Performance Evaluation of the TriLS System on Synthetic Data

This document presents the results of testing the TriLS system on multivariate synthetic time series batch data that was generated for the purpose of this work. This enabled the evaluation of the system's ability to adapt to a controlled drift. The description and a link to the generated set are provided for the reference.

0.1 Synthetic Data Description

A sum of sinusoidal terms was implemented as a process generating model of time series with dynamic profiles. The data were produced as time-series sequences (batches) to simulate process realizations. For simplicity, we set the generator to output values only in the interval $[0, 1]$ with the rate of 1 kHz ($T = 0.001$ s). The length of each realization was set to 1 000 samples. We created 90 realizations using the first model and switched to the second model for the next 60 realizations. This introduced a drift in which a function describing the relation between the features and the target change. We also introduced a series of gradual and abrupt real drifts by varying only the bias component of the process generating model. The equation of the first model is described as:

$$\begin{aligned} y_1(n) = a_0 + a_1 sin(2\pi f_1 T n) + a_2 sin(2\pi f_2 T n) + \\ + a_3 sin^3(2\pi f_3 T n) + a_4 sin(2\pi f_4 T n) + \xi(n). \end{aligned} \tag{1}$$

The component frequencies $\{f_m\}_{m=1}^4$ and amplitudes $\{a_m\}_{m=1}^4$ are provided in Table 1. The bias term a_0 was altered in a controlled manner to simulate a gradual concept drift. This part of the dataset depicts a repetitive process with a gradual drift.
The second part of the synthetic dataset was generated in accordance with the following model:

$$\begin{aligned} y_2(n) = a_0 + a_1 cos(2\pi f_1 T n) + a_2 sin^5(2\pi f_2 T n) \\ + a_3 sin(2\pi f_3 T n) + \xi(n). \end{aligned} \tag{2}$$

Frequencies and amplitudes of this model are also summarised in Table 1. Here, a_0 was altered in an abrupt manner, so that this part of the set depicts a repetitive process with several sudden large drifts. In both models,

Table 1: Parameters of process-generating models

	$\mathbf{y_1}$		$\mathbf{y_2}$	
m	$\mathbf{f_m}$ **[Hz]**	$\mathbf{a_m}$	$\mathbf{f_m}$ **[Hz]**	$\mathbf{a_m}$
0	0	$[0.4, 0.5]$	0	$[0.4, 0.6]$
1	11	0.03	3	0.1
2	4	0.05	5	0.2
3	1	0.1	35	0.02
4	40	0.03	-	-

$\xi(n) \sim N(\mu = 0, \sigma = 0.04)$ represents a real-valued white Gaussian noise. This corresponds to $SNR = 10.4\ dB$ per sample for the concept generated by y_1, and to $SNR = 19.5\ dB$ per sample for y_2. To ensure that only a_0 contributed to the bias change from one realization to another in both y_1 and y_2, all realizations had the same length of 1 000 and the summation of all terms averaged to a_0 over each realization. A snapshot of a realization produced by each model is shown in Fig. 1.

The features are a set of sinusoids with frequencies ranging in $f \in [0.5, 40]\ Hz$ with $0.5\ Hz$ separation. The total number of features is 80 functions. A complete snapshot of the set used for the evaluation is available at a GitHub repository[1].

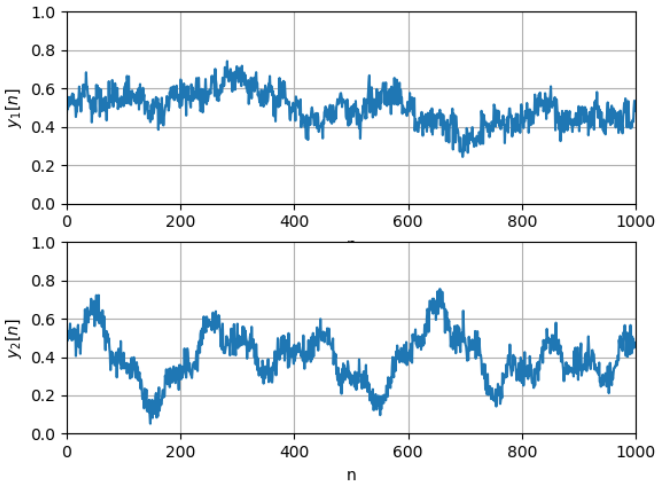

Figure 1: Samples of the target y variable generated by two models (one realization in each case)

