

# Détection d'événements à partir de capteurs sols – application au suivi de personnes fragiles

Soutenance de thèse

Ludovic Minvielle

Thèse industrielle entre le Centre Borelli (ENS Paris-Saclay) et Tarkett

Mercredi 15 Juillet 2020



# Introduction

## General context

- ▶ Elderly population is growing
- ▶ Higher levels of frailty globally
- ▶ Increasing demand for reliable monitoring devices
- ▶ Tarkett, French company: 12,500 employees, 13 industrial sites, 1.3 millions m<sup>2</sup> of flooring per day
- ▶ *Floor in Motion*: a floor-based sensor for elderly care

# Introduction

## General context

- ▶ Elderly population is growing
- ▶ Higher levels of frailty globally
- ▶ Increasing demand for reliable monitoring devices
- ▶ Tarkett, French company: 12,500 employees, 13 industrial sites, 1.3 millions m<sup>2</sup> of flooring per day
- ▶ *Floor in Motion*: a floor-based sensor for elderly care



# Introduction

## General context

- ▶ Elderly population is growing
- ▶ Higher levels of frailty globally
- ▶ Increasing demand for reliable monitoring devices
- ▶ Tarkett, French company: 12,500 employees, 13 industrial sites, 1.3 millions m<sup>2</sup> of flooring per day
- ▶ *Floor in Motion*: a floor-based sensor for elderly care

## Objective

Providing tools for elderly monitoring  
in nursing homes



# Introduction

## General context

- ▶ Elderly population is growing
- ▶ Higher levels of frailty globally
- ▶ Increasing demand for reliable monitoring devices
- ▶ Tarkett, French company: 12,500 employees, 13 industrial sites, 1.3 millions m<sup>2</sup> of flooring per day
- ▶ *Floor in Motion*: a floor-based sensor for elderly care

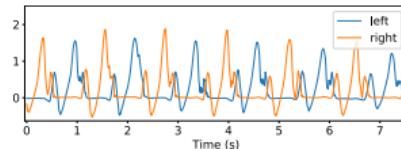


## Objective

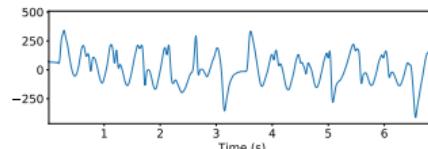
Providing tools for elderly monitoring  
in nursing homes

## Scientific context

- ▶ Processing and understanding time series
- ▶ Real world application



Foot-attached accelerometer



Tarkett's floor sensor

## Table of contents

1. Monitoring systems
2. Fall detection
3. Transfer learning
4. Elderly activity recognition
5. Conclusion

## Monitoring systems for fall detection

## Sensors

What makes a good monitoring system ?

- ▶ coverage and occlusion
- ▶ intrusiveness
- ▶ signal quality / information
- ▶ robustness
- ▶ ease of installation / use
- ▶ scalability

### Criteria

---

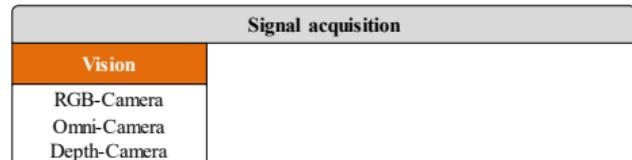
Coverage/Occlusion  
Intrusiveness  
Signal quality / info  
Robustness  
Ease of instal. / use  
Scalability

---

## Sensors

What makes a good monitoring system ?

- ▶ coverage and occlusion
- ▶ intrusiveness
- ▶ signal quality / information
- ▶ robustness
- ▶ ease of installation / use
- ▶ scalability



Criteria	RGB cam	Depth cam
Coverage/Occlusion	★☆☆	★☆☆
Intrusiveness	★☆☆	★☆☆
Signal quality / info	★★★	★★★
Robustness	★☆☆	★★★
Ease of instal. / use	★☆☆	★☆☆
Scalability	★☆☆	★☆☆

## Sensors

What makes a good monitoring system ?

- ▶ coverage and occlusion
- ▶ intrusiveness
- ▶ signal quality / information
- ▶ robustness
- ▶ ease of installation / use
- ▶ scalability

Signal acquisition	
Vision	Wearable
RGB-Camera	
Omni-Camera	Accelerometer
Depth-Camera	Gyroscope
	Barometric pressure

Criteria	RGB cam	Depth cam	Wearable
Coverage/Occlusion	★☆☆	★☆☆	★★★
Intrusiveness	★☆☆	★☆☆	★★☆
Signal quality / info	★★★	★★★	★★☆
Robustness	★☆☆	★★★	★★★
Ease of instal. / use	★☆☆	★☆☆	★★☆
Scalability	★☆☆	★☆☆	★★★

## Sensors

What makes a good monitoring system ?

- ▶ coverage and occlusion
- ▶ intrusiveness
- ▶ signal quality / information
- ▶ robustness
- ▶ ease of installation / use
- ▶ scalability

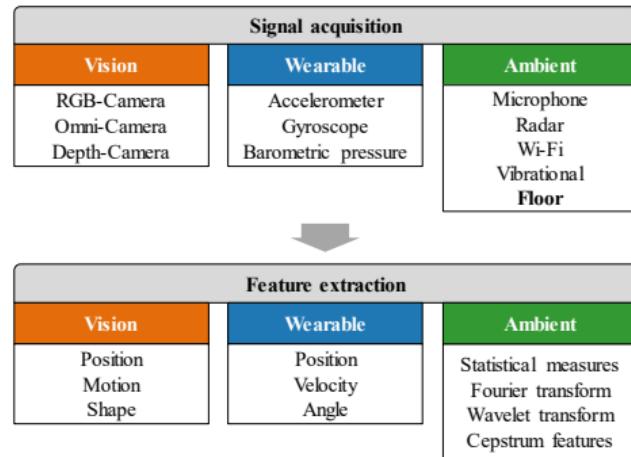
Signal acquisition		
Vision	Wearable	Ambient
RGB-Camera	Accelerometer	Microphone
Omni-Camera	Gyroscope	Radar
Depth-Camera	Barometric pressure	Wi-Fi
		Vibration
		Floor

Criteria	RGB cam	Depth cam	Wearable	Acoustic	Radar / Wi-Fi	Vibration	Floor
Coverage/Occlusion	★☆☆	★☆☆	★★★	★☆☆	★☆☆	★★★	★★★
Intrusiveness	★☆☆	★☆☆	★☆☆	★☆☆	★☆☆	★★★	★★★
Signal quality / info	★★★	★★★	★☆☆	★☆☆	★☆☆	★☆☆	★☆☆
Robustness	★☆☆	★★★	★★★	★☆☆	★☆☆	★☆☆	★☆☆
Ease of instal. / use	★☆☆	★☆☆	★☆☆	★☆☆	★☆☆	★★★	★☆☆
Scalability	★☆☆	★☆☆	★★★	★☆☆	★☆☆	★☆☆	★★★

## Information extraction

How to process the inputs ?

- ▶ All systems use feature extraction
- ▶ The “level” of feature engineering depends on the complexity / dimensionality of the input signal



## Information extraction

How to process the inputs ?

- ▶ All systems use feature extraction
- ▶ The “level” of feature engineering depends on the complexity / dimensionality of the input signal

How to deal with feature signals ?

- ▶ Use simple thresholds
- ▶ Use them as feature vectors for classification models (anomaly detection, classical supervised models)

Signal acquisition		
Vision	Wearable	Ambient
RGB-Camera	Accelerometer	Microphone
Omni-Camera	Gyroscope	Radar
Depth-Camera	Barometric pressure	Wi-Fi
		Vibrational
		Floor



Feature extraction		
Vision	Wearable	Ambient
Position	Position	Statistical measures
Motion	Velocity	Fourier transform
Shape	Angle	Wavelet transform
		Cepstrum features



Decision rule	
Threshold	Machine learning
	kNN SVM HMM Decision Tree

## A fall detection system

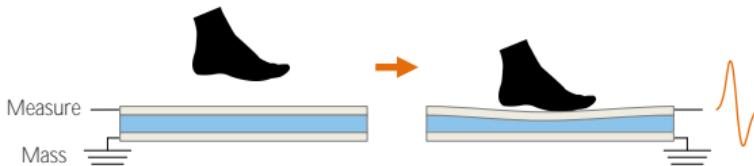
## Tarkett sensor

- ▶ Piezoelectric principle:

$$d = \frac{Q}{F},$$

(simple version) with  $d$  the *piezoelectric constant*.

When stressed or squeezed, the material emits charges.



## Tarkett sensor

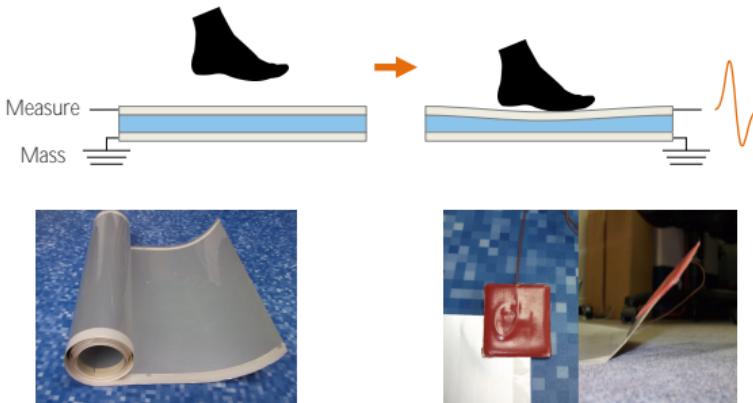
- ▶ Piezoelectric principle:

$$d = \frac{Q}{F},$$

(simple version) with  $d$  the *piezoelectric constant*.

When stressed or squeezed, the material emits charges.

- ▶ How does this look like ?  
0.3 mm thick and 60 cm wide roll  
with customizable length



## Tarkett sensor

- ▶ Piezoelectric principle:

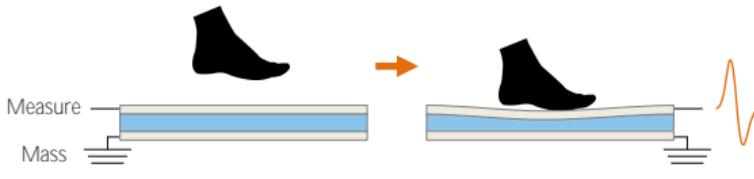
$$d = \frac{Q}{F},$$

(simple version) with  $d$  the *piezoelectric constant*.

When stressed or squeezed, the material emits charges.

- ▶ How does this look like ?  
0.3 mm thick and 60 cm wide roll  
with customizable length
- ▶ How is it installed ?

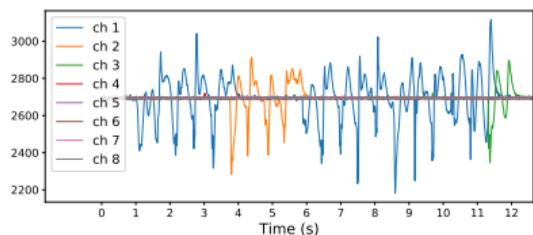
- ▶ Under the flooring
- ▶ Several connected bands for each area, hence one area corresponds to one input



## Data

### Preprocessing

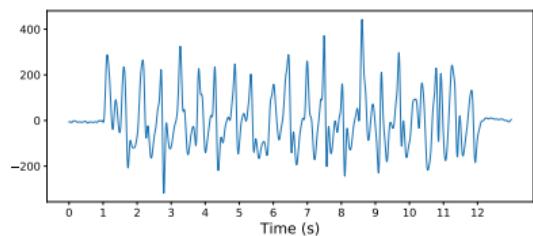
- ▶ linear detrending
- ▶ low-pass filtering
- ▶ zeroing low energy channels
- ▶ sum over all channels



## Data

### Preprocessing

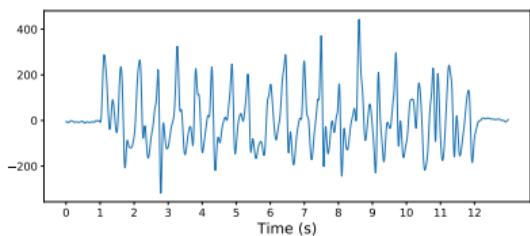
- ▶ linear detrending
- ▶ low-pass filtering
- ▶ zeroing low energy channels
- ▶ sum over all channels



## Data

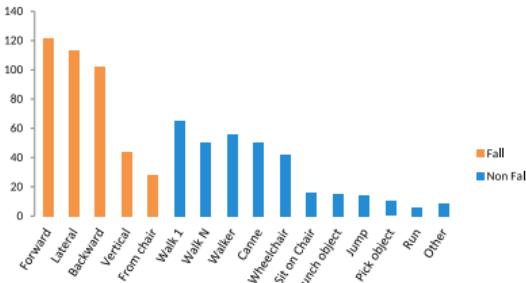
### Preprocessing

- ▶ linear detrending
- ▶ low-pass filtering
- ▶ zeroing low energy channels
- ▶ sum over all channels



### Experimental dataset

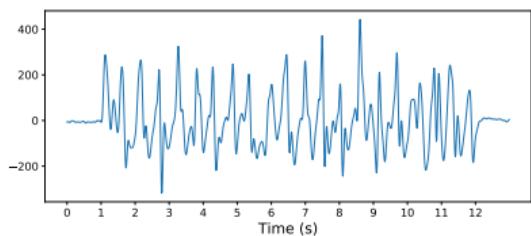
- ▶ 742 signals collected in **controlled environment**
- ▶ 55% fall, 45% non-fall
- ▶ varied fall events (forward, backward...) and activities of daily living (walking, sitting...)



## Data

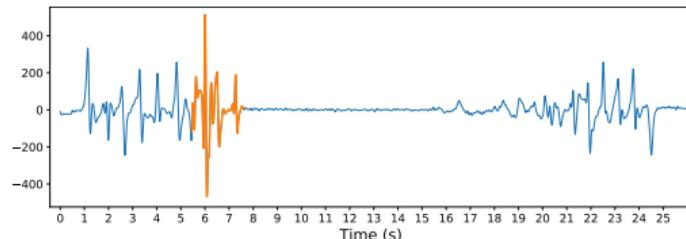
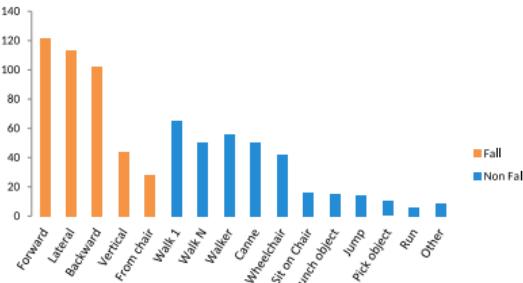
### Preprocessing

- ▶ linear detrending
- ▶ low-pass filtering
- ▶ zeroing low energy channels
- ▶ sum over all channels



### Experimental dataset

- ▶ 742 signals collected in **controlled environment**
- ▶ 55% fall, 45% non-fall
- ▶ varied fall events (forward, backward...) and activities of daily living (walking, sitting...)

