Real-time Forecasting of Vibrations with Nonstationarities

Ishrat Singh[1], Philip Conrad[1], Puja Chowdhury[2], Jason D. Bakos[1], and Austin Downey[2,3]

[1]Department of Computer Science and Engineering
[2] Department of Mechanical Engineering
[3] Department of Civil and Environmental Engineering
University of South Carolina, Columbia, SC 29208

ABSTRACT

With recent advances in structural and materials engineering, a new class of structures has emerged consisting of systems subject to highly dynamic environments. Such systems include civil structures exposed to blast, spacecraft prone to debris strikes, and aerial vehicles experiencing in-flight changes. Real-time modeling and prediction of structural state will introduce new real-time health monitoring capability for such systems. To be practical, the models must be deployable on lightweight integrated platforms such as field-programmable gate arrays (FPGAs) and microcontrollers. This paper evaluates two algorithms that employ multilayer perceptron neural networks (MLPs) that perform predictions and update the model in real-time on non-linear, non-stationary time series data. We show that these models perform well on real-world experimental datasets generated with a replica of the Air Force Research Lab's DROPBEAR apparatus, exhibiting accuracy on par with MLPs trained offline. Results show that these two algorithms serve as viable candidates for real-time structural health monitoring applications.

Keywords: Real-time, machine learning, online learning, non-stationarities, microcontrollers

INTRODUCTION

With recent advances in structural and materials engineering, a new class of structures has emerged consisting of systems subjected to high-rate dynamic environments. Such systems include civil structures exposed to blast (e.g. inner-city blast mitigation barriers [1]), spacecrafts prone to debris strikes, and aerial vehicles experiencing in-flight changes [2] [3]. Potential real-time monitoring systems present on such a structure would greatly benefit from technology capable of learning the condition of the structure and making accurate and efficient predictions about the state of the structure at future points in time.

A structure's vibration model can predict its current and future state. Models based on first principles are generally heavy weight and incur a long update latency, making them slow to adapt to changing operating conditions. Supervised data-driven models, such as those based on deep learning models, offer the promise of low-latency adaptability.

In general, a supervised machine learning model functions by distilling a large ground truth dataset into a relatively simple mathematical formulation. Such models are normally trained offline because ground truth is often only available in pre-curated datasets. However, forecasting applications allow the potential for online learning, since currently-observed data can refine the model for the purpose of forecasting future values.

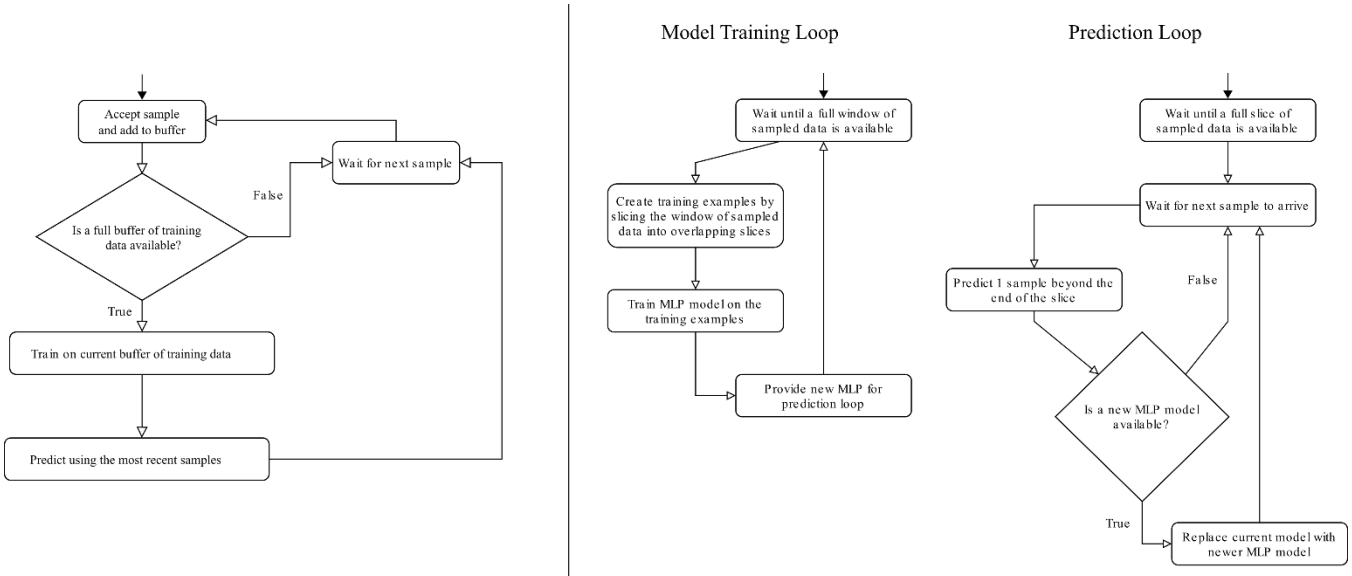

Figure 1: Algorithmic flowchart comparison between the I-MLP (left), and W-MLP (right) algorithms.

