Classification d'événements à partir de capteurs sols - Application au suivi de personnes fragiles.

Ludovic Minvielle

université
PARIS-SACLAY

Classification d'événements à partir de capteurs sol – application au suivi de personnes fragiles

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°574 : mathématiques Hadamard (EDMH)
Spécialité de doctorat: Mathématiques appliquées
Unité de recherche : Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, 94235, Cachan, France.
Référent : ENS Paris-Saclay

Thèse présentée et soutenue à Paris, le 15 juillet 2020, par

Ludovic MINVIELLE

Composition du Jury

Bernadette DORIZZI Directrice de recherche, Télécom SudParis	Présidente
François CHARPILLET Directeur de recherche, INRIA Nancy	Rapporteur & Examinateur
Amaury HABRARD Professeur, Université Jean Monnet de Saint-Étienne	Rapporteur & Examinateur
Miguel COLOM Directeur de recherche associé, ENS Paris-Saclay	Examinateur
Nicolas Vayatis Professeur, ENS Paris-Saclay	Directeur de thèse
Mathilde MOUGEOT Professeur, ENS Paris-Saclay	Co-directrice de thèse
Renan SERRA	Invité
Richard PERES Directeur de l'innovation, Tarkett	Invité

Thèse de doctorat

NNT : 2020UPASN023

Contents

Remerciements

Je remercie bien évidemment mon directeur de thèse Nicolas Vayatis et ma co-directrice Mathilde Mougeot pour leur confiance, leur optimisme et leur soutien durant ces années, dans les moments difficiles comme dans les autres. Je remercie également Richard Peres d'avoir permis que cette thèse se réalise et soit soutenue par Tarkett, ainsi que Renan Serra pour son accompagnement et son inaltérable motivation.

Je tiens à remercier tous les membres du jury d'avoir accepté de lire mon manuscrit et d'assister à ma soutenance.

Je remercie également ma cohorte de relecteur·ice·s: Camille, Jean-Marie et Rémy. Vos retours m'ont beaucoup aidé (si, si!).

Merci aussi aux chercheurs vétérants du Centre Borelli: Cédric, Charles, Thomas, Juan, Sergio et Julien, pour les collaborations, le soutien, les discussions et les bons conseils même si je n'ai pas toujours suivis ces derniers.

Merci évidemment aux copains et copines rencontré·e·s au labo, pour l'entraide, les rires, les soirées, la bienveillance. Mention spéciale à la brillante équipe des Saints-Pères: Thomas, Pierre, Batiste et Marie. Avec vous, le chemin a été plus facile. À celles et ceux qui n'ont pas encore fini leur thèse, je souhaite du courage et de la réussite, même si ce n'est probablement pas nécessaire.

Je remercie tout particulièrement mon partenaire de thèse industrielle, l'incroyable et inoxydable Mounir, avec qui j'ai partagé une grande partie de cette aventure et sans qui j'aurais eu certainement beaucoup moins de choses à raconter.

Puisque la vie est plus agréable en musique, je n'oublie pas les ami·e·s de divers ensembles plus ou moins musicaux, plus ou moins cuivrés, plus ou moins rock, vous vous reconnaîtrez: Odyssound, La Farigoule, FBTF et Tucky.

Merci à mes soeurs, mes parents, et Nanou pour leur soutien et leur affection.

Je remercie enfin ma première relectrice, Valentine. Merci pour tes remarques constructives et ton soutien indéfectible, notamment pendant la rédaction en terre landaise.

Abstract

Current data and forecasts show that the global population is ageing. As this elderly population is prone to frailty and potentially fatal falls, the growing demand for robust monitoring systems has led to a vast research upon the topic. This thesis addresses the subject of event detection in temporal signals for elderly monitoring by the use of an innovative pressure sensor installed directly under the flooring. The objective of this work is to present contributions that aim at building a reliable monitoring system in the context of an industrial application, which brings several challenges.

As many proposed systems are looking for the highest performances, pragmatic criteria are often overlooked. We first show that most systems do not meet main practical issues and that floor systems constitute promising candidates for monitoring tasks.

Since complex signals require sophisticated models, we propose a random-forest-based approach that detects falls with state-of-the-art accuracy and meets hardware constraints with a feature selection procedure. The model performance is improved with data augmentation and time aggregation of the random forest outputs.

Then, we address the issue of confronting our model to the real world with transfer learning methods that act on the core model of random forests, i.e. decision trees. These methods are adaptations of seminal work and are designed to tackle the class imbalance problem. Methods are tested on several data sets, showing interesting potential continuation, and a Python implementation is made available for reproducibility.