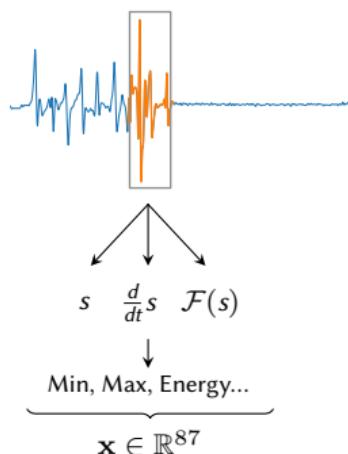


## Method

### Model

Time series as *feature vector*. At every timestamp:

1. Window over the signal: 2.5 s
2. Compute feature vector: 29 statistical measures (Min, Max, Shannon energy, Percentile,...) over three representations of the signal

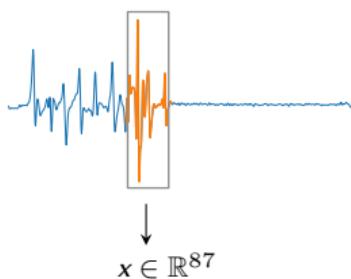


## Method

### Model

Time series as *feature vector*. At every timestamp:

1. Window over the signal: 2.5 s
2. Compute feature vector: 29 statistical measures  
(Min, Max, Shannon energy, Percentile,...) over  
three representations of the signal

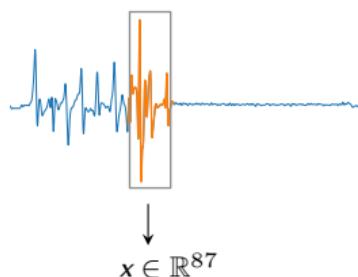


## Method

### Model

Time series as *feature vector*. At every timestamp:

1. Window over the signal: 2.5 s
2. Compute feature vector: 29 statistical measures (Min, Max, Shannon energy, Percentile,...) over three representations of the signal



3. Classification model: Random Forest [1], based on **decision trees**

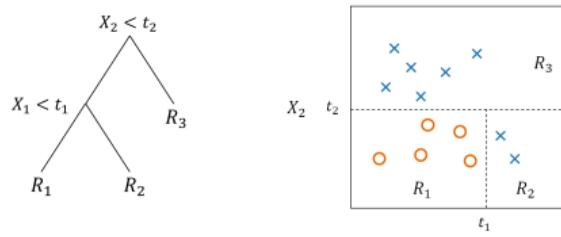
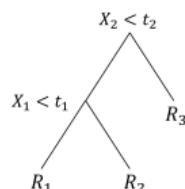
### Decision tree

Feature space  $\mathcal{X} = \mathbb{R}^Q$ . Division of  $\mathcal{X}$  into non-overlapping regions  $R_1, \dots, R_j$ . Algorithm CART: recursive binary splits [2] that solve:

$$\arg \max_{X_q, \tau} \text{IG}, \quad (\text{information gain})$$

$$\text{with } \text{IG}(X_q, \tau) = I(n) - \frac{N_l}{N_n} I(l) - \frac{N_r}{N_n} I(r),$$

$$\text{and } I(n) = \text{Gini}(n) = \sum_k p_{nk}(1 - p_{nk}).$$



$$\text{Prediction function: } f(x) = \sum_{j=1}^J c_j \mathbb{1}(x \in R_j)$$

## Method

### Model

#### Random forest

Decision trees  $d_1, \dots, d_{N_T}$  grown with two rules:

- ▶ Each tree is trained with a *bootstrap* of the training set
- ▶ At each split, access to a random subset of pool of features

Each tree is a “vote” for a class. The estimated probability of belonging to class  $k$  is then:

$$f_k(x) = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbb{1}(d_i(x) = k)$$

## Method

### Model

#### Random forest

Decision trees  $d_1, \dots, d_{N_T}$  grown with two rules:

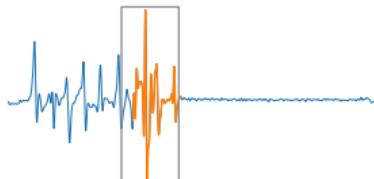
- ▶ Each tree is trained with a *bootstrap* of the training set
- ▶ At each split, access to a random subset of pool of features

Each tree is a “vote” for a class. The estimated probability of belonging to class  $k$  is then:

$$f_k(x) = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbb{1}(d_i(x) = k)$$

#### Data augmentation

Select  $r$  windows in training signals



$$\begin{bmatrix} x_{1,1} & \dots & x_{1,Q} \\ x_{2,1} & \dots & x_{2,Q} \end{bmatrix}$$

## Method

### Model

#### Random forest

Decision trees  $d_1, \dots, d_{N_T}$  grown with two rules:

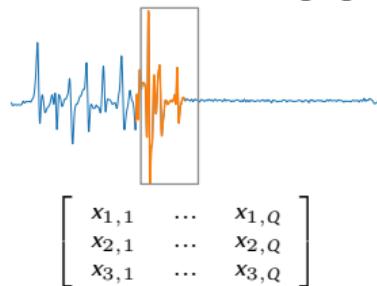
- ▶ Each tree is trained with a *bootstrap* of the training set
- ▶ At each split, access to a random subset of pool of features

Each tree is a “vote” for a class. The estimated probability of belonging to class  $k$  is then:

$$f_k(x) = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbb{1}(d_i(x) = k)$$

#### Data augmentation

Select  $r$  windows in training signals



## Method

### Model

#### Random forest

Decision trees  $d_1, \dots, d_{N_T}$  grown with two rules:

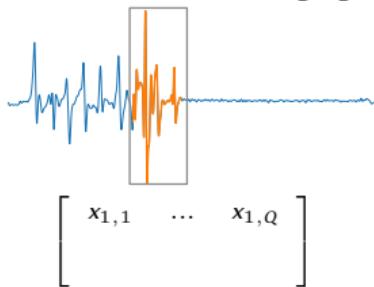
- ▶ Each tree is trained with a *bootstrap* of the training set
- ▶ At each split, access to a random subset of pool of features

Each tree is a “vote” for a class. The estimated probability of belonging to class  $k$  is then:

$$f_k(x) = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbb{1}(d_i(x) = k)$$

#### Data augmentation

Select  $r$  windows in training signals



## Method

### Model

#### Random forest

Decision trees  $d_1, \dots, d_{N_T}$  grown with two rules:

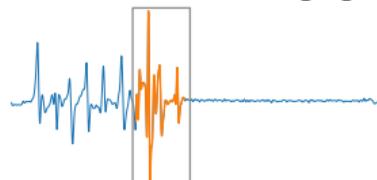
- ▶ Each tree is trained with a *bootstrap* of the training set
- ▶ At each split, access to a random subset of pool of features

Each tree is a “vote” for a class. The estimated probability of belonging to class  $k$  is then:

$$f_k(x) = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbb{1}(d_i(x) = k)$$

#### Data augmentation

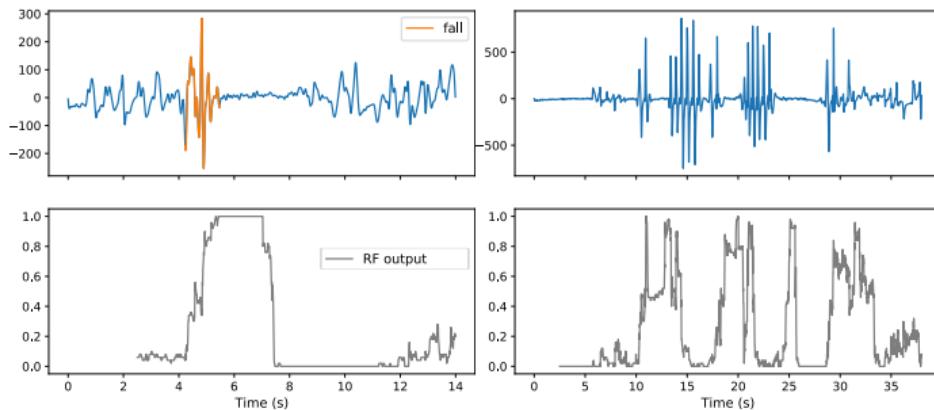
Select  $r$  windows in training signals



$$\begin{bmatrix} x_{1,1} & \dots & x_{1,Q} \\ \vdots & & \vdots \\ x_{r,1} & \dots & x_{r,Q} \end{bmatrix}$$

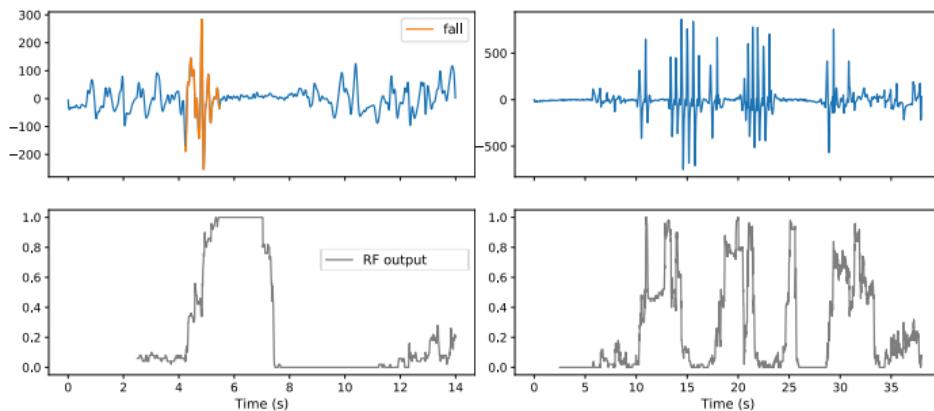
## Method

### Time aggregation



## Method

### Time aggregation



### Time aggregation

$N_f(t)$  : number of trees voting for *fall*

Use a buffer  $B_s \in \mathbb{N}$  and a threshold  $T_h \in [0, 1]$

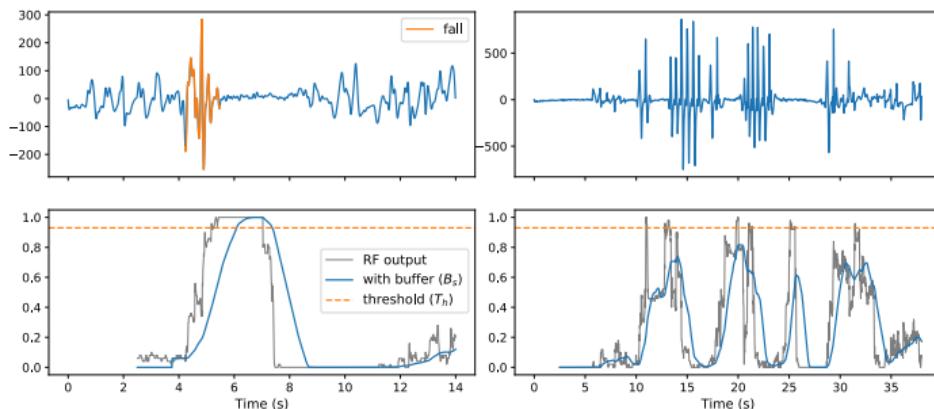
$$g(t) = \frac{\sum_{u=t-B_s+1}^t N_f(u)}{B_s \times N_T}$$

New binary classification function:

$$d(t) = \begin{cases} 1, & \text{if } g(t) > T_h \\ 0, & \text{otherwise} \end{cases}$$

## Method

### Time aggregation



### Time aggregation

$N_f(t)$  : number of trees voting for *fall*

Use a buffer  $B_s \in \mathbb{N}$  and a threshold  $T_h \in [0, 1]$

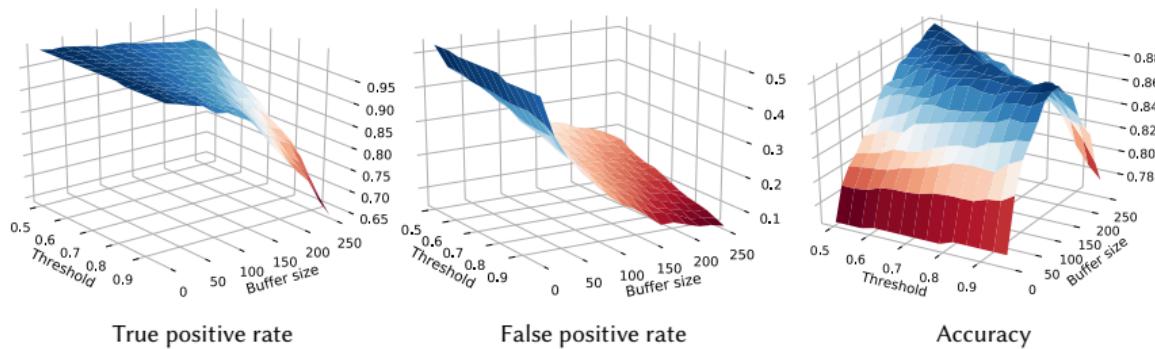
$$g(t) = \frac{\sum_{u=t-B_s+1}^t N_f(u)}{B_s \times N_T}$$

New binary classification function:

$$d(t) = \begin{cases} 1, & \text{if } g(t) > T_h \\ 0, & \text{otherwise} \end{cases}$$

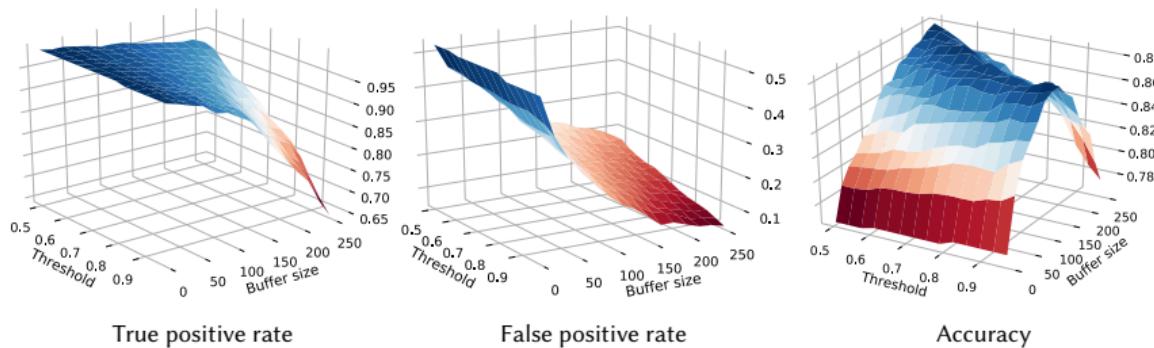
## Methods

### Time aggregation



## Methods

### Time aggregation



True positive rate

False positive rate

Accuracy

Decision rule is ready. Is it implementable ?

