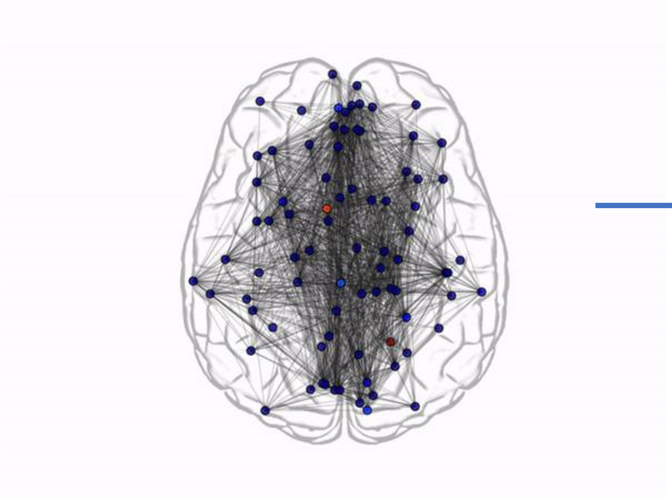
Detecting and Leveraging Changes in Temporal Data

Nauman Ahad

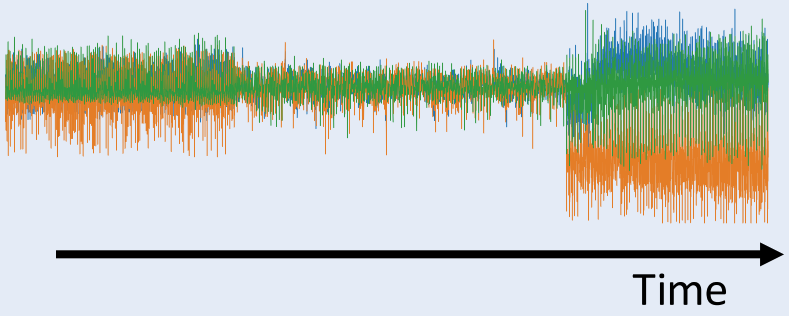
Ph.D. Defense

School of Electrical and Computer Engineering
Georgia Institute of Technology

Time Series Data is All Around Us

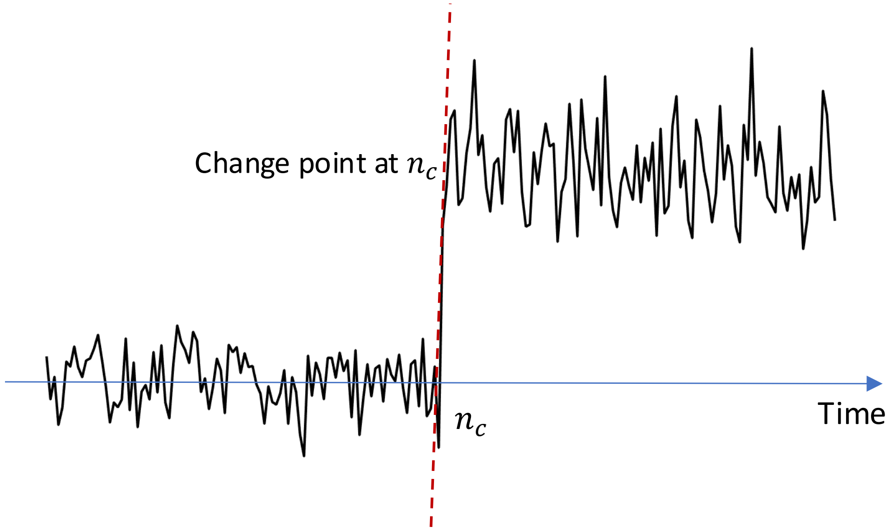


Require analyzing time series data



Detecting Changes in Temporal Data

Many applications require detecting changes in temporal data



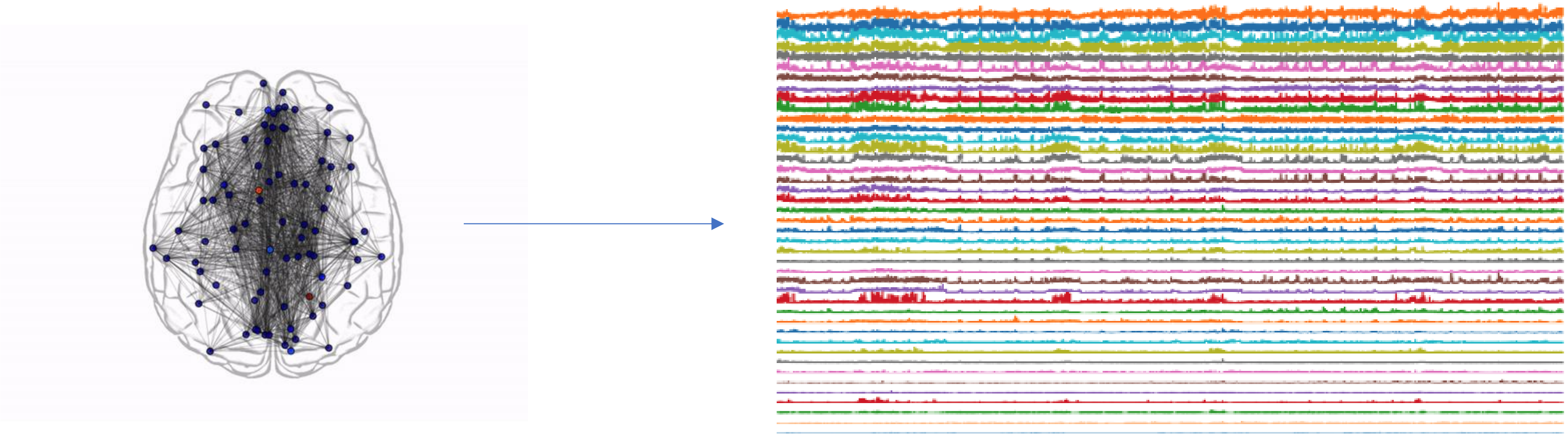
- Process control, vital signs, detecting network attacks, etc.
- Need to identify these changes *quickly*

Challenges in Change Detection

Modern applications require identifying changes which are **challenging** to detect

- Changes between *complicated,* non-parametric distributions
- *Multivariate* signals

  Some changes important, some not..
- Detecting *multiple* changes

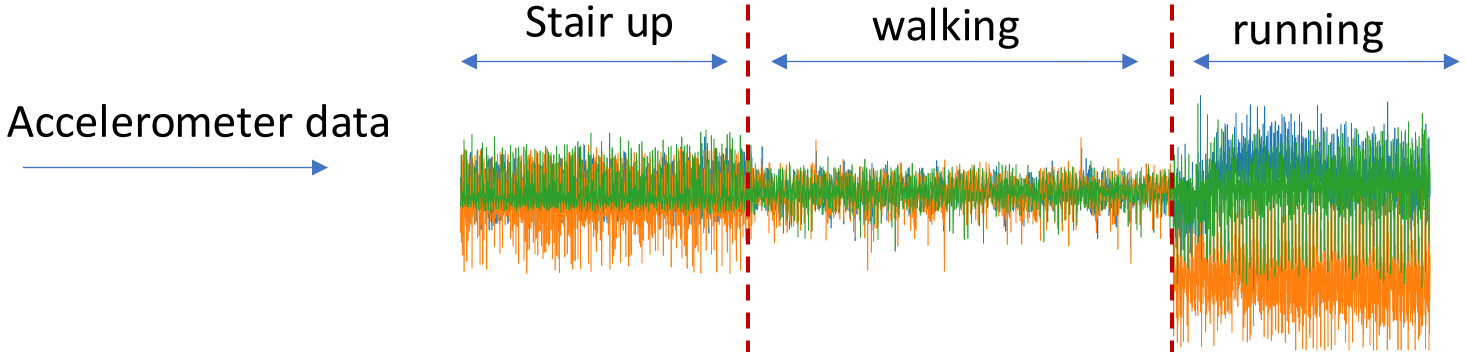


Classifying Temporal Data

Many applications require classifying time series data into different classes

For example:

Activity recognition



Accelerometer data

Stair up

walking

running

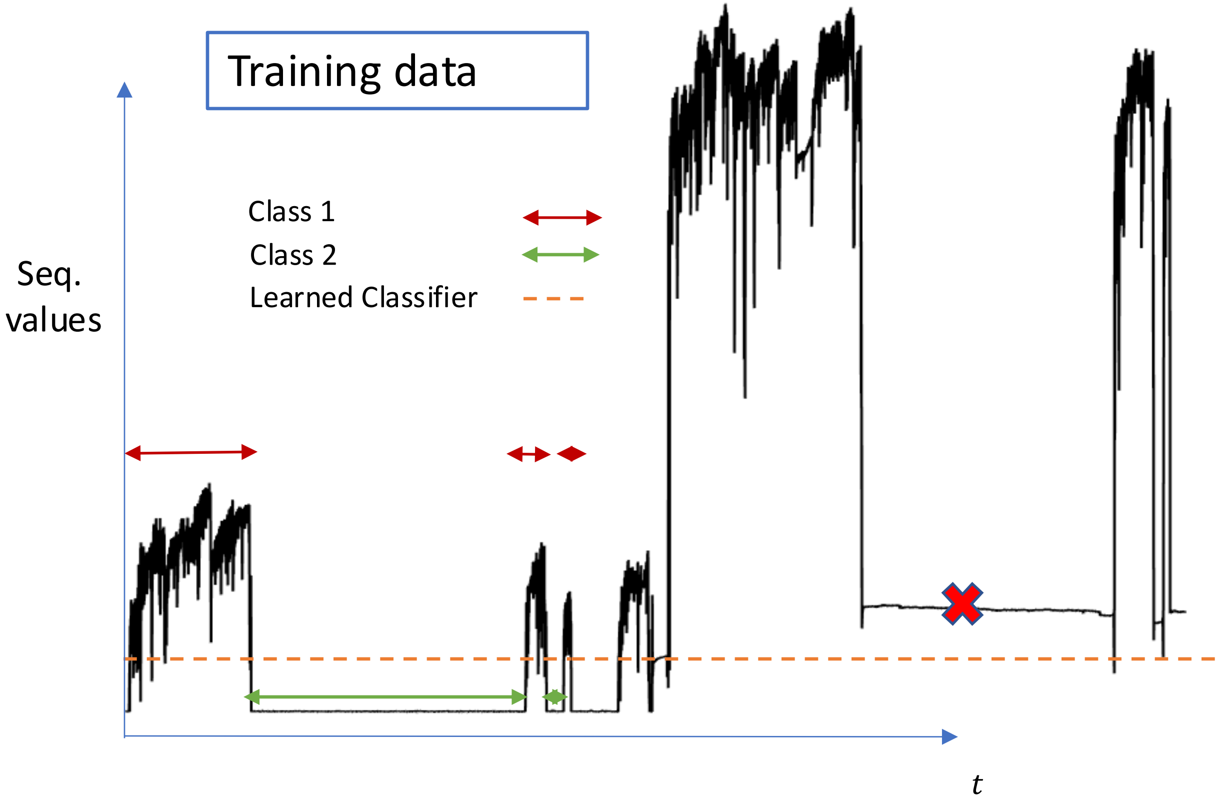
Machine learning models trained through supervision to classify sub-sequences

Challenges in Learning Supervised ML Models

Developing machine learning for the real-world is **challenging**

- Training data can be *limited*
- Supervised models can perform poorly in real-world settings under *distribution shifts*

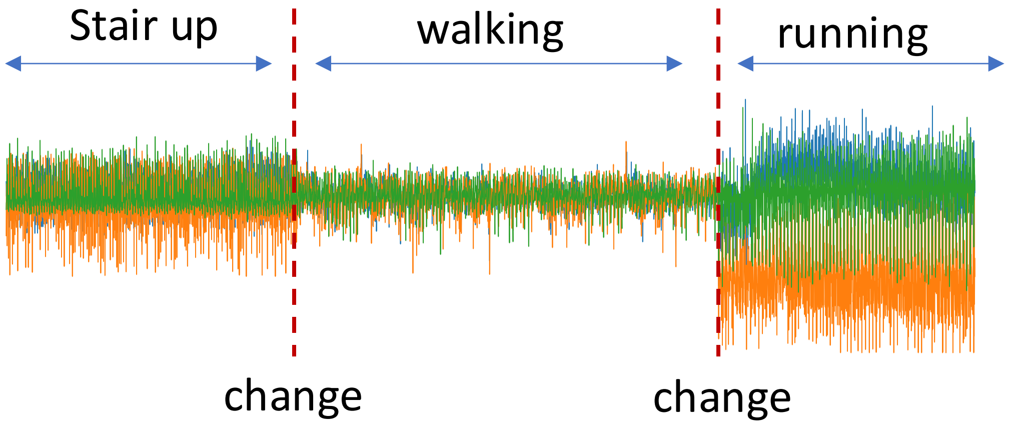
Supervised Models in the Real World



Supervised classifier *fails* as data distribution *changes*

Common Theme

Common theme between change detection and machine learning classification?



- Machine learning classification models:

  - ***Learn*** separability between different categories of interest through examples

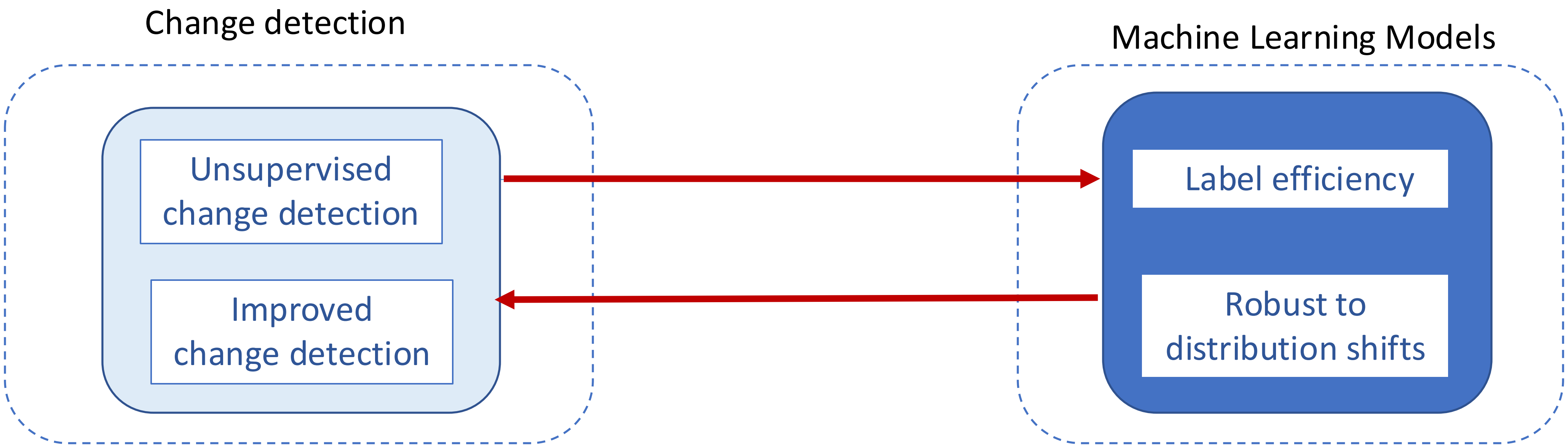
- Change points:

  - ***Identify*** where data distributions become different

A common theme: ***Separability between data distributions***

Leverage this common theme to see how machine learning and change detection can benefit each other ?

Overview of Thesis


Change detection
Unsupervised change detection
Improved change detection
Machine Learning Models
Label efficiency
Robust to distribution shifts

Proposes new methods that show how:

1 Change detection can help machine learning

2. Machine learning can help improve change detection

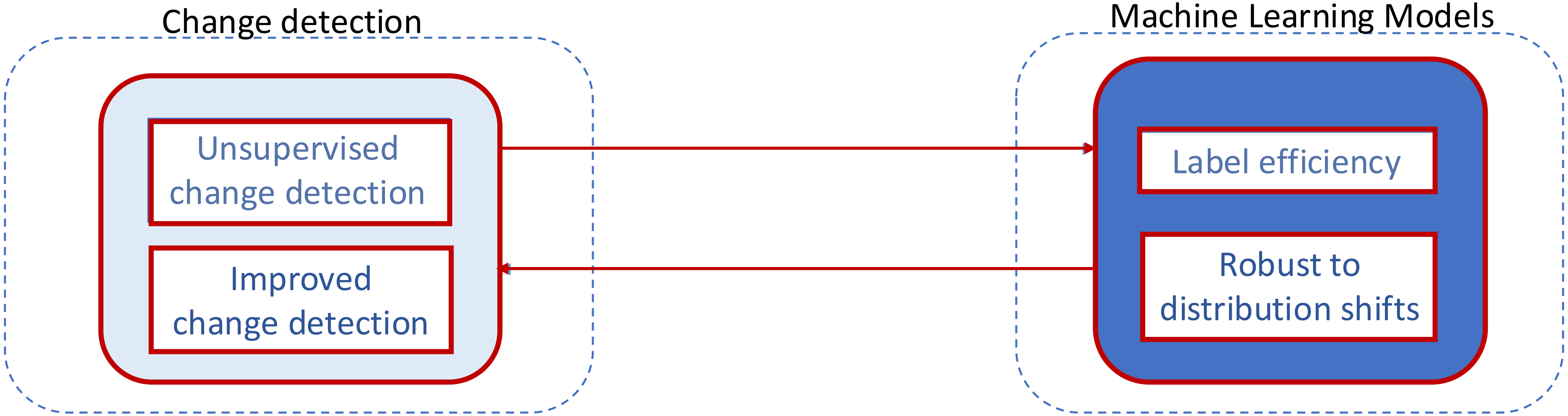
Four Main Aims of Thesis

1. Multiple change point detection in streaming data settings

2. Using change detection for label efficient supervised learning

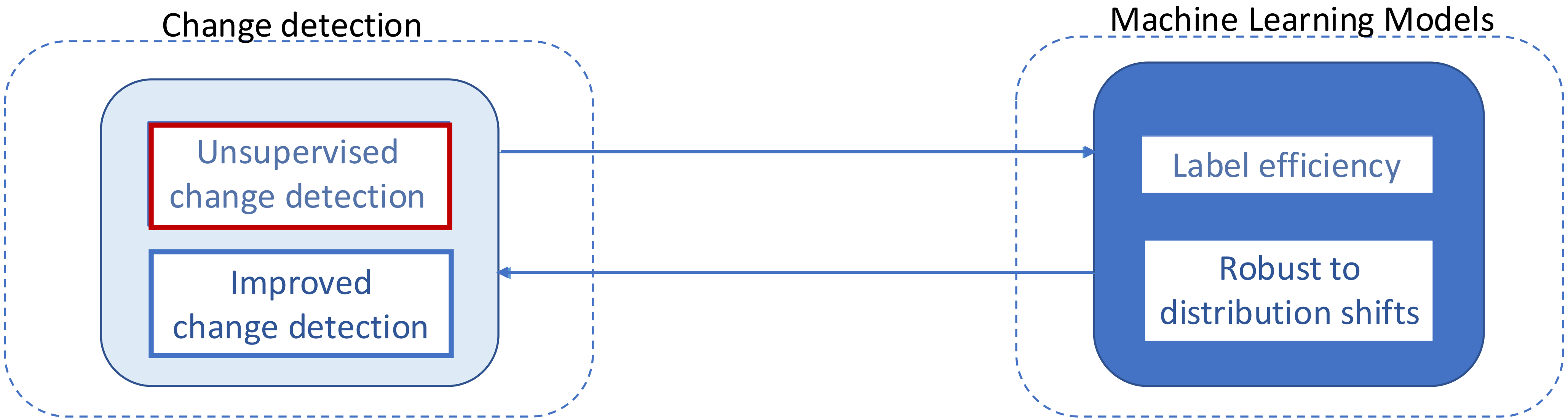
3. Using supervision tools from machine learning to improve change detection

4. Improving supervised models in the presence of distribution shifts


Change detection
Machine Learning Models
Unsupervised change detection
Improved change detection
Label efficiency
Robust to distribution shifts

Improving Change Detection

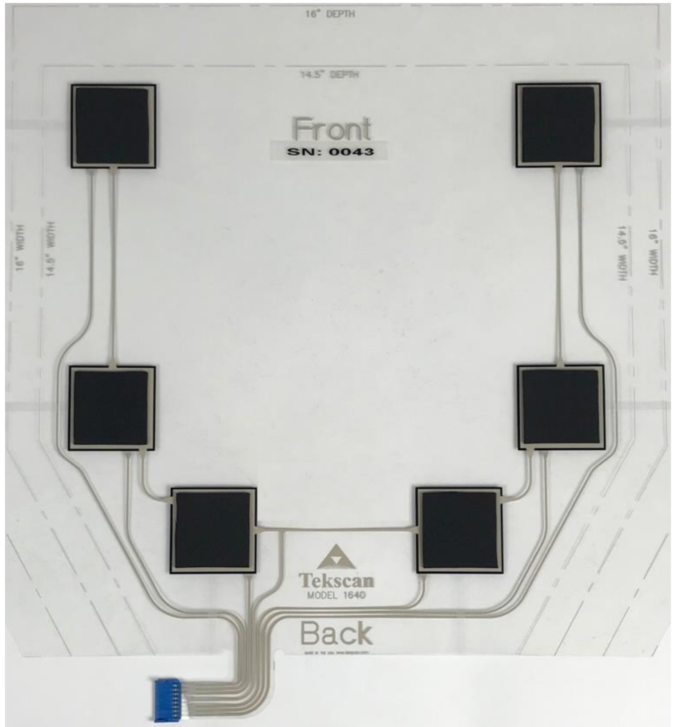
Aim 1. Multiple change point detection in streaming data settings


Change detection
Unsupervised change detection
Improved change detection
Machine Learning Models
Label efficiency
Robust to distribution shifts

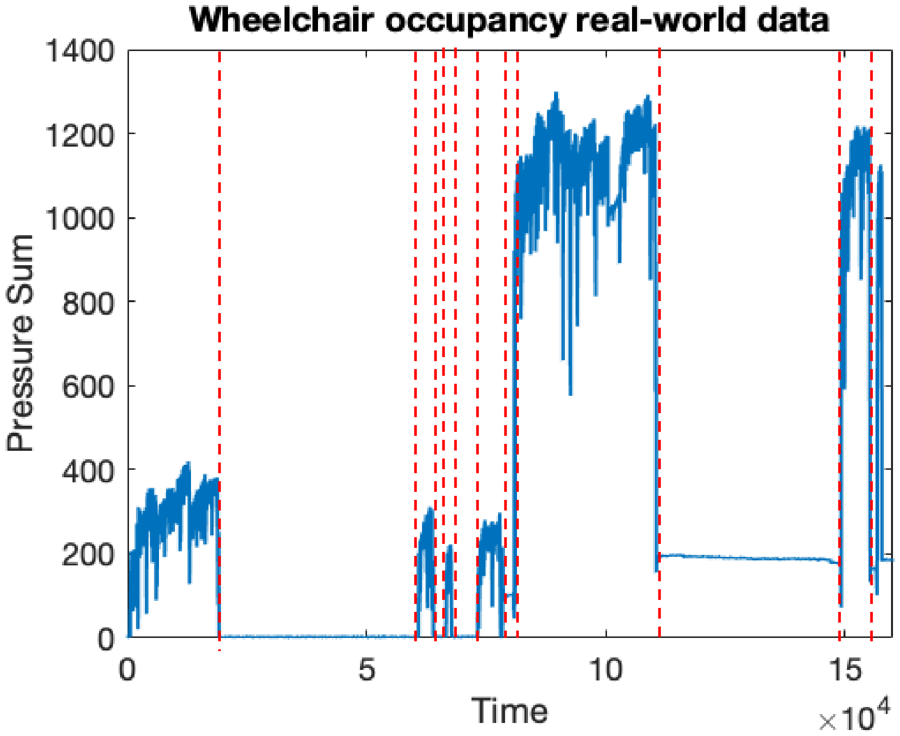
Sequential Change Detection

Background: Identify *multiple* change points sequentially within streaming manner

1. Wheelchair activity tracker[1]



2. Pressure sensor mat
(beneath seat cushion)



3. Pressure sensor readings

[1] Sonenblum et. al.

