Robust Time Series Recovery and Classification Using Test-time Noise Simulator Networks

Jeon, Eun Som; Lohit, Suhas; Anirudh, Rushil; Turaga, Pavan

TR2023-021 April 19, 2023

Abstract

Time-series are commonly susceptible to various types of corruption due to sensor-level changes and defects which can result in missing samples, sensor and quantization noise, unknown calibration, unknown phase shifts etc. These corruptions cannot be easily corrected as the noise model may be unknown at the time of deployment. This also results in the inability to employ pre-trained classifiers, trained on (clean) source data. In this paper, we present a general frame- work and models for time-series that can make use of (unlabeled) test samples to estimate the noise model—entirely at test time. To this end, we use a coupled decoder model and an additional neural network which acts as a learned noise model simulator. We show that the framework is able to "clean" the data so as to match the source training data statistics and the cleaned data can be directly used with a pre-trained classifier for robust predictions. We perform empirical studies on diverse application domains with different types of sensors, clearly demonstrating the effectiveness and generality of this method.

IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2023

ROBUST TIME SERIES RECOVERY AND CLASSIFICATION USING TEST-TIME NOISE SIMULATOR NETWORKS

Eun Som Jeon[⋆] *Suhas Lohit*[†] *Rushil Anirudh*[‡] *Pavan Turaga*[⋆]

[⋆] Geometric Media Lab, Arizona State University, Tempe, AZ, USA
[†]Mitsubishi Electric Research Laboratories, Cambridge, MA, USA
[‡]Lawrence Livermore National Laboratory, Livermore, CA, USA

ABSTRACT

Time-series are commonly susceptible to various types of corruption due to sensor-level changes and defects which can result in missing samples, sensor and quantization noise, unknown calibration, unknown phase shifts etc. These corruptions cannot be easily corrected as the noise model may be unknown at the time of deployment. This also results in the inability to employ pre-trained classifiers, trained on (clean) source data. In this paper, we present a general framework and models for time-series that can make use of (unlabeled) test samples to estimate the noise model—entirely at test time. To this end, we use a coupled decoder model and an additional neural network which acts as a learned noise model simulator. We show that the framework is able to "clean" the data so as to match the source training data statistics and the cleaned data can be directly used with a pre-trained classifier for robust predictions. We perform empirical studies on diverse application domains with different types of sensors, clearly demonstrating the effectiveness and generality of this method.

***Index Terms*—** Signal recovery, DNN, time series

1. INTRODUCTION

A common obstacle in deploying machine learning models in a practical setting, usually trained in ideal conditions, is that the sensor data may exhibit a domain shift from those employed during training-time. This can arise due to differing sensor configurations, such as sampling rate and calibration, defects leading to missing data, measurement noise, unknown phase shifts etc. Due to its unpredictable nature, the noise model for any particular setting is usually unknown. When dealing with 2D images, this has been studied under the umbrella of robustness, with several effective techniques that use augmentations [1, 2], adversarial training [3, 4], and other specialized strategies that expose the machine learning model to a large class of potential variations in the data during train time, so that they generalize well at test time, even when the data shift is unknown. Unfortunately, many heuristics like augmentations, and ℓ_p-norm perturbations developed in images do not work as well when dealing with time varying data – especially when dealing with noise models specific to time series such as phase shifts. Moreover, in some applications like healthcare, it can be important to recover the original signal instead of simply building invariance into a discriminative model, so that a person can visualize/use the recovered signal for decision-making.

This work of E. Jeon and P. Turaga was supported by NIH under Grant R01GM135927, as part of the Joint DMS/NIGMS Initiative to Support Research at the Interface of the Biological and Mathematical Sciences.

