REAL-TIME FORECASTING OF VIBRATIONS WITH NON-STATIONARITIES

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

UNIVERSITY OF South Carolina

UofSC South Carolina

Table of Contents

UofSC South Carolina

Structures Experiencing High-Rate Dynamic Events

Ballistics Packages

Hypersonic Vehicles

Active Blast Mitigation

South Carolina

Formal Definition for High-Rate Dynamic Events

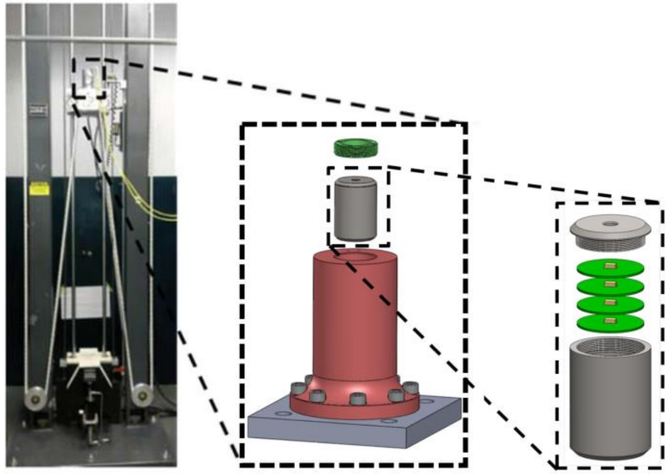
High-rate dynamics are described as a dynamic response from a high-rate (<100 ms) and high-amplitude (acceleration > 100 *g*) event such as a blast or impact.

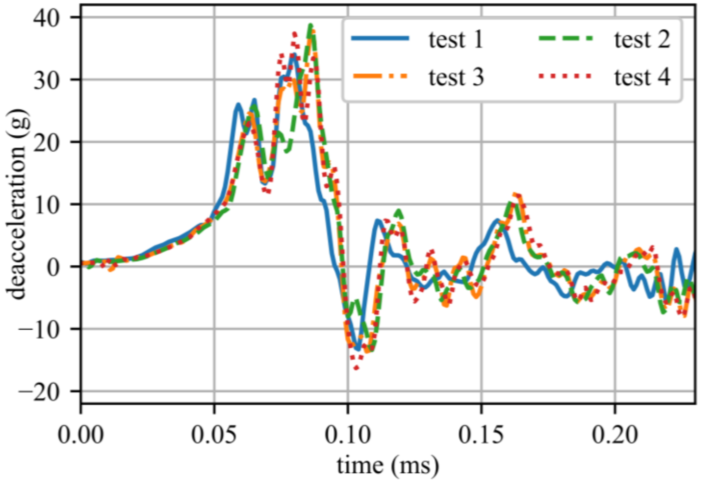
The high-rate problem contains many complexities that can be summarized as having:

1) large uncertainties in the external loads;

2) high levels of non-stationarities and heavy disturbances; and

3) generated unmodeled dynamics from changes in system configuration.



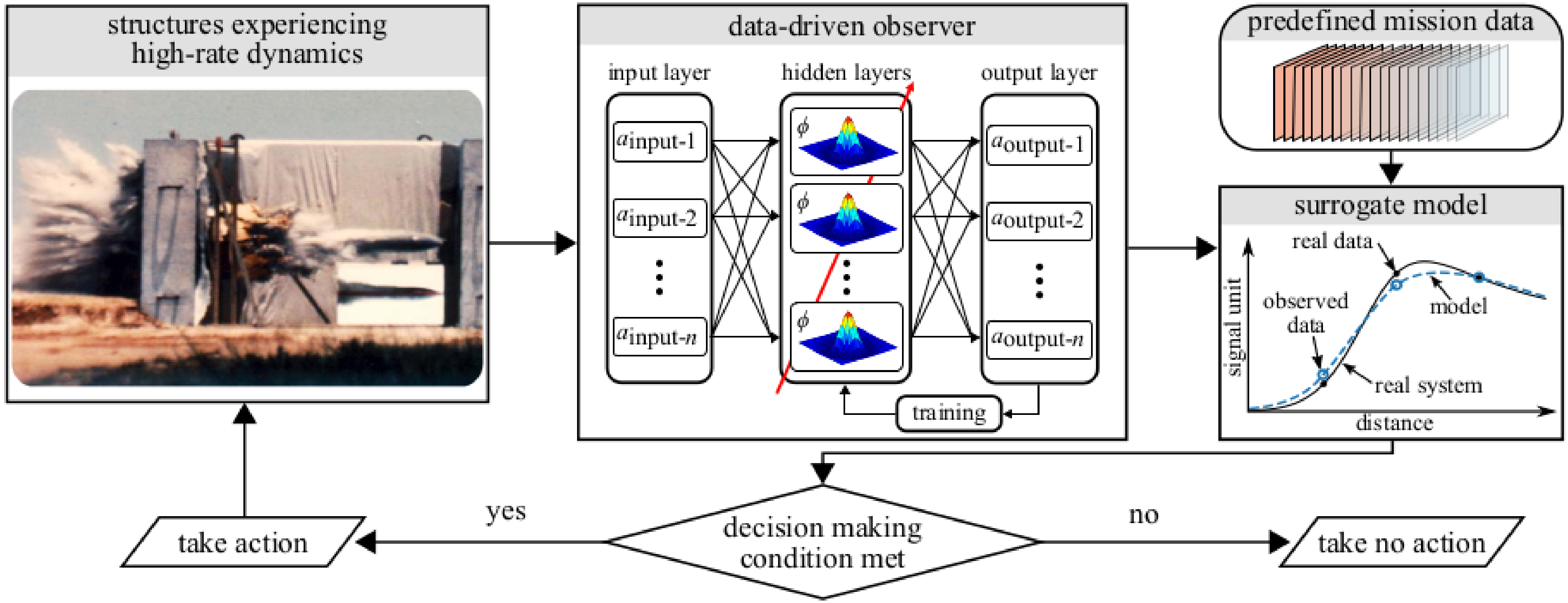


Long-Term Goal: Real-Time Decision-Making for Structures Experiencing High-Rate Dynamics

Real-time decision making requires the development of two key enabling technologies:

1) low-latency (2 ms) model updating; and

2) near-time prognostics of the system state.



