

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

Soutenance de thèse

Ludovic Minvielle

Directeur: Nicolas Vayatis, Centre Borelli, ENS Paris-Saclay
Co-encadrante: Mathilde Mougeot, Centre Borelli, ENS Paris-Saclay

Thèse industrielle entre l'ENS Paris-Saclay et Tarkett

Mercredi 15 Juillet 2020

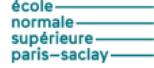


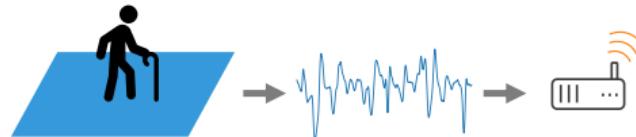
Table of contents

1. Introduction
2. A tour of monitoring systems
3. A fall detection system
4. Transfer learning
5. Step proposal network
6. Conclusion

Introduction

Context

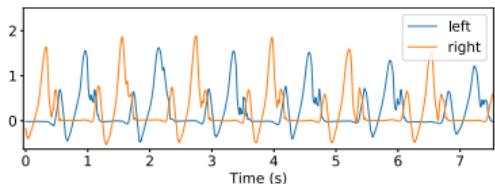
- ▶ Elderly population is growing
- ▶ Higher levels of frailty globally
- ▶ Increasing demand for reliable monitoring devices
- ▶ Tarkett, French company with 12,500 employees, 13 industrial sites, sells 1.3 millions m² of flooring every day
- ▶ *Floor in Motion*: a floor-based sensor for elderly care
- ▶ **Objective:** providing tools for elderly monitoring in nursing homes
 - ▶ First aimed application: fall detection



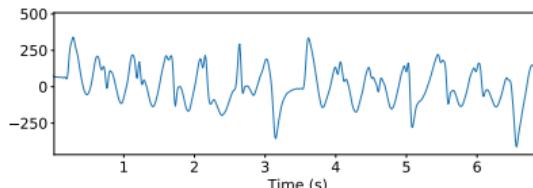
Introduction

Motivation

- ▶ Processing and understanding time series
 - ▶ Proliferation of sensor-based systems
 - ▶ Redundancy, interpretability, external perturbations
- ▶ Real world application
 - ▶ Real-time processing in a limited system
 - ▶ Convenient hypotheses not granted



Foot-attached accelerometer



Tarkett's floor sensor

A tour of monitoring systems

Systems

What makes a good monitoring system ?

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

Criteria

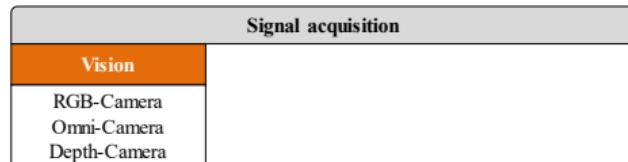
Coverage/Occlusion
Intrusiveness
Signal quality / info
Robustness
Ease of instal. / use
Scalability

A tour of monitoring systems

Systems

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	★☆☆	★☆☆

A tour of monitoring systems

Systems

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	Accelerometer
Omni-Camera	Gyroscope
Depth-Camera	Barometric pressure

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

A tour of monitoring systems

Systems

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

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

A tour of monitoring systems

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 processed signals ?

Time series classification

1. Series as *sequences*
 - ▶ Distance-based methods
2. Series as *feature vectors*
 - ▶ Computing several measures over a fixed size
 - ▶ Classification models (Anomaly detection, classical supervised models...)

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



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



Decision rule	
Threshold	Machine learning
	kNN SVM HMM Decision Tree

A tour of monitoring systems

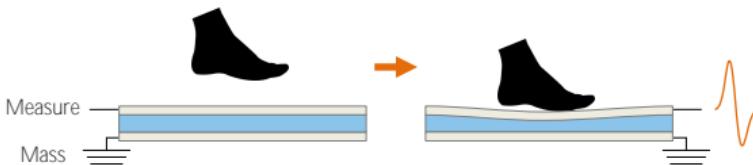
Tarkett sensor

- ▶ Piezoelectric principle:

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

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

When stressed or squeezed, the material emits charges.



A tour of monitoring systems

Tarkett sensor

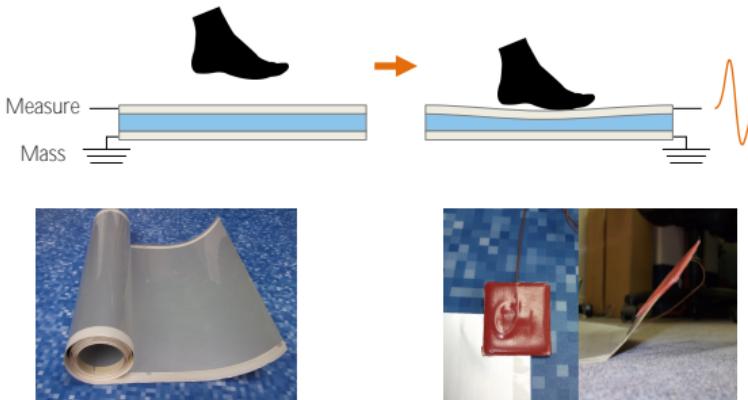
- ▶ Piezoelectric principle:

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

(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



A tour of monitoring systems

Tarkett sensor

- ▶ Piezoelectric principle:

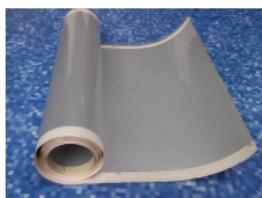
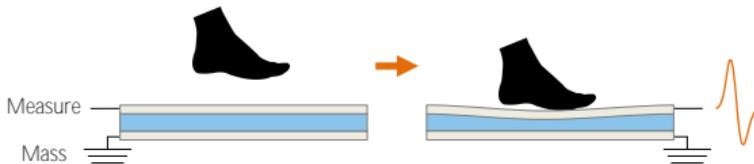
$$d = \frac{Q}{F},$$

(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

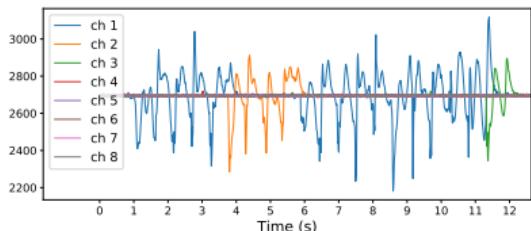


Fall detection

Data

Preprocessing

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

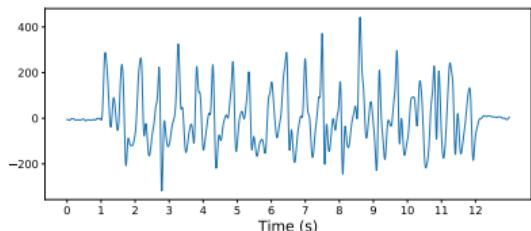


Fall detection

Data

Preprocessing

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

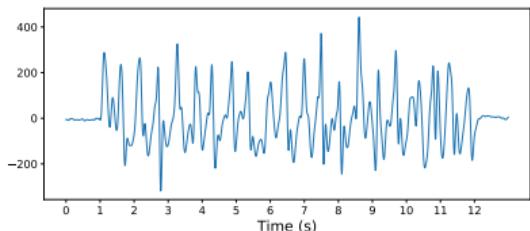


Fall detection

Data

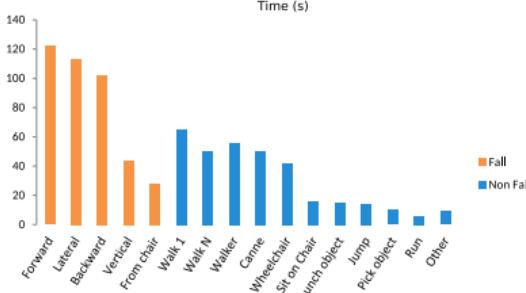
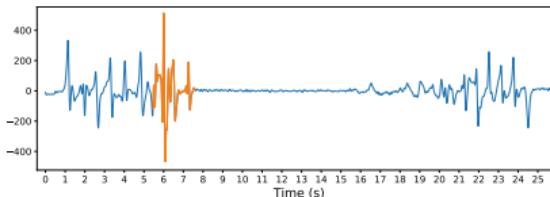
Preprocessing

