Multivariate Time Series Regression on Seismic Data Using Adaptive Residual CNNs

Jurgen O.D. van den Hoogen*[1,2], Stefan D. Bloemheuvel*[1,2] and Martin Atzmueller[3,4]

[1] Tilburg University, TSHD: Cognitive Science and Artificial Intelligence, Tilburg, The Netherlands
[2] Jheronimus Academy of Data Science (JADS), 's-Hertogenbosch, The Netherlands
[3] Semantic Information Systems Group, Osnabrück University, Osnabrück, Germany
[4] German Research Center for Artificial Intelligence (DFKI), Osnabrück, Germany

Abstract

Developments in Machine Learning and more specifically Deep Learning have boosted the analysis of large-scale datasets. These methods for automated learning are particularly useful for complex data originating from sensors, i. e., multivariate time series data. In this work, we present an adaptive residual CNN (ARes-CNN) that is able to process such multivariate time series data. The model utilizes an adaptive input layer that separately processes every time series (e. g., channels or sensors), learning their key individual characteristics. Furthermore, the model applies stacked residual learning throughout each layer. We compare ARes-CNN with traditional Machine Learning applications, as well as a CNN and LSTM developed for the task at hand. The models' performance are compared on two datasets retrieved from sensors in a network of seismic stations located in Italy. Across all experiments, ARes-CNN reports a performance increase of 17% (MSE) on average compared to the best performing baseline. Therefore, processing the channels independently (by employing adaptive input layers), together with residual learning are fit for multivariate time series regression, e. g., for analysis of seismic sensor data.

Keywords

Time Series Regression, Deep Learning, Sensors, Convolutional Neural Networks, Residual Networks

1. Introduction

Advances in Deep Learning (DL) have revolutionized data processing due to their ability to deal with raw data, automatically learn its structure and recognize patterns [1, 2, 3], e. g., for multivariate time series analysis. Thus, many researchers used DL techniques to analyze time series data in a variety of tasks such as anomaly detection and forecasting. A less common approach in time series analysis is Time Series Extrinsic Regression (TSER) [4, 5, 6]. Here, the predicted or target value relies on the whole time series, instead of the last few observations.

In this paper, we focus on a TSER problem using DL methods, and more specifically Convolutional Neural Networks (CNN) to improve model performance. Our proposed model is tested on two seismic datasets, both recorded in regions of Italy [7, 8]. We attempt to predict

[*] Both authors contributed equally to this work.

LWDA'22: Lernen, Wissen, Daten, Analysen. October 05–07, 2022, Hildesheim, Germany

j.o.d.hoogen@jads.nl (J. O.D. v. d. Hoogen*); s.d.bloemheuvel@jads.nl (S. D. Bloemheuvel*); martin.atzmueller@uni-osnabrueck.de (M. Atzmueller)
0000-0001-5723-6629 (J. O.D. v. d. Hoogen*); 0000-0002-2480-6901 (M. Atzmueller)

maximum ground-shaking, expressed as intensity measurements (IMs), which can be described as a multivariate time series regression problem, similar to TSER-oriented approaches [5, 9]. The data is particularly interesting for applying neural networks due to its high-frequency sampling rate, high-dimensionality and the multiple regression outputs that need to be generated. We combine the strength of one-dimensional (1D) CNNs for learning feature representations of the raw time series, followed by residual stacks that are used to create a deeper neural network architecture that is able to learn more complex structures of the raw data. In addition, we utilize an adaptive input layer that separates the time series based on its dimensionality (e. g., channels or sensors). Thus, the model can be scaled depending on the number of time series variables available. Our contributions are summarized as follows:

1. We propose a method to perform multivariate regression on time series originating from seismic sensor data. For this, we present a new model called ARes-CNN that utilizes convolutional layers with residual stacks (both one-dimensional and two-dimensional).
2. We implement adaptive convolutional layers that are able to separate the multivariate time series data in distinct input layers – therefore learning unique characteristics for each individual time series (e. g., channels or sensors).
3. Finally, we evaluate our model thoroughly on two seismological datasets that differ significantly from one another, evidencing the generality and potential of the proposed model. We discuss our results in detail and perform a comparison against the baseline model proposed in [5], an LSTM model and traditional Machine Learning (ML) methods with rich feature engineering.

The rest of the paper is structured as follows: in Section 2, we discuss related work on DL for time series data and also summarize the necessary background on Convolutional Neural Networks and Residual Networks. Next, we introduce our proposed model in Section 3, where we also describe the baseline models, a description of the problem setting and the data. After that, Section 4 presents and discusses the results from our experiments. Finally, Section 5 concludes with a summary and outlines interesting directions for future research.