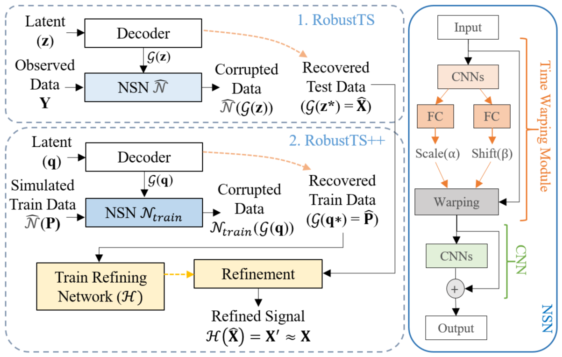

1. RobustTS
Latent (z)
Decoder
$\mathcal{G}(\mathbf{z})$
Observed Data Y
NSN $\widehat{\mathcal{N}}$
Corrupted Data $\widehat{\mathcal{N}}(\mathcal{G}(\mathbf{z}))$
Recovered Test Data $(\mathcal{G}(\mathbf{z}^*) = \widehat{\mathbf{X}})$
2. RobustTS++
Latent (q)
Decoder
$\mathcal{G}(\mathbf{q})$
Simulated Train Data $\widehat{\mathcal{N}}(\mathbf{P})$
NSN $\mathcal{N}_{train}$
Corrupted Data $\mathcal{N}_{train}(\mathcal{G}(\mathbf{q}))$
Recovered Train Data $(\mathcal{G}(\mathbf{q}*) = \widehat{\mathbf{P}})$
Train Refining Network ($\mathcal{H}$)
Refinement
Refined Signal $\mathcal{H}(\widehat{\mathbf{X}}) = \mathbf{X}' \approx \mathbf{X}$
Input
CNNs
FC
FC
Scale(α)
Shift(β)
Warping
Time Warping Module
CNNs
CNN
Output
NSN

Fig. 1: Overview of ROBUSTTS and ROBUSTTS++ and the Noise Simulator Network (NSN) architecture.

In this paper, we address the problem of robustness and recovery of time-series to several unknown test-time corruptions. We approach it using a *decoder model* trained on clean or ideal unlabeled training data to capture the entire space of possible signals. To account for the unknown noise, we use a shallower network called a Noise Simulator Network (NSN) which simulates the noise model relative to the training data used to train the decoder model. With this strategy, called ROBUSTTS , we can "clean" the data at test time by sampling the decoder model in such a way that when the simulated noise is applied to the generated data, it matches the observed samples. An important benefit, in addition to high fidelity signal recovery, is that the final output from the decoder model can be used in lieu of the measured data as input to a classifier trained on clean data. We design a further refinement step, ROBUSTTS++ , where the training and test sets are brought closer using the learned NSN from ROBUSTTS , and leads to further improved performance.

In multiple application domains and sensors, we find that ROBUSTTS and ROBUSTTS++ consistently and significantly improve performance over several baselines both from the denoising perspective (denoise and then classify) and robustness perspective (make the classifier robust to distribution shifts). By simulating commonly occurring noise models, we present extensive empirical analysis clearly demonstrating the advantages of the proposed method.

2. RELATED WORK

Unsupervised learning and decoder modeling for time series: The algorithms proposed in this paper rely on a pretrained decoder model to generate the time series and we optimize the latent space of a decoder model. Within the realm of neural network-based methods, different types of decoder models have been designed. Conceptually, the simplest one is the autoencoder (AE) trained to minimize the reconstruction loss, however, it cannot sample new data easily.

This drawback can be overcome with models such as GANs [5, 6], Variational AEs (VAE) [7] and Wasserstein AEs (WAE) [8], which allow for sampling the latent space easily. WAEs are the most stable.

Denoising, recovery and imputation methods for time series: Several algorithms exist for denoising and imputation of time series problems, including linear filtering techniques [9] and dictionary learning and sparse coding [10], which are consistently outperformed by deep learning techniques for large data [11–13]. In general, noise models are assumed to be known or are limited to a single type, which is not universal in nature. Noise learning-based denoising autoencoder [14] is not applicable if the noise at test time is unknown. Our work is inspired by a recent method [15], which was specifically designed to account for domain shift in the test 2D images. Instead, we develop a framework that is meant for time series and propose a further refinement algorithm which is crucial in improving performance.

Making deep classifiers more robust: As the eventual goal of denoising and imputation is high-level inference e.g. classification and anomaly detection, we can try to directly train classifiers and anomaly detectors to be more robust to test-time corruptions. Methods like domain adaptation [16] are related to our approach in that they employ unlabeled data from the test domain. However, they typically rely on learning invariant representations between source and target, which may not always be possible if the target domain is corrupted by several different random noise models, particularly in a few shot setting with only a few tens of samples available at test time. Furthermore, in most of augmentations and auxiliary tasks [2] are image-domain specific (like affine transforms, cutouts, etc.), and its unclear how they generalize to the time domain.

3. ALGORITHM AND ARCHITECTURE

Our main goal is to achieve robustness in recovering and classifying time series under unknown noise during test-time. An overview our approach is depicted in Fig 1. At its core, ROBUSTTS utilizes a decoder model trained on clean data, to jointly estimate the unknown noise using a NSN, and the latent space parameters corresponding to the cleaned test data.