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



Experimental dataset

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

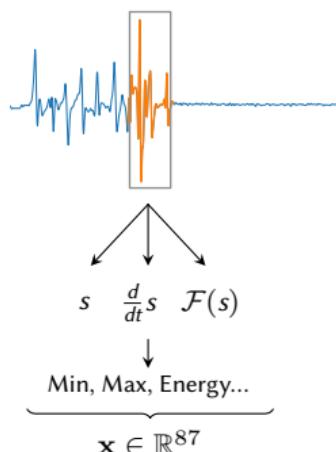


Fall detection

Method

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

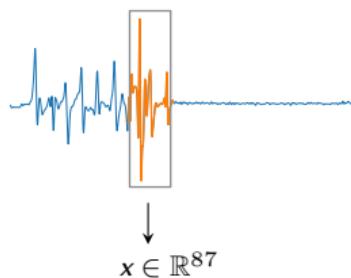


Fall detection

Method

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

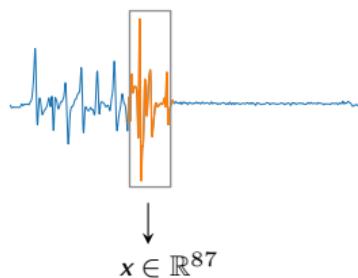


Fall detection

Method

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 [Breiman, 2001], based on **decision trees**

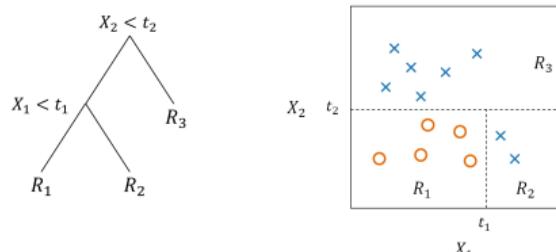
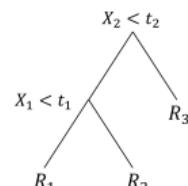
Decision tree

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 [Breiman et al., 1984] that solve:

$$\arg \min_{X_q, \tau} \text{IG} ,$$

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

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



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

Fall detection

Method

Random forest

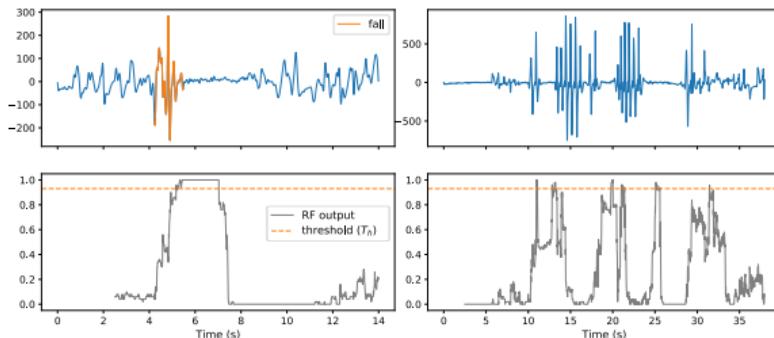
Decision trees d_1, \dots, d_{N_T} grown with two rules:

- ▶ 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 prediction function is then

$$f(x) = \arg \max_k f_k(x),$$

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



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

Fall detection

Method

Random forest

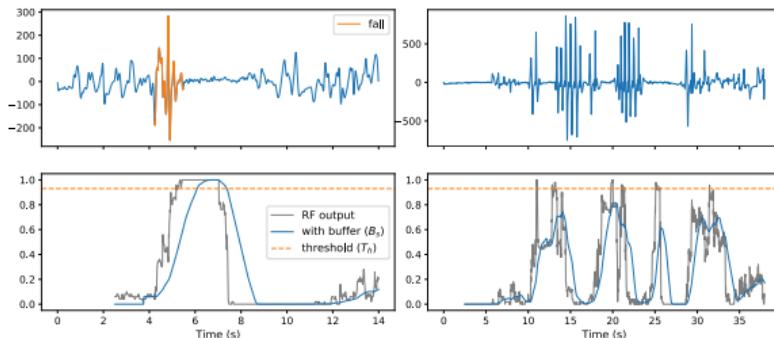
Decision trees d_1, \dots, d_{N_T} grown with two rules:

- ▶ 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 prediction function is then

$$f(x) = \arg \max_k f_k(x),$$

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



Time aggregation

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

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

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

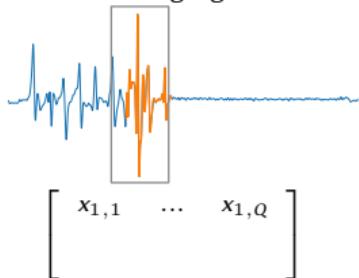
New binary classification function: $d(t) = \begin{cases} 1, & \text{if } g(t) > T_h \\ 0, & \text{otherwise} \end{cases}$

Fall detection

Method

Data augmentation

Select r windows in training signals.

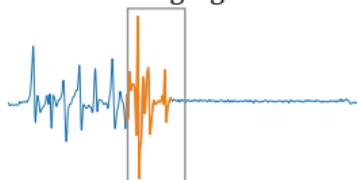


Fall detection

Method

Data augmentation

Select r windows in training signals.



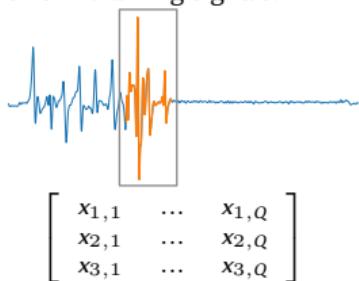
$$\begin{bmatrix} x_{1,1} & \dots & x_{1,Q} \\ x_{2,1} & \dots & x_{2,Q} \end{bmatrix}$$

Fall detection

Method

Data augmentation

Select r windows in training signals.

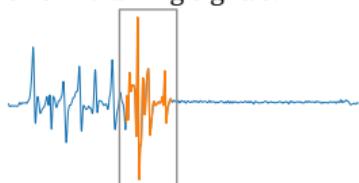


Fall detection

Method

Data augmentation

Select r windows in training signals.