2. Related Work and Background

This section covers relevant work relating to Deep Learning in general and specifically for time series, focusing on the use of Convolutional Neural Networks and Residual Networks. In addition, this section also covers the implementation of Deep Learning for seismic analysis.

2.1. Deep Learning

With the advances in computational power and hardware, traditional Machine Learning (ML) approaches make way for more automated learning techniques, i. e., Deep Learning (DL). A reason for this transition is that ML requires considerable efforts to acquire representative features, which is typically time-intensive and can potentially be a rather error-prone task. For this, DL provides an alternative solution, relying on “built-in” feature construction: it is able to automatically extract features from raw data using multiple layers that can utilise nonlinear processing, which is a powerful approach on complex data such as multivariate time series.

At first, a multi-layer perceptron (MLP) was introduced by [1]. Here, layers are fully linked to one another, which is computationally expensive, in particular for large models with much training data. Therefore, in recent years more advanced methods have been developed such as the Convolutional Neural Network (CNN) [2] or Recurrent Neural Networks (RNN). Especially RNNs seemed to be well-suited for time series data since they memorize long-term dependencies. Nonetheless, storing time-related dependencies requires vast amounts of memory, resulting in long training times. Therefore, RNNs are less applicable on large-scale time series retrieved from sensors.

2.1.1. Convolutional Neural Networks

Generally speaking, CNNs are regularized MLPs that are designed to process two-dimensional (2D) data. They were first proposed by LeCun in 1989 [2] and are frequently used in Computer Vision and Image Recognition, processing images with multiple color channels [2, 10]. Compared to an MLP, the main advantages of a CNN are the use of weight-sharing, sub-sampling and local receptive fields. Sharing weights in particular lowers the memory requirements, boosting algorithmic efficiency [11]. A convolutional layer typically includes three stages. Convolutions are performed in the first layer, then an activation function is initialized in the second layer to account for nonlinear relationships. Following that, pooling is applied [11].

However, in the domain of time series, and more specifically signal data from sensors, CNNs were less often employed than classic ML techniques, due to their 2D nature. Initially, to use one-dimensional (1D) data for CNNs, one had to reshape the data into a matrix using signal processing techniques. These 2D CNNs combined with conversion of the 1D data increases computational complexity drastically, which requires dedicated hardware for training. Then, [12] developed a 1D CNN that is able to directly process raw signal data.

After that, the popularity of CNNs for time series significantly increased in various fields such as biomedical analysis [12] and fault diagnosis [3, 13, 14]. These 1D models are less computationally intensive than their 2D counterparts and are suitable for high-frequency and noisy time series data [13]. Hence, 1D CNNs are increasingly applied on time series data.

To create output features, a convolutional layer convolves the input using filter kernels followed by an activation function. Through weight-sharing, each of these filters extracts local attributes from the input local region. These outputs are then fed to the activation unit that produces the final output features. The convolution operation is defined as:

$$y_i^{l+1}(j) = k_i^l * M^l(j) + b_i^l. \tag{1}$$

Here, $M^l(j)$ defines the j-th local region in layer l. b_i^l denotes the bias and k_i^l denotes the weights of the i-th filter kernel in layer l. In the convolution operation, $*$ signifies the dot product of the kernel and the local regions and $y_i^{l+1}(j)$ represents the input of the j-th neuron for feature map i of layer l + 1.

2.1.2. Residual Networks

Generally, the more layers added to a neural network, the better its potential performance [10, 15]. However, models with a deeper structure take longer to train, meaning more computational resources are needed when increasing model size. Next to that, [16] showed that when adding more layers to the network, it becomes more difficult for the layers to propagate information from earlier layers, which is often called the degradation problem.

As a result, deeper models tend to have a saturation point for their performance, and afterwards rapidly decrease in performance when adding more layers. To tackle this problem, [16] developed a deep residual learning framework to ease the process of training for deeper networks. With this method, neural networks are able to scale up with hundreds of layers.

