Deep Neural Networks Applications in Structural Response Prediction

Fouad Amin, Cong Chen, Maharshi Joshi

Github Repository: https://github.com/MJ3128/Deep-Learning-Final-Project

Abstract

Conventional ways to obtain the time history response of structures under dynamic loads include physics-based computational models, installing sensors on real structures, or testing scaled models. These methods are typically very accurate and sufficient for reliable structural design. However, sometimes, these methods are computationally intensive, time-consuming or expensive. The search for optimizing and expediting the analysis process has never stopped. In this project, the Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Temporal Convolutional Network (TCN) were used to predict the responses of single degree of freedom structures with linear and nonlinear data.

Introduction

Being able to predict how a building will respond in the event of an earthquake is an important topic for many architects and engineers. Traditional methods involve attaching sensors to real structures to gather data in hopes that engineers can use the data to prevent future disasters. This method however does not predict how these structures will behave. This is where the recent boom in deep learning methods can help out. This paper will utilize various deep learning methods and neural network architectures to see if deep learning is a viable alternative to the current methods being used by scientists to accurately predict a building's response during a seismic event.

Literature Review

Not much research has been conducted to predict a full-time history of structural response using deep neural networks, however, there have been some successful-related studies in recent years. A trained data-driven Temporal Convolutional Network(TCN) model was successfully used to detect complex responses for linear and nonlinear systems (S.Günay 2023). Long short-term memory (LSTM) is another model that was found capable of figuring out long-range data dependencies and works well for nonlinear structural seismic response prediction(R. Zhang 2019). Another article(Y.Liao 2023) did similar research with Recurrent Neural Networks (RNNs) and LSTMs. Their findings indicated that both models could predict the time history response.

RNN Model Background

Recurrent Neural Networks (RNNs) are a type of neural network with short-term memory designed to recognize patterns in sequences of data. Unlike simple feed-forward neural networks, RNN can use its internal state to process sequences of inputs, which makes them ideal for time series prediction, such as full-time history of structural response prediction. The core idea behind RNN is to make use of sequential information by performing the same task for every element of a sequence, with the hidden state h_t depending on its previous hidden state neuron h_{t-1} and current input x_t. Outputs $\hat{y}_t$ can be easily achieved with h_t.

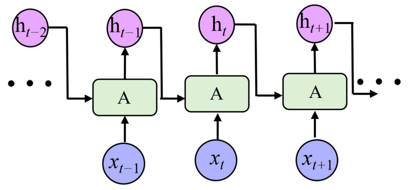

Figure 1: Schematic diagram of RNN units by Y.Liao(2023)

The forward propagation process to get h_t and $\hat{y}_t$:

$$h_t = \sigma(w_{xh}x_t + w_{hh}h_{t-1} + b_h)\hat{y}_t = \sigma(w_{hy}h_t + b_y) \quad (1)$$

The back propagation process to update parameters:

$$W_t = W_{t-1} - \eta * \nabla J(W_{t-1}) b_t = b_{t-1} - \eta * \nabla J(b_{t-1}) \quad (2)$$

LSTM Model Background

Long Short-Term Memory (LSTM) is a variation of RNN capable of learning long-term dependencies, which makes it more effective for dealing with sequential data such as time series date we're using in this project. LSTMs use gates to control the flow of information, mitigating the vanishing gradient problem of traditional RNNs. The forget gate is used to decide whether to keep or discard the unit state c_{t-1} of the $t-1$ time step. The output gate is used to calculate the hidden state h_t. Those gates allow LSTMs to maintain a longer-term memory compared to original RNN.