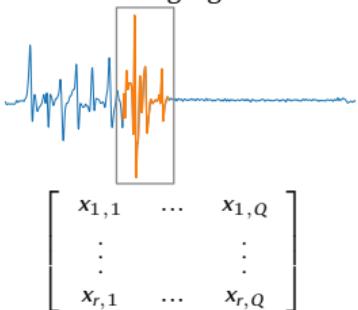
$$\begin{bmatrix} x_{1,1} & \dots & x_{1,Q} \\ \vdots & & \vdots \\ x_{r,1} & \dots & x_{r,Q} \end{bmatrix}$$

Fall detection

Method

Data augmentation

Select r windows in training signals.

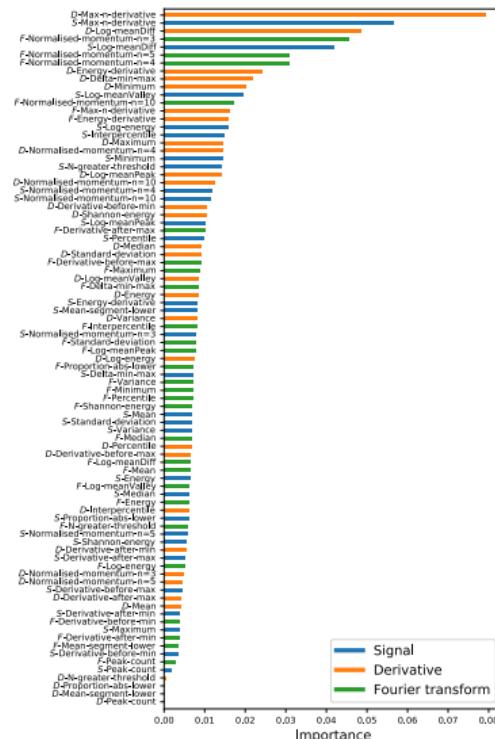


Feature reduction

Feature importance

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

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

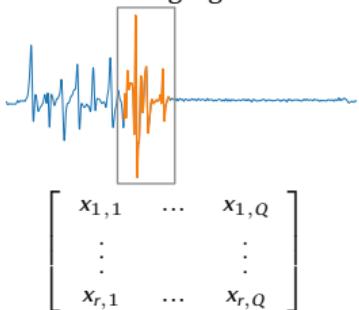


Fall detection

Method

Data augmentation

Select r windows in training signals.



Feature reduction

Feature importance

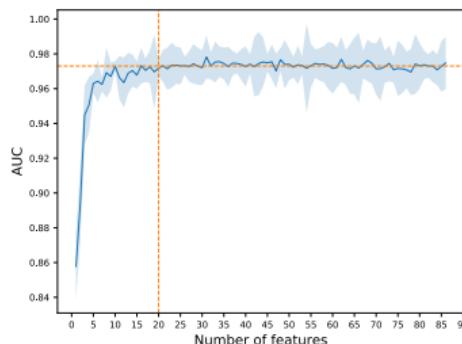
$$\text{Tree: } I(X_q) = \sum_{\text{nodes } t} p(t) \Delta i(t) \mathbb{1}(v(t) = X_q)$$

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

Recursive feature elimination

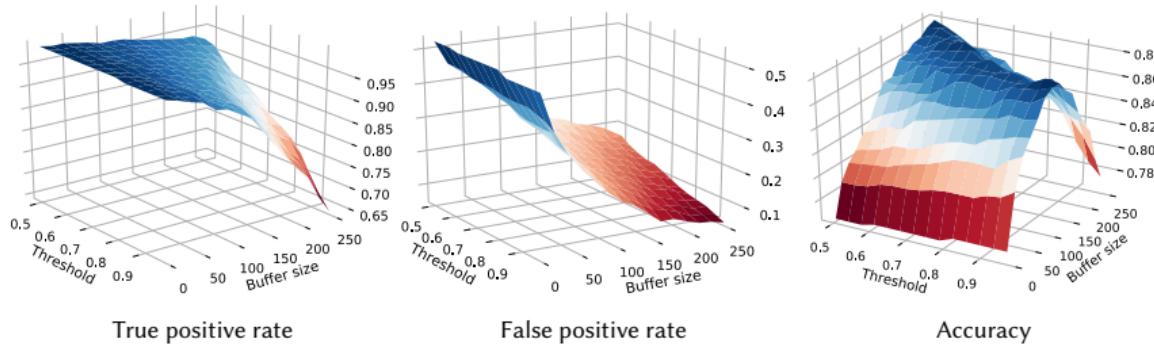
Initial pool of Q features X_1, \dots, X_Q .

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



Fall detection

Results



True positive rate

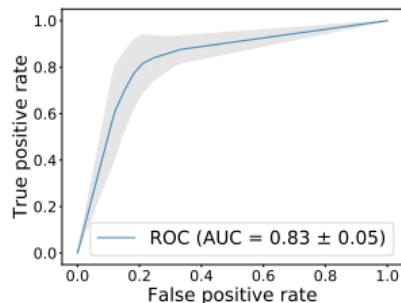
False positive rate

Accuracy

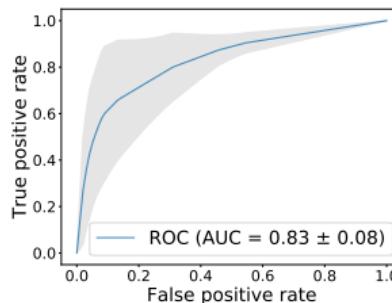
Model	Accuracy	TPR	FPR	$\text{TPR}_{\text{min}}^{\text{FPR} < 10}$	$\text{TPR}_{\text{max}}^{\text{FPR} < 10}$
$r = 5, Q = 20$					
LR	86.8 ± 1.5	90.5 ± 2.4	17.7 ± 4.9	67.0 ± 10.8	80.4 ± 6.4
LDA	85.5 ± 1.2	91.0 ± 2.1	21.7 ± 3.7	56.9 ± 7.0	78.7 ± 3.8
k-NN	87.0 ± 1.9	89.2 ± 1.4	16.0 ± 4.7	63.1 ± 4.2	83.1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

Transfer learning on decision tree

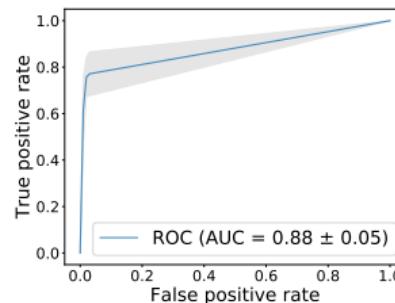
- ▶ Source domain: $\mathcal{D}_S = \{\mathcal{X}_S, P(X_S)\}$
 - ▶ Target domain: $\mathcal{D}_T = \{\mathcal{X}_T, P(X_T)\}$
 - ▶ Source task: $\mathcal{T}_S = \{\mathcal{Y}_S, P(Y_S|X_S)\}$
 - ▶ Target task: $\mathcal{T}_T = \{\mathcal{Y}_T, P(Y_T|X_T)\}$
- ▶ $\mathcal{X}_S = \mathcal{X}_T$
 - ▶ $P(X_S) \neq P(X_T)$
 - ▶ $\mathcal{Y}_S = \mathcal{Y}_T$
 - ▶ $P(Y_S|X_S) \neq P(Y_T|X_T)$



Source tested on source



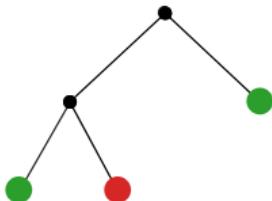
Source tested on target



Target tested on target

Transfer learning on decision tree

Structure Expansion / Reduction (SER)



