Path Asymmetry Reconstruction via Deep Learning

Nada Alhashmi
EBTIC
Khalifa University
Abu Dhabi, UAE
nada.alhashmi@ku.ac.ae

Nawaf Almoosa
EBTIC
Khalifa University
Abu Dhabi, UAE
nawaf.almoosa@ku.ac.ae

Gabriele Gianini
EBTIC/Khalifa University, Abu Dhabi
Università degli Studi di Milano
Milano, Italy
gabriele.gianini@unimi.it

***Abstract*—This paper proposes a novel scheme to enhance the accuracy of packet-switched network synchronization systems by estimating path asymmetry (PA) using convolutional denoising autoencoders (CDAEs). Network synchronization is a key enabler of several emerging applications, with increasingly tight accuracy requirements especially for 5G. Path asymmetry, which arises due to physical and stochastic network conditions, severely degrades synchronization accuracy. In this paper, we propose a novel technique based on the IEEE Precision Time Protocol (PTP), which accurately reconstructs PA information from PTP packets. The proposed PA estimator can be integrated with existing synchronization systems as a pre-processing method to enhance the overall performance. Simulation results using industry-standard traffic profiles demonstrate significant improvements in PA estimation accuracy compared to the state of the art.**

***Index Terms*—Time Synchronization, IEEE 1588 Precision Time Protocol, PTP, Path Asymmetry, Machine Learning, Deep Learning**

I. Introduction

Accurate synchronization is a key requirement for numerous applications including industrial automation, smart grids, high frequency trading and Internet of things (IoT) [1]–[3]. Synchronization requirements are especially prevalent in 5G, which imposes stringent accuracy requirements up to 260ns to support features such as coordinated multi-point (CoMP), carrier aggregation (CA) and enhanced inter cell interface coordination [4], [5].

The IEEE Precision Time Protocol (PTP) enables scalable and cost-effective synchronization by exchanging timing information over packet-switched networks between few highly accurate clocks (master nodes) that are often Global Navigation Satellite System (GNSS) based, and multiple less accurate yet relatively inexpensive slave clock devices [1], [6]. At the slave side, accurate synchronization is achieved by periodically estimating and correcting unknown time-varying errors between the master and slave clocks that include the phase offset and the relative clock skew [7]. The estimation is based on the exchanged timing messages, which represent samples of the master and slave clocks corrupted predominantly by queuing delays incurred along the forward (master to slave) and reverse (slave to master) paths [8]. Joint model-based estimation of the phase offset, skew, and queuing delays was shown to be infeasible as it yields an under-determined system [7]. A relaxed estimation problem pursued by several works involves only the offset and skew as estimands by assuming symmetrical (equal), and thereby mutually canceling, forward and reverse path delays [9]–[11]. However, this assumption cannot be guaranteed in practice given the non-deterministic nature of the network traffic leading to significantly biased estimation and degraded synchronization accuracy when the forward and reverse paths are asymmetric (unequal) [12].

This paper proposes a novel path asymmetry (PA) estimation scheme for packet-switched networks without on-path timing support using denoising autoencoders (DAEs), which are trained to accurately reconstruct PA information at the PTP slave side from corrupted observations. The proposed scheme can be combined with model-based phase and skew estimators to improve the synchronization accuracy by compensating for the PA-induced bias inherent in model-based estimates. Moreover, it can be flexibly generalized across implementation scenarios given its data-driven nature. Using simulation and industry-standard workloads, we demonstrate that the proposed scheme achieves high estimation accuracy compared with existing approaches.

The reminder of the paper is structured as follows: section II gives an overview of the synchronization related work, followed by the synchronization mathematical model based on the PTP messages in section III. The proposed DAE based PA estimation technique for pre-processing is discussed in Section IV, followed by the analysis and the results in section V. Section VI is devoted to the conclusions of this work.