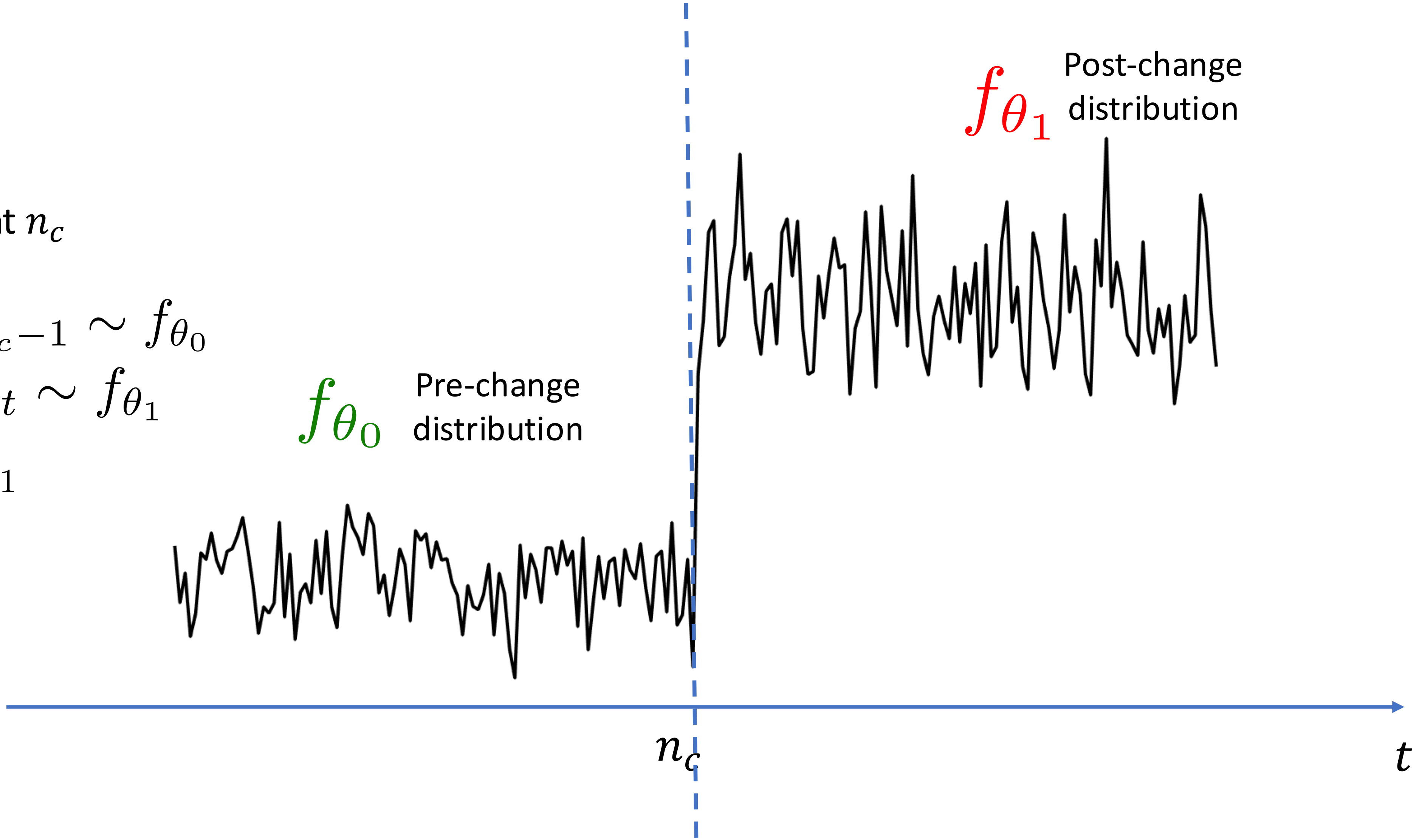
Change Point Detection

Change at n_c

$$x_{1,..,n_c-1} \sim f_{\theta_0}$$

$$x_{n_c,..,t} \sim f_{\theta_1}$$

$$\theta_0 \neq \theta_1$$



Sequential Change Point Detection

Testing at instance t for a change point before instance t

Null hypothesis: no-change
- All instances x_i i.i.d. ~ f_{θ_0} = $\mathcal{N}(\mu_0, \sigma_0^2)$

$$\mathcal{L}(\mathcal{H}_0|X) = \prod_{i=1}^{t} f_{\theta_0}(x_i)$$

Alternate hypothesis: change at n_c:
- Instances x_1. to $x_{n_c-1} \sim f_{\theta_0}$ = $\mathcal{N}(\mu_0, \sigma_0^2)$
- Instances $x_{n_c}$to x_t $\sim f_{\theta_1}$ = $\mathcal{N}(\mu_1, \sigma_1^2)$

$$\mathcal{L}(\mathcal{H}_1|X) = \prod_{i=1}^{n_c-1} f_{\theta_0}(x_i) \prod_{i=n_c}^{t} f_{\theta_1}(x_i).$$

Log likelihood ratio test for a change at n_c

$$\ell_{n_c}^t = \sum_{i=n_c}^{t} \log \frac{f_{\theta_1}(x_i)}{f_{\theta_0}(x_i)}.$$

Take maximum of these ratios over all change point instances

$$\ell^t = \max_{1<n_c<t} \ell_{n_c}^t$$

Compare to a threshold b to detect change

$$\ell^t > b$$

CUSUM[1]

Post-change distribution θ_1 *known*

- Recursive formulation:

$$S_t = \left(S_{t-1} + \log \frac{f_{\theta_1}(x_t)}{f_{\theta_0}(x_t)}\right)^+,$$

where,

$$(S_{t-1})^+ = \max(0, S_{t-1})$$

Compare to threshold to detect change $S_t > b$

[1] [Page, 1954, Lorden, 1971]

GLR Change Detection

Post-change distribution θ_1 *unknown*

- Use MLE to estimate post-change distribution

- Non recursive

$$\ell^t = \max_{1<n_c<t} \underbrace{\max_{\theta_t} \sum_{i=n_c}^{t} \log \frac{f_{\theta_t}(x_i)}{f_{\theta_0}(x_i)}}_{l_{n_c}^t}$$

- Compare to threshold to detect change

$$l_t > b$$

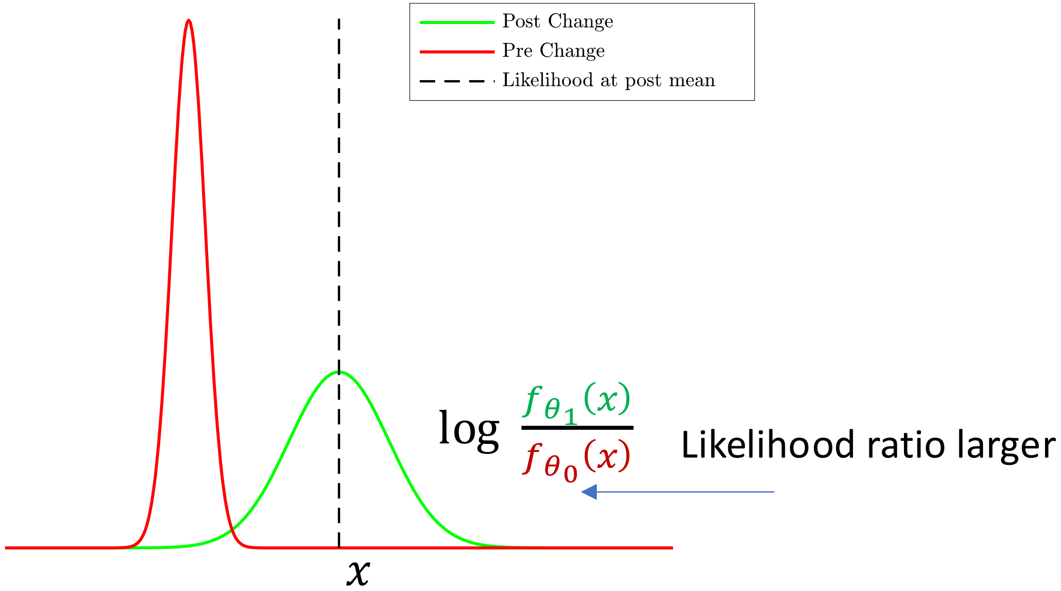
Once a change is detected, the corresponding post-change distribution is used as the pre-change distribution and the procedure is restarted to detect *multiple* change points.

Problem With CUSUM and GLR

Log Likelihood ratio is *asymmetrical*

- Example: Joint changes in mean and variance

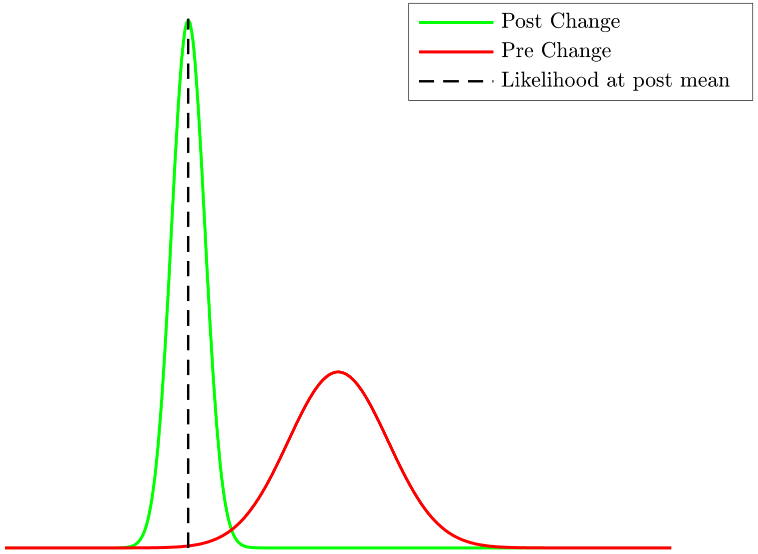
Change from low variance to high variance



$\log \frac{f_{\theta_1}(x)}{f_{\theta_0}(x)}$ Likelihood ratio larger

1. $\mathcal{N}(1,1) \rightarrow \mathcal{N}(10,3)$

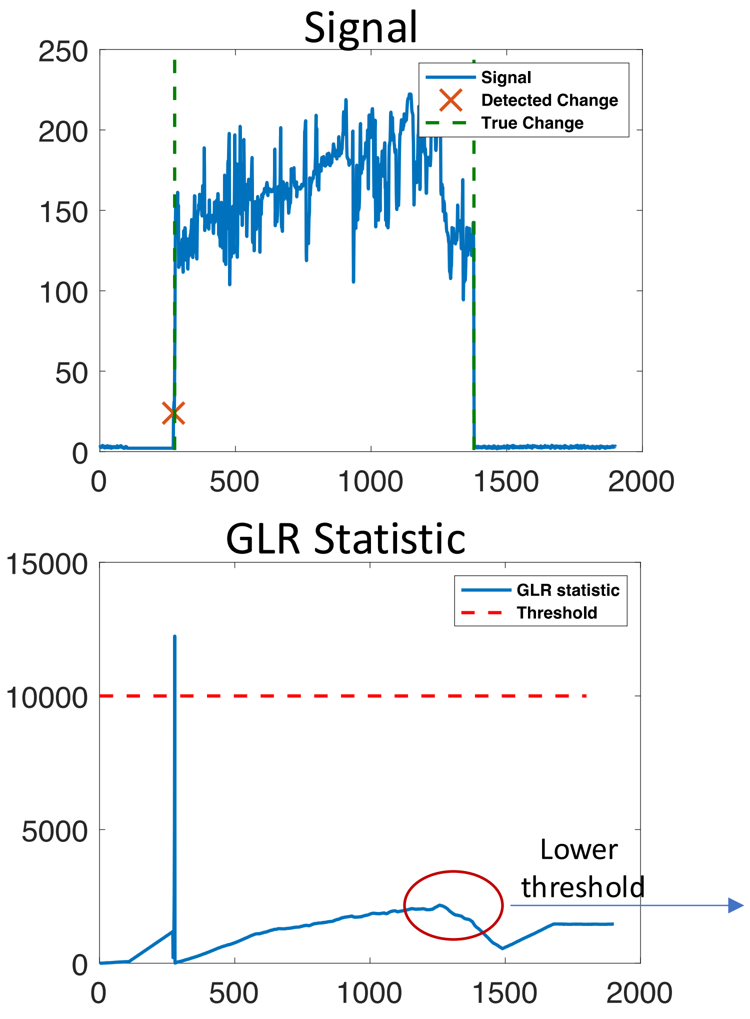
Change from high variance to low variance



2. $\mathcal{N}(10,3) \rightarrow \mathcal{N}(1,1)$

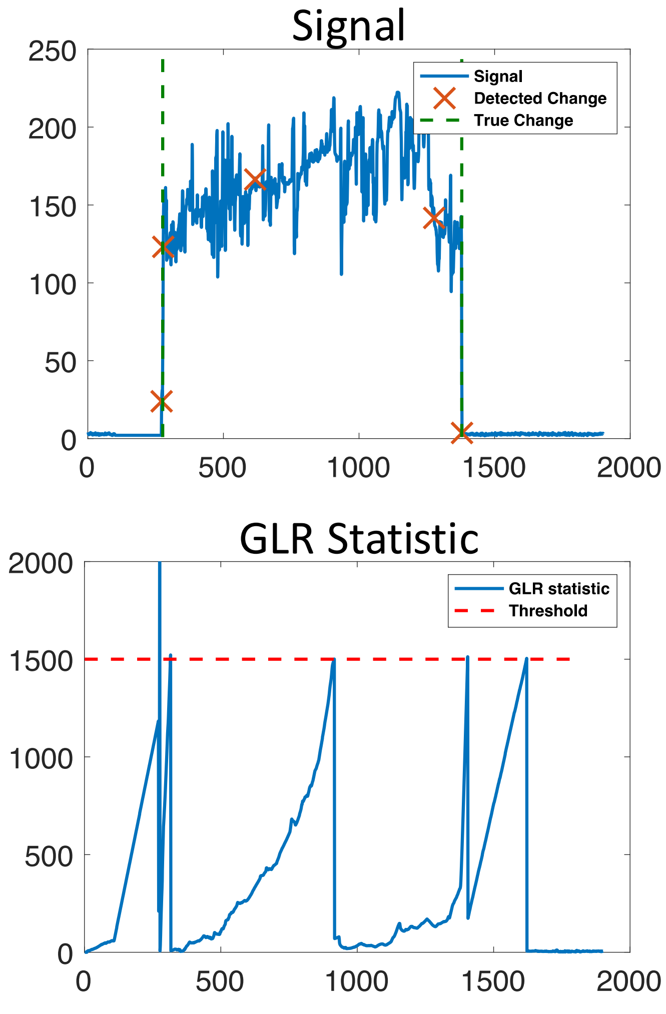
Asymmetric Change Statistic

In streaming settings, difficult to set detection threshold b



Lower threshold

Missed change points



False change points

Sequential Change Point Detection

CUSUM, GLR are based on likelihood ratio methods and are quick to detect changes

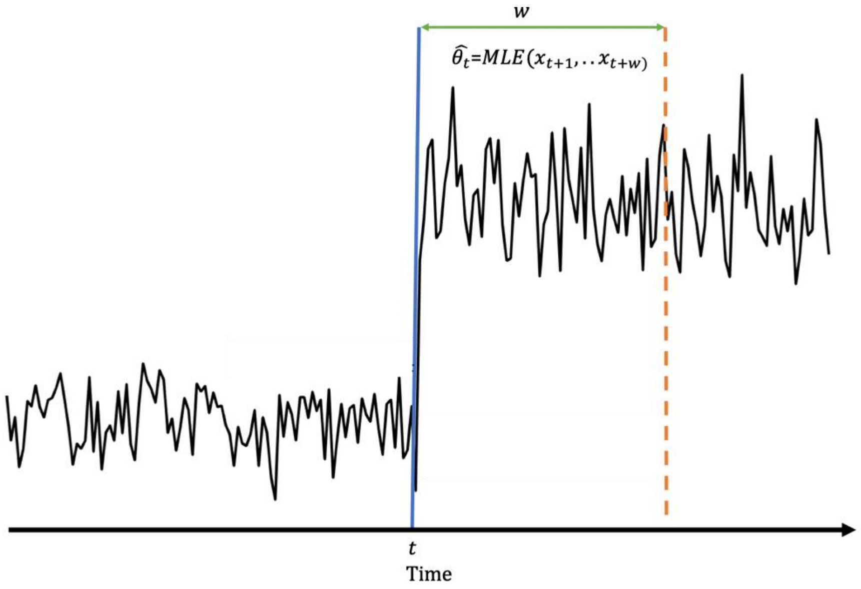
These methods, however, are *asymmetrical* which makes it difficult to set a detection threshold a priori to detect *multiple* change points

Objective: Develop symmetrical sequential change detection and provide results that relate detection delay and false alarm rate

Proposed Method

Data Adaptive Symmetrical CUSUM (DAS-CUSUM)

Estimate post-change $\hat{\theta}_t$ using window of length *w*



$$S_t = (S_{t-1})^+ + s_t$$

$$S_t = (S_{t-1})^+ + \log \frac{f_{\hat{\theta}_t}(x_i)}{f_{\theta_0}(x_i)} + \underbrace{D_{KL}(f_{\theta_0}(x) || f_{\hat{\theta}_t}(x))) - v}_{\text{Two additional terms}}$$

$$\mathbb{E}_{\theta_1}[s_t] = D_{KL}(\theta_1, \theta_0) + D_{KL}(\theta_0, \theta_1) - v$$

$$\mathbb{E}_{\theta_0}[s_t] = -v$$

EDD vs ARL

***Theorem* 1**: For a given ARL (γ), *the expected detection delay* (*EDD*) *for a change from distribution* $x \sim \mathcal{N}(\theta_0)$ *to* $x \sim \mathcal{N}(\theta_1)$ *, which is unkown and estimated using a window of length w* (*as w*→ ∞) *, is given by*:

$$\mathrm{EDD} = \frac{\log\gamma + o(1)}{\delta_0\left(D_{KL}(\theta_1,\theta_0) + D_{KL}(\theta_0,\theta_1)\right) + \log(1-\frac{{\delta_0}^2}{w})} + w.$$

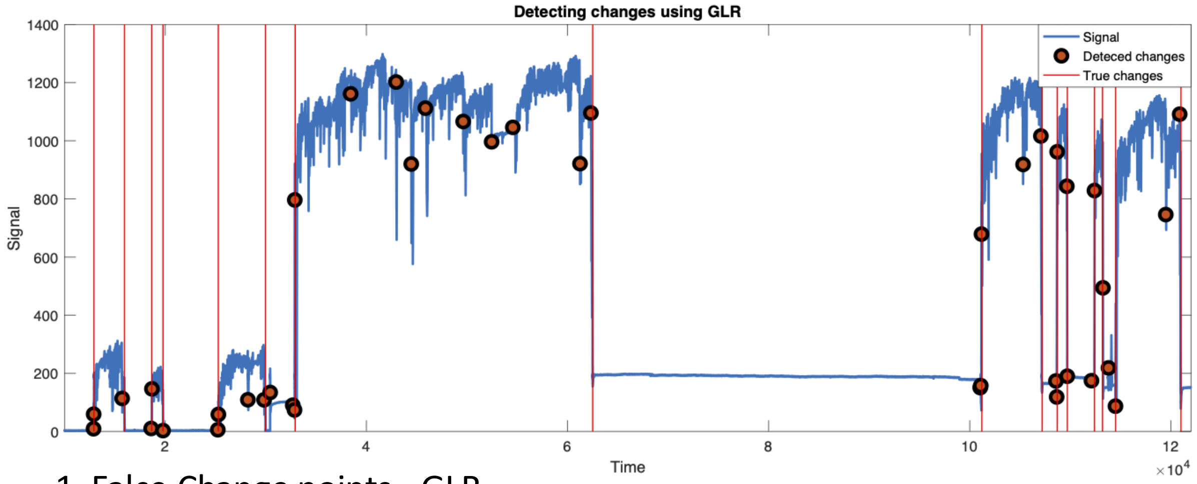
γ : ARL
θ_0 : pre-change dist.
θ_1 : post-change dist.
D_{KL}: KL Divergence
w: window length to estimate post-change dist.