Structure Transfer

1. Pruning
2. Threshold update

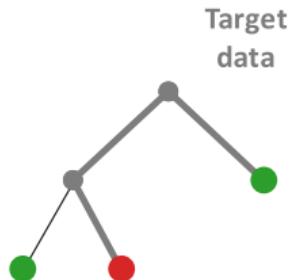
Drifts

1. Expansion
2. Reduction

Partition refinement or simplification

Transfer learning on decision tree

Structure Expansion / Reduction (SER)



1. Expansion
2. Reduction

Partition refinement or simplification

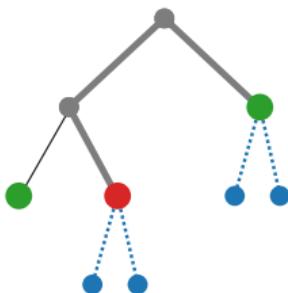
Structure Transfer

1. Pruning
2. Threshold update

Drifts

Transfer learning on decision tree

Structure Expansion / Reduction (SER)



1. Expansion
2. Reduction

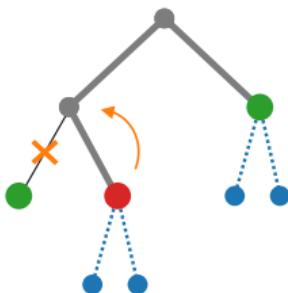
Partition refinement or simplification

Structure Transfer

1. Pruning
 2. Threshold update
- Drifts

Transfer learning on decision tree

Structure Expansion / Reduction (SER)



1. Expansion
2. Reduction

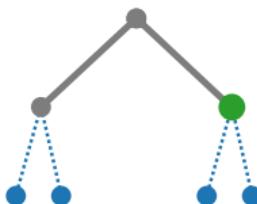
Partition refinement or simplification

Structure Transfer

1. Pruning
 2. Threshold update
- Drifts

Transfer learning on decision tree

Structure Expansion / Reduction (SER)



Structure Transfer

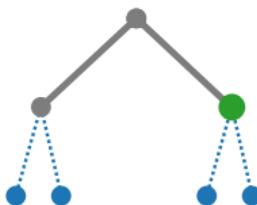
1. Pruning
 2. Threshold update
- Drifts

1. Expansion
2. Reduction

Partition refinement or simplification

Transfer learning on decision tree

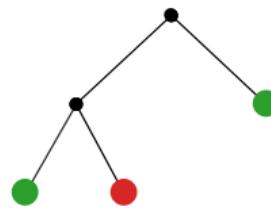
Structure Expansion / Reduction (SER)



1. Expansion
2. Reduction

Partition refinement or simplification

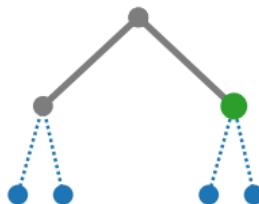
Structure Transfer



1. Pruning
 2. Threshold update
- Drifts

Transfer learning on decision tree

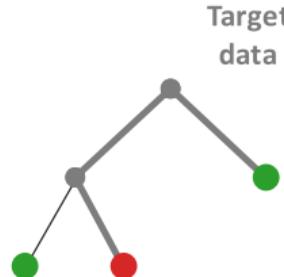
Structure Expansion / Reduction (SER)



1. Expansion
2. Reduction

Partition refinement or simplification

Structure Transfer

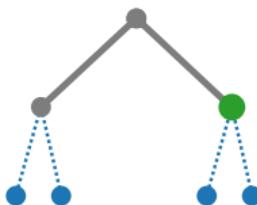


1. Pruning
2. Threshold update

Drifts

Transfer learning on decision tree

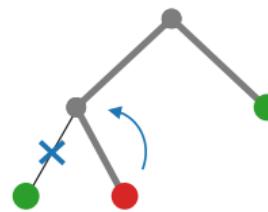
Structure Expansion / Reduction (SER)



1. Expansion
2. Reduction

Partition refinement or simplification

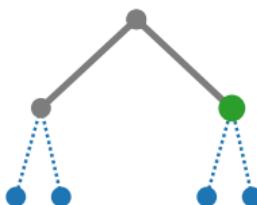
Structure Transfer



1. Pruning
 2. Threshold update
- Drifts

Transfer learning on decision tree

Structure Expansion / Reduction (SER)



1. Expansion
2. Reduction

Partition refinement or simplification

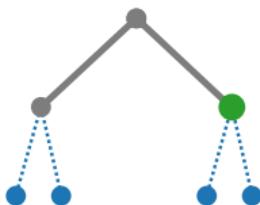
Structure Transfer



1. Pruning
 2. Threshold update
- Drifts

Transfer learning on decision tree

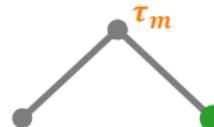
Structure Expansion / Reduction (SER)



1. Expansion
2. Reduction

Partition refinement or simplification

Structure Transfer



1. Pruning
 2. Threshold update
- Drifts

Transfer learning on decision tree

Leaf loss risk

$$p^T(x/y) = p^S(x/y) \quad (1)$$

$$p^T(y/x) = \lambda_y \frac{p^S(y/x)}{\int \lambda_y p^S(y/x) dy} \quad (2)$$

$$\text{with } \lambda_y = \frac{p_y^T}{p_y^S}$$

Leaf l that conserves the minority class k_{min} after Target update

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

Risk of losing the leaf:

$$R_{n_{k_{min}}}(l) = p^T(x \notin l/y = k_{min})^{n_{k_{min}}}$$

Transfer learning on decision tree

Leaf loss risk

$$p^T(x/y) = p^S(x/y) \quad (1)$$

$$p^T(y/x) = \lambda_y \frac{p^S(y/x)}{\int \lambda_y p^S(y/x) dy} \quad (2)$$

with $\lambda_y = \frac{p_y^T}{p_y^S}$

Leaf l that conserves the minority class k_{min} after Target update

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

Risk of losing the leaf:

$$R_{n_{k_{min}}}(l) = p^T(x \notin l/y = k_{min})^{n_{k_{min}}}$$

In homogeneous class imbalance:

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

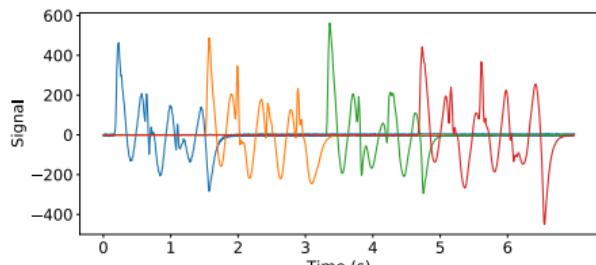
$$R_{n_{k_{min}}}(l) = p^S(x \notin l/y = k_{min})^{n_{k_{min}}}$$

Introduction

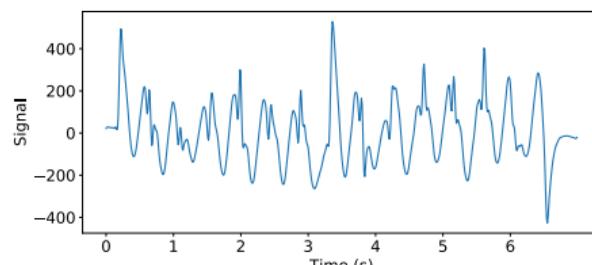
- ▶ **Issue:** one-dimensional signals for large areas
- ▶ **Goal:** Classify elderly from other individuals
 - ▶ Most signals are made of walks of staff individuals
- ▶ **Subtask:** Bring the model's attention over step-related signals
- ▶ A model to recognize steps ?