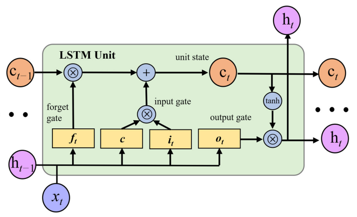

Figure 2: Schematic diagram of LSTM unit by Y.Liao(2023)

The forward propagation process:

$$f_t = \sigma(w_{fx}x_t + w_{fh}h_{t-1} + b_f)c \tag{3}$$

$$f_t = tanh(w_{cx}x_t + w_{fh}h_{t-1} + b_c)i_t \tag{4}$$

$$f_t = \sigma(w_{ix}x_t + w_{ih}h_{t-1} + b_i)o_t \tag{5}$$

$$f_t = \sigma(w_{ox}x_t + w_{oh}h_{t-1} + b_o)c_t \tag{6}$$

$$f_t = f_t \otimes c_{t-1} + i_t \otimes ch_t \tag{7}$$

$$f_t = o_t \otimes tanh(c_t\hat{y}_t) \tag{8}$$

$$f_t = \sigma(w_{hy}h_t + b_y) \tag{9}$$

RNN and LSTM Setup

Parameter	Linear RNN	Non-Linear RNN
Size of Hidden Layer	50	100
Number of FC Layers	1	1
Number of Epochs	200	200
Learning Rate	0.0002, 0.00005	0.0002, 0.00005
Optimizer	Adam	Adam

Parameter	Linear LSTM	Non-Linear LSTM
Size of Hidden Layer	128	128
Number of FC Layers	1	1
Number of Epochs	2000	2000
Learning Rate	0.00005	0.0001
Optimizer	Adam	Adam

RNN Results

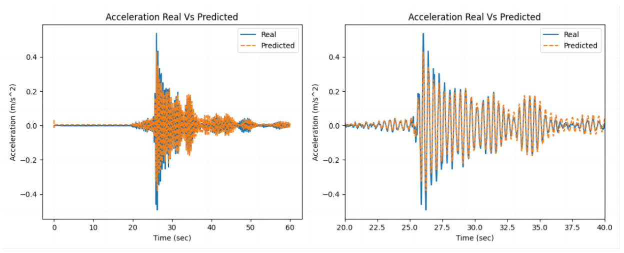

Figure 3: Acceleration output comparison; real vs predicted for the linear RNN model.

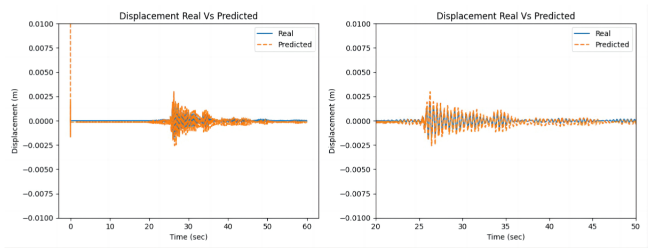

Figure 4: Displacement output comparison; real vs predicted for the linear RNN model.

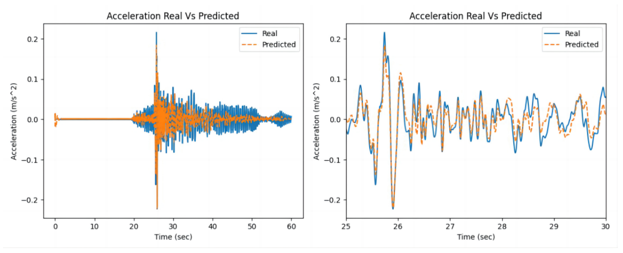

Figure 5: Acceleration output comparison; real vs predicted for the nonlinear RNN model.

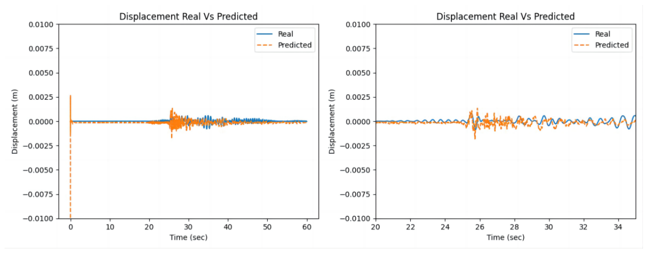

Figure 6: Displacement output comparison; real vs predicted for the nonlinear RNN model.