METHODOLOGY

Figure 1 presents the two proposed real-time machine learning algorithms described in this paper. The windowed-MLP (W-MLP) trains an MLP on a fixed-size window of input data, and then uses the MLP to predict the next window of data. The Iterative-MLP (I-MLP) algorithm uses a single MLP over the course of the whole dataset and is updated with each new sample observed. Figure 2 presents both the W-MLP and the I-MLP as a graphical representation in the time-domain.

Each algorithm utilizes an MLP neural network consisting of one hidden layer. The configuration of the I-MLP was selected by carrying out a grid search over its four parameters: (1) history length, the number of consecutive data points that comprise the input layer, (2) forecast length, the number of timesteps into the future for which the model makes a prediction, (3) hidden layer size, and (4) training epochs per sample.

The I-MLP maintains a variable-length buffer consisting of up to the most recent (history length + forecast length) number of samples observed, and its neural network is re-trained at each timestep with the first (history length) number of elements in this buffer as input values and the last element of this buffer as the target value. History length was varied from 50 to 500 in steps of 50, forecast length from 10 to 100 in steps of 10, hidden layer size from 20 to 200 in steps of 20, and training epochs from 1 to 10 in steps of 1. This grid search yielded an optimal history length of 40, forecast length of 10, hidden layer size of 100, and epochs per sample of 10.

A similar grid search was conducted for W-MLP over its parameters, (1) sample window, (2) training window size, (3) hidden layer size, and (4) epochs per window. The W-MLP trains a fresh neural network for each window. The W-MLP is trained across a fixed-length buffer of training window samples, sliced iteratively into sample windows with non-overlapping slices. As with the I-MLP, the last sample in each slice is the target value for each training example. The training window was varied from 20 to 100 in steps of 10, sample window from 10 to 100 in steps of 10, hidden layer size from 5 to 20 in steps of 5, and training epochs from 25 to 400 per training window in steps of 25. This grid search yielded an optimal training window of 100, sample window of 50, hidden layer size of 10, and epochs per window of 200.

In both cases, the neural network in each algorithm utilized the rectified linear unit (ReLU) activations between its input layer and hidden layer and a linear activation for its output unit. Each MLP is trained using the Adam optimizer [8] at a learning rate bounded by 0.001.