Problem setup and notation: We first train a decoder model on a clean training set $\mathbf{P}$ and use the trained decoder model $\mathcal{G} : \mathbb{R}^d \mapsto \mathbb{M}$, where $\mathbb{M}$ represents the manifold of clean time series and d is the dimensionality of the latent space. Consider a multi-variate time series signal $X \in \mathbb{M}$. X, clean test data, can be represented as a matrix of dimensions $c \times t$, where c and t are the number of channels and frames/samples, respectively. Let us represent the (unknown) corruption process as $Y = \mathcal{N}(X)$, where $\mathcal{N}(\cdot)$ is a noise function and Y, the observed signal at the time of deployment. When $\mathcal{N}$ is known, the problem becomes much simpler and the noisy signal Y can be cleaned. When $\mathcal{N}$ is unknown, we perform cleaning by projecting Y onto $\mathbb{M}$ and simultaneously approximating $\mathcal{N}$. Next, let the observed test-time sequences be $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_n\}$, where n is the number of sequences and $Y_i = \mathcal{N}(X_i)$, and $\mathbf{X} = \{X_1, X_2 \ldots, X_n\}$.

3.1. Joint signal recovery and noise simulator training

Let us denote by $\widehat{\mathcal{N}}$ the neural network, parameterized by Θ used to approximate the corruption process $\mathcal{N}$. Given noisy data Y_i's and the decoder $\mathcal{G}$, we want to find $\widehat{\mathcal{N}}$ and $\hat{\mathbf{X}}$ such that $\widehat{\mathcal{N}}(\hat{\mathbf{X}}) \approx Y$ and $\hat{\mathbf{X}} \approx \mathbf{X}$. Here $\hat{\mathbf{X}} = \mathcal{G}(\mathbf{z}^*)$, constrained by the sigmoid function to set values between 0 and 1, obtained after optimizing over the parameters Θ^*, and latent variables $\mathbf{z}$ which are the inputs to the decoder model. To this end, our goal is to optimize the following objective:

$$\Theta^*, \{\mathbf{z}_j^*\}_{j=1}^n = \underset{\Theta, \{\mathbf{z}_j\}_{j=1}^n}{\operatorname{argmin}} \sum_{j=1}^{n} \mathcal{L}(Y_j, \widehat{\mathcal{N}}(\mathcal{G}(\mathbf{z}_j), \Theta)) \qquad (1)$$

As both $\mathcal{G}$ and $\widehat{\mathcal{N}}$ are differentiable, the gradients of the objective in equation (1) can be evaluated with backpropagation and existing gradient-based optimizers. In this paper, we solve this optimization problem approximately using an alternating optimization framework which converges to a minimum fairly quickly in all the applications considered here. We outline the alternating optimization steps for the $(k+1)^{th}$ iteration below.

1. Fix $\widehat{\mathcal{N}}$ and optimize for $\mathbf{z}^*$ using a projected gradient descent (PGD) method similar to [17]. For each $j = 1, \ldots, n$,
$$\mathbf{z}_j^{(k+1)} = \underset{\mathbf{z}_j}{\operatorname{argmin}} \|Y_j - \widehat{\mathcal{N}}(\mathcal{G}(\mathbf{z}_j), \Theta^{(k)})\|_2^2 \qquad (2)$$
2. Fix $\{\mathbf{z}_j^*\}_{j=1}^n$ and optimize for Θ using the mean squared error as the loss function and stochastic/mini-batch gradient-based optimizers:
$$\Theta^{(k+1)} = \underset{\Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{j=1}^{n} \|Y_j - \widehat{\mathcal{N}}(\mathcal{G}(\mathbf{z}_j^{(k+1)}), \Theta)\|_2^2 \qquad (3)$$

It is important to note that the inner loop optimization problems (2) and (3) are solved (approximately) using gradient-based descent methods run for η and λ iterations respectively. We will denote by κ, the number of outer loop iterations, i.e. $k = 1, 2, \ldots, \kappa$. Once we reach convergence, the output of this algorithm is the cleaned version of the $\mathbf{Y}$ given by $\hat{X}_j = \mathcal{G}(\mathbf{z}_j^*), \forall j$ in the test set. We can also envision periodically using this strategy to update $\widehat{\mathcal{N}}$ to account for changes in the noise model during test time. Assuming that the pretrained decoder model is good, if the corruption function is well estimated by $\widehat{\mathcal{N}}$, then $\hat{X}_j \approx X_j$. However, if the unknown noise process is very complicated and there are no common patterns in the observed samples, it can be hard to estimate $\mathcal{N}$ accurately.

