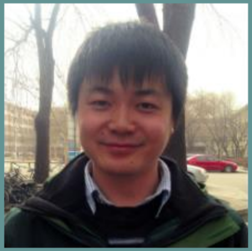
The 38th Annual AAAI Conference on Artificial Intelligence

WASHINGTON STATE UNIVERSITY

Tutorial on Advances in Robust Time-Series ML: From Theory to Practice

Taha Belkhouja


Yan Yan

Nghia Hoang

Ganapati Bhat

Jana Doppa

Website: https://robustml-tsd-tutorial.github.io/

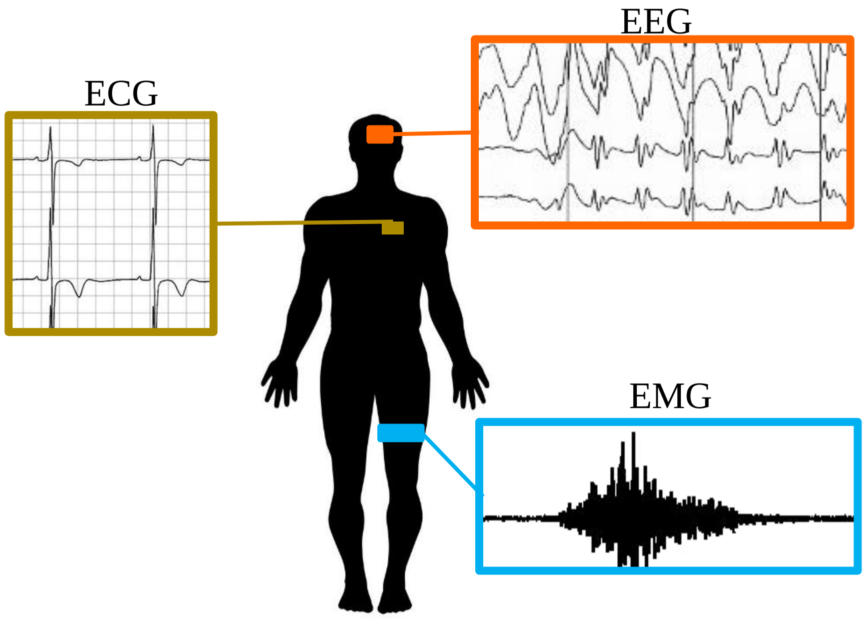
Outline

1. Machine Learning Robustness Challenges
2. Adversarial robustness for classification – Part 1
3. Robust applications for wearable sensors
4. Adversarial robustness for classification – Part 2
5. Adversarial robustness for forecasting
6. Out-of-Distribution detection
7. Future research directions

Time-Series ML and Robustness Challenges

Time-Series Data is Ubiquitous

- IoTs
  - Smart Homes
  - Smart Health
  - Wearables
- Finance
  - Sales/Stocks
  - Customer demand
- Monitoring systems
  - Smart grids



ML Application of Time-Series

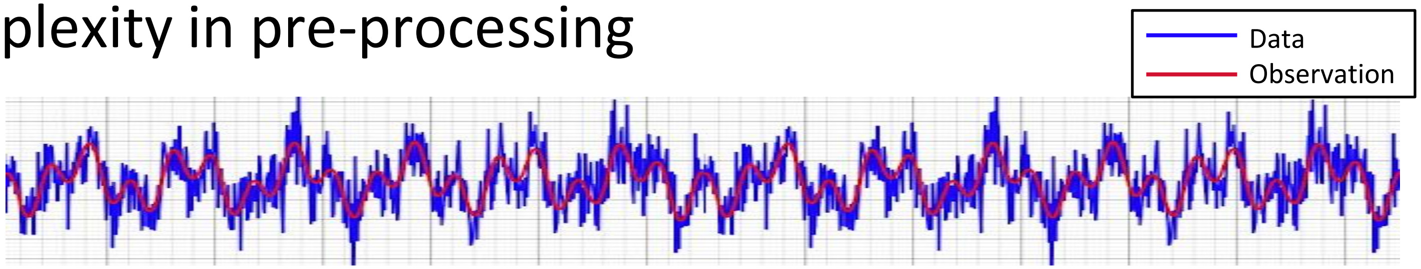
- Classification
  - Human Activity Recognition, Medical diagnosis
- Forecasting
  - Weather prediction, Stock prediction
- Imputation
  - Medical data collection
- Outlier detection
  - Monitoring systems

ML Models of Time-Series

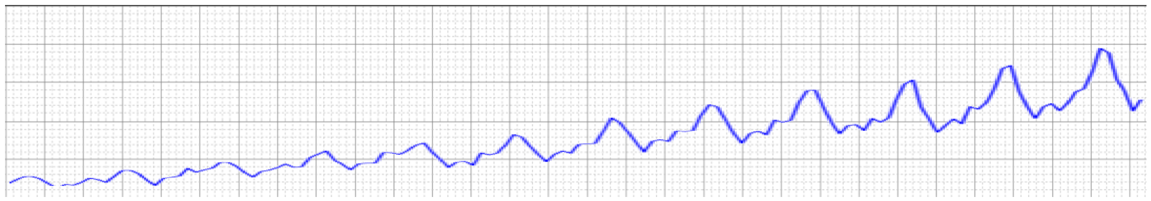
- Classical
  - ARIMA
  - VARMA
  - Linear regression
  - Regression random forest
- Deep learning
  - LSTM
  - Inception time-series
  - GAN time series
  - Transformers

Time-Series Challenges

- Noise
- Complexity in pre-processing



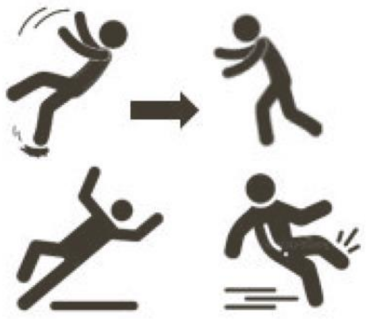
- Modeling temporal behavior
- Periodicity / Stationarity

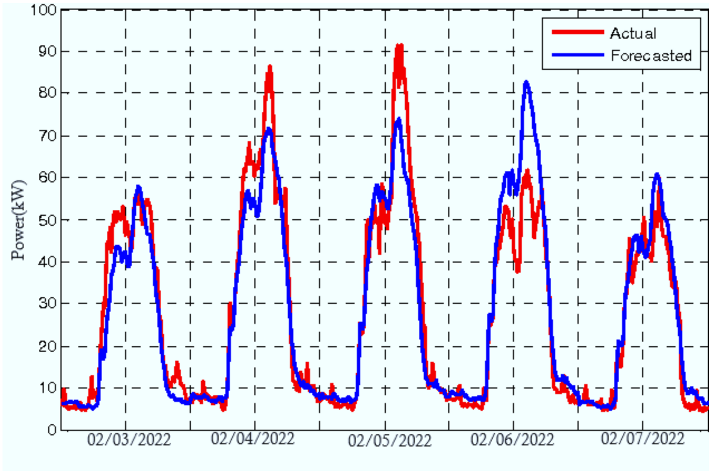


- Outliers

Time-Series Challenges

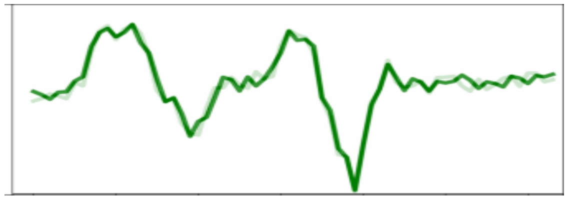
- The need for robust ML models for time-series data:
  - The data is vulnerable to several corruption threats
  - Importance for safety-critical application
  - Avoid catastrophic scenario
    - Fall detection, medical diagnosis, prediction stability of smart grid



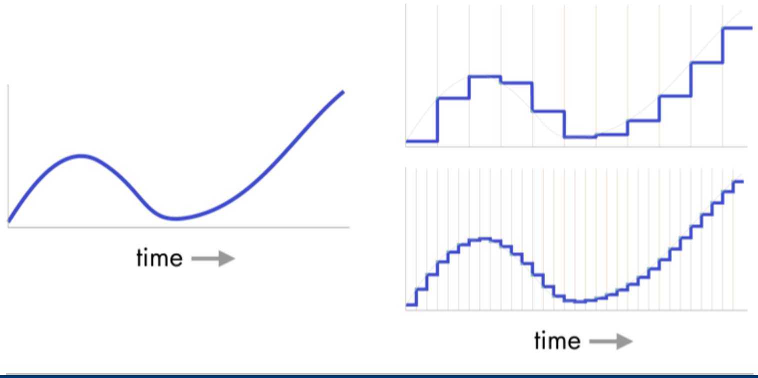


Robustness challenges: Natural perturbations

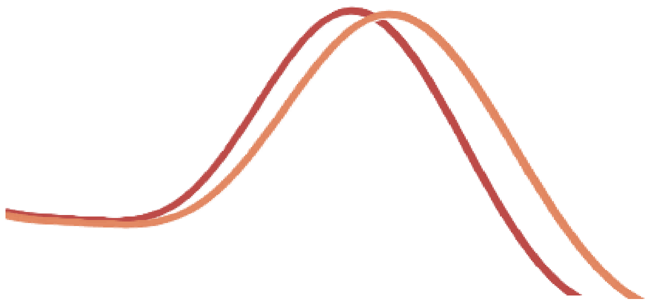
- Random Noise
- Deployment of IoTs in real-world settings:
  - Sensor mis-calibration
  - Sensor's change in orientation



  - Record sampling mismatch

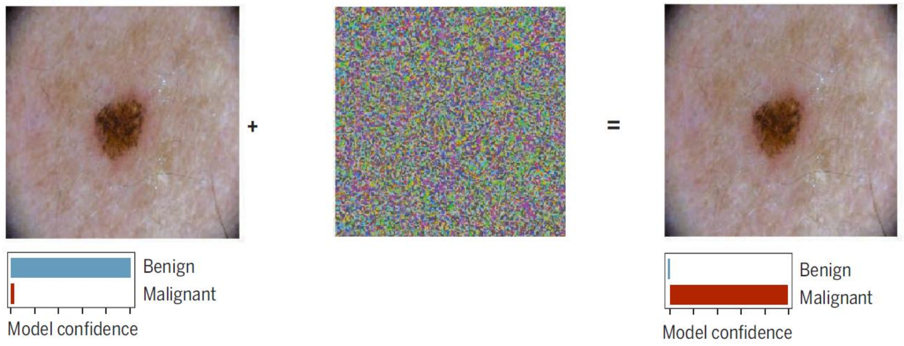


  - Temporal delays

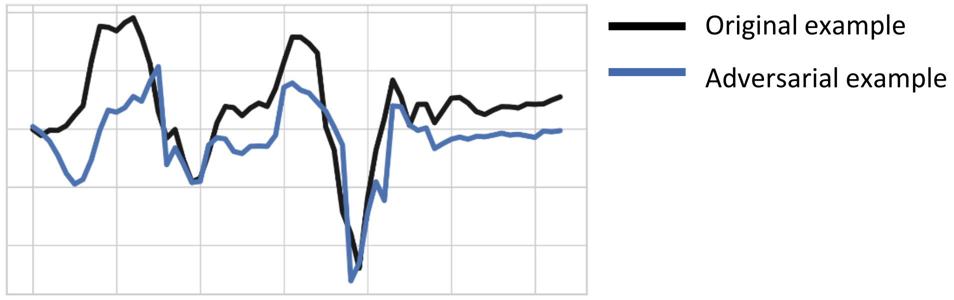


Robustness challenges: Adversarial perturbations

- Medical case example:



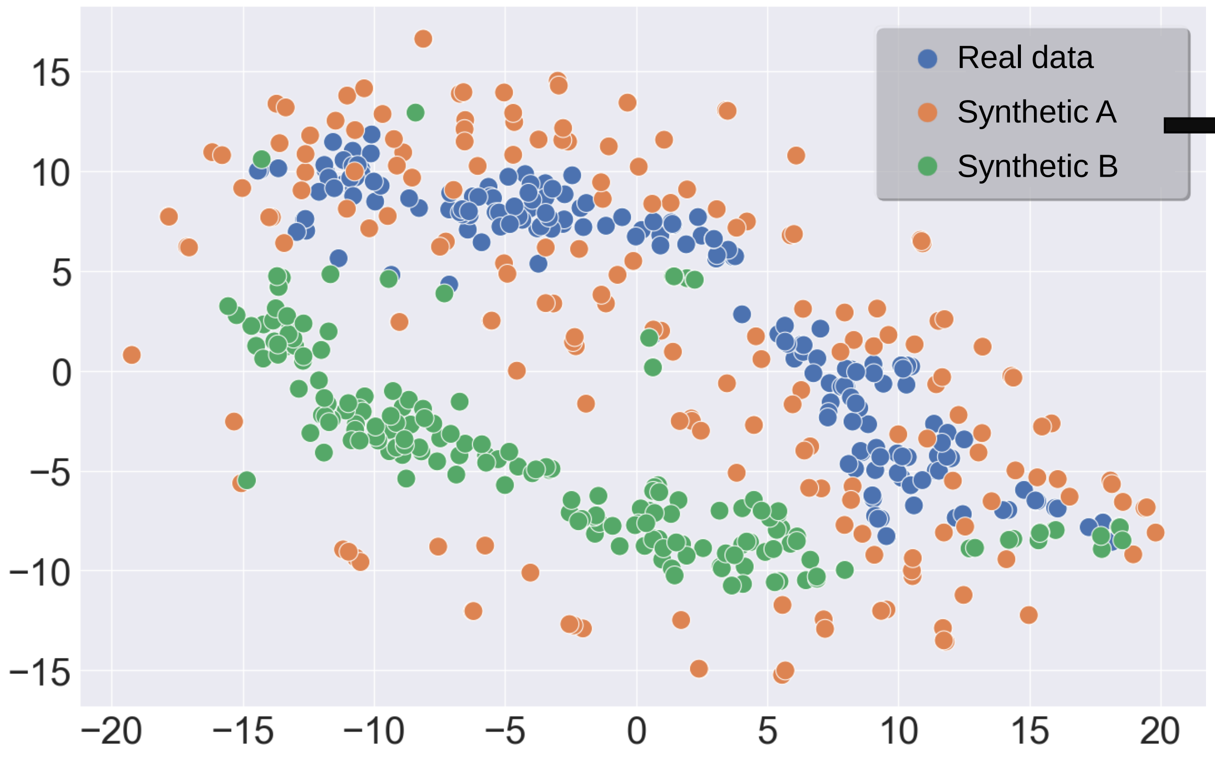
Adversarial misrepresentation of the data in image-based AI systems.[1]



Adversarial misrepresentation of the temporal clinical data in sensor-based AI systems.

[1]: Finlayson, Samuel G., et al. "Adversarial attacks on medical machine learning." *Science* 363.6433 (2019): 1287-1289.

Reliability challenges: Data Validation



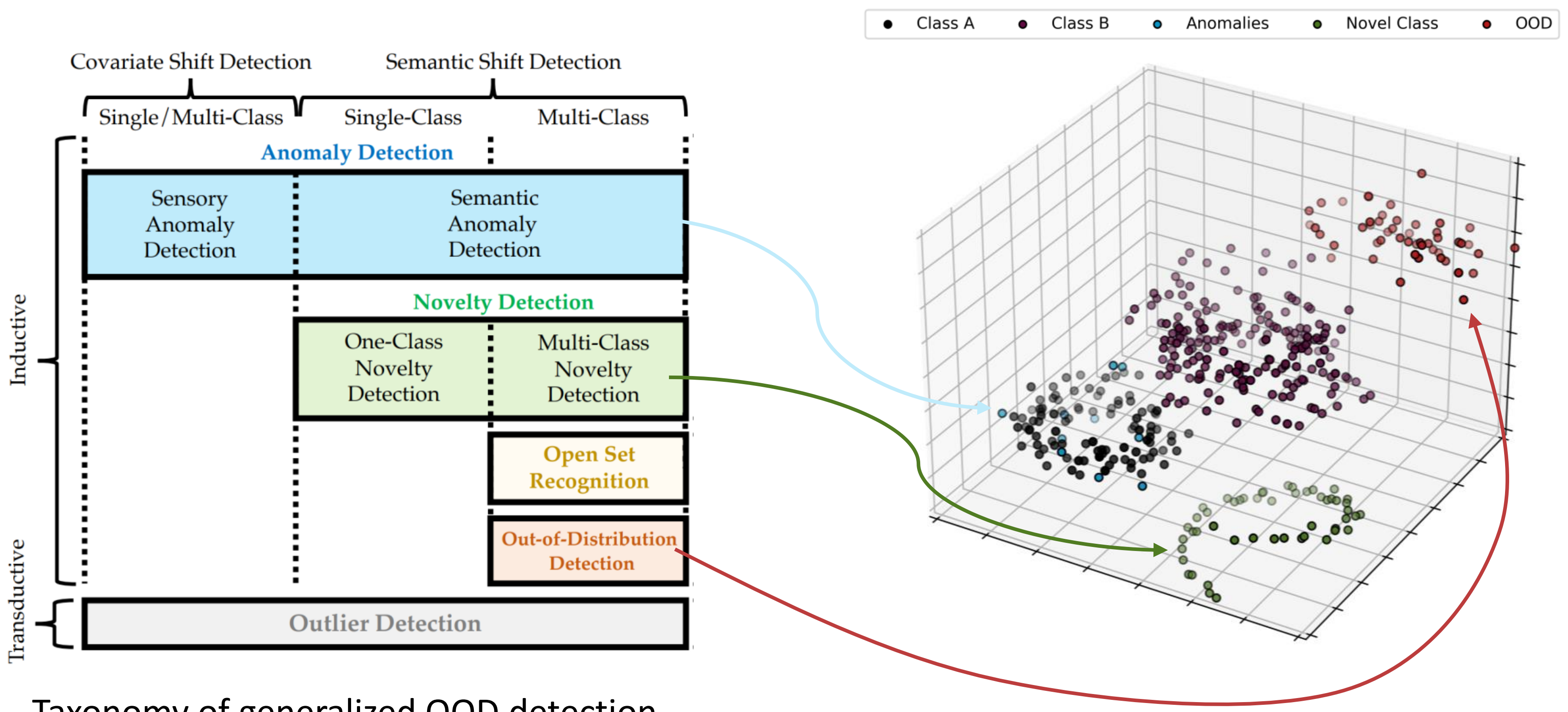
→ Which of the synthetic data should we retain?

t-Distributed Stochastic Neighbor Embedding showing the distribution of real-world data examples and synthetic data.

- The automatic validation of unseen time-series examples is ambiguous

Novel distribution taxonomy

- The need to understand to which underlying distribution the data belongs



Taxonomy of generalized OOD detection framework, illustrated by classification tasks.[2]

[2]: Yang, Jingkang, et al. "Generalized out-of-distribution detection: A survey." arXiv preprint arXiv:2110.11334 (2021).

Out-of-distribution for Time-Series Data

- Out-of-distribution detection is critical for safe AI deployment
- Challenges:
  - Non-stationary property
  - Temporal modeling complexity
  - Semantic features
  - Pre-processing
  - Ambiguity of using synthetic data
  - Ambiguity of Human-in-the-loop methods

Adversarial robustness for time-series

Classification Application for Time-Series

- Human activity recognition
- Monitoring systems
- Medical diagnosis

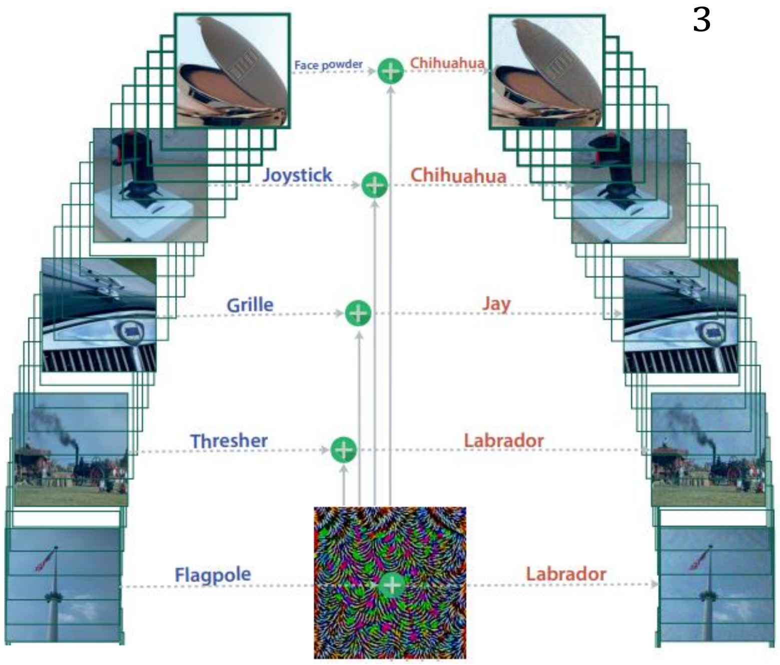
Why adversarial robustness

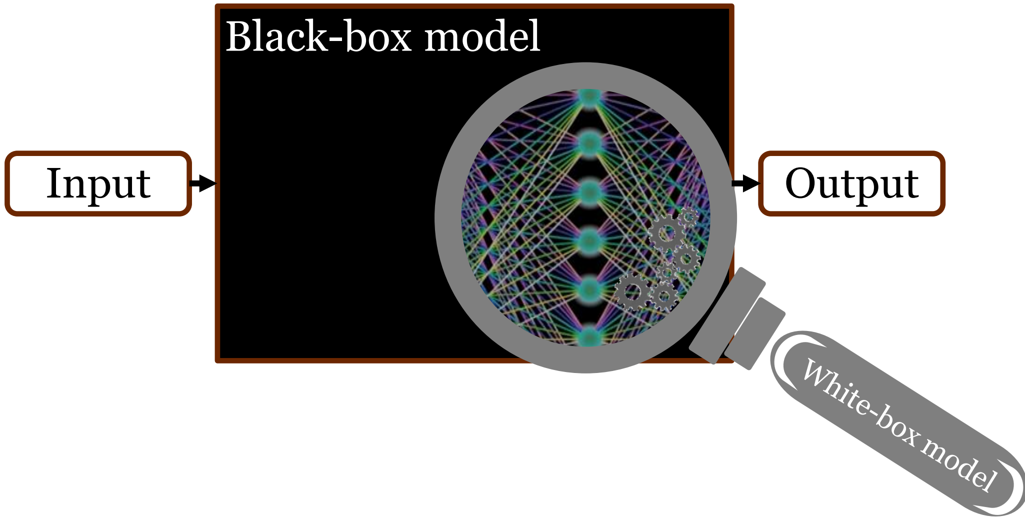
- Prediction reliability against natural and hand-crafted perturbation
- Investigate worst-case scenario within the threat vector
- Improves over standard data augmentation techniques
- Resilient against the over-confidence phenomenon of deep models

Types of adversarial attacks

Single instance vs Universal attack

Black-box vs White box





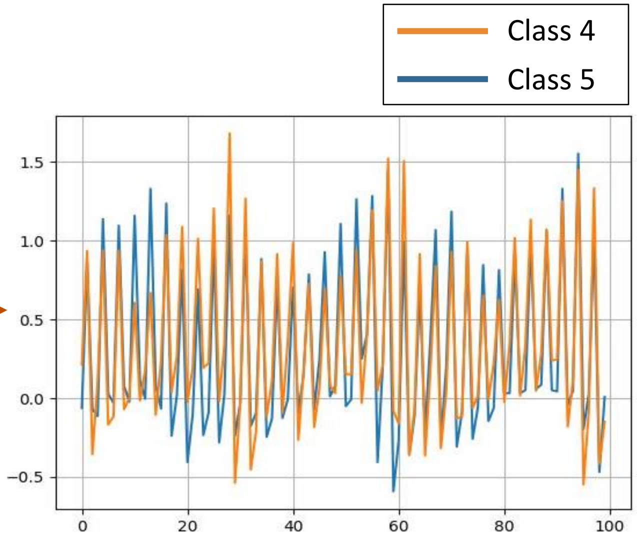
[3]:Moosavi−Dezfooli, et al. "Universal adversarial perturbations." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Limitations of standard l_p norms

- Standard adversarial algorithm rely on l_p-norm to constraint adversarial attacks
- Case Study: WISDM dataset

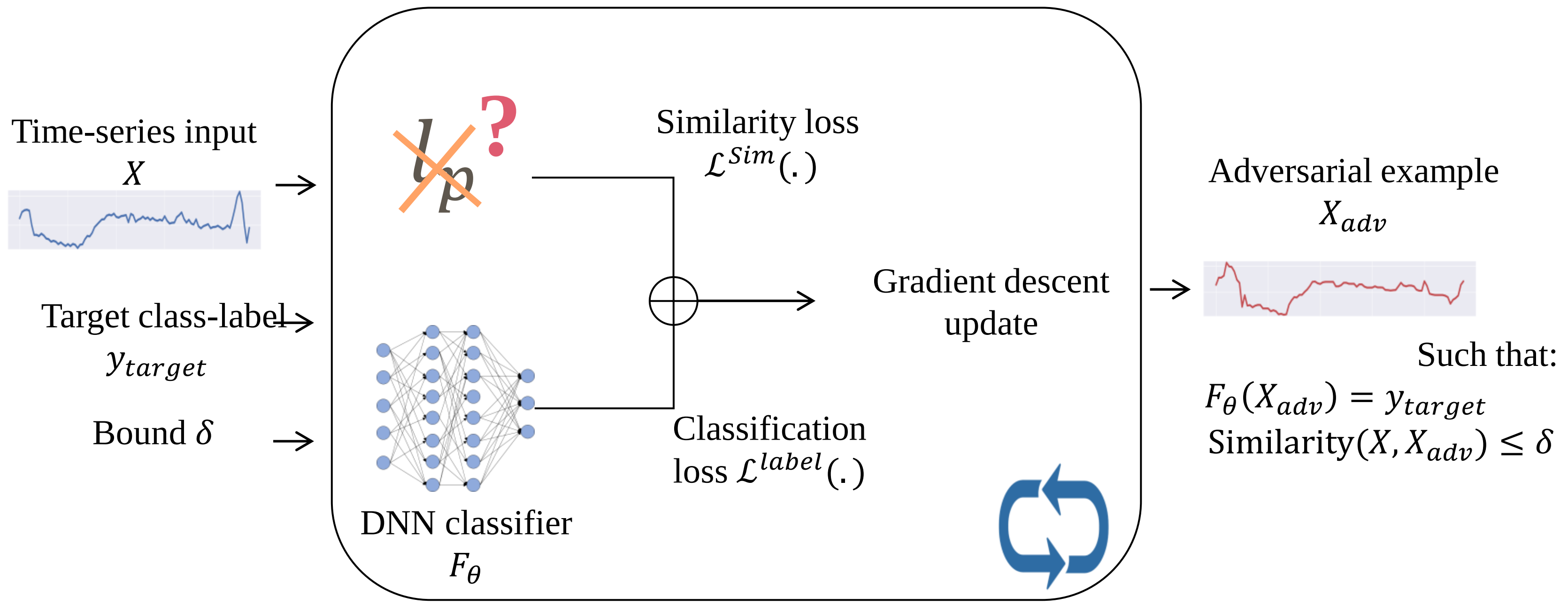
Class 2	0.015				
Class 3	0.009	0.014			
Class 4	0.011	0.012	0.011		
Class 5	0.009	0.014	0.008	0.008	
Class 6	0.007	0.012	0.007	0.008	0.003
	Class 1	Class 2	Class 3	Class 4	Class 5

Minimum normalized l_2 distance between examples from different classes



→ Perturbations based on Euclidean distance can result in adversarial time-series signals which semantically belong to a different class-label.