A residual network (ResNet), see Figure 1, incorporates a 'shortcut' or 'skip connection' between layers. In other words, the outcome of the previous neuron (identity) is directly added to the corresponding neuron of a follow-up layer. Between these layers, intermediate layers (residuals), which can be all kinds of layers (CNN, RNN, LSTM...), learn only changes from their input X. Afterwards, these residuals are added to the original identity and act as input for the follow-up layer. This provides an alternative shortcut for the gradients to pass through. As a result, one can make a building block for residual learning, visualized in Figure 1, which can be denoted as;

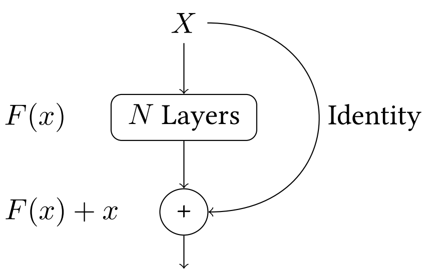

Figure 1: Example of a residual block.

$$Y(x) = R(x, W^i) + x \tag{2}$$

x represents the input from the original layer, that is the 'skip connection', and $R(x, W^i)$ represents the residual part to be learned. Combining both parts results in the output vector $Y(x)$ that is used for the follow-up layer. The benefit of including skip connections is that if any layer degrades the model's performance, it will be skipped.

2.2. Deep Learning for Seismic Analysis

With developments in data gathering and storage in the last decades, the seismic community collected enormous amounts of sensor data [17, 18]. The availability of these vast collections of time series data piqued the interest for seismologists towards ML and DL applications throughout the last years [19]. The data, representing waveform signals, are frequently used for earthquake detection [20], predicting maximum ground-shaking [5] or magnitude estimation [21].

In addition, [22] analyzed waveforms from multiple seismic stations, i. e., multivariate time series, for earthquake location estimation. In this paper, we propose a technique that applies 1D and 2D CNNs combined with residual stacks to perform multivariate time series regression. We show how to improve performance of an existing CNN model for seismic data proposed by [5], while significantly adapting their architecture.

3. Method

This section starts with a description of the problem setting and introduces the used datasets. Afterwards, we present our ARes-CNN model. There, we separate between the construction of the residual stack and the overall architecture of the model. At last, we describe model training, our baseline models for comparison, and describe the software and computational resources that we utilized. The code for the models can be accessed on Github: https://github.com/JvdHoogen/ARes-CNN.

3.1. Problem Setting

We perform regression on two seismic datasets recorded in Italy [7, 8]. The datasets consist of sensor readings retrieved from seismometers or accelerometers that are installed on seismic stations across Italy. During earthquake events, these sensors continually recorded the amplitude of seismic waves along three dimensions of ground motion, i. e., north-south, east-west and up-down. These recordings help seismologists to better understand the behavior of earthquakes.

Due to the fast transmission of information through telecommunication, seismologists were able to develop algorithms to predict the maximum intensity measurement (IM) of ground-shaking before the actual seismic wave arrives. This can be achieved by analyzing the data transmitted from stations that recorded the earthquake first, e. g., stations close to the epicenter/hypocenter. Within the seismological community this is called "Earthquake Early Warning" (EEW), which is important for the evacuation of residents affected by an earthquake. These IMs are divided into five separate metrics; *peak ground velocity* (PGV), *peak ground acceleration* (PGA) and *spectral acceleration* (SA) at 0.3, 1 and 3 second periods. The actual values of these metrics are used as target values for every earthquake.

Hence, every earthquake recording contains three waveforms from the seismic stations surrounding the epicenter of the geographical location. We used data with a length of 10 seconds sampled at 100 Hz for every recording related to the origin of the earthquake. Based on this information, we predict the five IMs for *all* stations in the geographical region represented in the dataset. Depending on the location of the earthquake recording, certain stations nearby will reveal more information related to the earthquake than stations far away. However, it is worth mentioning that for many earthquakes, most stations have not yet recorded the earthquake related ground-shaking within this 10-second period. In Figure 2, we visualize the task of our experiments based on an example.

3.2. The Data

Concerning the two datasets, the first dataset (CW) consists of three-component waveforms of 266 earthquakes recorded on 39 seismic stations in Central-Western Italy. The earthquake epicenters and station locations are within the coordinates [41.13°, 46.13°] (latitude) and [8.5°, 13.1°] (longitude), with recordings started from 1-1-2013 and 20-11-2017. The earthquakes' magnitudes vary between $2.9 < M \leq 5.1$, with crustal depth ranging between 3.3 km and 64.7 km. Therefore, the dataset characterizes itself as having the earthquake recordings scattered across a large geographical region.