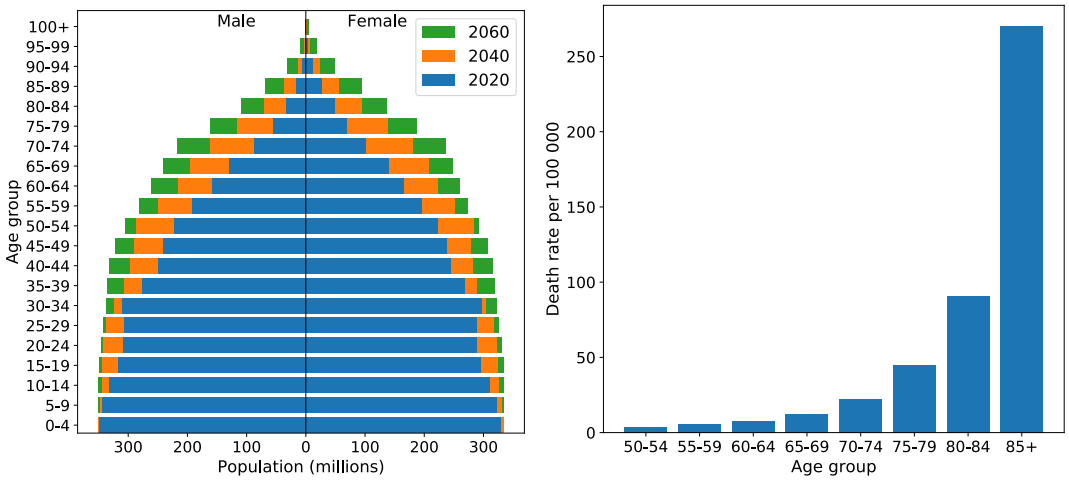
Finally, motivated by the issue of elderly monitoring while dealing with a single one-dimensional signal for a large area, we propose to distinguish elderly persons from younger individuals with a classification model that uses a neural network and convolutional dictionary learning. Since signals are mainly made of walks and that labelled data is scarce, we develop custom training procedures so that the first part of the model focuses attention on walk signals, and the last part of the model is trained with all previous layers frozen. This novel approach to gait classification allows to isolate elderly-generated signals with very high accuracy.

Résumé (en français)

Contents

1 Contexte de la thèse

Cette thèse de doctorat s'inscrit dans le cadre de la détection d'événement pour des systèmes de suivi de personnes. Ce travail a été réalisé dans le cadre du dispositif CIFRE (Convention Individuelle de Formation par la Recherche) de l'ANRT (Agence Nationale de la Recherche et de la Technologie) et a été sponsorisé par l'entreprise Tarkett, qui est une entreprise française multinationale dont l'activité principale est la production de revêtements de sol. En quelques mots, Tarkett regroupe dans le monde 12 500 salariés, 33 sites industriels et vend environ 1,3 million de mètres carrés de revêtement par jour, et ce pour divers secteurs comme les hôpitaux, écoles, logements, hôtels, bureaux, magasins et complexes sportifs. Tarkett a lancé en 2015 un projet innovant appelé Floor In Motion, dont le but premier est de concevoir un système de suivi de personnes âgées basé sur un capteur au sol. Ce système est conçu en premier lieu pour être installé dans des maisons de retraites, avec un potentiel développement vers l'utilisation dans les foyers particuliers. Avant de proposer du suivi général d'activité, la première application visée fut la détection de chute. Dans ce contexte, l'objectif principal de cette thèse est d'étudier et de proposer des solutions de détection d'événements pour le suivi de personnes âgées à partir de signaux temporels issus de capteurs au sol.



(a) Pyramide des âges mondiale

(b) Taux de mortalité dû à la chute aux États-Unis en 2018

FIGURE 0.1 : Prévisions de l'âge de la population mondial (a) et taux de mortalité des chutes selon l'âge aux États-Unis. Sources : (a) United Nations, Department of Economic and Social Affairs, Population Division [182], (b) Centers for Disease Control and Prevention, Injury Center, Injury Prevention & Control [36].

2 Motivations

2.1 Le suivi de personnes fragiles

Fragilité et chutes dans une population vieillissante. Selon l'Organisation Mondiale de la Santé, la part de population mondiale dont l'âge dépasse 60 ans est celle qui grossit le plus vite [196]. Si l'on se penche sur les prévisions (Figure 0.1-a), on constate que la pyramide des âges voit sa forme actuellement triangulaire évoluer vers une forme cloche, montrant ainsi l'augmentation significative de la population âgée dans le monde. Cette population vieillissante est sujette à la *fragilité*, une notion qui ses dernières années a reçu une attention accrue de la part de la communauté médicale [124]. La fragilité est généralement définie comme l'accumulation de signaux de détérioration significative de la santé chez les personnes âgées, et Fried et al. [60] la définissent plus précisément comme un syndrome clinique, dans lequel au moins trois critères sont présents parmi les suivants : la perte de poids involontaire, l'épuisement, la faible force de préhension, une vitesse de marche lente et une faible activité physique. Parmi d'autres facteurs, la fragilité contribue à augmenter le risque de chute. La fragilité augmentant avec l'âge, les personnes âgées deviennent alors particulièrement sujettes aux chutes – plus de 30% des personnes de plus de 60 ans chutent au moins une fois par an [196]. Bien que peu dangereuse pour une jeune personne en bonne santé, une chute peut en revanche entraîner de sérieuses blessures à une personne âgée, et peut parfois même être fatale. Les chiffres montrent en effet que le nombre de chutes fatales croît de manière exponentielle avec l'âge (Figure 0.1-b).