II. Related Work

Given its detrimental impact on synchronization accuracy [13], several works targeted PA estimation and mitigation [14]–[18]. [14] estimates path biases based on the assumption of a gamma-distributed queuing delays with *a-priori* known parameters. By contrast, our proposed solution is data-driven and not limited by distribution assumptions that might be violated in practice. [16] proposes an asymmetry mitigation technique for a PTP-aware network. PTP-aware networks have the ability to reduce the effect of the network queuing-induced packet delay variations (PDVs) using specialized nodes with timing support including boundary clocks (BCs) and transparent clocks (TCs) [19]. Although such enhancements increase the synchronization accuracy, PTP-aware networks incurs extra cost. Henceforth, our proposed solution is less restrictive and can be implemented on commodity ethernet networks without on-path timing support (PTP unaware networks).

A recursive path asymmetry estimator is proposed in [15] based on finite difference of exchanged timing messages. The technique can accurately estimate the inter-sample change

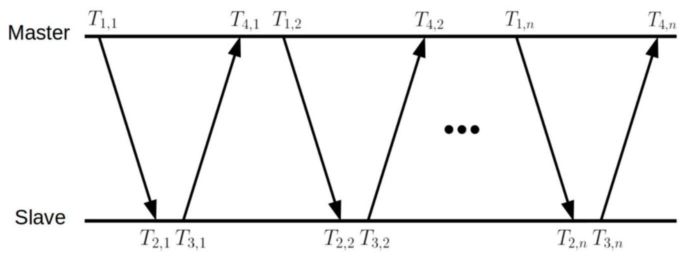


Fig. 1. PTP Messages Exchange Process

in PA. However, it is biased by the initial recursion PA value that is unknown and assumed zero. In [18], a joint maximum-likelihood estimator of phase and skew under path asymmetries caused by unknown deterministic propagation delays as well as random queuing delays was proposed, where a key component of the approach is to empirically estimate the queuing-delay distribution using Gaussian-Mixture Models (GMMs). Rather than tackling phase/offset estimation, our work can be considered as preprocessing technique that estimates and removes PA from timing measurements prior to the phase/skew estimation.

Estimating phase/skew in PTP-based synchronization was adapted by several works using Kalman Filtering [10], [20], likelihood estimation [21], [22], linear programming [9], [23], [24] and machine learning [25]. These works can be coupled with pre-processing techniques to reduce estimator input variance by filtering out PDVs [8], [26], [27] or by eliminating other noise and error factors such as temperature variations [28], and some of these techniques were analyzed by [13] for lightly loaded network scenarios. Our technique can be complementary to all of the above, where the estimated PA signal can update the extracted timing information from the PTP prior to further processing of the information.

III. System Model

Consider Fig. 1, which illustrates a simplified view of a PTP message exchange. The master initiates a message on the *forward* link, which contains a master's clock timestamp. Upon arrival, the slave clock generates a corresponding reception timestamp. We denote by $T_{1,n}$ and $T_{2,n}$ the forward-link master transmission and slave reception timestamps, respectively, where n is a non-negative message-exchange index.

Subsequently, the slave sends a response message that results in *reverse*-link timestamps corresponding to the slave transmission and master reception times, which are denoted as $T_{3,n}$ and $T_{4,n}$, respectively. Note that in practice, $T_{2,j} \approx T_{3,j}$, for any j, since the message processing delay is relatively negligible. The set of timestamps $\{T_{1,n}, T_{2,n}, T_{3,n}, T_{4,n}\}$ will be available at the slave side provided the message exchange is completed in the correct order.