[1] https://github.com/Orlamed/TriLS_SyntheticDataSet.git

0.2 TriLS System Performance Evaluation

In this experiment, the pool of basis functions for FOS at Layer 3 was composed of sinusoidal features and their second-order polynomial terms, with a total size of 3 320 basis functions. As one may note, the pool did not include higher-order terms comprising y_1 and y_2; thus, FOS was constrained to fit models with available terms only. The algorithm was set to build models of up to $\mathcal{M}_{max} = 5$ terms with $\mathcal{Q} = 0.0001$.

The terms and coefficients of an initial model $M_{in}(\vec{a}_{in})$ were calculated at Layer 3 using the first $L_{in} = 30$ process realizations, and set at Layers 1-2. For each subsequent realization at time $i = k$ ($k > L_{in}$), the deployed version of a model $M(\vec{a})$ provided online prediction and was assessed at the end of a realization. Depending on the state of the TriLS, a model was either modified or used during the next realization as is. The dynamic window length L was limited to a maximum of $L_{max} = L_{in} = 30$ realizations to reflect the limitation of local memory and also to prevent the E-ADWIN from growing the window indefinitely when a concept is stable. The δ parameter of E-ADWIN was set to 0.001 and stable concept waiting time was set to $D_s = 10$ realizations.

To provide a baseline for evaluation, we implemented two additional techniques using a sliding window with a constant length, which is a common approach in the literature used as a performance reference in adaptive modeling. In the first technique, an initial model $M_{in}(\vec{a}_{in})$ was tuned after every process realization in a static window of $L_{st} = L_{max} = 30$ recent realizations. In the second technique, a new model $M(\vec{a})$ was built at the cloud, after every process realization, using L_{st} recent realizations. We also added the performance of an initial model $M_{in}(\vec{a}_{in})$ with no tuning on tested realizations. This performance demonstrates a deterioration of a nonadaptive model due to a drifting concept. All parameters for this experiment are summarized in Table 2.

Dynamic window size L calculated by E-ADWIN after every realization and sample averages $\hat{m}_i$ of tested realizations in L are shown in Fig. 2. This figure also demonstrates the average of each realization m_i, which outlines the drifting bias a_0. In this plot, one can observe the designed gradual drift between $43 < i < 60$ and three abrupt drifts at $i = 90, 125, 136$. The drift at $i = 90$ also includes a process generating model change from y_1 to y_2.

Table 2: Experimental parameters setting, synthetic set

Parameter	**Value**
$\mathcal{M}_{max}$	5
$\mathcal{Q}$	0.0001
δ	0.001
L_{in}	30
L_{max}	30
L_{st}	30
D_s	10

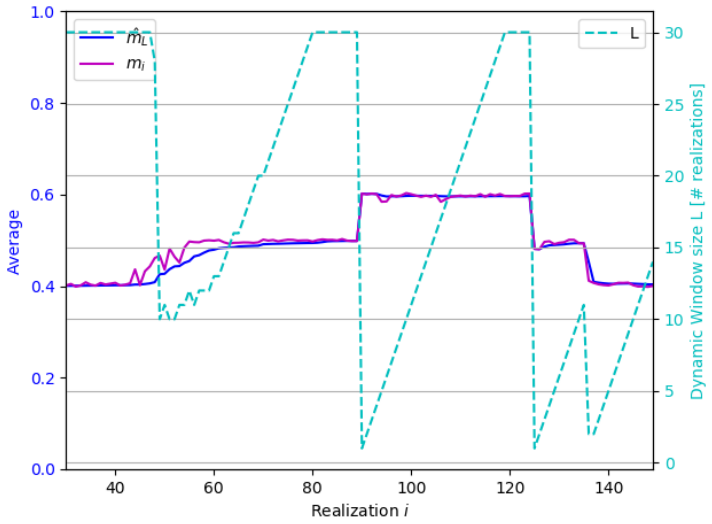

Figure 2: Synthetic Set: sample average of tested process realizations, dynamic window L size calculated by E-ADWIN, and sample average of realizations in L.

