information in every time step, we can use LSTM or GRU to keep the serial relation while extracting data information. LSTM and GRU have already achieved good results during our last presentation. This time, we modify it by combining CNN and GRU following the idea from image captioning. We first use CNN to extract the features. Then, we reshape the tensor and use the layer normalization to make it a tensor suitable for GRU. Afterward, GRU is used as what we did in our presentation. The parameters are shown in Table 3.

Table 3 Parameters for our CNN-GRU model

Layers	Parameters
Input	Input shape = (100, 4, 5), BN
Conv1	Kernel size = (2,4), depth = 16, BN, Dropout = 0.2, L2 regularizer
Conv2	Kernel size = (4,1), depth = 8, BN, Dropout = 0.2, L2 regularizer
Conv3	Kernel size = (8,1), depth = 16, L2 regularizer
Add	Add the output of Conv1 and Conv3
Conv4	Kernel size = (2,4), depth = 1, L2 regularizer
Reshape	Output shape = (100, 4), Layer Normalization
GRU	32 units, ReLU activation, L2 regularizer, Flatten and BN
Dense	16 units, L2 regularizer, BN
Dense	8 units, L2 regularizer
Output	Output = (2,), L2 regularizer

RESULTS: Enriching the input data set with wavelet transform reveals clear patterns in the heat maps. As seen in figure 3, the most important sections of the input data, which are shown in white, are found in the first 60 percent of the input vectors. These are the areas where useful features are located.

From the wavelet transform plot shown in figure 2, we also observe that the most meaningful features of the signal appear in the first half of the signal. This assumption will be verified later. Figure 3 shows heat maps from a convolution layer. The white spots represent the regions where the network identified useful features and the dark areas are those that are evaluated as non-useful features and zeroed in the training process.

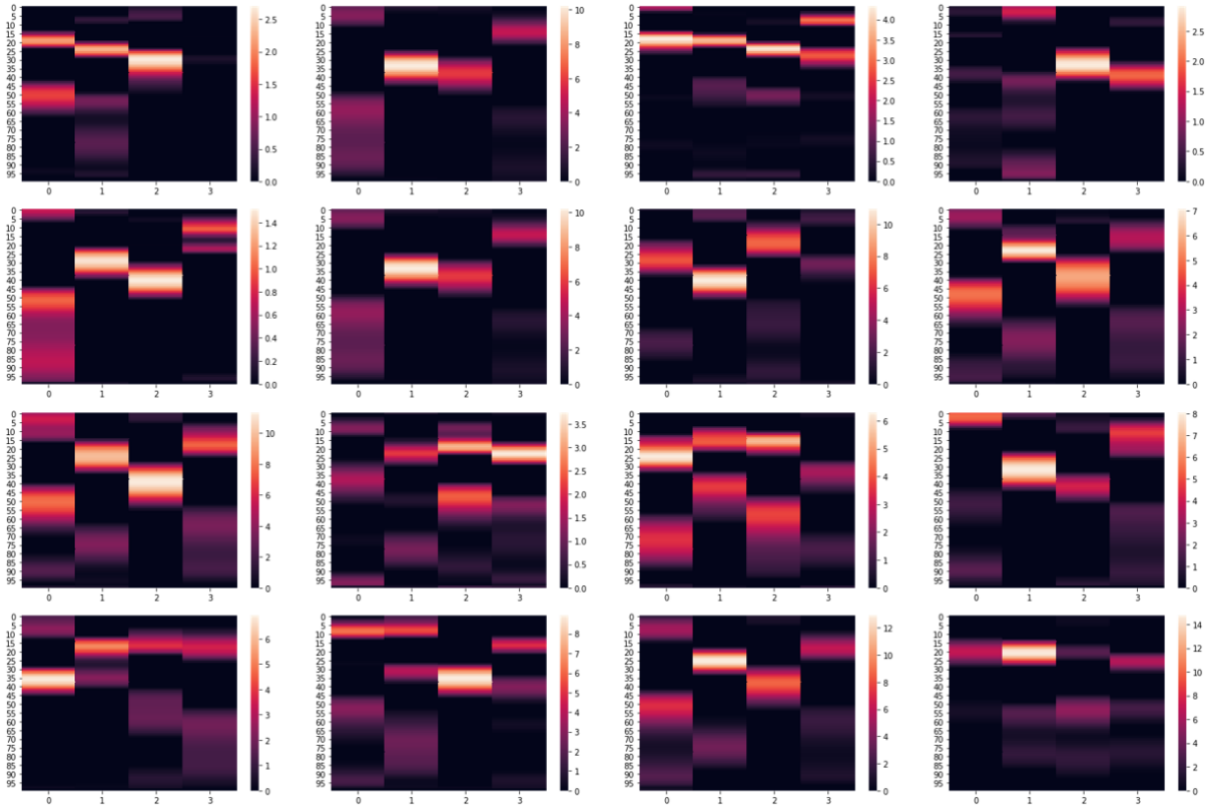

Figure 3 Heatmaps of 16 channels after the first convolutional layer. The horizontal axis marks the axes along which the wavelet transforms with different scales are applied. The vertical axes are those where the data points are shown.