Introduction

- ▶ **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 model to recognize steps ?



(a) Raw signal



(b) Preprocessed signal

Figure: Healthy individual walking on the sensor.

- ▶ Signals are complex
- ▶ How to **localize** steps ?
- ▶ **This presentation:** A step detector using convolutional neural network: Step Proposal Network

Region proposal network

Object detection

- ▶ Classification: What is the image class ?



Figure: Classification vs Object detection. Source: [Girshick et al., 2014], [Ren et al., 2015]

Region proposal network

Object detection

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

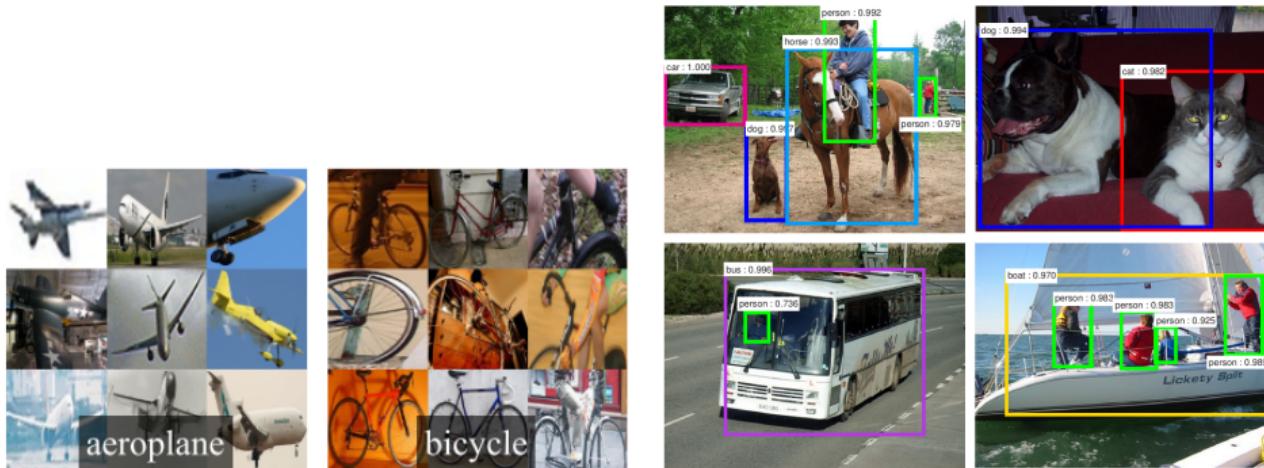


Figure: Classification vs Object detection. Source: [Girshick et al., 2014], [Ren et al., 2015]

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., 2016]
- ▶ Faster R-CNN [Ren et al., 2015]

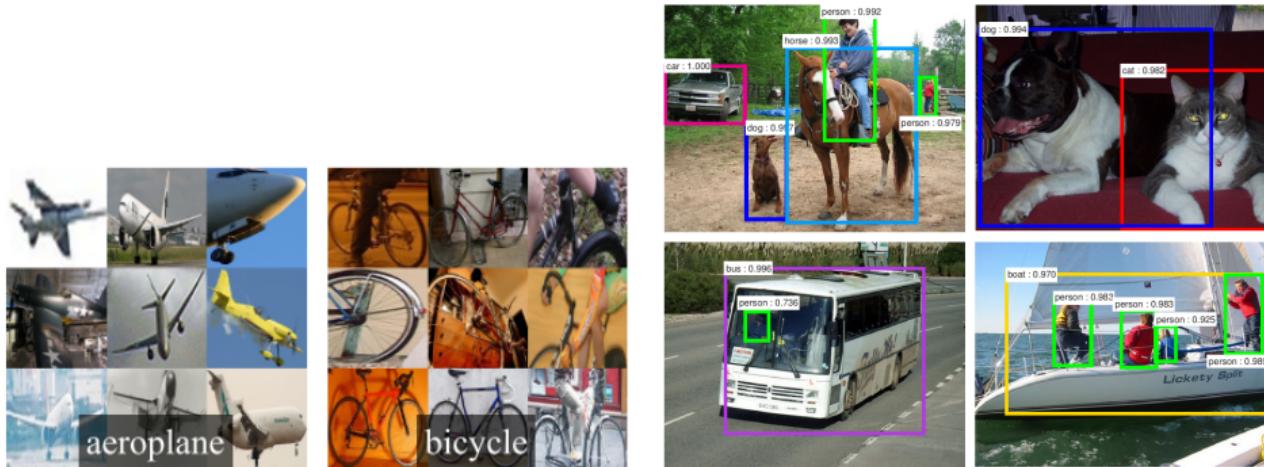


Figure: Classification vs Object detection. Source: [Girshick et al., 2014], [Ren et al., 2015]

Region proposal network

Faster R-CNN

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

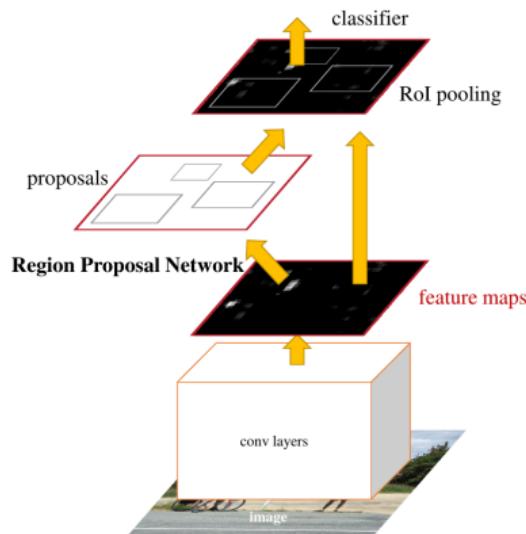


Figure: Region proposal network. Source: [Ren et al., 2015]

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)