The master-slave clock model is an affine relation parameterized by an intercept θ_0, which is a fixed and time-invariant *phase offset*, and a time-varying slope α (*skew*) representing the rate of change of the phase error vs time [7]. The relation between the forward path timestamps can be expressed as:

$$T_{1,n} = \theta_0 + (1+\alpha_n)T_{2,n} - d_{f,n} \ , \tag{1}$$

where $d_{f,n}$ denotes the delay experienced by the packet in the forward path. Similarly, we can model the relation between the reverse path timestamps as:

$$T_{4,n} = \theta_0 + (1+\alpha_n)T_{3,n} + d_{r,n} \ , \tag{2}$$

where $d_{r,n}$ is the reverse-path packet delay. Note that both $d_{f,n}$ and $d_{r,n}$ encompass a fixed propagation delay component, as well as random queuing delay components dependent on path traffic conditions. Adding (1) and (2), we obtain:

$$\begin{aligned}(T_{1,n} - T_{2,n}) + & (T_{4,n} - T_{3,n}) \\ &= 2\theta_0 + \alpha_n(T_{2,n} + T_{3,n}) + (d_{r,n} - d_{f,n}) \\ &= \theta_n + \Delta d_n,\end{aligned} \tag{3}$$

where θ_n represents the instantaneous value of the phase error growth due to the initial phase offset θ_0 and skew α_n, and the path asymmetry, which is given by:

$$\Delta d_n = d_{r,n} - d_{f,n} \ . \tag{4}$$

Note that (3) resembles the measurement equations used in [1], [9], [10], [20], which typically assume path symmetry ($\Delta d_n = 0$) and estimate θ_n using timestamp measurements (LHS of (3)). However, in practice path symmetry cannot be assured due to the random nature of queuing delays, leading to biased skew and offset estimates. For Kalman-Filtering based methods, it can be shown that the synchronization error contribution of path asymmetry is $\frac{\Delta d_n}{2}$. Based on experimentally generated data using a setup described in the sequel, Δd_n values range approximately between $-100,000$ and $89,000$ nanoseconds, with an arithmetic mean of $-10,751$ nanoseconds. These Δd_n values can introduce orders of magnitude of synchronization errors, which violates application specifications especially in mobile networks requiring sub-microsecond accuracy [4]. Estimating Δd_n and removing it from the measurement equation is therefore a key to maximize synchronization accuracy.

IV. Proposed PA-Estimation Solution

Machine learning has been used previously in communication networks [29], [30] and our proposed novel approach extend its use to cover network traffic based asymmetry reconstruction. The proposed solution is illustrated in Fig. 2. Rather than directly passing timestamp measurements expressed in (3) to the phase/skew estimation algorithm, we propose a preprocessing stage that produces a *cleaned* version through the estimation and removal of Δd_n, which is carried out using a Machine Learning approach that involves (1) the detection of the occurrence of a known PA pattern, followed by (2) the reconstruction of the PA pattern from noisy observations using Convolutional Denoising Autoencoders (CDAEs), where the scope of this paper will focus on the CDAE-based reconstruction part.

Denote by $p \in R^N$ a known time-series asymmetry pattern with values $p = \Delta d_j \ : \ j \in [k, k+N-1]$. We assume in this work that known patterns are found using offline profiling of network traffic data, and can be detected in realtime using an

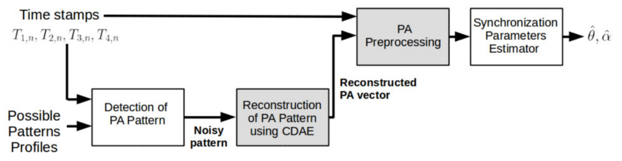

Time stamps
$T_{1,n}, T_{2,n}, T_{3,n}, T_{4,n}$
Possible Patterns Profiles
Detection of PA Pattern
Noisy pattern
Reconstruction of PA Pattern using CDAE
Reconstructed PA vector
PA Preprocessing
Synchronization Parameters Estimator
$\hat{\theta}, \hat{\alpha}$

Fig. 2. Proposed full solution

appropriate machine learning algorithm. When it occurs, the slave can only observe p through timestamp measurements that include the path asymmetry as well as instantaneous phase values as expressed in (3). Hence, we define the *corrupted pattern* observation as $\tilde{p}$, where

$$\tilde{p} = \Delta d_j + \theta_j \ : \ j \in [k, k+N-1] \tag{5}$$

Using DAEs, we seek to reconstruct the original path asymmetry vector p by *denoising* $\tilde{p}$; namely removing the instantaneous phase θ_j components.