Conséquences de la chute. Les répercussions de la chute chez les personnes âgées sont multiples [36]. En premier lieu, une chute peut évidemment avoir des conséquences physiques directes, la plupart d'entre elles étant des fractures, le plus souvent du poignet

et de la hanche, ou des blessures à la tête, ces dernières pouvant entraîner des lésions cérébrales. Outre la douleur et l'inconfort, les blessures physiques réduisent la mobilité, parfois pendant longtemps ou même de façon irréversible, ce qui entraîne une plus grande dépendance.

En outre, une chute, même sans blessure physique grave, peut déclencher chez la personne une peur de tomber à nouveau. Cette peur entraîne alors une réduction des interactions et des activités sociales, ce qui augmente considérablement le niveau de fragilité, et donc les chances de tomber à nouveau. Le manque d'exercice est particulièrement préjudiciable, car il a été démontré que l'activité physique réduit la faiblesse musculaire, augmente la mobilité, améliore la santé neuronale, limite la fragilité et réduit le risque global de décès [117]. Enfin, les chutes ont également un impact économique. Ces répercussions, qu'elles soient directes (médicaments, rééducation…) ou indirectes (pertes de productivité de la société), augmentent dans le monde entier [196].

Le suivi de personne fragile. Un système de surveillance peut être défini comme un dispositif d'assistance dont l'objectif principal est de donner un aperçu de l'activité d'une personne à des fins de santé, soit sous forme de tâches générales telles que des indicateurs d'activité, soit sous forme de tâches plus spécifiques telles que les alertes en cas de situations dangereuses. Les systèmes de surveillance des soins aux personnes âgées peuvent servir deux objectifs principaux, à savoir estimer le niveau de fragilité et alerter lorsque la personne est en danger, notamment en cas de chute.

La détection précoce de la fragilité est considérée comme étant d'un intérêt majeur par la communauté médicale à des fins de prévention, et il a été démontré que les stratégies thérapeutiques ciblées réduisent considérablement les conséquences de la fragilité [185]. En outre, il a été démontré que la peur de tomber est susceptible d'être diminuée par la conscience d'être surveillé par un détecteur de chute robuste [33]. Pour ces raisons, la surveillance de l'activité physique des personnes âgées dans les maisons de retraite est essentielle, et d'autant plus car cela permet de concentrer les ressources sur les personnes particulièrement vulnérables à la fragilité et aux chutes.

L'autre élément clé des systèmes de surveillance est le temps de réaction de l'aide extérieure (soignants ou entourage) après une chute. Non seulement un détecteur de chute efficace peut empêcher que des blessures graves ne conduisent à la mort, mais il limite également le temps qu'une personne peut passer sur le sol après une chute. En effet, même non blessée, environ une personne âgée sur deux est incapable de se relever sans assistance [112]. Il a été démontré qu'une longue période au sol (définie par Lord et al. [112] comme plus d'une heure passée sur le sol après une chute) peut entraîner de graves complications telles que l'hypothermie ou la déshydratation, mais est également associée à une fragilité accrue puisqu'en effet, la moitié de ceux qui subissent une longue période au sol meurent dans les six mois qui suivent l'accident.

Mise en place d'un système de suivi fiable. Au cours des vingt dernières années, on constate un intérêt croissant pour le développement de systèmes intelligents permettant de suivre les personnes fragiles dans leur vie quotidienne, les systèmes de détection des chutes représentant à eux seuls une littérature vaste et variée. Dans ce contexte, Tarkett vise à proposer un système basé sur un capteur sol pouvant répondre à plusieurs contraintes pratiques lorsqu'il s'agit d'une utilisation en conditions réelles [161]. Ces contraintes com-

prennent notamment le caractère intrusif de ces systèmes de surveillance, qui est défini à la fois comme la capacité à être caché au patient et le respect de sa vie privée. Plusieurs autres critères peuvent être définis en ce qui concerne les utilisations à grande échelle et les aspects pratiques. Cela soulève une première question qui est de savoir comment situer le système de Tarkett dans cette offre pléthorique de systèmes de surveillance. La Figure 0.2 présente une illustration d'un système de surveillance au sol.