E-ADWIN responds to all drifts by reducing L. During a gradual drift, the algorithm cuts and expands the window gradually while keeping it bounded. After a drift is over, the window keeps growing linearly until it reaches the limit at L_{max}, see Fig. 2. At the times of sudden drifts, the algorithm cuts the window to 1-2 realizations in an abrupt manner, as the older part of the data in the observation window becomes irrelevant.

The prediction RMSE of all techniques on each tested realization is shown in Fig. 3. The summary of MSE, RMSE, mean absolute error (MAE), and median absolute error (MED) metrics calculated on all tested realizations is provided in Table 3. It can be seen, that the prediction error of the initial model $M_{in}(\vec{a}_{in})$ increased as soon as the concept shifted at the realization $i = 43$. Its performance degraded further when a process-generating model change happened. Even though the bias of the generator returned to its original value at the end of this experiment, the $M_{in}(\vec{a}_{in})$ precision did not improve. This is obvious as this model was trained to describe the relation between features and the output of y_1 that changed with the drift. On the other hand, the adaptive techniques demonstrated significant improvement in performance as compared to nonadaptive $M_{in}(\vec{a}_{in})$. The improvement of each technique over $M_{in}(\vec{a}_{in})$ in percent is shown in Fig. 4.

In accordance with all metrics in Fig. 4, the TriLS approach demonstrated the largest improvement and tuning of $M_{in}(\vec{a}_{in})$ approach the smallest one. The reasons for the inferior performance of the tuning only technique are that it used the terms of $M_{in}(\vec{a}_{in})$ and a static window for adapting. As a result, it

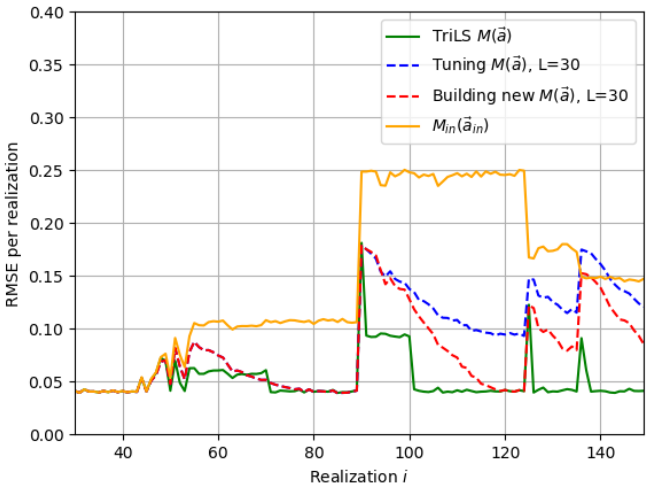

Figure 3: Synthetic Set, RMSE per tested realization: $M_{in}(\vec{a}_{in})$ (orange), tuned $M_{in}(\vec{a}_{in})$ in a static window (blue), new $M(\vec{a})$ in a static window (red), and adaptive model with TriLS (green).

Table 3: Synthetic Set: performance comparison, nonadaptive model, model tuning in a static window, building new model in a static window, and tuning/building with TriLS

Approach	**MSE**	**RMSE**	**MAE**	**MED**
$M_{in}(\vec{a}_{in})$	0.027	0.164	0.124	0.096
Tuning $M_{in}(\vec{a}_{in})$, Static $L = 30$	0.01	0.101	0.074	0.054
Building new $M(\vec{a})$, Static $L = 30$	0.007	0.086	0.065	0.05
TriLS, Dynamic L	0.003	0.055	0.042	0.033

performed well during gradual drifts when no change in a model was required. After abrupt drifts, this technique achieved the best performance after L_{st} realizations, when the static window included the recent concept only. When the change of a process-generating model happened, the tuning of an initial model became insufficient, resulting in a larger error as compared to TriLS and building a new $M(\vec{a})$ approaches. Building a new model after every realization generally succeeded to keep the good quality of performance in the presence of concept drift. However, as can be seen from the Fig. 3, this approach also took L_{st} realizations after an abrupt drift to obtain the best fitting model. Of course, shorter static windows may help to find the best model quicker, but this is a

