Paper ID: 2745

ICASSP 2023 — 4 - 10 JUNE, RHODES ISLAND, GREECE

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

Eun Som Jeon[a], Suhas Lohit[b], Rushil Anirudh[c], Pavan Turaga[a]

[a]Geometric Media Lab, Arizona State University [b]Mitsubishi Electric Research Laboratories

[c]Lawrence Livermore National Laboratory

Visit GML!

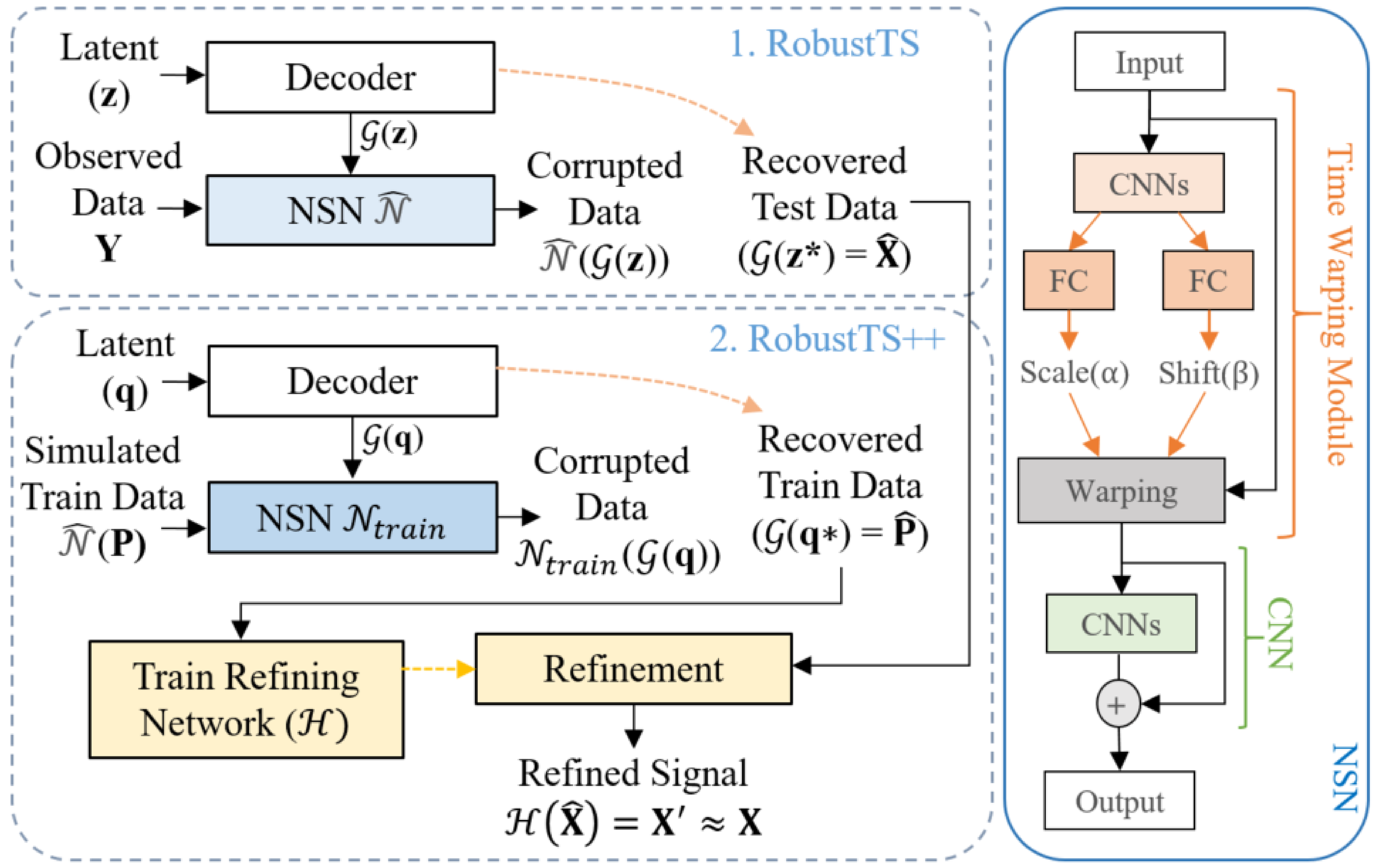
Code Available!

Introduction

- Given a pretrained decoder or a generative model trained on clean data, can we recover clean time series purely at test time, even when the noise model is unknown and can vary?
- We propose a robust framework for recovering time series under ***"unknown noise"***.
- We also show that the recovered time series can be directly used on pretrained classifiers, trained on clean data, without a large drop in accuracy.
- We demonstrate improved performance over existing methods for multiple datasets from diverse domains.

Proposed Frameworks: RobustTS and RobustTS++

A decoder model $\mathcal{G}$ trained on clean data $\mathbf{X}$ is used to estimate the unknown noise by using a Noise Simulator Networks (NSN).