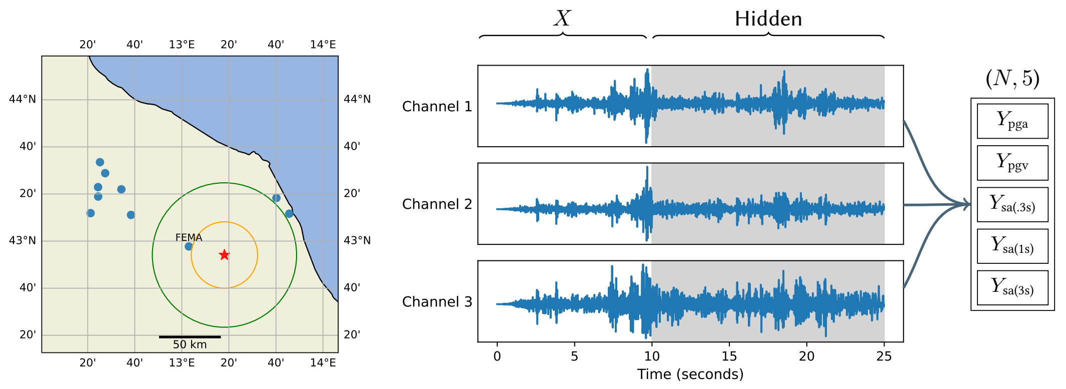

Figure 2: An example earthquake is shown on the left as a red star, with its P- and S-wave, green and orange respectively, cf. [4]. We predict the Y values that characterize the earthquake for each seismic station (with 3 channels of time series data) using the input X (10 seconds).

The second dataset (CI) has more recorded earthquakes (915) on a set of 39 stations containing three-component waveforms in Central Italy. The coordinates of the earthquake epicenters and station locations are within [42°, 42.75°] (latitude) and [12.3°, 14°] (longitude), with recordings starting from the 01-01-2016 until 29-11-2016. The crustal depth ranges between 1.6 km $\leq$ z $\leq$ 28.9 km with magnitudes varying between 2.9 $<$ M $\leq$ 6.5. The earthquakes are more concentrated than the earthquakes measured in the CW dataset, and are spread across a smaller geographical region. Thus, with the many more samples, we hypothesize that the performance on the CI dataset will be better than on the CW dataset since the task is expected to be less complex. Both datasets are normalized using the input maximum, i. e., the highest amplitude detected across multiple stations during the occurrence of an earthquake [5].

3.3. Residual Stack

Within the architecture of our proposed model, we implement residual stacks between the convolutional layers. For this, we developed two types of residual stacks, a 1D and 2D version. Generally, a residual unit consists of one or multiple layers with similar dimensionality as its input data, i. e., same amount of filters with kernel size and stride both set to 1. However, based on experimentation, we customized a residual stack (multiple units) to yield better performance.

Our residual stack includes two residual units. The first residual unit consists of two convolutional layers with ReLU and Tanh activation, processing the output from the past layer. Next, the residual output and the identity are combined by addition, as shown in Figure 3. Then, the new representation is used for the second residual unit, with the same settings as the first one. However, the second residual unit is learning the residuals of the identity plus residuals from the previous unit. Therefore, we call it a residual stack.

After processing, the residuals of the second unit are added to the next identity, which already contains a certain amount of residual learning from the previous unit. The final output is then fed to the follow-up layer in the model. Figure 3 visualizes the architecture of our residual stack, also showing the skip connections from its input to the output.

3.4. ARes-CNN Architecture

Our model is inspired by [5], which we also apply as a baseline for comparison. We exploit an adaptive input layer that processes multivariate time series in a univariate way, learning the characteristics of each time series individually (e. g., channels or sensors), which has proven to enhance model performance in [3, 14]. Hence, the amount of input layers needs to be adapted towards the amount of variables available which are processed in a univariate way. It can then be scaled towards use-cases containing more channels or sensors by adding the respective 1D adaptive layers (see Figure 4).

Thereby, the first part of the CNN architecture consists of three separate 1D convolutional layers with 10 filters each parallel processing the first 10 seconds of the three channels from the seismic waveform data, followed by a 1D residual stack and max pooling. After that, the outputs of the separate layers are concatenated. Next, a second 1D convolutional layer with 64 filters is utilized to further process the outputs of the separate layers, again followed by a 1D residual stack and max pooling. These convolutional layers specifically function as feature extractors to learn the underlying temporal patterns of every station. Both convolutional layers use wide kernels combined with small strides and increasing filters, which has proven to be effective for signal data [3, 5, 14]. After the temporal patterns are extracted, a 2D convolutional layer with 64 filters is initialized to capture the inter-station relationship [5]. Then, also a 2D residual stack is utilized, followed by a flatten layer and dropout (0.4) [11]. Empirical testing showed that the respective residual stacks between the convolutional layers improved model performance. However, adding more residual stacks here did not improve performance further. This provides for some indication of a potential saturation point for residual learning within the existing architecture.