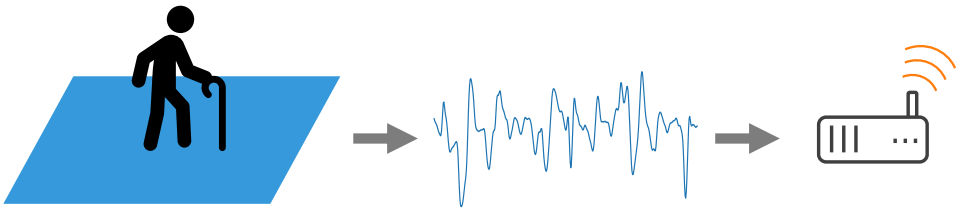
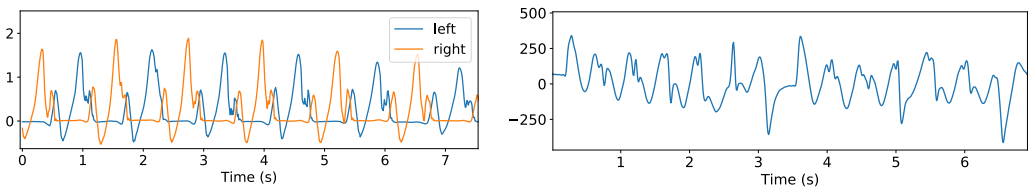


FIGURE 0.2 : Schéma d'un système de suivi de personne fragile basé sur un capteur sol.

2.2 Traitement des séries temporelles pour des applications terrain

Comprendre les séries temporelles. L'utilisation de systèmes de surveillance entraîne le traitement d'une grande quantité de signaux temporels. De manière générale, on constate ces dernières années une prolifération de systèmes basés sur des capteurs à des fins médicales, ce qui amène la problématique de savoir comment extraire le meilleur de ces données. Qu'ils soient unidimensionnels ou multivariés, ces signaux sont généralement temporels. Il peut s'agir par exemple d'électrocardiogrammes, d'électroencéphalogrammes, d'accélérations des parties du corps, d'enregistrements vidéo, de sons, etc.

Selon la tâche à accomplir, le traitement des signaux temporels peut soulever plusieurs enjeux. Tout d'abord, les signaux peuvent être très redondants ou de grande dimension. Par exemple, la détection d'événements à partir d'une vidéo peut être facile car les images contiennent beaucoup d'informations, mais se révèle être coûteuse en termes de calcul et exige donc certaines étapes de pré-traitement permettant de se ramener à des signaux utilisables pour les modèles de détection. Deuxièmement, les signaux peuvent être bien



(a) Accéléromètres attachés aux pieds (b) Capteur sol

FIGURE 0.3 : Exemples de signaux de marche issus de deux types de capteur. La figure (a) montre un signal de marche d'une personne en bonne santé, enregistrée avec deux accéléromètres attachés aux pieds. Chaque accéléromètre enregistre l'accélération et la vitesse angulaire sur quatre axes (X, Y, Z et l'axe vertical), ce qui donne une série temporelle de 16 dimensions. L'accélération affichée est celle le long de l'axe de la marche, pour chacun des pieds. La figure (b) contient une marche d'une membre du personnel soignant d'une maison de retraite, enregistrée avec le capteur sol.

adaptés à la tâche (par exemple, électroencéphalogramme pour la surveillance de l'anesthésie, accélération du corps pour l'analyse de la marche) ou beaucoup plus difficiles à interpréter. C'est le cas par exemple de l'utilisation des radars pour la surveillance des personnes, ou du son pour la détection des chutes. En conséquence, alors que certains modèles reposent sur de simples heuristiques pour accomplir leurs tâches prédictives, exécuter les mêmes tâches avec des modèles similaires et des signaux issus de capteurs différents peut échouer. Le système de Tarkett basé sur le sol est destiné à la surveillance des personnes, mais en raison de la nature du signal et de contraintes matérielles, il produit des signaux assez délicats à traiter. La Figure 0.3 présente un exemple illustrant les signaux de marche enregistrés par deux différents types de capteur : le capteur de pression au sol de Tarkett et un accéléromètre fixé au pied (le signal est tiré d'une base de données publique destinée à l'analyse de la marche [177]). Dans ce contexte, la question générale de la classification à partir de séries temporelles devient centrale pour notre problème.

Applications terrain. Un autre aspect important de notre cadre de travail est la confrontation de modèles étudiés en environnement contrôlé avec les applications sur le terrain. Cela peut s'exprimer de différentes manières.

