Online back-propagation of recurrent neural network for forecasting nonstationary structural responses

Zhymir Thompson[1,2], Alex Vereen[1], Austin Downey[1,3], Jason D. Bakos[2], Jacob Dodson[4], Adriane G. Moura[5]

[1] Department of Mechanical Engineering

[2] Department of Computer Science and Engineering

[3] Department of Civil & Environmental Engineering

University of South Carolina, Columbia, SC 29208

[4] Air Force Research Laboratory, Munitions Directorate

Eglin Air Force Base, FL 32542

[5] Applied Research Associates

Emerald Coast Division, Niceville, FL 32578

ABSTRACT

Online time series forecasting of dynamic structural responses subjected to nonstationary inputs enables the development of prognostic models used to compute the remaining useful life of a structure. Recurrent neural networks are deep learning models well suited for problems with sequential input data, such as time series. However, as the models are trained offline they cannot adapt to changes in the structures, due either to damage or inputs to the structure. Recurrent neural networks(RNNs) can have inference times at the millisecond timescale but suffer from slow training time on the order of hours if trained from an initial randomized state. This work presents a novel methodology to train an RNN online in real-time for forecasting nonstationary structural responses without the need for offline initialization. In this paper, a model capable of online learning without halting inferencing is constructed by utilizing two RNN models of identical architecture to make predictions and learn from past data concurrently. Typically, a model is unusable during training leading the user to choose between consistent model availability and using the most up-to-date parameters. Here a compromise is applied such that one can infer future structural responses and learn from the past for improved adaptation to changes in the structure. For the implementation, the predicting model responds to inputs from a queue. The learning model will look at past predictions to calculate loss. The loss will be used to calculate the gradients, and both models will be updated when a chosen number of passes have occurred. This paper provides an implementation of the proposed algorithm using acceleration data collected from a cantilever beam subjected to a contentious input with one discrete nonstationarity. The capability for the proposed system to learn over the nonstationarity and the time it takes to converge back to a steady-state are quantified. The effects of network size and artificial noise added to the signal are investigated and discussed in this preliminary work.

Keywords: nonstationarity, forecasting, RNN, pair architecture, online - Note:All manuscripts must include 5 keywords

INTRODUCTION

Anticipating and predicting the future dynamic states of a structure can enable structural control responses which have effectively zero latency between an incoming event and a control response [1]. In the structural domain, the recorded and predicted stat could be the acceleration the structure experiences or the strain at a particular point. In a broad theoretical sense, this challenge can be reduced to anticipating future points in an incoming stream of data. Assuming the data is periodic, one could construct an appropriate model to accomplish such a feat given sufficient time and resources. An unexpected disruption to the underlying structure providing the data would compromise the described model's ability to make future predictions. If the statistical properties of the signal change due to the unexpected disruption to the underlying structure, these events are called non-stationary events [2].

Models developed for temporal forecasting should be able to adjust for disruptions. However, there will be a delay for such models to collect new information post-non-stationary events and make new accurate predictions. This paper proposes the use of a machine learning architecture that uses a pair of recurrent neural networks (RNN) to learn a signal online, including over non-stationarities [3]. In theory, the pair of RNNs should be able to constantly make a prediction over data and dynamically adjust to changes in data caused by unexpected disruptions. The idea of using multiple models to infer and update simultaneously was proposed by Panahi et al. [3] and used a pair of multilayer perceptrons (MLP) of identical architecture to learn and infer. MLPs work very well for updating in a stochastic environment, but occasionally struggle to utilize the information learned in earlier training.