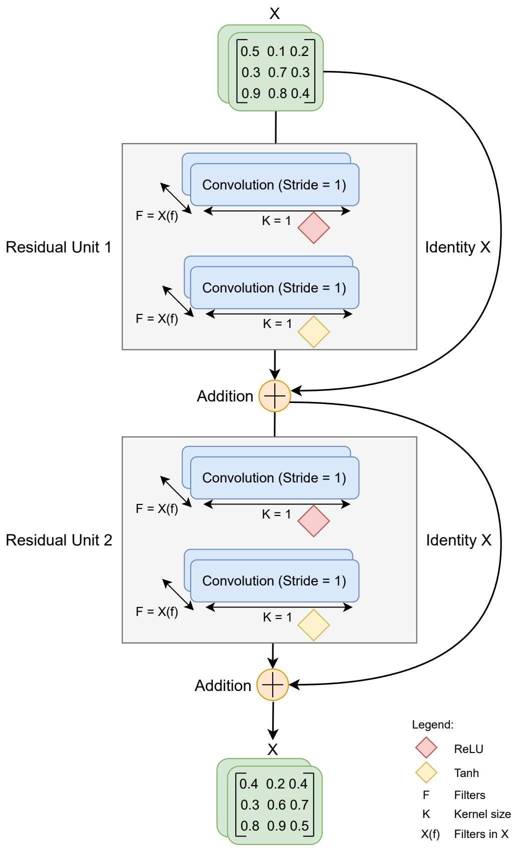

Figure 3: Residual stack architecture: Both residual units have a skip connection with identity X and ReLU/Tanh activation. The amount of filters for each residual layer is the same as for its input X.

In the last stage of the model, we concatenate the flattened output with a metadata vector consisting of latitude and longitude information of every station which has been proven to be valuable in [9]. Afterwards, a fully connected (FC) layer processes the previous outputs with 128 filters, followed by 5 fully connected regression layers representing the five distinct IMs. The output of the five regression layers represent the IMs for every of the 39 station in the sensor network for every earthquake. Figure 4 shows a full overview of our proposed model.