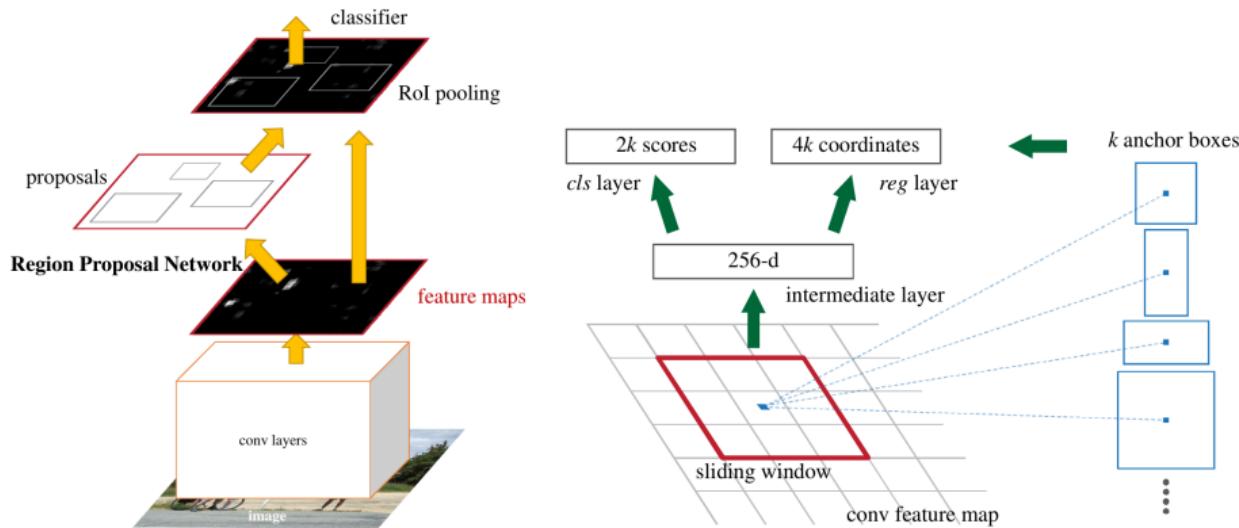


Figure: Region proposal network. Source: [Ren et al., 2015]

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

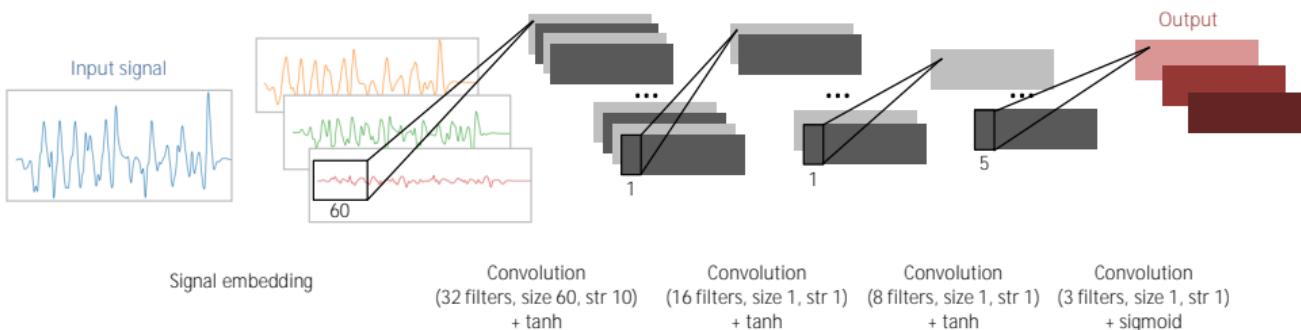


Figure: Architecture of SPN .

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

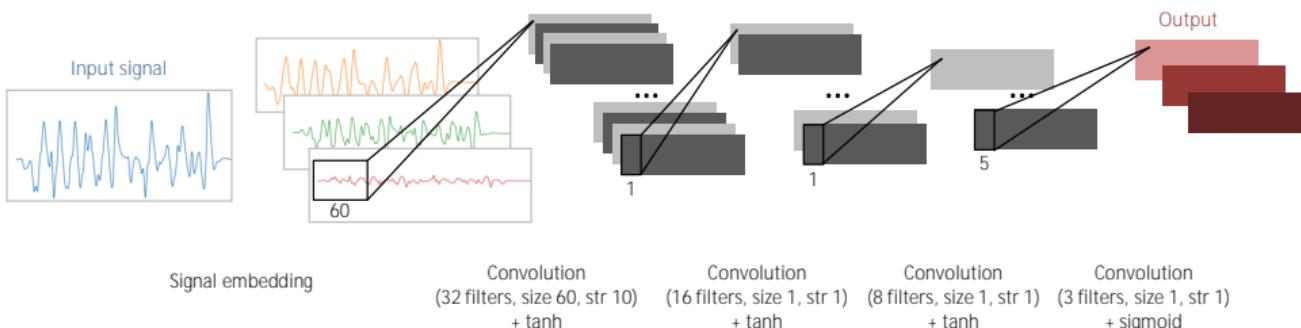


Figure: Architecture of SPN .

- ▶ Use the convolutional representation to “boost” training
- ▶ First layer (Signal embedding) of SPN 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

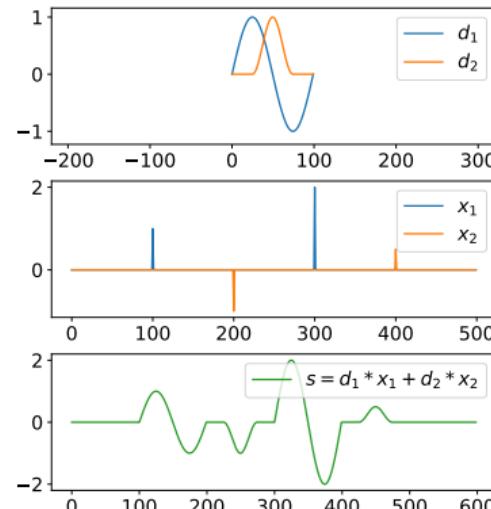


Figure: Convolutional dictionary learning.

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

CDL general problem:

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

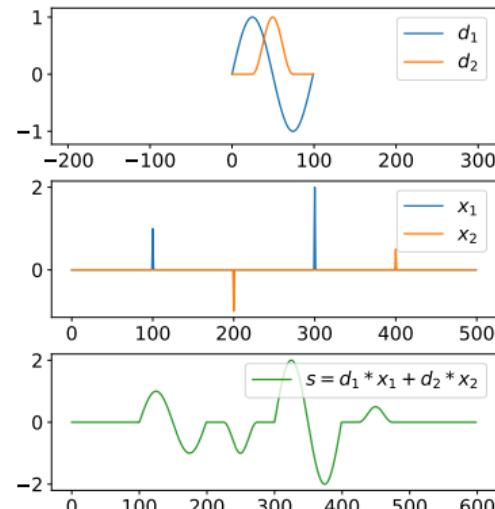


Figure: Convolutional dictionary learning.

Signal embedding

Learning step atoms

- ▶ Learning with Alternating Direction Method of Multipliers (ADMM) [Bristow et al., 2013]
- ▶ 3 atoms of length 0.7 second

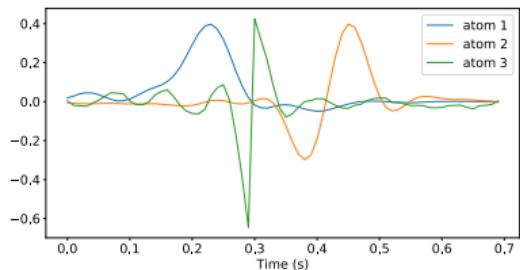
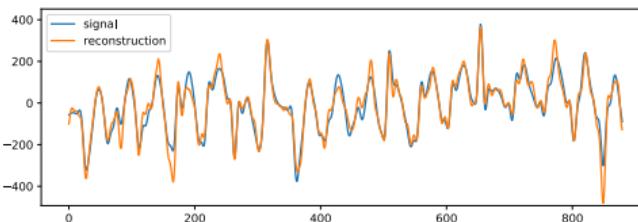
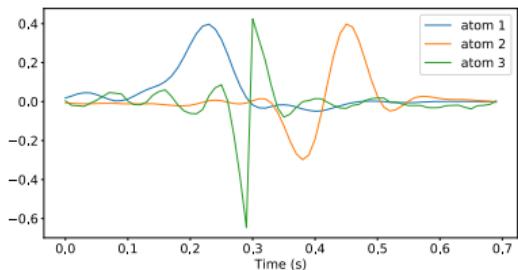


Figure: Dictionary.