Challenges Related to Computing on the Edge

In the development of solutions for this problem we are operating within the following constraints:

- Computational power at the edge is limited. This includes memory, processors, and available energy.
- The system is too complex to pre-calculate a library of existing fault cases.
- The inputs (forces, location) will never be known.
- Rare and extreme events will happen and must be accounted for.

Team Eglin Public Affairs

tonyrogers.com

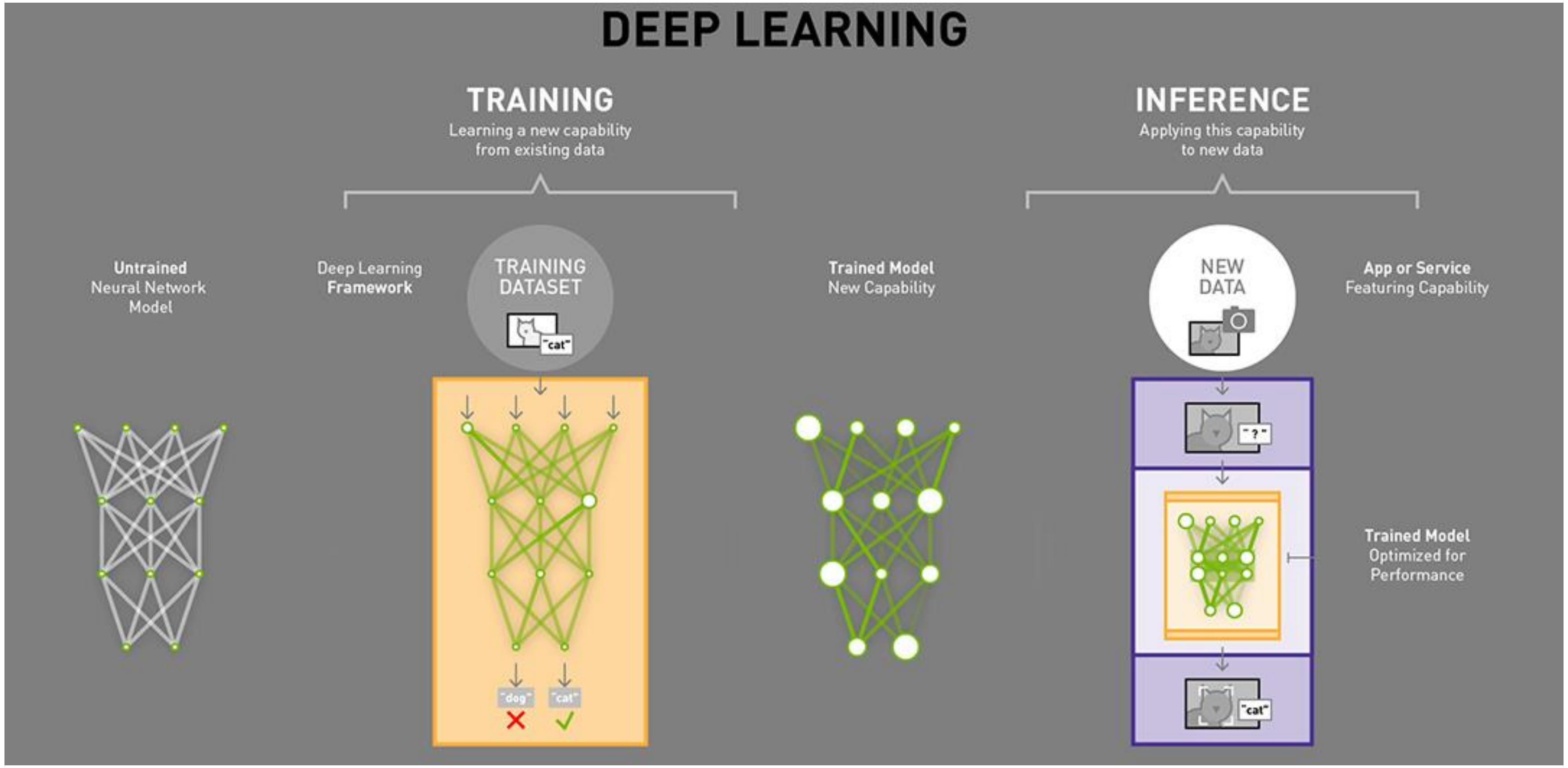
Team Eglin Public Affairs

Methodology

Background: Machine Learning, Training Vs. Inference

Inference is where capabilities learned during training are put to work:

- Training: Learning a new capability from existing data.
- Inference: Applying this capability to new data.