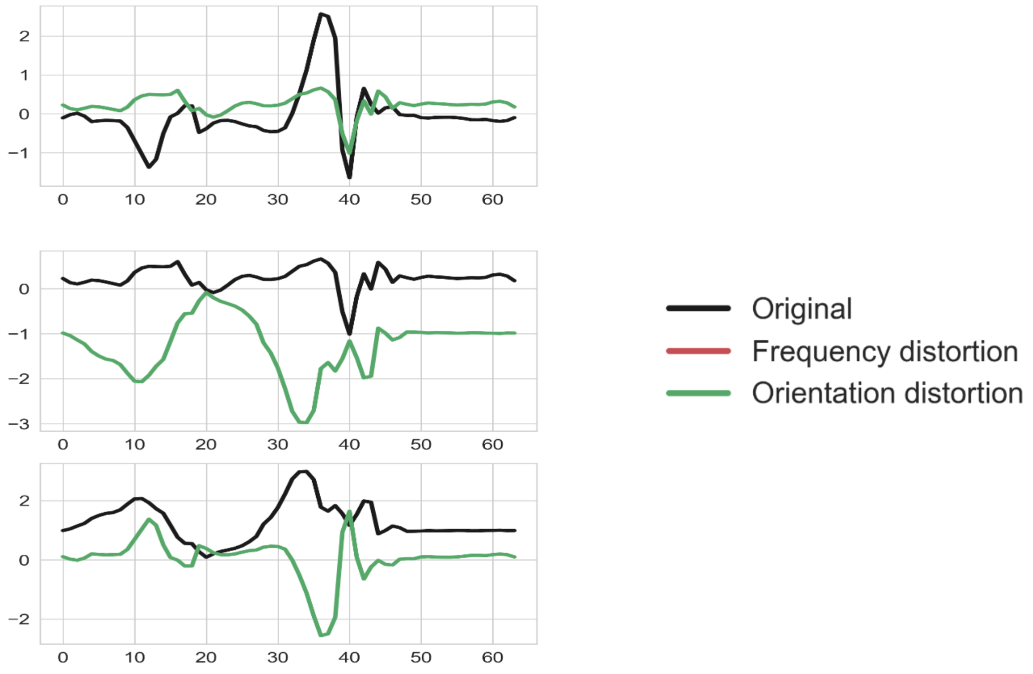
General Optimization Framework

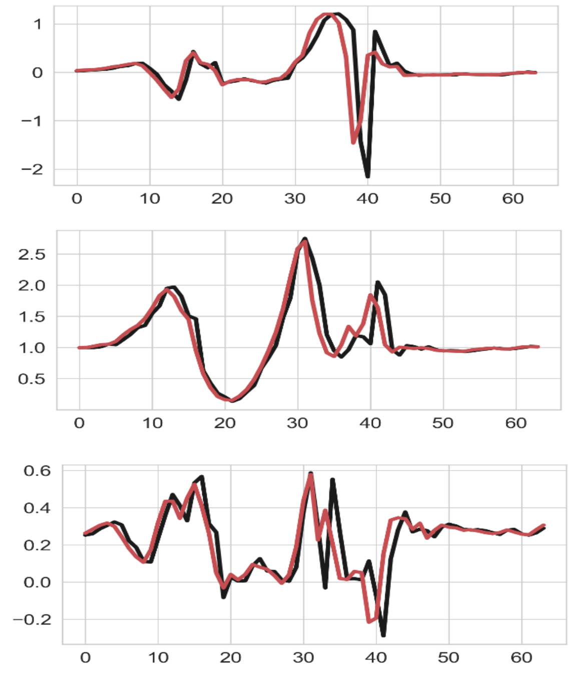


Time-series input X
Target class-label y_{target}
Bound δ
l_p?
Similarity loss $\mathcal{L}^{Sim}(.)$
DNN classifier F_θ
Classification loss $\mathcal{L}^{label}(.)$
Gradient descent update
Adversarial example X_{adv}
Such that:
$F_\theta(X_{adv}) = y_{target}$
$\text{Similarity}(X, X_{adv}) \leq \delta$

→ How to address the similarity loss?

Similarity Challenge





Approach 1:

- Extract statistical features that better represents the semantics of each class

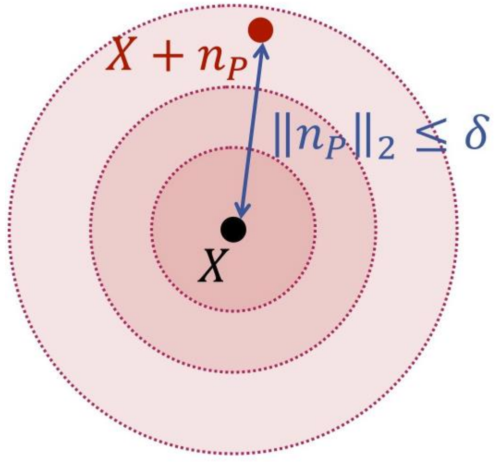
Approach 2:

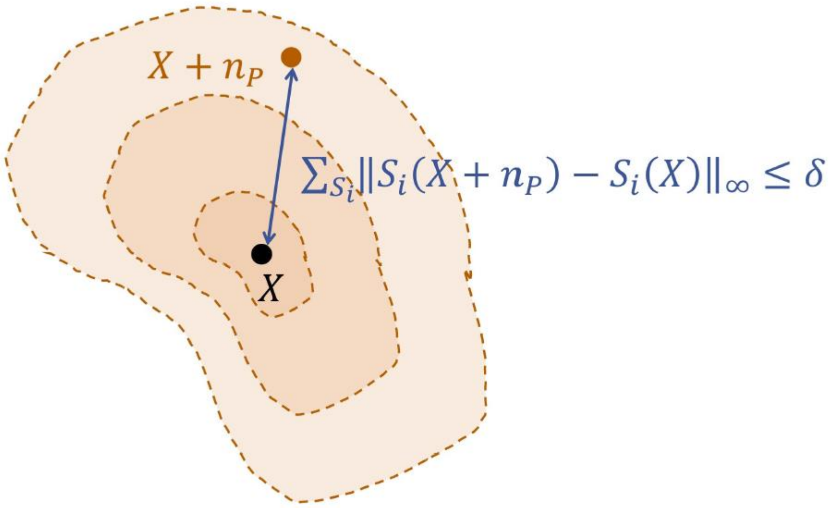
- Improve over l_p-norm-based distance by considering the temporal shifts

Approach 1: Similarity within the statistical space

- We investigate statistical features as similarity features

- Time-series data are comprehensible using multiple statistical tools
  - Mean, Standard Deviation, Skewness…





Adversarial transformation hypothesis

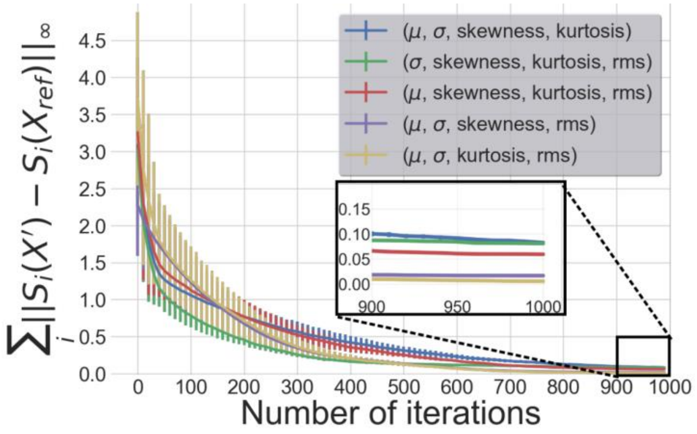
- Polynomial transformation: $X_{adv} = PT(X)$ vs. $X_{adv} = X + \delta$
  - Inspired by power series, we approximate adversarial transformation using a polynomial representation with a chosen degree *d*: $PT(X) = \sum_{k=0}^{d} a_k X^k + O(X^{d+1})$
  - Reduces search complexity by generalizing to a universal perturbation

Theorem. Using the statistical space, a larger set of possible adversarial attacks can be generated by polynomial transformations than standard additive perturbations.

$$\left\{ X_{adv} = \mathcal{PT}(X),\ \forall a_k \right\} \supsetneq \left\{ X_{adv} = X + \delta,\ \forall \delta \right\}$$

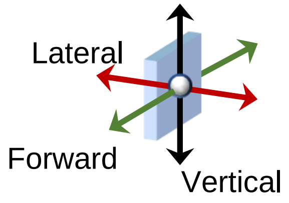
Which features to choose?

- Domain-agnostic

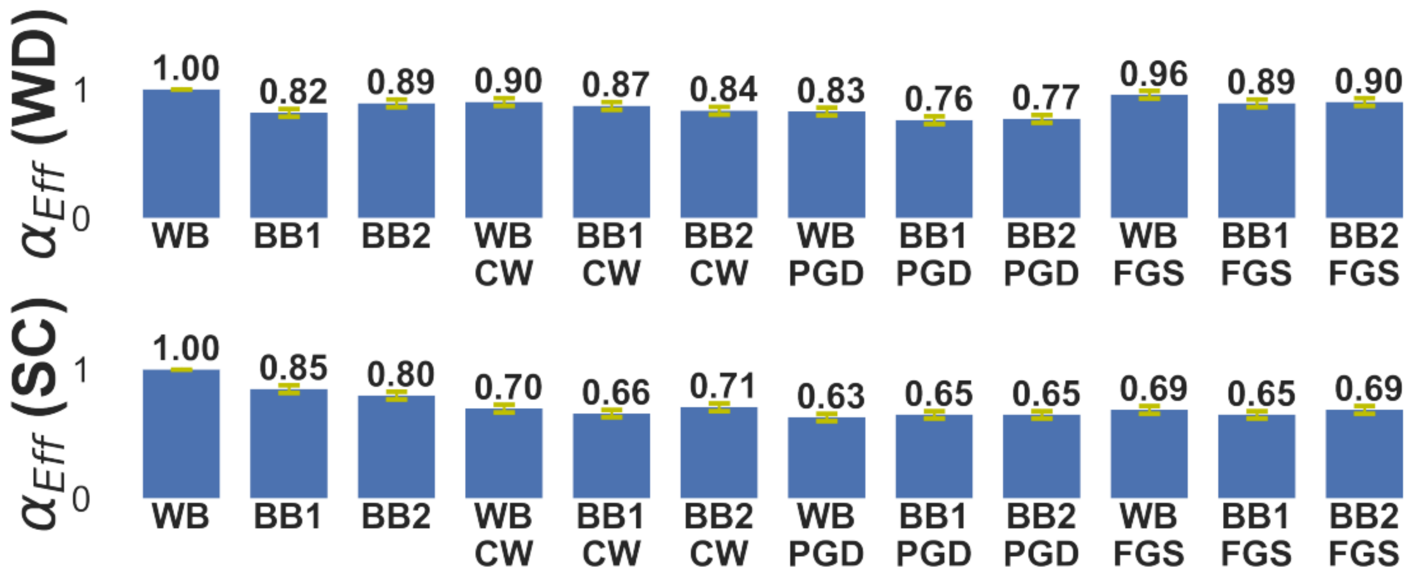


Convergence of the statistical loss using different statistical constraint sets

- Domain-specific
  - Accelerometers: Use of body acceleration computed from all accelerometer axes.

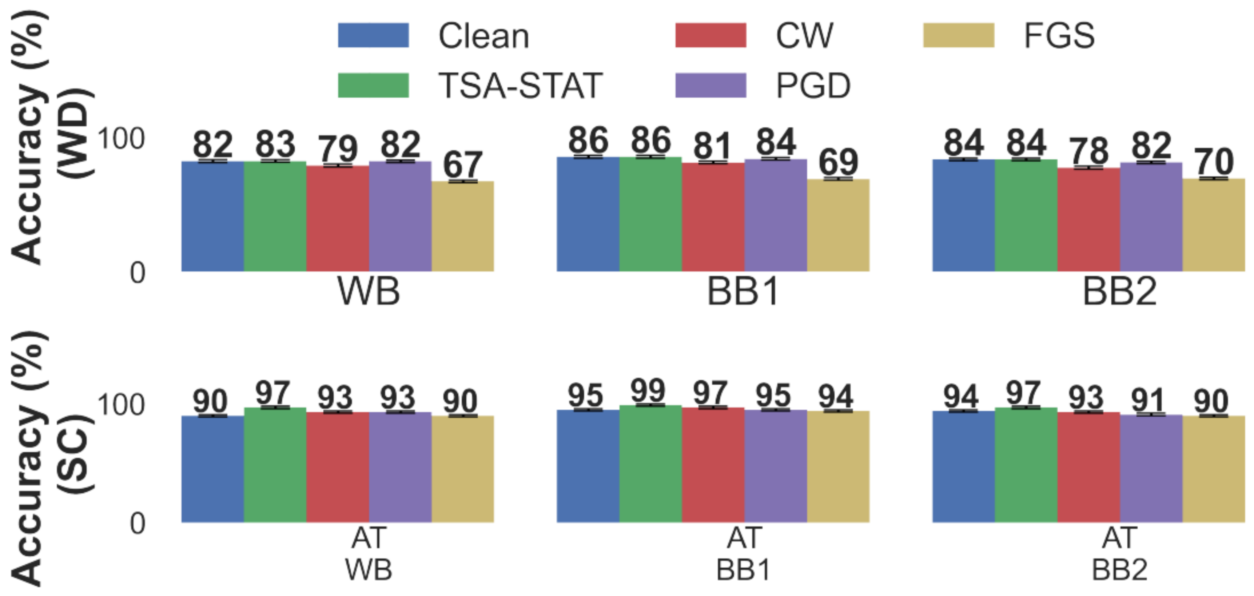


General results: Attack performance



Results of different adversarial algorithms attack performance under white-box and black-box settings on different deep models.

General results: Robustness performance



Results for adversarial training using adversarial examples from different adversarial algorithms for different deep models.

Robust Wearable Applications: Sensor Disturbances and Missing Data

Outline

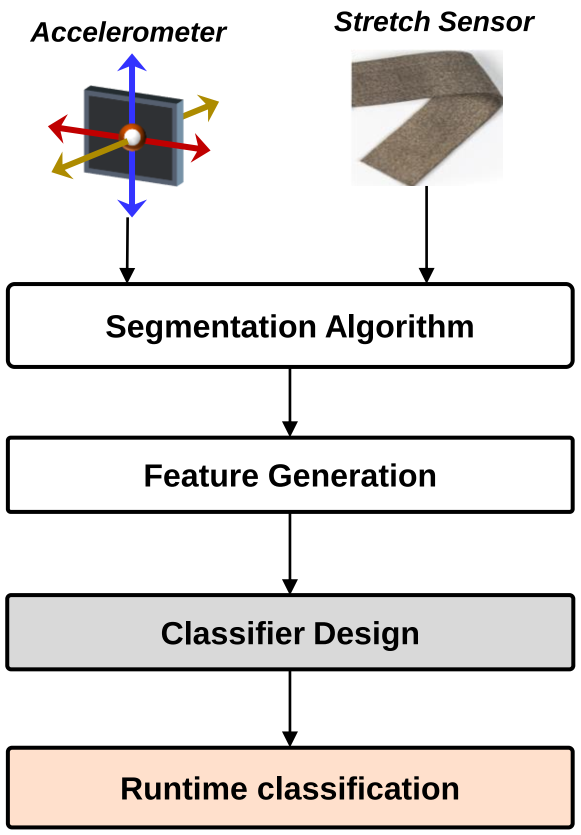
- Wearable applications and ML
- Sensor disturbances in wearable devices
- Missing data and solutions
  - Accuracy preserving imputation
  - Clustering-based imputation

Wearable Devices for Health Monitoring

- Cost of healthcare and incidence of chronic diseases are increasing
- Wearable devices are popular for monitoring
  - Low-cost and small form-factor wearable devices offer great potential
  - Integration of multiple sensors, communication and processing

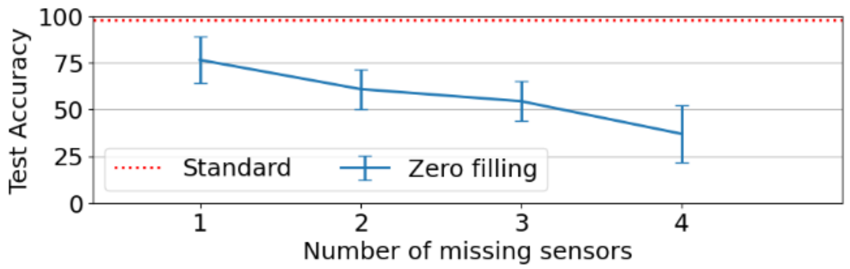
Implementing Health Apps on Wearables

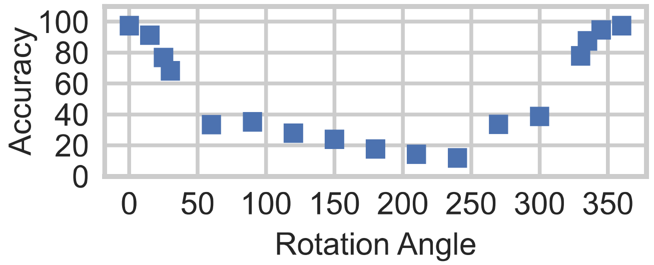
- Data collection and segmentation
  - Segment data into fixed or variable length windows
  - Label the data with actual activities
- Feature generation
  - Handcrafted features or deep learning
- Classifier training
  - Supervised learning to map features to activity
  - Trained using labeled feature and activity pairs
- Use trained classifier to identify activities at runtime



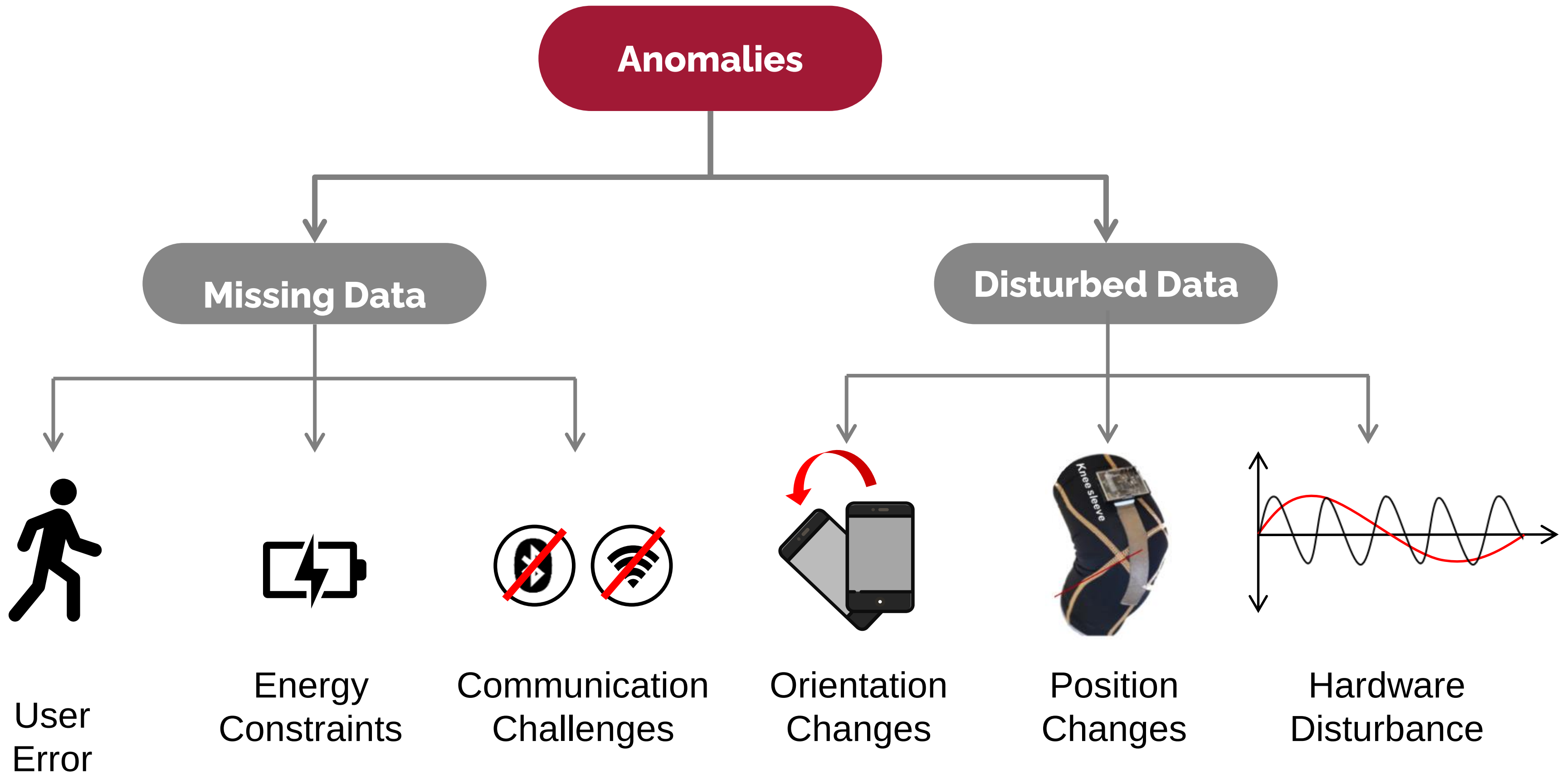
Challenges to Wearable Health Monitoring

- Health Monitoring approaches use multiple sensors for high accuracy
  - Accelerometers, gyroscopes, stretch sensor
  - Data from sensors used to train the classifier
- Most classifiers assume
  - Clean data without any missingness
  - Fixed sensor position and data distribution
- Changes in data distribution affects classification





Types of Anomalies in Wearable Applications



Outline

- Wearable applications and ML
- Sensor disturbances in wearable devices
- Missing data and solutions
  - Accuracy preserving imputation
  - Clustering-based imputation

Human Activity Recognition (HAR) Application

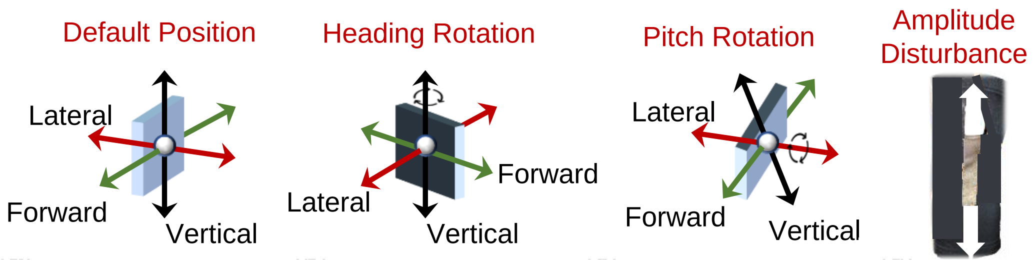
- HAR identifies activities, such as walking, sitting, driving, jogging
- It is the first step to solutions for movement disorders

We must know what the patient is doing to reach a conclusion

- HAR can provide valuable insight to health specialists
- Applications of HAR
  - Patient rehabilitation
  - Fall detection
  - Physical activity promotion

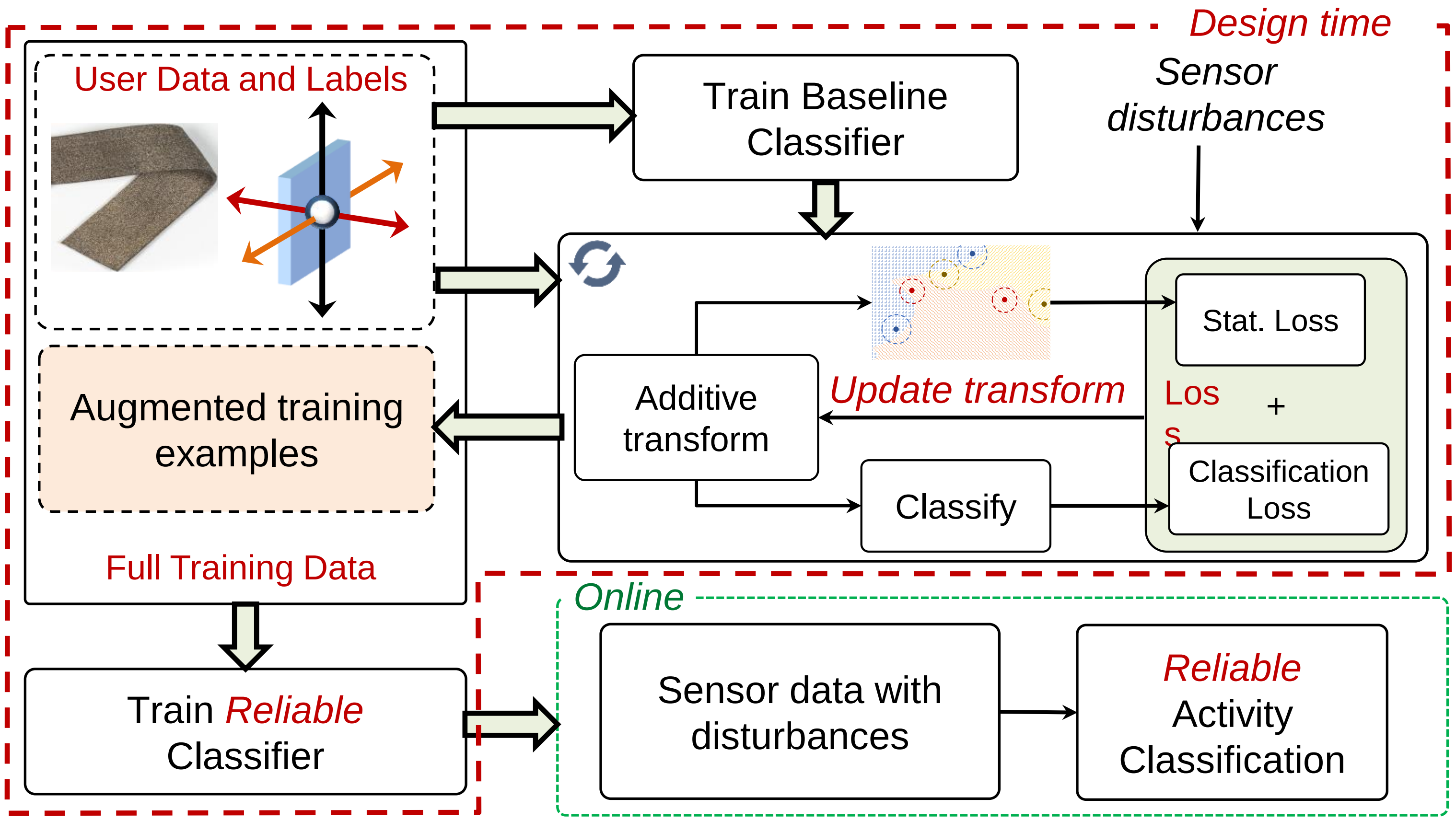
Sensor Disturbances in HAR

- We consider four major types of sensor disturbances
- Heading rotation
  - Changes sensor values in forward and lateral directions
- Pitch rotation
  - Changes sensor values in forward and vertical directions
- Amplitude and sensor hardware disturbances