Our goal is to optimize the following objective:

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

RobustTS: We solve the problem with an alternating optimization framework.

1) Fix $\widehat{\mathcal{N}}$ and optimize $\mathbf{z}^*$ using a projected gradient descent (PGD). j=1, …, n,

$$\mathbf{z}_j^{(k+1)} = \underset{\mathbf{z}_j}{\operatorname{argmin}} \|Y_j - \widehat{\mathcal{N}}(\mathcal{G}(\mathbf{z}_j), \Theta^{(k)})\|_2^2$$

2) Fix $\{z_j^*\}_{j=1}^n$ and optimize for θ (parameters of NSN):

$$\Theta^{(k+1)} = \underset{\Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{j=1}^{n} \|Y_j - \widehat{\mathcal{N}}(\mathcal{G}(\mathbf{z}_j^{(k+1)}), \Theta)\|_2^2$$

RobustTS++: To further refine the outputs obtained from the RobustTS stage, we revisit the training set $\mathbf{P}$. Recovered data $\widehat{\mathbf{P}}$ is created from the entire corrupted training data. We train a neural network $\mathcal{H}$. Finally, the refined $\mathbf{X}' = \mathcal{H}(\widehat{\mathbf{X}})$ is obtained, which is closer to $\mathbf{X}$.

Experiments & Results

We sample from a set of corruption functions generated randomly at test time, including Gaussian noise, time shift and scale, calibration error, etc.

We test our method on two activity recognition datasets: (1) PAMAP2 which uses IMUs and heart rate and (2) HDM05 which uses motion.

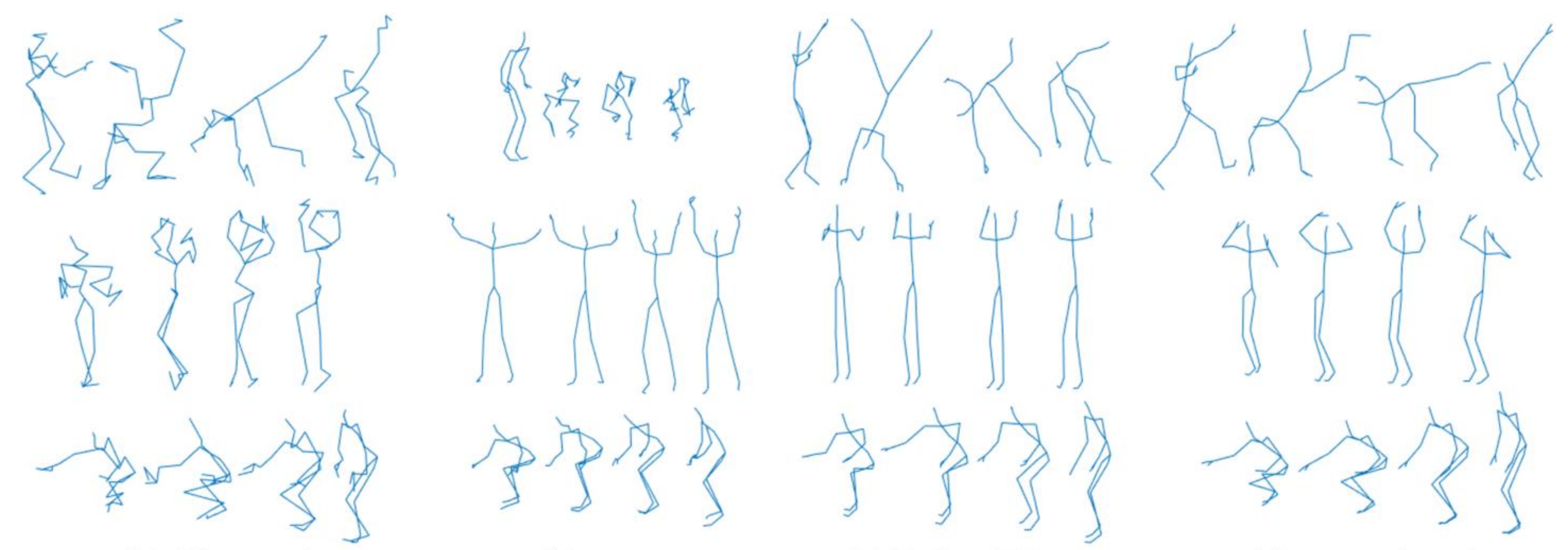
PAMAP2

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)