## Feature selection

### Feature importance

$$\text{Tree: } I(X_q) = \sum_{\text{nodes } t} p(t) \Delta i(t) \mathbb{1}(v(t) = X_q)$$

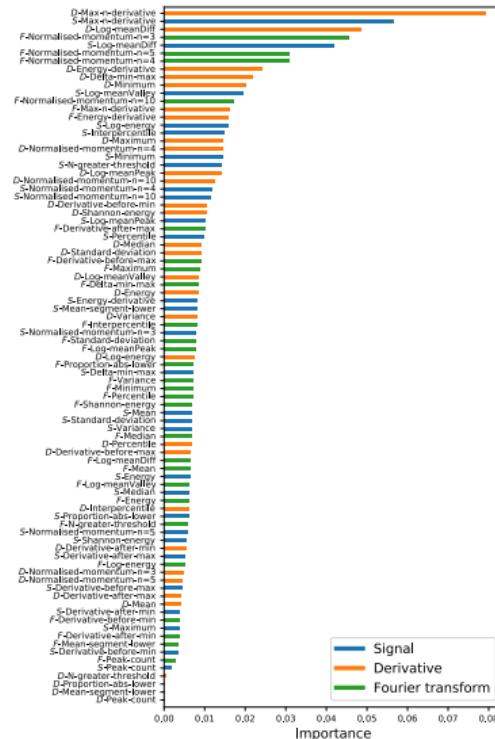
$$\text{Random forest: } I(X_q) = \frac{1}{N_T} \sum_{n=1}^{N_T} I(T_n, X_q)$$

## Feature selection

### Feature importance

$$\text{Tree: } I(X_q) = \sum_{\text{nodes } t} p(t) \Delta i(t) \mathbb{1}(v(t) = X_q)$$

$$\text{Random forest: } I(X_q) = \frac{1}{N_T} \sum_{n=1}^{N_T} I(T_n, X_q)$$



## Feature selection

### Feature importance

$$\text{Tree: } I(X_q) = \sum_{\text{nodes } t} p(t) \Delta i(t) \mathbb{1}(v(t) = X_q)$$

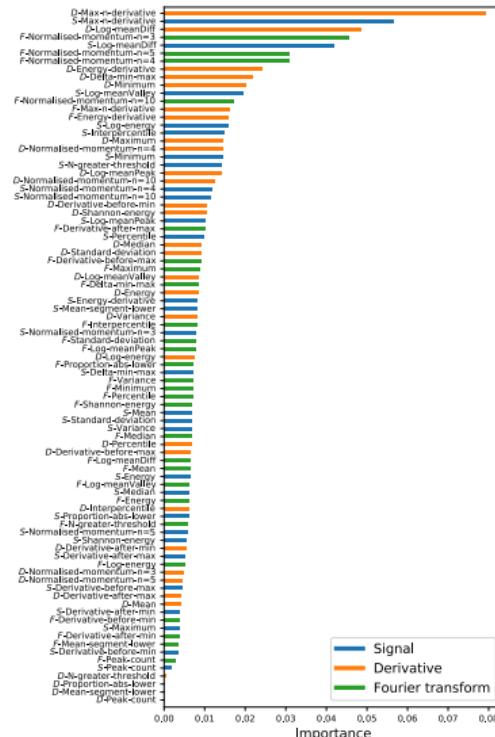
$$\text{Random forest: } I(X_q) = \frac{1}{N_T} \sum_{n=1}^{N_T} I(T_n, X_q)$$

### Recursive feature elimination

Initial pool of  $Q$  features  $X_1, \dots, X_Q$ .

1. Train several times and record variable importances
2. Average of importances over trainings.  

$$X_{q^*} = \arg \min_{X_i} I(X_i)$$
3. Remove  $X_{q^*}$  from the pool of features and back to step 1



## Feature selection

### Feature importance

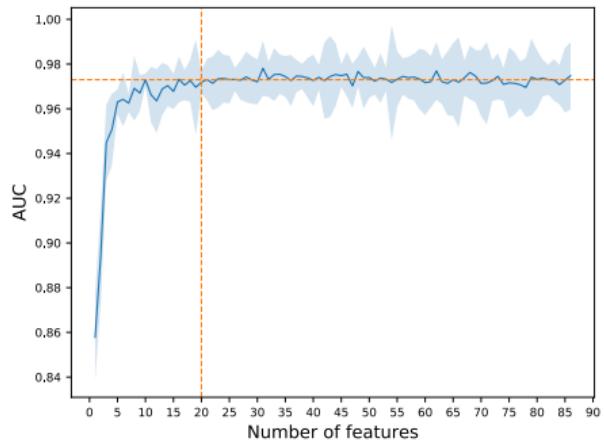
$$\text{Tree: } I(X_q) = \sum_{\text{nodes } t} p(t) \Delta i(t) \mathbb{1}(v(t) = X_q)$$

$$\text{Random forest: } I(X_q) = \frac{1}{N_T} \sum_{n=1}^{N_T} I(T_n, X_q)$$

### Recursive feature elimination

Initial pool of  $Q$  features  $X_1, \dots, X_Q$ .

1. Train several times and record variable importances
2. Average of importances over trainings.  
 $X_{q^*} = \arg \min_{X_i} I(X_i)$
3. Remove  $X_{q^*}$  from the pool of features and back to step 1



## Results

### Set up:

- ▶ Fixed params:  $r$  set to 5 and  $Q$  set to 20
- ▶ Varying  $T_h$  (0.5 to 1) and  $B_s$  (5 to 250)
- ▶ Record best Accuracy and show corresponding TPR, FPR
- ▶ If FPR is constrained to be max 10%, what is the range of TPRs ?

Model	Accuracy	TPR	FPR	$TPR_{min}^{FPR < 10}$	$TPR_{max}^{FPR < 10}$
$r = 5, Q = 20$					
LR	$86.8 \pm 1.5$	$90.5 \pm 2.4$	$17.7 \pm 4.9$	$67.0 \pm 10.8$	$80.4 \pm 6.4$
LDA	$85.5 \pm 1.2$	$91.0 \pm 2.1$	$21.7 \pm 3.7$	$56.9 \pm 7.0$	$78.7 \pm 3.8$
k-NN	$87.0 \pm 1.9$	$89.2 \pm 1.4$	$16.0 \pm 4.7$	$63.1 \pm 4.2$	$83.1 \pm 2.5$
SVM	$87.6 \pm 3.2$	$90.0 \pm 4.5$	$15.5 \pm 6.8$	$69.2 \pm 2.1$	$82.9 \pm 3.2$
MLP	$88.2 \pm 1.5$	$92.4 \pm 1.2$	$17.3 \pm 4.1$	$71.4 \pm 4.5$	$85.1 \pm 2.1$
RF	$88.2 \pm 1.5$	$91.7 \pm 3.5$	$16.2 \pm 6.2$	$63.8 \pm 6.8$	$84.3 \pm 7.9$

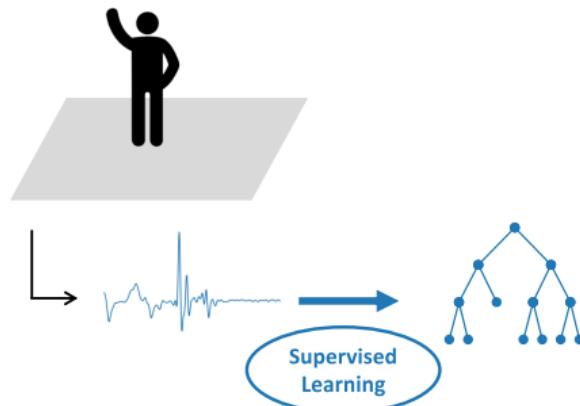
### Comments:

- ▶ Parametric methods perform slightly worse than non-parametric
- ▶ RF is slightly better than others
- ▶ Buffer/threshold trade-off useful for maintaining low FPR while improving TPR