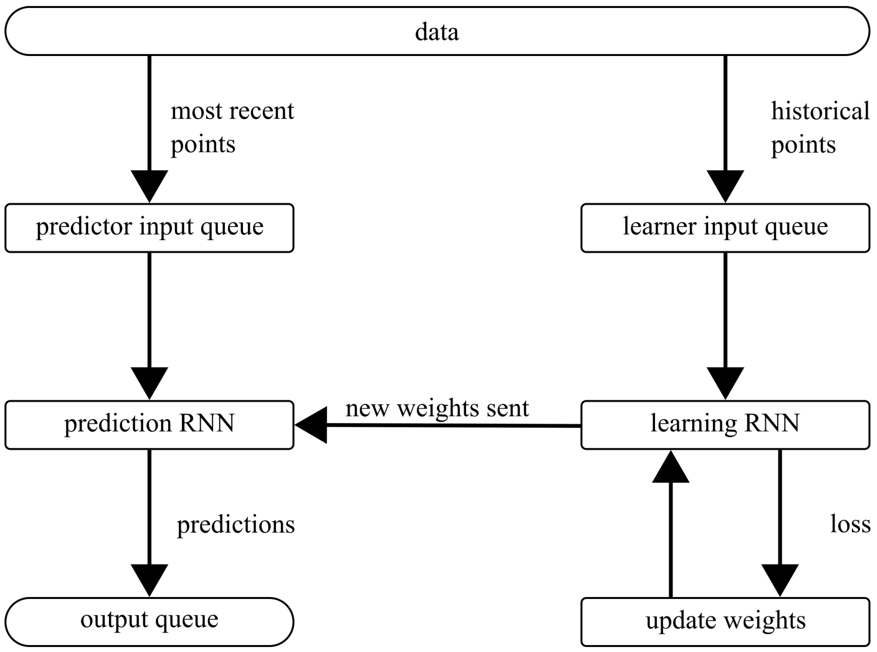

Figure 1: The general structure for the proposed pair of models.

BACKGROUND

RNNs are a useful model for making predictions on time-series data since they use the previous state to make future inferences. For this reason, it was believed that replacing MLPs with RNNs would improve prediction capability. Despite this, vanilla RNNs have known disadvantages that could negate the expected improvements. Variations of RNNs also exist that attempt to compensate for the drawbacks of vanilla RNNs. However, the discussion and evaluation of RNN variants go beyond the scope of this paper.

Figure 1 shows the general structure for the proposed model. Data is collected from the data stream and fed to queues. The inferring model receives the latest set of data, and the learning model receives a pair composed of the past data necessary to predict the current data point along with the current data point. The models concurrently pull data from the respective queues and perform their roles independently. The inferring model makes and sends predictions based on the data given in the input queue, and the learning model makes predictions, calculates loss between the predictions and recorded values, backpropagates and updates model parameters based on calculated loss, and sends those updated weights to the learning model. When the learning model has a pause between inferences, it updates its weights to the new weights sent by the learning model. This way, the inferring model can always be available to infer on new data and still be capable of adjusting to new information.

The hyperparameters were chosen by tuning using optuna [4]. The hyperparameters investigated were optimizer type (choices were SGD or Adam), learning rate, number of layers, and how many points to infer before backpropagation(sequence length). The best performing values for these hyperparameters were SGD, 0.07, 1 layer, and 2 respectively. Before online training, the model was tested in an offline setting to determine its baseline performance. The model was trained on 100,000 points of contiguous data [5]. The data used in the experiment can be found in the open source data set developed by Chowdhury et al. [5]. Figure 2 shows the data used for the experiment [5]. Important to the experiment is the change from 4.5s to 5.5s. That region shows the response of the system to the nonstationarity event.