LSTM Results

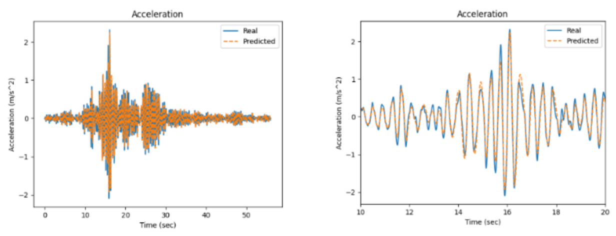

Figure 7: Acceleration output comparison; real vs predicted for the linear LSTM model.

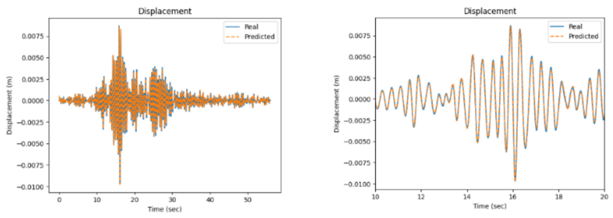

Figure 8: Displacement output comparison; real vs predicted for the linear LSTM model.

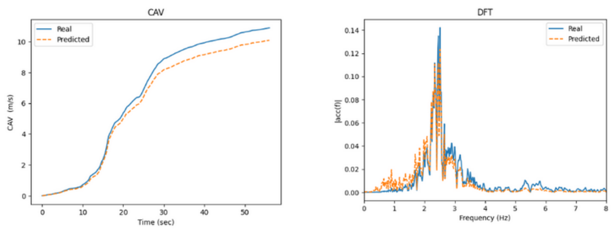

Figure 9: Acceleration metrics comparison; real vs. predicted for the linear LSTM model.

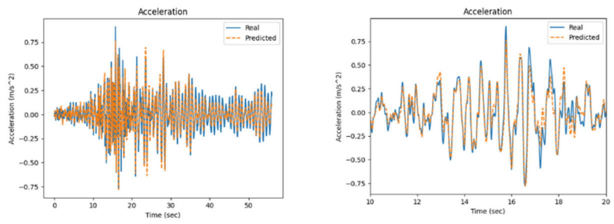

Figure 10: Acceleration output comparison; real vs predicted for the nonlinear LSTM model.

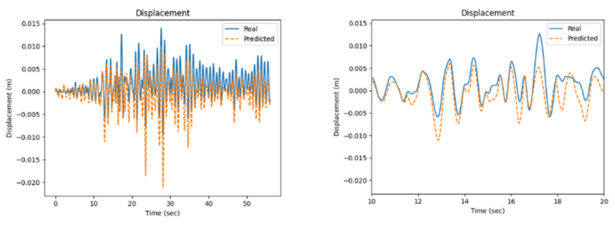

Figure 11: Displacement output comparison; real vs predicted for the nonlinear LSTM model.

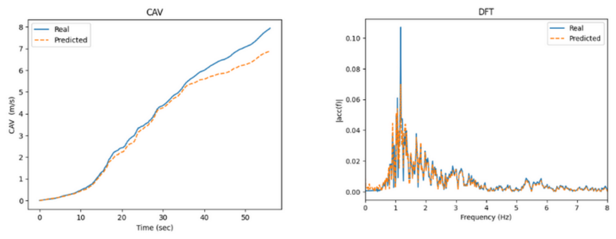

Figure 12: Acceleration metrics comparison; real vs predicted for the nonlinear LSTM model.