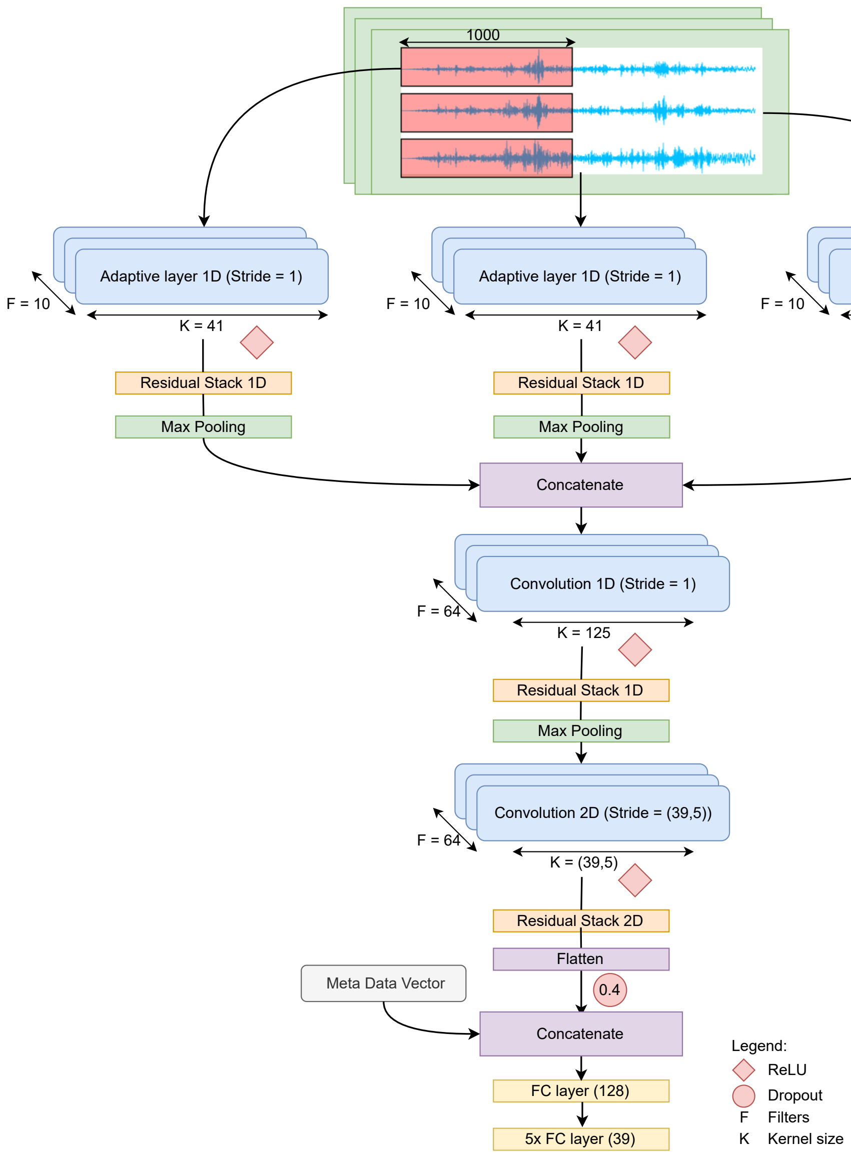

Figure 4: Architecture of the ARes-CNN model with adaptive input layers for each channel, residual stacks and pooling. The model utilizes 1D convolutional layers to learn time series features and a 2D convolutional layer to learn inter-station relationships.

3.5. Model Training

We follow a standard train, validation and test split of the dataset. First, we split the full dataset into 80% train and 20% test data. Next, the train set is divided into 80% train and 20% validation data. We apply five-fold cross-validation, where we predicted on the unseen test set for every fold/split. In total, we repeated the process five times with different random splits for the initial train/test set. We report the averaged value across all predictions on the test set.

To rule out early cutoff between training of the models, no early stopping was used in the experiments. We used MSE as loss function, RMSprop for optimization and a batch size of 20 with 100 epochs for training.

3.6. Baseline Models

Our ARes-CNN model is compared with traditional ML algorithms such as K-Nearest Neighbors (K-NN), Random Forest (RF), XGBoost and Support Vector Machine (SVM). These models do not lend themselves for processing raw multidimensional signals. Therefore, the data is preprocessed into a set of features in both the frequency and time domain, inspired by feature-sets described in [23, 24]. Features in the frequency-domain are calculated after transforming the time series to the frequency spectrum using the FFT algorithm that computes the one-dimensional discrete Fourier transform (DFT). In the frequency domain, Signal Energy; $\mathrm{E} = \sum(\mathrm{FFT}\,x_i)^2$ and Signal Power; $\mathrm{P} = \sum \frac{(\mathrm{FFT}\,x_i)^2}{\sum_i^t}$ were computed. In the time domain, mean, median, maximum, minimum, range (difference between maximum and minimum), variance and standard deviation were computed. The features were calculated for every station and earthquake. In total, the feature set consists of 9 features.

We used grid search optimization with five-fold cross-validation for all ML models to assess the best performing parameter setting. The parameter settings of the best performing model vary throughout the different experiments and between the two datasets. The properties of the grid search optimization are described in Table 1. Our model is also compared with the CNN developed by [5], which functions as a starting point for the development of our model. Furthermore, we compare our model with a shallow LSTM model, replacing the convolutional layers with two LSTM layers (containing 100 units), that directly process the time series data. Empirical testing showed no significant performance increase when adding more LSTM layers or increase the amount of units. For all DL models, we added the metadata vector containing latitude and longitude coordinates of every station.

3.7. Software and Resources

We used Python with Tensorflow and Keras to develop the models. All other algorithms, such as the ML models, are derived from the Sklearn package, also including Numpy/Pandas. The models were trained on a dedicated server with two Intel Xeon CPUs (3.2 GHz), 256 GB RAM and an Nvidia Quadro RTX 6000 (24 GB) GPU.

Table 1: Parameter settings used for grid search optimization per ML model.