nvidia.com/blog/2016/08/22/difference-deep-learning-training-inference-ai/

Algorithmic Approaches

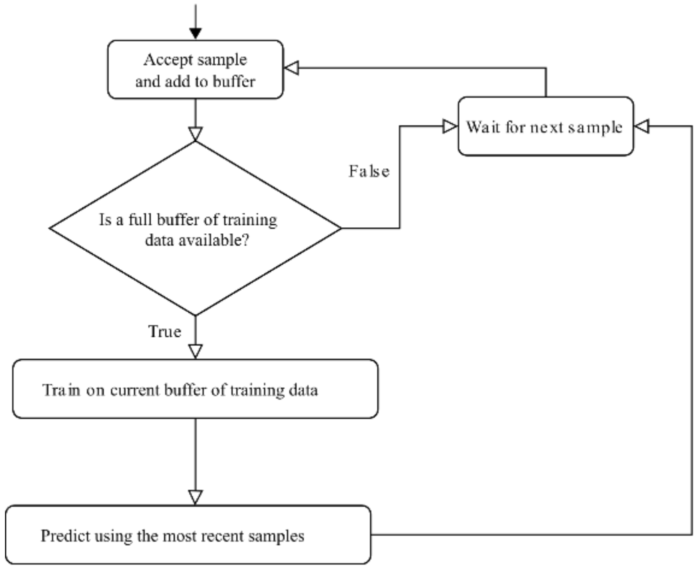
I-MLP (“**I**terative **MLP**”)
- A single MLP model is updated sample-by-sample over the course of the whole dataset
- New prediction is produced for a pre-specified number of timesteps in the future when an internal buffer reaches capacity
- Parameters: **history length**, **forecast length**

W-MLP (“**W**indowed **MLP**”)
- MLP model is replaced every certain number of timesteps
- Trained across a fixed-length buffer of training window samples, sliced iteratively into sample windows with non-overlapping slices
- Parameters: **training window**, **sample window**

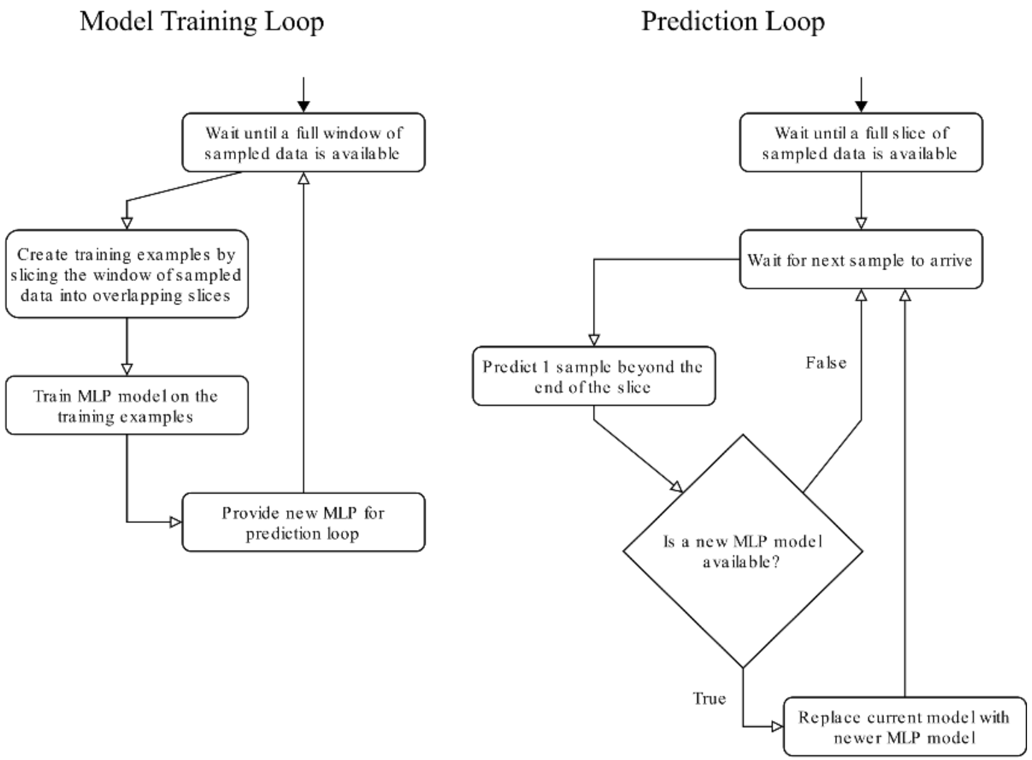
Common parameters: **training epochs**, **hidden layer size**