We show methods to handle disturbances with no overhead

Overview of the Proposed StatOpt Approach



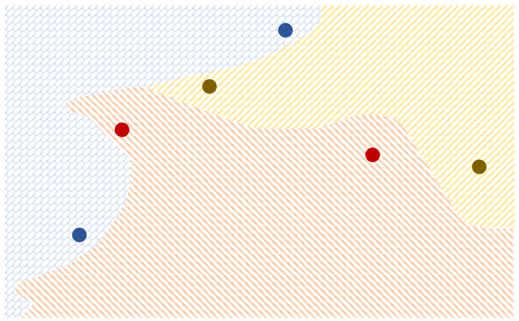
Statistical Optimization Problem setup

- StatOpt creates new training examples to capture disturbances
- We define disturbances as additive perturbations p
- Examples generated by StatOpt can be written as:

$$X' = X + p$$

- New examples must satisfy the following requirements
  - Misclassification by the classifier
  - p describes the natural disturbances in the sensor data

▪ It is challenging to define p explicitly



● Time-series sensor input

Natural sensor disturbances that can affect the classifier's prediction

▨ Class boundaries learned by ML classifier from training data

Statistical Optimization Algorithm Setup

- We employ statistical features S_i to generate training examples
  - The statistical features ensure that the distribution of data does not deviate
- Create perturbation p such that $\left||S_i(X) - S_i(X')|\right| < \epsilon$

1. Body acceleration
   - Total acceleration across all three directions of motion
   - Generally, body acceleration is same even if sensor is disturbed
2. Skewness
   - Measures the symmetry of the distribution
3. Kurtosis
   - Measures the distribution tail of the input values

Statistical Optimization Algorithm

- Define two losses to obtain perturbation p
- Statistical Loss

$$\mathcal{L}^{stat}(\boldsymbol{p}, \boldsymbol{X}) \triangleq \sum_{\boldsymbol{S}_i} ||\boldsymbol{S}_i(\boldsymbol{X} + \boldsymbol{p}) - \boldsymbol{S}_i(\boldsymbol{X})||$$

- Classification loss

$$\mathcal{L}^{label}(\boldsymbol{p}, \boldsymbol{X}, \boldsymbol{y}') \triangleq \max[\max_{\boldsymbol{y} \neq \boldsymbol{y}'} \left(\boldsymbol{Z}_{\boldsymbol{y}}(\boldsymbol{X} + \boldsymbol{p})\right) - \boldsymbol{Z}_{\boldsymbol{y}'}(\boldsymbol{X} + \boldsymbol{p}), \boldsymbol{\rho}]$$

where $\boldsymbol{Z}_{\boldsymbol{y}}$ is the output of the logits layer

- Final loss function

$$\mathcal{L}(\boldsymbol{p}, \boldsymbol{X}) = \mathcal{L}^{label}(\boldsymbol{p}, \boldsymbol{X}, \boldsymbol{y}') + \mathcal{L}^{stat}(\boldsymbol{p}, \boldsymbol{X})$$

Theoretical Analysis of StatOpt

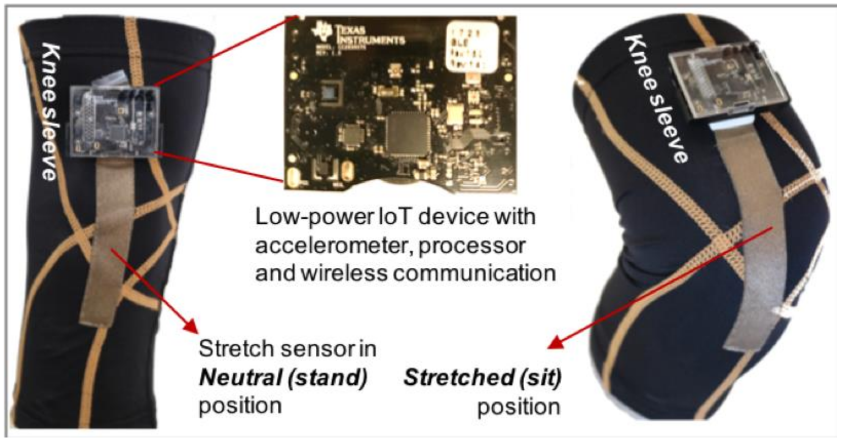
- We derive theoretical upper bounds for disturbances
- The bound ensures that classifier will provide *reliable classifications* if disturbances are within the bounds
  - Reliability certificates analogous to security certificates
- Euclidian distance-based certification is not suitable
- We propose to use Rényi divergence of a Gaussian distribution
- The certification provide the following guarantee,

$$\boldsymbol{F_\theta(X) = F_\theta(X+p), if\ \|p\| \leq E^{lim}}$$

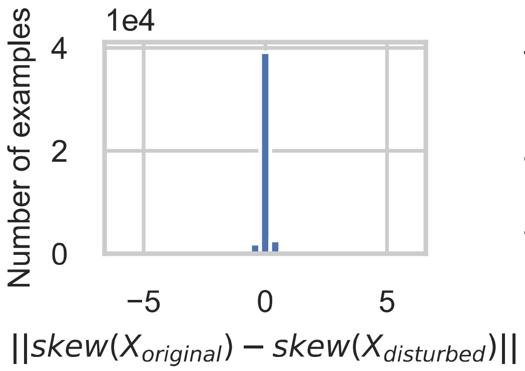
where $\boldsymbol{F_\theta}$ is the classifier and $\boldsymbol{E^{lim}}$ is the certification

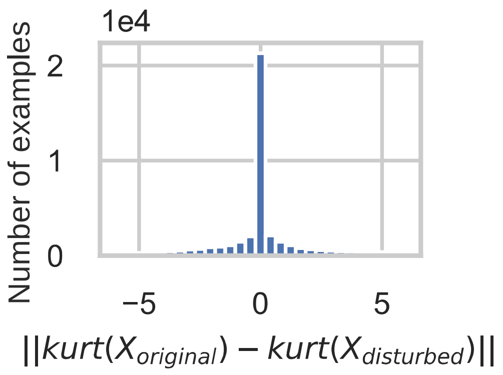
Experimental Setup

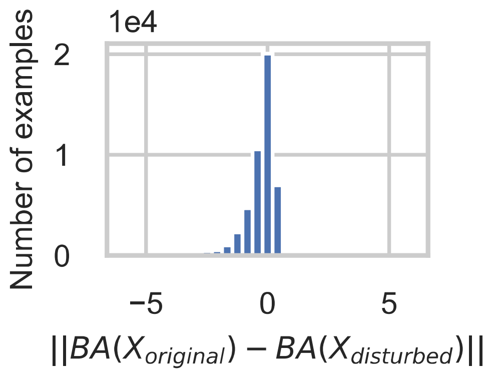
- Wearable Device
  - TI CC2650 MCU
  - ARM Cortex M4 processor
- Datasets
  - w-HAR HAR dataset
    - Accelerometer and stretch sensor data
    - 8 activities from 22 users
    - Orientation, amplitude, and sensor disturbance
  - WISDM
    - Accelerometer data from smartphone
    - 6 activities from 29 users
    - Orientation disturbance

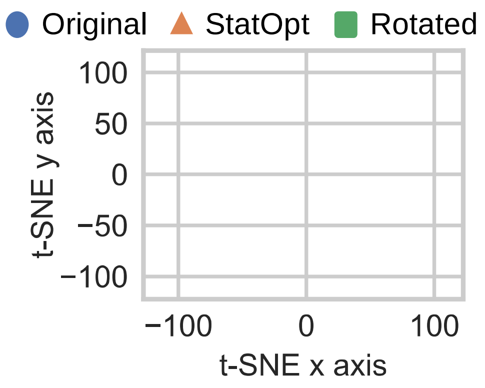

Knee sleeve
Knee sleeve
Low-power IoT device with accelerometer, processor and wireless communication
Stretch sensor in *Neutral (stand)* position
Stretched (sit) position

Accuracy of StatOpt Data Generation





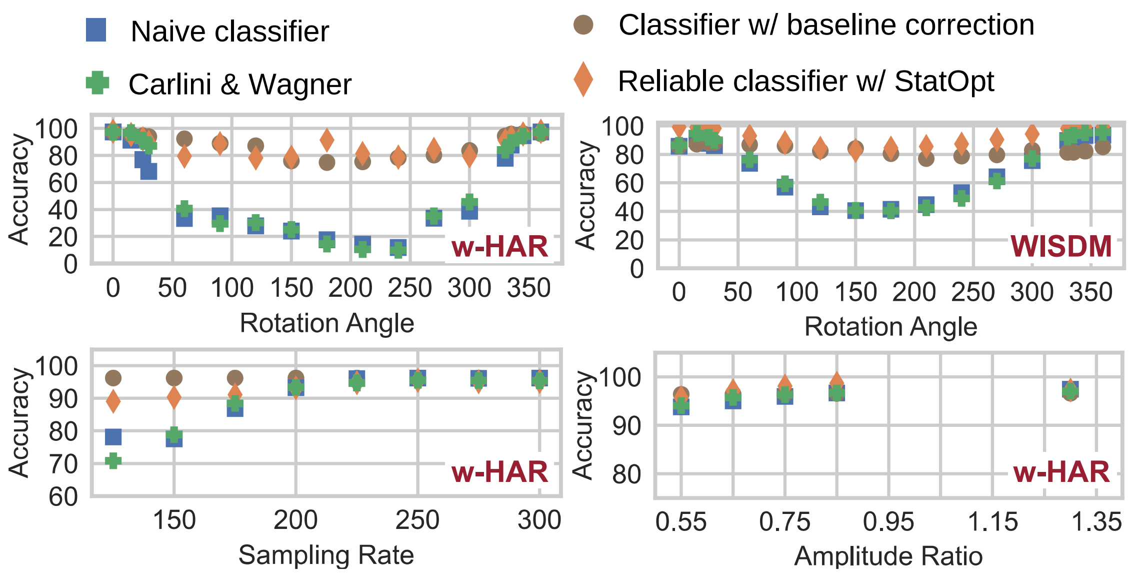




Statistical features closely match the original sensor data

t-SNE representations are overlapping for StatOpt data

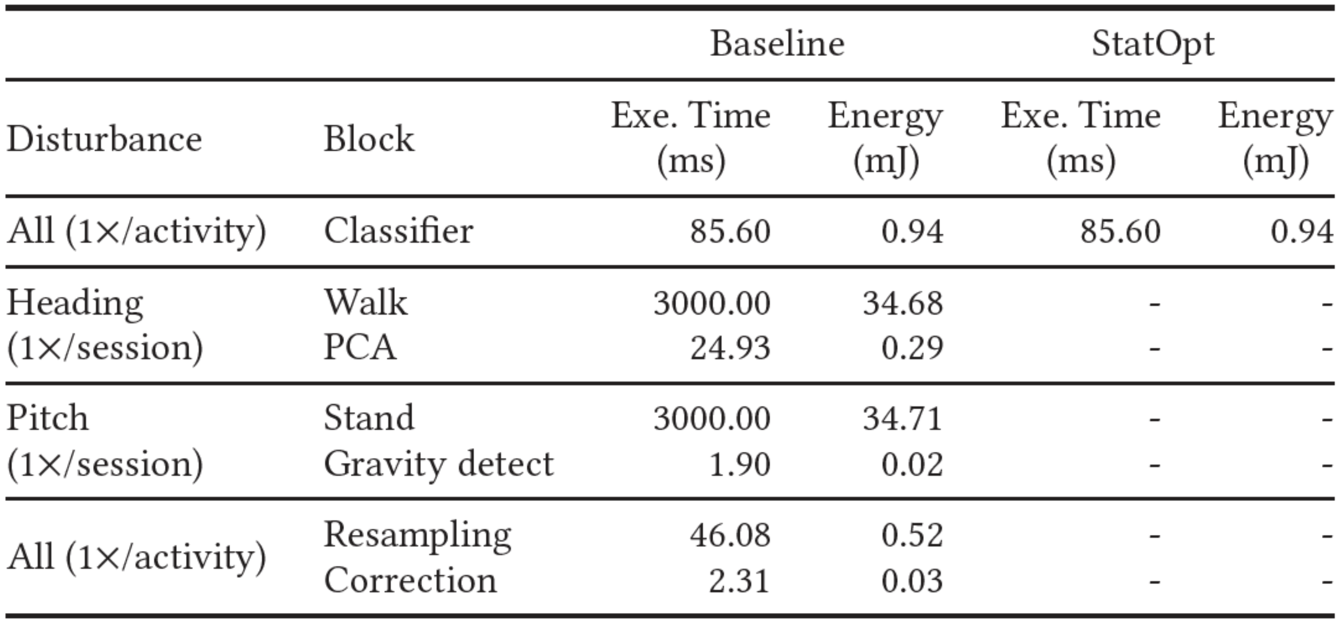
Classification Accuracy Comparison



- Reliable classifier has accuracy close to the no disturbance case
- C&W method fails to recover accuracy while the baseline has overhead

Implementation Overhead

- We characterize the implementation overhead on TI-CC2650
- StatOpt has no overhead for data recovery



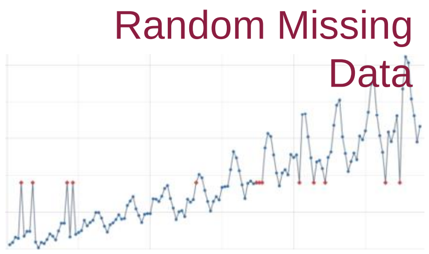
		Baseline		StatOpt	
Disturbance	Block	Exe. Time (ms)	Energy (mJ)	Exe. Time (ms)	Energy (mJ)
All (1×/activity)	Classifier	85.60	0.94	85.60	0.94
Heading (1×/session)	Walk	3000.00	34.68	-	-
	PCA	24.93	0.29	-	-
Pitch (1×/session)	Stand	3000.00	34.71	-	-
	Gravity detect	1.90	0.02	-	-
All (1×/activity)	Resampling	46.08	0.52	-	-
	Correction	2.31	0.03	-	-

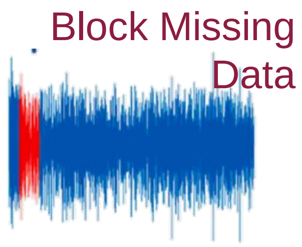
Outline

- Wearable applications and ML
- Sensor disturbances in wearable devices
- Missing data and solutions
  - Accuracy preserving imputation
  - Clustering-based imputation

Missing Data Overview

- Random Missing data
  - Isolated missing samples not clustered around any particular time instance.
  - Occurs due to limited communication bandwidth and buffer overflow in a sensor.
- Block Missing data
  - A sequence of samples are missing
  - Occurs when one or more sensors have to go into a low-power state

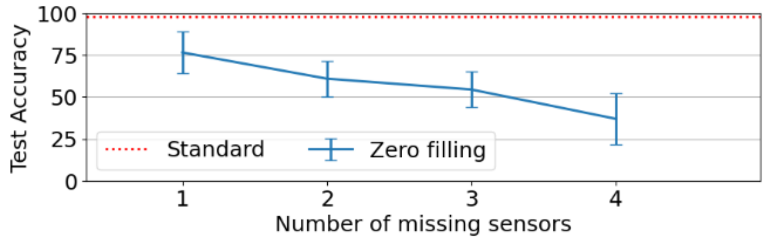




We showcase two complementary methods to impute data

Light-Weight ML Algorithm for Missing Data

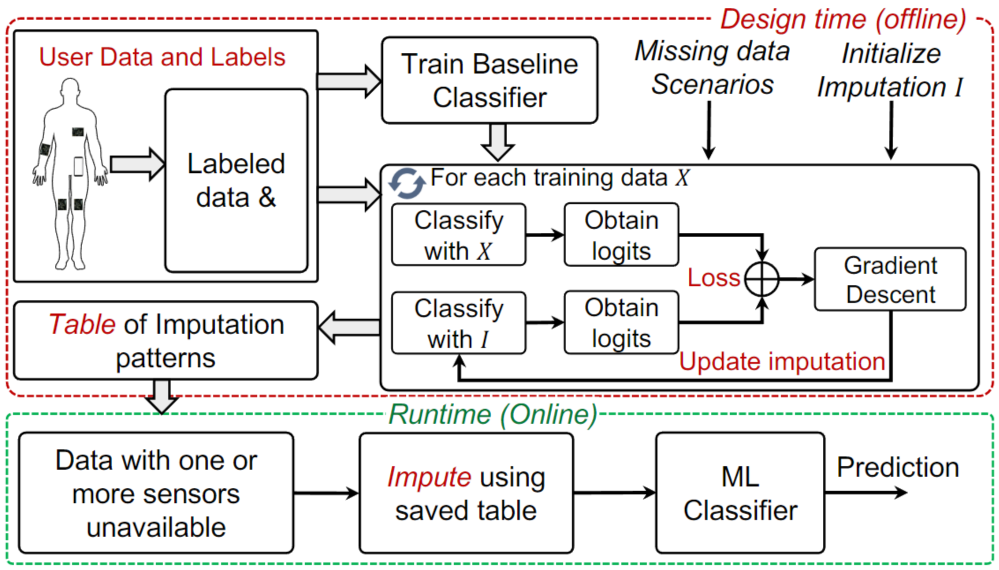
- Generative imputation networks recover raw data for classification
  - However, some applications do not need exact data recovery
  - Moreover, generative networks incur high overhead
- Trade-off between accuracy & overhead
- Maximize the accuracy without imputing the data exactly
  - Reduce memory overhead of imputation by avoiding generative networks



We show Accuracy-Preserving Imputation (AIM) for wearable apps

Accuracy-Preserving Imputation (AIM)

- AIM is based on two key insights
  - Exact data recovery is not needed if the application accuracy is preserved
  - Can train the ML model to be robust to small deviations from the exact data



AIM: Problem Setup

- Consider time-series sensor data $X \in R^{n \times T}$ data
  - n is the total number of sensor channels
  - T number of samples in each input window
- Class label for each window is y
- Subset of channels $\{\boldsymbol{j}\}_{0 \le j \le n}$ are missing at runtime for X
- Using zero inputs for missing data, we get sensor data as

$$\tilde{X}_{\{j\}} = \begin{cases} 0^T, & \text{if} \quad i \in \{j\} \\ X_i, & \text{if} \quad i \notin \{j\} \end{cases}$$

- Goal of AIM is to find imputation patterns for missing data

AIM Algorithm Setup

- AIM obtains a single imputation pattern for each missing data scenario
  - The pattern must maximize classification accuracy for application
  - We push the classifier to predict the correct labels from the available channels
- Write the imputation pattern as

$$\mathcal{J}_{\{j\}} = \begin{cases} \mathcal{J}_i, & \text{if } \; i \in \{j\} \\ X_i, & \text{if } \; i \notin \{j\} \end{cases} \qquad \text{and } F_\theta(X) \sim F_\theta(\mathcal{J}_{\{j\}})$$

- Imputation does not depend on observed sensor data
- Instead, we use a search algorithm to find best imputation patterns

AIM Algorithm: Imputation Pattern Search

- Given $\{j\}$: We find $\mathcal{I}_{\{j\}} s.t. \forall X, F_\theta(X) \approx F_\theta\left(\mathcal{I}_{\{j\}}\right)$
- Search objective

$$\min_{\mathcal{I}_{j\in\{j\}}} \mathcal{L}\left(Logits\left(F_\theta(X)\right), Logits\left(F_\theta\left(\mathcal{I}_{\{j\}}\right)\right)\right)$$

$\mathcal{L}$ *is the MSE* between *Logits* values *of* $F_\theta(X)$ *and* $F_\theta\left(\mathcal{I}_{\{j\}}\right)$

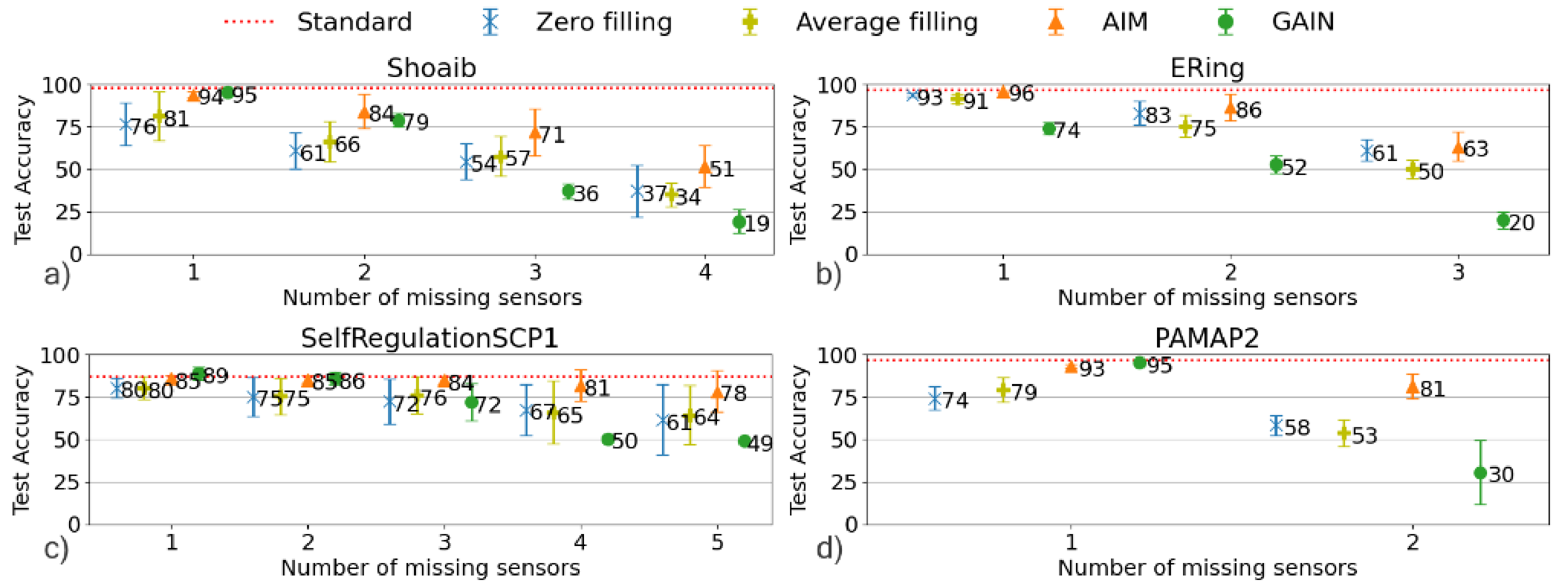
- Minimize objective to find imputation patterns
  - Classifier F_θ for original input X (no missingness) and input with imputation $\mathcal{I}_{\{j\}}$

Experimental Setup

- We employ the Odroid-XU3 board for evaluations
- We use four diverse wearable sensor-based timeseries datasets
  - Shoaib
  - PAMAP2
  - eRing
    - Ring to capture data along four dimensions
  - SR-SCP1
    - EEG data from six channels for a healthy subject.

- We use a 1-D convolutional neural network as classifier for all datasets
  - Adam optimizer over 20 epochs for training

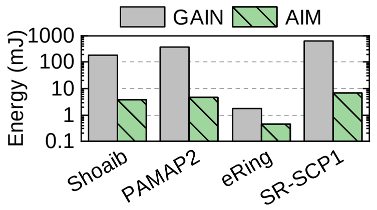
Classification Accuracy with AIM

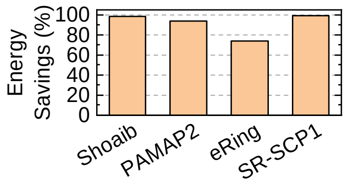


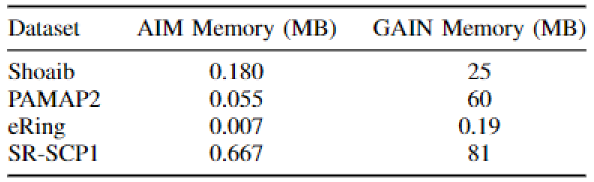
AIM achieves higher accuracy compared to baselines

Implementation Overhead

- Key advantage of AIM is lower energy and memory overhead
- AIM consumes less than 10 mJ per imputation
- Energy savings are close to 98% for all datasets except eRing
  - eRing has lower energy savings about 74% (has lower computation requirements for both GAIN and AIM)
- AIM can improve the battery life of wearable health monitoring devices by an order of magnitude







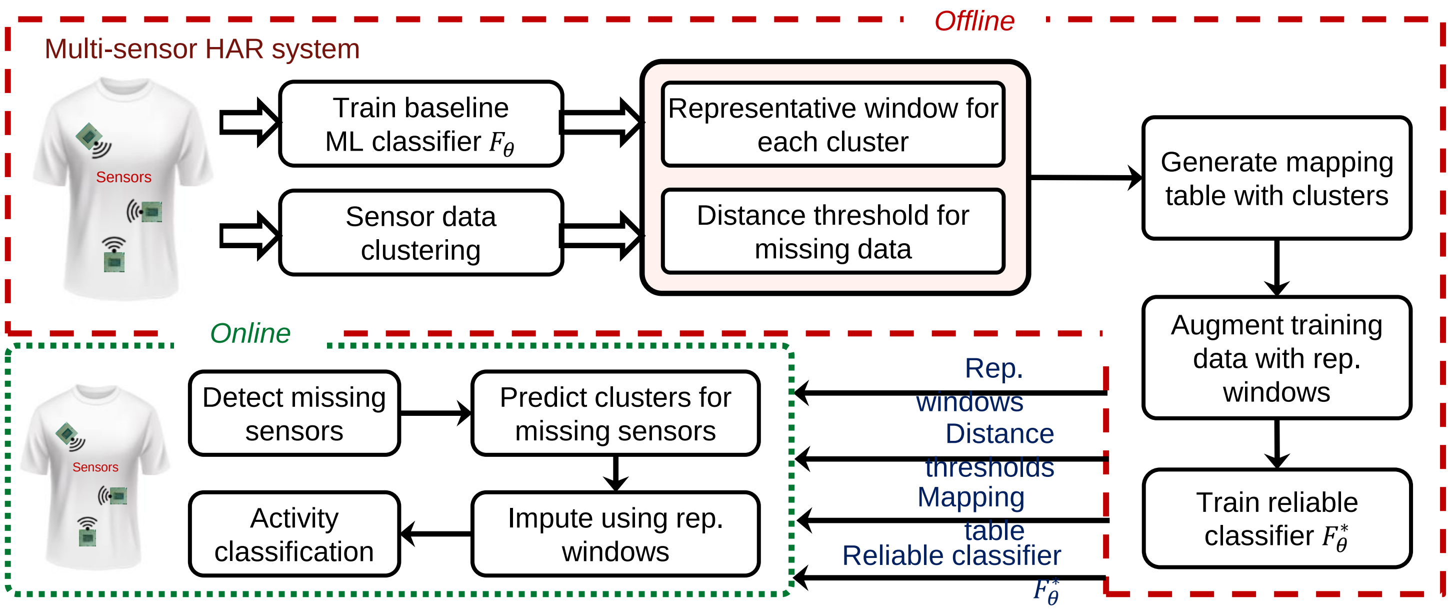
Dataset	AIM Memory (MB)	GAIN Memory (MB)
Shoaib	0.180	25
PAMAP2	0.055	60
eRing	0.007	0.19
SR-SCP1	0.667	81