CUSUM result[1] where post-change distribution known: $\mathrm{EDD} = \dfrac{\log\gamma + o(1)}{D_{KL}(\theta_1,\theta_0)}$

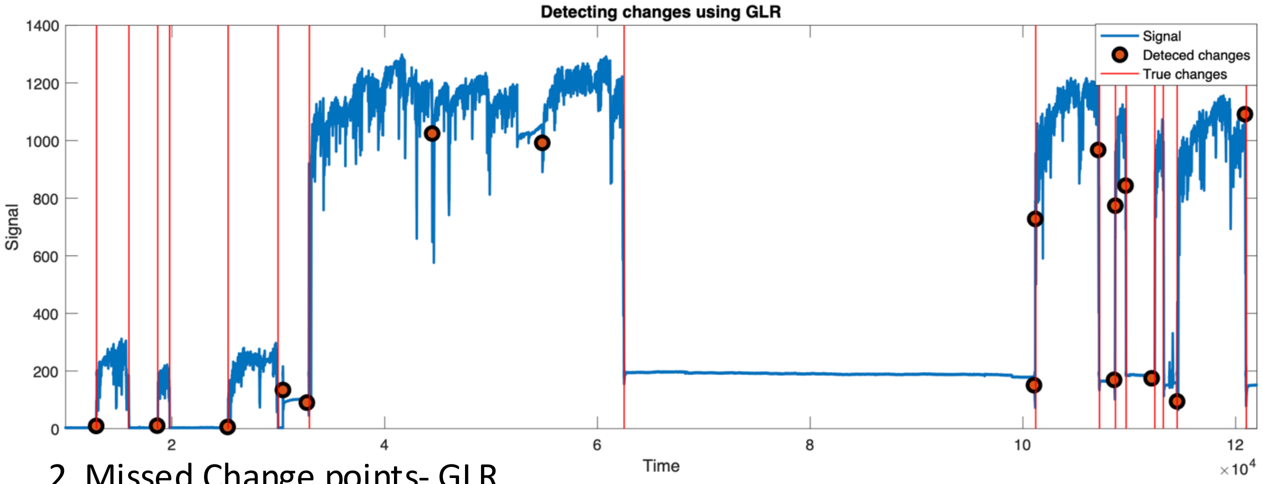
- *Symmetrical* term in the denominator

[1] [Lorden, 1971]

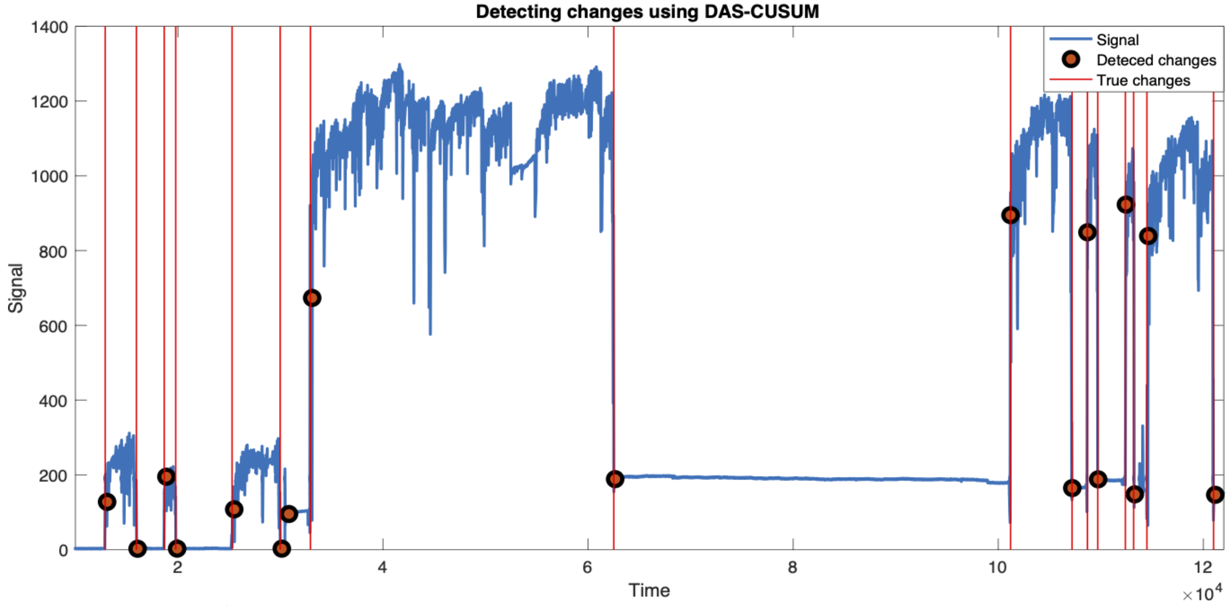
Real-world Results



1. False Change points - GLR



2. Missed Change points- GLR



3. Correct detection – DAS-CUSUM

Aim Summary & Contributions

1.Proposed a symmetrical sequential change detection procedure

2. A symmetrical statistic makes it easier to set a threshold for detecting multiple changes

3. Provided theoretical & empirical results that relate detection delay with false alarms

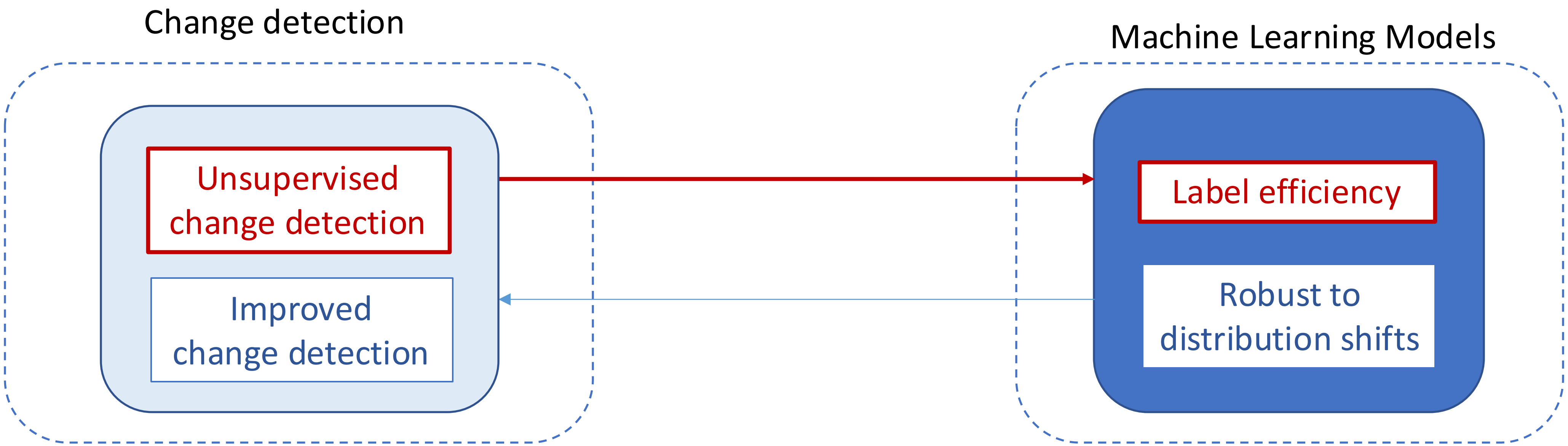
4. Used change detection to solve a real-world problem where supervised classifiers can fail

Publication

N. Ahad, M. A. Davenport, and Y. Xie, “Data-adaptive symmetric CUSUM for sequential change detection”,
Sequential Analysis, 43 (1), pp. 1-27, 2024

Improving Machine Learning Models

Aim 2. Using change points for semi-supervised learning


Change detection
Unsupervised change detection
Improved change detection
Machine Learning Models
Label efficiency
Robust to distribution shifts

Semi-supervised Learning

- Obtaining labeled data is expensive
  - Difficult to recruit participants for providing controlled, labeled data
  - Difficult to annotate labels in large unstructured datasets
- Unlabeled data
  - Inexpensive and widely available

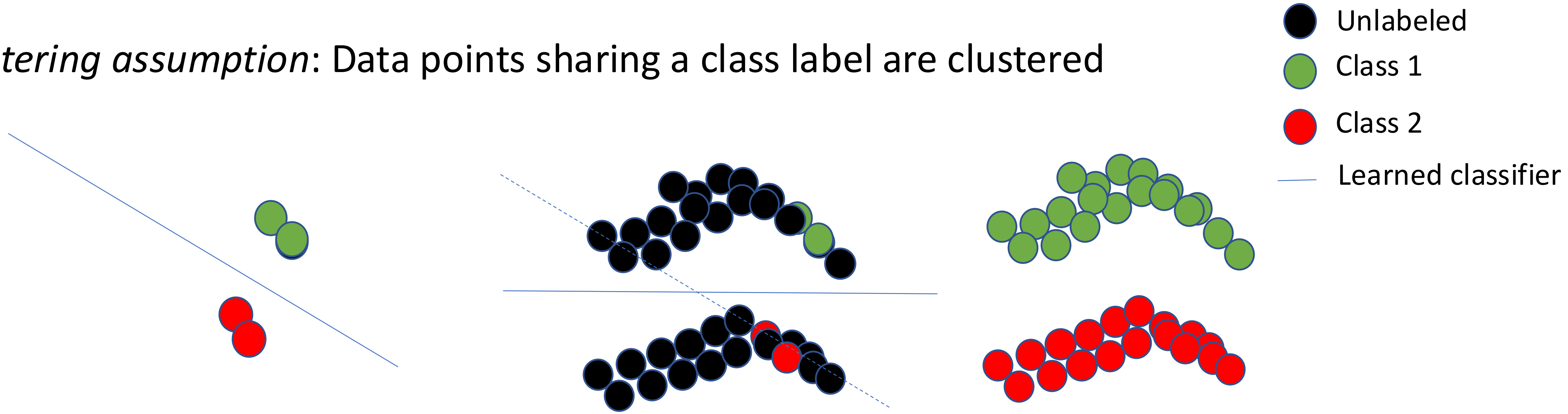
Can we utilize unlabeled data to improve classification performance?

Possible solution:

Semi-supervised learning:
Leveraging unlabeled data to complement labeled data for improved learning

Semi-supervised Learning

Clustering assumption: Data points sharing a class label are clustered



1. Using only available labels

2. Using unlabeled data

True Labels

Semi-sup methods perform better if data is well clustered!

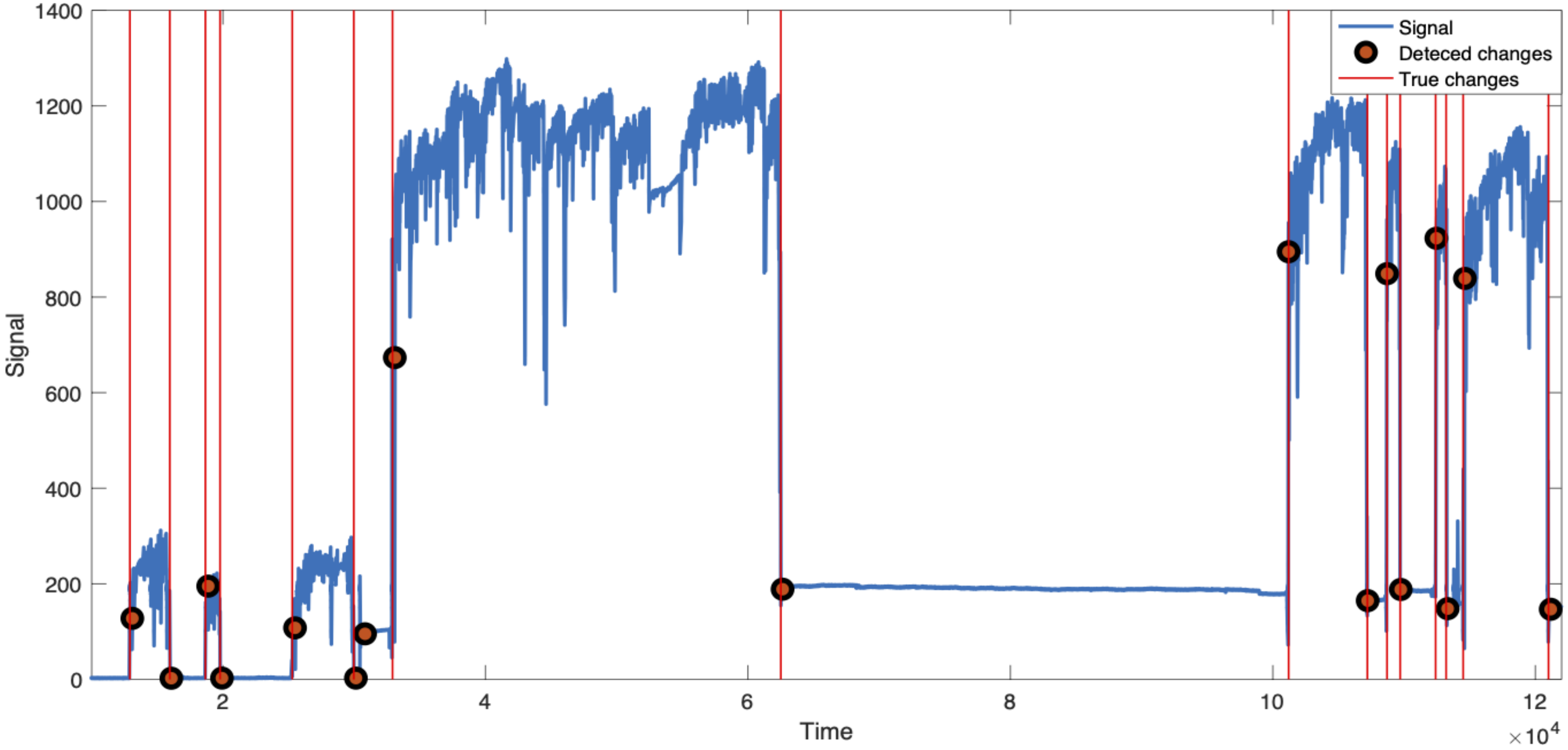
Objective: Use unsupervised change detection to obtained clustered representations for improved semi-supervised sequence learning

Proposed Method

First Step

First run an unsupervised change detection procedure on time series

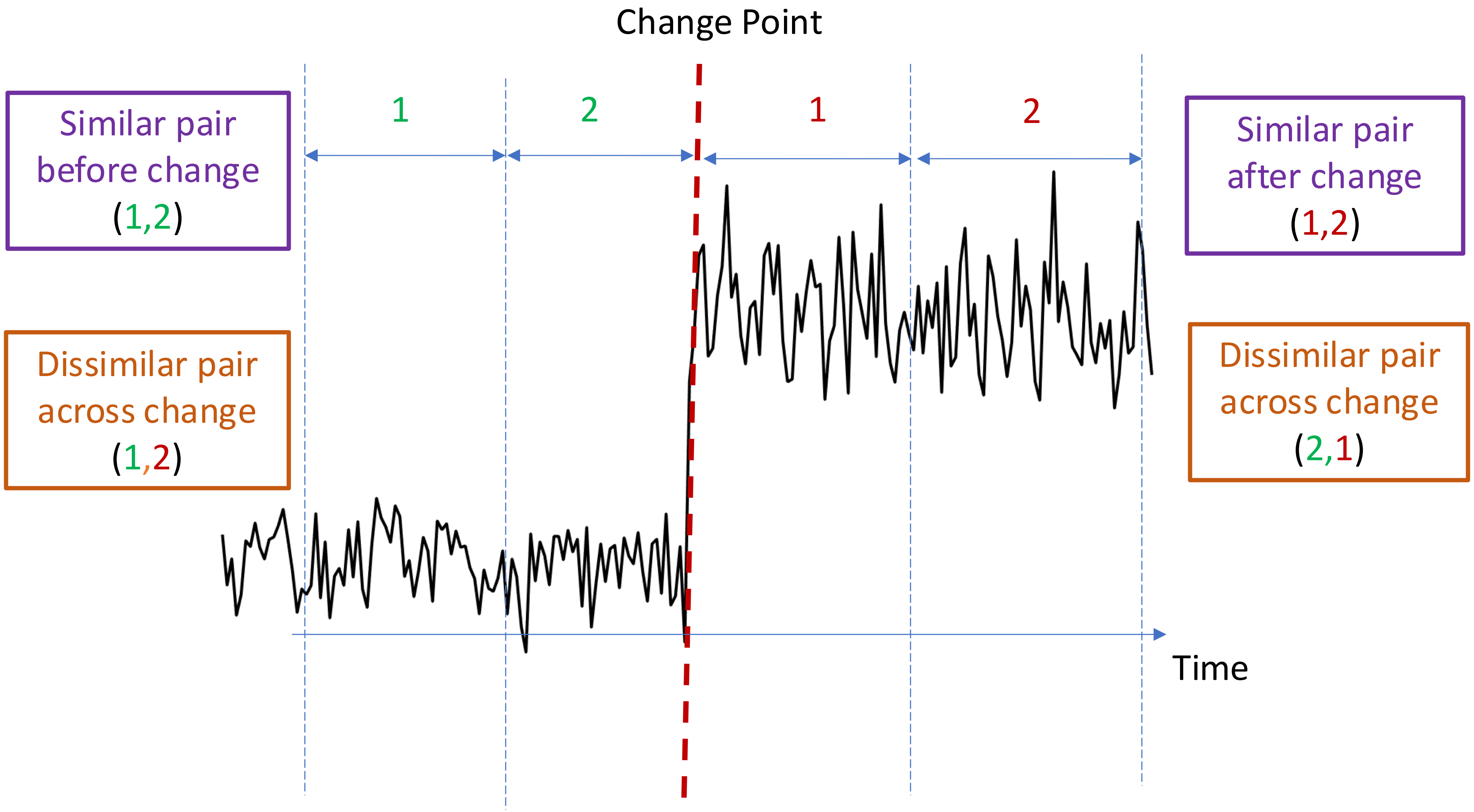
Note: Any change point detection method can be used as long if it identifies changes correctly



Obtain Similar/Dissimilar pairs

Second Step

Obtain similar dissimilar sequence pairs from detected change points



Change Point
1
2
1
2
Similar pair before change (1,2)
Dissimilar pair across change (1,2)
Similar pair after change (1,2)
Dissimilar pair across change (2,1)
Time

Learning Semi-supervised Network Representations

Pairs from true labels and pairs from change points used to learn neural network

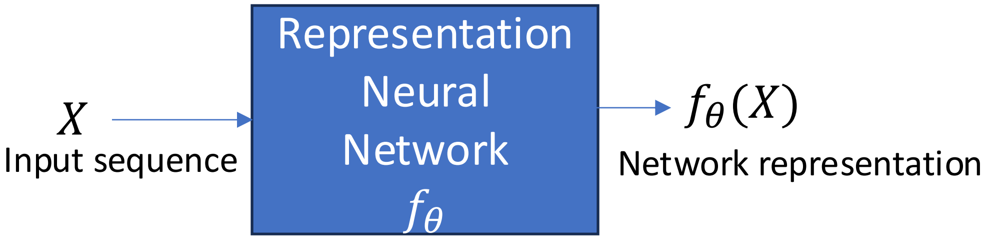
$$\mathcal{L} = \underbrace{\mathcal{L}_{pair}^{\text{Sup}}}_{\text{Pairs from labels}} + \underbrace{\mathcal{L}_{pair}^{\text{Unsup}}}_{\text{Pairs from Change Points}}$$

$$\mathcal{L}_{pair} = \begin{cases} D\left(f_\theta\left(X_1\right), f_\theta\left(X_2\right)\right) & \text{If } \left(X_1, X_2\right) \text{ similar} \\ k - D\left(f_\theta\left(X_1\right), f_\theta\left(X_2\right)\right) & \text{If } \left(X_1, X_2\right) \text{ dissimilar} \end{cases}$$

$D:$ Distance measure

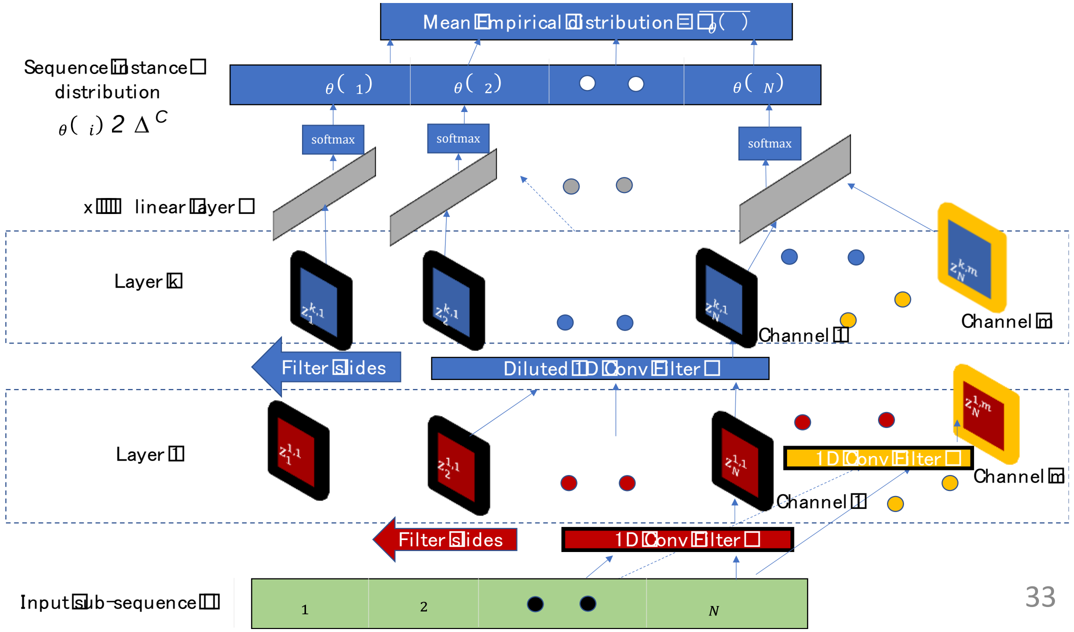
$k:$ Constant

$f_\theta:$ Neural Network



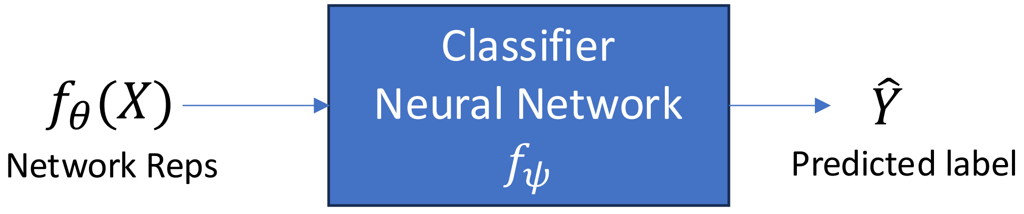
Type equation here.