## Transfer learning on decision tree

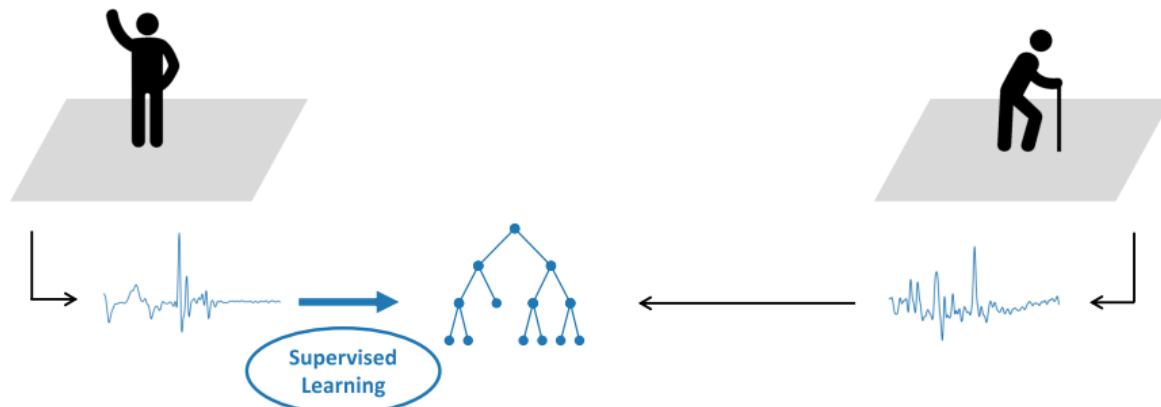
## Context

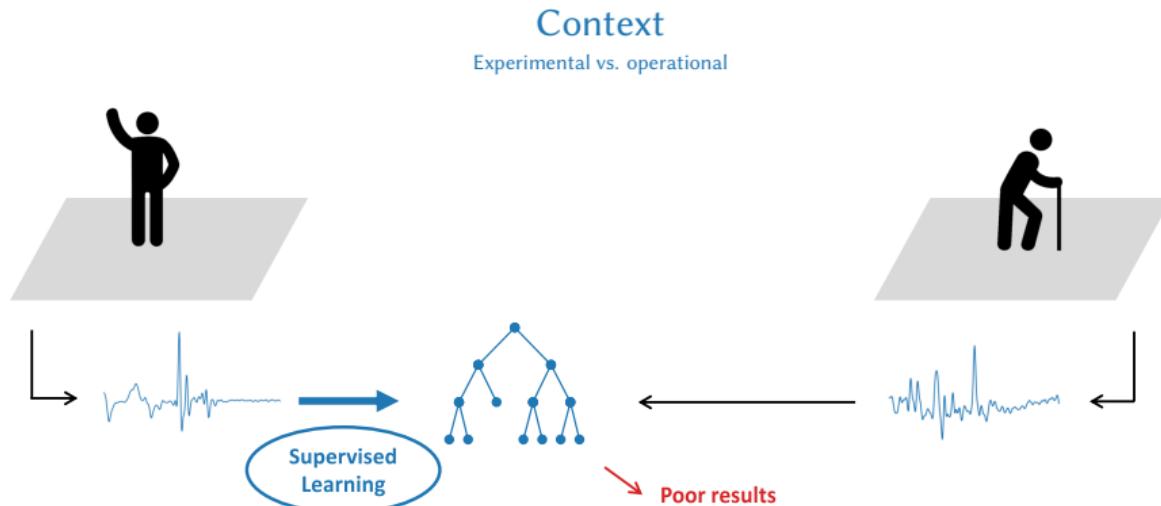
Experimental vs. operational



## Context

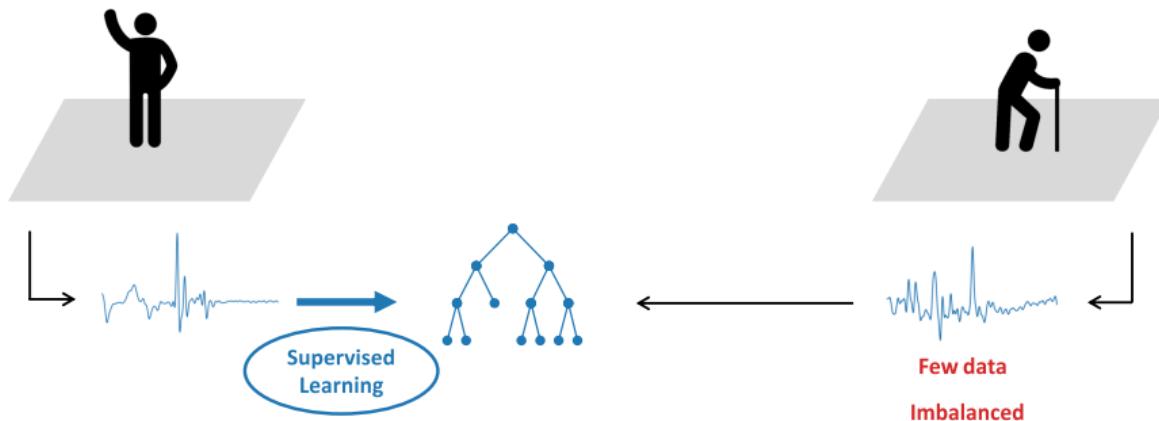
Experimental vs. operational





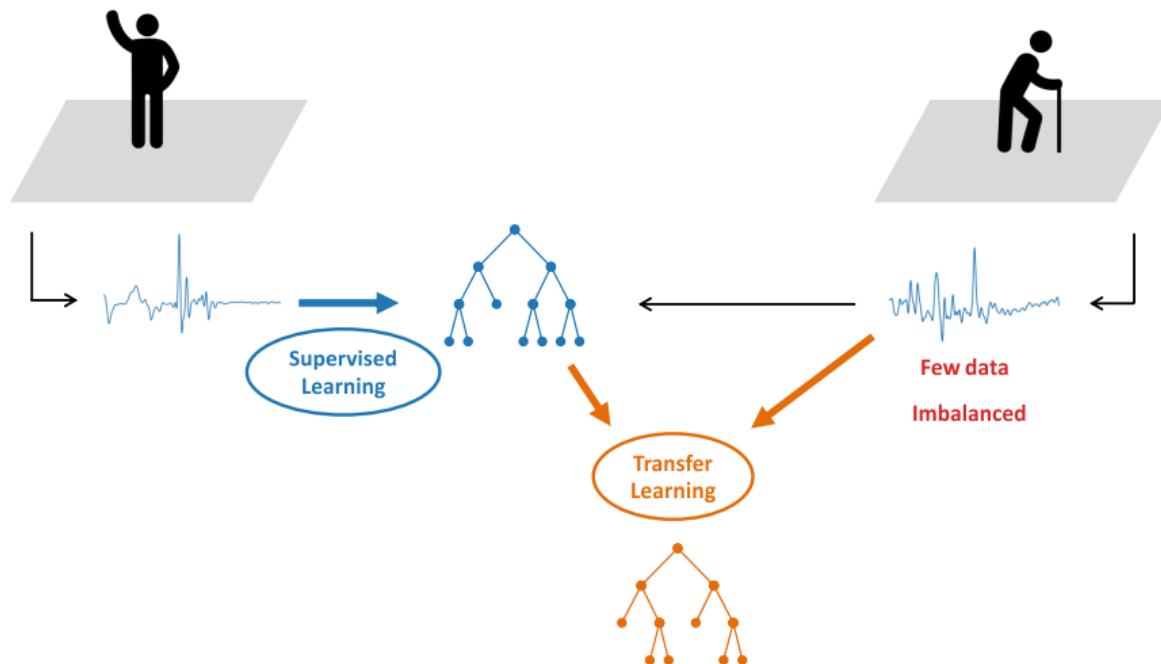
## Context

Experimental vs. operational



## Context

Experimental vs. operational



## Context

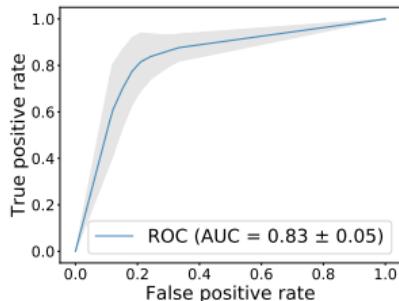
Transfer learning

### Transfer learning

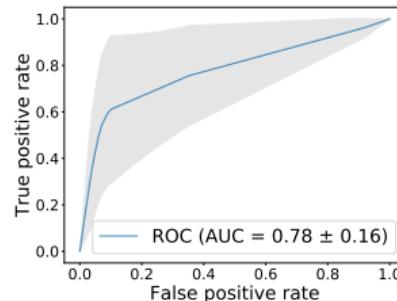
- ▶ Source domain:  $\mathcal{D}_S = \{\mathcal{X}_S, P(X_S)\}$
- ▶ Target domain:  $\mathcal{D}_T = \{\mathcal{X}_T, P(X_T)\}$
- ▶ Source task:  $\mathcal{T}_S = \{\mathcal{Y}_S, f^S\}$
- ▶ Target task:  $\mathcal{T}_T = \{\mathcal{Y}_T, f^T\}$

### Our case

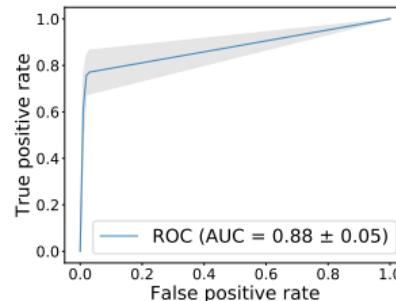
- ▶  $\mathcal{X}_S = \mathcal{X}_T$
- ▶  $P(X_S) \neq P(X_T)$
- ▶  $\mathcal{Y}_S = \mathcal{Y}_T$
- ▶  $f^S \neq f^T$



Source tested on source



Source tested on target

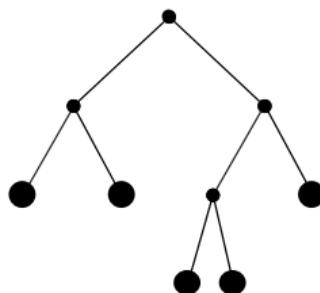


Target tested on target

## Model-based transfer

Segev et al. [7]

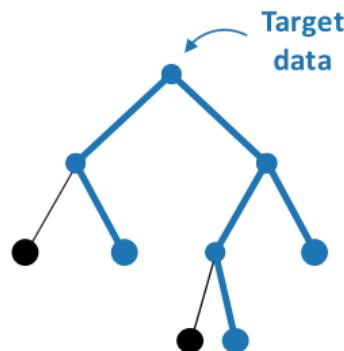
### Structure Expansion / Reduction (SER)



## Model-based transfer

Segev et al. [7]

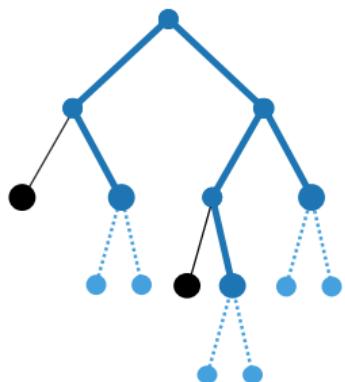
### Structure Expansion / Reduction (SER)



## Model-based transfer

Segev et al. [7]

### Structure Expansion / Reduction (SER)

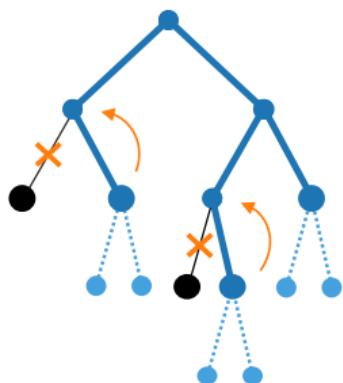


1. Expansion

## Model-based transfer

Segev et al. [7]

### Structure Expansion / Reduction (SER)

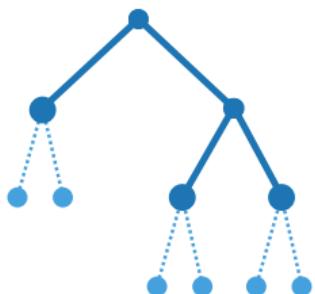


1. Expansion
2. Reduction

## Model-based transfer

Segev et al. [7]

### Structure Expansion / Reduction (SER)

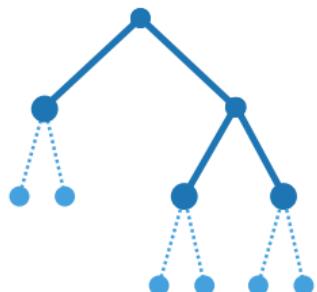


1. Expansion
2. Reduction

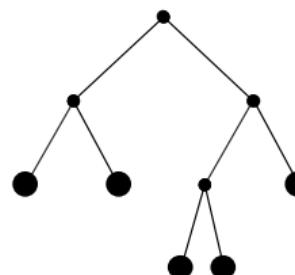
## Model-based transfer

Segev et al. [7]

### Structure Expansion / Reduction (SER)



### Structure Transfer (STRUT)

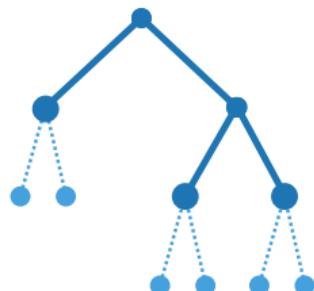


1. Expansion
2. Reduction

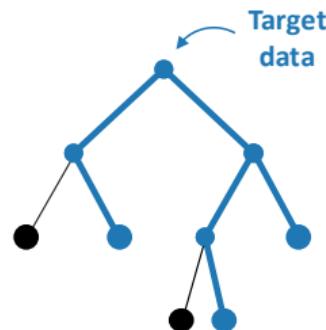
## Model-based transfer

Segev et al. [7]

### Structure Expansion / Reduction (SER)



### Structure Transfer (STRUT)

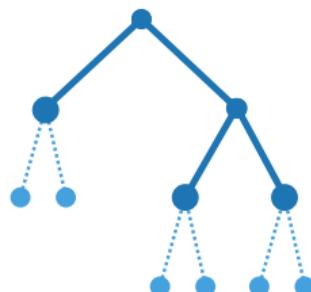


1. Expansion
2. Reduction

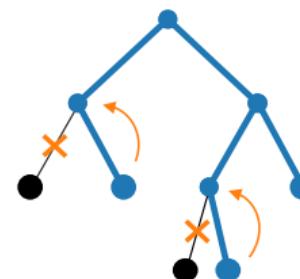
## Model-based transfer

Segev et al. [7]

### Structure Expansion / Reduction (SER)



### Structure Transfer (STRUT)



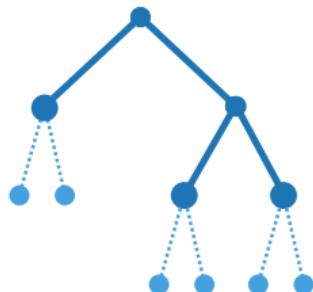
1. Pruning

1. Expansion
2. Reduction

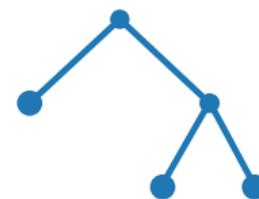
## Model-based transfer

Segev et al. [7]

### Structure Expansion / Reduction (SER)



### Structure Transfer (STRUT)



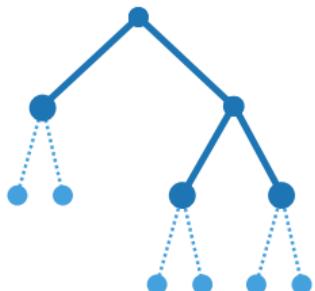
1. Pruning

1. Expansion
2. Reduction

## Model-based transfer

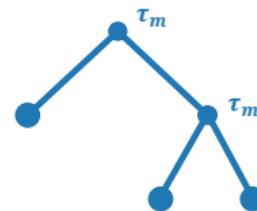
Segev et al. [7]

### Structure Expansion / Reduction (SER)



1. Expansion
2. Reduction

### Structure Transfer (STRUT)

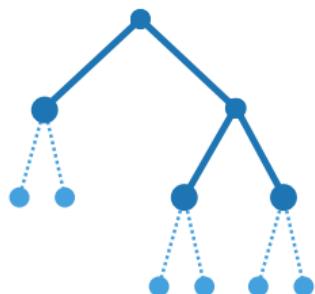


1. Pruning
2. Threshold update

## Model-based transfer

Segev et al. [7]

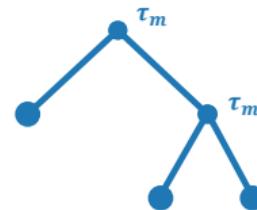
### Structure Expansion / Reduction (SER)



1. Expansion
2. Reduction

Partition refinement or simplification

### Structure Transfer (STRUT)



1. Pruning
2. Threshold update

Drifts

## Leaf loss risk

### Homogeneous class imbalance

$$P^T(x|y) = P^S(x|y)$$

$$P^T(y|x) = \lambda_y \frac{P^S(y|x)}{\int \lambda_y P^S(y|x) dy}$$

$$\text{with } \lambda_y = \frac{P^T(y)}{P^S(y)}$$

## Leaf loss risk

### Homogeneous class imbalance

$$P^T(x|y) = P^S(x|y)$$

$$P^T(y|x) = \lambda_y \frac{P^S(y|x)}{\int \lambda_y P^S(y|x) dy}$$

$$\text{with } \lambda_y = \frac{P^T(y)}{P^S(y)}$$

### Leaf loss risk

Significant leaf: Leaf  $l$  that conserves the minority class  $k_{min}$  after Target update:

$$\forall k \neq k_{min}, \quad P^T(y = k_{min} | x \in l) > P^T(y = k | x \in l)$$

Leaf loss risk:

$$R_L(l) = P^T(x \notin l | y = k_{min})^{n_{k_{min}}}$$

## Leaf loss risk

### Homogeneous class imbalance

$$P^T(x|y) = P^S(x|y)$$

$$P^T(y|x) = \lambda_y \frac{P^S(y|x)}{\int \lambda_y P^S(y|x) dy}$$

with  $\lambda_y = \frac{P^T(y)}{P^S(y)}$

### Leaf loss risk

Significant leaf: Leaf  $l$  that conserves the minority class  $k_{min}$  after Target update:

$$\forall k \neq k_{min}, \quad P^T(y = k_{min} | x \in l) > P^T(y = k | x \in l)$$

Leaf loss risk:

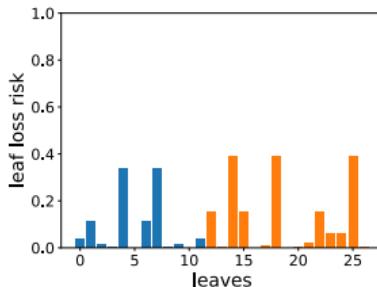
$$R_L(l) = P^T(x \notin l | y = k_{min})^{n_{k_{min}}}$$