UofSC South Carolina

Algorithmic Approaches

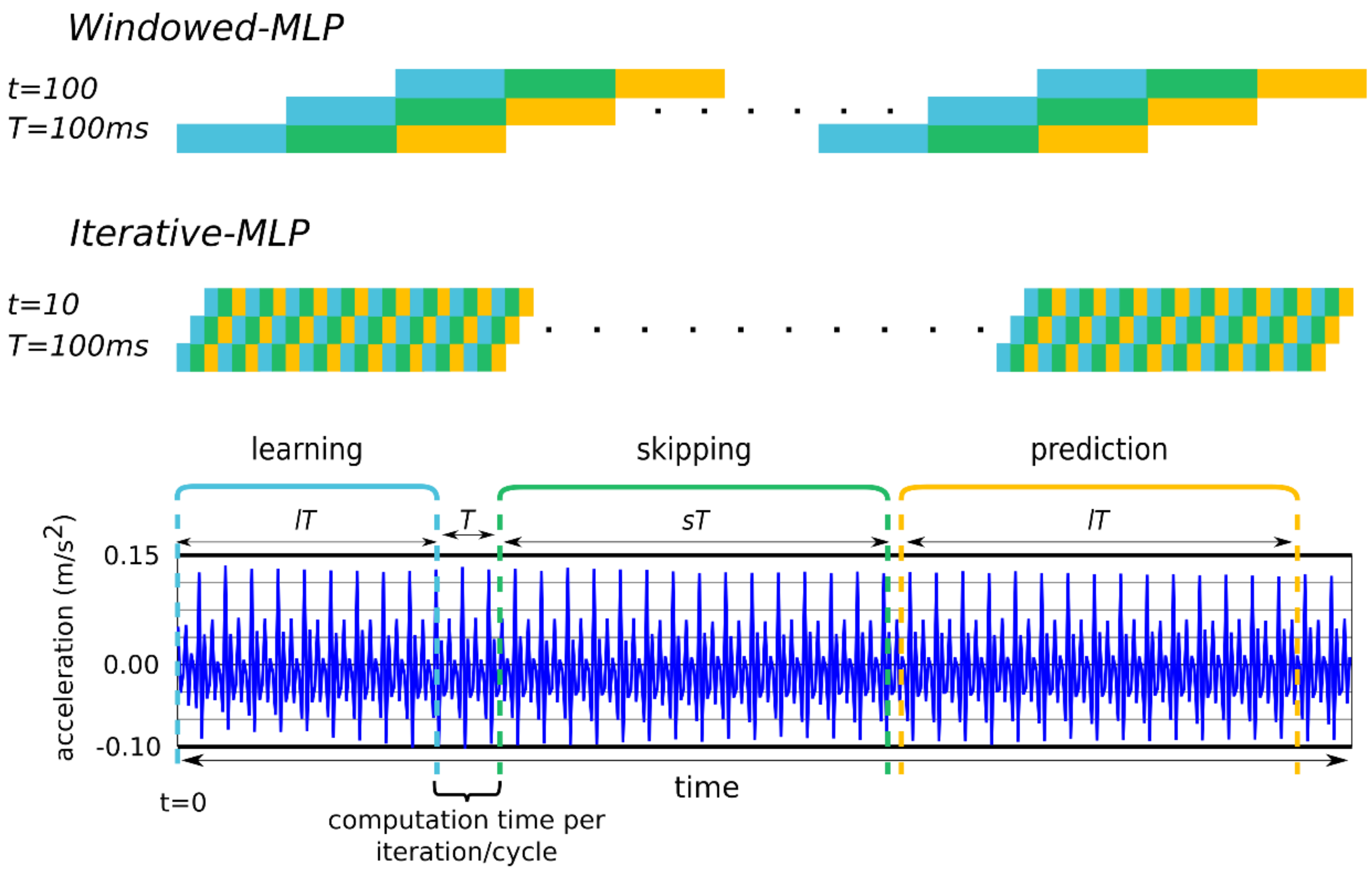


I-MLP Algorithm



W-MLP Algorithm

Algorithmic Approaches



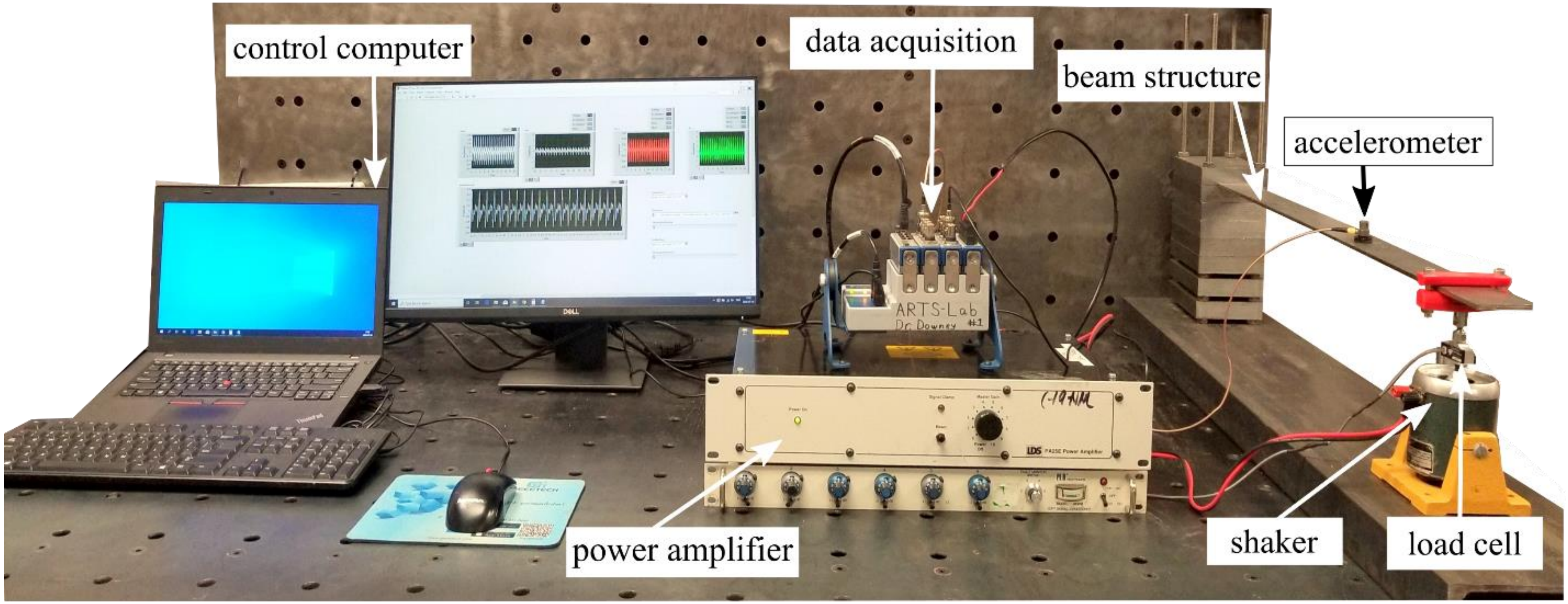
Test Bench
&
Training Data

Data Generation

- Steel cantilever beam setup is used to produce data consisting of high-rate dynamic events
- Data produced is a vibration signal that contains a **non-stationary event** (NSE) characterized by a sudden shift in amplitude
- Goal is to compare the performance of I-MLP and W-MLP when subject to the NSE
- This collapses down to an **online time-series prediction problem**