1D CNN used as f_θ



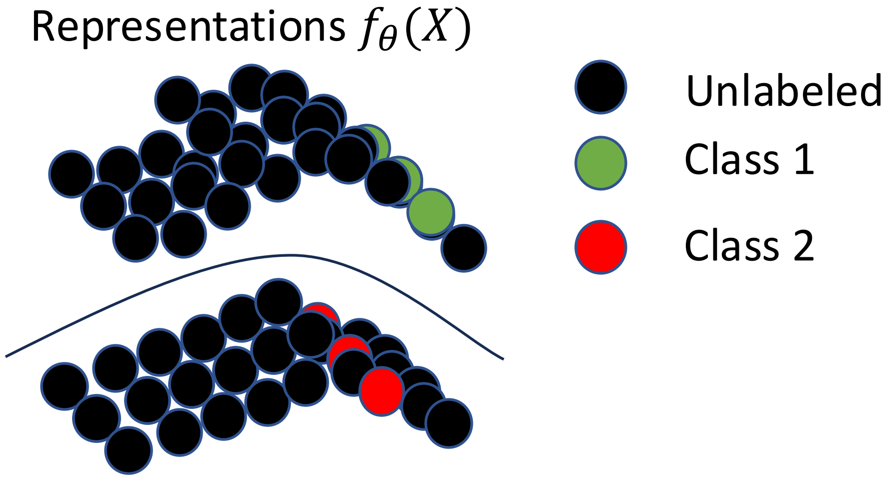
Train Classifier on Top of Representations

Train another classification network f_ψ to predict labels for learned representations $f_\theta(X)$, where X is the input



$$\mathcal{L}_{\mathrm{C}}(\psi) = \frac{1}{|\mathcal{X}_L|}\underbrace{\sum_{(X,Y)\in\mathcal{X}_L}\mathcal{L}_{\mathrm{CE}}(X,Y)}_{\text{Cross entropy (labeled data)}} + \frac{\lambda_C}{|\mathcal{X}_U|}\underbrace{\sum_{X\in\mathcal{X}_U}\mathcal{L}_{\mathrm{NE}}(X)}_{\text{Negative entropy (unlabeled data)}}.$$

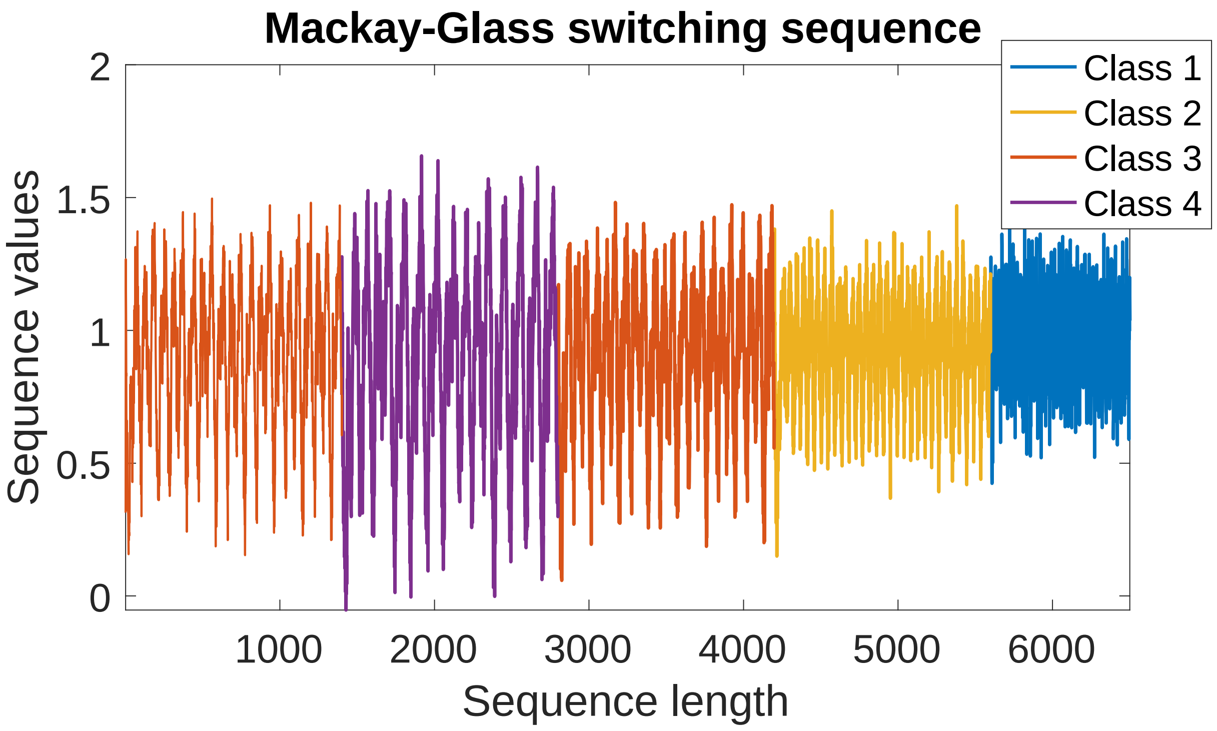
X: Input
Y: True label



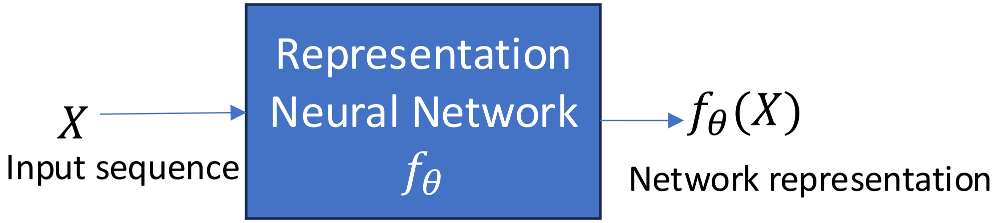
f_ψ for learning a classification boundary ⌒

Synthetic Experiments

Synthetic sequence that switches between different classes



Synthetic Experiments



X

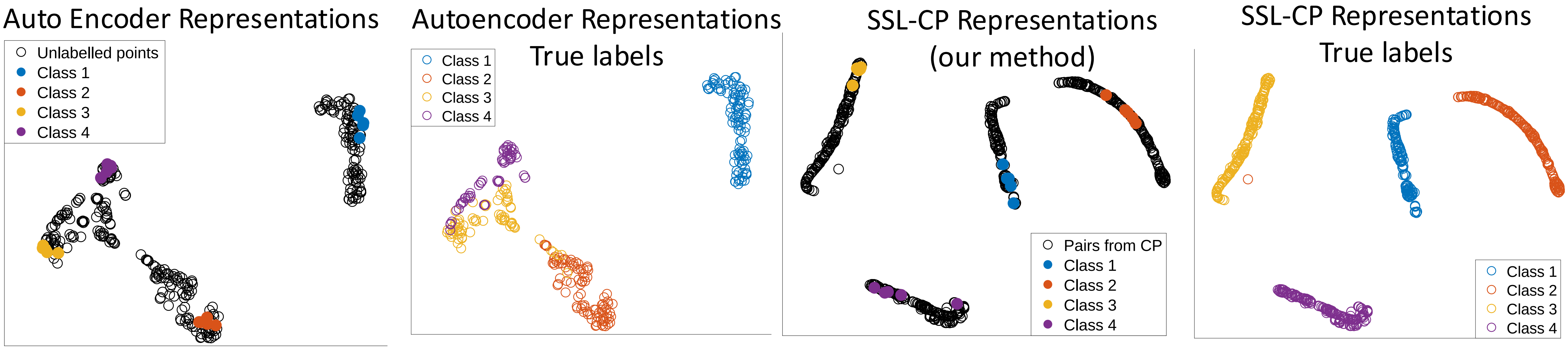
Input sequence

Representation Neural Network f_θ

$f_\theta(X)$

Network representation

T-SNE Visualizations



Auto Encoder Representations

Autoencoder Representations True labels

SSL-CP Representations (our method)

SSL-CP Representations True labels

F1 scores
(Higher is better)

Model	20 labels	30 labels	60 labels
Supervised	0.55 ±0.07	0.86 ± 0.04	0.95 ± 0.02
Autoencoder	0.73 ± 0.04	0.90 ± 0.02	0.98 ± 0.01
SSL-CP	0.96 ±0.02	0.98 ± 0.01	0.99 ± 0.01
SSL-CP (ER)	**0.99 ± 0.02**	**0.99 ± 0.01**	**0.99 ± 0.01**

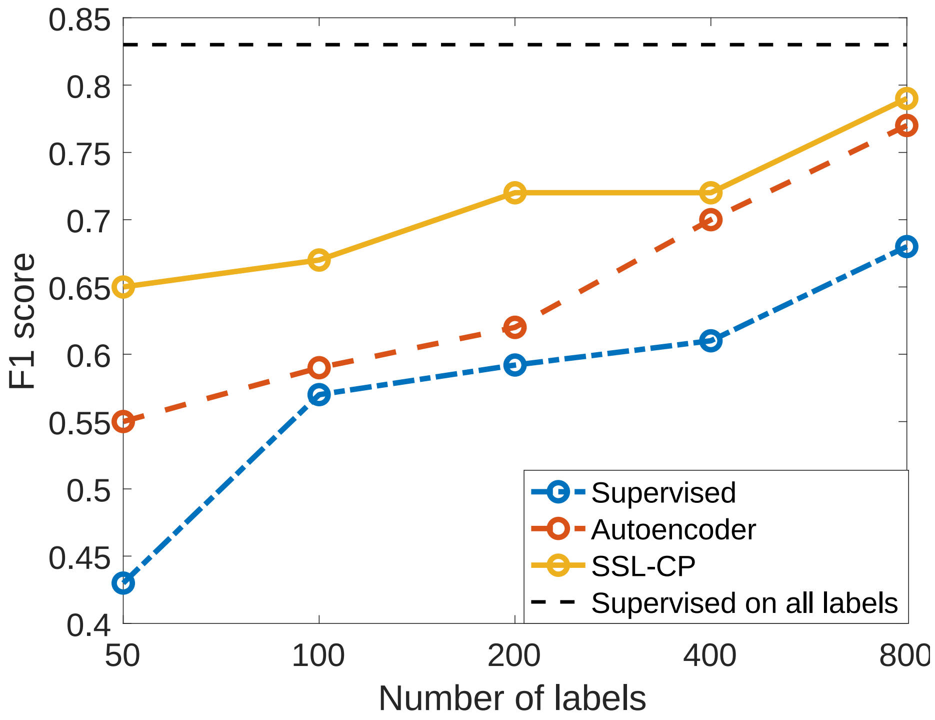
Real-world Experiments

Human activity recognition for fitness tracking:

- 3 axis accelerometer mounted on user’s arm
- Users do 6 activities
  (Walk, run, stair up, stair down, stand, sit)

F1 scores (Higher is better)

Method	F1 score
Supervised	0.45 ± 0.04
Autoencoder	0.54 ± 0.02
SSL-CP (All users)	0.53 ± 0.03
SSL-CP (Filtered users)	0.65 ± 0.02
SSL-CP (True CPs, all users)	0.66 ± 0.01
SSL-CP-ER (Filtered users)	0.65 ± 0.01
SSL-CP-ER (True CPs, all users)	**0.69** ± 0.01



Aim Summary and Contributions

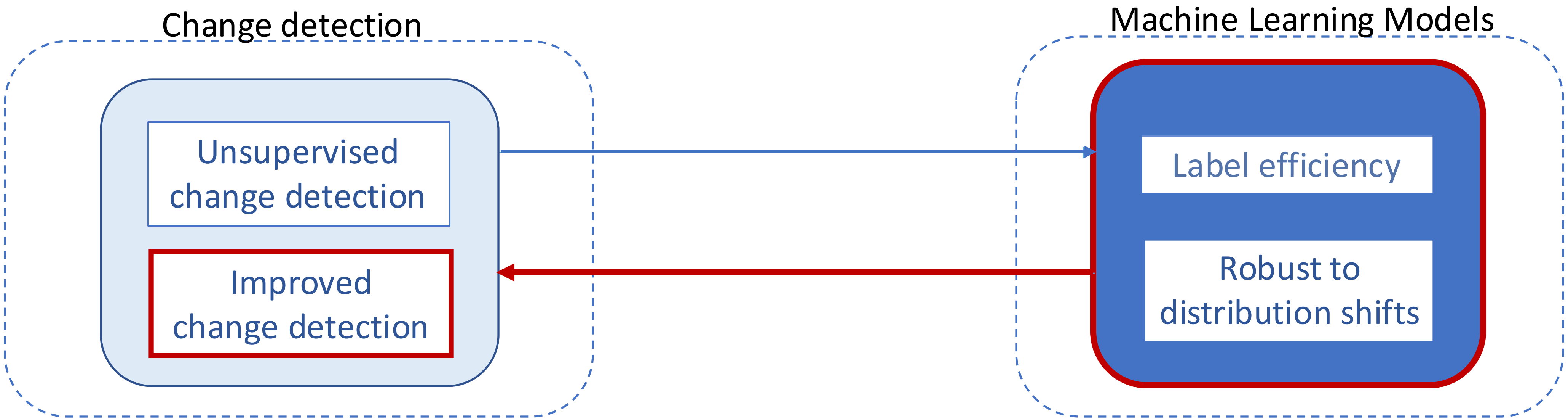
- Method that can use detected change points for learning semi-supervised neural network representations
- First method, that we know of, that proposes using unsupervised change-point detection for semi-supervised learning

Publication

N. Ahad and M. Davenport, “Semi-supervised Sequence Classification through Change Point Detection”, *AAAI Conference on Artificial Intelligence*, 2021.

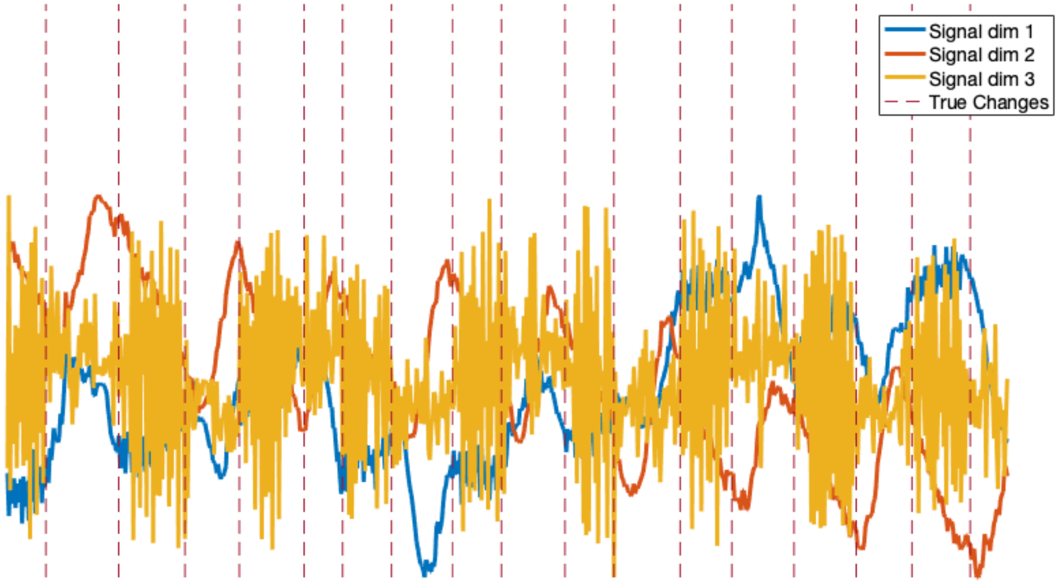
Improving Change Detection

Aim 3 . Using supervision from available changes to improving change detection


Change detection
Unsupervised change detection
Improved change detection
Machine Learning Models
Label efficiency
Robust to distribution shifts

How to Use Available True Change Points ?

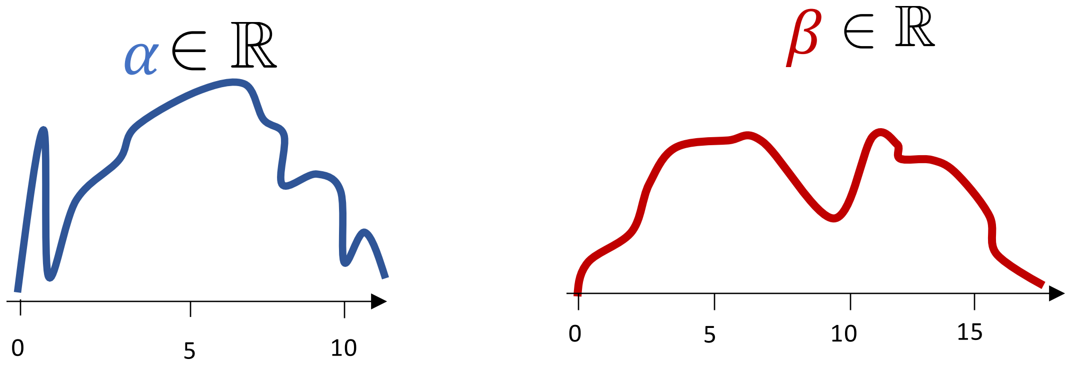
Many signals require us to *detect some kinds* of changes while *ignoring other kinds of* changes



Can we use true change information to improve change detection performance?

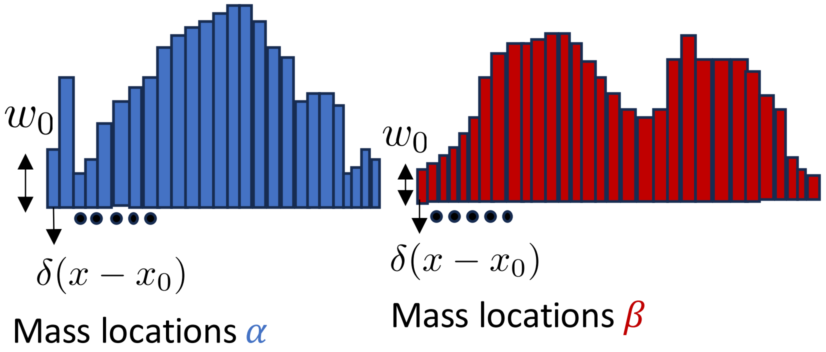
Comparing Two Distributions

First, how to compare how dissimilar two sets of points are with no distributional assumptions?

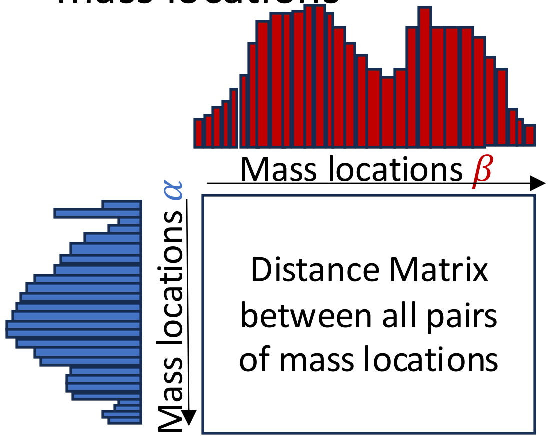


One way: Entropic regularized Wasserstein distances (Also known Sinkhorn Divergence)

1. Compute Histograms and obtain mass locations

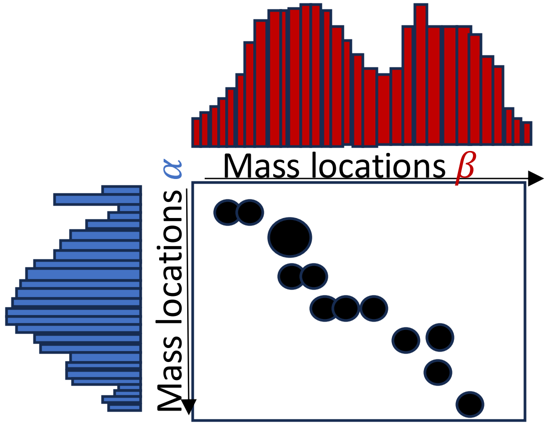


2. Compute Cost Matrix between pairwise mass locations



Distance Matrix between all pairs of mass locations

3. Compute Transport Plan using Cost between mass locations



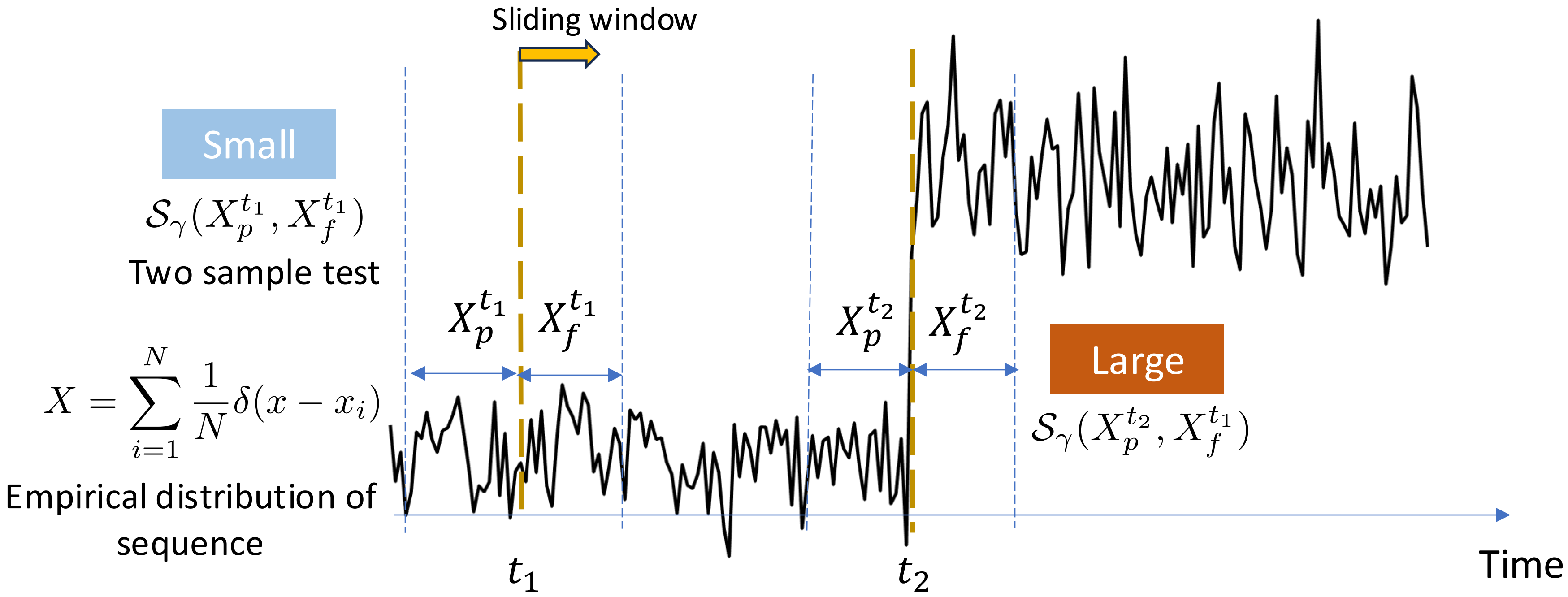
4. Get Sink Divergence:

Total cost to move mass locations from α to β

Detecting Change Points Through Sinkhorn Divergence

Sinkhorn Divergence ($\mathcal{S}_\gamma$)

- For measuring the difference between two distributions



Sliding window

Small

$\mathcal{S}_\gamma(X_p^{t_1}, X_f^{t_1})$

Two sample test

$$X = \sum_{i=1}^{N} \frac{1}{N} \delta(x - x_i)$$

Empirical distribution of sequence

Large

$\mathcal{S}_\gamma(X_p^{t_2}, X_f^{t_1})$

Detect change point at instances where Sinkhorn divergence greater than a specified threshold

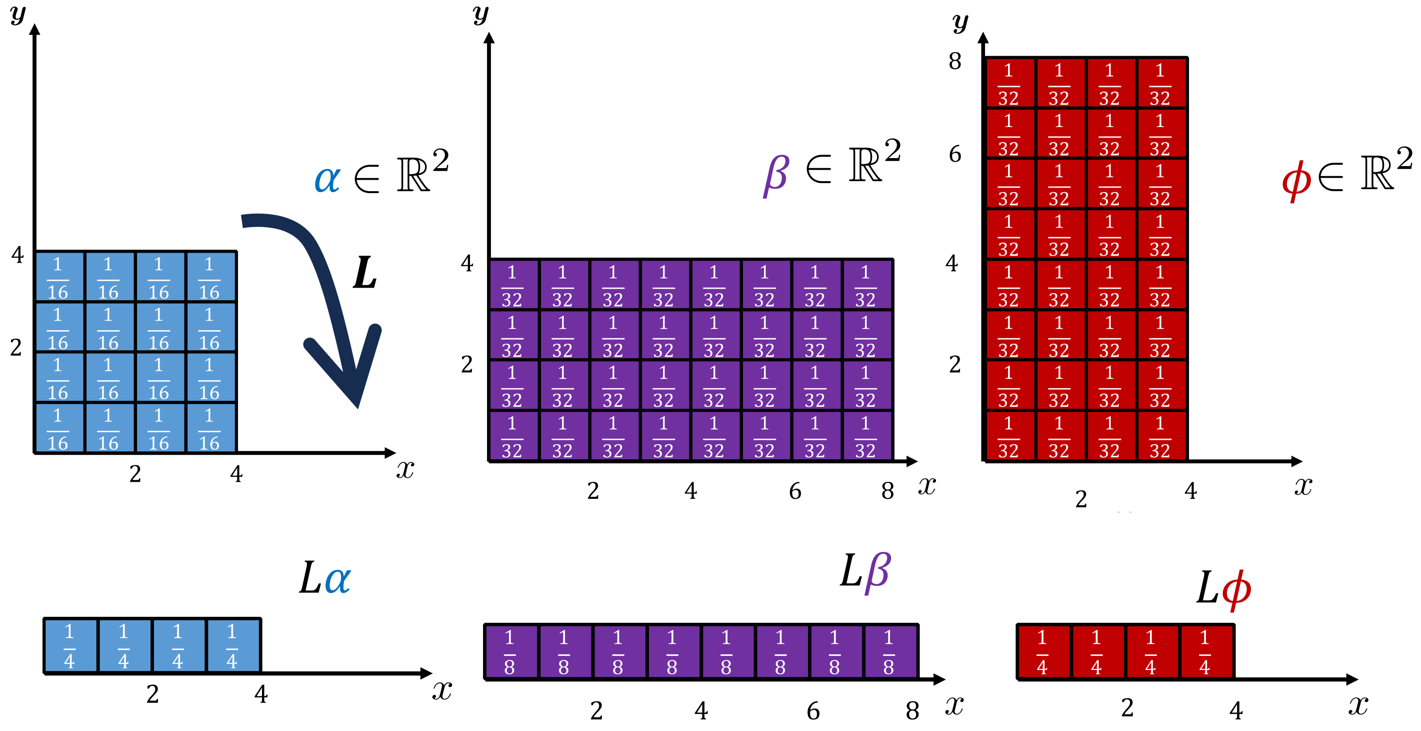
How to Use Supervision?