3.2. Noise simulator network (NSN) architecture

The NSN plays a crucial role in obtaining robustness to unknown noise models within ROBUSTTS . We design a NSN to deal with commonly occurring time series errors such as missing data, spikes, noise injection, and unknown time scaling and phase shifts. The network includes two parts – a time warping module (affine transform) and a 1D CNN, which is shown in Fig. 1. To solve for unknown phase shifts and linear time scaling, we adopt smaller network to first predict these parameters which are applied to input via linear interpolation, which is a differentiable operation [18]. The effective warping function denoted by $\gamma(\tau) = \alpha\tau + \beta$, $\tau \in \{1, 2, \ldots, t\}$, α is the parameter for time scaling, and β is for time shifting. The time warping module (TWM) is used to undo the effect of unknown time scaling and shifting as the CNN layers alone may not be sufficient. Importantly, we find that the NSN provides robustness to the *span* of its constituent functions (i.e., combinations of different functions) resulting in a significantly larger space of recoverable noise models. Moreover, as we demonstrate, the NSN is also effective in cleaning samples that are all independently corrupted in a batch with different noise models.

3.3. Further refinement

To further refine the outputs $\mathbf{X}$ obtained from the ROBUSTTS stage, we propose an additional refinement stage, ROBUSTTS++ , by *revisiting the training set* $\mathbf{P}$. Having estimated the unknown noise model, $\widehat{\mathcal{N}}$, with ROBUSTTS on the observed test data, we use it to corrupt a subset of the clean training data, $\mathbf{P_s} \subset \mathbf{P}$, by computing $\widehat{\mathcal{N}}(P_{s_i}), \forall P_{s_i} \in \mathbf{P_s}$. Next, we repeat the noise estimation process using $\mathbf{P_s}$ and train a new NSN, $\widehat{\mathcal{N}}_{train}$ in the same fashion as in ROBUSTTS . This allows us to create a "cleaned" version of the entire corrupted training data, denoted by $\widehat{\mathbf{P}}$. The advantage of this process is that it gives us correspondence between the estimated clean data

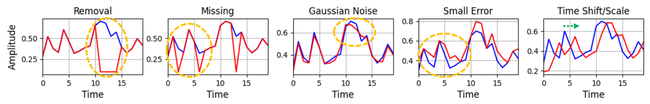


Fig. 2: Corruption functions studied in this paper. We also consider combinations of these corruptions in addition to simulating calibration errors with a random linear function.

and the true clean data. Consequently, we train a neural network denoiser $\mathcal{H}$ with the same architecture and training protocol as $\mathcal{N}$. $\mathcal{H}$ is parameterized by Θ' to map $\widehat{\mathbf{P}}$ to $\mathbf{P}$. $\mathcal{H}$ is an autoencoder with 1D convolutional layers. It is trained by minimizing the usual mean squared error $L(\Theta') = \sum_{j=1}^{m} \|\mathcal{H}(\widehat{P}_j, \Theta') - P_j\|_2^2$. Finally, using the trained $\mathcal{H}$, we can obtain the refined $\mathbf{X}' = \mathcal{H}(\widehat{\mathbf{X}})$ that is closer to $\mathbf{X}$, compared to $\widehat{\mathbf{X}}$. We will substantiate this claim empirically.

4. EXPERIMENTAL RESULTS

In this section, we conduct experiments on a subset of the publicly available PAMAP2 [19] and HDM05 [20] datasets to verify the effectiveness of the proposed method. For PAMAP2, the length of a sequences is 100 with 40 channels, recorded from the heart rate and IMUs for 9 subjects. We use only subjects in this dataset, about 3.5K sequences each, for which examples from all 12 activity classes are available and allow for measuring the effect of the proposed method on downstream classification better. We use ResNet-18 as a classifier, whose the number of trainable parameters is approximately 248K. The clean test accuracy using leave-one-subject-out evaluation on subject 1, 4, and 6 were 75.17%, 96.17% and 89.65%, respectively. For the dataset, we use $\kappa = 25$, and $n = 512$ corrupted/noisy test sequences for training the NSN. HDM05 is a challenging MoCap dataset of 3D human actions with 2337 samples of 5 subjects for 130 classes, consisting of 100 samples for a sequence with 31 3D joints – 93 channels. We perform 5-fold cross validation and report averaged results. The accuracy of the model for the clean test data was 78.00%. We use $\kappa = 15$ and $n = 512$ for the dataset.