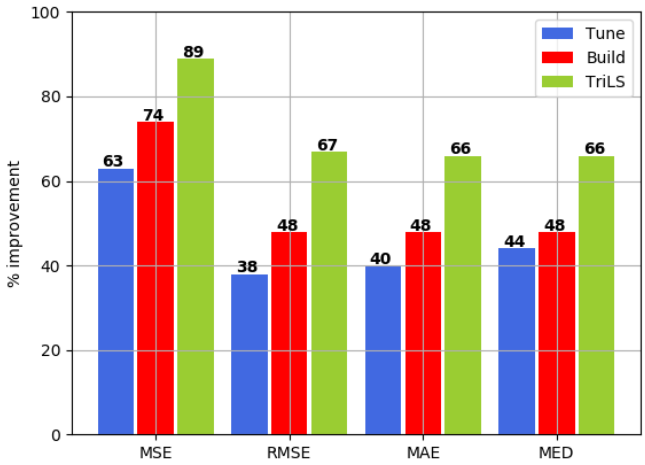

Figure 4: Synthetic Set: percentage of prediction quality improvement by adaptive techniques over nonadaptive $M_{in}(\vec{a}_{in})$.

design decision that cannot be taken a priori without knowing the dataset. In addition, this technique is slow and consumes a large amount of resources, as FOS has to search through thousands of basis functions after every realization. On the other hand, the TriLS system utilized building and tuning models in a dynamic window that contained only the relevant data. It also minimized calls for a new model, thus saving time and computational effort. Hence, it efficiently adapted the model to the drifting concept with the smallest delay among all techniques. It is also worth noting that when a process generating model change happened at $i = 90$, the TriLS first kept tuning the coefficients for the following D_s realizations. During this time, the elevated prediction error can be observed. Once the concept was assumed stabile and a new model was built using the cloud, the prediction error dropped to its minimal value of the noise standard deviation σ. The parameter D_s was a hyperparameter of the TriLS system in these studies but can be linked to the distribution of concept drifts. This is outside of the scope of this work and will be investigated in future projects.

It is interesting to note that there is no difference in the performance of tuning of a current model and building a new model approaches on the initial concept ($30 \leq i < 90$). The reason is that as long as there is no change in an underlying process generating model, building new model results in the same predictive model terms. Hence, this approach returns the same initial model $M_{in}(\vec{a}_{in})$ with an updated $\vec{a}_{in}$, which is equivalent to the result of the tuning technique, but with more consumed resources.

Our experiment also demonstrated that the TriLS considerably reduced compu-

tational effort for the IIoT device as compared to the adaptive techniques with a static window. The cumulative numbers of visits of each state by the TriLS system are summarized in Table 4. As can be seen from the table, in 83.3% of

Table 4: Synthetic Set: cumulative number of states' visits by TriLS

State	**Action**	**# of visits**	**% of visits**
0	Do nothing	100	83.3
1	Tune the current model in L	17	14.2
2	Retrain a new model in L	3	2.5
Total of tested realizations		120	100

model evaluations, the system concluded that no adaptation is necessary. Further, 14.2% resulted in a minor computational effort for local tuning, and only 2.5% called for the building of a new model on the cloud. The new model was built after realizations $i = 80, 100, 146$, all of which mark $D_s = 10$ evaluations with no detected drift. This can be easily seen from Fig. 2, where L is either growing or stays at $L_{max} = 30$ for 10 consecutive realizations.
To compare communication reduction, we calculated the number of process realizations that were shared with Layer 3 when using the TriLS and building a new model approaches. Tuning $M_{in}(\vec{a}_{in})$ does not require any communication as the data stays at the device side. Table 5 summarizes the number of shared

Table 5: Communications Reduction with TriLS, Synthetic Set

Uploaded Data	**TriLS**	**Building new** $M(\vec{a})$	**Reduction % (samples)**
# Realizations	53	120	-
# Samples	53000	120000	55.8

realizations for both techniques, the total number of shared samples, and the overall percentage of communication overhead reduction calculated for samples only. With the TriLS, 55.8% less data was shared than with the building a new model approach. This is a large cut on redundant communication. At the same time, the TriLS achieved a better performance than other adaptive techniques, as was previously demonstrated.