Three different distributions

Supervised Information Requiring

$\mathcal{S}_\gamma(\alpha, \beta)$ ↑
should be large

$\mathcal{S}_\gamma(\alpha, \phi)$ ↓
should be small



We can learn a transformation $\boldsymbol{L}$ that projects the y dimension of all points onto x axis

$\mathcal{S}_\gamma(L\alpha, L\beta)$
Large!

$\mathcal{S}_\gamma(L\alpha, L\phi)$
Small!

***L* meets the supervised requirement**

☑

Supervision for Learning L?

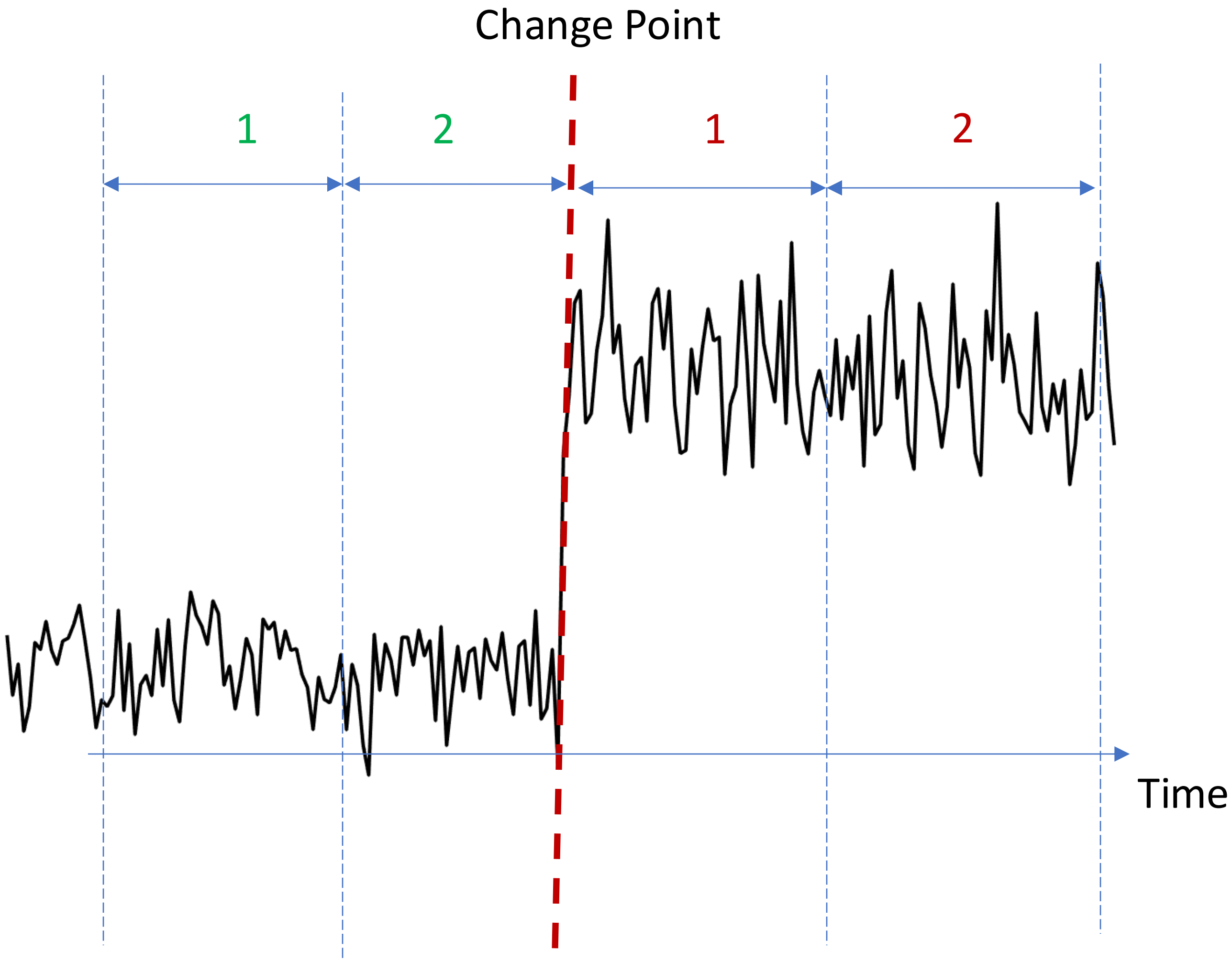
Use Change Points to obtain supervision for learning $\boldsymbol{L}$

$$\underbrace{\mathcal{S}_{L,\gamma}(1,2)}_{\text{Sink Div between similar}} < \underbrace{\mathcal{S}_{L,\gamma}(2,1)}_{\text{Sink Div between dissimilar}}$$

Triplet pair (2,1,1)

$$\mathcal{S}_{L,\gamma}(1,2) < \mathcal{S}_{L,\gamma}(1,2)$$

Triplet pair (2,1,1)

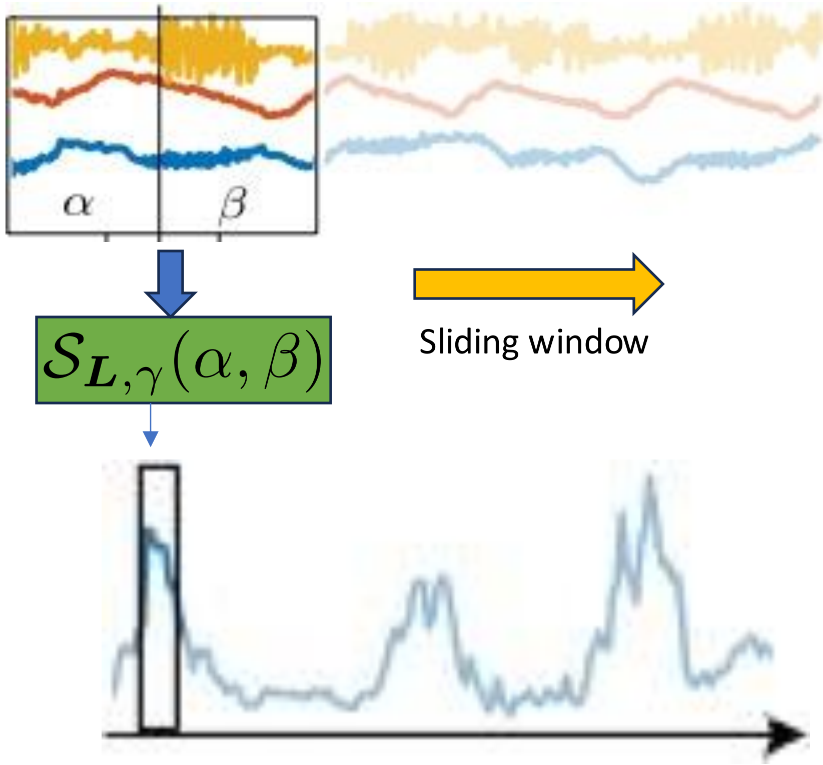


How to Use Available True Change Points ?

Learn $\boldsymbol{L}$ by minimizing triplet loss

$$\min_{\boldsymbol{L}} \sum_{i\in\text{Trip pairs}} [c - (\underbrace{\mathcal{S}_{\boldsymbol{L},\gamma}(\boldsymbol{X}_i,\boldsymbol{X}_{i_d})}_{\text{Dissimilar pair in triplet}} - \underbrace{\mathcal{S}_{\boldsymbol{L},\gamma}(\boldsymbol{X}_i,\boldsymbol{X}_{i_s})}_{\text{Similar pair in triplet}})]^+$$

Once L learned, use $\mathcal{S}_{\boldsymbol{L},\gamma}$ in two sample tests over sliding windows to detect change points



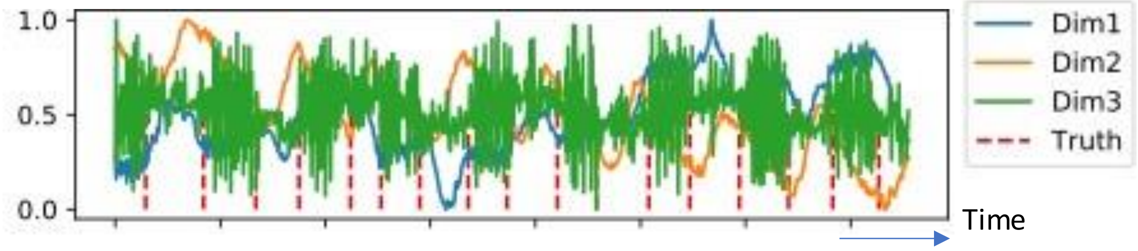
$\mathcal{S}_{\boldsymbol{L},\gamma}(\alpha,\beta)$

Change statistic
(from $\mathcal{S}_{\boldsymbol{L},\gamma}(\alpha,\beta)$)

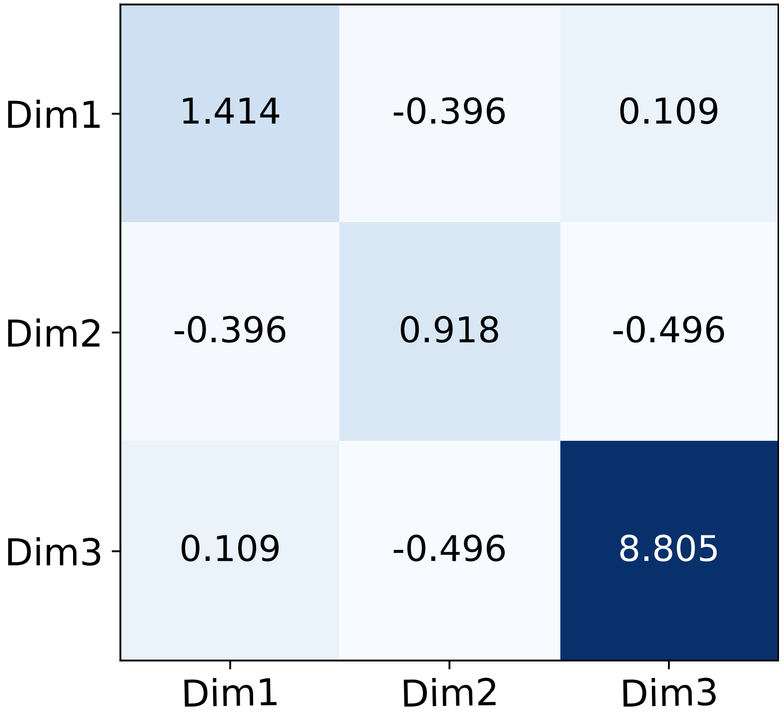
Detect change point whenever
$\mathcal{S}_{\boldsymbol{L},\gamma} > \tau$ (threshold)

Learned Metric Improves Performance

Bee Dance Dataset



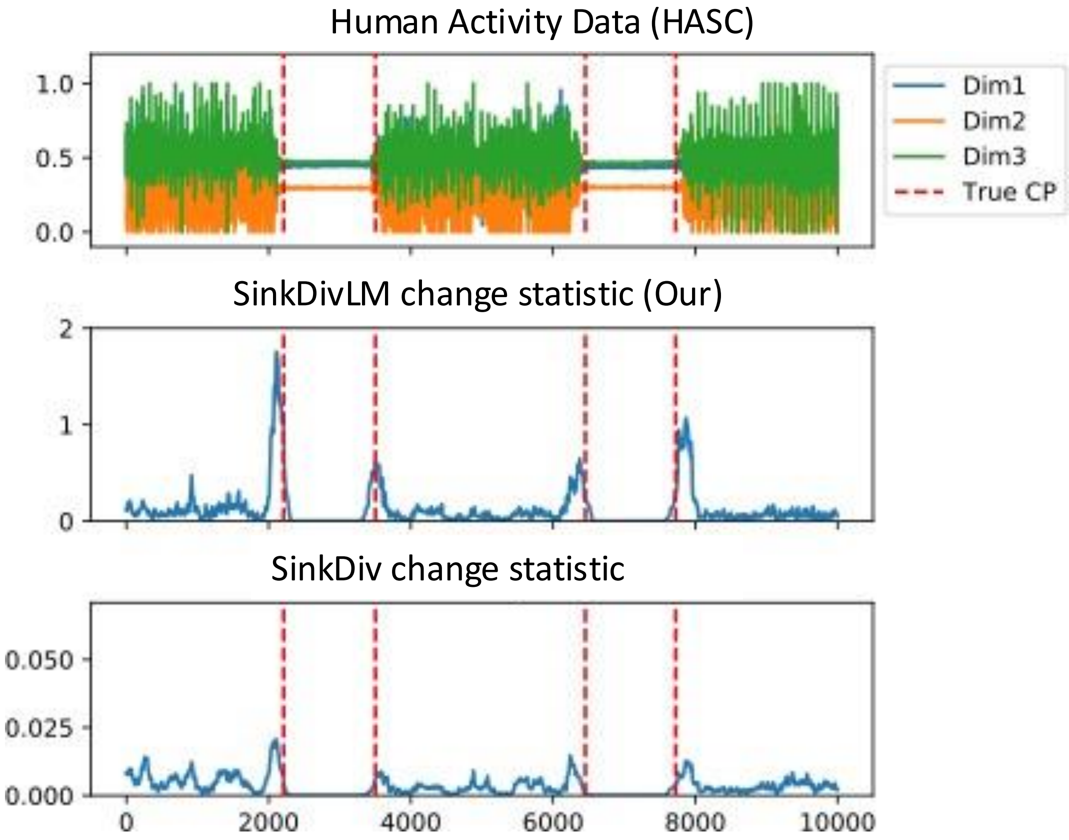
Learned Metric $L^T L$ for Bee Dance



	Dim1	Dim2	Dim3
Dim1	1.414	-0.396	0.109
Dim2	-0.396	0.918	-0.496
Dim3	0.109	-0.496	8.805

Learned metric improves performance

Learned Metric Improves Performance

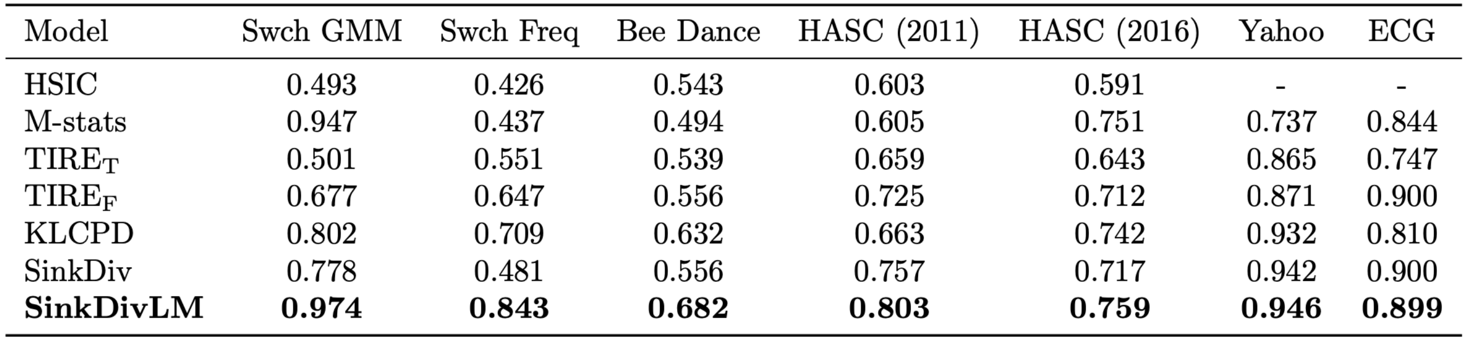


Human Activity Data (HASC)

SinkDivLM change statistic (Our)

SinkDiv change statistic

Area Under the ROC Curve (AUC) scores
(Higher is better)



Model	Swch GMM	Swch Freq	Bee Dance	HASC (2011)	HASC (2016)	Yahoo	ECG
HSIC	0.493	0.426	0.543	0.603	0.591	-	-
M-stats	0.947	0.437	0.494	0.605	0.751	0.737	0.844
TIRE_T	0.501	0.551	0.539	0.659	0.643	0.865	0.747
TIRE_F	0.677	0.647	0.556	0.725	0.712	0.871	0.900
KLCPD	0.802	0.709	0.632	0.663	0.742	0.932	0.810
SinkDiv	0.778	0.481	0.556	0.757	0.717	0.942	0.900
SinkDivLM	**0.974**	**0.843**	**0.682**	**0.803**	**0.759**	**0.946**	**0.899**

Our Method (SinkDivLM) does much better!

Flexible Framework

$$\min_{\boldsymbol{L}} \sum_{i \in \text{Trip pairs}} \left[c - \left(\mathcal{S}_{\boldsymbol{L},\gamma}(\boldsymbol{x}_i, \boldsymbol{x}_{i_d}) - \mathcal{S}_{\boldsymbol{L},\gamma}\left(\boldsymbol{x}_i, \boldsymbol{x}_{i_s}\right)\right)\right]^+ + \lambda \|\boldsymbol{L}\|_1$$

L1 regularization leads to a sparse $\boldsymbol{L}$

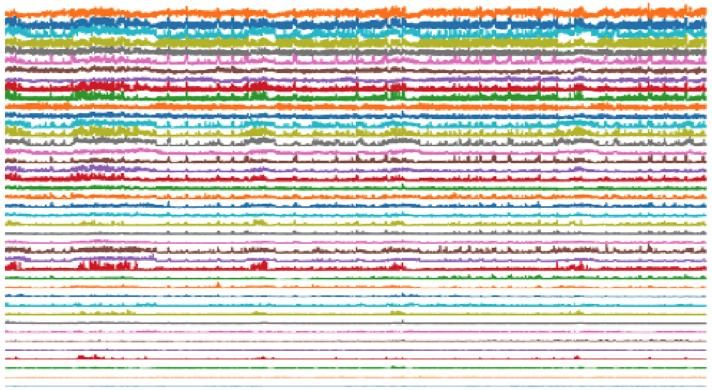
Sparse $\mathbf{L}$ can help interpretability!

Important channels in time series that are responsible for causing changes!

Neural Sleep Stage Dataset

- Electrode arrays implanted in mice hippocampus record neural firing data
- Spike sorted data and binned 42 neurons
- 12 hour annotated recordings where mice switch between REM,nREM and awake states
- Available true change points to:

  1. Improve state sleep state change detection
  2. Learn a sparse ground metric $\boldsymbol{L}^T\boldsymbol{L}$ which helps interpret what neurons are responsible for changes



[1] Keith B Hengen, Alejandro Torrado Pacheco, James N McGregor, Stephen D Van Hooser, and Gina G Turrigiano. Neuronal firing rate homeostasis is inhibited by sleep and promoted by wake. *Cell,* 165(1):180– 191, 2016.

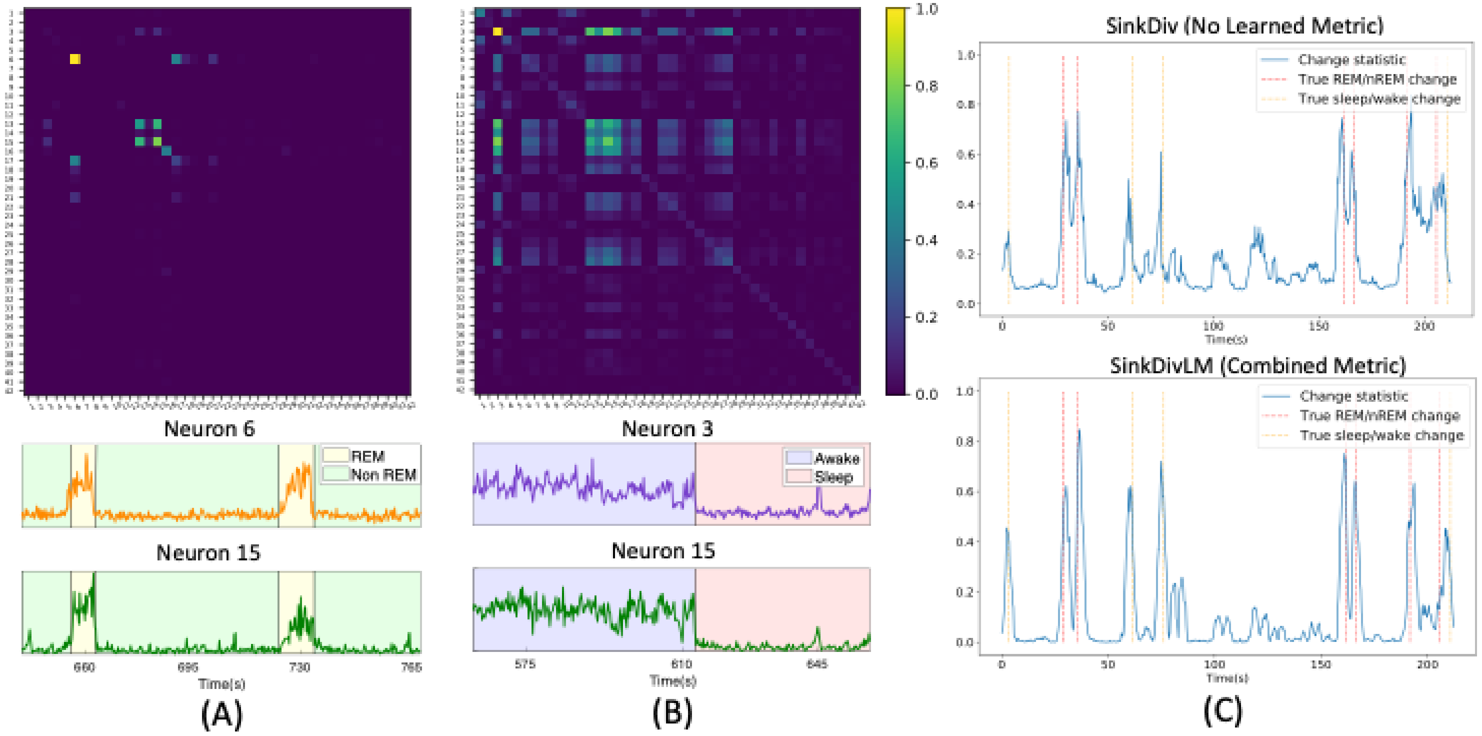
Sparse Interpretable Metric

REM/nREM changes

Sleep/Awake changes

Learned Metric $L^T L$

Top 2 identified Neurons



	SinkDiv	SinkDivLM
Trained on sleep/wake		
Sleep/wake	0.58	**0.85**
REM/nREM/wake	0.79	**0.72**
Trained on REM/nREM		
REM/nREM	0.92	**0.95**
REM/nREM/wake	0.79	**0.82**
Combined sleep metrics		
REM/nREM/wake	0.79	**0.85**