South Carolina

Test Bench Setup

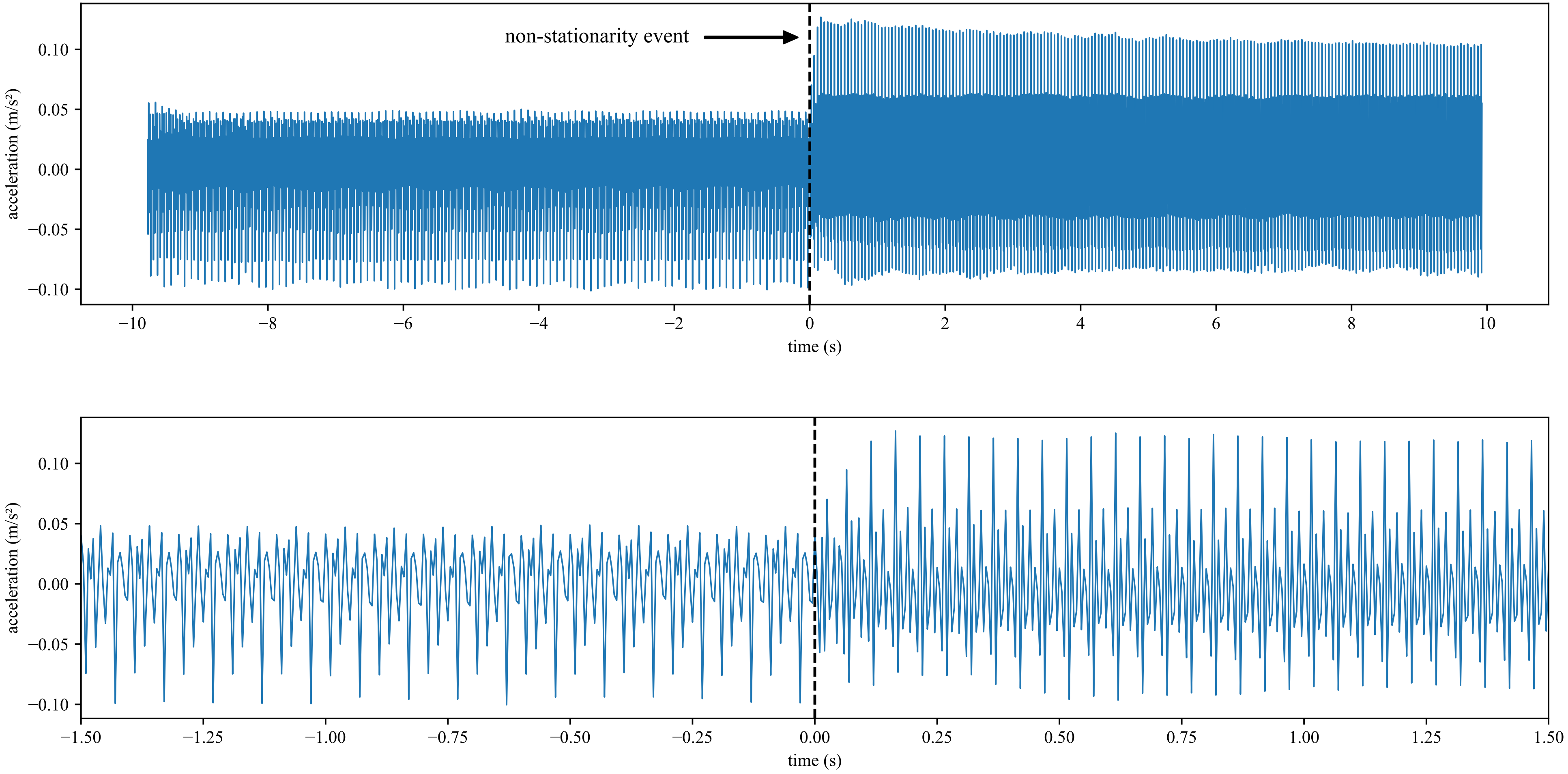


South Carolina

Training Data

- Total of 19.7 s of vibration data, with NSE occurring 9.775 s into the dataset
- Dataset is **grounded** at NSE
- Final dataset is a 256x down-sample of the original raw data to accelerate training and inference times

Training Data



South Carolina

Results

UofSC South Carolina

Optimal Model Configurations

I-MLP:
History length = 40
Forecast length = 10
Epochs per sample = 10
Hidden layer size = 100

W-MLP:
Training window = 100
Sample window = 50
Epochs per window = 200
Hidden layer size = 10

Convergence Time Analysis

- To directly compare each algorithm's response to the NSE, we analyze the **100 ms sliding standard deviation** of each algorithm's predictions compared to the same for the ground truth