Clustering-based Energy-Efficient Data Imputation

- AIM imputation maintains high classification accuracy
- However, it does not consider input data when obtaining imputation
- Actual data patterns for feedback besides the classification
  - Leads to better health outcomes through careful analysis

We show a clustering-based approach to account for input data in imputations

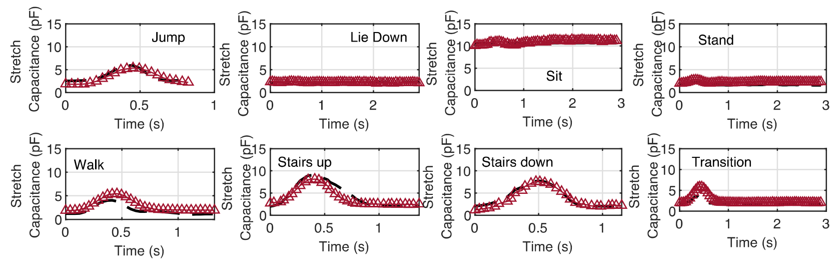
Clustering-based Energy-Efficient Data Imputation

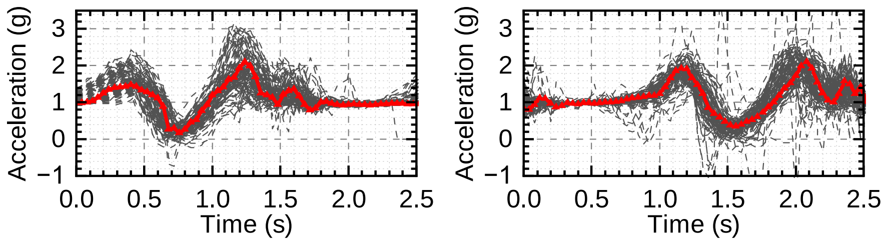

Offline
Multi-sensor HAR system
Sensors
Train baseline ML classifier F_θ
Sensor data clustering
Representative window for each cluster
Distance threshold for missing data
Generate mapping table with clusters
Augment training data with rep. windows
Train reliable classifier F_θ^*
Online
Sensors
Detect missing sensors
Predict clusters for missing sensors
Impute using rep. windows
Activity classification
Rep. windows
Distance thresholds
Mapping table
Reliable classifier F_θ^*

CIM enables efficient detection and imputation at runtime

Sensor Data Clustering

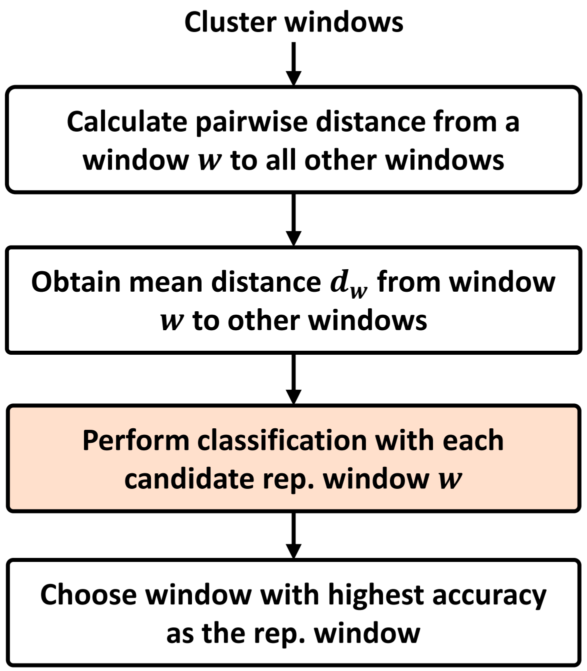
- Human activities are generally repeatable in nature
  - Activity patterns have variations across users and time
  - Repeatability of activities, in turn, leads to repeatability of sensor data patterns
- Different classes of activities have distinct sensor data patterns
- Based on this insight, we can cluster similar data patterns
- We employ k-means clustering
  - Input: Sensor data $\mathbb{X}$, No. of clusters
  - Output: Cluster centroids $\mathcal{C}$



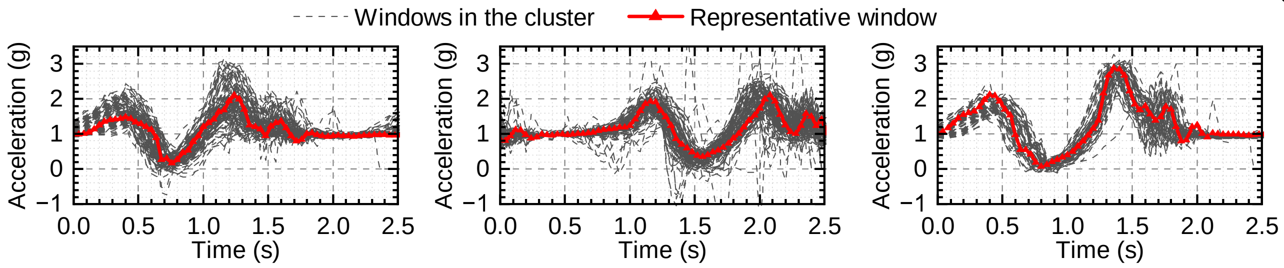


Representative Windows for Each Cluster

- Sensor data clusters are used for imputation
  - Infeasible to store all windows in a cluster
- Obtain a representative window for each cluster
- Representative window presents an *average* case Window that is closest to all other windows
  - Provides high classification accuracy for the cluster



Cluster windows
Calculate pairwise distance from a window w to all other windows
Obtain mean distance d_w from window w to other windows
Perform classification with each candidate rep. window w
Choose window with highest accuracy as the rep. window



Data Imputation with Sensor Clusters

- We use sensor data clusters to impute missing data at runtime
- Construct a mapping table to learn cluster mapping across sensors
  - Obtain unique combinations of clusters and record # occurrences

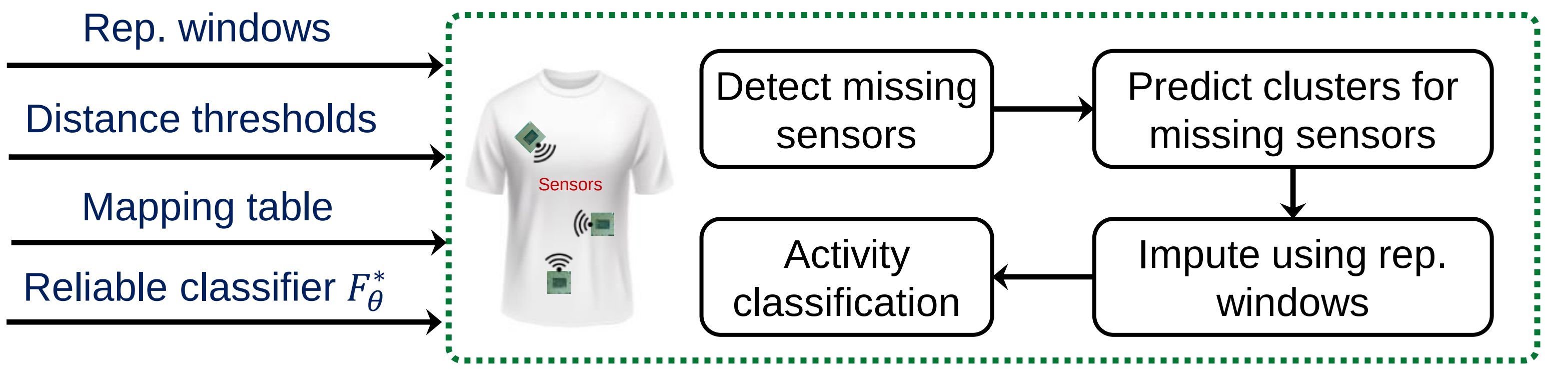
Sensor 1 cluster	Sensor 2 cluster	⋯	Sensor M cluster	Count
3	4		5	20
1	2		12	25
3	4		5	10

- Mapping table is used to predict cluster of missing sensor
- Use representative window of missing cluster as imputation

Low-overhead imputation by avoiding generative networks

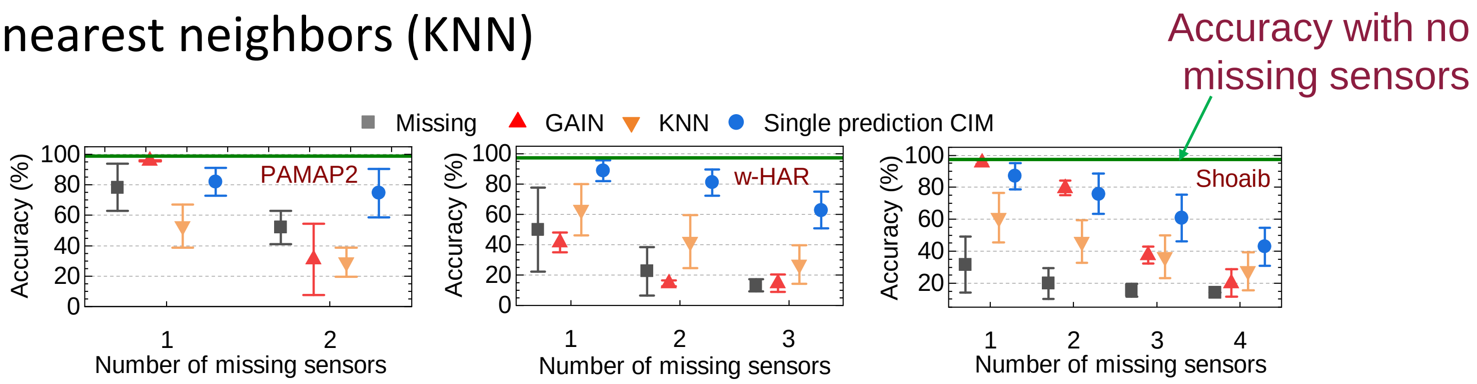
Summary of Imputation with CIM

CIM performs following steps to detect and impute missing data at runtime



Classification Accuracy with CIM

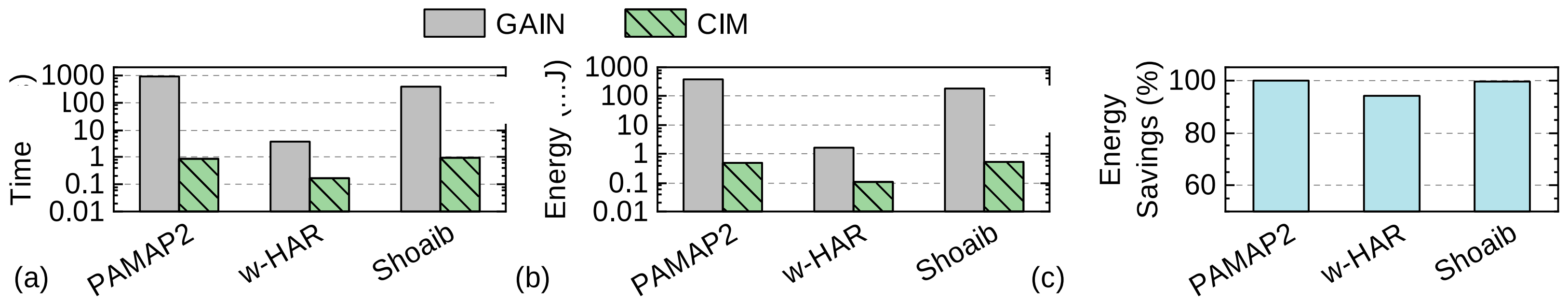
- Impute missing data across all missing data scenarios for three datasets
- Compare CIM against two baseline approaches
  - Generative adversarial imputation networks (GAIN)
  - K-nearest neighbors (KNN)



- CIM has higher accuracy than GAIN and KNN
  - GAIN loses accuracy with increasing number of missing sensors

Implementation Overhead

- We characterize the implementation overhead on Odroid-XU3
- Execution time overhead of CIM is lower than 1 ms for all three datasets
- CIM achieves close to 100% energy savings compared to GAIN

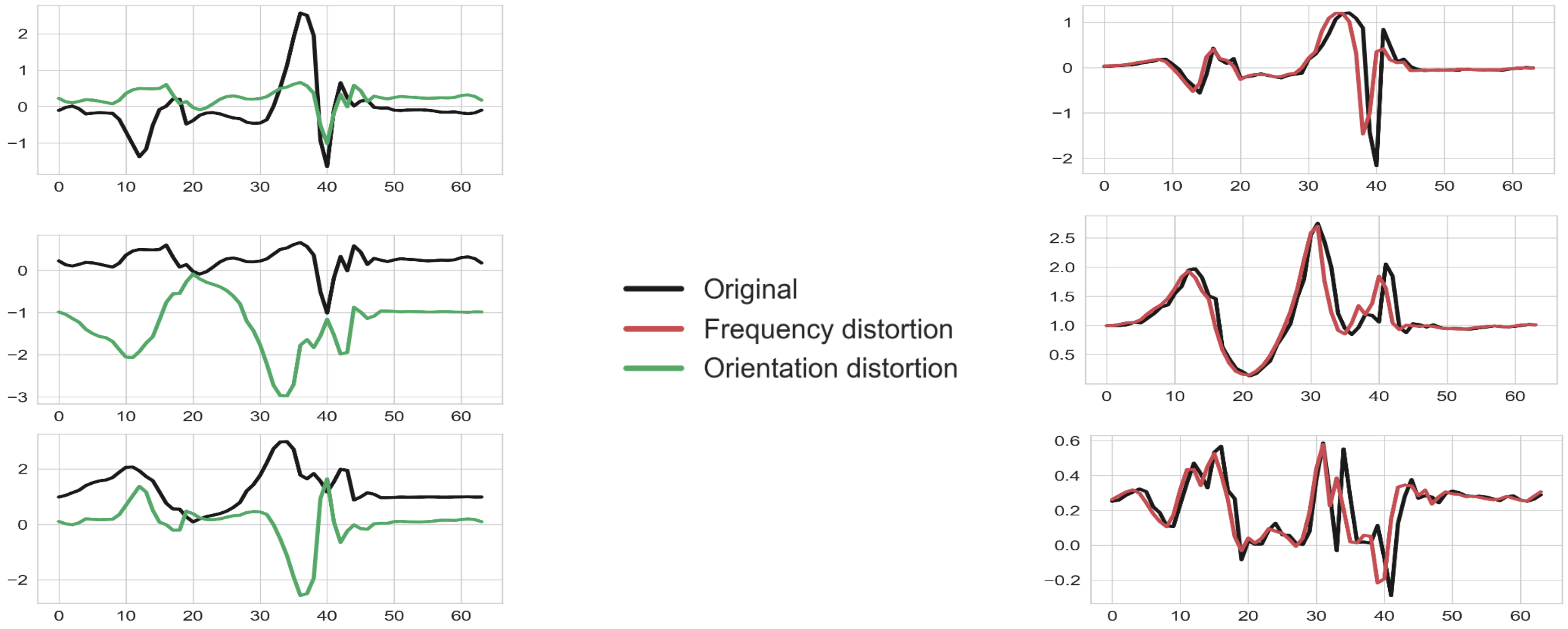


Take-home Messages

- Wearable devices are enabling interesting applications
- Sensor disturbances and missingness lower real-world performance
- We must account for these at runtime
- Key requirements to consider:
  - Application accuracy and performance
  - Similarity of data imputation or generation
  - Overhead in terms of energy, memory, and latency
- Trading-off accuracy and overhead is critical for device sustainability and user experience

Temporal adversarial robustness for time-series classification

Similarity Challenge



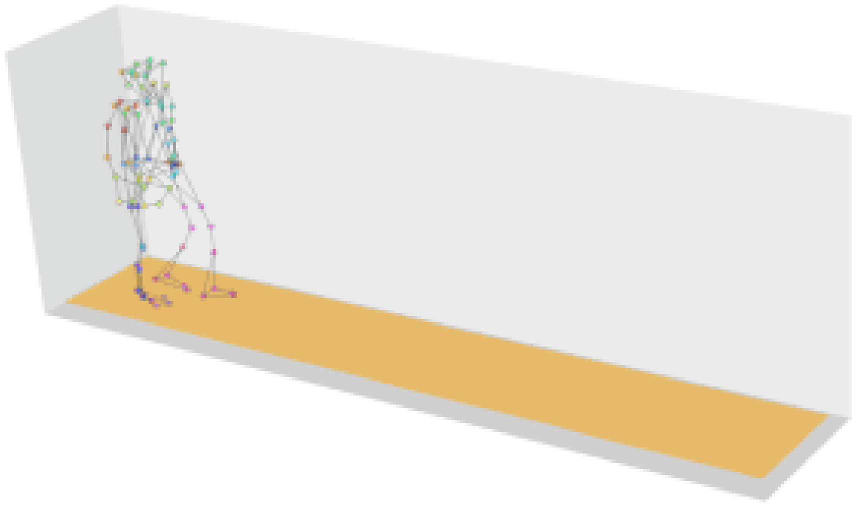
TSA-STAT can capture successfully

this similarity case

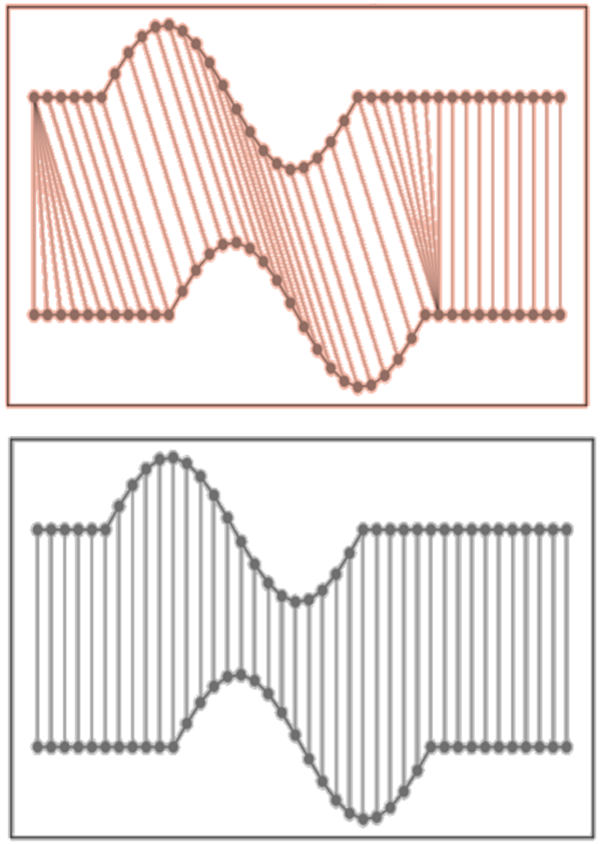
TSA-STAT is too complex for this perturbation case

Approach 2: Temporal behavior

- A similarity measure that accounts for shifts along temporal axis, scaling and frequency change

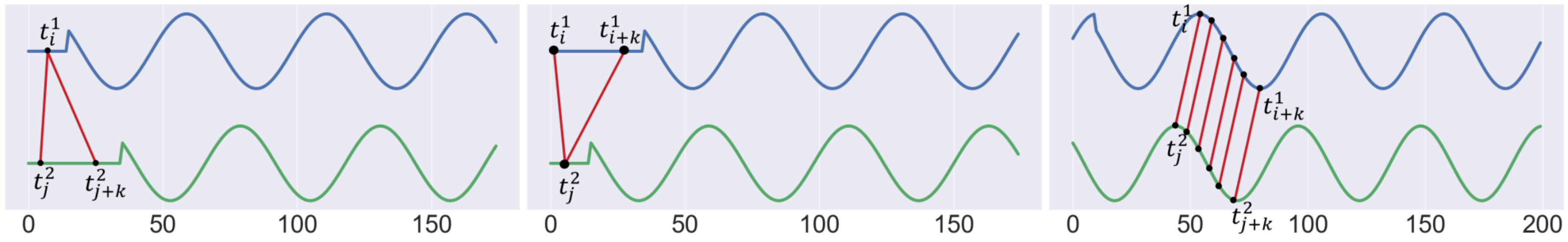


Two repetitions of the same walking sequence were recorded using a motion-capture system.[4]



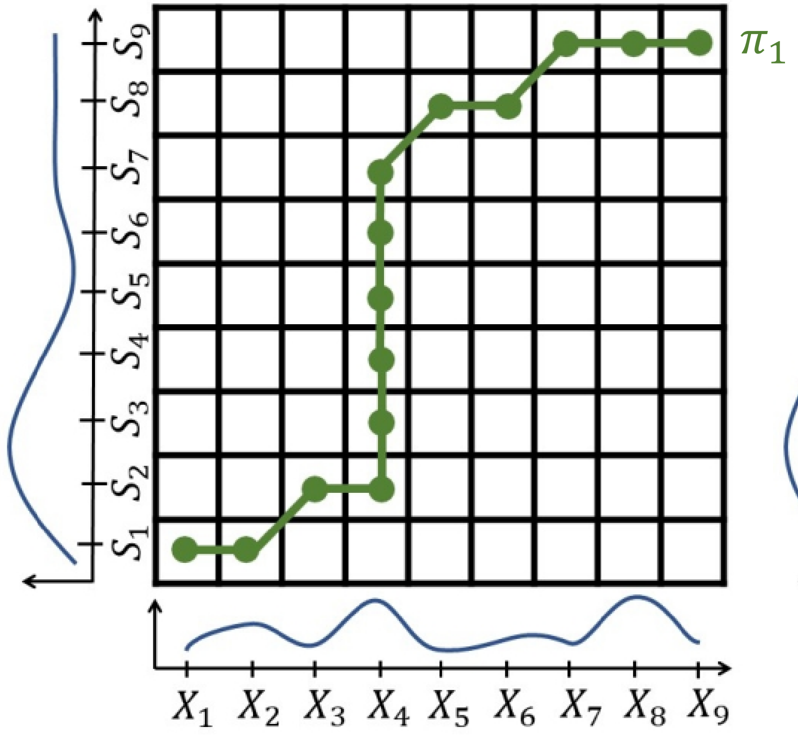
4: Olsen, Niels Lundtorp *et al*. "Simultaneous inference for misaligned multivariate functional data." Journal of the Royal Statistical Society Series C: Applied Statistics 67.5 (2018): 1147-1176.

Dynamic Time Warping

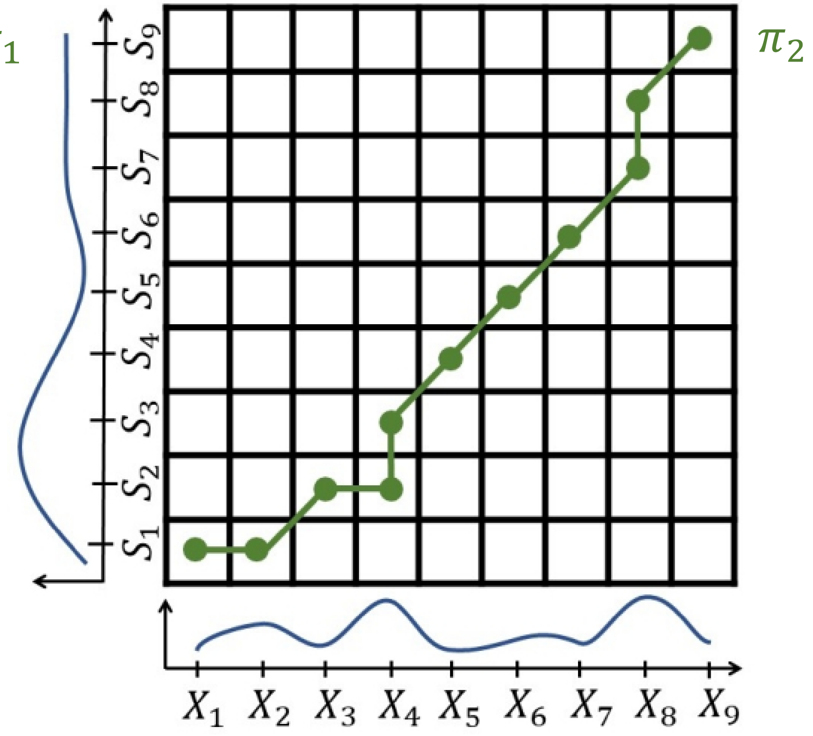


- We investigate Dynamic Time Warping approach for adversarial attacks.
- Dynamic Time Warping seeks for the optimal temporal alignment.
- A temporal alignment is a matching between time indexes $t_i$of the two time-series t^1 and t^2.

Dynamic Programming for DTW



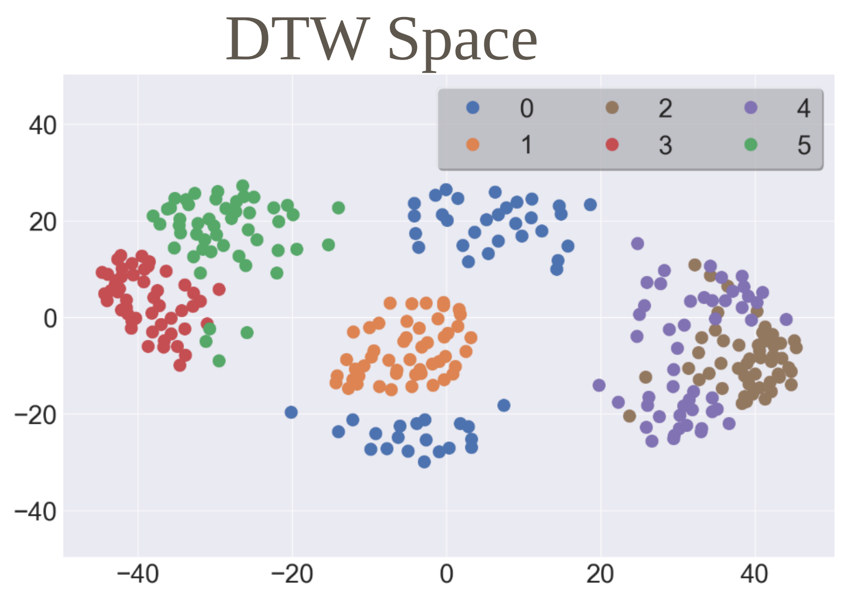
π_1
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9
X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 X_9

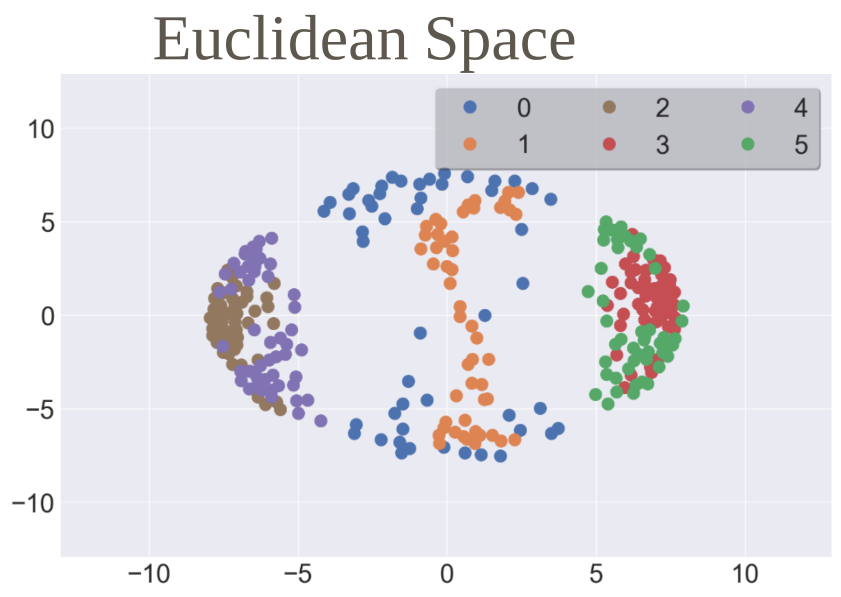


π_2
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9
X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 X_9

- $DTW(X,S) = \min_{\pi_k} \| X_i - S_i \|$
- Iterative – Quadratic complexity – Slow!!