### Leaf loss risk under homogeneous class imbalance

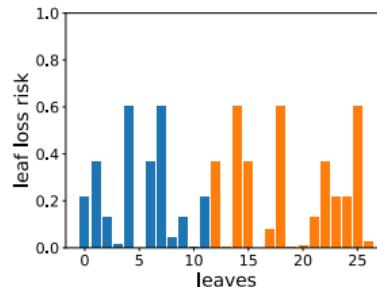
$$\forall k \neq k_{min}, \quad \lambda_{k_{min}} P^S(y = k_{min} | x \in l) > \lambda_k P^S(y = k | x \in l)$$

$$R_L(l) = P^S(x \notin l | y = k_{min})^{n_{k_{min}}}$$

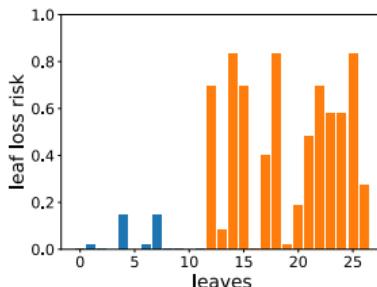
## Leaf loss risk



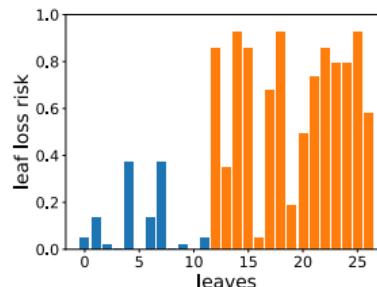
(a) Balanced data with 200 instances



(b) Balanced data with 100 instances



(c) Imbalanced data (10% ratio) with 200 instances

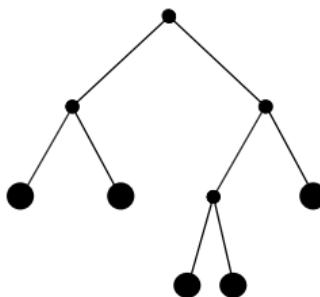


(d) Imbalanced data (10% ratio) with 100 instances

## SER for class imbalance

$SER_R, SER_{LL}$

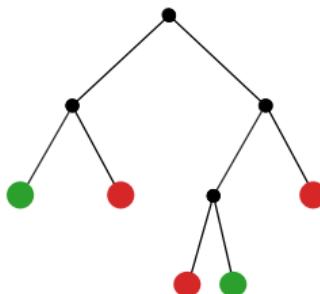
### Structure Expansion and controlled Reduction



## SER for class imbalance

$SER_R, SER_{LL}$

### Structure Expansion and controlled Reduction

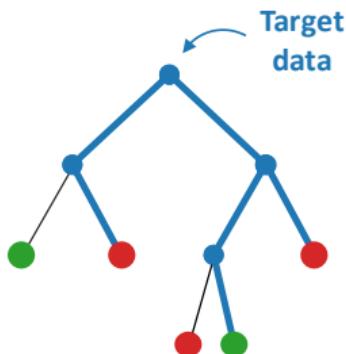


**Minority class**

## SER for class imbalance

$SER_R, SER_{LL}$

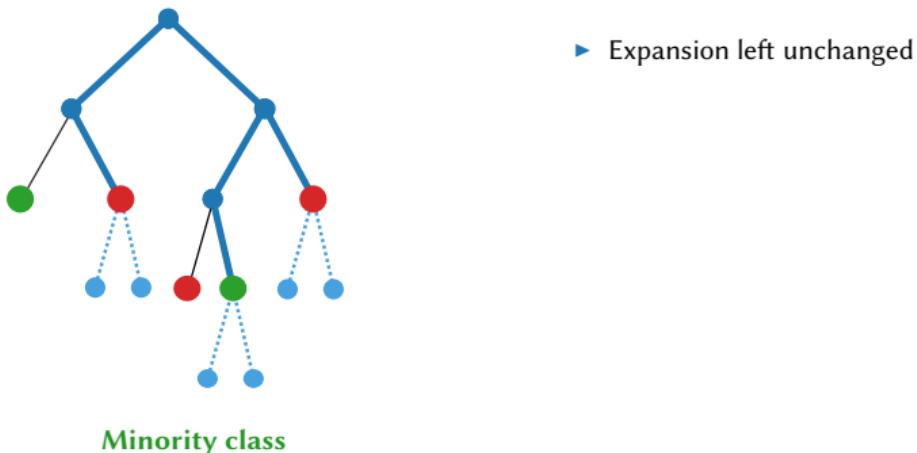
### Structure Expansion and controlled Reduction



## SER for class imbalance

$SER_R, SER_{LL}$

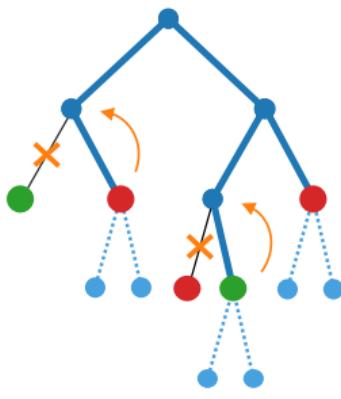
### Structure Expansion and controlled Reduction



## SER for class imbalance

$SER_R, SER_{LL}$

### Structure Expansion and controlled Reduction



- ▶ Expansion left unchanged
- ▶ Reduction constrained

#### $SER_R$

If node is of minority class, then no pruning

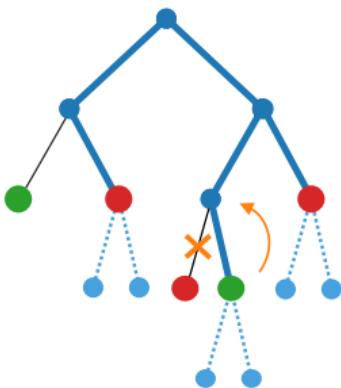
#### $SER_{LL}$

If node is of minority class **and** still significant considering Target **and**  $R_L > 0.5$ , then no pruning

## SER for class imbalance

$SER_R, SER_{LL}$

### Structure Expansion and controlled Reduction



- ▶ Expansion left unchanged
- ▶ Reduction constrained

#### $SER_R$

If node is of minority class, then no pruning

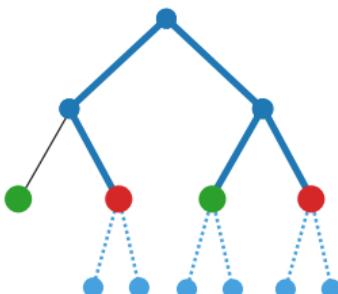
#### $SER_{LL}$

If node is of minority class **and** still significant considering Target **and**  $R_L > 0.5$ , then no pruning

## SER for class imbalance

$SER_R, SER_{LL}$

### Structure Expansion and controlled Reduction



Minority class

- ▶ Expansion left unchanged
- ▶ Reduction constrained

#### $SER_R$

If node is of minority class, then no pruning

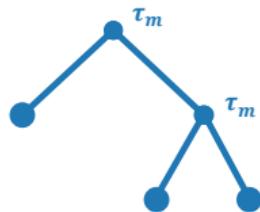
#### $SER_{LL}$

If node is of minority class **and** still significant considering Target **and**  $R_L > 0.5$ , then no pruning

## STRUT for class imbalance

STRUT optimization

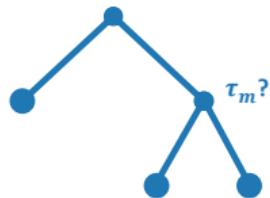
**STRUT: How are updated the new thresholds ?**



## STRUT for class imbalance

STRUT optimization

**STRUT: How are updated the new thresholds ?**

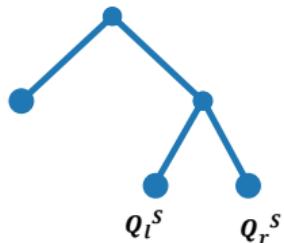


## STRUT for class imbalance

STRUT optimization

**STRUT: How are updated the new thresholds ?**

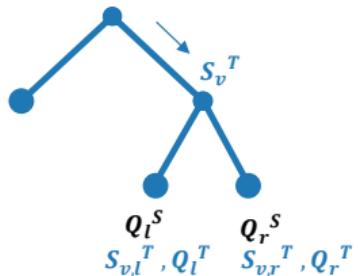
$Q_l^S, Q_r^S$ : class proportions of source data in children w.r.t. the *original* split



## STRUT for class imbalance

### STRUT optimization

**STRUT: How are updated the new thresholds ?**



$Q_l^S, Q_r^S$ : class proportions of source data in children w.r.t. the *original* split

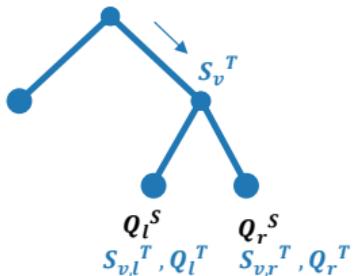
$S_{v,l}^T, S_{v,r}^T$ : subsets of  $S_v^T$  that fall in the children nodes of  $v$

$Q_l^T(\tau), Q_r^T(\tau)$ : class proportions of target data in children w.r.t. the *new* split

## STRUT for class imbalance

STRUT optimization

**STRUT: How are updated the new thresholds ?**



$Q_l^S, Q_r^S$ : class proportions of source data in children w.r.t. the *original* split

$S_{v,l}^T, S_{v,r}^T$ : subsets of  $S_v^T$  that fall in the children nodes of  $v$

$Q_l^T(\tau), Q_r^T(\tau)$ : class proportions of target data in children w.r.t. the *new* split

**Divergence Gain:** similarity between the original label distributions and the new ones

$$DG(\tau) = 1 - \frac{|S_{v,l}^T|}{|S_v^T|} JSD(Q_l^S, Q_l^T) - \frac{|S_{v,r}^T|}{|S_v^T|} JSD(Q_r^S, Q_r^T)$$

Jensen-Shannon divergence:

$$JSD(P, Q) = \frac{1}{2} (D_{KL}(P||M) + D_{KL}(Q||M))$$

$$M = \frac{1}{2} (P + Q)$$

Kullback-Leibler divergence:

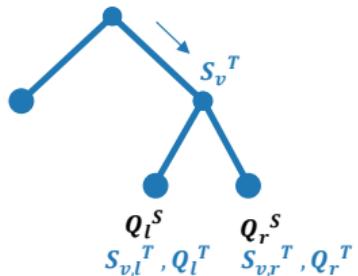
$$D_{KL}(P||Q) = \sum_k P(k) \ln \left( \frac{P(k)}{Q(k)} \right)$$

## STRUT for class imbalance

STRUT optimization

**STRUT:** How are updated the new thresholds ?

**Goal:** Maximize DG while being in a local maximum of Information Gain (IG) (here IG = Gini gain)

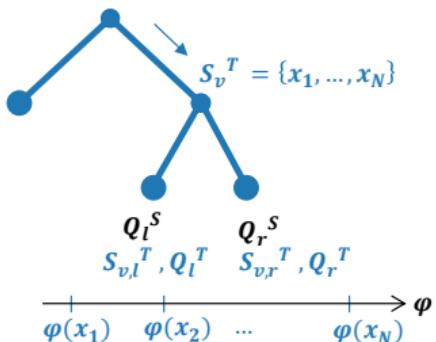


## STRUT for class imbalance

STRUT optimization

**STRUT: How are updated the new thresholds ?**

**Goal:** Maximize DG while being in a local maximum of Information Gain (IG) (here IG = Gini gain)

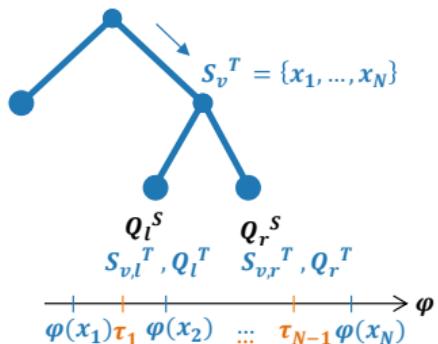


## STRUT for class imbalance

STRUT optimization

**STRUT: How are updated the new thresholds ?**

**Goal:** Maximize DG while being in a local maximum of Information Gain (IG) (here IG = Gini gain)

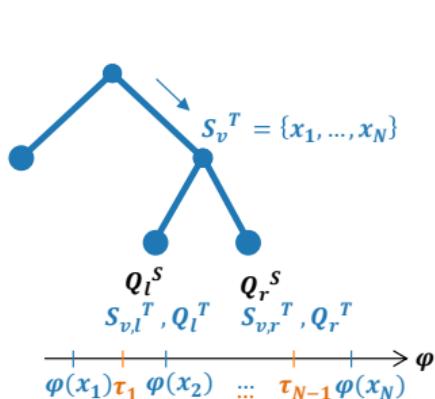


## STRUT for class imbalance

STRUT optimization

**STRUT: How are updated the new thresholds ?**

**Goal:** Maximize DG while being in a local maximum of Information Gain (IG) (here IG = Gini gain)



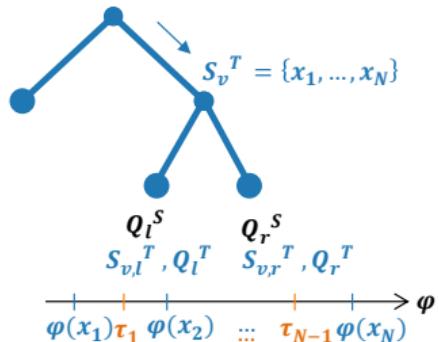
$$\tau_m = \arg \max_{\tau \in T_v} (DG(\tau, Q_l^T(\tau), Q_r^T(\tau)))$$

$$\text{s.t. } IG(\tau_{m-1}) < IG(\tau_m) \text{ and } IG(\tau_m) > IG(\tau_{m+1})$$

## STRUT for class imbalance

STRUT optimization

**STRUT: How are updated the new thresholds ?**



**Goal:** Maximize DG while being in a local maximum of Information Gain (IG) (here IG = Gini gain)

$$\tau_m = \arg \max_{\tau \in T_v} (DG(\tau, Q_l^T(\tau), Q_r^T(\tau)))$$

$$\text{s.t. } IG(\tau_{m-1}) < IG(\tau_m) \text{ and } IG(\tau_m) > IG(\tau_{m+1})$$

- ▶  $Q^S$  have less meaning when going deeper
- ▶ Do we really want to keep  $Q^S$  and  $Q^T$  close ?