Computing the correlation coefficients:
For training data,

	Acceleration	Displacement
Linear	0.97	0.99
Non-Linear	0.63	0.44

For testing data,

	Acceleration	Displacement
Linear	0.97	0.98
Non-Linear	0.55	0.20

ConvLSTM Model Background

Convolutional Long-Term Short-Term (ConvLSTM) models provide extensions of the capabilities provided by LSTM models. The default input-to-state and state-to-state transitions in LSTM cannot maintain the spatial information of sequential data. The convolution enhances the LSTM detection of features of neighbors that are spatially close. The feedforward process contains a repetition of convolutional LSTM processes through which data is provided for data grids. The following equations provide the kernel gates for update, forget, and output (Torky and Ohno 2021).

$$\Gamma_u^t = \sigma(W_{xu} * X^t + W_{au} * A^{t-1} + W_{cu} \cdot C^{t-1} + B_u)\Gamma_f^t \quad (10)$$

$$\Gamma_u^t = \sigma(W_{xf} * X^t + W_{af} * A^{t-1} + W_{cf} \cdot C^{t-1} + B_f)C^t \quad (11)$$

$$\Gamma_u^t = \Gamma_f^t \cdot C^{t-1} + \Gamma_u^t \cdot tanhtanh(W_{xc} * X^t + W_{ac} * A^{t-1} + B_c)\Gamma_o^t \quad (12)$$

$$\Gamma_u^t = \sigma(W_{xo} * X^t + W_{ao} * A^{t-1} + W_{co} \cdot C^t + B_o)A^t \quad (13)$$

$$\Gamma_u^t = \Gamma_o^t \cdot tanhtanh(C^t) \quad (14)$$

Typically, data grids are three-dimensional, with two dimensions representing spatial location and one dimension for time. This architecture is simpler than using several fully connected layers for spatial feature detection. This kind of network can be very effective in spatiotemporal sequence forecasting.

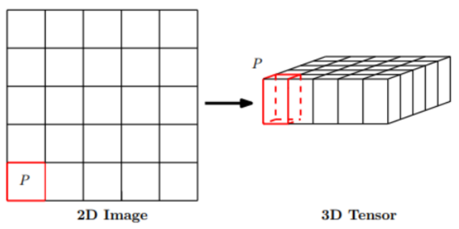

Figure 13: Transforming 2D image into 3D (Shi et al. 2015)

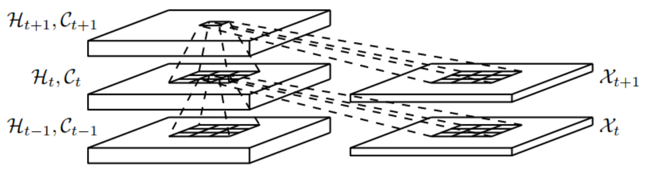
$\mathcal{H}_{t+1}, \mathcal{C}_{t+1}$
$\mathcal{H}_t, \mathcal{C}_t$
$\mathcal{H}_{t-1}, \mathcal{C}_{t-1}$
$\mathcal{X}_{t+1}$
$\mathcal{X}_t$

Figure 14: Inner structure of ConvLSTM (Shi et al. 2015)

Temporal Convolution Networks Background (Shaojie Bai, J. Zico Kolter, and Vladlen Koltun, 2018) and (Selim Günay, Issac Kwok-Tai Pang, and Khalid M. Mosalam, 2023)

Temporal Convolution Networks (TCNs) are the final type of model we will be using for this project. TCN's are a newer type of architecture introduced to help combine some of the benefits of both Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). TCNs are considered causal networks, meaning that the future does not influence the present or past. They also, like RNNs, are able to take input of varying time steps and produce an output with the same size as the input. To achieve, these properties, the TCN leverages a fully connected 1D convolution layer with zero padding. On top of this, These convolutions are causal, meaning that at each time step, the output is convolved with elements from the current time step and previous time steps. This type of architecture did however require a very deep network or large filters in order to have an effective history. This is where dilated convolutions can help.