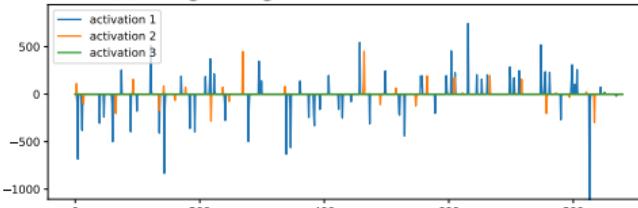
Signal embedding

Learning step atoms

- ▶ Learning with Alternating Direction Method of Multipliers (ADMM) [Bristow et al., 2013]
- ▶ 3 atoms of length 0.7 second



(a) Original signal and its reconstruction



(b) Signal activations

Figure: Dictionary.

Signal embedding

Learning step atoms

- ▶ Learning with Alternating Direction Method of Multipliers (ADMM) [Bristow et al., 2013]
- ▶ 3 atoms of length 0.7 second
- ▶ Use the following embedding:

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

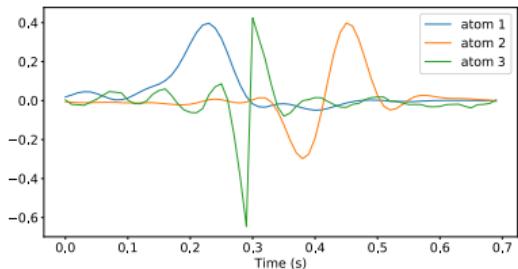
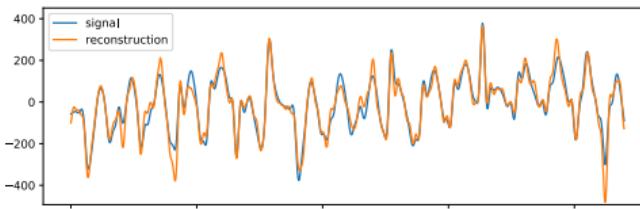
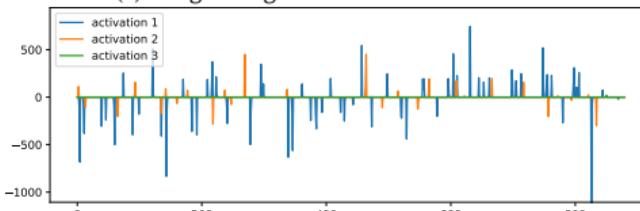


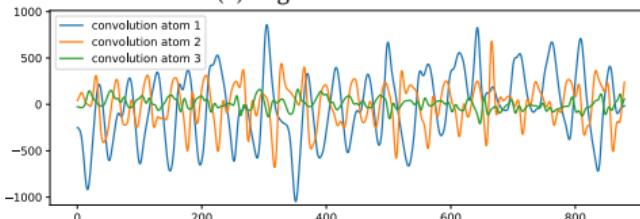
Figure: Dictionary.



(a) Original signal and its reconstruction



(b) Signal activations



(c) Embedding

Signal embedding

Learning step atoms

- ▶ Learning with Alternating Direction Method of Multipliers (ADMM) [Bristow et al., 2013]
- ▶ 3 atoms of length 0.7 second
- ▶ Use the following embedding:

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

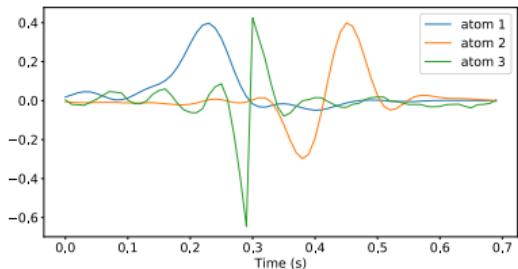
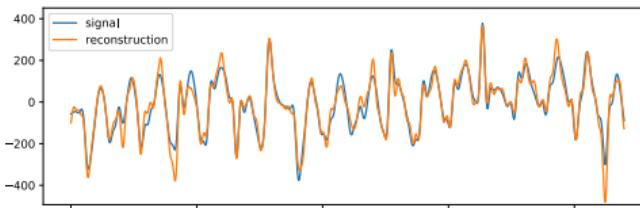
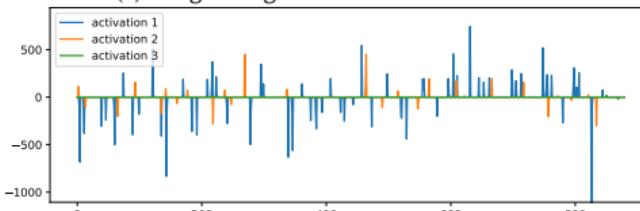


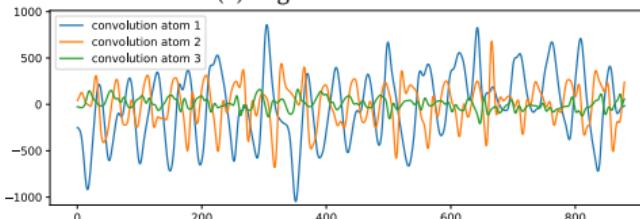
Figure: Dictionary.



(a) Original signal and its reconstruction



(b) Signal activations



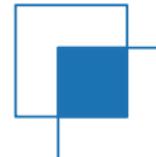
(c) Embedding

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:

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

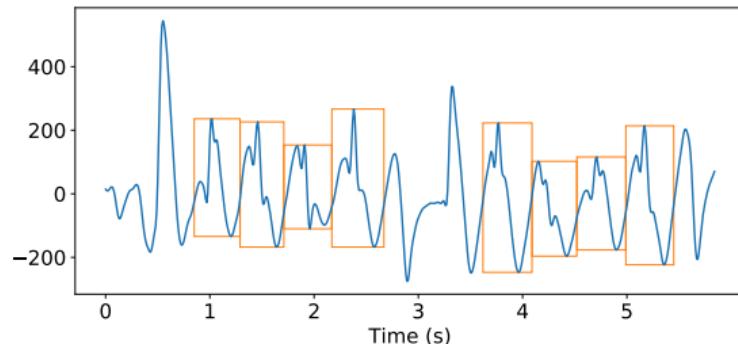
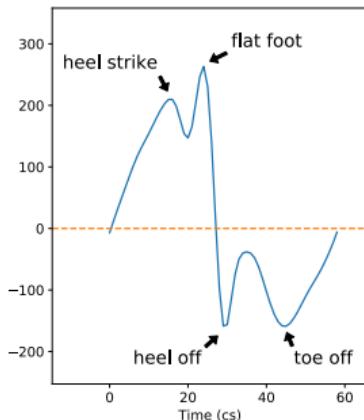
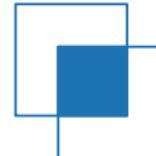


Step proposal network

Principle

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

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



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:

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

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

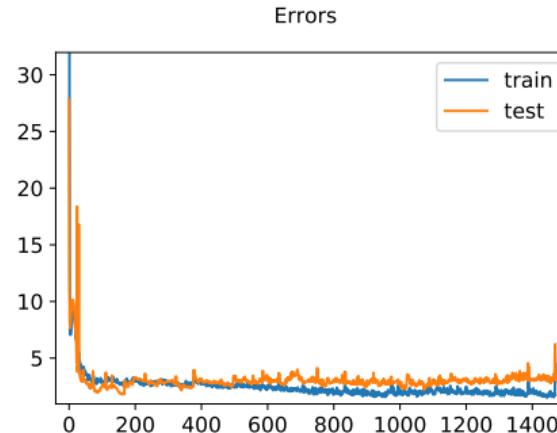


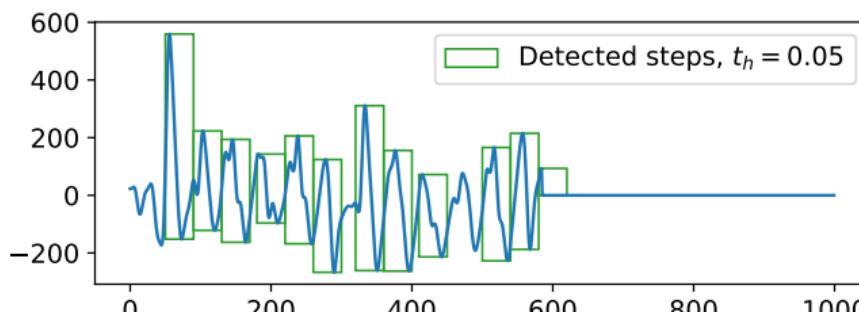
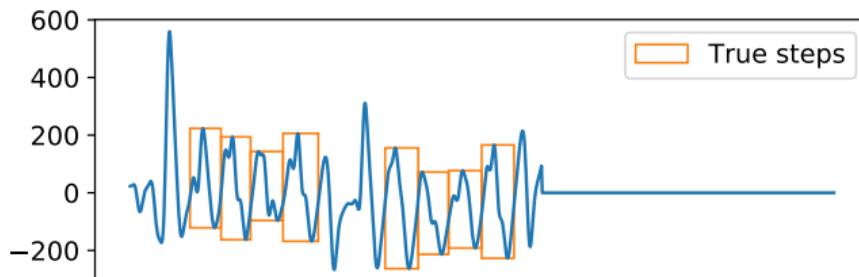
Figure: SPN training and testing errors.

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%

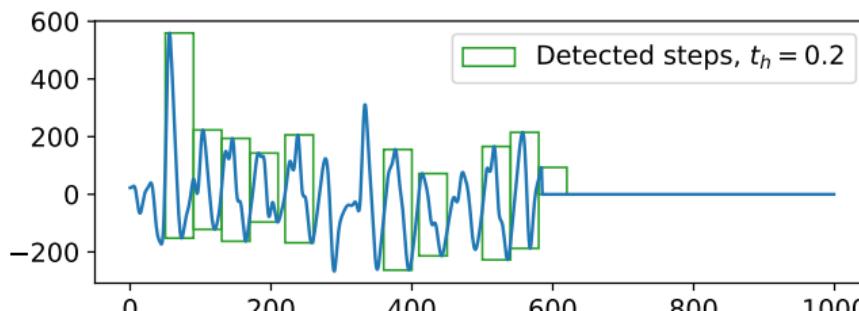
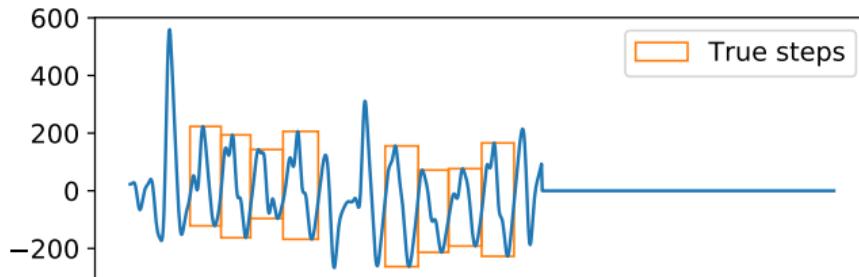
Results

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



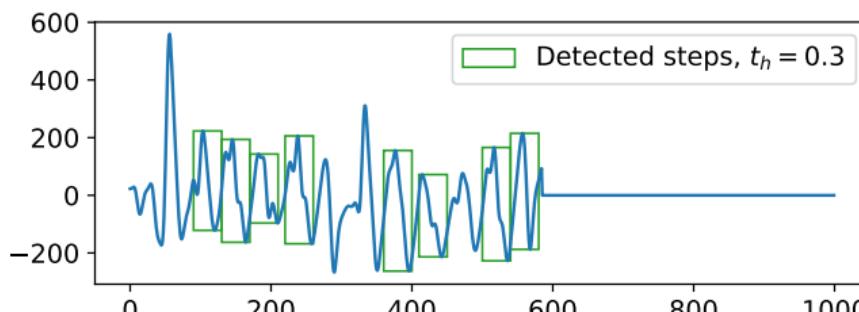
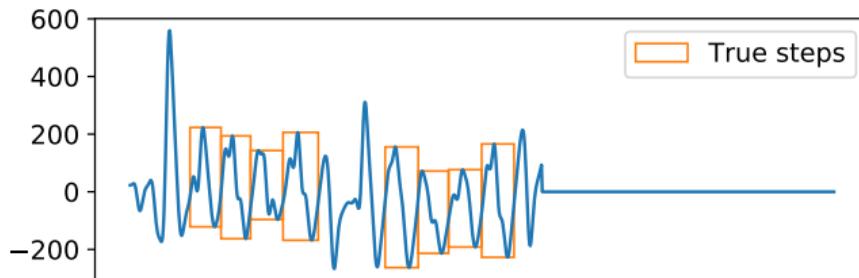
Results

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



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%



Conclusion

Conclusion

- ▶ SPN uses the convolutional representation to detect steps
- ▶ Allows to located steps in complex signals
- ▶ Training and inference are fast

Future work

- ▶ Add a regression layer on the step proposals
- ▶ Tests on step detection benchmarks data sets

Conclusion

Conclusion

- ▶ SPN uses the convolutional representation to detect steps
- ▶ Allows to located steps in complex signals
- ▶ Training and inference are fast

Future work

- ▶ Add a regression layer on the step proposals
- ▶ Tests on step detection benchmarks data sets

Thanks

Contact: minvielle@cmla.ens-cachan.fr

Reference: L. Minvielle and J. Audiffren. Nursenet : Monitoring elderly levels of activity with a piezoelectric floor. Sensors, 19(18), 2019 [link](#)

References

-  Breiman, L. (2001).
Random forests.
Machine Learning, 45(1):5–32.
-  Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984).
Classification and regression trees.
Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software.
-  Bristow, H., Eriksson, A., and Lucey, S. (2013).
Fast convolutional sparse coding.
In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
-  Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014).
Rich feature hierarchies for accurate object detection and semantic segmentation.
In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587.
-  Hosang, J., Benenson, R., Dollár, P., and Schiele, B. (2016).
What makes for effective detection proposals?
IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(4):814–830.
-  Ren, S., He, K., Girshick, R., and Sun, J. (2015).
Faster r-cnn: Towards real-time object detection with region proposal networks.
In *Advances in neural information processing systems*, pages 91–99.