## STRUT for class imbalance

STRUT<sub>IG</sub>, STRUT<sub>HI</sub>

### STRUT<sub>IG</sub>

STRUT without DG: maximization of IG

## STRUT for class imbalance

STRUT<sub>IG</sub>, STRUT<sub>HI</sub>

### STRUT<sub>IG</sub>

STRUT without DG: maximization of IG

### STRUT<sub>HI</sub>

Framework of homogeneous class imbalance, use

$$p^T(y/x) = \lambda_y \frac{p^S(y/x)}{\int \lambda_y p^S(y/x) dy}$$

to change the source class proportions in DG:

$$Q_l^{S'} = \lambda_k \frac{Q_l^S}{\sum_k \lambda_k Q_l^S} \quad Q_r^{S'} = \lambda_k \frac{Q_r^S}{\sum_k \lambda_k Q_r^S}$$

## STRUT for class imbalance

STRUT<sub>IG</sub>, STRUT<sub>HI</sub>

### STRUT<sub>IG</sub>

STRUT without DG: maximization of IG

### STRUT<sub>HI</sub>

Framework of homogeneous class imbalance, use

$$p^T(y/x) = \lambda_y \frac{p^S(y/x)}{\int \lambda_y p^S(y/x) dy}$$

to change the source class proportions in DG:

$$Q_l^{S'} = \lambda_k \frac{Q_l^S}{\sum_k \lambda_k Q_l^S} \quad Q_r^{S'} = \lambda_k \frac{Q_r^S}{\sum_k \lambda_k Q_r^S}$$

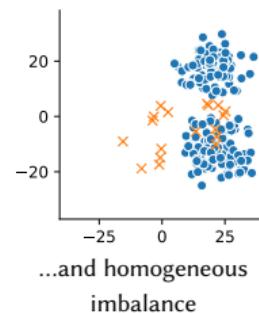
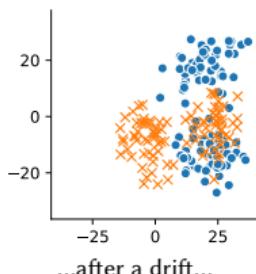
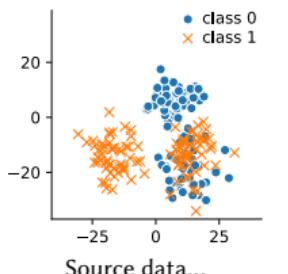
STRUT<sub>HI</sub> can be seen as a generalization of STRUT

## Results

### Synthetic data

#### Gaussian generator

Source dataset: combination of several multivariate Gaussian clusters. We use  $N_{\text{source}} = 200$  and  $N_{\text{clust}} = 10$ . Initial parameters are randomly drawn from Uniform distribution:  $\mu_i \in [-70, 70]$  and  $\sigma_i \in [5, 15]$



#### Transformations

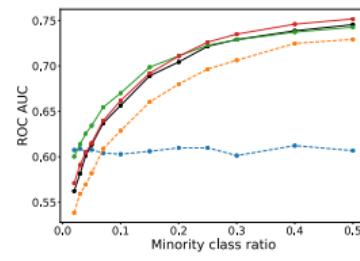
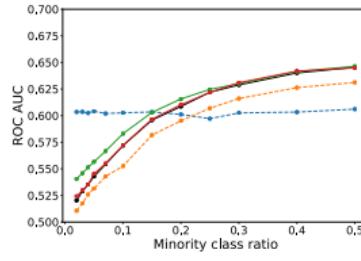
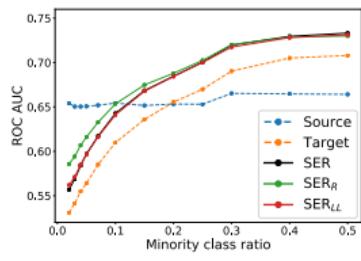
Basic transformations on Source clusters are applied to get a Target dataset:

- ▶ Drifts ( $\mu_i$ )
- ▶ Stretch / Squeeze ( $\sigma_i$ )
- ▶ Adding / Remove clusters

Transformations are combined with imbalance ratio (from 2% to 50%), and  $N_{\text{target}} = 1000$

# Results

Synthetic data



(a) Drift

(b) Stretch / Squeeze

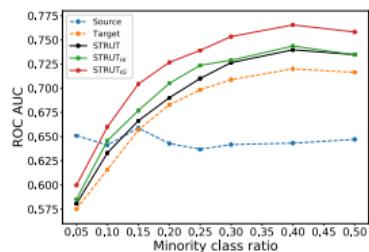
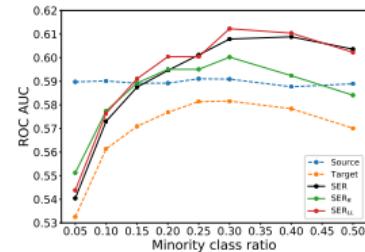
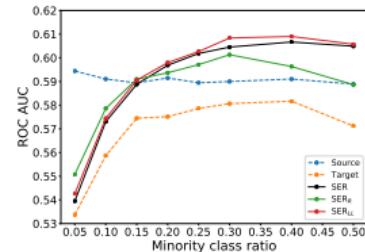
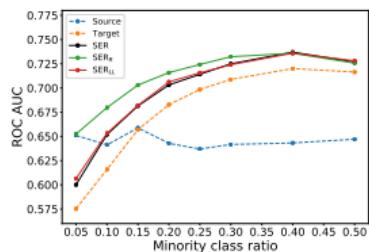
(c) Add / Remove

- ▶ Strong imbalance leads to negative transfer
- ▶ SER: SER<sub>LL</sub> close to SER, SER<sub>R</sub> better than SER (esp. with high imbalance)
- ▶ STRUT: both variants beat the original STRUT

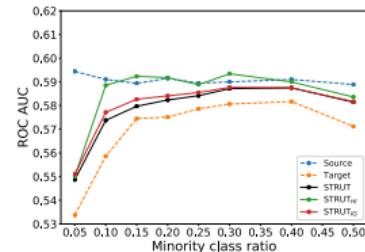
# Results

Public data

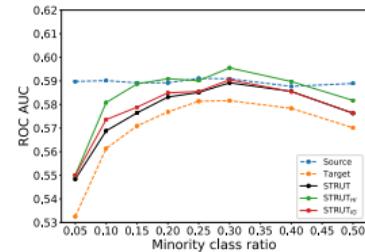
Two real data sets where the imbalance ratio is controlled with downsampling.



(a) Magic Gamma Telescope



(b) Office-Caltech: *amazon* → *webcam*



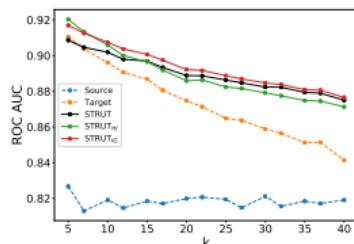
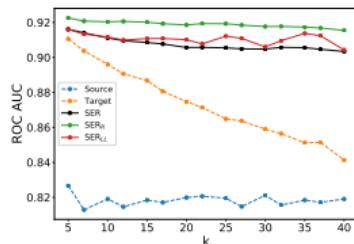
(c) Office-Caltech: *caltech* → *webcam*

- ▶ SER: SER<sub>LL</sub> close to SER, SER<sub>R</sub> either better... or worse
- ▶ STRUT: both variants perform as well or better

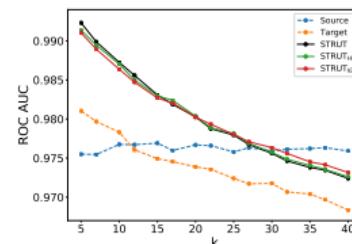
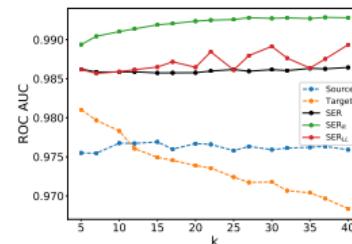
# Results

## Fall data

Operational data set: 174 fall events and 2619 non-fall events (6%)  
k-fold testing with varying  $k$  to observe lack of data



(a) Decision tree model



(b) Random forest model with 10 decision trees

- ▶ SER: variants give similar or better results
- ▶ STRUT: STRUT<sub>IG</sub> better, STRUT<sub>HI</sub> better *only when enough data*

- ▶ Same overall conclusions

## Results

### Overall

- ▶ SER<sub>LL</sub> quite close to original SER
- ▶ SER<sub>R</sub> generally better except with one real dataset
- ▶ Both STRUT variants give better results with one exception on fall data (when data is scarce)
- ▶ This suggests a data-dependency of the algorithms

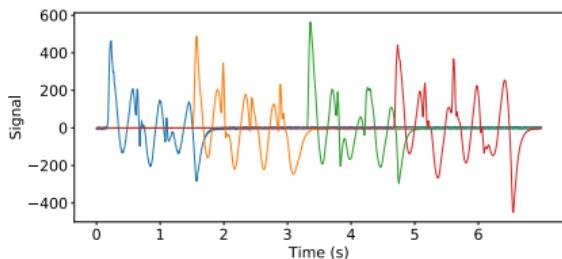
# Elderly activity recognition with convolutional representations

## Motivation

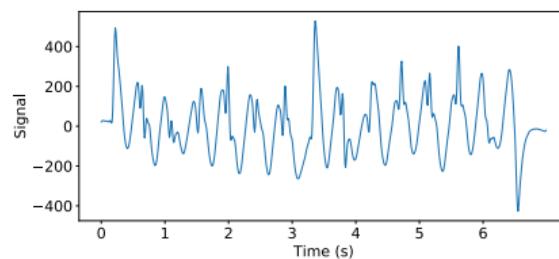
- ▶ **Issue:** one-dimensional signals for large areas
- ▶ Goal: Classify elderly from other individuals
  - ▶ Most signals are made of walks of staff individuals
- ▶ Subtask: Bring the model's attention over step-related signals

## Motivation

- ▶ **Issue:** one-dimensional signals for large areas
- ▶ Goal: Classify elderly from other individuals
  - ▶ Most signals are made of walks of staff individuals
- ▶ Subtask: Bring the model's attention over step-related signals



(a) Raw signal

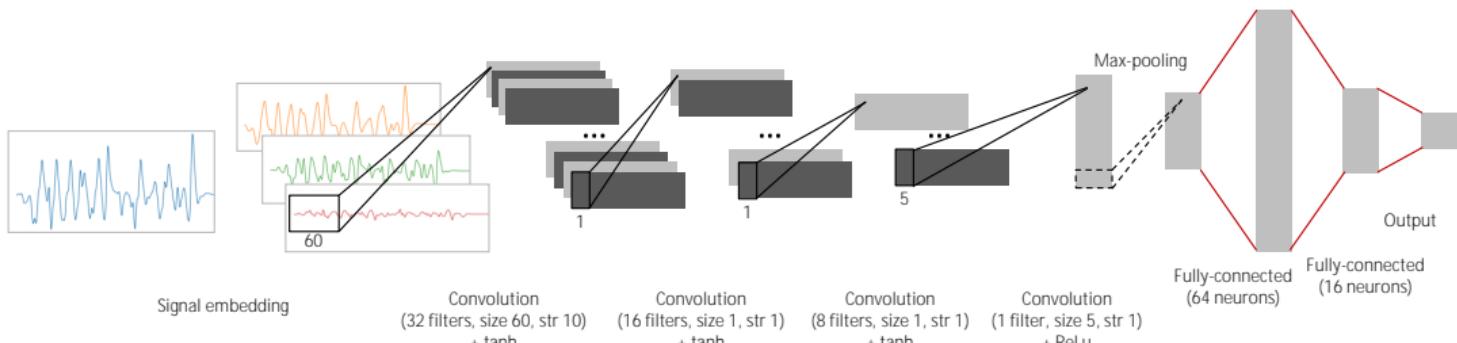


(b) Preprocessed signal

**What we propose:** A model to recognize elderly activity using a convolutional neural network and three training steps:

1. Signal embedding using convolutional dictionary learning
2. Step proposal network inspired from Region proposal network
3. The final classification task

## Main architecture



- ▶ Use the convolutional representation to “boost” training
- ▶ First layer (Signal embedding) of NURSENET is trained **separately** using convolutional dictionary learning

## Signal embedding

Convolutional dictionary learning

- ▶  $\mathbf{s}$  : data to be represented
- ▶ Objective : find  $M$  atoms  $\mathbf{d}_m$  and activation signals  $\mathbf{x}_m$  such that

$$\mathbf{s} \approx \sum_{m=1}^M \mathbf{x}_m * \mathbf{d}_m$$

- ▶  $*$  : convolution

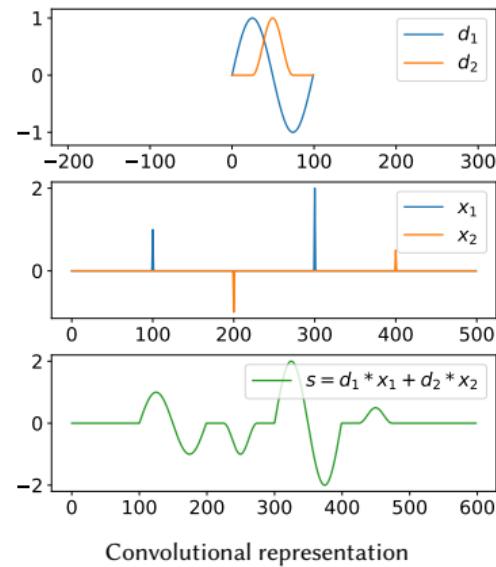
## Signal embedding

### Convolutional dictionary learning

- ▶  $s$  : data to be represented
- ▶ Objective : find  $M$  atoms  $\mathbf{d}_m$  and activation signals  $\mathbf{x}_m$  such that

$$s \approx \sum_{m=1}^M \mathbf{x}_m * \mathbf{d}_m$$

- ▶  $*$  : convolution



Convolutional representation

## Signal embedding

### Convolutional dictionary learning

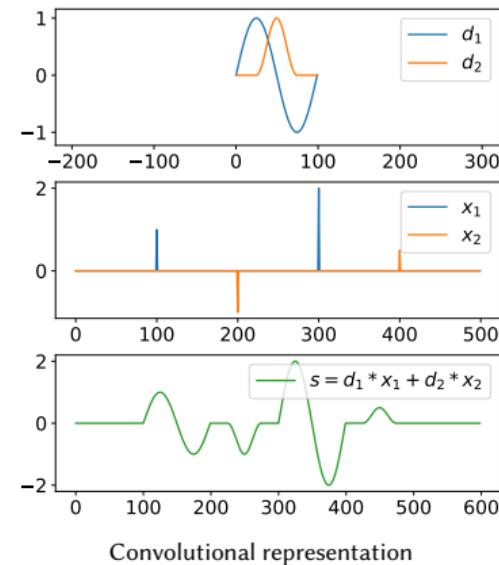
- ▶  $\mathbf{s}$  : data to be represented
- ▶ Objective : find  $M$  atoms  $\mathbf{d}_m$  and activation signals  $\mathbf{x}_m$  such that

$$\mathbf{s} \approx \sum_{m=1}^M \mathbf{x}_m * \mathbf{d}_m$$

- ▶  $*$  : convolution

CDL general problem:

$$\begin{aligned} \arg \min_{\mathbf{x}_m, \mathbf{d}_m} & \frac{1}{2} \left\| \sum_{m=1}^M \mathbf{x}_m * \mathbf{d}_m - \mathbf{s} \right\|_2^2 + \lambda \sum_{m=1}^M \|\mathbf{x}_m\|_1 \\ \text{s.t. } & \|\mathbf{d}_m\|_2 \leq 1 \quad \forall m . \end{aligned}$$