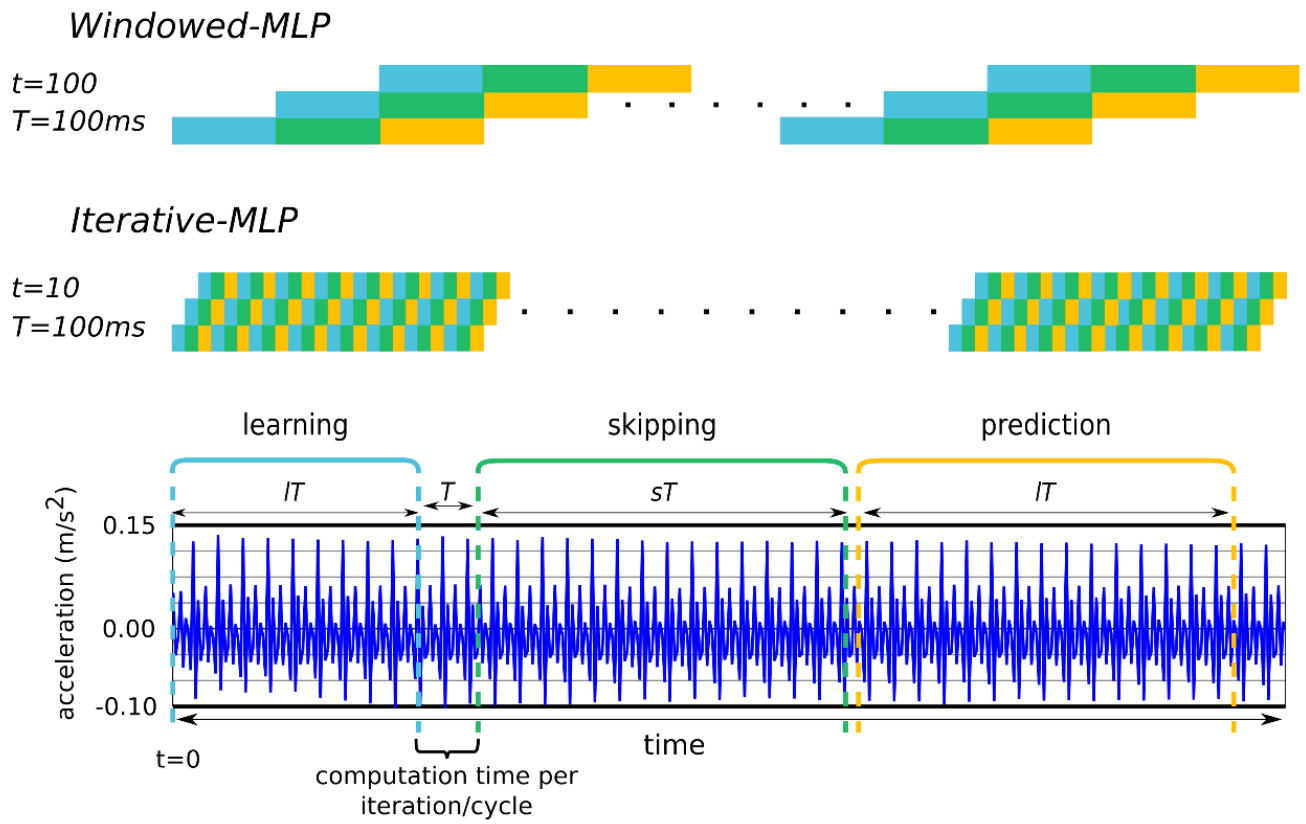

Figure 2: A visual schematic of how each algorithm (W-MLP, top; and I-MLP, bottom) operates with respect to an incoming time series data stream.

TEST BENCH & TRAINING DATA

Figure 3 shows the experimental setup used in this work. A steel cantilever beam structure of 759 x 50.66 x 4.19 mm is used for the experiments and a single Integral Electronics Piezoelectric (IEPE) accelerometer (PCB Piezotronics model J352C33) is placed near the edge of the beam structure. This accelerometer has a sensitivity of 100 mV/g with a frequency measurement range from 0.5 to 9k Hz. Using a 24-bit NI-9234 IEPE signal conditioner developed by National Instruments, the sensor data is digitized. The beam is excited by an electromagnetic shaker (LDS model V203R) that has a useful frequency range of 5-13000 Hz and a peak sinusoidal force of 17.8 N, itself powered by a power amplifier (LDS model PA25E-CE). A load cell with a range of 0-45 N (Transducer Techniques model MLP-10) is installed between the structure of the shaker and the beam. To acquire the load-cell data, a 24-bit bridge input signal conditioner (National Instruments NI-9237) is used. The experiment is controlled by custom code written in the LabVIEW environment.

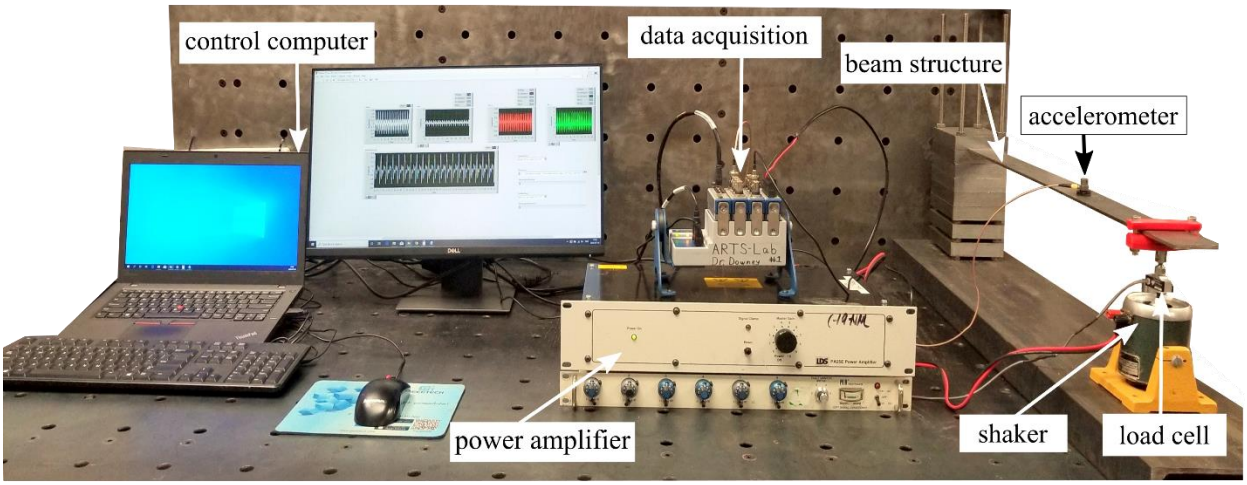

Figure 3: A cantilever beam experimental setup with the main components and set up for data acquisition.

For a composite sinusoidal input from the shaker, Figure 4 reports the measured acceleration response of the structure. The composite signal in this work consists of frequencies of 100, 120, and 150 Hz. At the t=0 s point, a 50% nonstationary is present as two sine wave signals are added together. The first half of the signal is paired with frequencies of 100 and 120 Hz and the other half with frequencies of 100, 120, and 150 Hz. As determined by a 50% rise in the standard deviation of the signal, a 50% nonstationary event is added at 0 s. For achieving an input signal of 0.25 V is used before t=0 while a signal of 0.375 V is used after t=0.