Table 1: Classification accuracy (%) (PSNR in dB) with various corruption functions on test subject 1 of PAMAP2 with AE decoder. (α, β) are (1.1, 7.0). Note, § denotes without TWM.

Method	Noise / corruption type				
	Removal	Missing	Removal + Gaussian Noise	Missing + Small Error	Time Scale and Shift
No denoising	5.39 (11.71)	5.44 (11.72)	5.39 (11.69)	5.36 (12.35)	5.33 (10.91)
RobustTS §	58.84±1.14 (26.85±0.09)	60.00±0.51 (27.11±0.18)	58.44±1.38 (26.80±0.12)	59.17±0.55 (26.92±0.07)	51.58±0.25 (21.73±0.02)
RobustTS	60.11±0.13 (26.86±0.03)	61.20±0.22 (27.31±0.03)	59.82±0.18 (26.86±0.02)	59.90±0.30 (26.97±0.06)	58.54±0.31 (26.93±0.06)

4.1. Noise/corruption types for experiments

In this paper, we consider various corruption errors commonly encountered in time series [21–24], including continuous/discrete missing, Gaussian noise, small error, time axis error, calibration error, and multiple noise error, as illustrated in Fig. 2. As a more challenging setting ("Random"), we randomly choose and apply a single noise function for each test sequence and then apply a random "Time Scale/Shift". **The noisy sequences thus generated are henceforth treated as the test set for all the algorithms**. We use the following values of noise parameters in all our experiments unless otherwise stated. For "Removal" and "Missing" corruptions on PAMAP2, 15% of the samples in a sequence are used. For HDM05, 20% of the samples in a sequence are used for both corruption functions. Standard deviation of "Gaussian noise" is 0.2 and the offset value for "Small Error" is 0.2 for 50% of the frames in the sequence. For applying time scale/shift are chosen by uniformly sampling from $0.8 \leq \alpha \leq 1.2$ and $-15 \leq \beta \leq 15$. Finally, we apply a linear function to model unknown calibration using $aY(t) + b$, and we fix them at arbitrarily chosen small values of $a = 0.22$ and $b = 0.19$.

Baseline methods: We measure our frameworks against both (a) signal recovery methods i.e. **denoise-then-classify** where we first denoise the signal and pass it through a classifier trained only on clean data and (b) **robust classifiers**. (a) We denote **Noisy input** which is directly feeding the noisy test signals to the classifier trained on clean data. **DAE [25], DAE+AT [26], TCDAE [13]** are used as baslines with an assumption that **noise types that occur at test time are known beforehand, even though the exact noise parameters are not**. We sample corresponding noise parameters randomly from an interval of 0.1 to 4 times the true noise parameters used to generate observed data. We employ a latent space optimization (**LSO**) framework for test-time adaptation, referring to recent works like [17]. (b) With the assumption that the **exact noise types that occur at test time are unknown**, we trained classifiers with the different number of corruption functions that we use for generating observed data. Besides the **Naïve augmentation (Aug)**, **Mixup [1]** with 0.25 and **Adversarial training (Adv)** [3] with PGD [4] and multiple perturbations (mPGD) [27] for attacks are implemented as baselines. The attack parameters for number of steps, step size, and ϵ are set 7, 0.1, and 0.3, respectively. Also, we train **ChoiceNet** [28] on clean data and evaluate with corrupted data.

4.2. Implementation details and results

The total number of iterations for training NSN η and PGD optimization λ are 20 and 25. We compare the proposed methods with the baselines in terms of peak signal-to-noise ratio (PSNR) between the denoised/recovered time sequences and the ground-truth sequences as well as classification accuracy using classifiers trained on clean training data. We report the results with mean and std. dev over 5 random initializations. Note, § denotes without TWM. For augmented classifier, one noise type is selected and averaged accuracy is reported. For PAMAP2 and HDM05, the sizes of latent space of encoder-decoder network are 64 and 200, respectively. The networks are trained with a batch size of 128 and 64, respectively. We use Adam optimizer for training the models. The total number of epochs is set to 200 for all datasets with the learning rate of 10^{-4}.

Results on PAMAP2: In Table 1 and 2, classification accuracy and PSNR under different unknown noise settings for RobustTS are described. We observe that RobustTS with TWM yields the best results, prominently shown with time scale and shift corruption. TWM in the NSN helps further improves performance. It is easy to see that RobustTS++ yields the best results, followed by RobustTS where the NSN employs the TWM. Additionally, as shown in 3, RobustTS with TWM achieves the best performance with a decoder part of a WAE in accuracy and AE in PSNR. Results with VAE generate the lowest in both accuracy and PSNR.