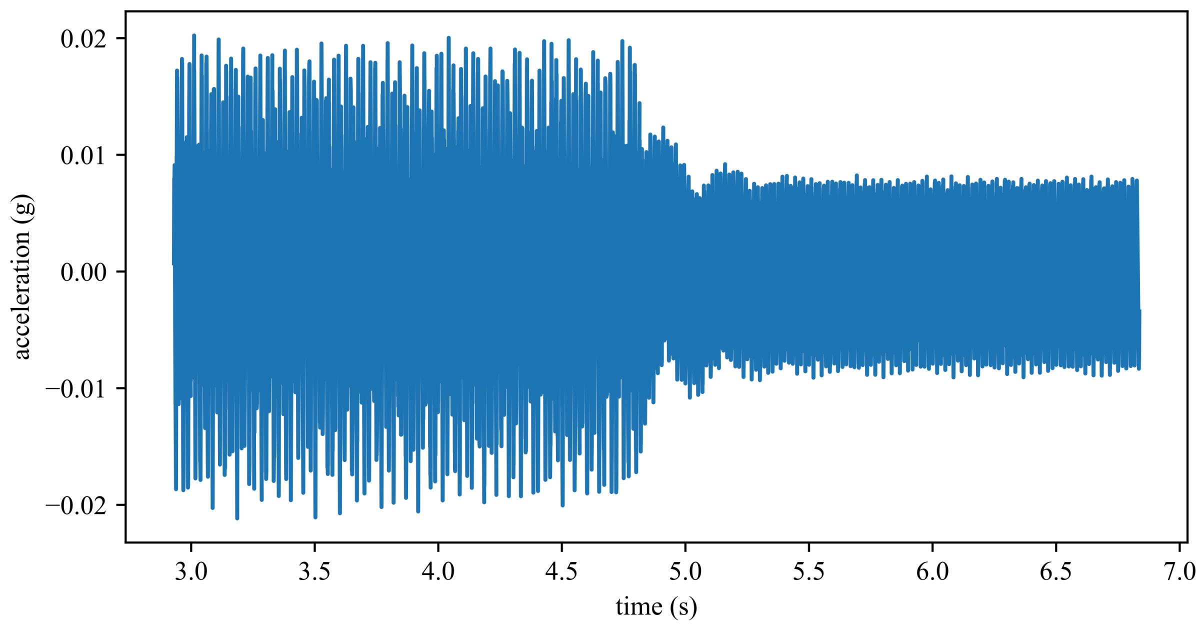

Figure 2: Data to be forecasted with a non-stationary event at approximately 4.75 s, from an open-source dataset [5].

For online training, each model of the pair took seven input values and produced a single output. The models looked at the last 7 time-steps of acceleration data to make a prediction of the acceleration 10 time-steps into the future. Every time the window slid ten time-steps, the learner would backpropagate, update weights according to error, send the updated weight data to the predictor, and reset the hidden state. The predicting model would simultaneously infer on given data and its weights would update whenever new weight parameters were available. 200,014 points of acceleration data were fed to the model, and the predicting model made 200,000 predictions over the data [5]. The set chosen included the introduction of nonstationarity to observe model performance before and after the event. Mean squared error (MSE) was the chosen function chosen for training, and that metric was converted to RMSE in the loss curve to provide a more tangible loss value. Time Response Assurance Criterium (TRAC) [6] and Signal-to-noise ratio (SNR) were used for evaluation. The TRAC value quantifies the similarity between time traces on a scale from 0 to 1 through comparing the numerical error and time delay of each estimation.

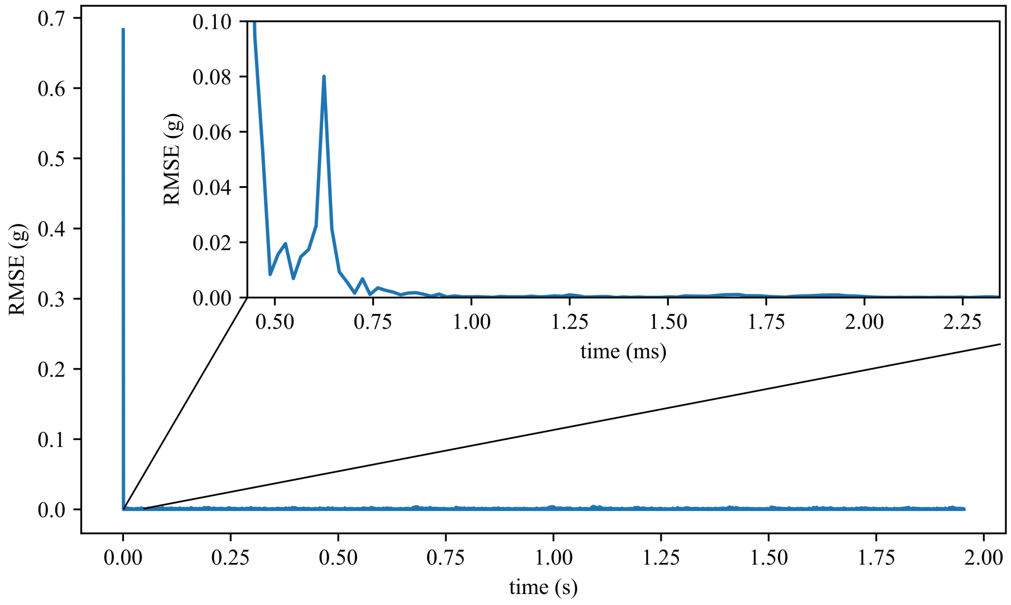

Figure 3: The loss curve over the course of training.