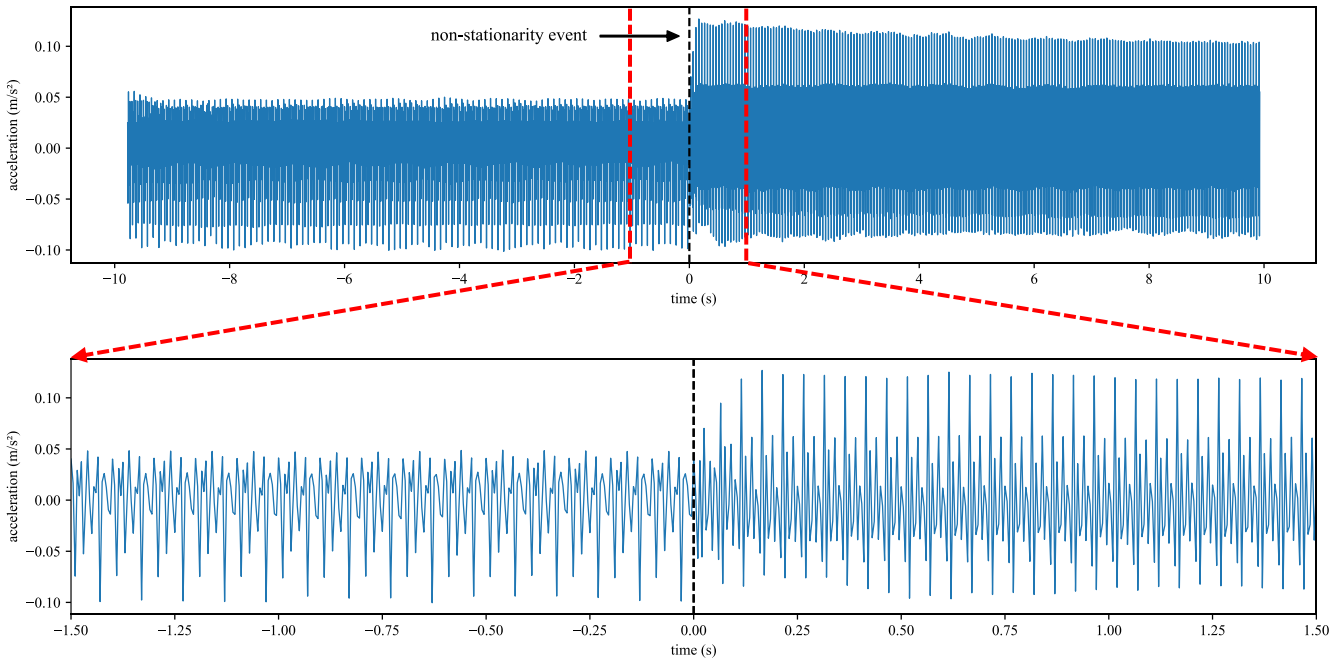

Figure 4: Experimental data showing the: (top) full dataset that the models were trained on, and; (bottom) a close-up within +/- 1 s of the non-stationarity event. Note that data is adjusted so that the non-stationarity event occurs at 0 s.

Each algorithm's MLP was trained on the data produced by DROPBEAR as shown if Figure 4 but down-sampled by factor of 256 to accelerate training and inference times. In the figures that follow, the *x*-axis is adjusted by 9.775 s, therefore resulting in the non-stationary event to occur at *t*=0 in the experimentally generated signal.