Premièrement, de nombreuses applications exigent de travailler en temps réel ou au moins en temps court. C'est évidemment le cas des systèmes de détection des chutes mais aussi de toute tâche de surveillance ayant une valeur essentielle dans la détection précoce des événements, par exemple lors du contrôle de l'anesthésie pendant une intervention chirurgicale, ou qui nécessite une réponse rapide pour une utilisation pratique, ce qui est la cas de méthodes de diagnostic médical. Bien que de nombreuses applications puissent bénéficier de processeurs puissants capables de traiter facilement les tâches visées, d'autres sont intégrées dans des appareils qui imposent des limitations matérielles. C'est le cas par exemple des véhicules qui intègrent de plus en plus de capteurs et qui nécessitent l'exécution de tâches complexes (comme la conduite autonome) en temps réel et avec un matériel limité. Notre cas n'est pas si différent dans le sens où le système se présente comme un ensemble constitué du capteur et de son unité de traitement associée, ce qui entraîne des contraintes matérielles, tant en termes d'espace mémoire que de puissance de traitement.

Deuxièmement, la plupart des méthodes traditionnelles d'apprentissage machine tiennent généralement compte d'hypothèses commodes, la première étant que la procédure d'entraînement ait accès à un ensemble de données étiquetées suffisamment important, ce qui, dans les applications réelles, est déjà discutable. Lorsque des systèmes innovants sont conçus, la collecte de données est la plupart du temps coûteuse, que ce soit en temps ou en argent. Les données peut être fastidieuses à étiqueter pour plusieurs raisons. La difficulté d'étiquetage peut être due à la complexité des signaux à interpréter, certains requérant un regard expert (par exemple les données médicales), ou à la grande variété de signaux qui nécessitent d'être étiquetés afin d'englober la plus large diversité possible de signaux d'entrée (par exemple les signaux d'activité de la vie quotidienne pour la détection de chute). Cette limitation peut conduire à une petite quantité de données étiquetées, ou à une distribution déséquilibrée des classes, c'est-à-dire lorsqu'une classe est sous-représentée par rapport aux autres. Par conséquent, la rareté des données et le déséquilibre des classes peuvent devenir un problème majeur lorsque l'on souhaite mettre en place un modèle d'apprentissage fiable.

Enfin, pour pallier les problèmes de pénurie de données ou de coûts, les données peuvent

être collectées dans un environnement contrôlé ou même synthétisées, ce qui induit alors une dissemblance avec les données dites opérationnelles, à savoir les données que l'on vise à traiter avec le modèle final. Ce phénomène pouvant se produire dans un très large éventail de sujets, le problème général du traitement des différences entre l'environnement de création des modèles ou des données et la tâche finale souhaitée fait l'objet d'un intérêt croissant, et relève d'un domaine de l'apprentissage machine appelé *apprentissage par transfert*. Dans notre cadre, nous rencontrons une situation de transfert typique avec deux ensembles de données similaires, l'un étant construit dans un environnement contrôlé et l'autre étant collecté dans sur le terrain.

3 Contributions

3.1 Chapitre 2 : Suivi de personne : revue des systèmes et méthodes de détection

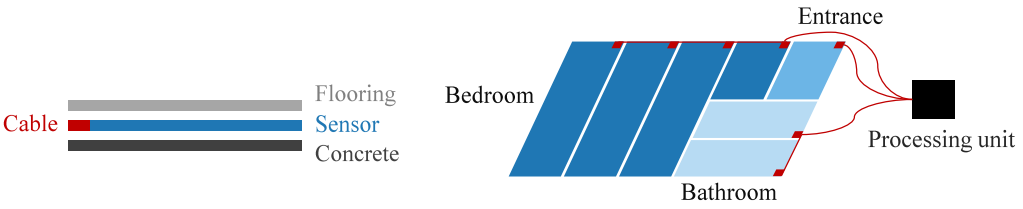
La première contribution de cette thèse consiste en un aperçu des systèmes de surveillance existants conçus pour la détection des chutes. Cet examen est mené tout au long de la chaîne de traitement, depuis la génération du signal jusqu'à la décision finale. Cette revue bibliographique permet d'abord de situer le système Tarkett parmi cette vaste gamme de systèmes et, à l'aide d'un ensemble de critères établis pour d'utilisation pratique de tels système, elle met en évidence les avantages des systèmes au sol. Une partie de cette étude est consacrée à la classification des séries temporelles, soulignant l'utilisation plus fréquente de méthodes de classification supervisée avancées là où les méthodes basées sur des mesures de distances ne sont pas assez efficaces.

3.2 Chapitre 3 : Capteur sol et détection de chute

Ce chapitre décrit plus précisément le capteur utilisé, et propose un modèle basé sur l'algorithme de la forêt aléatoire, et appris à l'aide d'un jeu de données acquis en environnement contrôlé.

Les capteurs sols peuvent se diviser en quatre familles : les capacitifs, qui se basent sur la modification d'un champ électrique par le corps humain ; les capteurs de contacts, qui sont des capteurs de pression délivrant des valeurs binaires sur des grilles à la résolution plus ou moins élevée ; les piézorésistifs, détectant les variations de pression par modification de la résistance électrique du matériau ; les piézoélectriques, qui émettent un courant lors qu'il subissent une pression. Pour plusieurs raisons pratiques (humidité, alimentation électrique, installation matérielle), le capteur Tarkett se base sur la technologie piézoélectrique, et s'installe comme des bandes de largeur fixe (60 cm) et de longueur variable, permettant ainsi de couvrir assez facilement différentes surfaces (voir Figure 0.4).