Model	Parameter	Grid Search Range
K-NN	K	1-20
	Weight Options	Distance / Uniform
SVM	C	25, 30, 40
	Kernel	Linear / RBF
	Gamma	0.0001, 0.001
RF	Nr. of Estimators	800, 900, 1000
	Feature Estimation	Log2 / Square Root
XGBoost	Nr. of Estimators	800, 900, 1000
	Max Depth	5, 10, 15
	Gamma	0.0-0.4 (0.1 per step)

4. Results

Table 2 shows the averaged results across the five IM prediction values for every model. We observe a substantial gap between the algorithmic performance on the CI and the CW dataset. This behavior was expected since the CI dataset contains more earthquakes than the CW dataset, i. e., 915 against 266. Also, the earthquakes recorded in the CW dataset outline a larger geographical surface with greater distances between the seismological stations. Furthermore, the earthquakes in the CW dataset have a broader crustal depth range of the hypocenters.

In Table 2, we observe, that the CNNs clearly outperform the other models on the described metrics, displaying the value of CNNs in multivariate time series regression tasks. However, the LSTM model performs worse than any other model, indicating that RNNs do not lend themselves for directly processing high-frequency time series data as in our application context.

Table 2: Result metrics for IM prediction for every model on the CW and CI datasets.

Dataset	Model	MSE	RMSE	MAE
CW	LSTM	0.61	0.76	0.61
	K-NN	0.50	0.71	0.55
	SVM	0.49	0.70	0.54
	XGBoost	0.47	0.69	0.53
	RF	0.46	0.68	0.52
	CNN [5]	0.35	0.57	0.44
	ARes-CNN	**0.29**	**0.53**	**0.40**
CI	LSTM	0.41	0.63	0.46
	SVM	0.39	0.63	0.45
	K-NN	0.36	0.60	0.43
	XGBoost	0.31	0.56	0.40
	RF	0.31	0.56	0.40
	CNN [5]	0.22	0.46	0.33
	ARes-CNN	**0.18**	**0.41**	**0.30**

Overall, our proposed ARes-CNN improves performance by a large margin, suggesting that processing time series separately, using adaptive input layers, combined with residual layers beneficially influences model performance. We see that the average performance of the ARes-CNN compared with the second best performing model (CNN by [5]) is 17% on MSE, 9% on RMSE and 8% on MAE while only increasing the amount of trainable parameters with $\approx 4\%$, from 1.364.291 (CNN [5]) to 1.421.271 (ARes-CNN) parameters.

Another interesting observation from the results described in Table 2 is the high deviation between model performance within the Machine Learning algorithms. We see that the ensemble methods, e. g., RF and XGBoost perform significantly better than the SVM (which shows low performance on both datasets). Therefore, it seems that tree-based methods are quite capable in processing multivariate time series provided that the data is sufficiently preprocessed into a rich feature set.

5. Conclusion

This work proposes a Deep Learning (DL) model for multivariate time series regression in the context of seismic maximum ground-shaking prediction. Our model processes multivariate time series (e. g., channels or sensors) in a univariate way, using adaptive input layers, combined with residual learning. The experiments are conducted on two seismic datasets with varying characteristics. Here, our proposed model is compared to traditional Machine Learning (ML) applications, a shallow LSTM model and a Convolutional Neural Network (CNN) specifically designed for the task, developed by [5], which functions as a starting point for our model.

As the results in Table 2 indicate, the CNN applications outperform the other models, emphasizing the capabilities of CNNs for such complex and high-frequency time series data. In contrast, we also observe a relatively poor performance of the LSTM model, suggesting that a shallow LSTM model is not well suited for such contexts. Overall, our model improves performance with 17% (MSE) compared to the best performing baseline. This indicates that combining adaptive input layers with residual learning to the existing CNN architecture proposed by [5] is a valuable addition for multivariate time series regression tasks.

For future work, we plan to execute additional experimentation on large-scale multivariate time series datasets in various fields and deviating tasks, i. e., weather predictions and traffic forecasting. In addition, we will investigate the scalability of ARes-CNN when increasing the amount of time series channels, and thus more separate input layers. Furthermore, we aim to develop new methods of processing sensor data from multiple stations, e. g., using ideas from graph signal processing [25, 26] and by combining residual stacks with a Graph Neural Network structure proposed by [4], to improve model performance.

Acknowledgments

This work has been funded by the Interreg North-West Europe program (Interreg NWE), project Di-Plast - Digital Circular Economy for the Plastics Industry (NWE729).

We want to thank A. Michelini (Istituto Nazionale di Geofisica e Vulcanologia) and D. Jozinović (Swiss Seismological Service) for their domain knowledge considering the CW and CI datasets.

References

[1] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (1986) 533–536.

[2] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, The handbook of brain theory and neural networks 3361 (1995) 1995.