DTW vs. Euclidean Space: Similarity Comparison





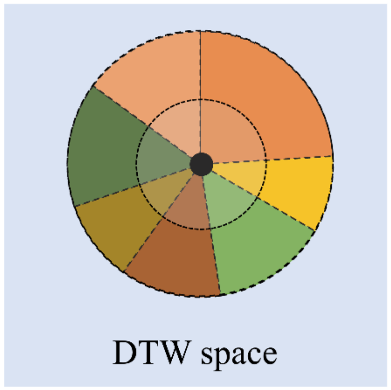
t-Distributed Stochastic Neighbor Embedding showing the empirical class distribution of real-world data examples and their similarity using DTW measure and Euclidean distance.

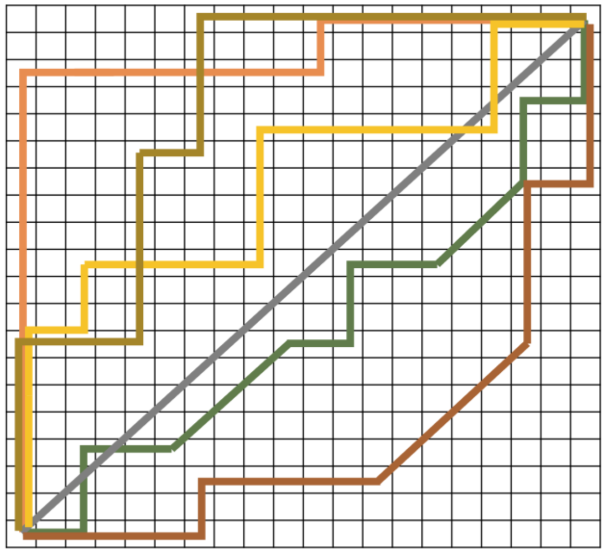
- DTW space exhibits better clustering for same-class data than Euclidean space

→ How to overcome the challenges of implementing DTW in the adversarial setting?

Naïve DTW Approach

- At each iteration:
    1. Compute the optimal DTW alignment
    2. Use the alignment in the gradient and create an adversarial example





- High computational cost
- Single adversarial example

Key Insight: Random Alignment Path Approximation

- We formalize the following theorem to devise an effective and efficient algorithm

Theorem. For a time-series X and a random alignment path P, the resulting adversarial example from DTW-AR is equivalent to using standard DTW computation (tight approximation).

Key Insight: Random Alignment Path Approximation

Algorithm 1 DTW-AR based Adversarial Algorithm

Input: time-series X; DNN classifier F_θ; target class-label y_{target}; learning rate η; maximum iterations MAX
Output: adversarial example X_{adv}

```
1: $P_{rand} \leftarrow$ random alignment path
2: Initialization: $X_{adv} \leftarrow X$
3: for $i$=1 to MAX do
4:     $\mathcal{L}(X_{adv}) \leftarrow \mathcal{L}^{label}(X_{adv}) + \mathcal{L}^{DTW}(X_{adv}, P_{rand})$
5:     Compute gradient $\nabla_{X_{adv}} \mathcal{L}(X_{adv})$
6:     Perform gradient descent step:
       $X_{adv} \leftarrow X_{adv} - \eta \times \nabla_{X_{adv}} \mathcal{L}(X_{adv})$
7: end for
8: return optimized adversarial example $X_{adv}$
```

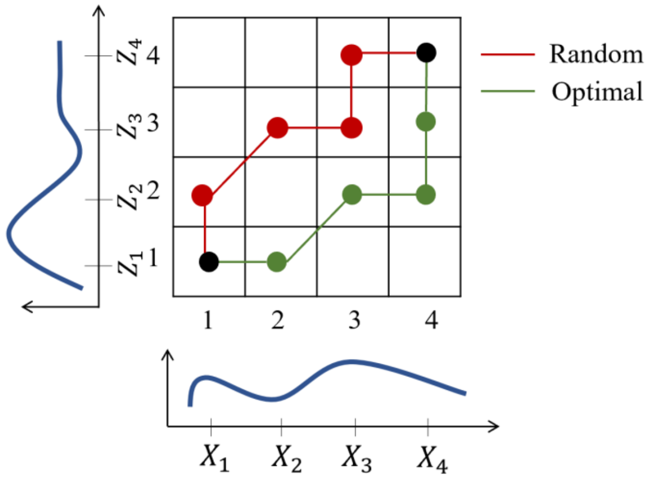
Theorem. For a time-series X and a random alignment path P, the resulting adversarial example from DTW-AR is equivalent to using standard DTW computation (tight approximation).

How?

Path-specific distance optimization

- We define a new metric *PathSim* as a similarity measure between two alignment paths *P1* and *P2* in the DTW cost matrix that satisfies the distance axioms

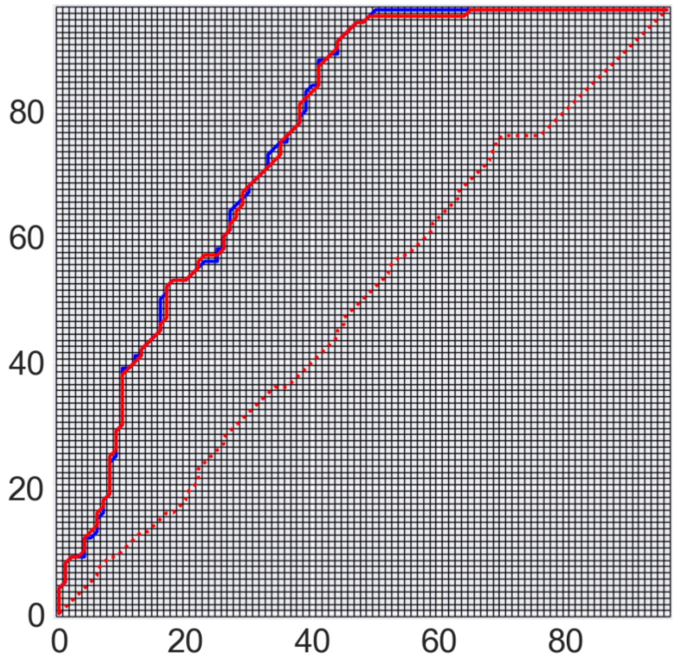
- We define $P_i = \{c_1^i, \ldots, c_{len(P_i)}^i\}$

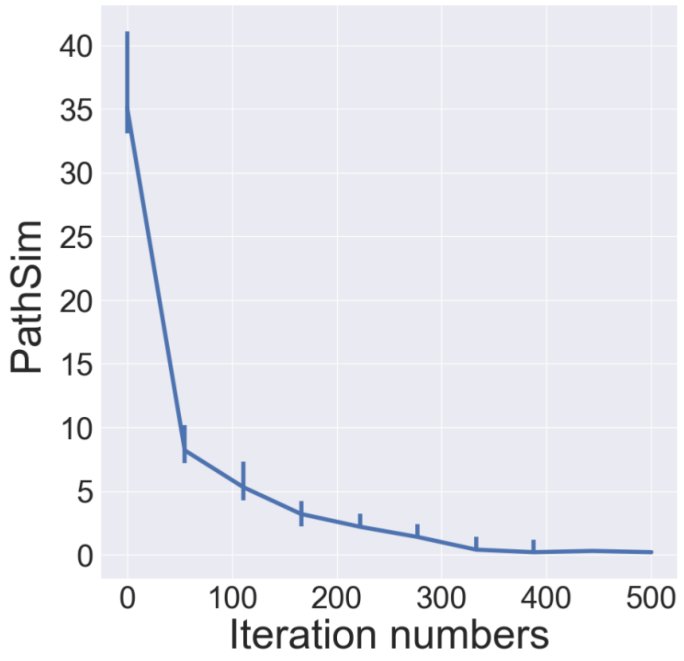


$$\mathtt{PathSim}\left(P_1, P_2\right) =$$

$$\frac{1}{2T}\left(\sum_{c_i^1} \min_{c_j^2} \|c_i^1 - c_j^2\|_1 + \sum_{c_i^2} \min_{c_j^1} \|c_i^2 - c_j^1\|_1\right)$$

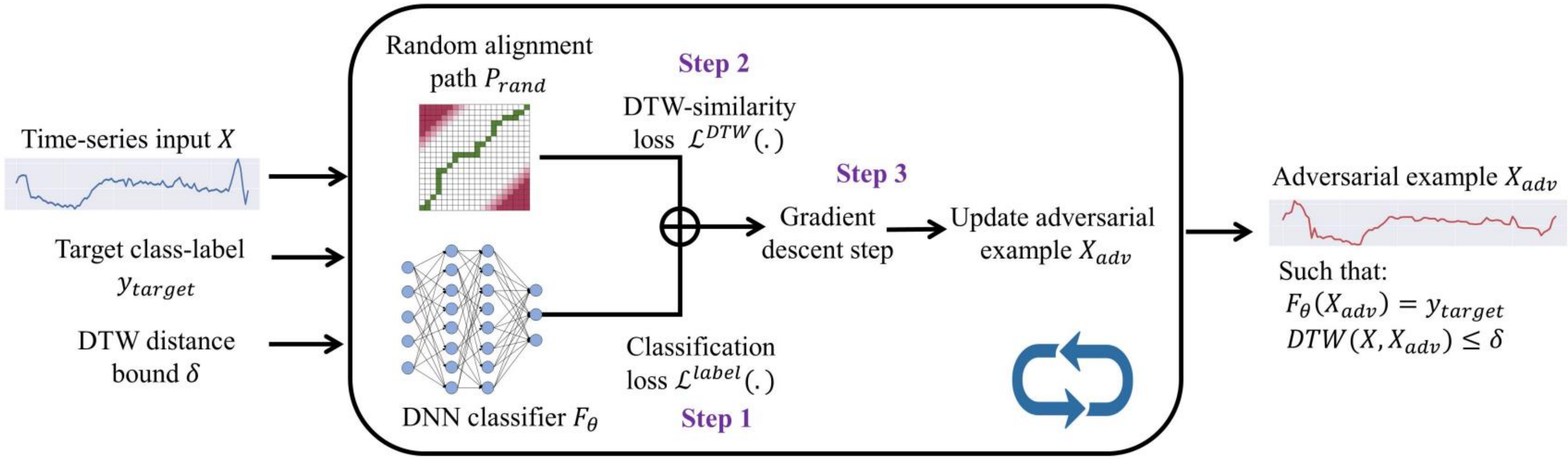
Path-specific distance optimization



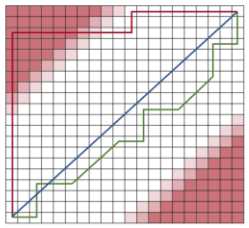


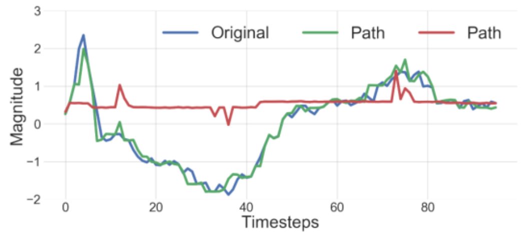
Example of the empirical convergence of the optimal alignment path between the optimized example and the original example at the start of the algorithm (dotted red path) and at the end (red path) to the given random alignment path (blue path).

DTW-AR Framework



Time-series input X
Target class-label y_{target}
DTW distance bound δ
Random alignment path P_{rand}
Step 2
DTW-similarity loss $\mathcal{L}^{DTW}(.)$
Step 3
Gradient descent step
Update adversarial example X_{adv}
DNN classifier F_θ
Classification loss $\mathcal{L}^{label}(.)$
Step 1
Adversarial example X_{adv}
Such that:
$F_\theta(X_{adv}) = y_{target}$
$DTW(X, X_{adv}) \leq \delta$

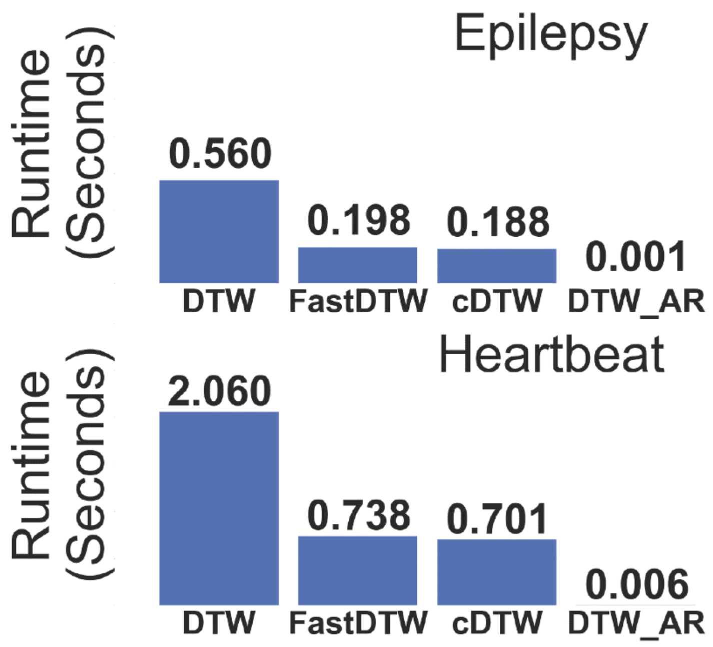




- The choice of the alignment range is important for better example generations

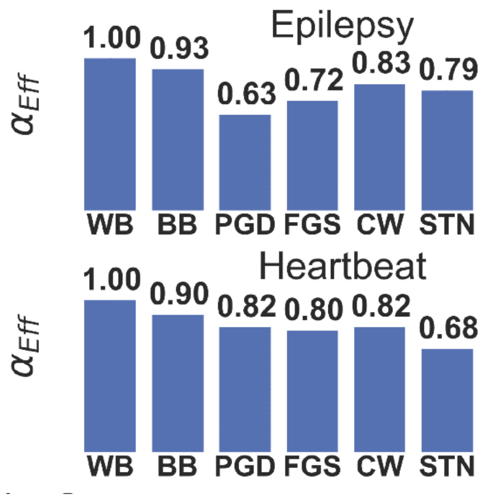
DTW-AR Results: Computational Efficiency

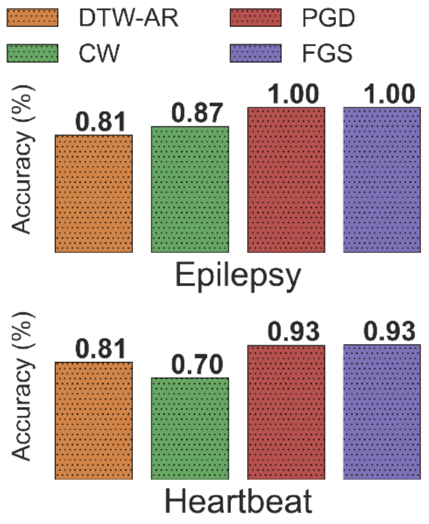
- Overall computational cost is significantly reduced using DTW-AR



Average runtime per iteration

DTW-AR Results: Effectiveness of Adversarial Examples





- DTW-AR is capable of fooling DNNs w/o existing adversarial training.
- DTW-AR is effective in predicting the original label of adversarial examples with high accuracy.

Explicit Min-Max Adversarial Training

All Alignment Path Approximation

- Instead of a single **optimal** alignment between x and x', we desire a measure that takes all possible alignments $\pi \in \mathrm{A}$ into consideration

$$k_{\mathrm{GAK}}(x, x') = \sum_{\pi \in \mathcal{A}} \exp\Big(-\frac{d_\pi(x, x')}{\nu}\Big)$$

$$d_\pi(x, x') = \sum_{i=1}^{|\pi|} \mathrm{dist}(x_{\pi_1(i)}, x'_{\pi_2(i)})$$

Global Alignment Kernel (GAK)

- Advantages of using GAK as an alignment-based similarity measure:

1. Differentiable
2. Positive definite
3. Coherent measure over all possible alignments
4. More general than optimal alignment measure (Dynamic Time Warping)

$$0 \leq D_{DTW}(x, x') - D_{\mathrm{GAK}}(x, x') \leq \nu \log(|\mathcal{A}|)$$

Proposed approach: Explicit training for robustness

- Explicit training for robustness

The inner maximization problem serves the role of an attacker whose goal is to find adversarial examples that achieves the largest loss.

$$\min_{w\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \boxed{\max_{a_i} \ell(f(x_i + a_i, w), y_i)}$$

$$\text{s.t. } d(x_i, x_i + a_i) \le \varepsilon$$

Proposed approach: Explicit training for robustness

- Explicit training for robustness

$$\boxed{\min_{w\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \boxed{\max_{a_i} \ell(f(x_i + a_i, w), y_i)}}$$

$$\text{s.t. } d(x_i, x_i + a_i) \leq \varepsilon$$

The inner maximization problem serves the role of an attacker whose goal is to find adversarial examples that achieves the largest loss.

The outer minimization problem serves the role of a defender whose goal is to find the optimal parameters of the deep model

Optimization Challenges

- GAK gradient estimation is computationally expensive
- Randomly sample a constant number of alignments $\pi \in A$ at each iteration to estimate the gradient
- Sampling paths leads to biased gradient estimate → SGDA does not hold
- No previous analysis on min-max optimization with compositional structure

Method: SCAGDA Optimization Algorithm

- Novel stochastic compositional alternating gradient descent ascent algorithm
- Solves a family of nonconvex-nonconcave min-max compositional problems

$$\min_{w} \max_{a_i} \frac{1}{n} \sum_{i=1}^{n} \phi_i(w, a_i) := f_i(w, a_i) - g\left(\tfrac{1}{m} \textstyle\sum_{j=1}^{m} h_{i,j}(a_i)\right)$$

$$\boxed{\min_{w \in \Theta} \max_{a_i} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i + a_i, w), y_i) + \lambda \log\left(k_{\mathrm{GAK}}(x_i, x_i + a_i)\right)}$$

RO-TS Instantiation of SCAGDA

$$\min_{w\in\Theta}\ \max_{a_i} \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i + a_i, w), y_i) + \lambda \log\left(k_{\text{GAK}}(x_i, x_i + a_i)\right)$$

∇_a for gradient ascent

∇_w for gradient descent

- Do not require all alignments for GAK estimation
  - ✓ Efficient training
  - ✓ Higher scalability

RO-TS Instantiation of SCAGDA

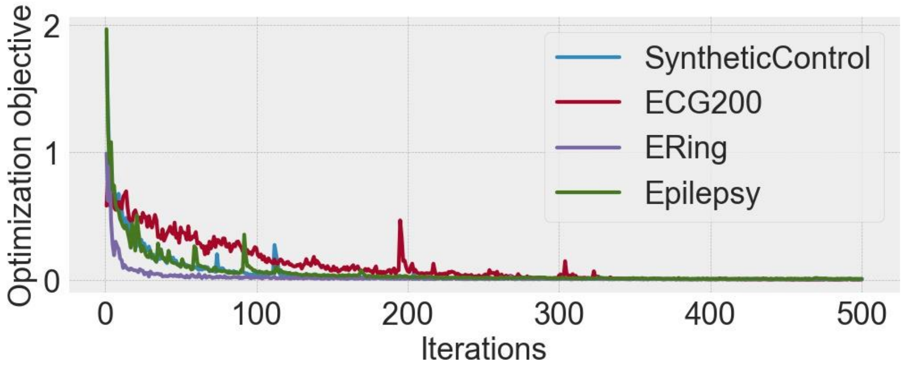
The training algorithm:

1. Sample a mini-batch of training data
2. Compute the stochastic gradient ∇_ω and update primal variable ω
3. Apply Moving Average (MA) to control the variance of the estimation of k_{GAK}
4. Estimate the stochastic gradient ∇_a over a random subset of alignments of k_{GAK}
5. Update dual variable a using gradient ascent

$$\min_{w \in \Theta} \max_{a_i} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i + a_i, w), y_i) + \lambda \log\left(k_{\mathrm{GAK}}(x_i, x_i + a_i)\right)$$

RO-TS Theoretical Results

- Theorem #1: SCAGDA converges in $O(\frac{1}{\epsilon^2})$ iterations to achieve ϵ-primal gap

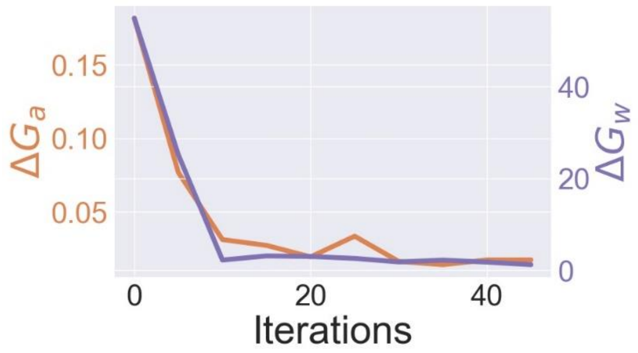


Empirical convergence of ROTS algorithm.

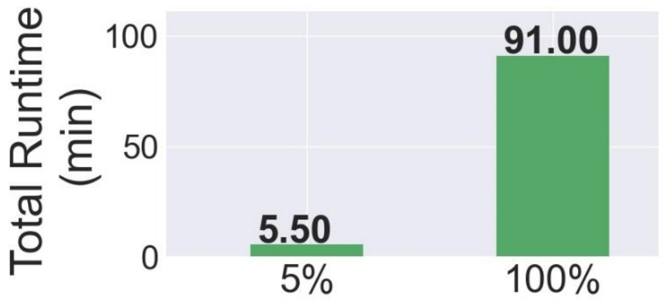
- The empirical results match the proposed theory

RO-TS Theoretical Results

- Theorem #2: The approximation error of gradient computation with sampled alignment paths reduces over iterations and becomes tight when SCAGDA converges



The accuracy gap in the gradients over weights G_w and over perturbations G_a using 5% vs. all of the alignments

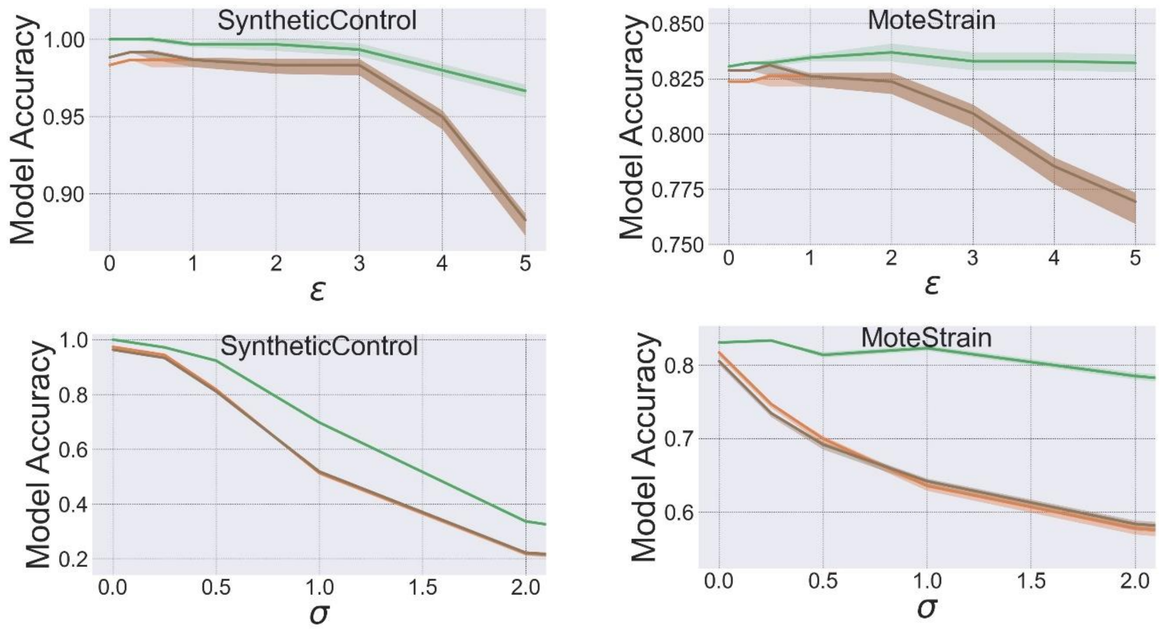


Comparison of the computational runtime between both settings

- The empirical results match the proposed theory

RO-TS Training Effectiveness – Setting #1

RO-TS vs. Adversarial training



Comparison of

🟩 RO-TS algorithm

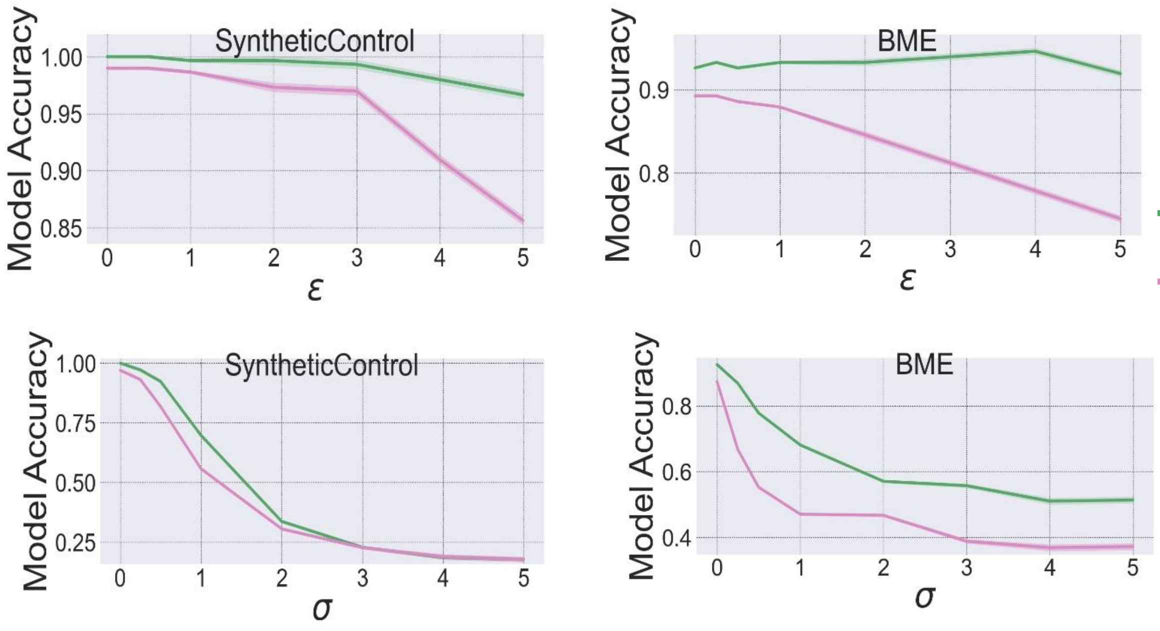
🟧 Fast Gradient Sign

🟫 Projected Gradient Descent

using Gaussian perturbation σ and adversarial perturbation ε on input

RO-TS Training Effectiveness – Setting #2

RO-TS: GAK vs. l_2



Comparison of

▬ RO-TS algorithm

▬ l_2

using Gaussian perturbation σ and adversarial perturbation ε on input

Robust Multivariate Time-Series Forecasting: Adversarial Attacks and Defense Strategies