Dilated convolution is defined as

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d\cdot i} \tag{15}$$

Where F is the dilated convolution operation, d is the dilation factor, k is the filter size, and $s - d \cdot i$ accounts for the direction of the past. Now, in order to get an effective history, filter sizes can increase, or the dilation factor can increase with the effective history of one layer being $(k-1)d$.

For the earthquake data, there are a few metrics that will be used in order to determine the efficacy of the model. Correlation coefficient, probability distribution of the errors, errors in peak response, and cumulative absolute velocity. These metrics are defined as,

$$r = \frac{\sum (y_i - \overline{y})(\hat{y}_i - \overline{\hat{y}})}{\sqrt{\sum (y_i - \overline{y})^2 (\hat{y}_i - \overline{\hat{y}})^2}} \tag{16}$$

Where r is the correlation coefficient, y is the actual-measured value and $\hat{y}$ is the predicted value by the network.

$$err_i = \frac{y_i - \hat{y}_i}{max(y_i)} \tag{17}$$

Where err is the probability distribution of errors.

$$err_{peak_i} = \frac{max(\hat{y}_i) - max(y_i))}{max(y_i)} \tag{18}$$

Where err_{peak_i} is the errors in the peak response.

$$CAV(T) = \int_{t=0}^{t=T} |\ddot{u}(t)| dt \tag{19}$$

Where CAV is the cumulative absolute velocity, T is the current time, and $\ddot{u}$ is the response acceleration.

TCN Setup

The TCN model was acquired using the keras-tcn package from GitHub (link in references). Since this model is able to take variable length inputs, a little bit of data preprocessing was done. No scaling was used, and instead each earthquake dataset was padded with zeros at the end. The model architecture is shown below,

Layer (type)	Output Shape	Num Parameters
Masking	(None, 26799, 1)	0
TCN	(None, 26799, 64)	267584
Dense	(None, 26799, 1)	65

In this, the 26799 represents the maximum amount of time steps across all earthquakes and the 64 in the TCN layer represents the number of filters to use. For additional parameters required by the TCN layer, the parameters are shown below,

Parameter	Linear TCN	Non-Linear TCN
Number of Filters	64	128
Kernel Size	5	9
Padding	Causal	Causal
Dilations	[1, 2, 4, 8, 16 32, 64]	[1, 2, 4, 8, 16, 32, 64, 128]

$$\begin{aligned} R_{field,linear} &= 1 + 2 \cdot (K_{size} - 1) \cdot N_{stack} \cdot \sum_i d_i \\ &= 1 + 2(64-1)(1)(127) \\ &= 16003 \end{aligned}$$

$$\begin{aligned} R_{field,non-linear} &= 1 + 2 \cdot (K_{size} - 1) \cdot N_{stack} \cdot \sum_i d_i \\ &= 1 + 2(128-1)(1)(255) \\ &= 64771 \end{aligned}$$

This represents the max number of steps back in time the filter can hit, also known as the effective history. For the rest of the model parameters, the values were as shown below,

Parameter	Linear TCN	Non-Linear TCN
Activation Function	tanh	tanh
Loss Function	MSE	MSE
Optimizer	Adam	Adam
Epochs	50	50
Learning Rate	0.001	0.001

TCN Results

The following results were obtained after training the model on 50 epochs and then using the validation dataset for predictions. First, with the linear dataset,

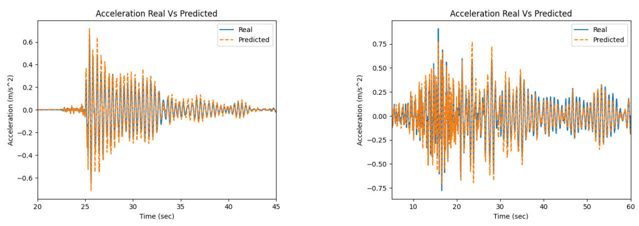

Figure 15: Acceleration Output Comparison; Real vs Predicted for the Linear and Non-Linear TCN models.