[3] J. van den Hoogen, S. Bloemheuvel, M. Atzmueller, Classifying multivariate signals in rolling bearing fault detection using adaptive wide-kernel cnns, Applied Sciences 11 (2021).

[4] S. Bloemheuvel, J. van den Hoogen, D. Jozinovic, A. Michelini, M. Atzmueller, Graph neural networks for multivariate time series regression with application to seismic data, International Journal of Data Science and Analytics (2022).

[5] D. Jozinović, A. Lomax, I. Štajduhar, A. Michelini, Rapid prediction of earthquake ground shaking intensity using raw waveform data and a convolutional neural network, Geophysical Journal International 222 (2020) 1379–1389.

[6] C. W. Tan, C. Bergmeir, F. Petitjean, G. I. Webb, Time series extrinsic regression, Data Mining and Knowledge Discovery 35 (2021) 1032–1060.

[7] A. Michelini, L. Margheriti, M. Cattaneo, G. Cecere, G. Dapos;Anna, A. Delladio, et al., The Italian National Seismic Network and the earthquake and tsunami monitoring and surveillance systems, Advances in Geosciences 43 (2016) 31–38.

[8] P. Danecek, S. Pintore, S. Mazza, A. Mandiello, M. Fares, I. Carluccio, E. Della Bina, D. Franceschi, M. Moretti, V. Lauciani, M. Quintiliani, A. Michelini, The Italian Node of the European Integrated Data Archive, Seismological Research Letters 92 (2021) 1726–1737.

[9] D. Jozinović, A. Lomax, I. Štajduhar, A. Michelini, Transfer learning: Improving neural network based prediction of earthquake ground shaking for an area with insufficient training data, Geophys J. Int. (2021).

[10] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: 3rd International Conference on Learning Representations, 2015.

[11] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, Cambridge, MA, USA, 2016.

[12] S. Kiranyaz, T. Ince, R. Hamila, M. Gabbouj, Convolutional neural networks for patient-specific ecg classification, in: Proc. IEEE EMBC), IEEE, 2015, pp. 2608–2611.

[13] A. Zhang, S. Li, Y. Cui, W. Yang, R. Dong, J. Hu, Limited data rolling bearing fault diagnosis with few-shot learning, IEEE Access 7 (2019) 110895–110904.

[14] J. van den Hoogen, S. Bloemheuvel, M. Atzmueller, An improved wide-kernel cnn for classifying multivariate signals in fault diagnosis, in: International Conference on Data Mining Workshops, 2020, pp. 275–283.

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

[16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[17] S. M. Mousavi, Y. Sheng, W. Zhu, G. C. Beroza, Stanford earthquake dataset (stead): A global data set of seismic signals for ai, IEEE Access 7 (2019) 179464–179476.

[18] A. Strollo, D. Cambaz, J. Clinton, P. Danecek, C. P. Evangelidis, A. Marmureanu, et al., EIDA: The European Integrated Data Archive and Service Infrastructure within ORFEUS, Seismological Research Letters 92 (2021) 1788–1795.

[19] P. Jiao, A. H. Alavi, Artificial intelligence in seismology: Advent, performance and future trends, Geoscience Frontiers 11 (2020) 739–744.

[20] S. M. Mousavi, W. L. Ellsworth, W. Zhu, L. Y. Chuang, G. C. Beroza, Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking, Nature communications 11 (2020) 1–12.

[21] A. Lomax, A. Michelini, D. Jozinović, An investigation of rapid earthquake characterization using single-station waveforms and a convolutional neural network, Seismological Research Letters 90 (2019) 517–529.

[22] M. P. van den Ende, J.-P. Ampuero, Automated seismic source characterization using deep graph neural networks, Geophysical Research Letters 47 (2020) e2020GL088690.

[23] S. Mazilu, A. Calatroni, E. Gazit, D. Roggen, J. M. Hausdorff, G. Tröster, Feature learning for detection and prediction of freezing of gait in parkinson's disease, in: Intl. workshop on machine learning and data mining in pattern Recognition, Springer, 2013, pp. 144–158.

[24] S. Masiala, W. Huijbers, M. Atzmueller, Feature-set-engineering for detecting freezing of gait in parkinson's disease using deep recurrent neural networks, arXiv preprint arXiv:1909.03428 (2019).

[25] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE signal processing magazine 30 (2013) 83–98.

[26] S. Bloemheuvel, J. van den Hoogen, M. Atzmueller, A computational framework for modeling complex sensor network data using graph signal processing and graph neural networks in structural health monitoring, Applied Network Science 6 (2021) 97.