RESULTS AND DISCUSSION

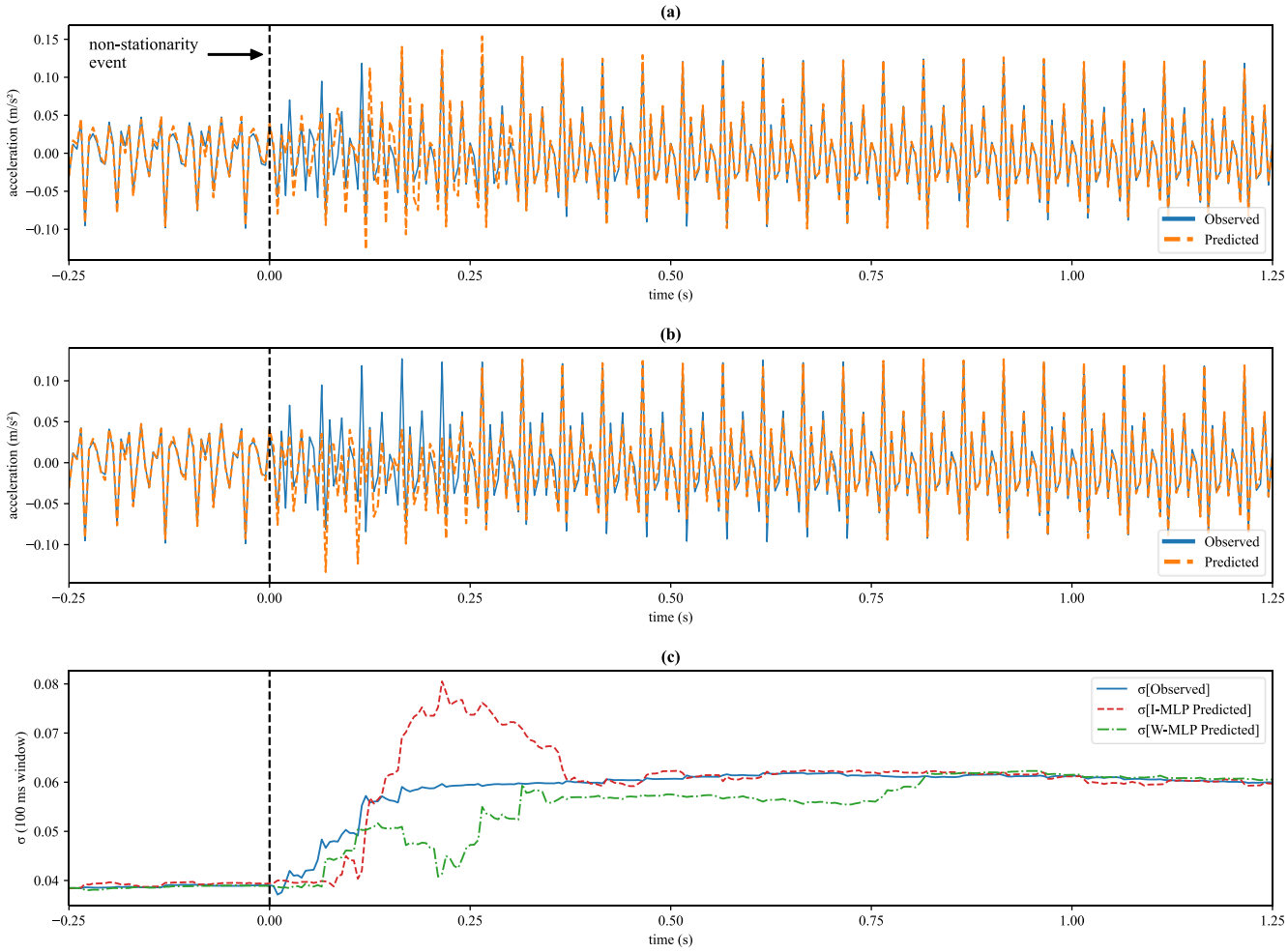

Figure 5: The time series predictions made by (a) I-MLP, (b) W-MLP within [-0.25, +1.25] s of the non-stationarity event, and; (c) the sliding standard deviations of the observed data and the predictions made by each algorithm using a 100 ms window.

To analyze the prediction results of the two algorithms, the behavior of the prediction results near the non-stationary event is evaluated. To evaluate the response of each algorithm to the non-stationarity event in a decoupled fashion, we analyze the behavior of the standard deviation using a 100 millisecond sliding window for each algorithm's time series predictions and compare these results to the 100 millisecond sliding standard deviation of the observed, ground truth signal. The sliding standard deviation is calculated per-sample using the following equation:

$$\boldsymbol{\sigma}[D](t) = \sqrt{\frac{1}{N}\sum_{i\in\mathcal{I}(t)}(x_i - \bar{x})^2}$$

where D represents the dataset over which the sliding standard deviation is being calculated (here, the ground truth, the predictions made by I-MLP, and the predictions made by W-MLP), x_i is the i^{th} sample of the dataset, $\mathcal{I}(t)$ is the set of indices of the samples with time values in the range $[t - 100\ ms, t]$, N is the cardinality of $\mathcal{I}(t)$, and $\bar{x}$ is the mean of all $x_i, i \in \mathcal{I}(t)$.

As shown in panel (c) of figure 5, it is interesting to note that the I-MLP algorithm experiences a significant jump in its sliding standard deviation before converging to the standard deviation of the ground truth, whereas in the case of the W-MLP algorithm, the sliding standard deviation experiences a less sudden change in standard deviation but takes a longer time to converge to the ground truth sliding standard deviation.