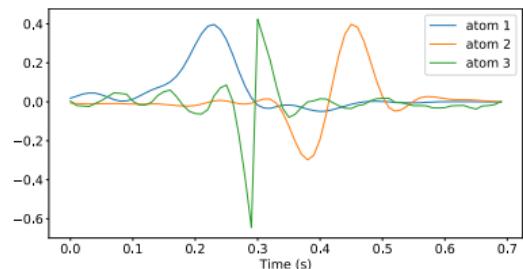


Convolutional representation

## Signal embedding

Learning step atoms

- ▶ Learning with Alternating Direction Method of Multipliers (ADMM) [3]
- ▶ 3 atoms of length 0.7 second

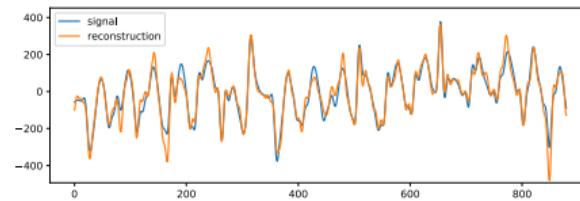
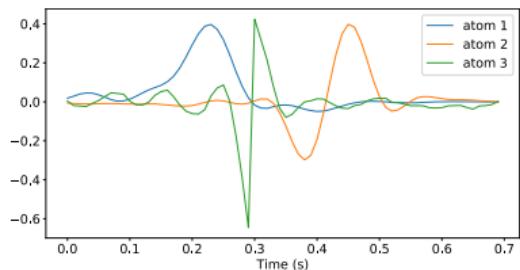


Dictionary

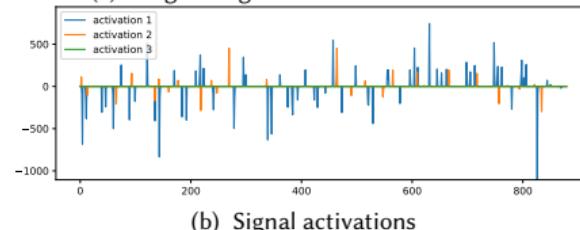
## Signal embedding

### Learning step atoms

- ▶ Learning with Alternating Direction Method of Multipliers (ADMM) [3]
- ▶ 3 atoms of length 0.7 second



(a) Original signal and its reconstruction



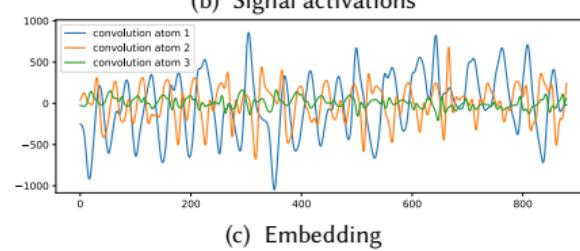
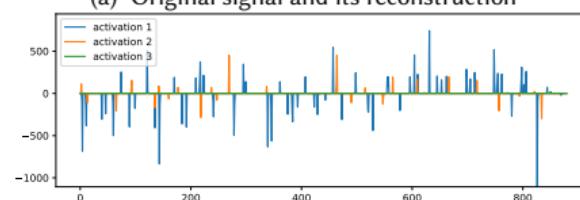
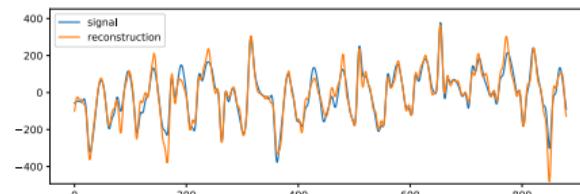
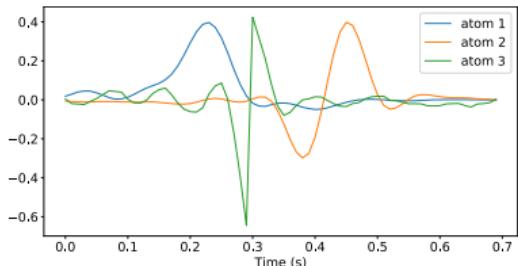
(b) Signal activations

## Signal embedding

### Learning step atoms

- ▶ Learning with Alternating Direction Method of Multipliers (ADMM) [3]
- ▶ 3 atoms of length 0.7 second
- ▶ Use the following embedding:

$$\mathbf{S} \doteq (\mathbf{s} * \mathbf{d}_m)_{1 \leq m \leq 3}$$



## Region proposal network

Object detection

- ▶ Classification: What is the image class ?

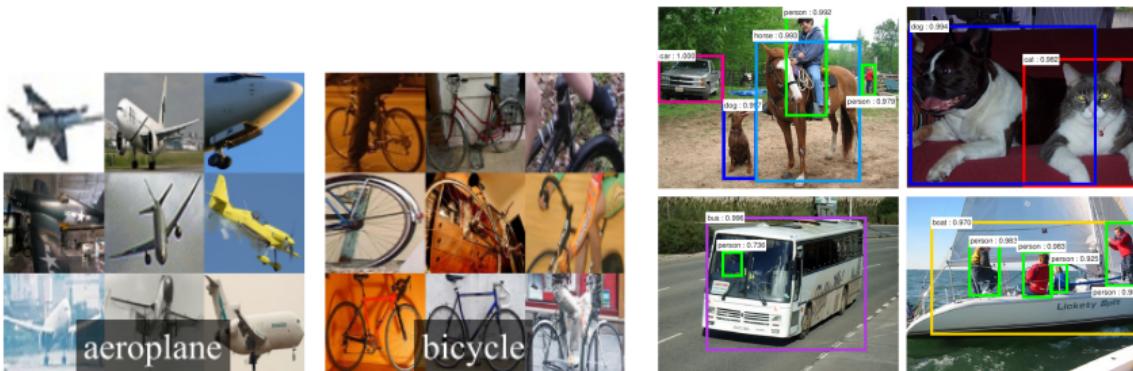


Classification vs Object detection. Source: Girshick et al. [4], Ren et al. [6]

## Region proposal network

Object detection

- ▶ Classification: What is the image class ?
- ▶ Object detection: Where are the objects and what are they classes ?

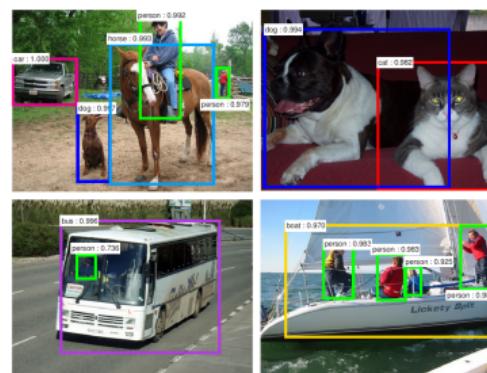


Classification vs Object detection. Source: Girshick et al. [4], Ren et al. [6]

## Region proposal network

Object detection

- ▶ Classification: What is the image class ?
- ▶ Object detection: Where are the objects and what are they classes ?
- ▶ How to efficiently localize objects ?
- ▶ Proposal models (Hosang et al. [5])
- ▶ Faster R-CNN (Ren et al. [6])

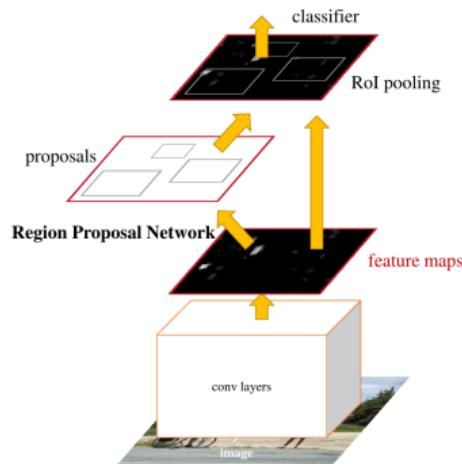


Classification vs Object detection. Source: Girshick et al. [4], Ren et al. [6]

## Region proposal network

Faster R-CNN

- ▶ Main idea: proposals are generated by a CNN called Region Proposal Network

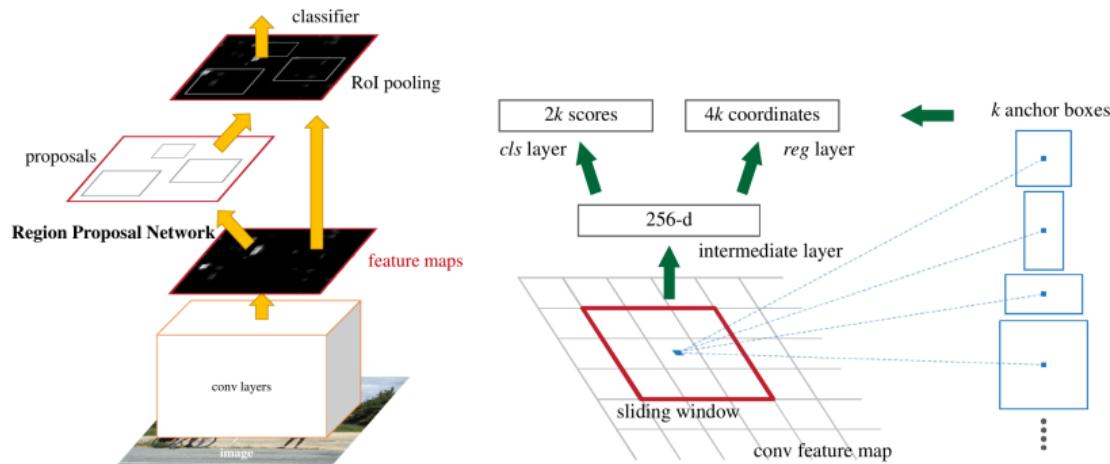


Region proposal network. Source: Ren et al. [6]

## Region proposal network

Faster R-CNN

- ▶ Main idea: proposals are generated by a CNN called Region Proposal Network
- ▶ A sliding window is passed: multiple *anchors* over each location (various sizes and scales)
- ▶ Two layers: Classification (Object / Not Object) and Regression (anchor coordinates)

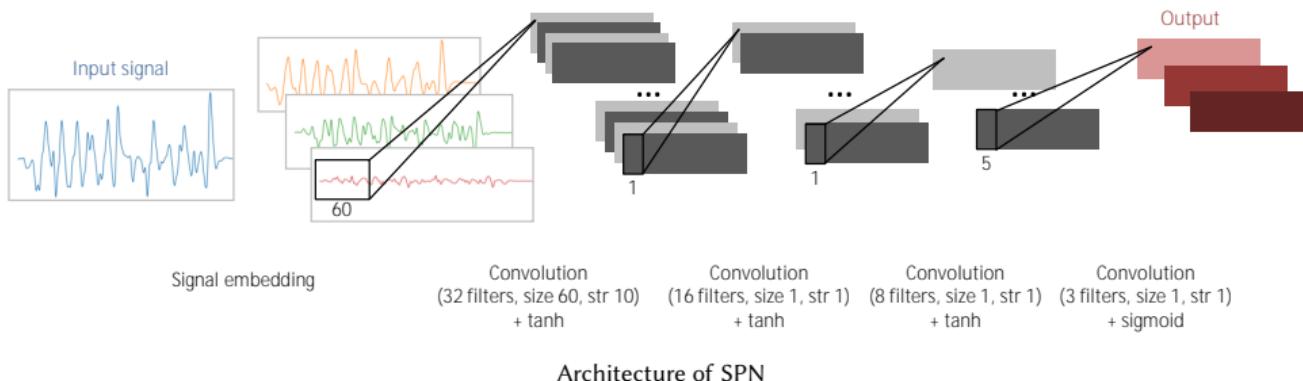


Region proposal network. Source: Ren et al. [6]

## Step proposal network

### Main architecture

- ▶ Directly inspired from RPN
- ▶ Simple architecture with three hidden layers, all **convolutional**
- ▶ Output: probability of having a step at a specific window location and size
  - ▶ Here 3 sizes and all discrete locations are considered

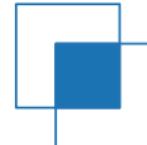


## Step proposal network

### Principle

- ▶ Objective of SPN : output boxes with largest Intersection over Union (IoU)
- ▶ IoU:  $\mathbf{b}_j$  are labelled boxes,  $\hat{b}$  is an estimated box:

$$\text{IoU}(\hat{b}) \doteq \max_j \frac{|\mathbf{b}_j \cap \hat{b}|}{|\mathbf{b}_j \cup \hat{b}|}$$

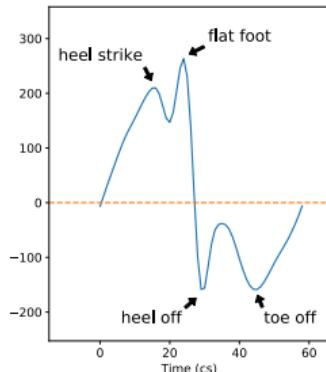
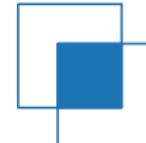


## Step proposal network

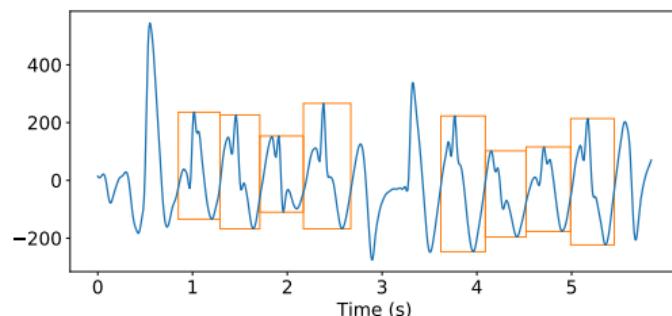
### Principle

- ▶ Objective of SPN : output boxes with largest Intersection over Union (IoU)
- ▶ IoU:  $b_j$  are labelled boxes,  $\hat{b}$  is an estimated box:

$$\text{IoU}(\hat{b}) \doteq \max_j \frac{|\mathbf{b}_j \cap \hat{b}|}{|\mathbf{b}_j \cup \hat{b}|}$$