HDM05

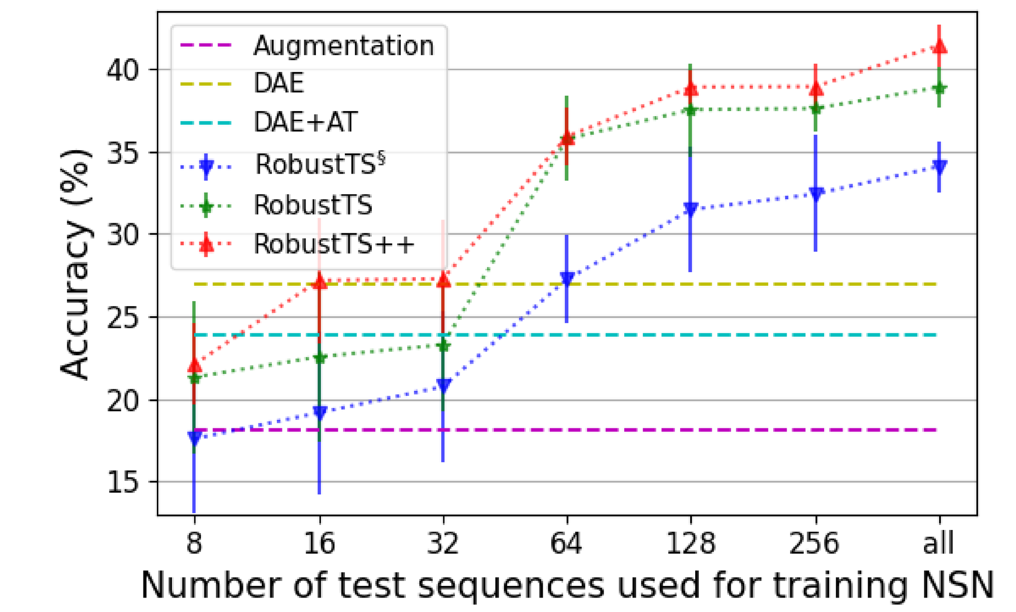
Denoise-then-classify			Robust classification	
Method	Corruption Scale/Shift	Corruption Random	Method	Corruption Random
LSO	0.99 (13.39)	0.93 (15.49)	Mixup [1]	3.01
DAE* [25]	55.53 (28.68)	26.97 (26.67)	Augmentation	18.14
RobustTS §	10.48 (20.41)	34.09 (25.88)	Aug.+Mixup	13.23
RobustTS	54.09 (29.53)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

Reconstructed actions on HDM05

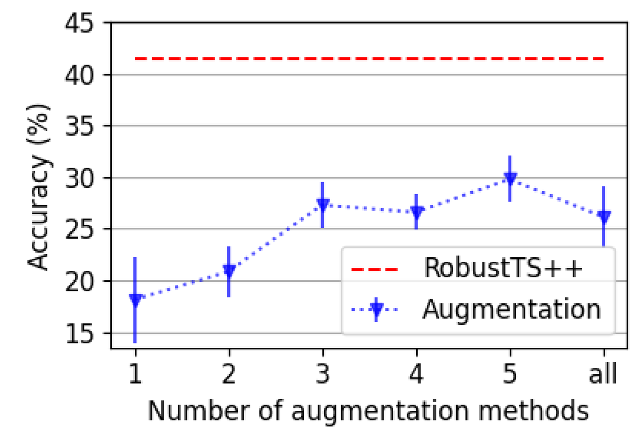


(a) Observed (b) DAE (c) RobustTS++ (d) Ground truth

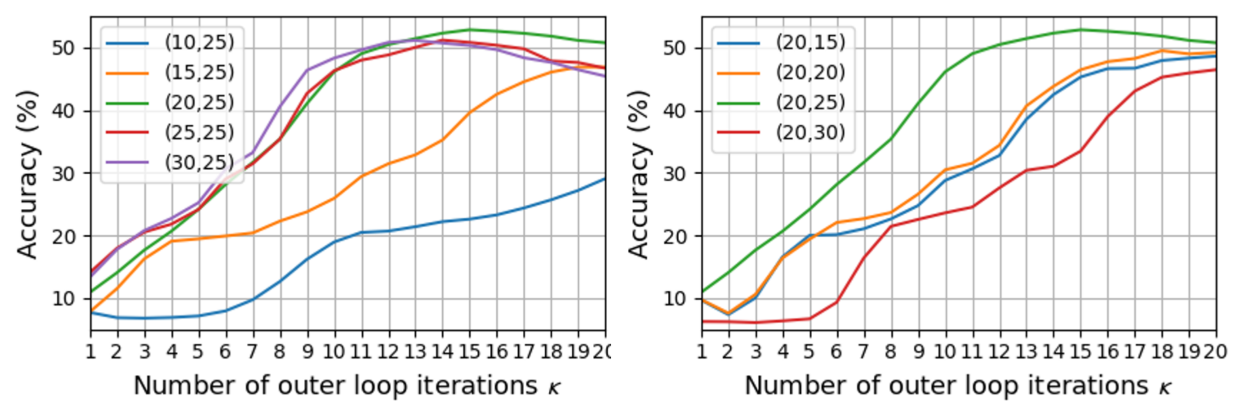
Effect of size of dataset for training NSN



Augmentation methods



Sensitivity to different η, λ



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

Conclusion

We proposed RobustTS and RobustTS++ which provide a framework to adapt to different noise models at the time of deployment. Using these methods, we are able to ***recover clean signals at test time*** and ***use pre-trained classifiers*** with no need for fine tuning.