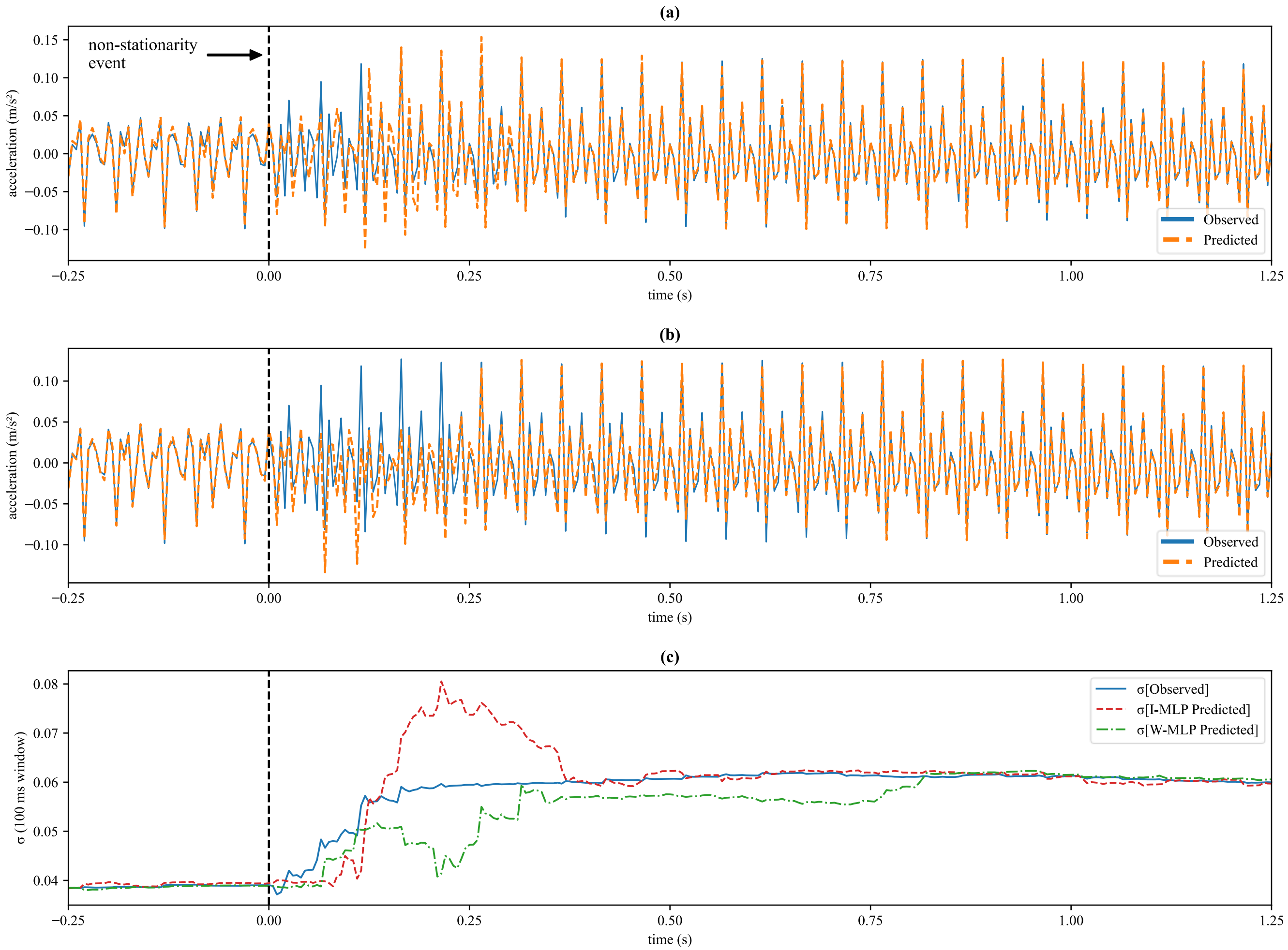
- The sliding standard deviation is calculated per-sample using the following equation:

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

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, $\mathcal{J}(t)$ is the set of indices of the samples with time values in the range $[t-100\ ms, t]$, N is the cardinality of $\mathcal{J}(t)$, and $\bar{x}$ is the mean of all $x_i, i \in \mathcal{J}(t)$

South Carolina

Convergence Time Analysis



Left: 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.

It is interesting to note that **I-MLP experiences a significant jump** in its sliding standard deviation before converging to the standard deviation of the ground truth, whereas **in the case of W-MLP, 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.