ANALYSIS

Figure 3 shows the loss curve over the course of training. The model quickly converged to a low loss value and maintained low loss throughout the duration of training. Despite an initial error of 0.7 g_n, the model quickly adapted and began producing accurate predictions with less than 0.01 g error before 0.75 ms had passed.

Figure 4 shows slices of the predictions and observed values recorded during online training. Part (a) shows the predictions and observed acceleration over the entire slice of data used. Part (b) of figure 4 suggests a large initial convergence time. The model has a comparatively large error that slowly converges to fit the observed signal. The most probable causes for this kind of initial error are improper model architecture or a consequence of the pair architecture. The increase in error initially suggests a mixture of the two, but more than likely the model architecture is the biggest contributor. However, when the model eventually converges it fits to the signal well. This behavior is seen in part (c). There exists small errors near peaks and troughs, but the difference is negligible. The model shows difficulty fully fitting to the model after the nonstationarity as shown by part (d). This behavior is expected, but modifying model architecture would likely improve model performance.

Table 1 shows the scores for TRAC, SNR, and RMSE for the slices in time shown in parts (c) and (d) of Figure 4. Row one shows the metrics for the signal output predicted before the non-stationarity event. Row two shows the predicted signal at a later point after the nonstationarity. The given metrics suggest a strong fit to the reference signal. The SNR shows the predictions match the anticipated signal with minor interference. The high TRAC score shows that the signals closely match. These metrics are further supported by RMSE which shows that the predicted model has an average error of 0.007 g_n. After the nonstationarity, there is a clear drop in performance. The SNR decreases by approximately 3.4 dB, and the TRAC decreases