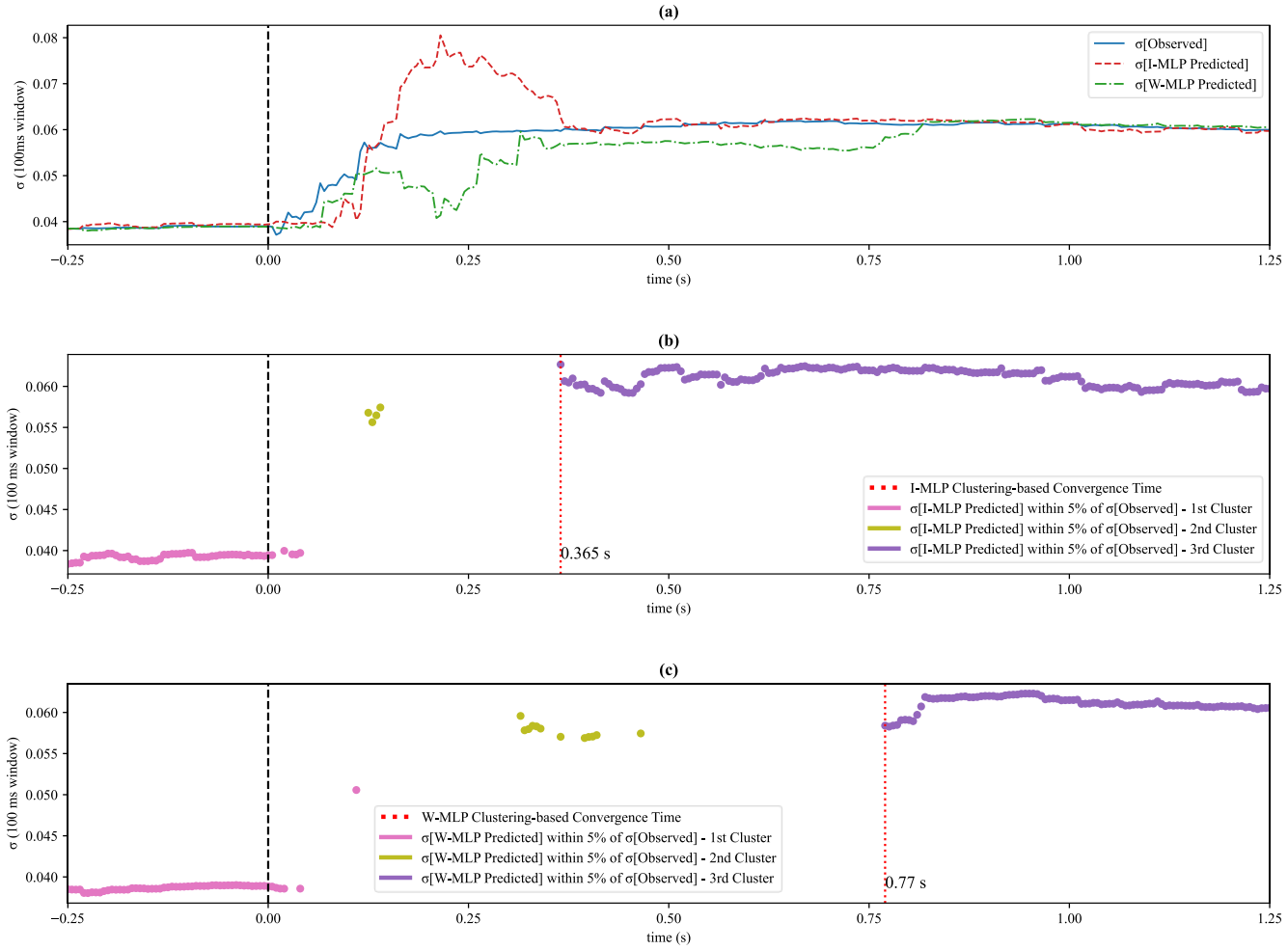

Figure 6: Clustering-based convergence time showing: (a) the sliding standard deviations of the observed data and the predictions made by each algorithm using a 100 ms window (same as Figure 4, panel (c)); (b) results of hierarchical clustering on standard deviation values within 5% of the observed for I-MLP, and; (c) results of hierarchical clustering on standard deviation values within 5% of the observed for W-MLP.

To quantify the amount of time it takes each model to converge to the sliding standard deviation of the ground truth, we apply agglomerative hierarchical clustering [9] along the time dimension of the datapoints within +/- 5% of the sliding standard deviation values of the ground truth. This method accounts for any instances where the sliding standard deviation of a model comes close to the sliding standard deviation of the ground truth but then re-diverges before truly converging. This work uses a cluster size of 3 and employs a linkage criterion based on the minimum Euclidean distances between the samples in each cluster. As depicted in Figure 6, the convergence time for each

algorithm is equal to the time coordinate of the first datapoint of the rightmost cluster. Based on this method, I-MLP is shown to converge just over twice as fast as W-MLP. This indicates that, in real-world scenarios, I-MLP would theoretically be able to more quickly adapt to sudden changes in sub-millisecond environments, as unlike W-MLP it does not re-initialize the weights of its neural network after a certain number of timesteps.

Figure 7 analyzes the 10 and 100 ms sliding RMSE windows for each algorithm, as calculated between the predictions of each algorithm and the ground truth observations. Based on the overlayed 10 ms windows in Figure 7(c), it is noted that in the short-term lookback case, I-MLP appears to behave less volatile and remain mostly below the values of W-MLP after the non-stationarity event. The long-term lookback case shows the sliding RMSE of W-MLP to peak above the sliding RMSE of I-MLP before re-convergence. Here, I-MLP outperforms W-MLP on the interval from 0.37 s to 0.82 s, though both algorithms perform comparably from thereon.