Based on a joint work with
Linbo Liu[1], Youngsuk Park[1], Hilaf Hasson[2] and Luke Huan[1]

https://openreview.net/forum?id=ctmLBs8lITa

[1]AWS AI Labs
[2]Intuit

OUTLINE

- Adversarial Attack in Forecasting
- New Attack in Multi-variate Time-Series Settings
  - Deterministic Attack
  - Probabilistic Attack
- Defense Strategies
  - Provable Defense
  - Minimax Defense
- Empirical Studies
  - Dataset & Metric
  - Experiment Setup & Results

OUTLINE

- Adversarial Attack in Forecasting
- New Attack in Multi-variate Time-Series Settings
  - Deterministic Attack
  - Probabilistic Attack
- Defense Strategies
  - Provable Defense
  - Minimax Defense
- Empirical Studies
  - Dataset & Metric
  - Experiment Setup & Results

Adversarial Noises

- Deep Learning (DL) models are brittle against adversarial noises

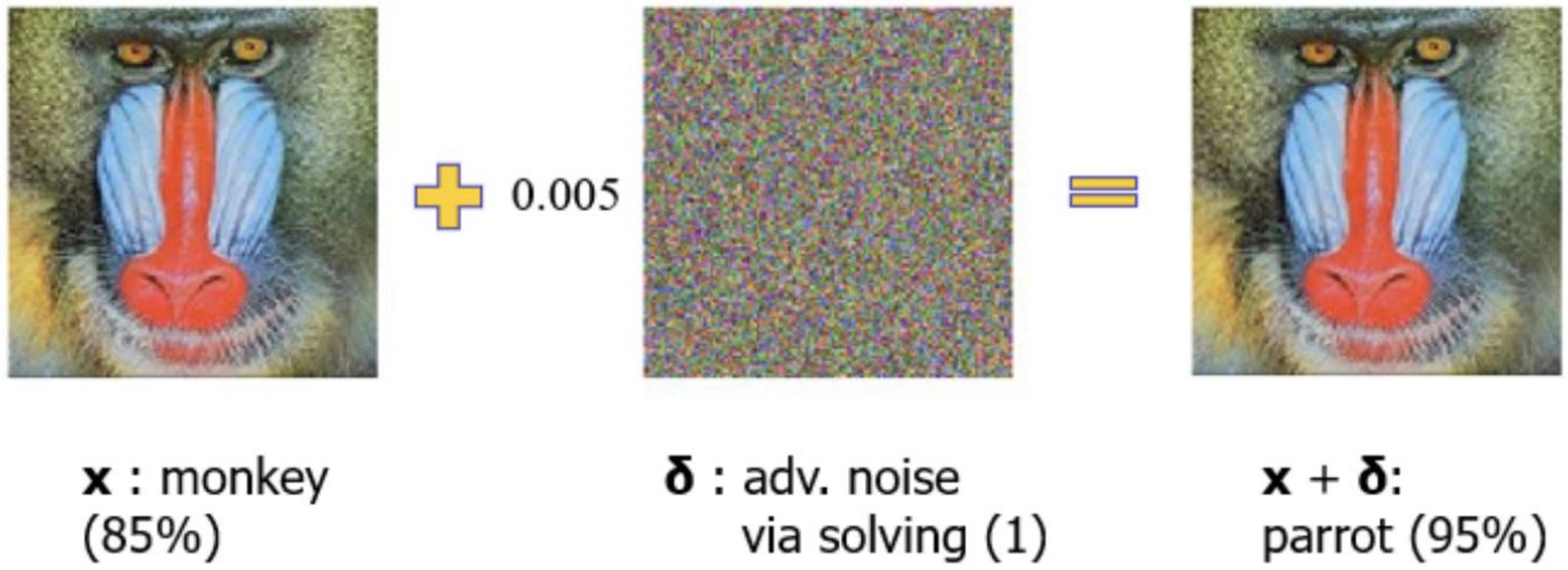
  [Szegedy et al., 2013, Goodfellow et al., 2014]

- Misleading Classification with Adversarial Noises:

$$\begin{aligned}\boldsymbol{\delta}^* &= \arg\min_{\boldsymbol{\delta}} \|\boldsymbol{\delta}\| \\ \text{s.t} \quad g(\mathbf{x}+\boldsymbol{\delta}) &= c\end{aligned}$$

- Notations:
  - g(**x**) is a trained classifier; **x** is a given input to be attacked
  - c != class(**x**) is the target of the attack

Example: Classification


0.005
$\mathbf{x}$: monkey (85%)
$\boldsymbol{\delta}$: adv. noise via solving (1)
$\mathbf{x} + \boldsymbol{\delta}$: parrot (95%)

Adversarial noise can mislead the classification outcome

Regression

- **Regression:** No discrete class, continuous output
- **A (slight) reformulation:** [Dang-Nhu et al., 2020]

$$\begin{aligned} \boldsymbol{\delta}^* &= \arg\min_{\boldsymbol{\delta}} \; \phi(g(\mathbf{x}+\boldsymbol{\delta}), z) \\ \text{s.t} & \; \boxed{\|\boldsymbol{\delta}\| \le \epsilon} \end{aligned}$$

constrains the noise to be human-imperceptible

- **Notations:**
  - g(**x**) is the trained regressor; **x** is a given input to be attacked
  - z is the target attack

Regression

- **Regression:** No discrete class, continuous output
- **A (slight) reformulation:** [Dang-Nhu et al., 2020]

$$\begin{aligned}\boldsymbol{\delta}^* &= \arg\min_{\boldsymbol{\delta}} \boxed{\phi\left(g(\mathbf{x}+\boldsymbol{\delta}), z\right)} \\ \text{s.t} \quad \|\boldsymbol{\delta}\| &\leq \epsilon\end{aligned}$$

distance function
e.g., L_2 distance

- **Notations:**
  - g(**x**) is the trained regressor; **x** is a given input to be attacked
  - z is the target attack

Probabilistic Forecasting

- Given $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ and prediction horizon $h > 0$

  compute $p(\mathbf{z} \mid \mathbf{x})$ where $\mathbf{z} = (x_{T+1}, x_{T+2}, \ldots, x_{T+h})$

- **Probabilistic Auto-Regressive Model:**

$$p(\mathbf{z} \mid \mathbf{x}) = \prod_{t=\tau}^{h-1} q_\theta(x_{t+1} \mid x_1, \ldots, x_t)$$

where θ is a (deep) neural net parameterization

Probabilistic Forecasting: DeepAR [Salinas et al., 2020]

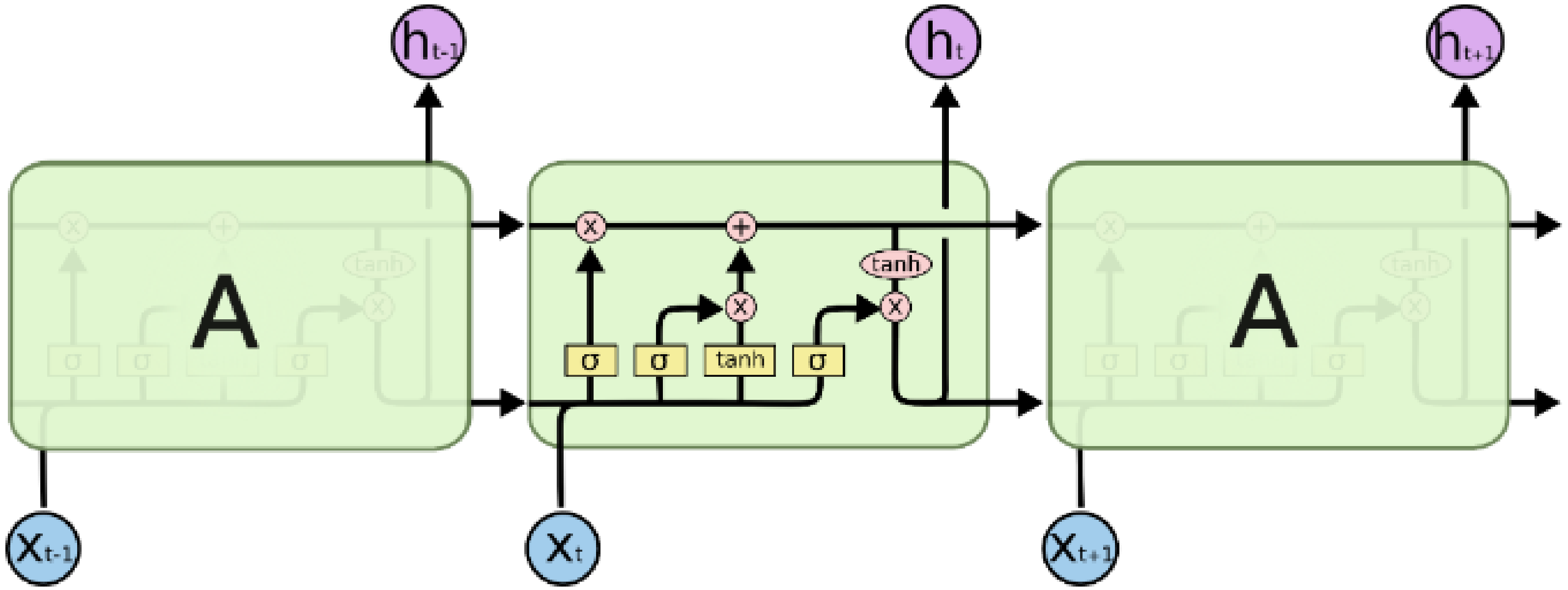
- Part of the Amazon SageMaker toolkit
- DeepAR parameterization

$$\begin{aligned} q_\theta(x_t \mid x_1, \ldots, x_{t-1}) &= \mathbb{N}\Big(x_t \mid \mu_\theta(\mathbf{h}_t), \sigma_\theta^2(\mathbf{h}_t)\Big) \\ &\triangleq \frac{1}{\sigma_\theta(\mathbf{h}_t)\sqrt{2\pi}} \cdot \exp\left[-\frac{1}{2}\left(\frac{x_t - \mu_\theta(\mathbf{h}_t)}{\sigma_\theta(\mathbf{h}_t)}\right)^2\right] \end{aligned}$$

where **h** is implemented by a long short-term memory network [Hochreiter & Schmidhuber, 1997]

parameterized by θ: $\quad \mathbf{h}_t = a(\mathbf{h}_{t-1}, x_t; \theta)$

Recurrent Network with LSTM cell



Fitting DeepAR

- Maximizing the probability of observing training data

$\mathbf{x} = (x_1, x_2, \dots, x_T)$

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \prod_{t=1}^{\tau} \mathbb{N}\left(x_t \mid \mu_\theta(\mathbf{h}_t), \sigma_\theta^2(\mathbf{h}_t)\right) \\ &= \arg\min_{\theta} \sum_{t=1}^{\tau} \left[\log \sigma_\theta(\mathbf{h}_t) + \frac{1}{2}\left(\frac{x_t - \mu_\theta(\mathbf{h}_t)}{\sigma_\theta(\mathbf{h}_t)}\right)^2\right]\end{aligned}$$

How do we attack this model?

Attacking Probabilistic Forecasting Model

- Downstream decision-making depends on a statistic

  s(**z**) of $\mathbf{z} \sim q_{\theta^*}(\mathbf{z} \mid \mathbf{x})$ where **z** = $(x_{T+1}, x_{T+2}, \ldots, x_{T+h})$

  For example, s(**z**) = $(x_{T+h} / x_T) - 1$
  i.e., gain of investing 1$ at time T & selling at time T + h

  Market manipulation: Artificially influence stock price to make profit [Allen & Gale, 1992; Diaz et al., 2011]

  i.e., perturbing x to influence s(z),
  triggering reaction of others

Attacking Probabilistic Forecasting Model

- Finding perturbation such that the forecast s(**z**) is as close as possible to a manipulation target $\mathbf{t}_{\text{adv}}$

  [Dang-Nhu et al., 2020; Yoon et al., 2022]

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}} \left\{ F(\boldsymbol{\delta}) \triangleq \left\| \mathbb{E}_{\mathbf{z}\sim q_\theta(\mathbf{z}|\mathbf{x}+\boldsymbol{\delta})}\left[s(\mathbf{z}) \right] - \mathbf{t}_{\text{adv}} \right\|^2 \right\}$$
$$\text{s.t} \quad \boxed{\|\boldsymbol{\delta}\| \leq \epsilon}$$

constrains the noise to be less visible

Attacking Multivariate Forecasting Model

- Previous work shows feasible attacks in univariate time-series (TS) setting

- Are there different attacks in multivariate time-series (MTS) settings?

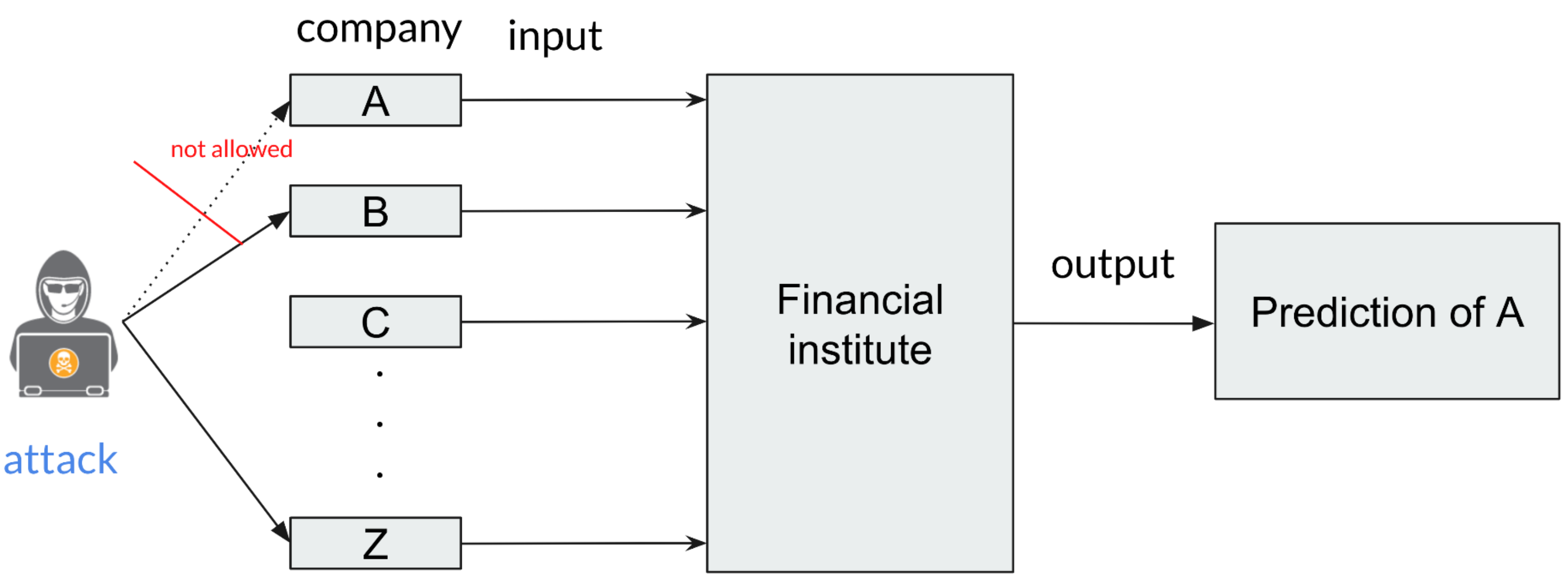
Attacking Multivariate Forecasting Model

- Are there different attacks in multivariate time-series (MTS) settings?

- More stealth attack patterns:
  - Attack one TS by perturbing observations of other TS (indirect attack)
  - Attack one TS by perturbing a small subset of other TS (sparse attack)

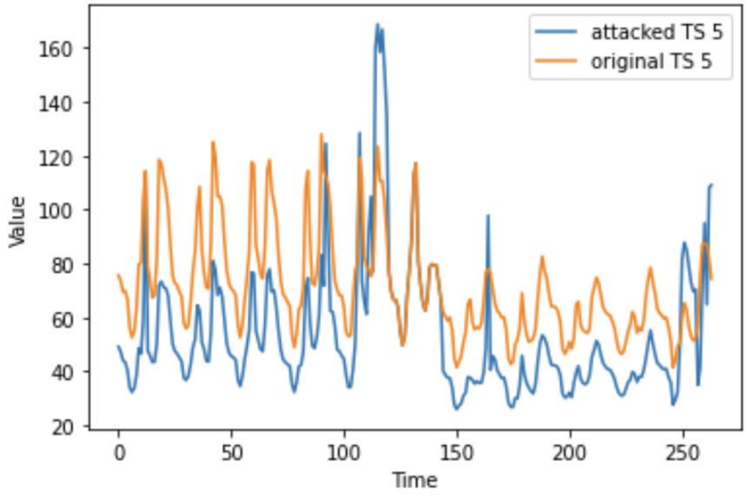
Example

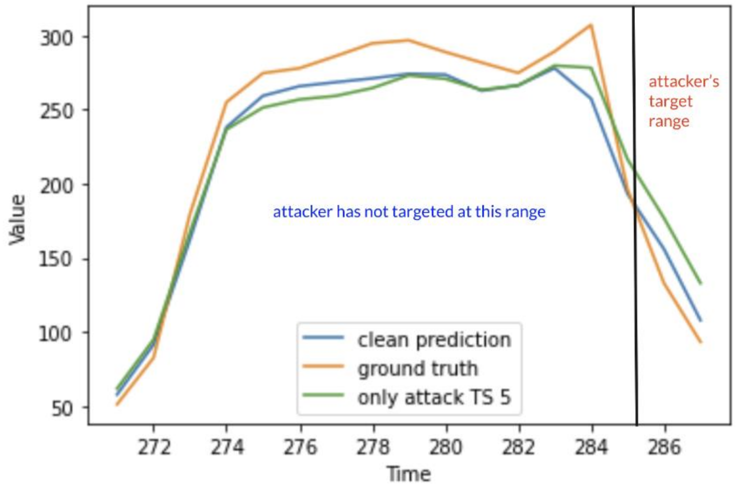
- The adversary is only able to select a few stocks to manipulate
- To make the attack stealth, stock A must not be manipulated



Simulated Experiment

- 10-dim DeepVAR [Salinas et al., 2020] to model 10 TS
- Attack TS 5 to mislead the prediction of TS 1

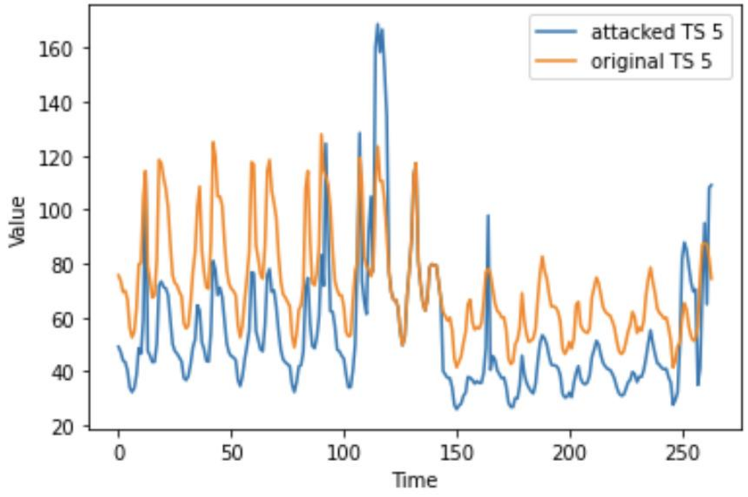


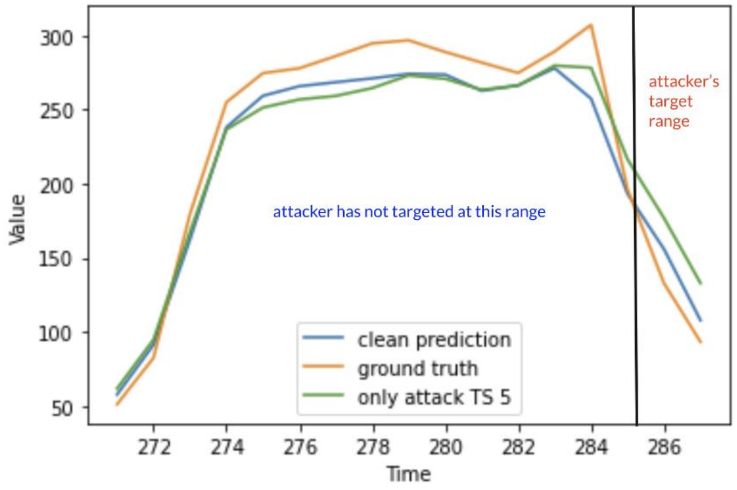


- Left: plot of authentic (orange) and perturbed (blue) TS 5
- No attack on TS 1 to 4 & TS 6 to 10

Simulated Experiment

- 10-dim DeepVAR [Salinas et al., 2020] to model 10 TS
- Attack TS 5 to mislead the prediction of TS 1





- Right: plot of prediction of TS 1
- Attacking TS 5 can mislead the prediction of TS 1

Tutorial Focus

- Two new attack patterns to MTS forecasting:
  - Deterministic Attack
  - Probabilistic Attack

- Two defense strategies:
  - Provable Defense via generalized random smoothing
  - Minimax Defense via empirical attack simulation

OUTLINE

- Adversarial Attack in Forecasting
- New Attack in Multi-variate Time-Series Settings
  - Deterministic Attack
  - Probabilistic Attack
- Defense Strategies
  - Provable Defense
  - Minimax Defense
- Empirical Studies
  - Dataset & Metric
  - Experiment Setup & Results

Adversarial Noise Revisitation

- Previous noise characterization:

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}} \left\{ F(\boldsymbol{\delta}) \triangleq \left\| \mathbb{E}_{\mathbf{z}\sim q_\theta(\mathbf{z}|\mathbf{x}+\boldsymbol{\delta})}\left[s(\mathbf{z})\right] - \mathbf{t}_{\mathrm{adv}} \right\|^2 \right\}$$

$$\text{s.t} \quad \|\boldsymbol{\delta}\| \leq \epsilon$$

- Low-energy noise, but not necessarily sparse & indirect
- How to ensure sparsity, indirectness & low energy

Sparse, Indirect & Low Energy

Let $\mathcal{I} \subset [d]$ denotes the attack targets (i.e., stock indices)

$$\boldsymbol{\delta}^* \quad = \quad \arg \min_{\boldsymbol{\delta} \in \mathbb{R}^{\tau \times d}} \left\{ F(\boldsymbol{\delta}) \ \triangleq \ \left\| \mathbb{E}_{\mathbf{z} \sim q_\theta(\mathbf{z}|\mathbf{x}+\boldsymbol{\delta})} \left[s(\mathbf{z}) \right] \ - \ \mathbf{t}_{\mathrm{adv}} \right\|^2 \right\}$$

$$\text{s.t} \quad \boldsymbol{\delta} \quad \in \quad \text{(a set of constraints)}$$

Notations:

- $\boldsymbol{\delta} = ([\boldsymbol{\delta}_t]_i)_{t=1}^{\tau}$ – noise matrix (row : timestep; column : stock index)
- $c(\boldsymbol{\delta}) = \left| \{ i \in [d] \setminus \mathcal{I} \ : \ \boldsymbol{\delta}^i \neq \mathbf{0} \} \right| \ \leq \ \kappa$ – at most κ non-zero columns

Sparse, Indirect & Low Energy

Let $\mathcal{I} \subset [d]$ denotes the attack targets (i.e., stock indices)

$$\boldsymbol{\delta}^* \quad = \quad \arg \min_{\boldsymbol{\delta} \in \mathbb{R}^{\tau \times d}} \left\{ F(\boldsymbol{\delta}) \;\triangleq\; \left\| \mathbb{E}_{\mathbf{z} \sim q_\theta(\mathbf{z}|\mathbf{x}+\boldsymbol{\delta})} \left[s(\mathbf{z}) \right] \;-\; \mathbf{t}_{\mathrm{adv}} \right\|^2 \right\}$$

$$\mathrm{s.t} \quad \boldsymbol{\delta} \quad \in \quad \text{(a set of constraints)}$$

Constraints:

- $\|\boldsymbol{\delta}\|_{\max} \leq \epsilon$ – low-energy constraint
- $c(\boldsymbol{\delta}) \;\leq\; \kappa$ – sparse constraint
- $\boldsymbol{\delta}^{\mathcal{I}} \;=\; \mathbf{0}$ – no direct perturbation on attack targets (indirect)

Putting All Together …

Let $\mathcal{I} \subset [d]$ denotes the attack targets (i.e., stock indices)

$$\boldsymbol{\delta}^* \;=\; \arg\min_{\boldsymbol{\delta}\in\mathbb{R}^{\tau\times d}} \left\{ F(\boldsymbol{\delta}) \triangleq \left\| \mathbb{E}_{\mathbf{z}\sim q_\theta(\mathbf{z}|\mathbf{x}+\boldsymbol{\delta})}\left[s(\mathbf{z})\right] - \mathbf{t}_{\mathrm{adv}} \right\|^2 \right\}$$

$$\text{s.t} \quad \|\boldsymbol{\delta}\|_{\max} \le \epsilon \,,\; c(\boldsymbol{\delta}) \le \kappa \,,\; \boldsymbol{\delta}^{\mathcal{I}} = \mathbf{0}$$

Deterministic Attack: Low-Energy Constraint

- Solving

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}\in\mathbb{R}^{T\times d}} \left\{ F(\boldsymbol{\delta}) \triangleq \left\| \mathbb{E}_{\mathbf{z}\sim q_\theta(\mathbf{z}|\mathbf{x}+\boldsymbol{\delta})}\left[s(\mathbf{z})\right] - \mathbf{t}_{\mathrm{adv}} \right\|^2 \right\}$$
$$\text{s.t} \quad \|\boldsymbol{\delta}\|_{\max} \leq \epsilon \,, \boxed{c(\boldsymbol{\delta}) \leq \kappa} \,, \boldsymbol{\delta}^{\mathcal{I}} = \mathbf{0}$$

is intractable due to the discrete constraint

Deterministic Attack: Low-Energy Constraint

- Low-Energy Approximation: Using projected GD

$$\boldsymbol{\delta} \leftarrow \mathfrak{P}_{\mathcal{B}_\infty(0,\epsilon)}\Big(\boldsymbol{\delta} - \alpha \cdot \nabla_{\boldsymbol{\delta}} F(\boldsymbol{\delta})\Big)$$

$\mathfrak{P}_{\mathcal{B}_\infty(0,\epsilon)}$: projecting $\boldsymbol{\delta}$ onto the ℓ_∞-norm ball with radius ϵ.