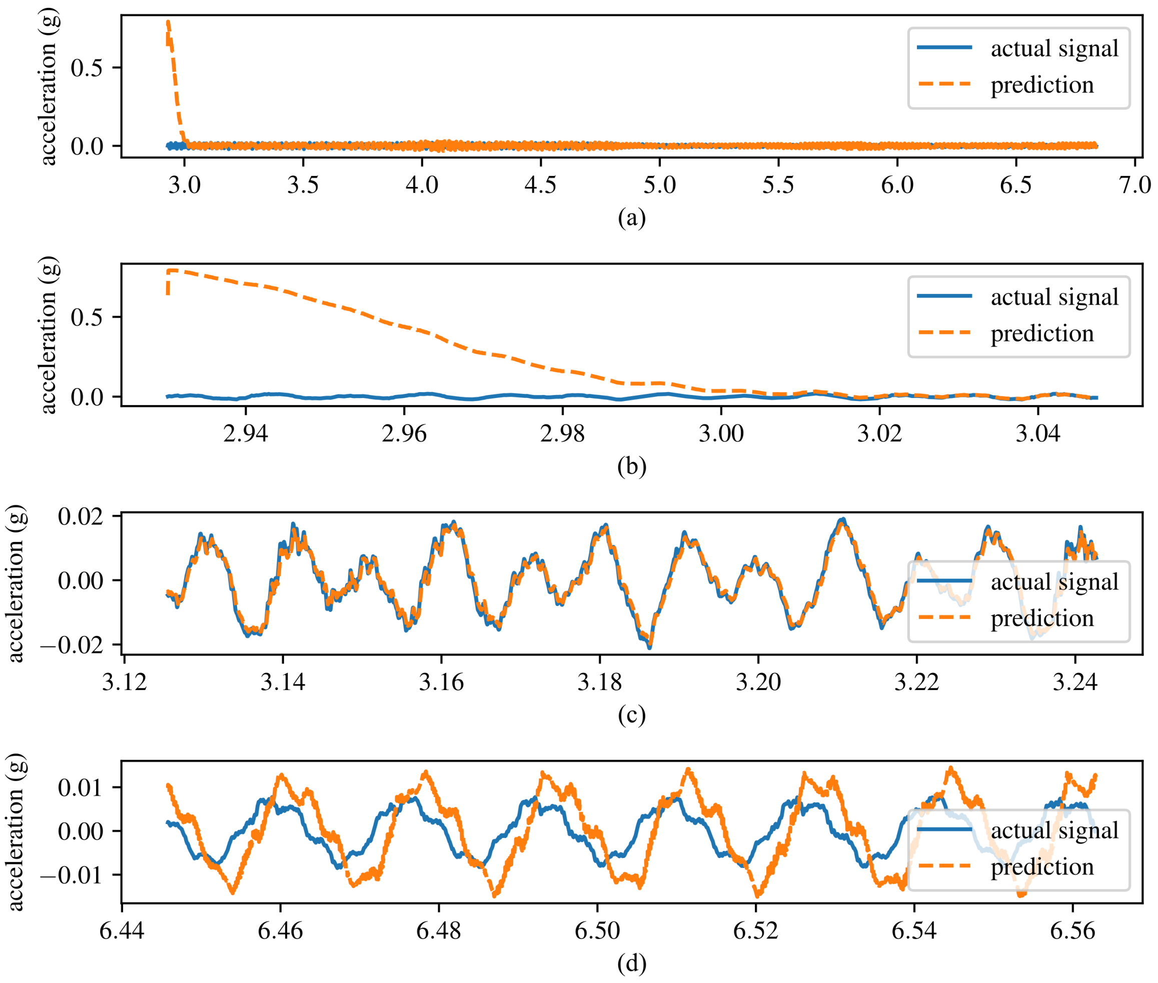

Figure 4: Comparison of predictions and actual acceleration values at different time segments for online training.(a) The entire slice of data observed, (b) beginning of observed data, (c) slice of time before nonstationarity, (d) slice of time after nonstationarity.

by about 0.11. In this case, these metrics paint a better picture of the ability for the model to adapt than RMSE which suggests minor error. While the error is small, there is also a noticeable drop in amplitude for the observed signal, as shown in figure 4(d).

Table 1: Metrics for comparison of model prediction to observed values.

time frame	SNR_{dB}	TRAC	RMSE
pre non-stationarity	54.120	0.997	0.001
post non-stationarity	39.457	0.840	0.007

CONCLUSION

This paper demonstrates the ability of a pair of RNNs to simultaneously predict and learn in an online environment. There still exists an initial delay due to necessary convergence time, but the model adjusts well to changes to the environment. Future work includes improving model response time and utilizing different architecture types to find ideal performance (LSTM, GRU, etc.). Future work could include the combination of this pair architecture with a physics-based model to improve overall prediction power. The accuracy of a physics-based model in conjunction with the responsiveness of a data-driven model could develop into a better predictor overall than either alone.

ACKNOWLEDGEMENTS

This material is based upon work supported by the United States Air Force through the Air Force Research Lab Summer Faculty Fellowship Program and by the Air Force Office of Scientific Research (AFOSR) through award no. FA9550- 21-1-0083. This work is also partly supported by the National Science Foundation Grant numbers 1850012, 1956071, and 1937535. The support of these agencies is gratefully acknowledged. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the United States Air Force. (Distribution A. Approved for public release; distribution unlimited (XXX)).

REFERENCES

[1] Matthew Nelson, Vahid Barzegar, Simon Laflamme, Chao Hu, Austin R.J. Downey, Jason D. Bakos, Adam Thelen, and Jacob Dodson. "Multi-step ahead state estimation with hybrid algorithm for high-rate dynamic systems". *Mechanical Systems and Signal Processing*, 182:109536, jan 2023.

[2] J.K. Hammond and P.R. White. "The analysis of non-stationary signals using time-frequency methods". *Journal of Sound and Vibration*, 190(3):419–447, feb 1996.

[3] Atiyehsadat Panahi, Ehsan Kabir, Austin Downey, David Andrews, Miaoqing Huang, and Jason D. Bakos. "High-rate machine learning for forecasting time-series signals". In *2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 1–9, 2022.

[4] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. "Optuna: A next-generation hyper-parameter optimization framework". In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[5] Puja Chowdhury, Austin Downey, Jason D. Bakos, and Philip Conrad. "Dataset-4-univariate-signal-with-non-stationarity". https://github.com/High-Rate-SHM-Working-Group/Dataset-4-Univariate-signal-with-non-stationarity, April 2021.

[6] Peter Avitabile and Pawan Pingle. “Prediction of full field dynamic strain from limited sets of measured data”. *Shock and Vibration*, 19(5):765–785, 2012.