Le signal subit un pré-traitement simple consistant à filtrer et sommer sur tous les canaux d'entrée de l'unité de traitement, ce qui donne un signal uni-dimensionnel représentant toute la surface de surveillance. Afin de mettre en place un modèle de détection de chute, une base de donnée a été constituée en environnement contrôlé. Pour cela, un site pilote a été installé et des volontaires ont suivi un protocole permettant de générer des signaux de chutes et d'activité de la vie quotidienne. Cette base de donnée dite *expérimentale* nous permet ainsi d'entraîner un modèle de classification.

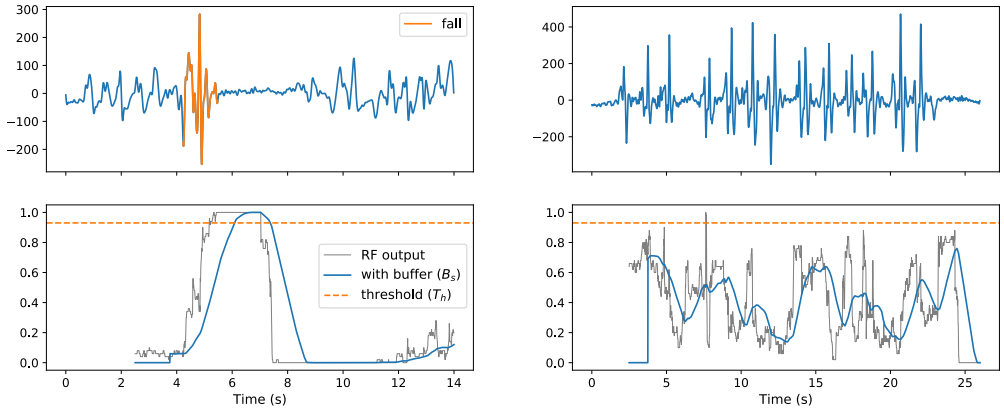


(a) Installation du capteur (b) Chambre équipée du sytème de détection

FIGURE 0.4 : Exemple schématique du capteur de pression en pratique. Le capteur est posé directement sur le béton et recouvert du sol (a). Dans cet exemple (b), il y a trois zones distinctes, dont la chambre qui dans ce cas est équipée de quatre bandes interconnectées, qui génèrent donc un seul signal pour cette zone. Ce signal est ici une des trois entrées qui alimentent l'unité de traitement locale.

Comme la plupart des modèles de classification traitent une entrée comme un vecteur de variables de longueur fixée, le signal n'est pas traité tel quel par le classifieur, mais est d'abord transformé en un vecteur de caractéristiques. Ces variables sont des mesures statistiques classiques inspirées de la littérature sur le traitement des séries temporelles, et calculées sur une fenêtre de taille fixe du signal, de sa dérivée, et de sa transformée de Fourier.

Le modèle étant destiné à fonctionner en temps réel, nous proposons une agrégation temporelle des sorties de la forêt aléatoire afin de réduire le nombre de fausses détections. Les sorties successives de la forêt aléatoire sont prises en compte par un tampon (*buffer*) de taille B_s et un seuil (*threshold*), noté T_h, est alors utilisé sur la sortie. Ainsi, les fausses alarmes susceptibles d'être provoquées par des événements courts et de grande énergie peuvent être écartées (voir Figure 0.5).



(a) Chute – *vrai positif* (b) Marche avec une canne – *vrai négatif*

FIGURE 0.5 : Exemples de détection avec et sans l'agrégation temporelle (caractérisée par le tampon B_s et le seuil T_h). Avec l'agrégation, une chute peut être toujours détectée (a) mais un signal pouvant déclencher une fausse alarme est alors écarté (b).

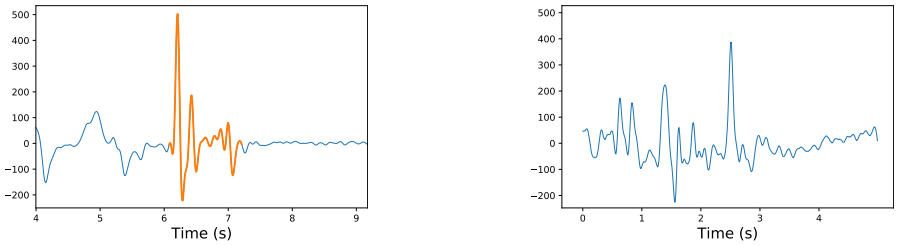
Afin d'augmenter les performances, une procédure d'augmentation des données est éga-

lement proposée et améliore considérablement la précision du modèle. Cette procédure se base sur une sélection aléatoire de fenêtres dans le signal initial. De plus, comme nous visons à produire un modèle destiné à une application embarquée et donc limitée en terme de capacité de calcul, nous étudions une méthode permettant de réduire le coût du calcul des caractéristiques tout en conservant la majeure partie du pouvoir discriminant. Cette méthode se base sur de la réduction récursive de variables et nous montrons que combiné à l'augmentation des données d'entraînement, le modèle donne alors de bons résultats.