Results on HDM05: As described in Table 4, RobustTS++ achieves, over 41%, the best results compared to all the baselines including all the robust classifiers. Supervised denoiser-based methods including DAE [25] and TCDAE [13] performed good signal recovery, about 26 dB in PSNR. However, their classification results are poor, about 27% at best. As shown in Fig. 3, RobustTS++ shows better performance in all examples.

4.3. Ablation study

Number of test sequences for training NSN: Additionally, since we rely on the observed test samples to estimate the unknown noise in the data, we explore the effects on the numbers of test sequence.

Table 2: Classification accuracy (%) (PSNR in dB) for "Random" corruption on PAMAP2 with the decoder part of an AE for ROBUSTTS . * results for DAE and Adv.+mPGD outperform DAE+AT and Adv+PGD, respectively.

Subject	LSO	DAE*	Mixup	Aug.	Aug. + mixup	Adv. + mPGD*	ROBUSTTS (w/o TWM)	ROBUSTTS (w/ TWM)	ROBUSTTS++
1	8.68±0.08 (20.66±0.001)	44.04±0.20 (25.37±0.04)	10.36 ±0.40	22.32 ±3.48	23.91 ±3.80	17.77 ±0.64	52.03±0.24 (25.00±0.03)	52.62±0.23 (25.02±0.01)	**60.40±0.19** (25.44±0.02)
4	11.53±0.17 (22.83±0.002)	73.05±0.45 (28.72±0.06)	12.38 ±1.07	25.37 ±4.62	27.03 ±4.66	18.47 ±0.72	70.50±0.17 (21.77±0.01)	71.42±0.28 (21.83±0.01)	**76.94±0.23** (27.31±0.06)
6	9.74±0.16 (22.24±0.002)	66.27±0.83 (27.60±0.004)	14.83 ±0.89	28.30 ±4.17	26.66 ±4.52	24.32 ±0.49	70.95±0.06 (25.27±0.04)	71.17±0.16 (25.31±0.02)	**73.28±0.44** (25.97±0.05)

Table 3: Classification accuracy (%) (PSNR in dB) with "Time scale/shift" and "Random" corruption functions on test subject 1 of PAMAP2.

Method	Time Scale/Shift			Random		
	WAE	VAE	AE	WAE	VAE	AE
LSO	7.59±0.29 (20.61±0.03)	8.17±0.07 (21.67±0.003)	8.55±0.16 (20.66±0.001)	7.76±0.15 (20.64±0.001)	8.16±0.05 (21.71±0.002)	8.68±0.08 (20.66±0.001)
DAE	50.51±0.92 (25.31±0.03)	9.62±0.05 (21.44±0.001)	62.05±0.58 (27.55±0.06)	38.90±0.58 (24.72±0.05)	9.60±0.06 (21.44±0.002)	44.04±0.20 (25.37±0.04)
DAE+AT	50.57±0.60 (25.34±0.06)	9.58±0.06 (21.43±0.01)	62.34±0.73 (27.38±0.18)	38.75±0.63 (24.69±0.05)	9.55±0.05 (21.45±0.001)	43.60±0.67 (25.27±0.05)
ROBUSTTS §	35.72±0.51 (20.84±0.04)	34.92±0.17 (23.08±0.06)	51.58±0.25 (21.73±0.02)	57.98±0.32 (24.51±0.45)	44.12±0.12 (23.28±0.003)	52.03±0.24 (25.00±0.03)
ROBUSTTS	**63.07±0.81** (26.49±0.18)	45.55±0.45 (23.28±0.02)	58.54±0.31 (26.93±0.06)	**58.22±0.19** (24.94±0.03)	44.32±0.15 (23.27±0.01)	52.62±0.23 (25.02±0.01)

Table 4: Classification accuracy (%) (PSNR in dB) on HDM05, averaged over the 5 folds, with the decoder part of an AE. * result outperforms DAE+AT and TCDAE.