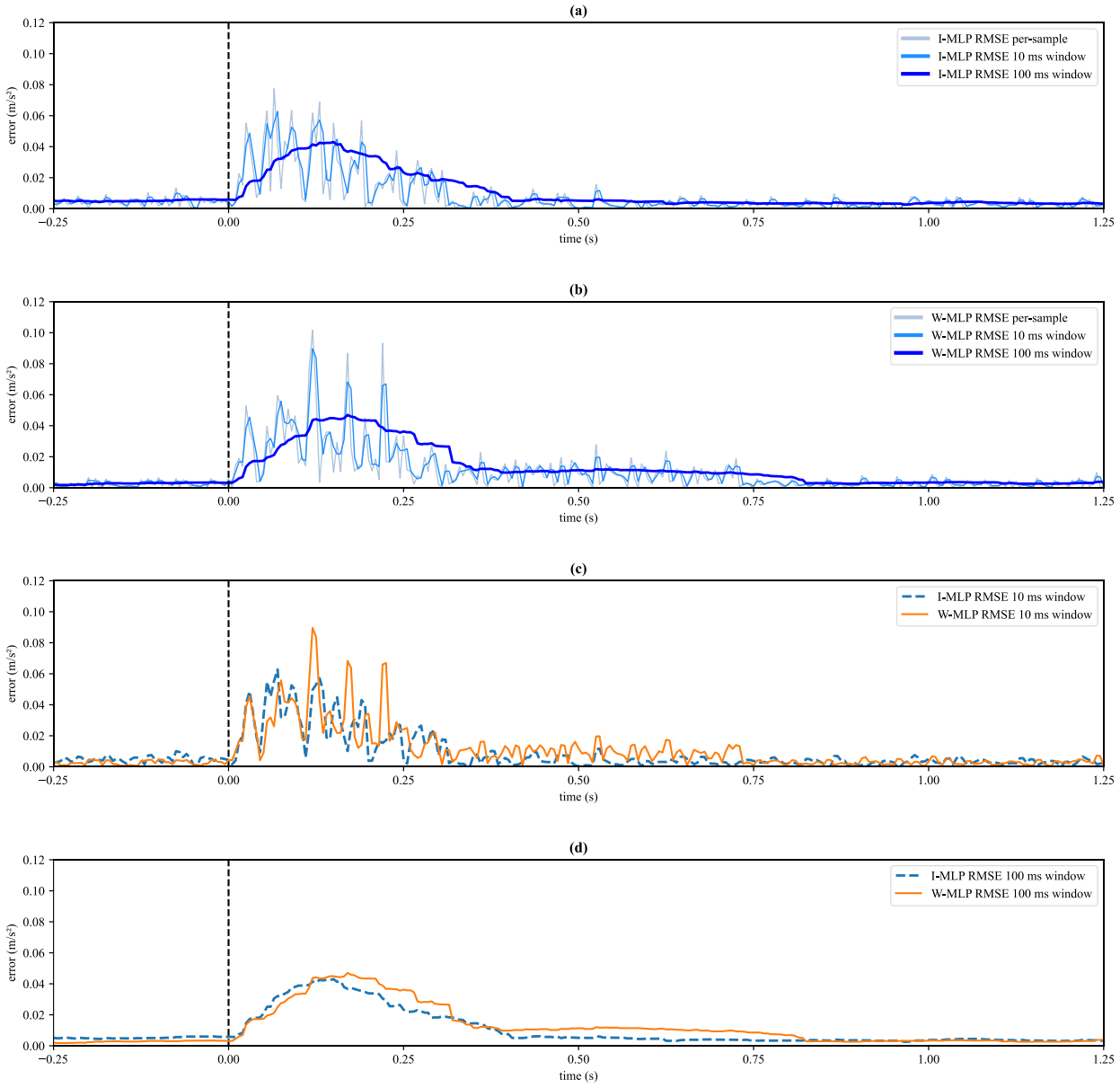

Figure 7: Sliding RMSEs for (a) I-MLP and (b) W-MLP; overlays of the (c) 10 ms RMSE sliding window and (d) 100 ms RMSE sliding window for the I-MLP and W-MLP respectively.

The discrepancy in the sliding RMSE is captured in the global RMSE values calculated over the whole of the dataset for each algorithm. As shown in figure 8, the overall RMSE for I-MLP is nearly 40% higher than the overall RMSE of W-MLP, indicating that periodically re-initializing a neural network's weights has an overall positive effect on prediction accuracy. Furthermore, it is also worth noting that I-MLP takes over three times as long to run compared to W-MLP, likely due to its need to re-train its neural network using multiple epochs upon obtaining a new sample. In the future, it may be amenable to test the performance of I-MLP with higher learning rates using a single epoch per sample, as this could provide the added benefit of shorter computation times with similar levels of performance to the results yielded by the multiple samples per epoch approach used in this work.

Figure 8 shows that both models achieve comparable time response assurance criterion (TRAC) scores [10]. TRAC was developed to quantify the similarity between time traces and has a value between 0 (no correlation) and 1 (perfect match). The W-MLP's better overall accuracy is captured by its slightly higher TRAC score.