A. Denoising Autoencoder for PA Reconstruction

Denoising Autoencoders (DAEs) are variants of autoencoders that are trained with noisy versions of the original data, and the task of the DAE is to learn a function for recovering the original signal from noisy observations [31], [32]. DAEs can be thought of as consisting of two functions: $\psi()$, which encodes the input by mapping to a lower dimensionality space, and $\chi()$, which decodes or reconstructs the output of the encoding function $\psi()$. The DAE encoding and decoding processes can be represented as follows:

$$\psi(\tilde{p}) = \sigma(w * \tilde{p} + b) \tag{6}$$

$$p^* = \chi(\psi(\tilde{p})) = \sigma(w * \psi(\tilde{p}) + b) \tag{7}$$

where $wx + b$ represents the affine mapping performed by each of network nodes with a weight of w and a bias of b, $*$ represents the convolution operation and $\sigma()$ refers to the nonlinear (activation) operation.

By feeding the corrupted asymmetry pattern $\tilde{p}$ to the DAE, it aims to learn a representation that reconstructs the original pattern p by undoing the corruption effects. If we denote by $\mathcal{L}$ a loss function penalizing input difference [31], the aim of the DAE training procedure is to minimize the objective function:

$$\mathcal{L}(p, \chi(\psi(\tilde{p}))) \tag{8}$$

As a loss function one can use the Mean Absolute Percentage Error (MAPE) and its definition will be recalled in the next Section. In the training process, the data set is fed to the DAE and the error between the original p and reconstructed pattern p^* is iteratively minimized using back propagation until convergence to an acceptable reconstruction error [31], [32].

For this work, a convolutional DAE has been selected following the architecture shown in Fig. 3 for a vector length of 64. The rectified linear unit (ReLU) in the figure represents the activation function used in the layers, where mathematically its output y can be defined as function of its input x as following: $y = max\{0, x\}$

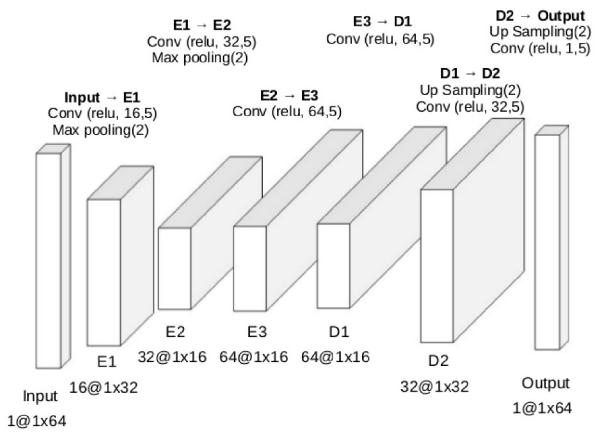


Fig. 3. Architecture of the Convolutional DAE (input vector = 64)

B. Path Asymmetry Preprocessing

At the output of the DAE, we obtain a DAE-reconstructed PA vector represented as:

$$p^* = [\Delta d_1^* \ \ \Delta d_2^* \ \ \cdots \ \ \Delta d_N^*]^\top \tag{9}$$

The aperiodicity of DAE-based estimates prohibits their direct utilization in a uniform-sampling rate systems such as offset and skew estimation algorithms. Accordingly, to leverage p^* in PA preprocessing, we propose a hybrid approach by combining it with the recursive PA estimator in [15].