Denoise-then-classify			Robust classification	
Method	Corruption Scale/Shift	Corruption Random	Method	Corruption Random
LSO	0.99 (13.39)	0.93 (15.49)	Mixup [1]	3.01
DAE* [25]	55.53 (28.68)	26.97 (26.67)	Augmentation	18.14
ROBUSTTS §	10.48 (20.41)	34.09 (25.88)	Aug.+Mixup	13.23
ROBUSTTS	54.09 (29.53)	38.88 (25.95)	Adv.+PGD [4]	4.23
ROBUSTTS++	**61.52** (29.67)	**41.43** (26.32)	ChoiceNet [28]	2.14

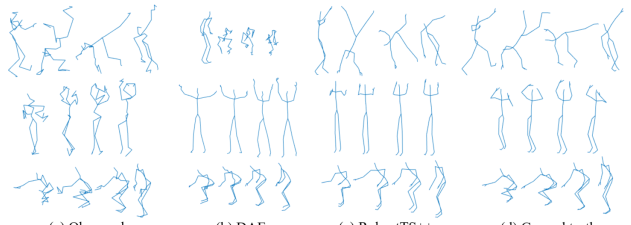

(a) Observed (b) DAE (c) RobustTS++ (d) Ground truth

Fig. 3: Reconstructed actions with "Random" corruption for the HDM05. $\mathcal{G}$ for ROBUSTTS++ is the decoder part of an AE trained on clean data. From the top row, the actions shown are "Cartwheel", "Throw basketball", and "Grab low".

As shown in Fig. 4, using more test sequences improves the performance and about 64 sequences are needed for good results using ROBUSTTS (NSN with TWN) and ROBUSTTS++ .

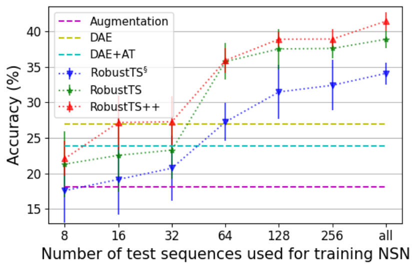
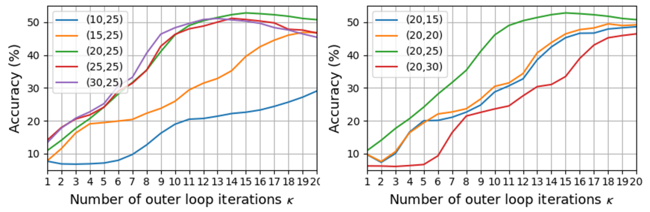


Fig. 4: Accuracy (%) of ROBUSTTS /ROBUSTTS++ on the HDM05 with different numbers of test sequences used for training NSN for "Random" corruption functions at test time.

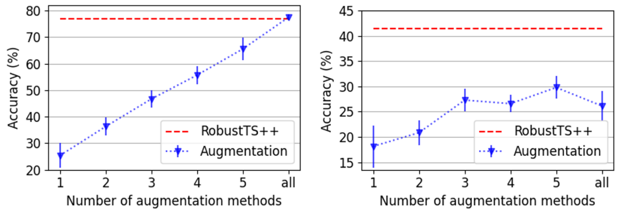
Effect of main hyperparameters: As shown in Fig. 5, we trained NSN with different combinations of parameters (η, λ) which denote the number of inner loop iterations of PGD (Equation (2)) and NSN training (Equation (3)) we use. We observe that the final classification accuracy and PSNR are relatively stable over different values of η and λ. Also importantly, ROBUSTTS exhibits convergence as k, the outer loop iterations reach about 20.



(a) Iterations for training NSN η (b) Iterations for PGD optimization λ

Fig. 5: Accuracy (%) of RobustTS with different number of iterations for training noise simulator networks η and PGD optimization λ on unknown ("Random") corruption functions for subj.1 of PAMAP2 (η, λ). (a) is results with different η (λ = 25) and (b) is results with different λ (η = 20).

Number of augmentation methods: Here, we show that recovering the signal using ROBUSTTS++ and using a classifier trained only on clean data performs better than directly classifying a noisy signal using a classifier trained using augmented noisy data, irrespective of the number of noise types used for augmentation. This is an important advantage of our method especially when the noise types may be unknown when the classifier is trained. For the augmented classifer, we choose every combination of k out of the 6 noise types we have, $k = 1, \ldots, 6$ and report average classification accuracy for each k. No augmentation is used for the classifier for ROBUSTTS++ . As shown in Fig. 6, for PAMAP2 dataset (subject 4), except when all augmentations are shown, ROBUSTTS++ significantly outperforms the robust classifier. For HDM05 dataset, ROBUSTTS++ performs much better in all cases. Therefore, ROBUSTTS++ is clearly the better approach when the noise process is different from during classifier training.



(a) PAMAP2 (Subject 4) (b) HDM05

Fig. 6: Accuracy (%) with "Random" corruption function for the different number of augmentation methods. The left is results for PAMAP2 (test subject 4) and the right is results for HDM05.