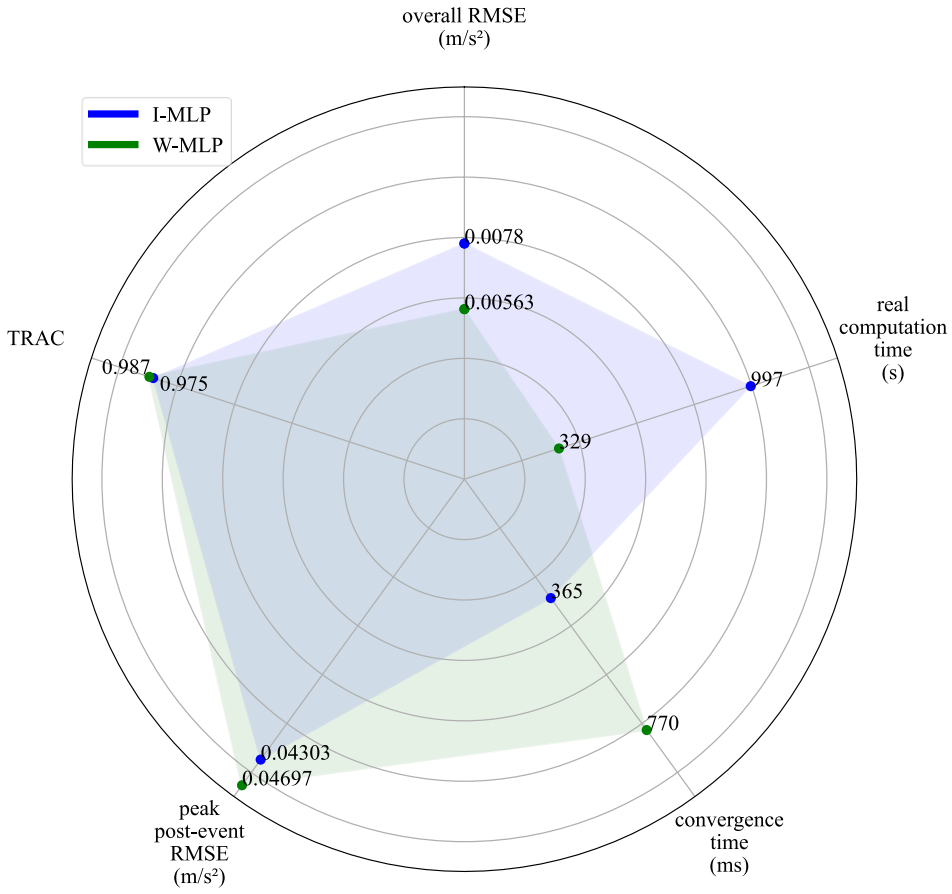

Figure 8: Radar plot presenting cumulative metrics for each algorithm. Real computation time values are based on a full traversal of the dataset on a workstation computer with an Intel Core i7-7600U series CPU.

CONCLUSION

This research presents a comparison of two neural network-based approaches for making real-time time series predictions on vibration signals that exhibit non-stationarity events. The first algorithm is the Windowed-MLP (W-MLP) which trains an MLP on a fixed-size window of input data, and then uses that MLP to predict one or more future windows of data, repeatedly training and replacing the MLP as new input data becomes available. The second algorithm is the Iterative-MLP (I-MLP) algorithm which uses a single MLP over the course of the whole dataset

and is updated with each new sample that is observed. Metrics for the preliminary results show that W-MLP performs better in overall error (measured as the root mean square error) and requires less computational resources while I-MLP converges faster following a non-stationarity event. In broader terms, preliminary results show that periodically re-initializing a neural network's weights leads to higher overall accuracy in such a setting, at the expense of taking longer to re-converge after experiencing non-stationary events.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation Grant numbers 1937535 and 1956071. This work is also partially funded by the University Of South Carolina through the Magellan Scholar Award (155400-20-53644). 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 University of South Carolina. The authors would also like to thank the University of South Carolina's Research Computing Infrastructure team for giving us access to their Hyperion high-performance computing cluster for training our models for this research. We would also like to thank our collaborators at Iowa State University for their valuable input.

REFERENCES

[1] L. Rüdiger, M. Andrae, P. Warnstedt, W. Xiao and N. Gebbeken, "Blast Mitigation Barriers for Urban Places," in *ISIEMS Proceedings*, 2017.

[2] J. Hong, S. Laflamme and J. Dodson, "Study of Input Space for State Estimation of High-rate Dynamics," *Structural Control and Health Monitoring,* vol. 25, no. 6, p. e2519, 2018.

[3] A. Downey, J. Hong, B. Joyce, J. Dodson, C. Hu and S. LaFlamme, "Methodology for Real-time State Estimation at Unobserved Locations for Structures Experiencing High-rate Dynamics," in *Structural Health Monitoring*, 2019.

[4] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *MIT Press,* vol. 9, no. 8, pp. 1735-1780, 1997.

[5] P. Filonov, A. Lavrentyev and A. Vorontsov, "Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model," in *NIPS Time Series Workshop,* Barcelona, 2016.

[6] F. A. Gers, D. Eck and J. Schmidhuber, "Applying LSTM to Time Series Predictable Through Time-Window Approaches," in *Neural Nets WIRN Vietri-01*, 2002.

[7] Y. Fan, K. Xu, H. Wu, Y. Zheng and B. Tao, "Spatiotemporal Modeling for Nonlinear Distributed Thermal Processes Based on KL Decomposition, MLP and LSTM Network," *IEEE Access,* vol. 8, pp. 25111-25121, 2020.

[8] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[9] F. Nielsen, "Chapter 8: Hierarchical Clustering," in *Introduction to HPC with MPI for Data Science*, Springer, Cham, 2016, pp. 195-211.

[10] T. Van Zandt, "Development of Efficient Reduced Models for Multi-body Dynamics Simulations of Helicopter Wing Missile Configurations," ProQuest Dissertations Publishing, 2006.