Deterministic Attack: Sparse, Indirect Constraints

- Impose Sparse & Indirect via

$$\begin{aligned} \boldsymbol{\delta} \quad \leftarrow \quad & \arg\min_{\boldsymbol{\delta}'} \left\| \boldsymbol{\delta}' - \boldsymbol{\delta} \right\|_{\mathrm{F}} \\ \text{s.t} \quad & c(\boldsymbol{\delta}) \leq \kappa \;,\; \boldsymbol{\delta}^{\mathcal{I}} \;=\; \mathbf{0} \end{aligned}$$

which can be solved effectively

Deterministic Attack: Sparse, Indirect Constraints

- Solving:

$$\begin{aligned} \boldsymbol{\delta} \quad &\leftarrow \quad \arg\min_{\boldsymbol{\delta}'} \left\| \boldsymbol{\delta}' - \boldsymbol{\delta} \right\|_{\mathrm{F}} \\ \text{s.t} \quad & c(\boldsymbol{\delta}) \leq \kappa \ , \ \boldsymbol{\delta}^{\mathcal{I}} \ = \ \mathbf{0} \end{aligned}$$

- Keeping κ columns with largest sum, discarding the rest
  - For each column $i \in [d] \setminus \mathcal{I}$, compute $p_i = \sum_{t=1}^{T} |[\boldsymbol{\delta}_t]_i|$
  - Let π be the descending order: $p_{\pi_1} \geq \ldots \geq p_{\pi_d}$
  - Set $\boldsymbol{\delta}^{\pi_i} \leftarrow \mathbf{0}$ if $i > \kappa$

Deterministic Attack: Algorithm

- Initialize $\delta = 0$
- Per iteration:
  - Update δ via PGD to enforce low energy

$$\boldsymbol{\delta} \leftarrow \mathfrak{P}_{\mathcal{B}_\infty(0,\epsilon)}\Big(\boldsymbol{\delta} - \alpha \cdot \nabla_{\boldsymbol{\delta}} F(\boldsymbol{\delta})\Big)$$

  - Make δ sparse via solving

$$\boldsymbol{\delta} \leftarrow \arg\min_{\boldsymbol{\delta}'} \left\|\boldsymbol{\delta}' - \boldsymbol{\delta}\right\|_{\mathrm{F}}$$
$$\text{s.t} \quad c(\boldsymbol{\delta}) \leq \kappa \,, \; \boldsymbol{\delta}^{\mathcal{I}} = \mathbf{0}$$

Adversarial Noises as Random Variables

Let $\mathcal{S} = [d] \setminus \mathcal{I}$ denote the set of non-target indices,

- For $i \notin \mathcal{S}$, $\boldsymbol{\delta}^i \leftarrow \mathbf{0}$
- Otherwise, $\boldsymbol{\delta}^{\mathcal{S}}$ is a random variable

$$\boldsymbol{\delta}^{\mathcal{S}} \sim q\left(\boldsymbol{\delta}^{\mathcal{S}} \mid \mathbf{x}\right) = \prod_{i \in \mathcal{S}} q_i\left(\boldsymbol{\delta}^i \mid \mathbf{x}\right)$$

Relaxing sparse constraint: Find a parameterization for q_i such that

$$\mathbb{E}\left[c\left(\boldsymbol{\delta}^{\mathcal{S}}\right)\right] \leq \kappa$$

i.e., q has sparse support – we must also be able to sample from q

Distribution with Sparse Support

Intuition: To make sample from q sparse, its column $\boldsymbol{\delta}^i$ must have measurable probability mass at zero

Hence, a simple construction:

$$q_i\left(\boldsymbol{\delta}^i \mid \mathbf{x}\right) \;\triangleq\; r_i \cdot p_i(\boldsymbol{\delta}^i) \;+\; (1 - r_i) \cdot \mathfrak{D}\left(\boldsymbol{\delta}^i\right)$$

where $\mathfrak{D}\left(\boldsymbol{\delta}^i\right)$ is a Dirac density & p_i is any distribution that we can sample from, i.e., $p_i \equiv \mathbb{N}(\boldsymbol{\delta}^i \mid \mu(\mathbf{x}; \beta), \sigma^2(\mathbf{x}; \beta))$

Expected Sparsity

A simple construction:

$$q_i\left(\boldsymbol{\delta}^i \mid \mathbf{x}\right) \triangleq r_i \cdot p_i(\boldsymbol{\delta}^i) + (1 - r_i) \cdot \mathfrak{D}\left(\boldsymbol{\delta}^i\right)$$

Obviously, $\mathbb{P}(\boldsymbol{\delta}^i = \mathbf{0}) = 1 - r_i$

Furthermore, r_i can be selected to enforce expected sparsity:

Lemma 3.1

Choosing $r_i = \kappa \cdot \gamma_i^{\frac{1}{2}} \cdot \left(\sum_i \gamma_i\right)^{-\frac{1}{2}} \cdot d^{-\frac{1}{2}}$, it follows that $\mathbb{E}[c(\boldsymbol{\delta})] \leq \kappa$

High-Confidence Sparsity

A simple construction:

$$q_i\left(\boldsymbol{\delta}^i \mid \mathbf{x}\right) \triangleq r_i \cdot p_i(\boldsymbol{\delta}^i) + (1 - r_i) \cdot \mathfrak{D}\left(\boldsymbol{\delta}^i\right)$$

Obviously, $\mathbb{P}(\boldsymbol{\delta}^i = \mathbf{0}) = 1 - r_i$

Furthermore, r_i can be selected to enforce expected sparsity:

Lemma 3.1

Choosing $r_i = \xi \cdot \kappa \cdot \gamma_i^{\frac{1}{2}} \cdot \left(\sum_i \gamma_i\right)^{-\frac{1}{2}} \cdot d^{-\frac{1}{2}}$, $\mathbb{P}\left[c(\boldsymbol{\delta}) \leq \kappa\right] \geq 1 - \xi$

Can We Sample From It?

A simple construction:

$$q_i\left(\boldsymbol{\delta}^i \mid \mathbf{x}\right) \triangleq r_i \cdot p_i(\boldsymbol{\delta}^i) + (1 - r_i) \cdot \mathfrak{D}\left(\boldsymbol{\delta}^i\right)$$

Can we sample from q_i? YES

Lemma 3.2

Let $\boldsymbol{\delta}^{i'} \sim p_i$ and $u_i \sim \mathbb{N}(0,1)$. Then, $\boldsymbol{\delta}^i \triangleq \boldsymbol{\delta}^{i'} \cdot \mathbb{I}\left(u_i \le \Phi^{-1}(r_i)\right) \sim q_i$

Φ^{-1} denotes the inverse CDF of $\mathbb{N}(0,1)$.

Putting Together: Sparse Layer

Sparse (Differentiable) Layer:

$$\boldsymbol{\delta}^{i} \quad = \quad \boldsymbol{\delta}^{i'} \cdot \mathbb{I}\left(u_i \leq \Phi^{-1}\left(r_i(\gamma)\right)\right)$$

with $\boldsymbol{\delta}^{i'} \sim \mathbb{N}(\mu(\mathbf{x};\beta), \sigma^2(\mathbf{x};\beta))$ where (γ, β) are learnable parameters.

Adversarial Noises via Optimizing Sparse Layer

Optimizing Sparse Layer:

$$(\beta, \gamma) \quad = \quad \arg\min_{(\beta,\gamma)} \mathbb{E}_{\boldsymbol{\delta}\sim \boldsymbol{q}} \left\| \mathbb{E}_{\mathbf{z}\sim q_\theta(\mathbf{z}|\mathbf{x}+\boldsymbol{\delta})} \left[s(\mathbf{z}) \right] \;-\; \mathbf{t}_{\mathrm{adv}} \right\|^2$$

with $\boldsymbol{\delta}^{i'} \sim \mathbb{N}(\mu(\mathbf{x};\beta), \sigma^2(\mathbf{x};\beta))$ where (γ, β) are learnable parameters.

We can optimize Eq. (21) via end-to-end training

(due to the differentiable sparse layer)

OUTLINE

- Adversarial Attack in Forecasting
- New Attack in Multi-variate Time-Series Settings
  - Deterministic Attack
  - Probabilistic Attack
- Defense Strategies
  - Provable Defense
  - Minimax Defense
- Empirical Studies
  - Dataset & Metric
  - Experiment Setup & Results

Randomized Smoothing [Cohen et al., 2019]

Post-Training Defense in Classification:

- Replace original classifier $g(\mathbf{x})$ with

$$g_\sigma(\mathbf{x}) \triangleq \arg\max_c \mathbb{P}\left(g(\mathbf{x}+\boldsymbol{\epsilon}) = c\right) \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathbb{N}(\mathbf{0}, \sigma^2\mathbf{I})$$

- Robustness Guarantee: $g_\sigma(\mathbf{x}+\boldsymbol{\delta}) = g_\sigma(\mathbf{x})$ if $\|\boldsymbol{\delta}\| \leq \mathbb{O}(\sigma)$

Adaptation of RS to Forecasting

Post-Training Defense in Probabilistic Forecasting:

- Replace $z(\mathbf{x}) \sim q_\theta(\mathbf{z} \mid \mathbf{x})$ with $z_\sigma(\mathbf{x}) = \mathbb{E}\left[z(\mathbf{x} + \boldsymbol{\epsilon})\right]$ where

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \mathbb{N}(\mathbf{0}, \sigma^2 \mathbf{I}) \\ z(\mathbf{x} + \boldsymbol{\epsilon}) &\sim q_\theta(\mathbf{z} \mid \mathbf{x} + \boldsymbol{\epsilon}) \end{aligned}$$

- Robustness Guarantee: Let $\mathbb{P}(z_\sigma(\mathbf{x}) \preceq \mathbf{r})$ and $\mathbb{P}(z_\sigma(\mathbf{x} + \boldsymbol{\delta}) \preceq \mathbf{r})$ denote the CDF of $z_\sigma(\mathbf{x})$ and $z_\sigma(\mathbf{x} + \boldsymbol{\delta})$,

Lemma 3.3

$$\sup_{\mathbf{r}} \left| \mathbb{P}(z_\sigma(\mathbf{x}) \preceq \mathbf{r}) - \mathbb{P}(z_\sigma(\mathbf{x} + \boldsymbol{\delta}) \preceq \mathbf{r}) \right| \leq \frac{\sqrt{d}}{\sigma} \cdot \|\boldsymbol{\delta}\|_{\mathrm{F}}$$

Minimax Defense

- Model parameters are updated to minimize worst-case impact of a simulated attack (in-training defense)

- Min-step: Update attack parameters to minimize the gap between the forecast & manipulation target

- Max-step: Update forecast parameters to maximize the gap between the forecast & manipulation target

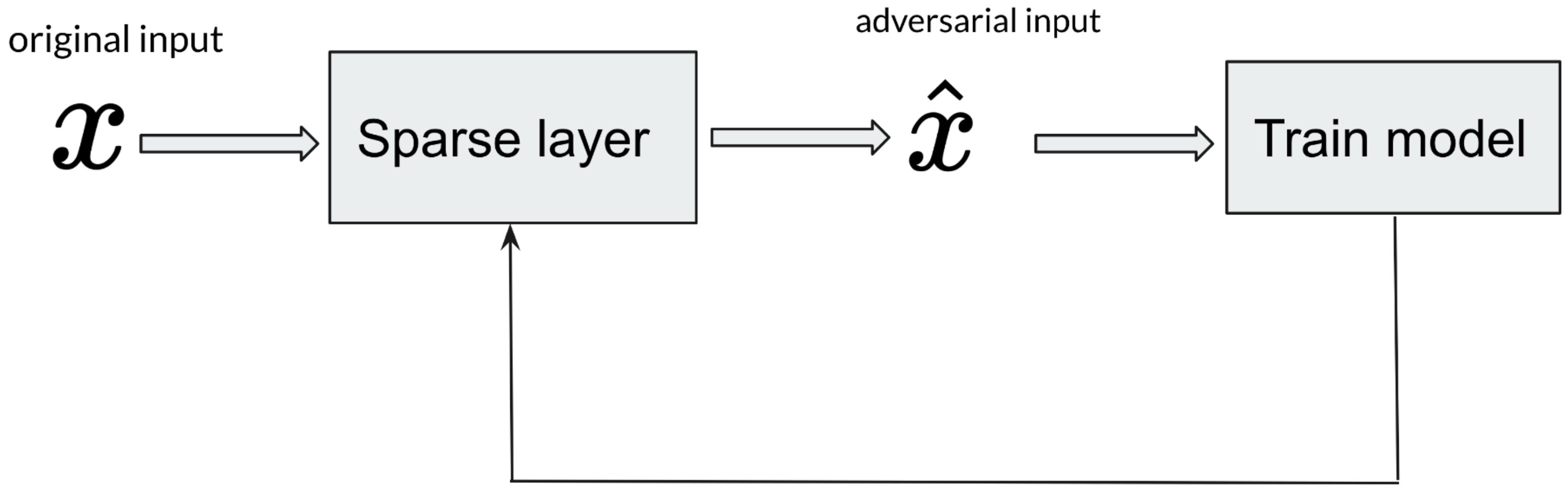
Minimax Defense

- Min-step: Update attack parameters to minimize the gap between the forecast & manipulation target

$$(\beta, \gamma) = \underset{(\beta,\gamma)}{\arg\min} \, \mathbb{E}_{\boldsymbol{\delta} \sim q} \left\| \mathbb{E}_{\mathbf{z} \sim q_\theta(\mathbf{z}|\mathbf{x}+\boldsymbol{\delta})} \left[s(\mathbf{z}) \right] - \mathbf{t}_{\mathrm{adv}} \right\|^2$$

Minimax Defense

- Max-step: Update forecast parameters to maximize the gap between the forecast & manipulation target



Minimax Defense

- Min-step: Update attack parameters to minimize the gap between the forecast & manipulation target

- Max-step: Update forecast parameters to maximize the gap between the forecast & manipulation target

- Probabilistic attack (thanks to its differentiability) can be utilized to enable end-to-end optimization

OUTLINE

- Adversarial Attack in Forecasting
- New Attack in Multi-variate Time-Series Settings
  - Deterministic Attack
  - Probabilistic Attack
- Defense Strategies
  - Provable Defense
  - Minimax Defense
- Empirical Studies
  - Dataset & Metric
  - Experiment Setup & Results

Dataset

- TRAFFIC [Asuncion & Newman, 2007]: hourly occupancy rate (between 0 & 1) of 963 SF car lanes
- ELECTRICITY [Asuncion & Newman, 2007]: hourly consumption rate from 370 customers
- TAXI [Taxi and Commission, 2015]: no. of taxi rides every 0.5 hour across 1214 NYC locations (2015-2016)
- WIKI [Lai, 2017]: daily page views of 2000 Wiki pages

Data Statistics

Dataset	Prediction Horizon h	Dimension	Time steps
TRAFFIC	24	963	10413
TAXI	24	1214	1488
WIKI	30	2000	792
ELECTRICITY	24	370	5790

Metric

$\{\mathbf{z}_{i,t}^{(e)}\}_{e=1}^{m}$: m prediction samples at time t for TS i generated by $q_\theta(\mathbf{z} \mid \mathbf{x})$

$\mathbf{z}_{i,t}^{\alpha}$: the α-quantile ($\alpha \in (0,1)$) value of the sorted sequence of $\{\mathbf{z}_{i,t}^{(e)}\}_{e=1}^{m}$

Metric: Weighted Quantile Loss (WQL)

$$\begin{aligned}\mathrm{WQL}(\alpha) &= 2 \cdot \left(\sum_{i,t} |\mathbf{x}_{i,t}|\right)^{-1} \\ &\times \left[\sum_{i,t}\left(\alpha \cdot \max(\mathbf{x}_{i,t} - \mathbf{z}_{i,t}^{\alpha}, 0) + (1-\alpha) \cdot \max(\mathbf{z}_{i,t}^{\alpha} - \mathbf{x}_{i,t}, 0)\right)\right]\end{aligned}$$

Metric

Weighted Quantile Loss (WQL)

$$\begin{aligned}\mathrm{WQL}(\alpha) &= 2 \cdot \left(\sum_{i,t} |\mathbf{x}_{i,t}| \right)^{-1} \\ &\times \left[\sum_{i,t} \Big(\alpha \cdot \max(\mathbf{x}_{i,t} - \mathbf{z}^{\alpha}_{i,t}, 0) \; + \; (1-\alpha) \cdot \max(\mathbf{z}^{\alpha}_{i,t} - \mathbf{x}_{i,t}, 0) \Big) \right]\end{aligned}$$

Small α : penalize overestimation

Large α : penalize underestimation

Experiment Setup

- **TRAFFIC**: statistic to be attacked $s(\mathbf{z}) = (\mathbf{x}_{1,\tau+h-1}, \mathbf{x}_{1,\tau+h}, \mathbf{x}_{5,\tau+h-1}, \mathbf{x}_{5,\tau+h})$ prediction horizon $h = 24$, context length $\tau = 96$

- **ELECTRICITY** & **TAXI**: statistic to be attacked $s(\mathbf{z}) = \mathbf{x}_{1,\tau+h}$, prediction horizon $h = 24$, context length $\tau = 96$

- **WIKI**: statistic to be attacked $s(\mathbf{z}) = \mathbf{x}_{1,\tau+h}$, prediction horizon $h = 30$, context length $\tau = 120$

Experiment Setup

Defense Baselines:

- No Defense
- Data Augmentation (DA): Train the forecaster with Gaussian perturbed input, i.e. $\mathbf{x} + \epsilon$ where $\epsilon \sim \mathbb{N}(\mathbf{0}, \mathbf{I})$
- Adapted Randomized Smoothing (RS)
- Minimax Defense

Result: ELECTRICITY

Table: Average wQL on **Electricity** dataset under **deterministic** and **probabilistic** attack. Target TS $\mathcal{I} = \{1\}$ and attacked time stamp $H = \{\tau\}$.

	deterministic attack				probabilistic attack			
sparsity (κ)	no defense	DA	RS	mini-max	no defense	DA	RS	mini-max
no attack	0.2853	0.2288	0.2176	**0.2154**	0.2909	0.2374	**0.2237**	0.2342
1	0.3410	0.2949	**0.2826**	0.2990	**0.4364**	0.5923	0.5940	0.4935
3	0.4559	**0.3655**	0.3757	0.3775	0.7245	0.5738	**0.4581**	0.8079
5	0.5770	0.5554	0.5560	**0.5273**	0.9143	0.8422	0.9276	**0.5265**
7	0.6687	0.7076	0.7072	**0.6506**	0.9991	0.8267	1.0100	**0.6161**
9	0.8282	0.8412	0.8327	**0.7503**	1.0317	0.8139	0.8919	**0.6466**

Observations:

- All attacks become more effective as κ increases
- Probabilistic attack is much more effective – see **no defense** column
- Both randomized smoothing (RS) and minimax are more effective than data augmentation (DA)

Result: TRAFFIC

Table: Average wQL on **Traffic** dataset under **deterministic** and **probabilistic** attack. Target time series $\mathcal{I} = \{1, 5\}$ and attacked time stamp $H = \{h-1, h\}$.

	deterministic attack				probabilistic attack			
sparsity (κ)	no defense	DA	RS	mini-max	no defense	DA	RS	mini-max
no attack	0.2283	0.1573	**0.1529**	0.1837	0.2283	0.1573	**0.1529**	0.1837
1	0.2190	0.1543	**0.1529**	0.1701	0.2428	0.1807	**0.1796**	0.1904
3	0.2150	0.1884	0.1890	**0.1687**	0.2219	0.2564	0.2467	**0.1714**
5	0.2772	0.2729	0.2648	**0.1688**	0.2719	0.3026	0.3003	**0.1883**
7	0.3620	0.3597	0.3535	**0.1779**	0.3529	0.2893	0.2824	**0.1846**
9	0.4635	0.4058	0.4240	**0.1970**	0.4075	0.3544	0.3376	**0.1911**

Observations:

- Attack becomes more effective as value for κ increases
- Probabilistic attack is competitive with deterministic attack, and is better with smaller values of κ.
- Best defenses are either randomized smoothing (RS) or minimax

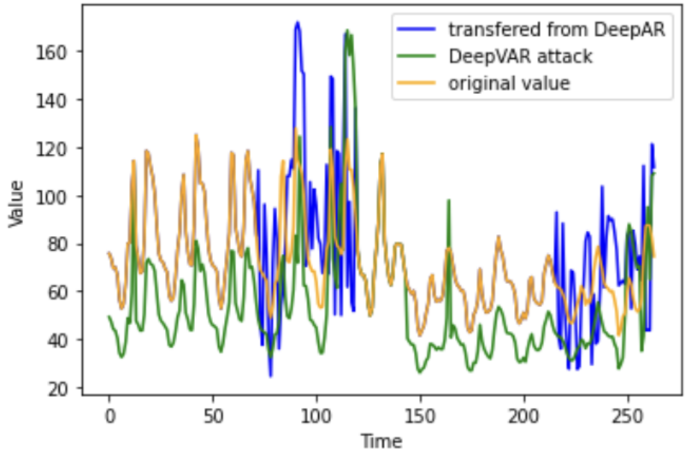
Result: Attack Transferability

Can we transfer the attack from univariate TS to multivariate TS?
Previous experiment shows that

$$\text{TS } 5 \overset{\text{attack}}{\rightarrow} \text{TS } 1$$

- Consider univariate model, e.g. DeepAR
- Generate adversarial attack on TS 5 from DeepAR
- Transfer the attack to DeepVAR
- Observe whether prediction of TS 1 can be altered

Result: Attack Transferability



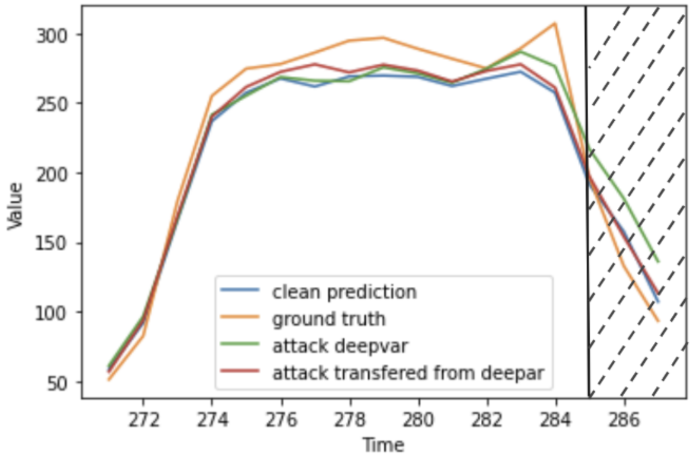
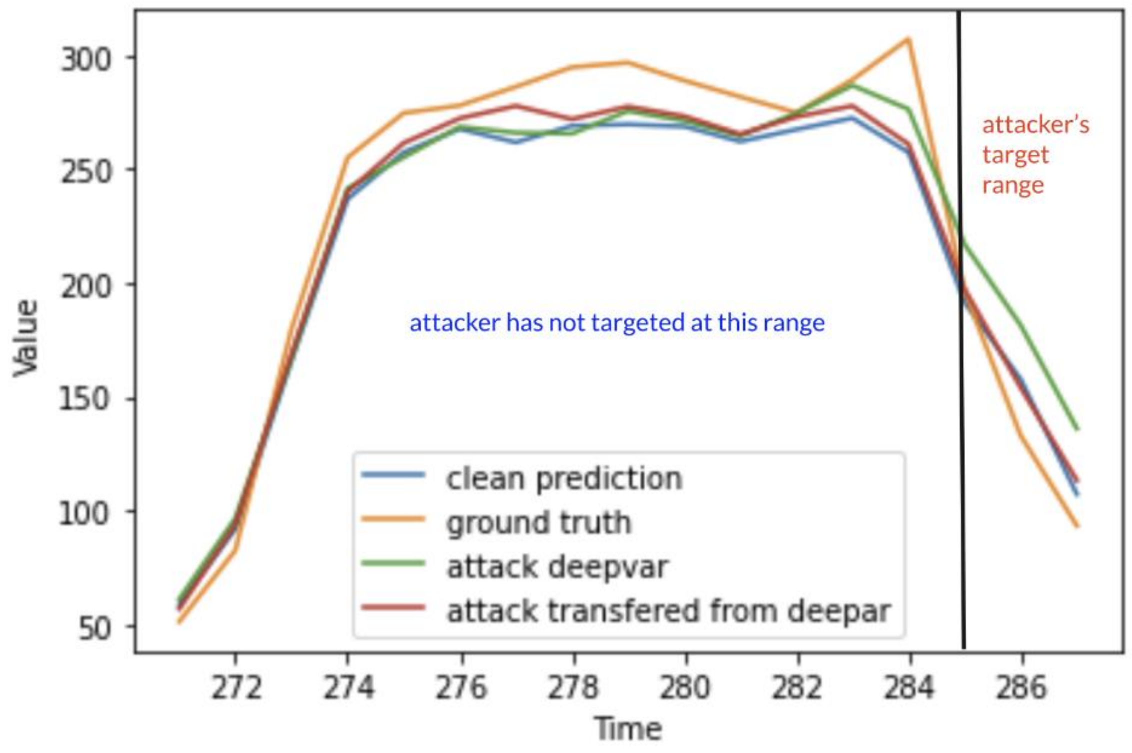


Figure: Plots of (a) authentic (orange), DeepAR-attacked (blue) and DeepVAR-attacked (green) versions of time-series (TS) 5; and (b) ground-truth (orange), no-attack (blue), under-DeepAR-attack (red) and under-DeepVAR-attack (green) predictions for TS 1. Shaded area is attacker's target range.

Compared to clean prediction, the value of TS 1 at the attack time step ($t = 288$) were adversely altered by DeepVAR-attack (green) but only slightly altered by DeepAR-attack (red).