Table 4 Comparison of MAE of 4 different NNs for 3 different datasets

MAE [mm]								
	GRU*		**FCNN**		**CNN**		**CNN-GRU**	
	Validation	Test	Validation	Test	Validation	Test	Validation	Test
Dataset A	4.48	6.25	2.79	6.00	6.54	8.51	5.76	9.15
Dataset B	N/A	N/A	N/A	N/A	5.27 (-19%)	7.11 (-16%)	6.40 (+11%)	7.75 (-15%)
Dataset C	N/A	N/A	N/A	N/A	5.76 (-12%)	6.58 (-23%)	3.86 (-33%)	5.96 (-35%)

* NN architecture implemented in our previous work

In our previous work, we achieved the best results with our GRU model. In our most recent attempt, we are able to improve our previously best-reported validation MAE by 37 percent from 4.48mm to 2.79mm achieved by our FCNN model. On the test data specifically given to our group, the best MAE is improved by 5 percent from 6.25mm (GRU, Dataset A) to 5.96mm (CNN-GRU, Dataset C).

To see the influence of wavelet transform on the performances for neural network architectures we measure the MAE in three folds by using each of the datasets as input, as seen in table 4. Overall, we observe the most prediction enhancement for CNN architectures when using dataset C. For validation, we observe an MAE improvement of 12 percent with CNN, and 33 percent with CNN-GRU whereas, for the test, we observe an MAE improvement of 23 percent with CNN and 35 percent with CNN-GRU. Interestingly, applying just the wavelet transform on the input also yields noteworthy improvements for most results. Moreover, we notice that the

error increases a little for the CNN-GRU model with dataset B. Possible reasons could be from the optimizers, weight initializations, etc. nonetheless, adding such feature extraction layers remains beneficial to the overall result.

DISCUSSION: Our initial pre-processed data set includes 100 data points from each sensor, corresponding to a measurement period of about 13ms. After analyzing the heatmaps and wavelet transform plots, we are able to reduce the measurement time to 7.5ms, which is equivalent to 60 data points per sensor, without increasing the MAE on validation and test. This significantly reduces the number of trainable parameters needed and improves training time.

As expected, the implementation of wavelet transform considerably increases the prediction accuracy of the CNN models. We attribute this increase to the scaling of meaningful features of the signal and the damping of noise and less relevant frequencies found in the input data. Although wavelet transformed data generate better predictions than pre-processed data alone, we find that the best results are achieved when using dataset C.

In general, we observe that the model has a greater MAE on the test dataset than on the training/validation dataset. This is also known as overfitting, which indicates that the model fits a particular pattern of the dataset too well. Since dropout and regularization are already applied in this project, further potential cure to limit overfitting could be the manipulation of the input data, e.g., k-fold cross-validation or reducing some of the input features. It is also important to consider the distance of impact location from the sensor array. In our previous study, we found that the accuracy of prediction is inversely proportional to the distance between the impact location and the sensor array. Since the test set includes several more impacts that are farther away from the sensor array than the validation dataset, it is expected to observe a slightly larger MAE for the test set.

Comparing our predicted coordinates to the actual coordinates, we are able to achieve the best MAE of 5.96mm with the hybrid CNN-GRU architecture. All our best predictions from all tuned architectures throughout the project converge to a value around 6mm, which could be the accuracy limit due to the uncertainty of the experimental data. Further analysis of the experimental data is needed to confirm this. Assuming 6mm is the limit of the predictive performance metric, we could extend our comparison of the networks by measuring performance over the number of parameters. (Canziani et al., 2017) This would allow us to measure the computational complexity of a net.

CONCLUSION: For this project, different network architectures with different hyperparameters are implemented for the prediction of the impact location of a small ball. FCNN, CNN, RNN, and their possible combinations are designed and tested on the provided data. Different data preprocessing options are implemented and compared. For example, some heuristic decisions to cut the data are based on a certain criterion or use the wavelet to transform the original data. Based on the results, for all the architectures, they show a substantial decrease ranging from 12 percent to 35 percent in the MAE for most cases. Among those, CNN-GRU gives the best results on the test set with an error of 5.96mm. Additionally, wavelet transform allows a better analysis of the data utilization inside the neural network by analyzing the heat maps generated, which leads to the use of a reduced data set with improved runtime and approximately equal accuracy. Limited experimental data could be one possible limitation preventing us from pushing the accuracy further. Suboptimal alignment of numerical data and experimental data could also limit our performance. The current method tries to filter out the small oscillation at the beginning and capture the main shape of the signals. However, the choice of hyperparameters in this part is intuitive and thus makes this approach limited for some anomalistic samples. In further investigations, a more robust alignment method could be developed and applied to improve feature engineering.

REFERENCES:

Azuara, G., Ruiz, M., & Barrera, E. (2021). Damage Localization in Composite Plates Using Wavelet Transform and 2-D Convolutional Neural Networks. Sensors, 21(17). doi:10.3390/s21175825

Canziani, A., Paszke, A., & Culurciello, E. (2017). An Analysis of Deep Neural Network Models for Practical Applications. arXiv [cs.CV]. Opgehaal van http://arxiv.org/abs/1605.07678

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Cortes, C., Mohri, M., & Rostamizadeh, A. (2012). L2 regularization for learning kernels. arXiv preprint arXiv:1205.2653.

Hesser, D. F., Kocur, G. K., & Markert, B. (2020). Active source localization in wave guides based on machine learning. Ultrasonics, 106, 106144. doi:10.1016/j.ultras.2020.106144

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., ... & Vapnik, V. (1995, November). Comparison of learning algorithms for handwritten digit recognition. In International conference on artificial neural networks (Vol. 60, pp. 53-60).

Moser, F., Jacobs, L. J., & Qu, J. (1999). Modeling elastic wave propagation in waveguides with the finite element method. NDT & E International, 32(4), 225–234. doi:10.1016/S0963-8695(98)00045-0

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).