The core of the recursive PA algorithm in [15] is finding the integral of the finite queuing differences for each individual path. The finite queuing difference between any two consecutive messages in the forward direction δd_f and reverse direction δd_r can be found as following:

$$\delta d_{f,n} = (1 + \alpha_n)\delta T_{2,n} - \delta T_{1,n} \tag{10}$$

$$\delta d_{r,n} = \delta T_{4,n} - (1 + \alpha_n)\delta T_{3,n}, \tag{11}$$

where $\delta T_{m,k} = T_{m,k} - T_{m,k-1}$. The previous equations lead to a delay estimation $\hat{d}$ for any sample n, represented as:

$$\hat{d}_{f,n} = \hat{d}_{f,n-1} + \delta d_{f,n} \tag{12}$$

$$\hat{d}_{r,n} = \hat{d}_{r,n-1} + \delta d_{r,n}. \tag{13}$$

The recursive estimator output $\Delta\hat{d} = \hat{d}_{r,n} - \hat{d}_{f,n}$ is subtracted from the measurement equation (3), under the assumption that the start of the recursion $\Delta\hat{d}_0 = 0$. This yields to a biased estimation by the actual value of Δd_0. Arbitrary initialization of the recursion amounts to random sampling of the asymmetry distribution, which can yield significant bias. Thus, through the proposed hybrid approach, DAE-based estimation can provide accurate initial conditions for the recursion, namely Δd_0^*, resulting in an overall enhanced synchronization performance.

V. System Evaluation and Results

To validate the suggested approach, two test cases defined by the International Telecommunication Union (ITU), Recommendation G.8261, of traffic profiles that standardize IEEE PTP device testing from a traffic load perspective [33] were used. The test cases can be summarized as follows:

- *tc13*: represents a network with sudden and big changes in the load conditions. The forward path of tc13 starts with a background cross-traffic of 80% for 1 hour, then fluctuates between 20% and 80% with equal period of 1 hour each. The reverse path for the same test case background traffic starts at 50% for 1.5 hours, then fluctuates between 10% and 50% with equal period of 1 hour each except for the last segment to assure equal length with the forward path.
- *tc14*: represents a network with a large slow-varying load, where both the forward and reverse paths vary over a 24-hours period, however, the background cross-traffic is on different ranges where the forward direction changes with slow rate from 20% to 80% then back, and the reverse changes from 10% to 55% then back.

The network delay profile associated with each test case was generated experimentally by injecting the traffic into a network of 10 switches connected based on the ITU-T G.8261 setup, and capturing a trace of inter-packet arrival times experienced in both the forward and the reverse paths.

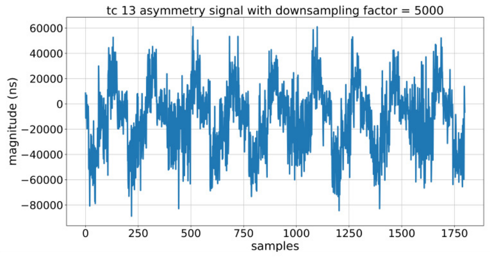
A. DAE Simulation and Training

To simulate and train the designed CDAE, we utilize tools based on standard open source libraries: *Keras* [34], which is built on *Tensorflow* backend [35], and *sklearn* [36]. As highlighted is section IV, the proposed technique is based on denoising known/reference asymmetry patterns. In this work, we arbitrarily select 30 reference patterns from each of tc13 and tc14 time series since the focus is evaluating the denoising CDAE performance rather than reference-pattern discrimination. Each reference pattern is a vector of length 64 following downsampling by a factor of 5000. Fig. 4 illustrates reference pattern examples from tc13 and tc14, respectively. To evaluate the denoising performance of the CDAE, we trained two independent CDAE models one for each test case and with 30 different reference patterns each. The selected patterns were corrupted with a fixed initial offset θ_0 and skew α values from a uniform distributions observed based on experiments, where α is assumed to be constant during the capturing window. The sampling rate of both patterns is 128 messages per second, which corresponds to one of the standard packet rates in the PTP synchronization protocol. Examples of training samples from tc13 and tc14 are shown in Fig. 5.