We see that the model was able to very accurately able to predict the response acceleration from this particular earthquake. With the non-linear dataset, as expected, the result is not as clean as the non-linear relationship is more difficult to capture. Both plots had the x-axis zoomed in order to show the most important portion of the signal. Now we can take a look at the various metrics discussed earlier. First up, the correlation coefficient,

Earthquake Number	Linear r	Non-linear r
1	0.95	0.35
2	0.85	0.48
3	0.97	0.71
4	0.94	0.96
5	0.93	0.38
6	0.96	0.60
7	0.96	0.41
8	0.95	0.62

In the probability distribution of errors,

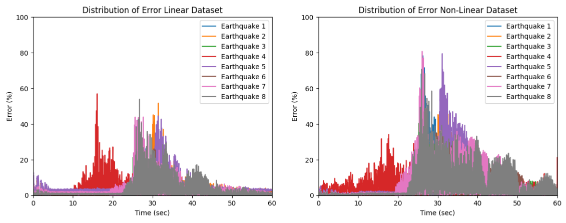

Figure 16: Probability Distribution of Errors for the Linear TCN Model.

We can see that the error was far worse for the non-linear dataset. The peak error,

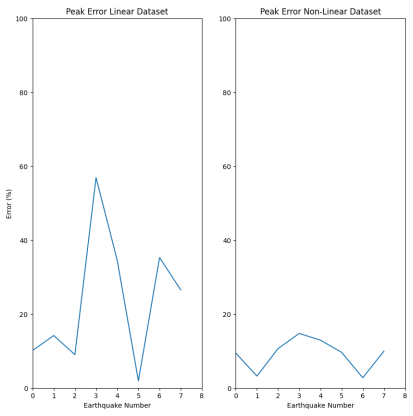

Figure 17: Peak Error for Both Linear and Non-Linear TCN Models

Here, surprisingly, the non-linear model outperformed the linear. The non-linear error was better able to calculate the peak value, despite not being as good at calculating the rest of the signal. Lastly, the CAV,

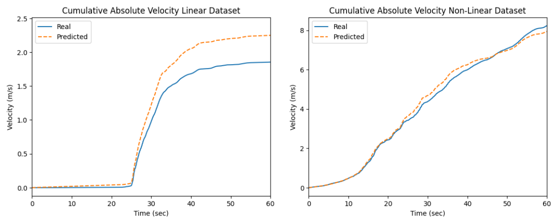

Figure 18: CAV for Linear TCN Model

Again, the non-linear actually outperformed the CAV of the linear model.

Conclusion

RNN

After seeing the results of RNN's, it was clear that RNN can roughly predict the structural response in both linear and nonlinear version. However, RNN model got into trouble with training taking too long, and obviously its performance in predicting nonlinear structural response is not as good as which in linear area. Meanwhile, the RNN model fits better in terms of relative acceleration prediction compared to relative displacement one.

LSTM

Several parameters were investigated, and their effects can be summarized in the following points:

- Batch size: Training using a batch size of 4 or 8 did not affect the accuracy too much. However, the training speed was faster with size 8.
- Learning rate: The learning rate had an enormous effect on the training quality. Large training rates resulted in non-consistent behavior, and loss values kept changing

abruptly. Low training rates of 0.00005 and 0.0001 were convenient for linear and nonlinear models, respectively.

- Hidden layer size: Three values were inspected (64,128,256). The middle option (128 layers) resulted in good accuracy with a reasonable training time.
- The number of LSTM layers: One and two layers were investigated. Using one layer increased training speed with no accuracy loss.
- Added zeros to sequence start: There was an overshooting phenomenon in the output, in which the start of each sequence was predicted to be very large compared to the real sequence. Adding two zeros to the start of each sequence helped mitigate and reduce this effect.
- Appending zeros to sequence end: Several approaches were available to deal with histories of different lengths. LSTM can deal with inputs of different lengths. However, in our study, different options were investigated. Appending zeros to the end of each sequence to have equal length for all histories was tested. In contrast to TCN, padding zeros for the LSTM model resulted in lower accuracy than using original trimmed signals without padding.