Aim Summary and Contributions

- A novel method that proposes learning a metric for change detection
- Improves change detection performance
- Provides interpretable metric that helps identify underlying changes of interest

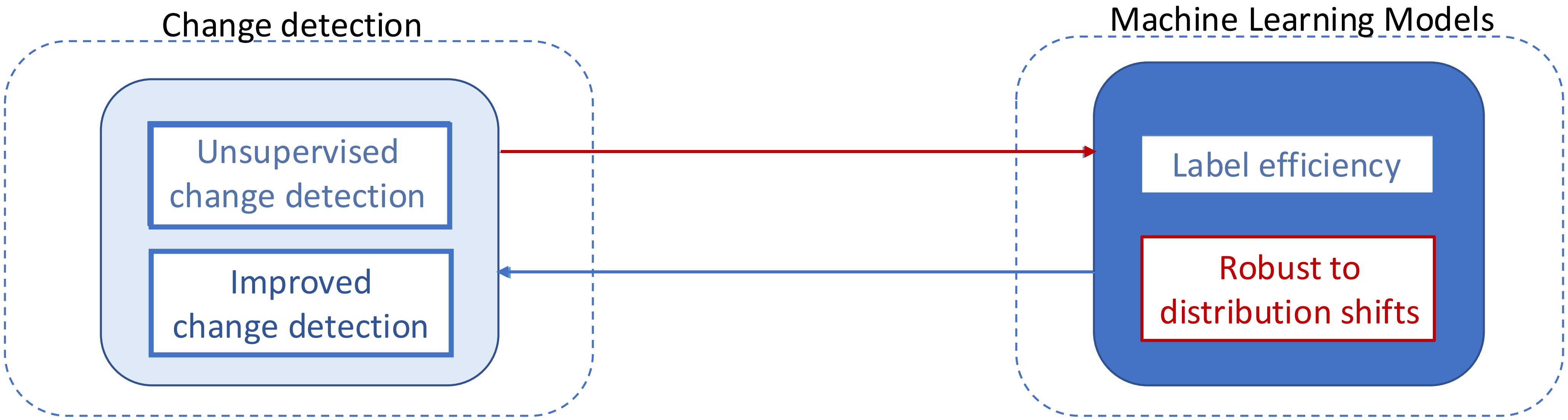
Publications

C. Uzray, N. Ahad, M. Abazou, E. Dyer, " Detecting change points in neural population activity with contrastive metric learning", *IEEE Conference on Neural Engineering*, 2023

N. Ahad, E. Dyer, K. Hengen, Y. Xie, M. Davenport, " Learning Sinkhorn Divergences for Change Point Detection", *In revision, IEEE Transactions on Signal Processing*

Improving Machine Learning Models

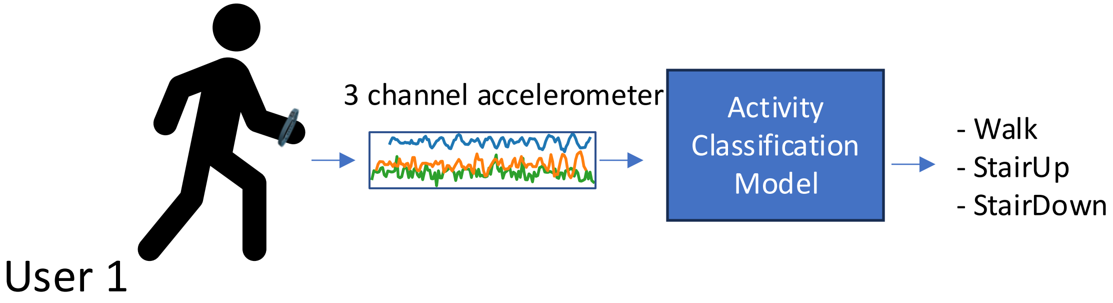
Aim 4 . Unsupervised domain adaptation for time series through selective channel masking



Change detection
Unsupervised change detection
Improved change detection
Machine Learning Models
Label efficiency
Robust to distribution shifts

Generalizing trained ML models on Newer Data

Train Machine learning model
on available labels from user 1



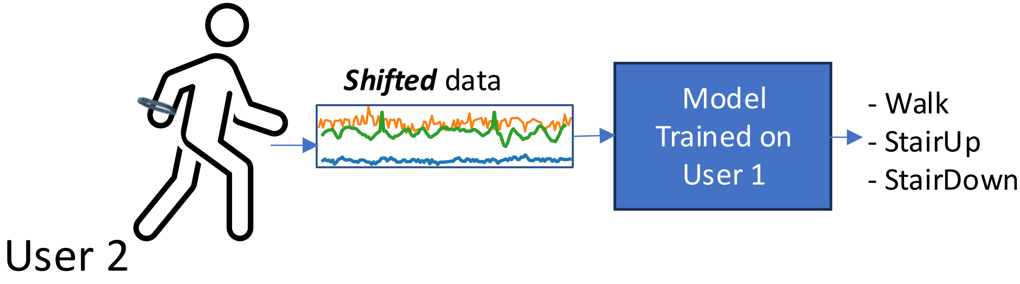
User 1

3 channel accelerometer

Activity Classification Model

- Walk
- StairUp
- StairDown

Test model on user 2
Using model trained on user 1



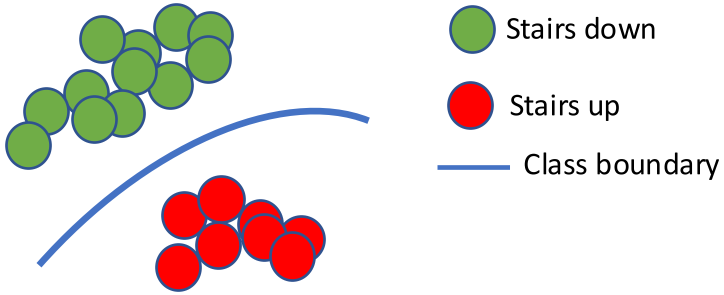
User 2

Shifted data

Model Trained on User 1

- Walk
- StairUp
- StairDown

Learned activity classifier



🟢 Stairs down

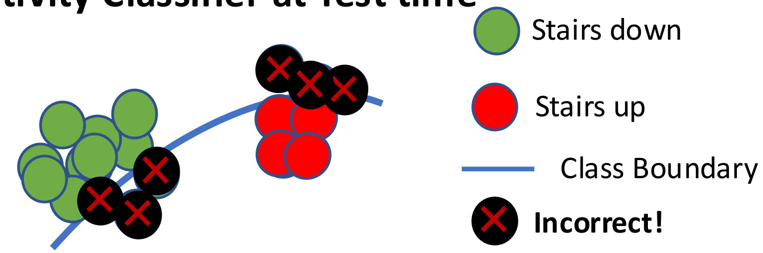
🔴 Stairs up

— Class boundary

Representation space

Deploy trained classifier

Activity Classifier at Test time



🟢 Stairs down

🔴 Stairs up

— Class Boundary

❌ **Incorrect!**

Representation space

Trained Machine learning models can fail to generalize as ***test-time data distributions change!***

How to adapt and transfer a trained multi channel classification model on new data ?

Unsupervised Domain Adaptation

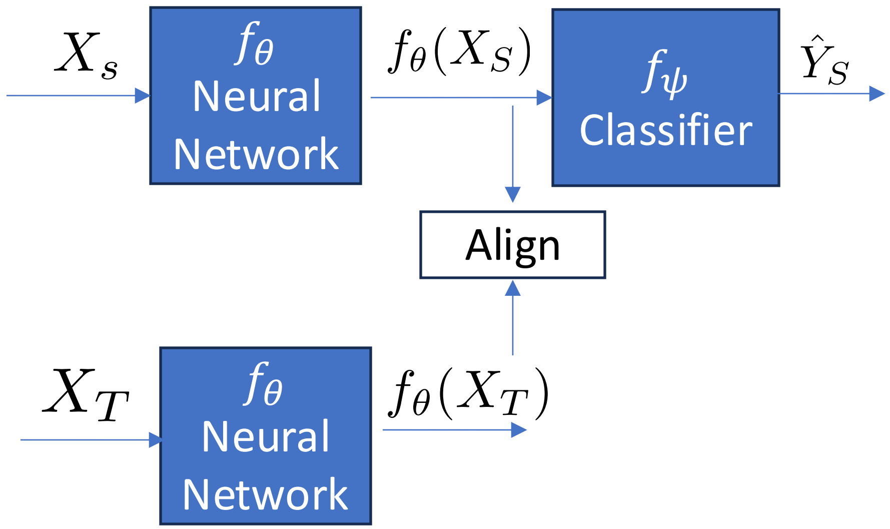
Adapt supervised source domain models to unsupervised target domains

X_S: Source domain data
Y_S: Source domain labels available
X_T : Target domain data (unlabeled)

Popular strategy for Unsupervised Domain adaptation:

1. Supervised classification on available source labels
2. Learn representations where ***source and target aligned/invariant***

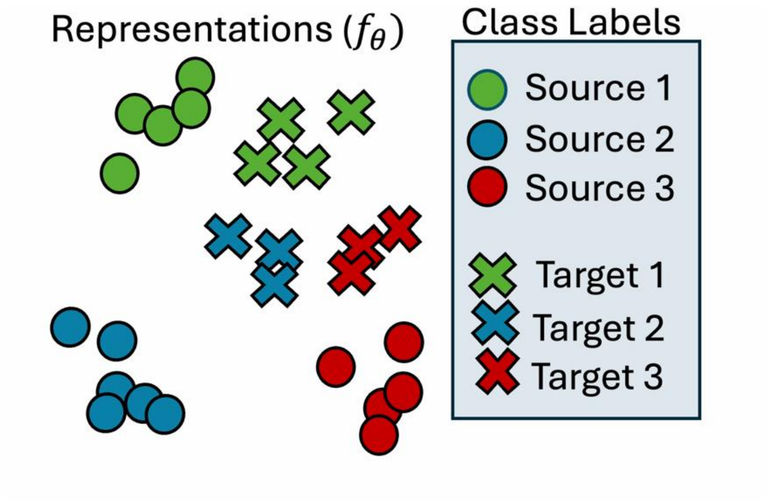
Commonly used existing approach



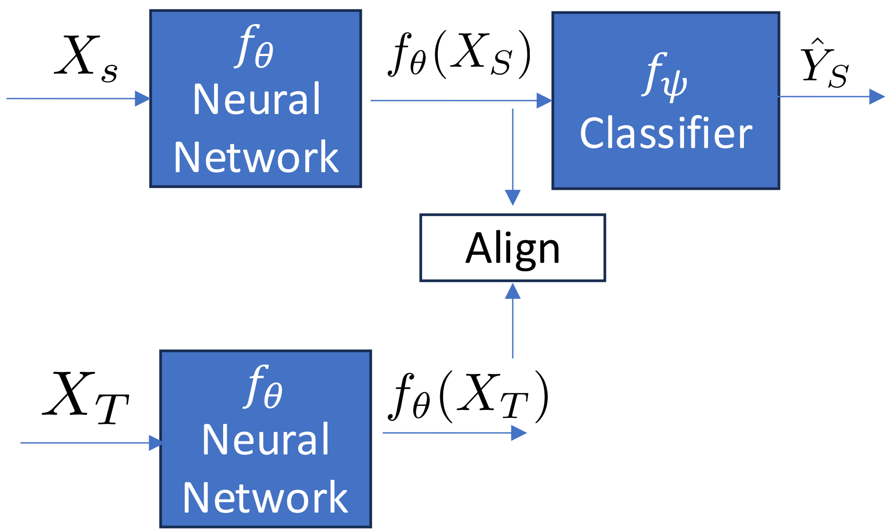
$$\underbrace{\min_{\theta,\psi} \mathcal{L}_{CE}(f_\psi(f_\theta(X_s)), Y_s)}_{\text{Supervised Cross Entropy loss on available source labels}} + \underbrace{D\left(f_\theta(X_s), f_\theta\left(X_t\right)\right)}_{\text{“Distance” or “Align” loss between source and target repres. Could be: Adversarial, MMD, Sinkhorn, etc.}}$$

Unsupervised Domain Adaptation

Toy example for aligning domains



Commonly used existing approach



X_s

f_θ Neural Network

$f_\theta(X_S)$

f_ψ Classifier

$\hat{Y}_S$

Align

X_T

f_θ Neural Network

$f_\theta(X_T)$

$$\min_{\theta,\psi} \mathcal{L}_{CE}(f_\psi(f_\theta(X_s)), Y_s) + \underbrace{D\left(f_\theta(X_s), f_\theta\left(X_t\right)\right)}$$

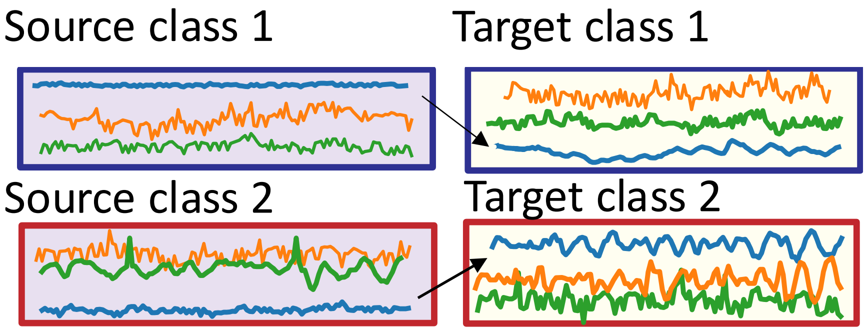
Supervised Cross Entropy
loss on available source labels

"Distance" or "Align" loss between
source and target repres.
Could be: Adversarial, MMD, Sinkhorn, etc.

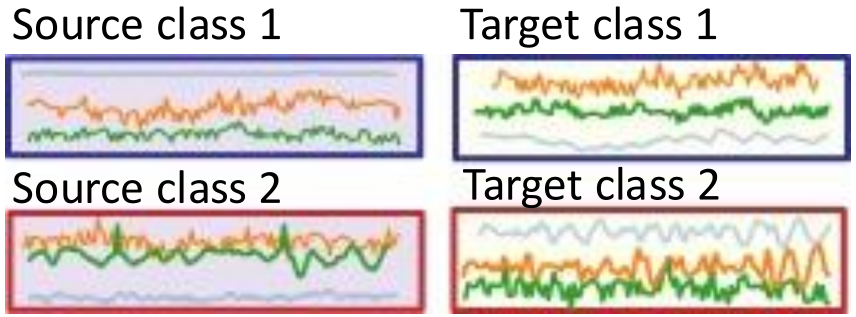
Assumption to work: ***Source and Target points for the same class closer than other classes***

Domain Shifts in Time Series are Channel

For multi channel time series, domain shifts can be more severe in some channels

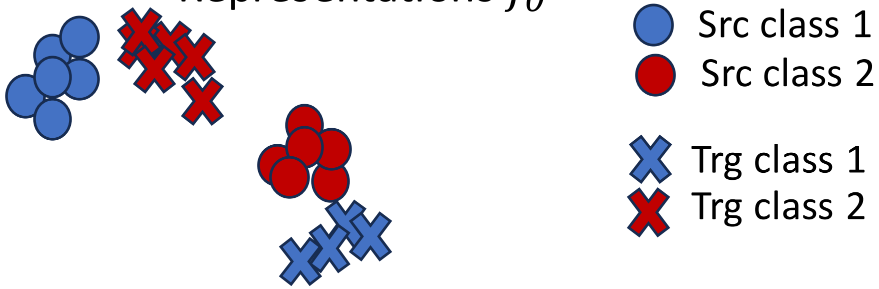


Large shift in blue channel across source and target



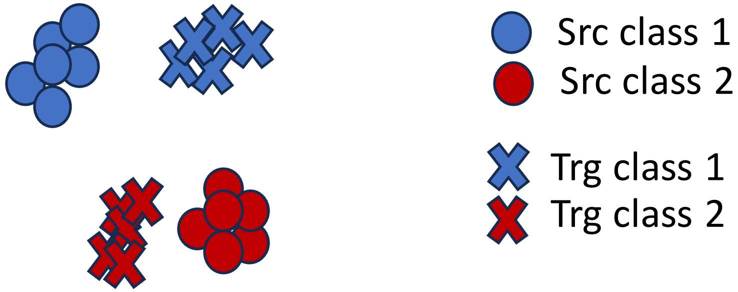
A possible solution, ignoring blue channels

Representations f_θ



Domain adaptation method likely to fail

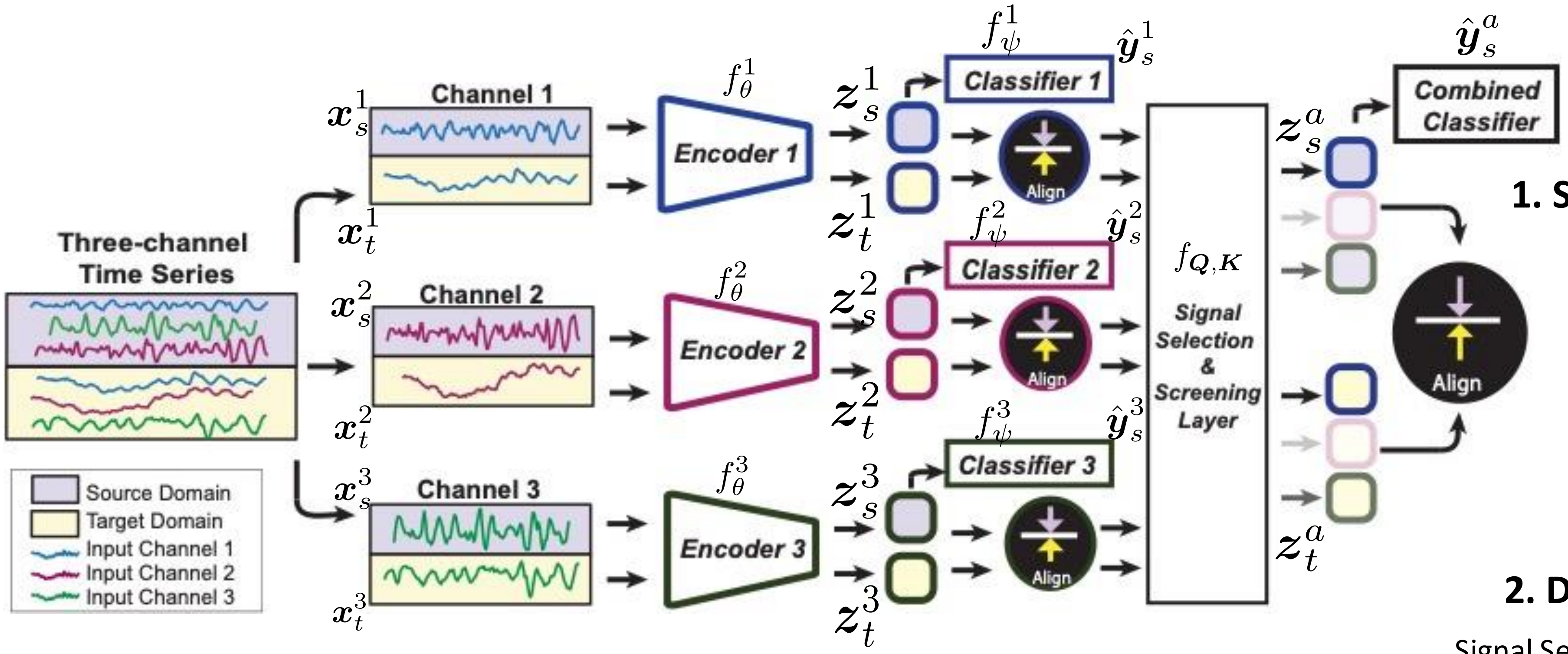
Representations f_θ



Domain adaptation method likely to succeed

Can we learn to ignore certain channels to improve domain adaptation?

Proposed Method



1. Separate Dom Adapt for each channel

Source input channel : x_s^c

Target input channel: x_t^c

Encoder channel c : f_θ^c

Classifier channel c : f_ψ^c

Source reps channel c: $z_s^c = f_\theta^c(x_s^c)$

Target reps channel c: $z_t^c = f_\theta^c(x_t^c)$

Source predict channel c: $\hat{y}_s^c = f_\psi^c(z_s^c)$

2. Downweigh channels and aggregate

Signal Selection & Screening layer layer: $f_{Q,K}$

Aggregated source reps: $z_s^a = f_{Q,K}\left([z_s^1, z_s^2, .., z_s^C]\right)$

Aggregated target reps: $z_t^a = f_{Q,K}\left([z_t^1, z_t^2, .., z_t^C]\right)$

Aggregated src predict: $\hat{y}_s^c = f_\psi^c(z_s^c)$

$$\mathcal{L} = \underbrace{\sum_{c=1}^{C} S_\gamma(z_s^c, z_t^c) + \mathcal{L}_{CE}(\hat{y}_s^c, y_s)}_{\text{Dom. adapt. for each channel}} + \underbrace{S_\gamma(z_s^a, z_s^b) + \mathcal{L}_{CE}(\hat{y}_s^a, y)}_{\text{Dom. adapt. for aggregated reps.}}$$

Sinkhorn distance: S_γ
Cross Entropy: $\mathcal{L}_{CE}$

Signal Selection and Screening Layer

Downweighs and aggregates channels

$$\boldsymbol{z}_a = f_{\boldsymbol{K},\boldsymbol{Q}}\left(\left[\boldsymbol{z}^1, \boldsymbol{z}^2, ..., \boldsymbol{z}^C\right]\right)$$

Input: Representations from each channel, $\boldsymbol{z}_c \in \mathbb{R}^d$

Learnable parameters: $\boldsymbol{K}, \boldsymbol{Q} \in \mathbb{R}^{d\times d}$

1. **Obtain *query* and *key* embeddings for each channel**

$$\boldsymbol{q}^c = \boldsymbol{Q}\boldsymbol{z}_c$$
$$\boldsymbol{k}^c = \boldsymbol{K}\boldsymbol{z}_c$$

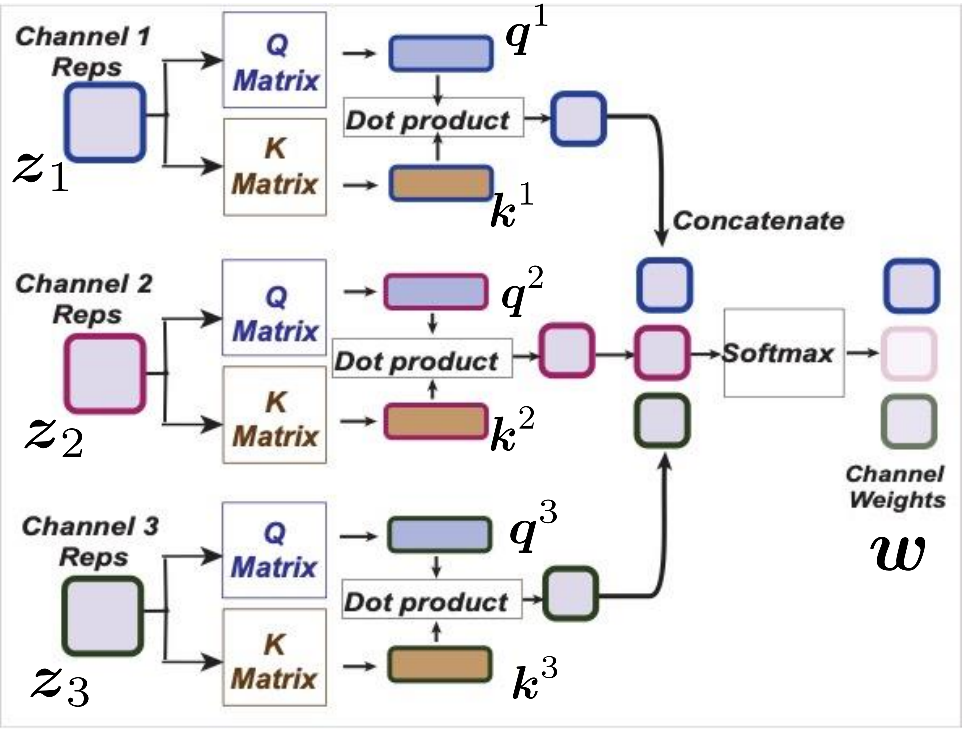
2. Obtain weights

$$\boldsymbol{w} = \mathrm{softmax}\left(\frac{1}{\tau}\left[\frac{1}{\sqrt{d}}\left((\boldsymbol{q}^1)^\intercal\boldsymbol{k}^1, \ldots, (\boldsymbol{q}^C)^\intercal\boldsymbol{k}^C\right)\right]\right)$$