5. CONCLUSION

In this work, we proposed ROBUSTTS and ROBUSTTS++ which provide a framework to adapt to different noise models at the time of deployment. Using these methods, we showed how to recover clean signals at test time, and we are able to use pre-trained classifiers on the cleaned data with no need for fine tuning.

6. REFERENCES

[1] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[2] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan, "Augmix: A simple method to improve robustness and uncertainty under data shift," in *International Conference on Learning Representations*, 2020.

[3] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[4] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[5] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio, "Generative adversarial nets," in *Neural Information Processing Systems*, 2014.

[6] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar, "Time-series generative adversarial networks," *Neural Information Processing Systems*, 2019.

[7] Otto Fabius, Joost R van Amersfoort, and Diederik P Kingma, "Variational recurrent auto-encoders," in *International Conference on Learning Representations Workshops*, 2015.

[8] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf, "Wasserstein auto-encoders," in *International Conference on Learning Representations*, 2018.

[9] Steffen Moritz, Alexis Sardá, Thomas Bartz-Beielstein, Martin Zaefferer, and Jörg Stork, "Comparison of different methods for univariate time series imputation in r," *arXiv preprint arXiv:1510.03924*, 2015.

[10] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro, "Online dictionary learning for sparse coding," in *International Conference on Machine Learning*, 2009, pp. 689–696.

[11] Ali Mousavi and Richard G Baraniuk, "Learning to invert: Signal recovery via deep convolutional networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 2272–2276.

[12] Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, and Xiaojie Yuan, "Multivariate time series imputation with generative adversarial networks," in *International Conference on Neural Information Processing Systems*, 2018, pp. 1603–1614.

[13] Naohiro Tawara, Tetsunori Kobayashi, and Tetsuji Ogawa, "Multi-channel speech enhancement using time-domain convolutional denoising autoencoder.," in *INTERSPEECH*, 2019, pp. 86–90.

[14] Woong-Hee Lee, Mustafa Ozger, Ursula Challita, and Ki Won Sung, "Noise learning-based denoising autoencoder," *IEEE Communications Letters*, vol. 25, no. 9, pp. 2983–2987, 2021.

[15] Rushil Anirudh, Jayaraman J Thiagarajan, Bhavya Kailkhura, and Peer-Timo Bremer, "MimicGAN: Robust projection onto image manifolds with corruption mimicking," *International Journal of Computer Vision*, pp. 1–19, 2020.

[16] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151–175, 2010.

[17] Suhas Lohit, Rushil Anirudh, and Pavan Turaga, "Recovering trajectories of unmarked joints in 3d human actions using latent space optimization," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2342–2351.

[18] Kaushik Koneripalli, Suhas Lohit, Rushil Anirudh, and Pavan Turaga, "Rate-invariant autoencoding of time series," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020.

[19] Attila Reiss and Didier Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *International Symposium on Wearable Computers*, 2012, pp. 108–109.

[20] Meinard Müller, Tido Röder, Michael Clausen, Bernhard Eberhardt, Björn Krüger, and Andreas Weber, "Documentation mocap database hdm05," *Technical Report CG-2008-2*, June 2007.

[21] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2013.

[22] Andrew A Cook, Göksel Mısırlı, and Zhong Fan, "Anomaly detection for iot time-series data: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2019.

[23] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu, "Time series data augmentation for deep learning: A survey," in *Proceedings of the International Joint Conference on Artificial Intelligence,*, 8 2021, pp. 4653–4660.

[24] Xi Wang and Chen Wang, "Time series data cleaning: A survey," *IEEE Access*, vol. 8, pp. 1866–1881, 2019.

[25] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, "Extracting and composing robust features with denoising autoencoders," in *International Conference on Machine Learning*, 2008, pp. 1096–1103.

[26] Bing Ma, Xiaoru Wang, Heng Zhang, Fu Li, and Jiawang Dan, "Cbam-gan: generative adversarial networks based on convolutional block attention module," in *International Conference on Artificial Intelligence and Security*, 2019, pp. 227–236.

[27] Florian Tramèr and Dan Boneh, "Adversarial training and robustness for multiple perturbations," in *Conference on Neural Information Processing Systems*, 2019, vol. 32.

[28] Sungjoon Choi, Sanghoon Hong, Kyungjae Lee, and Sungbin Lim, "Task agnostic robust learning on corrupt outputs by correlation-guided mixture density networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3872–3881.