Result: Attack Transferability



In the target range of the attacker:

- Multivariate attack can mislead the prediction
- Univariate attack has almost no effect

More Experiments

Please refer to:

https://openreview.net/forum?id=ctmLBs8lITa

Our source code is available at:

https://github.com/awslabs/gluonts/tree/dev/src/gluonts/nursery/robust-mts-attack

Take-Home Messages

Attack:

1. Attack to MTS forecasting can be indirect & sparse
2. Probabilistic attack is more effective than deterministic attack under extreme sparse settings
3. Attack crafted in univariate settings does not translate to MTS settings

Take-Home Messages

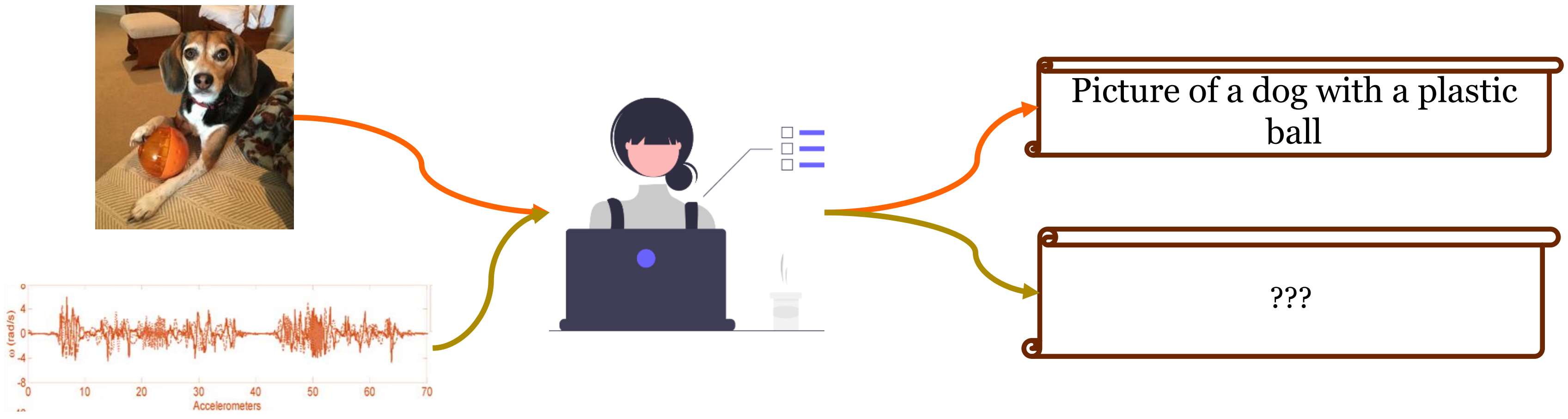
Defense:

1. Randomized Smoothing (RS) [Cohen et al., 2019] can be adapted to forecasting with provable guarantee
2. Minimax Defense achieves the best results in most cases (though with no guarantee)

OOD detection and synthetic data generation

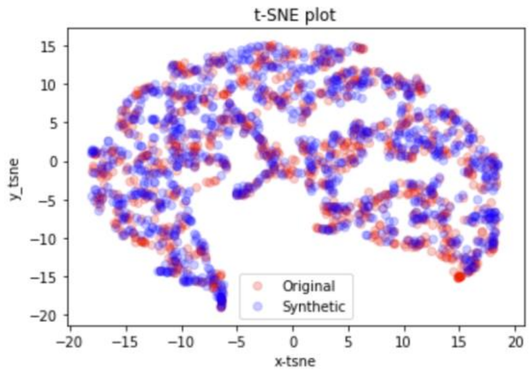
Challenges of data collection

- Vast amounts of time series are collected from IoTs and wearable devices
- Data labeling is costly:
  - User self-labeling
  - Event rarity that leads to imbalanced classes
  - Ambiguity of labeling post collection



Generative AI as a solution for synthetic data

- Several approaches were proposed for time-series:
  - Adversarial networks
    - [6] Yoon, Jinsung, Daniel Jarrett, and Mihaela Van der Schaar. "Time-series generative adversarial networks." *Advances in neural information processing systems* 32 (2019).
    - Festag, Sven, Joachim Denzler, and Cord Spreckelsen. "Generative adversarial networks for biomedical time series forecasting and imputation." *Journal of Biomedical Informatics* 129 (2022): 104058.
  - Long-term forecasting
    - Li, Yan, et al. "Generative time series forecasting with diffusion, denoise, and disentanglement." *Advances in Neural Information Processing Systems* (2022)

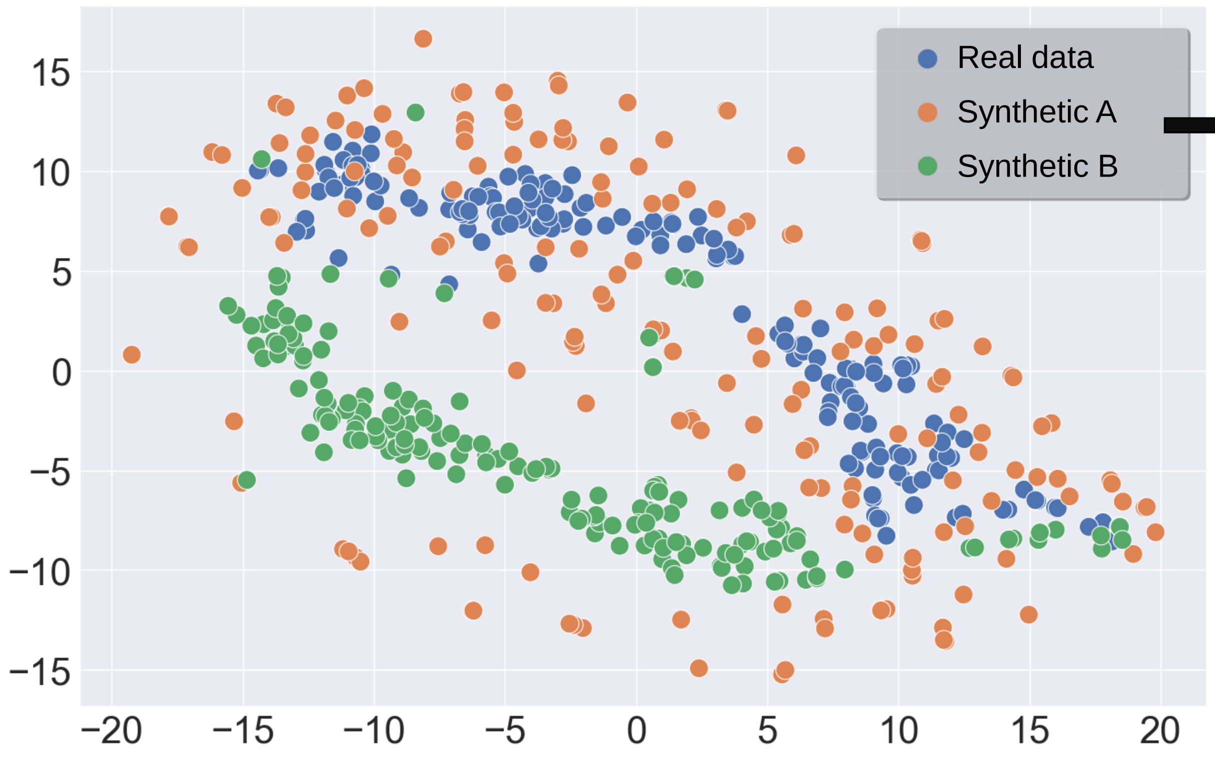


t-SNE visualization of the distribution of Time-GAN generated examples[6]

Challenges of Generative AI for time-series data

- Conditional generative process
  - Class-specific synthetic data
  - Condition unseen during data collection
- Reconstruction error related metrics
  - Depends on the data collected
  - Does not generalize on unseen feature combinations
- Discriminative models that aims to classify synthetic vs original data
  - Modeling/Data bias
- Train on Original/Test on Synthetic or Train on Synthetic/Test on Original
  - Modeling/Data bias

Reliability challenges: Data Validation



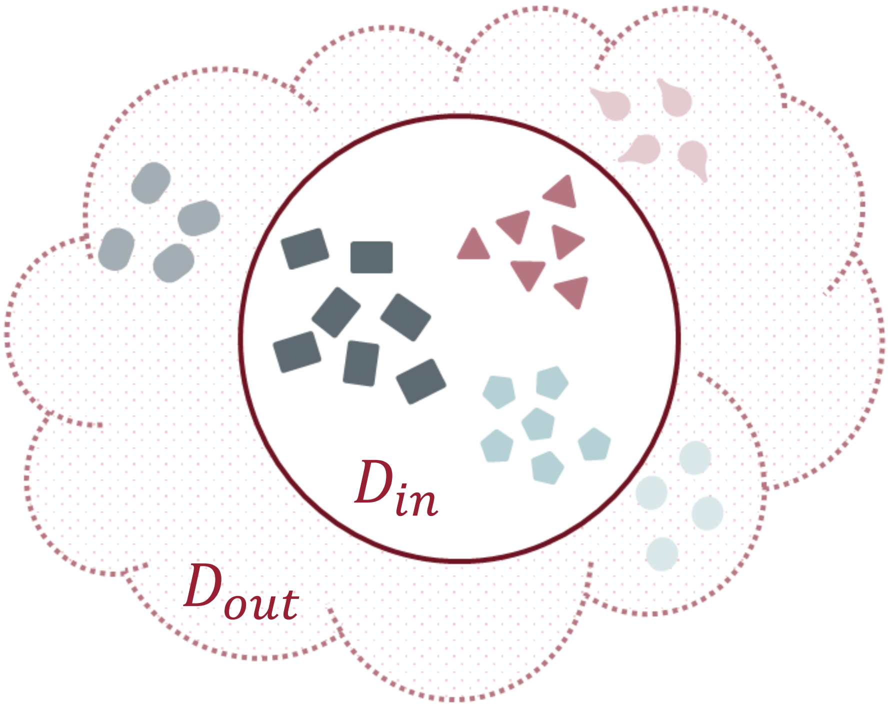
Which of the synthetic data should we retain/discard?

t-Distributed Stochastic Neighbor Embedding showing the distribution of real-world data examples and synthetic data.

- Data validation through Out-of-Distribution Lens

Data Distribution Challenge

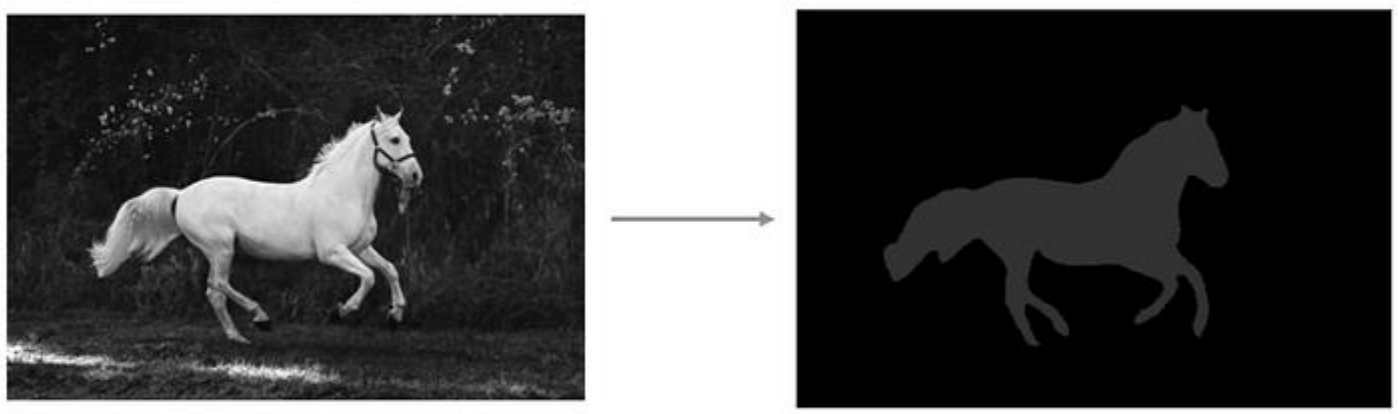
- Out-Of-Distribution (OOD) samples are anomalous or deviant data from the training set
- OOD space is a main concern of AI safety in general



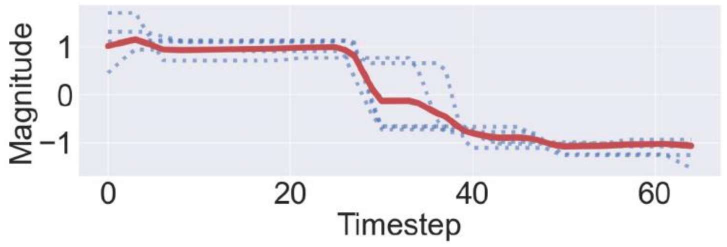
$\rightarrow D_{in} \cap D_{out} = \emptyset$

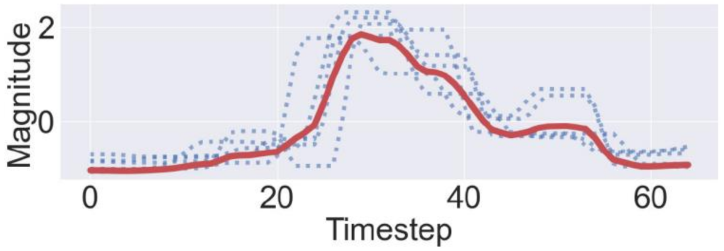
Key insight: Decomposition

- Image: Foreground(Important) + Background



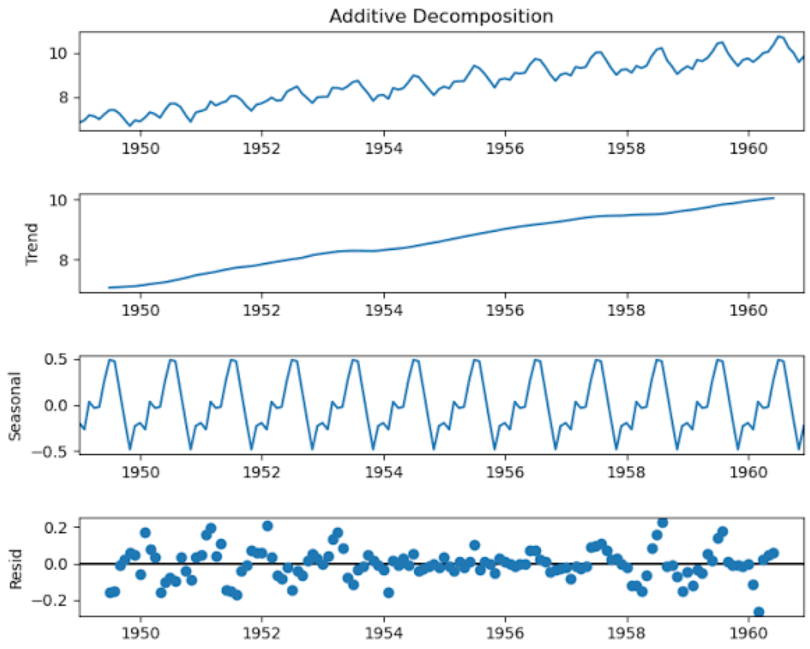
- Time-series: Pattern + Noise





STL Decomposition

- Seasonal-Trend decomposition using LOESS (STL) is a statistical method of decomposing a Time Series into three components:



- Trend: Describes the increasing /decreasing in the observations overtime.
- Seasonal: The regular temporal pattern in the observations.
- Remainder: Noise

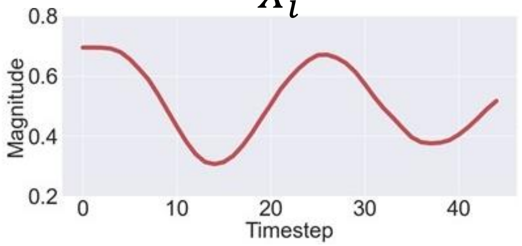
OOD in Time-Series Domain: Hypothesis

- Each time-series example (X_i, y_i) from the in-distribution data D_{in} consists of two components.

  1. A class-wise semantic pattern S_y for each class label $y \in Y$ representing the meaningful semantics of the class label y.

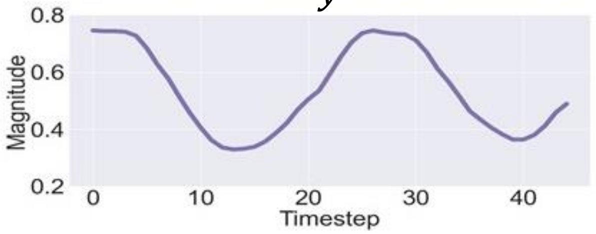
  2. A remainder noise r_i representing an additive perturbation to the semantic portion.

X_i



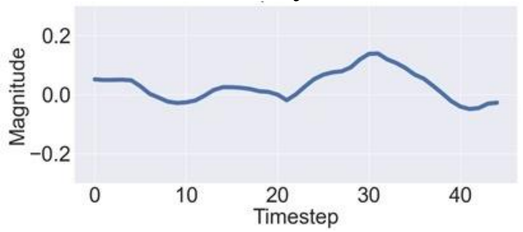
$=$

S_y



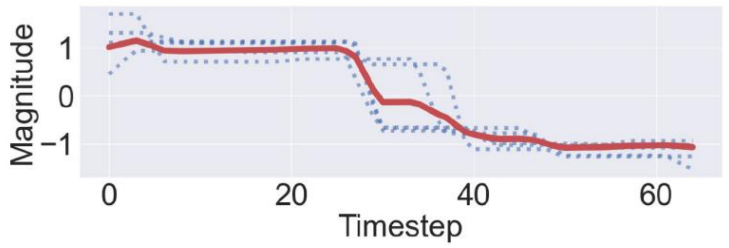
$+$

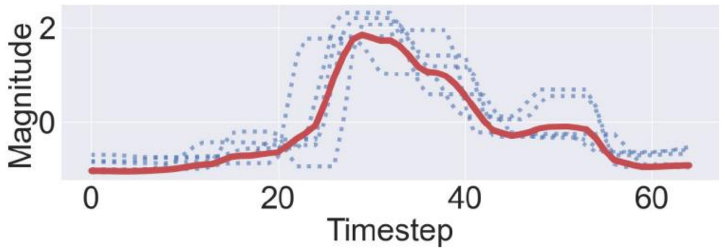
r_i



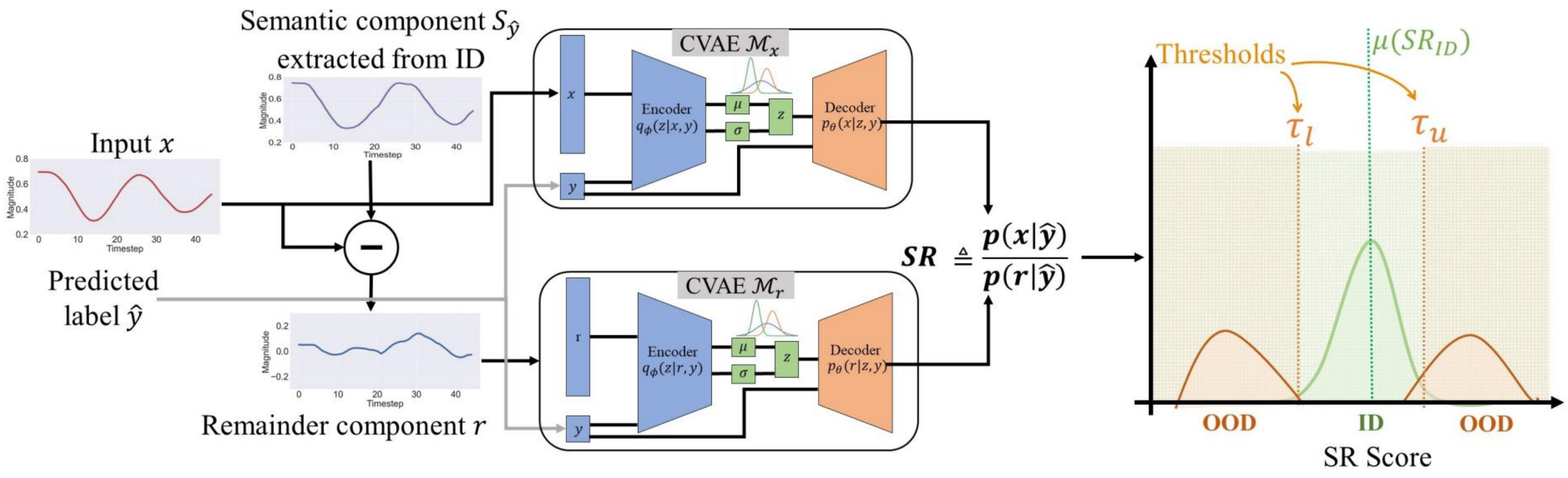
OOD in Time-Series Domain: Hypothesis

- Let $X \in R^{n \times T}$ is a time-series signal and $y \in Y = \{1, \dots, C\}$ be the corresponding class label. As $X = S_y + r$ and S_y is a deterministic component:
  - X is an OOD example if $p(X|y) \neq p(r|y)$
  - X is an in-distribution example if $p(X|y) = p(r|y)$.





Seasonal Ratio Score Framework



Motivation for VAEs

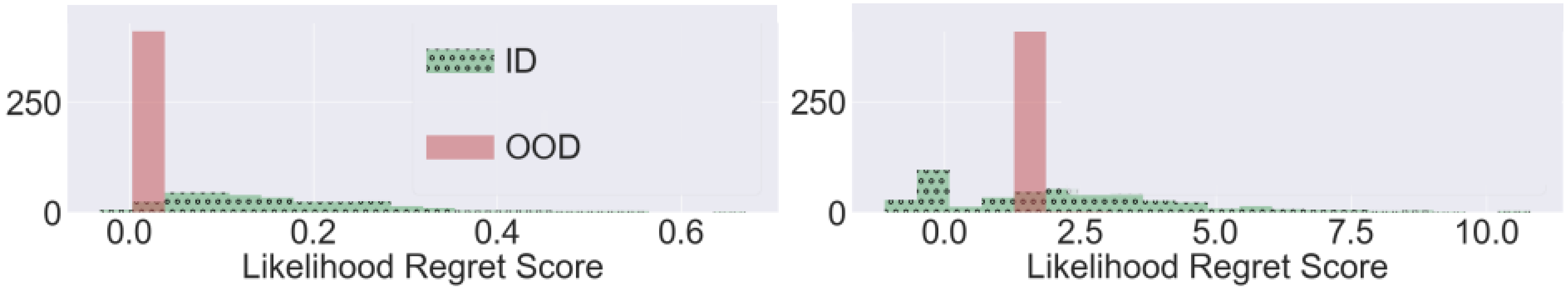
- Ease of Training
  - Compared to other methods such as GANs, VAEs are easier to train
- Structured Latent Space
  - VAEs are designed to produce a regularized continuous latent space
- Interpretable Generative Tasks
  - Interpretable representations
  - Learns the underlying data distribution is crucial
  - Stochastic by design

Likelihood Regret Approach

Algorithm 1 Computing Likelihood Regret (LR)

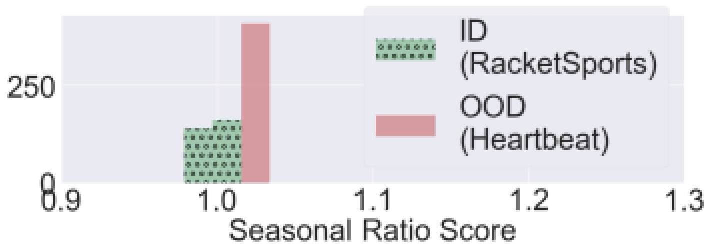
```
Input: Test sample $\mathbf{x}$, trained VAE $(E_{\phi^*}, D_{\theta^*})$, number of posterior samples for likelihood estimation $K$, number of optimization step $S$, learning rate $\gamma$.
1: $L_{\text{VAE}} = \mathcal{L}_K(\mathbf{x}; \theta^*, \phi^*)$        ▷ Estimate the log likelihood of $\mathbf{x}$ under the VAE model by (2)
2: Set $\phi = \phi^*$
3: for $S$ iterations do
4:     $\phi \leftarrow \text{Adam}(\phi, \nabla_\phi(-\mathcal{L}(\mathbf{x}; \theta^*, \phi)), \gamma)$        ▷ Optimize $\phi$ by maximizing the ELBO objective
5: $L_{\text{OPT}} = \mathcal{L}_K(\mathbf{x}; \theta^*, \phi)$        ▷ Estimate the log likelihood of $\mathbf{x}$ with optimized encoder
6: LR $= L_{\text{OPT}} - L_{\text{VAE}}$
```

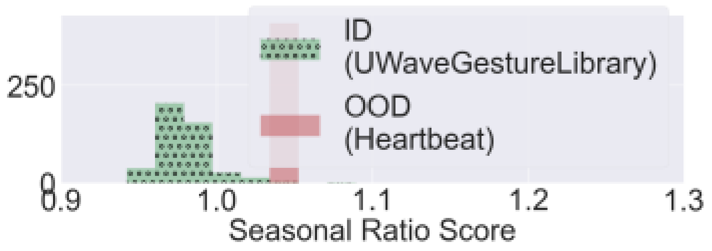
Likelihood Regret Failure



Histogram showing the non-separability of ID and OOD examples using Likelihood Regret scores

Seasonal Ratio Score success





Histogram showing the separability of ID and OOD examples using Likelihood Regret scores

Seasonal Ratio Score performance

- In-Domain OOD: Both ID and OOD datasets belong to the same domain application.
- Cross-Domain OOD: Both ID and OOD datasets come from two different domains.

	In-Domain OOD	Cross-Domain OOD
Superior Performance by Likelihood Regret	22.3%	17.7%
Ties	15.7%	20.0%
Superior Performance by Seasonal Ratio Score	62.0%	62.3%

AUROC results for LR, and SR on 23 different datasets for both in-domain and cross-domain OOD setting.

References

- T. Belkhouja and J. Doppa. *Adversarial Framework with Certified Robustness for Time-Series Data via Statistical Features*. Journal of Artificial Intelligence Research (JAIR), 73: 1435-1471, 2022.
- Taha Belkhouja* , Dina Hussein* , Ganapati Bhat, and Janardhan Rao Doppa. *Reliable Machine Learning for Wearable Activity Monitoring: Novel Algorithms and Theoretical Guarantees*. ACM/IEEE International Conference on Computer-Aided Design (ICCAD), 2022.
- Taha Belkhouja* , Dina Hussein* , Ganapati Bhat, and Janardhan Rao Doppa. *Energy Efficient Missing Data Recovery in Wearable Devices: A Novel Search-based Approach*. ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2023.
- Dina Hussein and Ganapati Bhat*. CIM: A Novel Clustering-based Energy-Efficient Data Imputation Method for Human Activity Recognition*. ACM Transactions on Embedded Computing Systems 22, no. 5s (2023): 1-26.
- Dina Hussein , and Ganapati Bhat. *SensorGAN: A Novel Data Recovery Approach for Wearable Human Activity Recognition*. ACM Transactions on Embedded Computing Systems (2023).

References

- Dina Hussein† , Aaryan Jain, and Ganapati Bhat. *Robust Human Activity Recognition using Generative Adversarial Imputation Networks*. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2022.
- T. Belkhouja, Y. Yan, and J. Doppa. *Dynamic Time Warping based Adversarial Framework for Time-Series Domain*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 45(6): 7353-7366, 2022
- T. Belkhouja, Y. Yan, and J. Doppa. *Training Robust Deep Models for Time-Series Domain: Novel Algorithms and Theoretical Analysis*. Proceedings of 36th AAAI Conference on Artificial Intelligence, 2022.
- Linbo Liu, Youngsuk Park, Trong Nghia Hoang, Hilaf Hasson, and Jun Huan. *Robust Multivariate Time-Series Forecasting: Adversarial Attacks and Defense Mechanisms*. International Conference on Learning Representation (ICLR), 2023.
- H. A. Dau, D. F. Silva, F. Petitjean, G. Forestier, A. Bagnall, A. Mueen, and E. Keogh. *Optimizing dynamic time warping's window width for time series data mining applications*. Data mining and knowledge discovery, 2018

References

- T. Belkhouja, Y. Yan, and J. Doppa. *Out-of-Distribution Detection in Time-Series Domain: A Novel Seasonal Ratio Scoring Approach.* ACM Transactions on Intelligent Systems and Technology (TIST), 15(1): 9:1-9:24, 2023.
- M. Cuturi. *Fast global alignment kernels*. Proceedings of the 28th international conference on machine learning (ICML). 2011.
- Wen, Qingsong, Jingkun Gao, Xiaomin Song, Liang Sun, Huan Xu, and Shenghuo Zhu. *RobustSTL: A robust seasonal-trend decomposition algorithm for long time series.* Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. No. 01. 2019.
- Zhisheng Xiao, Qing Yan, and Yali Amit. 2020. *Likelihood Regret: An Out-of-Distribution Detection Score For Variational Auto-encoder*. In Advances in Neural Information Processing Systems (NeurIPS),2020.

Outstanding Challenges and Future Research Directions

Outstanding Challenges

- Interpretability and explainability of time-series ML models
- Problem space of robustness for time-series ML is relatively new
  - Need to evaluate on more challenging application settings
- Development of deep generative models for time-series data
  - Interpretability is a huge challenge unlike other data modalities
- Generating valid synthetic time-series data (especially long T)
  - Deep generative models + OOD detectors?
- Uncertainty quantification for time-series tasks
  - Prediction intervals/sets with coverage guarantees
- Lack of large labeled datasets
  - Methods to leverage prior knowledge and weak supervision