3. Aggregate channel representations

$$\boldsymbol{z}^a = \mathrm{vec}\left(\boldsymbol{w} \odot \boldsymbol{Z}\right) = \mathrm{vec}\left(\left[w^1\boldsymbol{z}^1, w^2\boldsymbol{z}^2, ..., w^C\boldsymbol{z}^C\right]\right).$$

Signal selection & Screening Layer



Experiments Datasets

Evaluated our method on

1. Simulated data. 4 channels
    - Normally distributed channels whose mean values shift to get 4 classes.
    - Mean of 1 random selected channel shifted for each class to get target domain
2. Real world datasets included:

    a). UCIHAR 9 channels:

    - Activity recognition. 5 classes (running/jog/sitting/stand/walking up stair/walking down/)
    - Triaxial accelerometer data on 3 devices (Wrist, chest, hip)
    - 10 pairs of users selected. 1st in pair used as source, the 2nd as target

    b) HHAR 3 channels:

    - Activity recognition. 5 classes
    - 10 pairs of users selected. 1st in pair used as source, the 2nd as target

    c) WISDM 3 channels Activity recognition

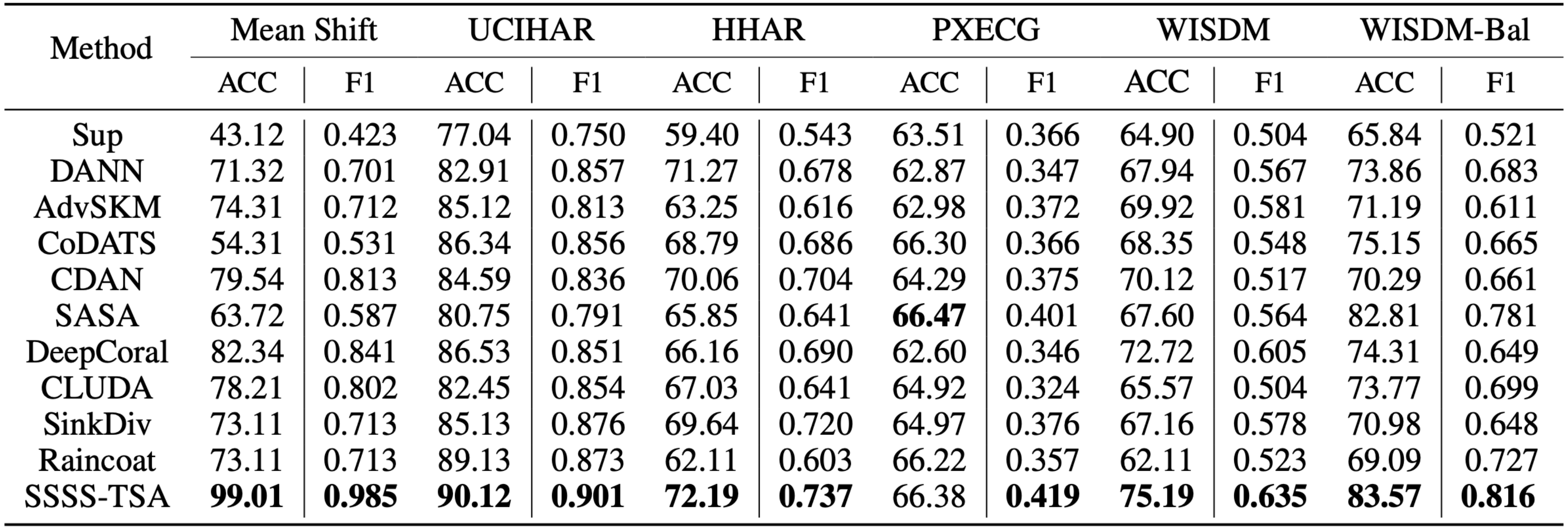
    - Triaxial accelerometer data on device.
    - Different type devices, ranging from different smart phones phone to different smart watches for different users
      Activity recognition. 5 classes
    - 10 pairs of users selected. 1st in pair used as source, the 2nd as target

    d) Multichannel ECG signals (12 channels),

    - 5 classes (Different heat states Normal, Myocardial Infraction, Conduction disturbance, Hypertrophy, ST/T-change)
    - 10 pairs of different sites. First used as source, the second as target

Results

Mean Accuracy and Macro F1 scores over 5 different runs. Higher is better



Method	Mean Shift		UCIHAR		HHAR		PXECG		WISDM		WISDM-Bal	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
Sup	43.12	0.423	77.04	0.750	59.40	0.543	63.51	0.366	64.90	0.504	65.84	0.521
DANN	71.32	0.701	82.91	0.857	71.27	0.678	62.87	0.347	67.94	0.567	73.86	0.683
AdvSKM	74.31	0.712	85.12	0.813	63.25	0.616	62.98	0.372	69.92	0.581	71.19	0.611
CoDATS	54.31	0.531	86.34	0.856	68.79	0.686	66.30	0.366	68.35	0.548	75.15	0.665
CDAN	79.54	0.813	84.59	0.836	70.06	0.704	64.29	0.375	70.12	0.517	70.29	0.661
SASA	63.72	0.587	80.75	0.791	65.85	0.641	**66.47**	0.401	67.60	0.564	82.81	0.781
DeepCoral	82.34	0.841	86.53	0.851	66.16	0.690	62.60	0.346	72.72	0.605	74.31	0.649
CLUDA	78.21	0.802	82.45	0.854	67.03	0.641	64.92	0.324	65.57	0.504	73.77	0.699
SinkDiv	73.11	0.713	85.13	0.876	69.64	0.720	64.97	0.376	67.16	0.578	70.98	0.648
Raincoat	73.11	0.713	89.13	0.873	62.11	0.603	66.22	0.357	62.11	0.523	69.09	0.727
SSSS-TSA	**99.01**	**0.985**	**90.12**	**0.901**	**72.19**	**0.737**	66.38	**0.419**	**75.19**	**0.635**	**83.57**	**0.816**

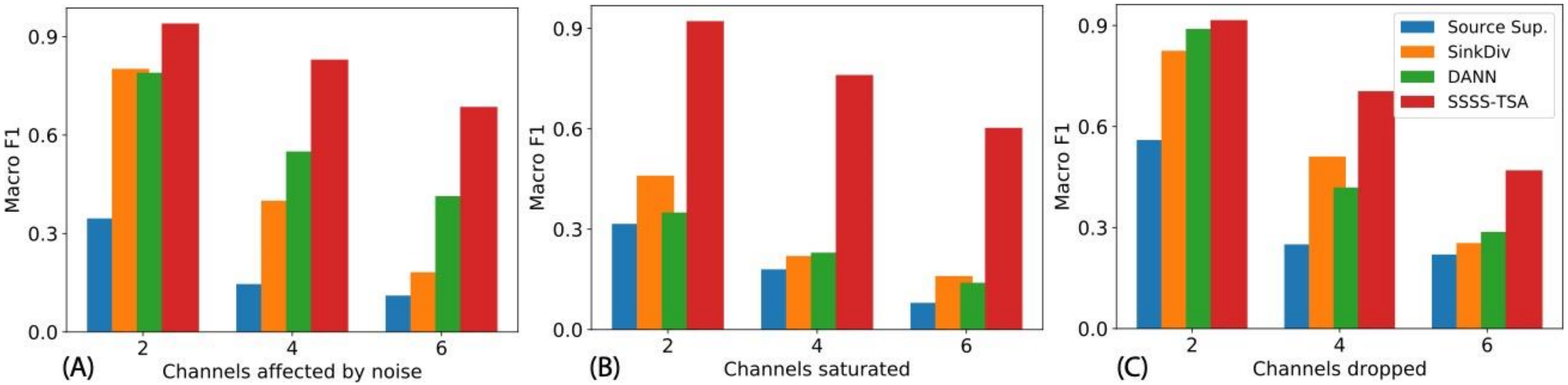
Our method performs, SSSS-TSA, performs better on most datasets as compared to popular baselines

Performance Under Enforced Channel Shifts

To further test our method, we created more severe domain shifts to existing domain shift scenarios in UCIHAR datasets.

This was done by varying the number of target channels that were:

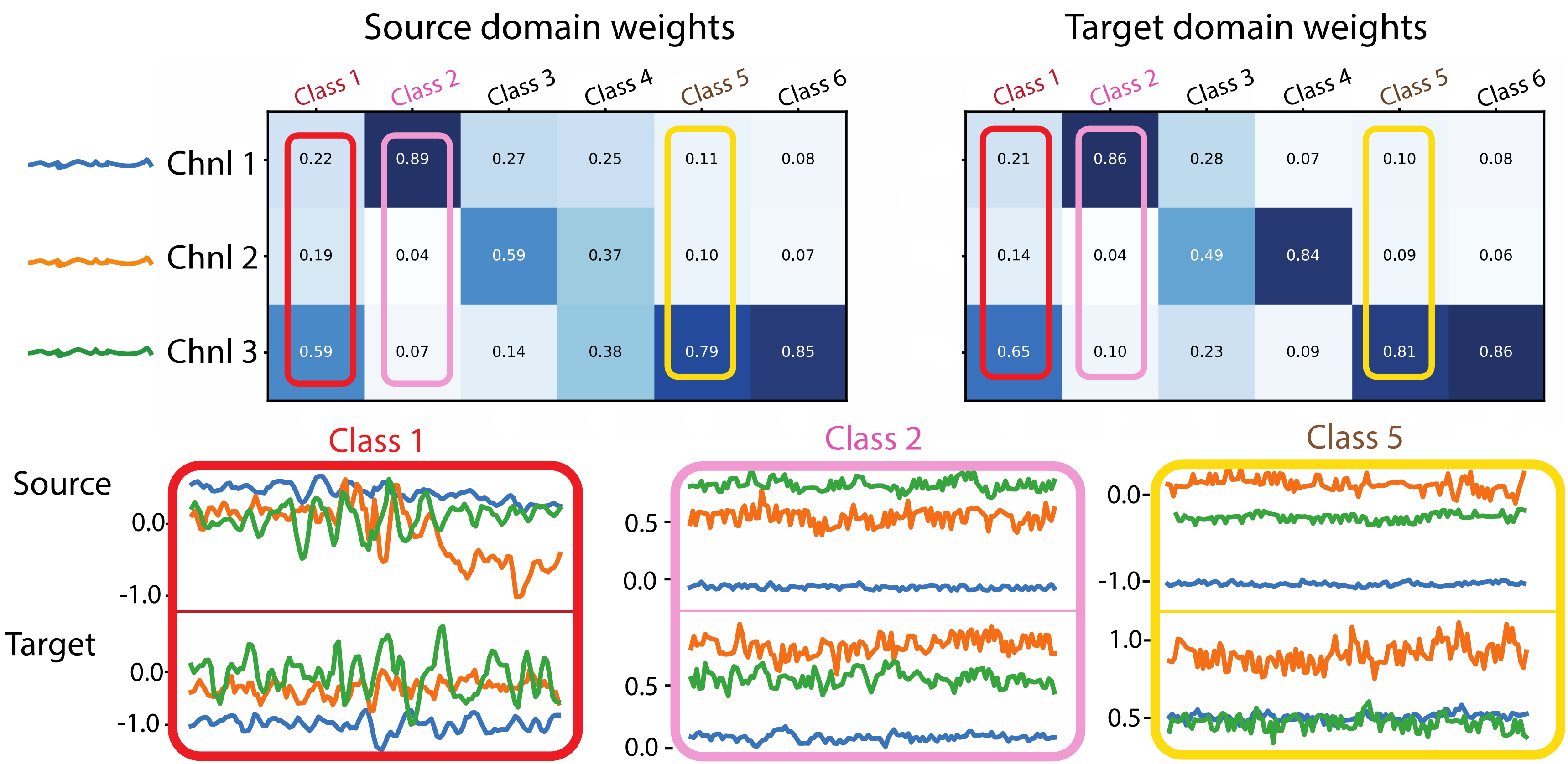
1. Adding noise channels.
2. Saturating channels
3. Dropping Channels



Our proposed method SSSS-TSA much more robust to such corruptions

Examples of Weights Learned to Select Channels

Example of weights learned by channel selection layer on UCIHAR dataset



Source domain weights

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Chnl 1	0.22	0.89	0.27	0.25	0.11	0.08
Chnl 2	0.19	0.04	0.59	0.37	0.10	0.07
Chnl 3	0.59	0.07	0.14	0.38	0.79	0.85

Target domain weights

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Chnl 1	0.21	0.86	0.28	0.07	0.10	0.08
Chnl 2	0.14	0.04	0.49	0.84	0.09	0.06
Chnl 3	0.65	0.10	0.23	0.09	0.81	0.86

Methods learns to give smaller weights to channels with large shifts across source and target domain

Aim Summary and Contributions

1.Proposed a new domain adaptation method based on channel selection

2. Can downweigh channels with severe corruptions for improved domain adaptation

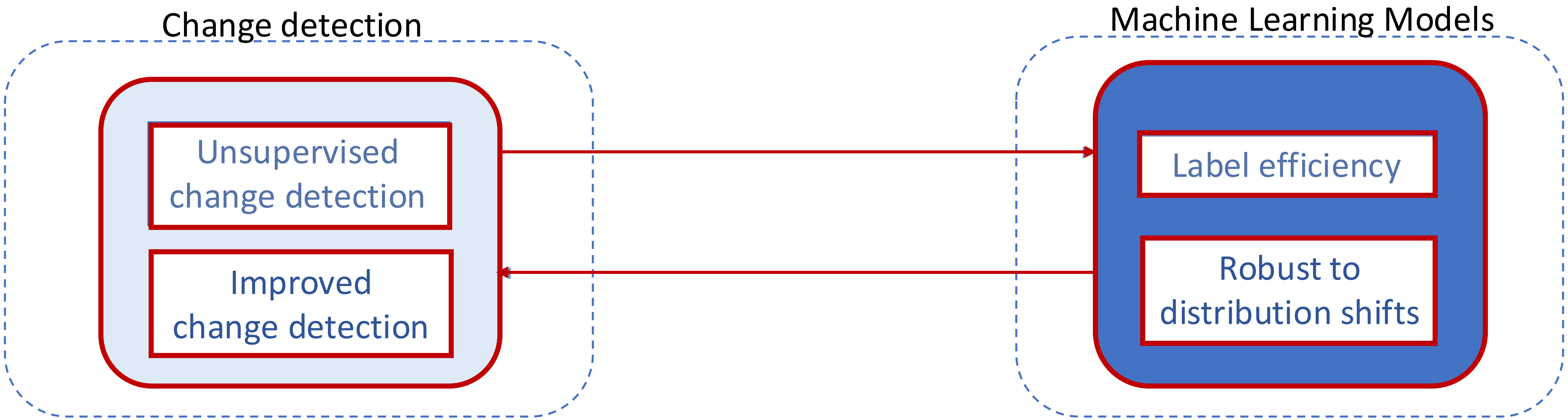
3. Learned weights can help interpret what channels are important for classification

Paper submitted

N. Ahad, M. Davenport, E. Dyer, *"Time series domain adaptation via channel-selective representation alignment"*, under review 2024

Summary of Thesis

1. Multiple change point detection in streaming data settings
2. Using change detection for label efficient supervised learning
3. Using supervision tools from machine learning to improve change detection
4. Improving supervised models in the presence of distribution shifts


Change detection
Unsupervised change detection
Improved change detection
Machine Learning Models
Label efficiency
Robust to distribution shifts

Publications

Journal papers and Pre-prints

1. **N. Ahad**, S. Sonenbum, M. Davenport, S. Sprigle, “ Validating a Wheelchair In-Seat Activity Tracker”, *Assistive Technology*, 2021

2. **N. Ahad**, M. Davenport, Y. Xie, “Data Adaptive Symmetrical CUSUM ”, *Sequential Analysis,* 2024

3. **N. Ahad**, E. Dyer, K. Hengen, Y. Xie, M. Davenport, “Learning Sinkhorn divergences for supervised change point detection”, *in revision, IEEE Transactions on Signal Processing , arxiv-preprint 2202.04000*

4. **N. Ahad**, M. Davenport, E. Dyer, “Time series domain adaptation via channel-selective representation alignment”, preprint 2024

Conference papers and peer-reviewed abstracts

1.N. Ahad, M. Davenport, “ Semi-supervised Sequence Classification through Change Point Detection ”, *AAAI,* 2021

2.C. Uzray∗, **N.Ahad**∗, M. Azabou, E. Dyer, ”Detecting change points in neural population activity with contrastive metric learning”, *IEEE Conference on Neural Engineering* , 2023

3. N.Ahad∗, N. Nadagouda*, E. Dyer, M. Davenport, “Active learning for time instant classification”, *DMLR workshop, ICML 2023*

4. M. Azabou, M. Mendelson, **N.Ahad**, M.Sorkin, S. Thakook, C. Uzray, E.Dyer, *“*Relax, it doesn’t matter how you get there: A new self-supervised approach for multitimescale behavior analysis*”, NeurIPS, 2023*

5.F. Zhu, A. Sedler, H. Grier, **N. Ahad**, M. Davenport, M. Kaufman,A. Giovannucci, C.Pandarinath “ Deep inference of latent dynamics with spatio-temporal super-resolution using selective backpropagation through time ”, *NeurIPS,* 2021

6.J. Quesada, L. Sathidevi, R. Liu, **N. Ahad**, J. M. Jackson, M. Azabou, ..,E. L. Dyer “ MTNeuro: A Benchmark for Evaluating Representations of Brain Structure Across Multiple Levels of Abstraction ”, *NeurIPS Datasets and Benchmarks Track*, 2022

7.A.D. McRae, A. Xu, J. Jin, N.Nadagouda, **N. Ahad**, P. Guan, S. Karnik, M. Davenport,“ Delta distancing: A Lifting Approach to localizing items from user comparisons”, *ICASSP*, 2022

Acknowledgements

My advisor and collaborators

- *Mark Davenport*
- *Sharon Sonenblum*
- *Yao Xie*
- *Eva Dyer*
- *Namrata Nadagouda*
- *Carolina Uzray*
- *Mehdi Azabou*

Labmates & wider COTN group

Family!

Research Support

NSF

NIH
National Institutes of Health

Wallace H. Coulter Foundation

Questions

Paraplegic Wheelchair Users & Sepsis

- 46% of people with spinal cord injuries develop pressure ulcers[1]
- 20% require expensive surgeries to manage these ulcers[2]
- When infected, these ulcers can lead to sepsis

[1] N.S.C.I.S.C. (2015). *Annual statistical report by the national spinal cord injury statistical center*.
[2] Saunders, L. et. al... Association of race, socioeconomic status, and health care access with pressure ulcers after spinal cord injury. *Archives of Physical Medicine and Rehabilitation*, *93*(6), 972–977. https://doi.org/10.1016/j.apmr.2012. 02.004,

Managing Pressure Ulcers through Pressure Offloading

Clinical experts recommended pressure relief movements, called ***weight shifts***

But:

1. Wheelchair users often forget to perform these weight shifts
2. Large scale study needed to understand relationship between weight shift frequency and ulcer development

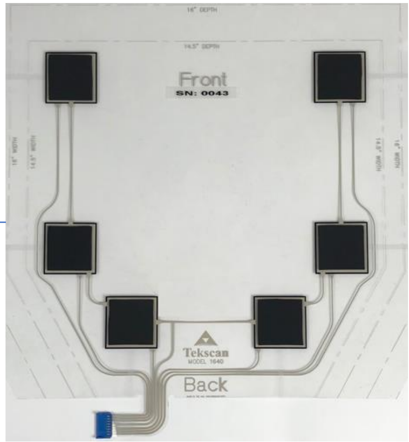
Goal: Design an in-seat activity tracker for wheelchair users
Akin to a Fitbit for wheelchair users

WISAT: Wheelchair In-seat Activity Tracker

User moves in the chair

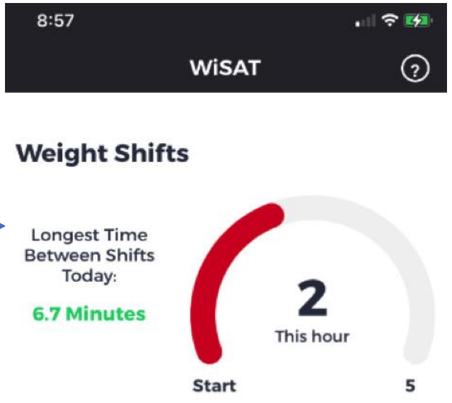
Pressure sensor mat inserted beneath cushion

Sensor values change



ML algorithms on sensor data

Weight shift detection



Weight Shifts

Longest Time Between Shifts Today:

6.7 Minutes

2

This hour

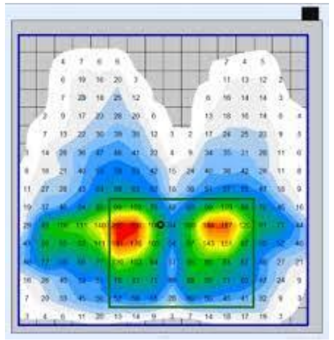
Tracker should work with different types of cushions

Dataset

- 8 minute training protocol for providing training data where 20 participants performed different movements
- High resolution mat placed above cushion for providing:
  1. Identifying timestamps in the protocol where users perform weight shifts
  2. Ground truth for sufficient pressure offloading

High resolution mat

provides

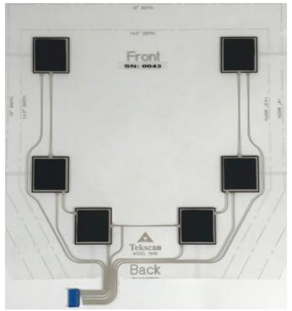


Pressure offloading ground truth

Can’t use this high resolution mat directly!