South Carolina

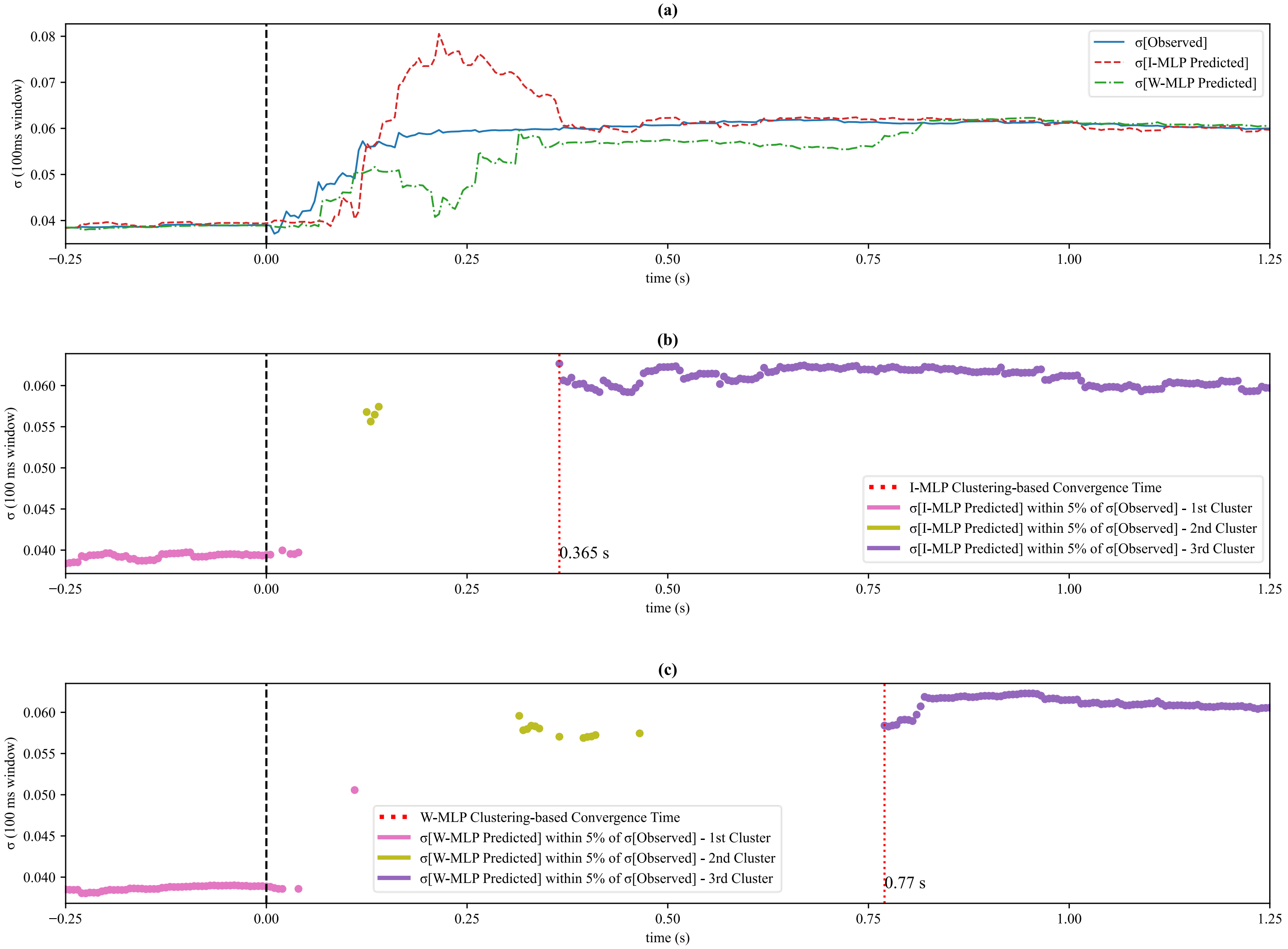
Convergence Time Analysis

- To objectively determine the amount of time each model takes to converge, we apply **agglomerative hierarchical clustering** 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

South Carolina

Convergence Time Analysis



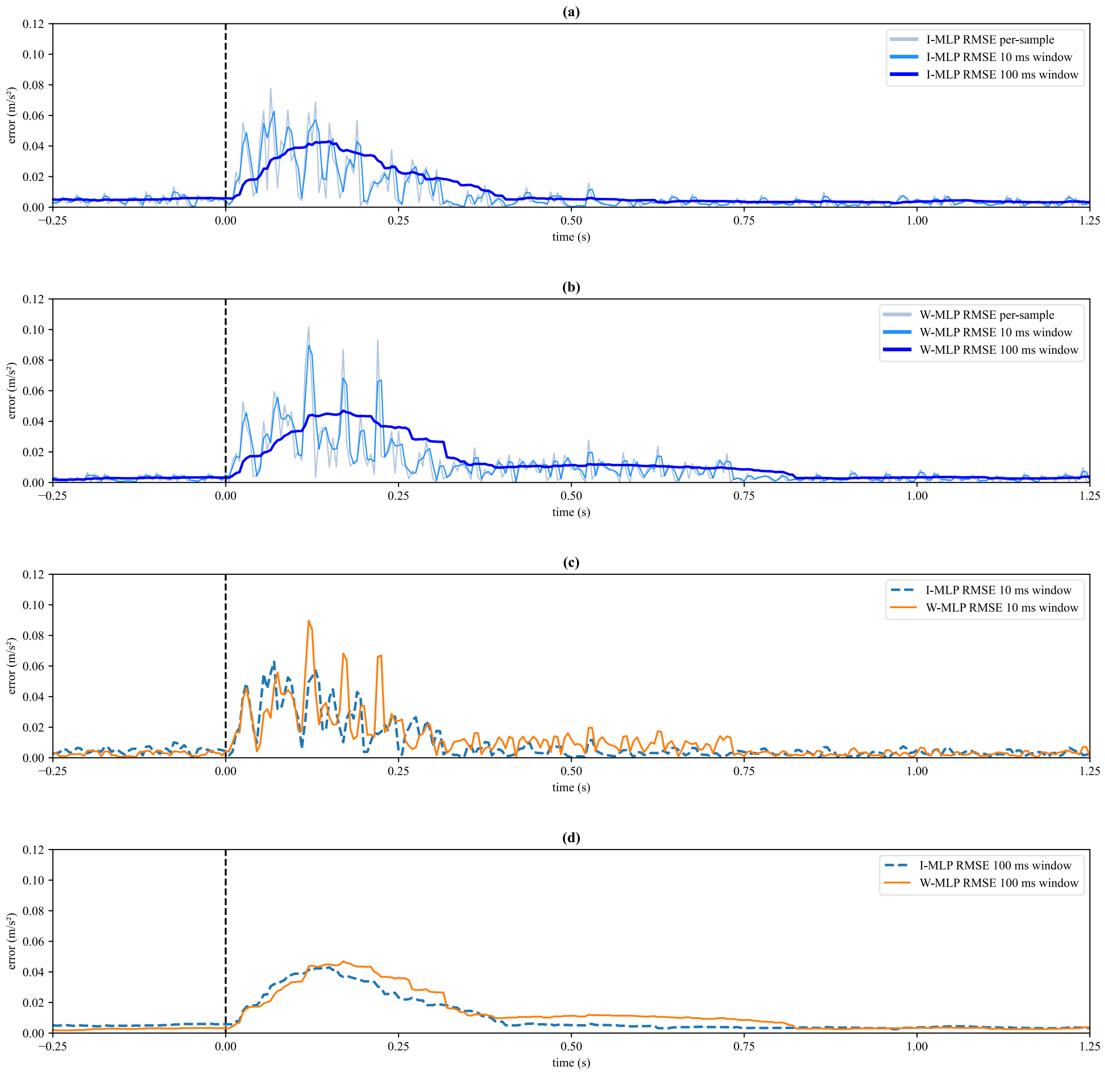
Left: (a) Sliding standard deviations of the observed data and the predictions; (b) results of hierarchical clustering for I-MLP, and; (c) results of hierarchical clustering for W-MLP. The convergence time for each algorithm is equal to the time coordinate of the first datapoint of the rightmost cluster.