3.3 Chapitre 4 : Apprentissage par transfert sur arbre de décision

Dans ce chapitre, nous proposons des méthodes d'apprentissage par transfert sur des arbres de décision, en prenant en compte le déséquilibre de classes.

Le premier modèle de détection des chutes proposé au chapitre précédent nous amène à explorer ses performances sur des données réelles, c'est-à-dire des données enregistrées dans des maisons de retraite, et nous nous interrogeons ainsi sur les pistes d'amélioration. En effet, dans le cadre classique de l'apprentissage, on considère que les données qui servent à l'apprentissage du modèle sont issues de la même distribution que les données qui seront considérées par la tâche finale. Or, cette hypothèse ne se vérifie pas toujours, qui plus est dans des domaines d'application industriels, où la récupération de données terrains étiquetées est complexe. Ce problème est plus généralement formulé par le cadre du l'apprentissage par transfert, où l'on désigne par $\mathcal{D}^S$ le domaine *source*, c'est à dire le domaine d'apprentissage du modèle, et par $\mathcal{D}^T$ le domaine cible (*target* en anglais), celui sur lequel on souhaite appliquer notre modèle.



(a) Chute simulée (b) Chute réelle

FIGURE 0.6 : Signaux de chute provenant de la base de données *expérimentale* (a) et de la base de données *opérationnelle* (b).

Nous souhaitons utiliser la connaissance de $\mathcal{D}^S$ où nous disposons d'un jeu de données fiable et équilibré – la classe *Chute* y est en effet autant représentée que la class *Non-chute*, et $\mathcal{D}^T$ où nous disposons d'un jeu de données à la fois de petite taille et déséquilibré, la chute étant un événement rare. Pour se faire, nous explorons les procédures d'apprentissage par transfert sur le classifieur qui constitue le modèle de base de la forêt aléatoire, à savoir l'arbre de décision. Notre travail se base sur des travaux antérieurs aux nôtres qui proposent deux méthodes de transfert sur les arbres de décision. Ces procédures agissent sur des arbres préalablement appris avec $\mathcal{D}^S$ en les modifiant à l'aide des données nouvellement étiquetées venant de $\mathcal{D}^T$.

Nous proposons plusieurs variantes en tentant de répondre au problème de déséquilibre des classes. Pour cela, nous nous plaçons dans le cadre d'un déséquilibre de classe ho-

mogène. Nous montrons qu'il existe un risque de perte de noeuds pour laquelle la classe minoritaire est toujours significative selon $\mathcal{D}^T$, et nous tentons soit de limiter l'élagage de ces noeuds, soit de prendre en compte ce déséquilibre homogène de classe dans les transformations des arbres. Les algorithmes proposés montrent des résultats prometteurs sur les données déséquilibrées et nous suggérons des améliorations pour une plus grande généralisation.

3.4 Chapitre 5 : Suivi de personne fragile avec réseaux de neurones

Le dernier chapitre explore les perspectives d'un suivi plus général de l'activité à l'aide du capteur sol. Rappelons que le système délivre un signal unidimensionnel pour toute une zone prédéfinie, par exemple une chambre, un couloir etc. Ainsi, le principal inconvénient de ce type de capteur est son incapacité à distinguer les utilisateurs. Nous proposons une méthode qui vise à faire la distinction entre les personnes âgées et les soignants ou les visiteurs. Pour ce faire, nous utilisons un réseau de neurones convolutifs qui contient deux parties.

Les premières couches du modèle sont entraînées à la reconnaissance de pas dans les signaux, en s'inspirant de la littérature portant sur la détection d'objets dans les images. Cette initiative est motivée par le fait que la plupart des signaux rencontrés sont des marches des aides soignants. Le caractère convolutif du modèle est renforcé par l'entraînement séparé de la première couche à l'aide d'un dictionnaire sous contrainte de parcimonie, et ce sur des signaux de marches, portant ainsi l'attention du modèle sur ce type de signaux. La deuxième partie du modèle est finalement entraînée sur la tâche de classification finale.

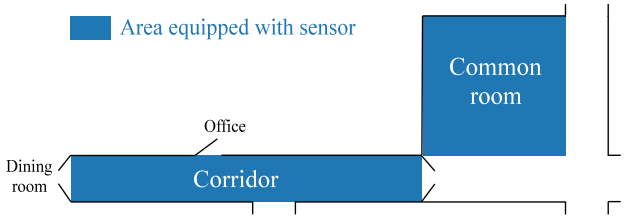


FIGURE 0.7 : Installation du système dans la maison de retraite.