1. Extremely expensive
2. Not an ideal contact surface for extended use

Goal: use pressure mat beneath cushion to detect these pressure offloading



Challenging problem

- Different cushions have different dampening behavior
- Sensor mat can slide beneath the cushion
- Training data is relatively not that extensive
- Makes it difficult to apply neural networks directly on sensor data

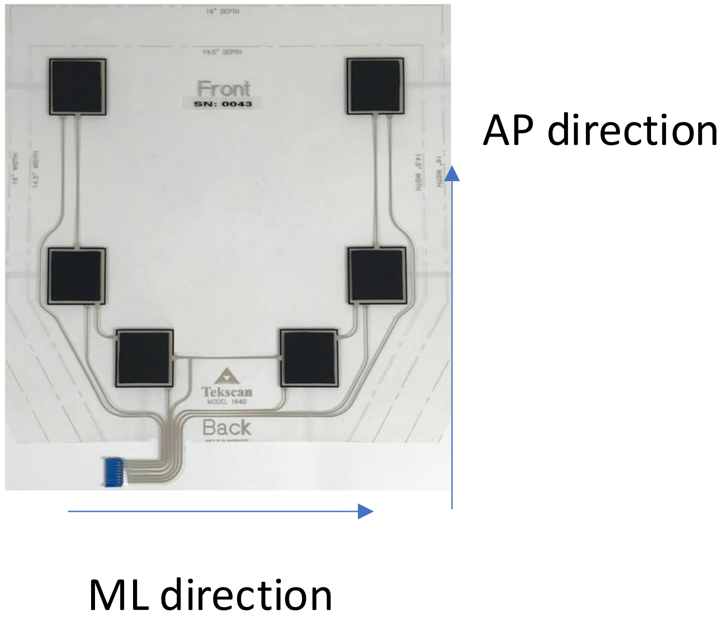
Training: precision and recall > 90%
Validation: mean precision and mean recall < 65%

Poor generalization to real world settings

Using Domain Knowledge

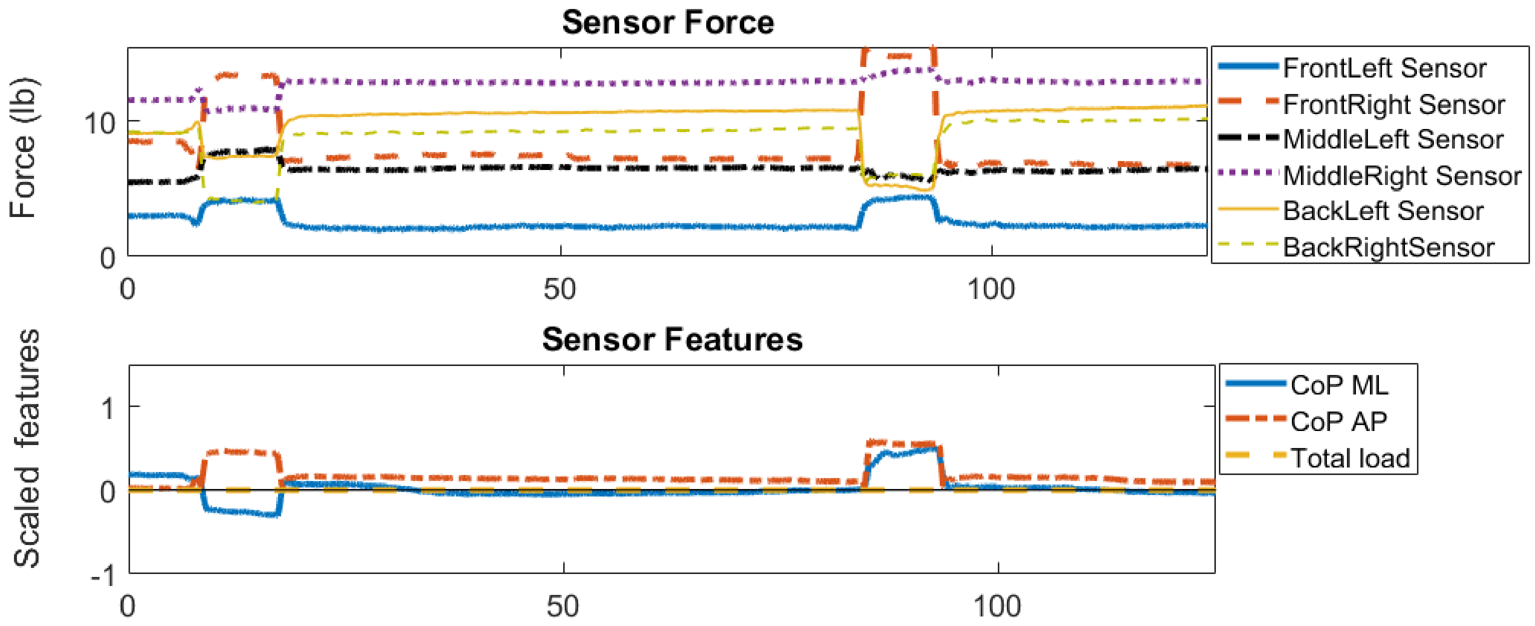
Rather than using raw sensor values, computed three features with the help of domain experts:

1. Center of Pressure Medial lateral (COP_{ML})
2. Center of Pressure Anterior Posterior (COP_{AP})
3. Total Load (sum of sensor values)



AP direction

ML direction

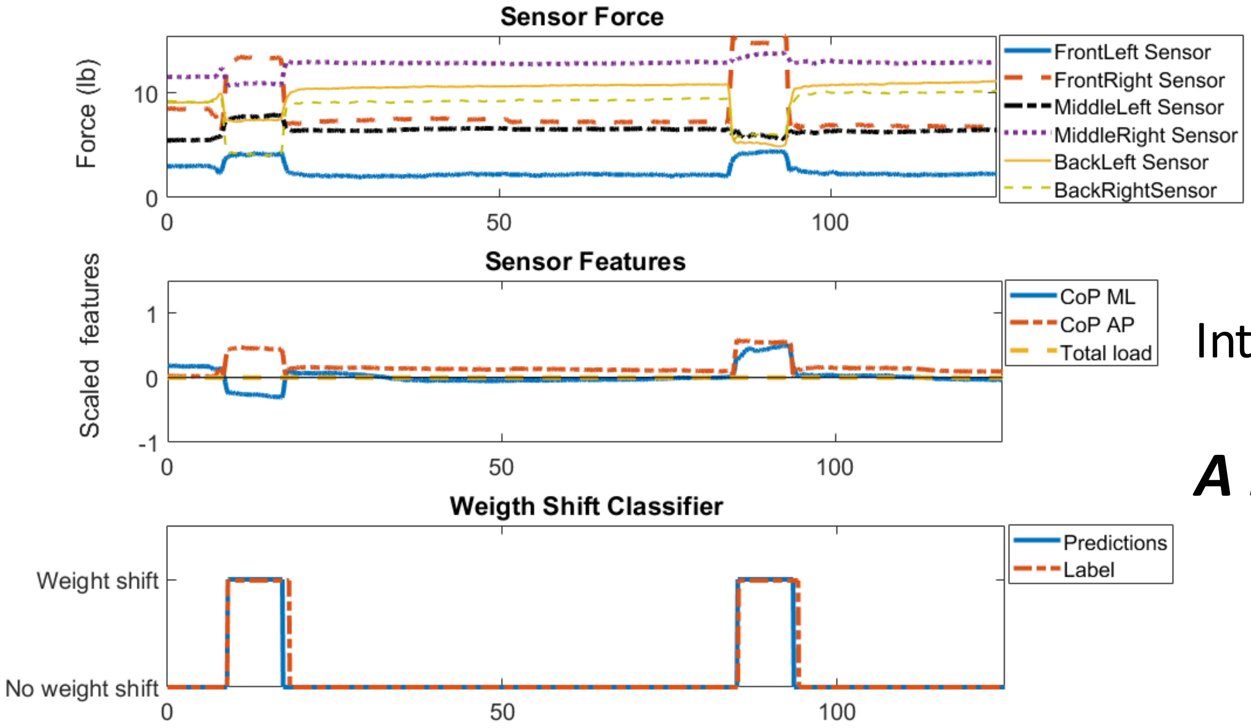


Sensor Force

Sensor Features

Weight Shift Classifier

- 3 features COP_{ML} , COP_{AP}, and Total Load to train a ***support vector machine (SVM) classifier***
- Unlike black box models like neural networks, SVMs. can provide a quadratic expression relating features to output



Interpretable features and an interpretable classifier

A reliable partnership in real world noisy settings!

Performance

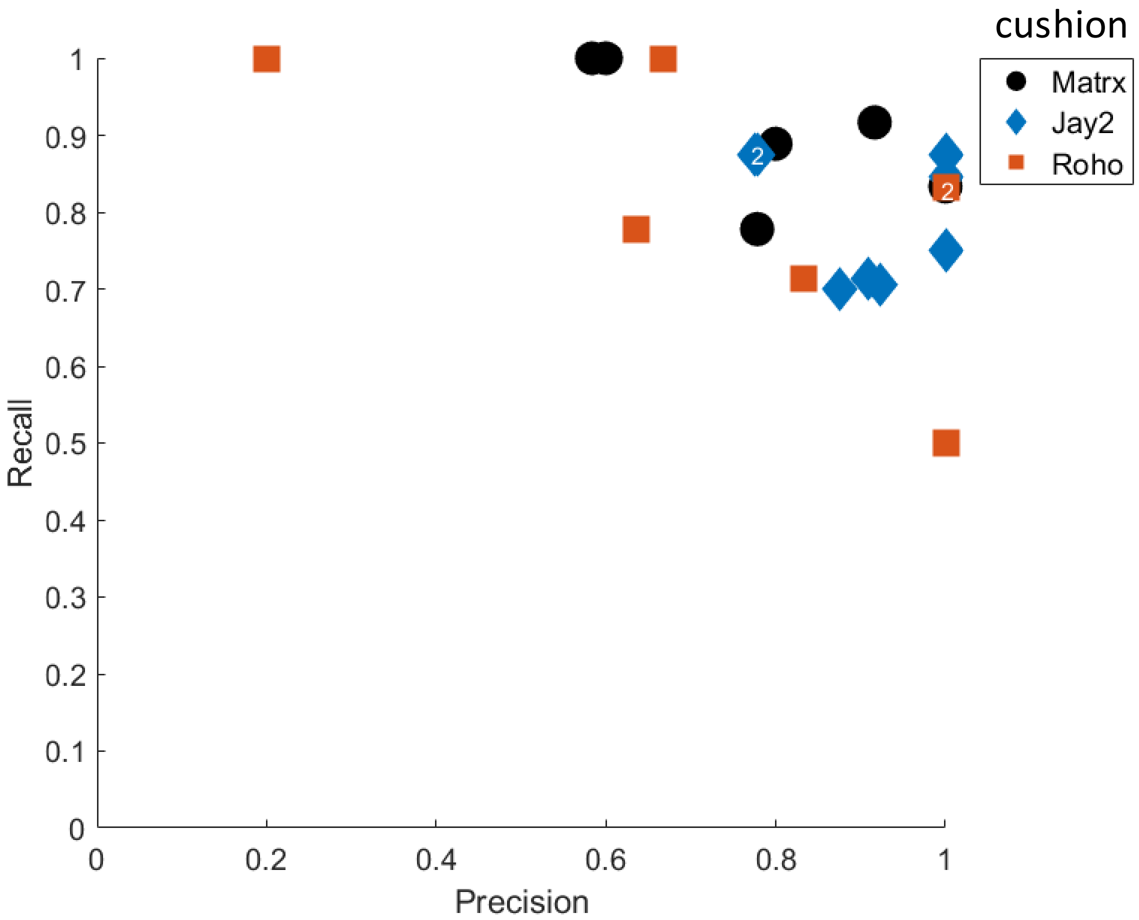
Precision:
Of all detected weight shift segments, how many true weight shift segments

Recall:
Of all true weight shift segments, how many were detected correctly

(Higher is better)

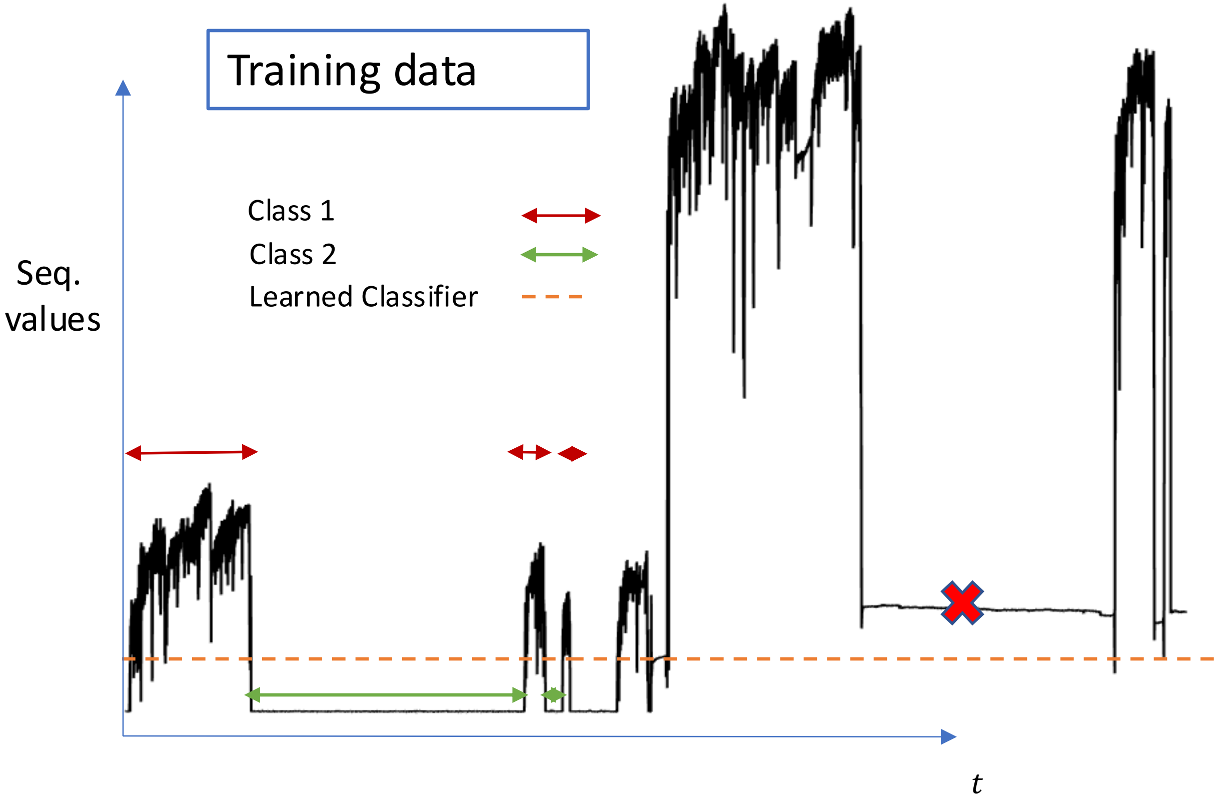
Mean Precision score: **0.81**
Mean Recall score: **0.80**

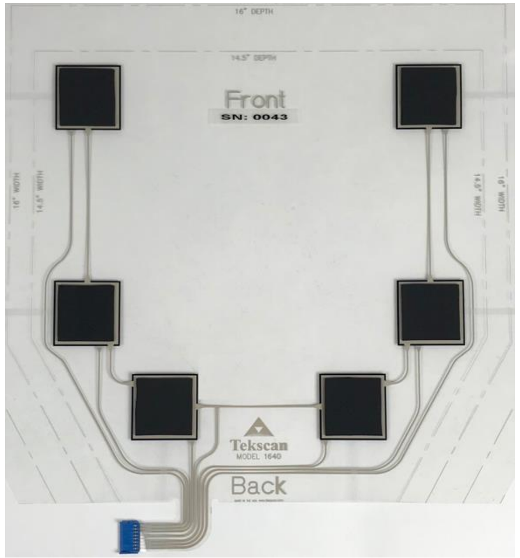
Performance not 99% because of nature of the problem !



Outliers on Roho (air filled) cushion

How difficult a problem?
Consider Occupancy Detection



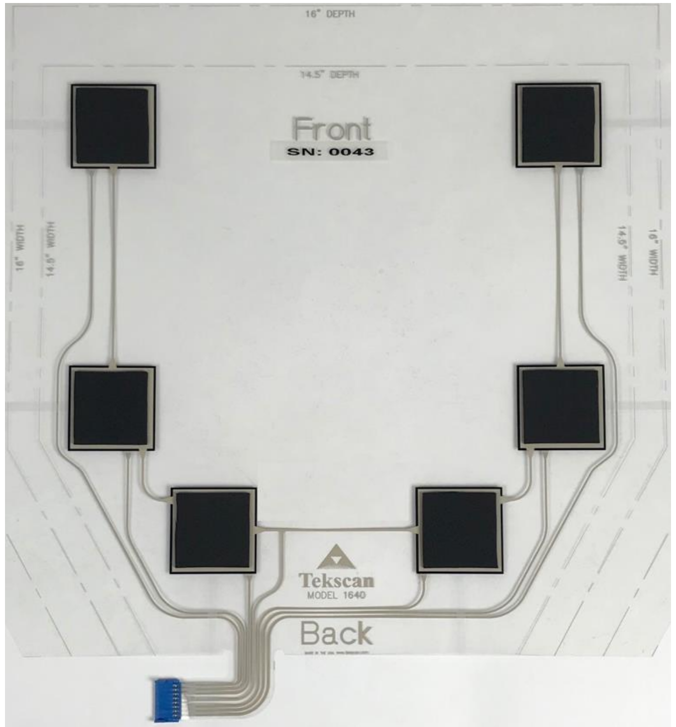


Occupancy classifier

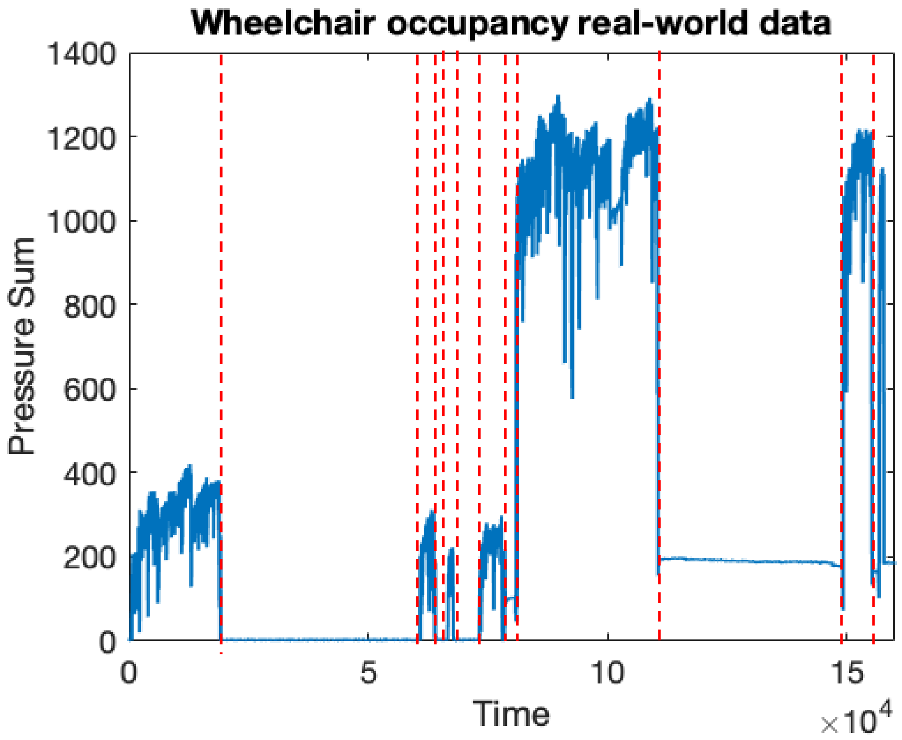
Supervised classifier fails as data distribution changes

Unsupervised Change Point Detection

Use ***unsupervised change point detection*** in place of a supervised classifier



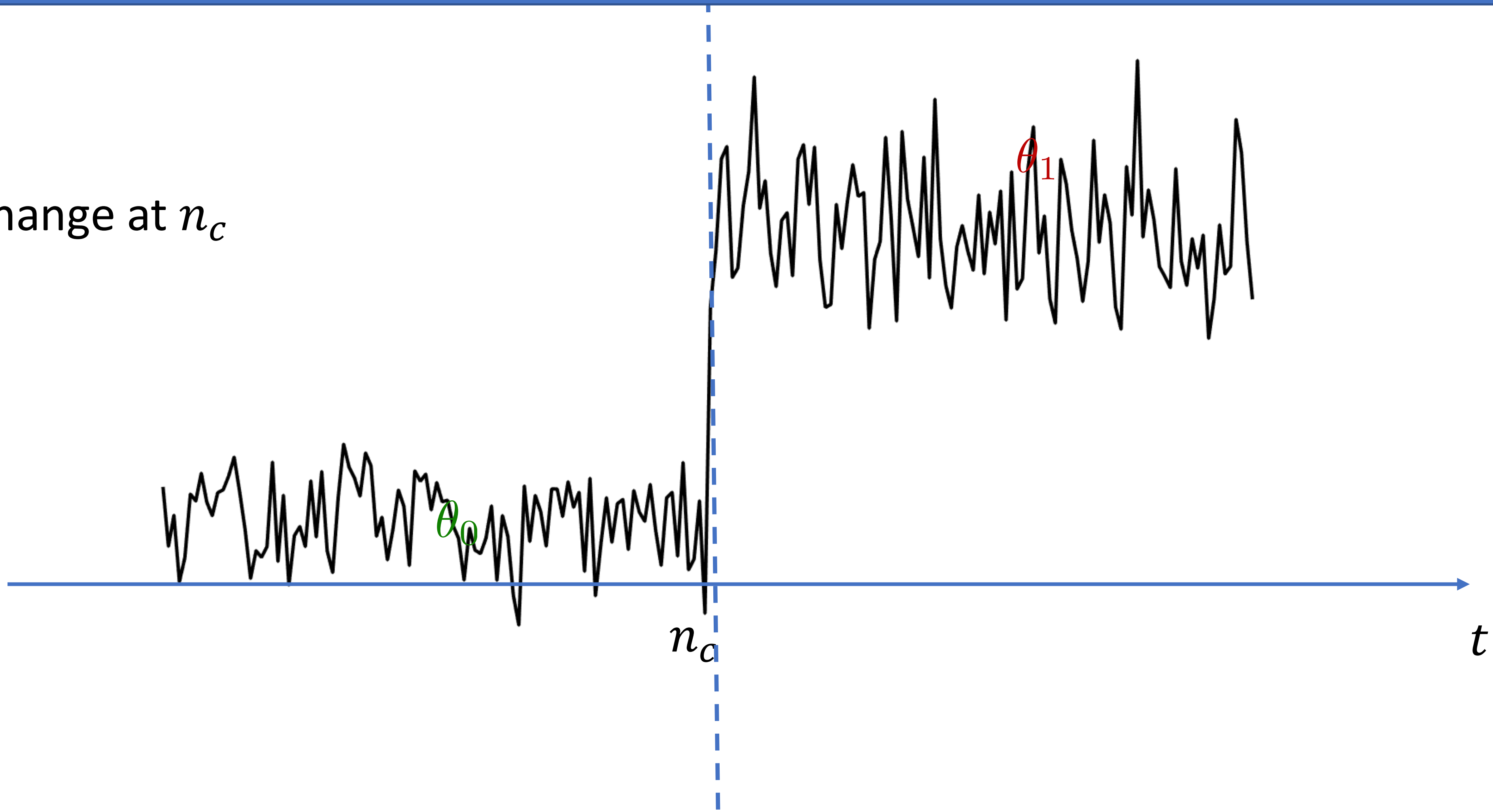
1. Pressure sensor mat
(beneath seat cushion)



2. Pressure sensor readings

Change Detection

Change at n_c



Goal: Identify where change points (n_c) are located in a time series