Overall, LSTM displayed much higher accuracy than basic RNN did.

TCN

TCN was only shown with response acceleration data and did not include response displacement. This was due to the fact that despite trying many different versions of the model with various parameters and different data manipulation, it was difficult to get the model to accurately predict both acceleration and displacement. Through the various tests, we found that one of the two was going to be less accurate, and we chose being able to predict response acceleration as the more important starting point for the model. Also, not shown for all 3 models is a multi degree of freedom structure, such as multiple floors on a building. The data needs to be properly prepared in order to maintain spatial and temporal coordinates in the data, and such data is not as simple to prepare or find, and so we decided to omit the structure from the report. In terms of performance, the non-linear data seemed to be best handled by the TCN since one set of earthquake data was actually predicted very accurately, and again, with better tuning, this would by far be the best choice especially since normal non-linear analysis is very computationally expensive and the TCN is relatively simple.

Future Work

Much of the future work involves adapting these models to more complex data and seeing if these models can hold up as complexity is added. For example, as discussed before, multi degree of freedom structures. For TCN's, attempting to tune the model correctly as well as correctly preprocessing the data can make being able to predict response displacement a trivial task. As shown in the paper "Structural Response Prediction Using Deep Neural Networks". With a correctly tuned model, this model proves to have the most potential in being able to adapt to multi degree of freedom structures due to its advantages over typical RNNs/LSTMs as well as typical CNNs. In addition, convLSTM's can be looked into more, as they may also be able to overcome some of the downsides of normal RNNs/LSTMs. In the paper "Temporal convolutional networks for transient simulation of high-speed channels", they found that TCN still outperformed the convLSTM when it came to convergence speed and required fewer data to train the model however it's entirely possible that given enough data, convLSTM network may be the best.

References

[1] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: *arXiv:1803.01271* (2018).

[2] Youjun Chen et al. "Attention mechanism based neural networks for structural post-earthquake damage state prediction and rapid fragility analysis". In: *Computers & Structures* 281 (2023), p. 107038. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2023.107038. URL: https://www.sciencedirect.com/science/article/pii/S0045794923000688.

[3] Selim Günay, Issac Kwok-Tai Pang, and Khalid M. Mosalam. "Structural Response Prediction Using Deep Neural Networks". In: *SMIP23 Seminar Proceedings* (2023).

[4] Colin Lea et al. *Temporal Convolutional Networks for Action Segmentation and Detection*. 2016. arXiv: 1611.05267 [cs.CV].

[5] Yangyang Liao et al. "Response Prediction for Linear and Nonlinear Structures Based on Data-Driven Deep Learning". In: *Applied Sciences* 13.10 (2023). ISSN: 2076-3417. DOI: 10.3390/app13105918. URL: https://www.mdpi.com/2076-3417/13/10/5918.

[6] Philippe Remy. *Temporal Convolutional Networks for Keras*. https://github.com/philipperemy/keras-tcn. 2020.

[7] Xingjian Shi et al. *Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting*. 2015. arXiv: 1506.04214 [cs.CV].

[8] Ahmed A. Torky and Susumu Ohno. "Deep Learning Techniques for Predicting Nonlinear Multi-Component Seismic Responses of Structural Buildings". In: *Computers & Structures 252 (August): 106570. https://doi.org/10.1016/j.compstruc.2021.106570.* (2021).

[9] Ruiyang Zhang et al. "Deep long short-term memory networks for nonlinear structural seismic response prediction". In: *Computers & Structures* 220 (2019), pp. 55–68. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2019.05.006. URL: https://www.sciencedirect.com/science/article/pii/S0045794919302263.