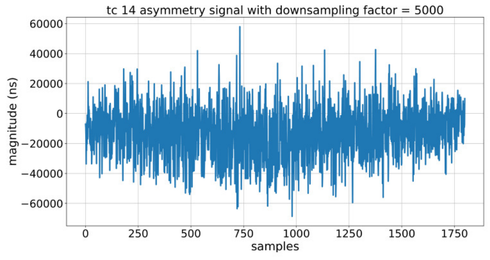
All the selected parameters for the training phase of the autoencoder are listed in table I. The loss function $\mathcal{L}$ is set as the mean absolute percentage error function defined as :

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{(y_i - \hat{y}_i)}{y_i} \right| \tag{14}$$

In addition, the created data set was divided between training and validation with a ratio of (9:1). The associated number of samples with each of these sets is presented in the table below.

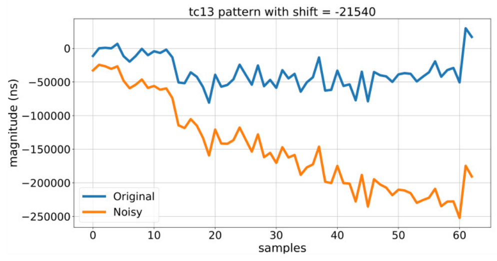


a. tc13

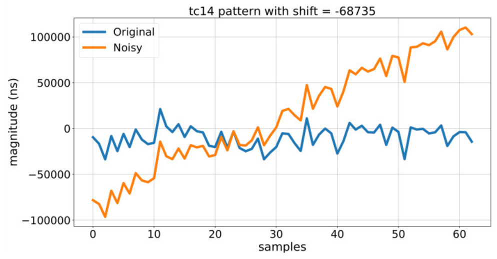


b. tc14

Fig. 4. Training Reference Patterns Examples



b. tc13 pattern with negative offset and positive skew



b. tc14 pattern with negative offset and negative skew

Fig. 5. Examples of Training Samples from tc13 and tc14

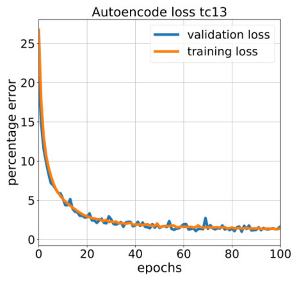
Fig. 6 illustrates the training and validation loss as a function of the number of epochs, focusing on 100 epochs, where it is clear that the convergence is achieved.

B. Asymmetry Estimation Results

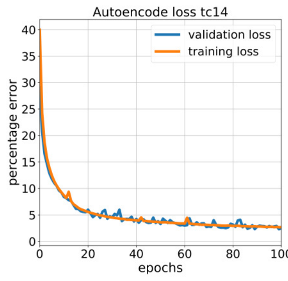
To illustrate the improved estimation performance of the proposed machine learning approach, we integrate it with the recursive estimator discussed in [15] as described in section IV then compare the error estimation of the proposed hybrid approach with the recursive estimator as a baseline. The integrated machine learning approach with the recursive estimator is referred as the proposed hybrid ML-recursive

TABLE I
TRAINING PHASE PARAMETERS

Parameter	Selected Value
Number of epochs	500
Total number of samples	600000
Training samples	540000
Validation samples	60000
Batch size	512
Range of training skew	Uniform Distribution $[-6e-8, 6e-8]$
Range of training offset	Uniform Distribution $[-50000, 50000]$
Loss Function	Mean Absolute Percentage Error
Optimizer	ADAM



a. tc13 pattern autoencoder



b. tc14 pattern autoencoder

Fig. 6. Loss Function for the DAE Model, x-axis Range of (0-100 epochs)

estimator. The error metric we use is the percentage of the absolute normalized error e defined as:

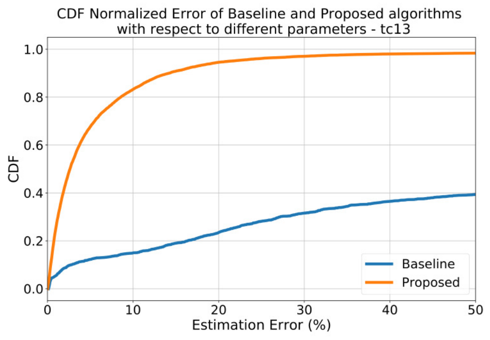
$$e = 100\% \left| \frac{(\Delta d_i - \Delta \hat{d}_i)}{\Delta d_i} \right|. \tag{15}$$

For the baseline recursive estimator, we evaluate the performance using a range of initial recursion (bias) conditions based on the selected pattern. For the proposed hybrid ML-recursive estimator, we evaluate the performance using a range of initial bias conditions drawn from the reconstructed pattern, as well as a range of the phase shift values θ_0 and skew α that are outside of the training set. Fig. 7 plots the empirical cumulative distribution function (CDF) of the normalized error of the baseline and the proposed hybrid ML-recursive estimator respectively for both tc13 and tc14, where each test include 4 different combinations of θ_0 and α values. It is apparent that, for both test cases, the performance gets significantly improved by the ML approach, since 75% of the error distribution is concentrated in 30% error. For the same error percentage, the baseline algorithm managed to estimate less than 40% of the data. From these results it is also shown that tc13 results were better than tc14, this may depend on the pattern characteristics, however, the impact of different characteristics on the learning process is outside the scope of this paper. The results in Fig. 7 has confirmed that when compared with the baseline algorithm, the proposed CDAE managed to enhance the PA estimation accuracy.

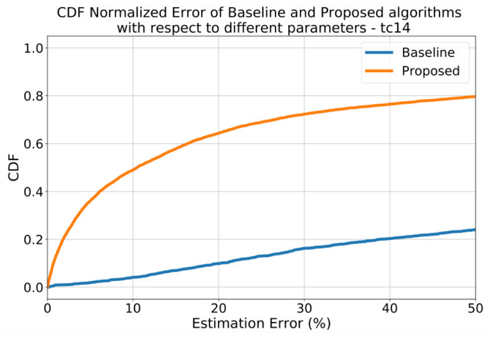
C. Generalization of the Proposed Technique

To examine the generalization of the proposed solution, the tc13-trained CDAE was tested for new patterns that are outside the training set, corrupted with offset θ_0 and skew α that were also not covered during the training. The CDF associated with 3 different new patterns is shown in Fig. 8 .

The results in Fig. 8 demonstrate the effectiveness of the method and its ability to reconstruct different signals using the same computational power. This supports the conclusion that the proposed solution is applicable in general and that the DAE can be trained and implemented for different observed patterns from the network (rather than implementing a dedicated autoencoder for each of the possible patterns which might not be practical).



a. tc13



b. tc14

Fig. 7. CDF of the Baseline and Proposed Algorithms of an Absolute Normalized Error for Both Systems with a x-axis Range of $(0 - 50\%)$

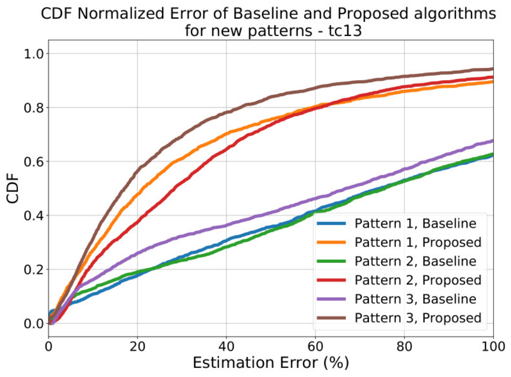


Fig. 8. CDF of tc13 DAE for New Patterns Corrupted by Different Values (Pattern1: $\theta = 1.2E+5$, $\alpha = 7E-08$, Pattern2: $\theta = -1.3E+5$, $\alpha = 7E-08$,Pattern3: $\theta = 1.5E+5$, $\alpha = -9E-08$) with x-axis Range of (0-100%)