Hyperparameters: cluster size of 3, linkage criterion based on the minimum Euclidean distances between the samples in each cluster

Based on this method, **I-MLP converges just over twice as fast as W-MLP**.

UofSC South Carolina

Sliding RMSE Windows



Left: 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.

➢In the short-term lookback case, I-MLP appears to behave less volatile and remain mostly below the values of W-MLP after the NSE

➢Long-term lookback case shows the sliding RMSE of W-MLP to peak above the sliding RMSE of I-MLP before re-convergence

➢**I-MLP outperforms W-MLP on the interval from 0.37 s to 0.82 s**, though both algorithms perform comparably from thereon

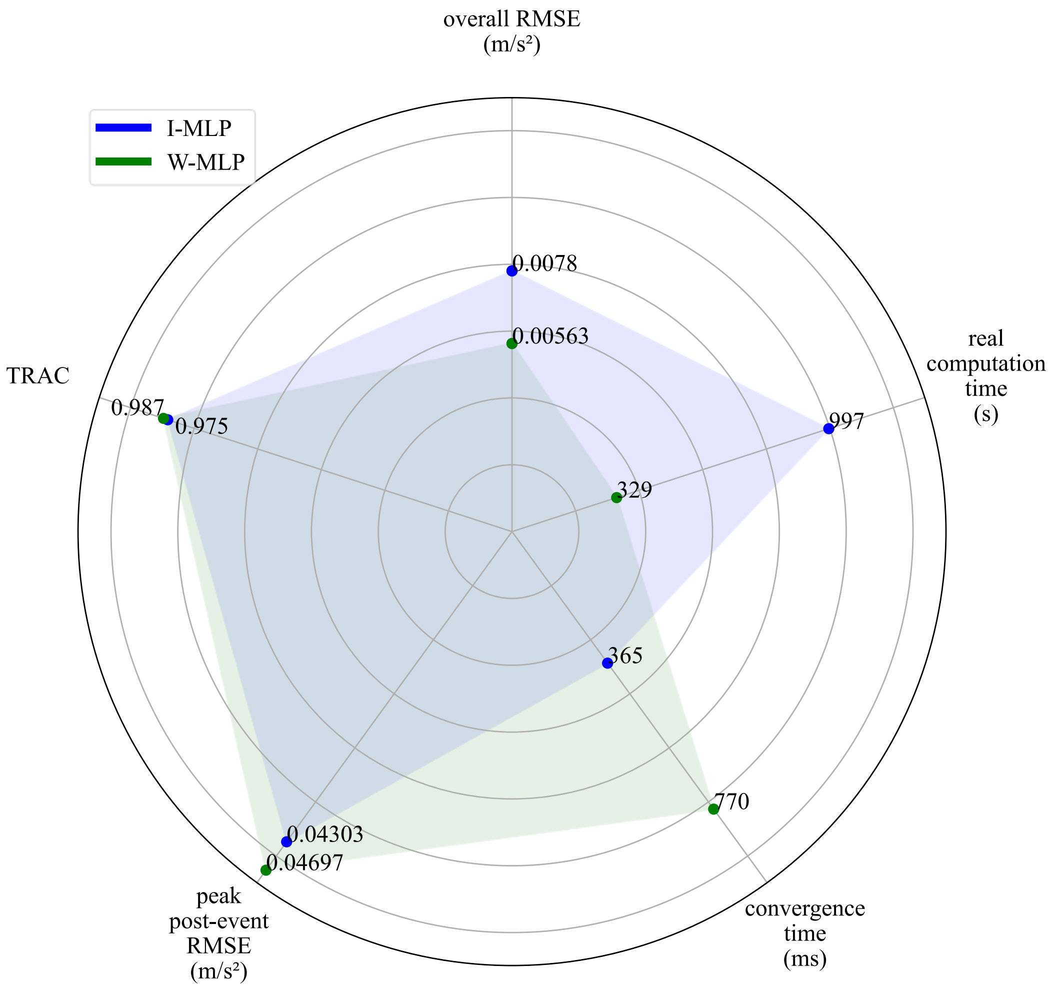
South Carolina

Cumulative Metrics

Right: 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.

➢**I-MLP overall RMSE is nearly 40% higher than W-MLP overall RMSE**, indicating that periodically re-initializing a neural network's weights has a **positive effect** on prediction accuracy

➢**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



South Carolina

Conclusions & Future Work

South Carolina

- **W-MLP performs better in overall error** (measured as the root mean square error) and requires less computational resources

- **I-MLP converges faster** following an NSE

- More broadly: **periodically re-initializing a neural network's weights leads to higher overall accuracy** in online time series prediction, **at the expense of longer re-convergence** after experiencing NSEs

Future work:

- Evaluate tradeoff between learning rate and epochs for I-MLPs
- Experiment with other machine learning architectures (LSTM, gradient boosting, random forest, etc.)

Thanks!

UofSC South Carolina