(a) Step signal



(b) Step labels over a walk signal

## Step proposal network

### Training

- ▶ Output: a matrix  $\mathbf{W} \in \mathbb{R}^{T \times K}$ 
  - ▶  $T$ : signal length
  - ▶  $K$ : number of different box sizes
- ▶  $\mathbf{W}_{t,k}$ : probability that the box  $b_t^k$  starting at time  $t$  and of size 0.4s, 0.5s, or 0.6s (for respectively  $k = 1, 2$ , or 3) has a large IoU score
- ▶ Positive boxes:  $\text{IoU}(b_t^k) > \sqrt{0.7}$
- ▶ Negative boxes:  $\text{IoU}(b_t^k) < \sqrt{0.3}$
- ▶ Other are not used for training

The loss function  $\mathcal{L}$  over a signal  $\mathbf{s}$  is defined as:

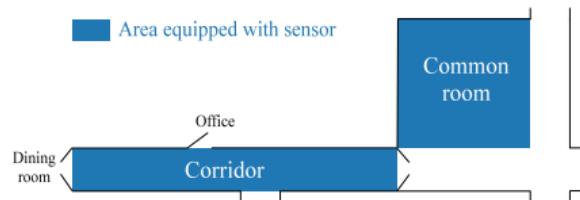
$$\mathcal{L}(\mathbf{s}, \mathbf{W}) = \sum_t \sum_{k \in [1, 2, 3]} \mathbb{1}_{\text{IoU}(b_t^k) > \sqrt{0.7}} \log(\mathbf{W}_{t,k}) + \mathbb{1}_{\text{IoU}(b_t^k) < \sqrt{0.3}} \log(1 - \mathbf{W}_{t,k}).$$

## Step proposal network

Results

### Data

- ▶ 43 signals recorded in a nursing home
- ▶ Manually labeled steps

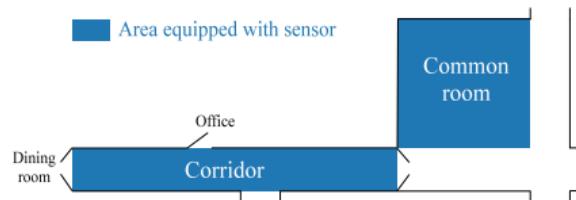


## Step proposal network

Results

### Data

- ▶ 43 signals recorded in a nursing home
- ▶ Manually labeled steps



### Training

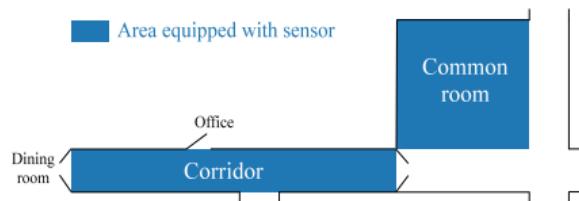
- ▶ SPN is trained using classical gradient descent
- ▶ Training time: < 5 minutes
- ▶ Inference (detection over a 10s signal): < 1 second
- ▶ Optimization details
  - ▶ learning rate of  $10^{-3}$
  - ▶ learning rate decay ( $\times 0.9$  every 10 epochs)
  - ▶ Nesterov momentum

## Step proposal network

Results

### Data

- ▶ 43 signals recorded in a nursing home
- ▶ Manually labeled steps



### Results

- ▶ Object detection use the mean Average Precision (mAP): area under the Precision-Recall curve
- ▶ **Without** embedding, mAP = 72,5%
- ▶ **With** embedding, mAP = 78,6%

### Training

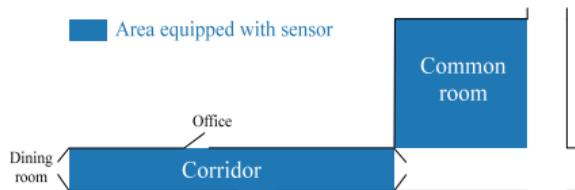
- ▶ SPN is trained using classical gradient descent
- ▶ Training time: < 5 minutes
- ▶ Inference (detection over a 10s signal): < 1 second
- ▶ Optimization details
  - ▶ learning rate of  $10^{-3}$
  - ▶ learning rate decay ( $\times 0.9$  every 10 epochs)
  - ▶ Nesterov momentum

## Step proposal network

### Results

#### Data

- ▶ 43 signals recorded in a nursing home
- ▶ Manually labeled steps

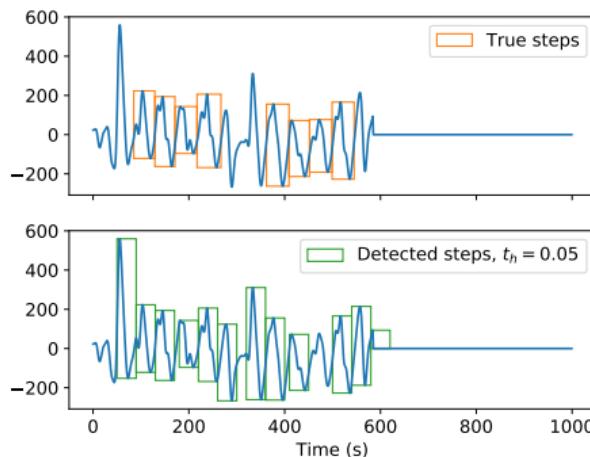


#### Training

- ▶ SPN is trained using classical gradient descent
- ▶ Training time: < 5 minutes
- ▶ Inference (detection over a 10s signal): < 1 second
- ▶ Optimization details
  - ▶ learning rate of  $10^{-3}$
  - ▶ learning rate decay ( $\times 0.9$  every 10 epochs)
  - ▶ Nesterov momentum

#### Results

- ▶ Object detection use the mean Average Precision (mAP): area under the Precision-Recall curve
- ▶ **Without** embedding, mAP = 72,5%
- ▶ **With** embedding, mAP = 78,6%

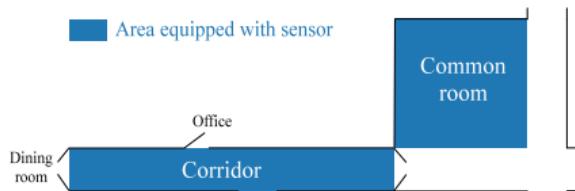


## Step proposal network

### Results

#### Data

- ▶ 43 signals recorded in a nursing home
- ▶ Manually labeled steps

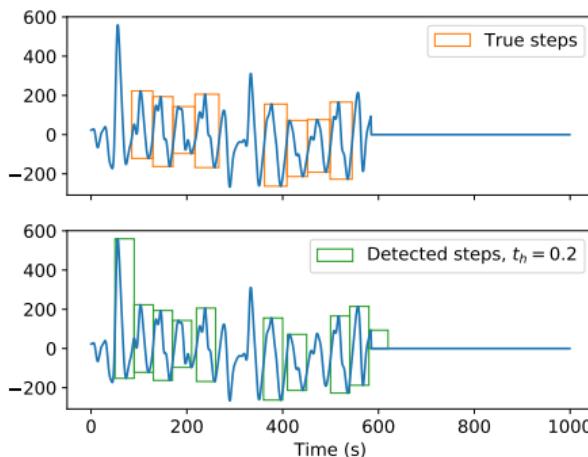


#### Training

- ▶ SPN is trained using classical gradient descent
- ▶ Training time: < 5 minutes
- ▶ Inference (detection over a 10s signal): < 1 second
- ▶ Optimization details
  - ▶ learning rate of  $10^{-3}$
  - ▶ learning rate decay ( $\times 0.9$  every 10 epochs)
  - ▶ Nesterov momentum

#### Results

- ▶ Object detection use the mean Average Precision (mAP): area under the Precision-Recall curve
- ▶ **Without** embedding, mAP = 72,5%
- ▶ **With** embedding, mAP = 78,6%

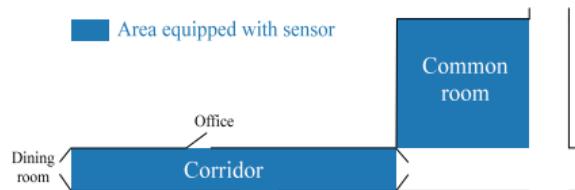


## Step proposal network

### Results

#### Data

- ▶ 43 signals recorded in a nursing home
- ▶ Manually labeled steps

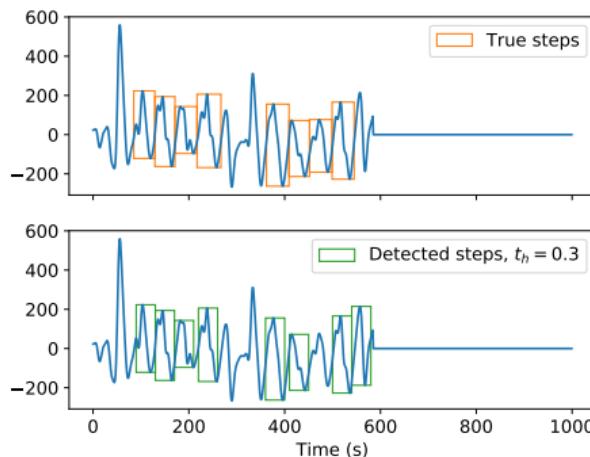


#### Training

- ▶ SPN is trained using classical gradient descent
- ▶ Training time: < 5 minutes
- ▶ Inference (detection over a 10s signal): < 1 second
- ▶ Optimization details
  - ▶ learning rate of  $10^{-3}$
  - ▶ learning rate decay ( $\times 0.9$  every 10 epochs)
  - ▶ Nesterov momentum

#### Results

- ▶ Object detection use the mean Average Precision (mAP): area under the Precision-Recall curve
- ▶ **Without** embedding, mAP = 72,5%
- ▶ **With** embedding, mAP = 78,6%



## Classification

### Training

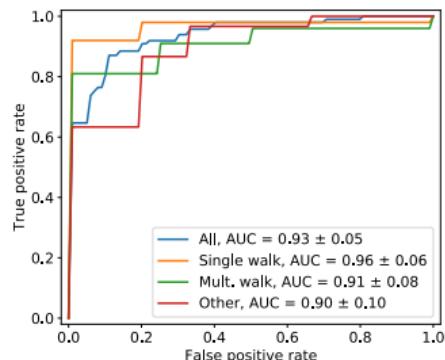
$$\mathcal{L}(\mathbf{s}_i, \hat{y}_i) = -(\mathbb{1}_{y_i=1} \log(\hat{y}_i) + \mathbb{1}_{y_i=0} \log(1 - \hat{y}_i))$$

Sublabel	Staff	Elderly
Single walk	43	31
Multiple walks	19	11
Other	19	23

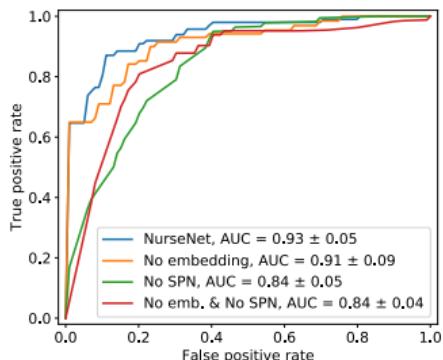
- ▶ SPN is trained using classical gradient descent
- ▶ Training time: < 5 minutes
- ▶ Inference (detection over a 10s signal): < 1 second
- ▶ Optimization details
  - ▶ learning rate of  $10^{-3}$
  - ▶ learning rate decay ( $\times 0.9$  every 10 epochs)
  - ▶ Nesterov momentum

## Classification

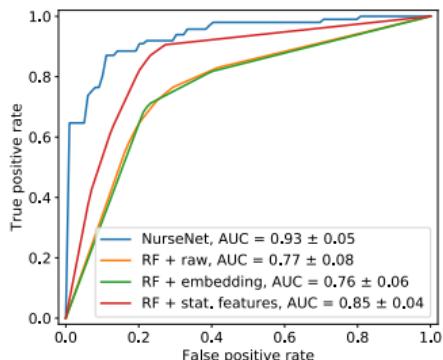
### Results



(a) NURSENET results



(b) NURSENET ablation analysis



(c) NURSENET and several RF-based models

## Conclusion

## Conclusion

### Contributions

- ▶ A simple and practical model for fall detection
- ▶ Transfer procedure for decision tree adapted to class imbalance
- ▶ A model to distinguish elderly vs. other with high accuracy

### Future work

- ▶ Explore meta models for transfer procedures
- ▶ What if features change ? Consider heterogeneous transfer
- ▶ Improve step detection precision (with regression layer ?)
- ▶ Test on benchmarks step data sets
- ▶ Is it possible to distinguish activities or individuals ?

## Publications and communications

### Publications

- ▶ L. Minvielle, M. Atiq, R. Serra, M. Mougeot, and N. Vayatis. [Fall detection using smart floor sensor and supervised learning](#). In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3445–3448, July 2017
- ▶ L. Minvielle, M. Atiq, S. Peignier, and M. Mougeot. [Transfer learning on decision tree with class imbalance](#). In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1003–1010, Nov 2019
- ▶ Ludovic Minvielle and Julien Audiffren. [Nursesnet: Monitoring elderly levels of activity with a piezoelectric floor](#). *Sensors*, 19(18), 2019
- ▶ P. Humbert, B. Le Bars, L. Minvielle, and N. Vayatis. [Robust Kernel Density Estimation with Median-of-Means principle](#). *Submitted to Neural Information Processing Systems 2020 (NeurIPS)*, 2020

### Communications

- ▶ Transfer learning on decision tree with imbalanced data, 3<sup>rd</sup> Summer school on transfer learning, 2019, Passau, Germany
- ▶ Step detection with proposal network. 4<sup>th</sup> French-German Summer School on Artificial Intelligence, 2020, Online.

## References

- [1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [2] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984.
- [3] Hilton Bristow, Anders Eriksson, and Simon Lucey. Fast convolutional sparse coding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.
- [4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.
- [5] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):814–830, 2016.
- [6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [7] N. Segev, M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv. Learn on source, refine on target: A model transfer learning framework with random forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1811–1824, Sep. 2017.