Le jeu de données est constitué de signaux d'activités variées se déroulant dans un couloir et une salle commune d'un établissement partenaire (voir installation en Figure 0.7).

L'évaluation du modèle de détection des pas montre la bonne performance des réseaux de neurones convolutifs pour la détection d'objet et pourrait être considéré à plus long terme pour traiter des séries temporelles. Un exemple de détection est donné en Figure 0.8. Le modèle de classification final est comparé à un modèle de forêt aléatoire utilisant des caractéristiques décrites au 3, montrant ainsi l'intérêt d'une telle architecture. Nous analysons l'influence des différentes parties du modèle à l'aide des résultats obtenus par ablations successives de chaque partie, concluant ainsi sur la pertinence des apprentissages séparés. En outre, le modèle parvient aussi à reconnaître les signaux de personnes âgées sur des signaux plus complexes que de simples marches, ce qui ouvre la voie au développement de tâches plus complexes telles que la classification des activités ou éventuellement l'identification des individus.

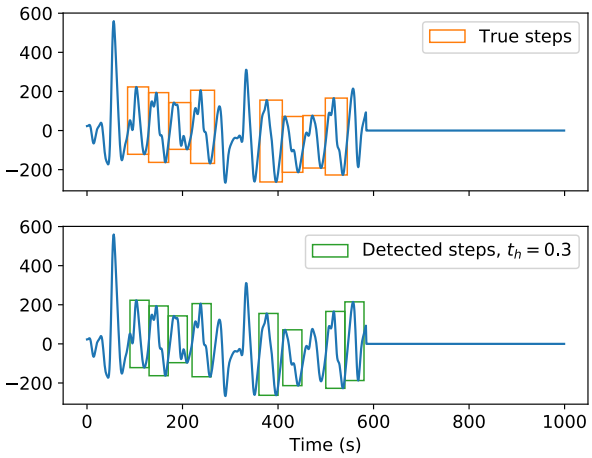


FIGURE 0.8 : Détection de pas à l'aide d'un réseau de neurones convolutifs.

4 Publications

Les chapitres 3, 4 et 5 de cette thèse ont chacun fait l'objet de d'une publication.

- 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
  https://ieeexplore.ieee.org/document/8037597
- 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
  https://ieeexplore.ieee.org/document/8995296
- L. Minvielle and J. Audiffren. Nursenet : Monitoring elderly levels of activity with a piezoelectric floor. *Sensors*, 19(18), 2019
  https://www.mdpi.com/1424-8220/19/18/3851

1 Introduction

Contents

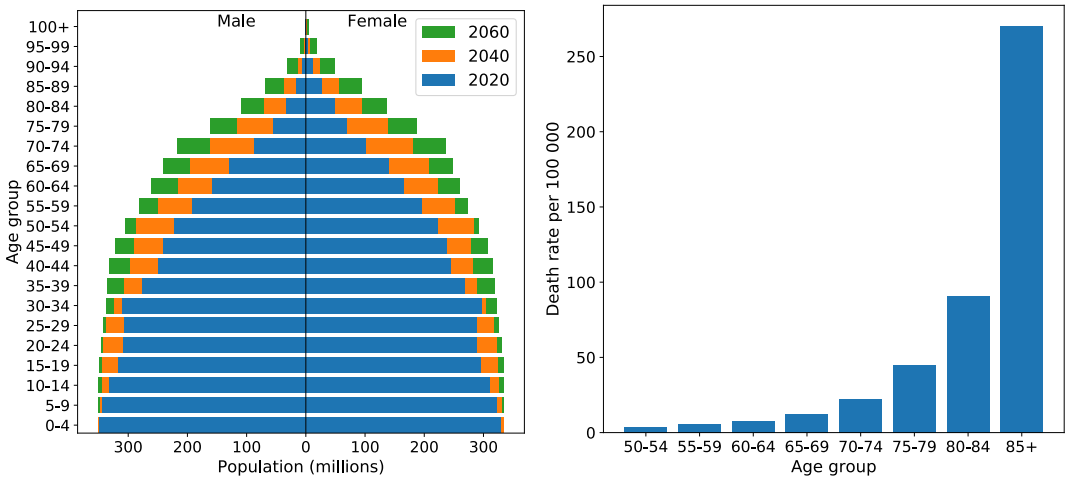
1 Context

This doctoral thesis falls within the framework of event detection for activity monitoring systems, and was carried out thanks to CIFRE (Convention Individuelle de Formation par la Recherche) and commissioned by the ANRT (Agence Nationale de la Recherche et de la Technologie). It was sponsored by Tarkett, which is a French multinational company whose main activity is the production of flooring. To give a glimpse at the company's activity, Tarkett has 12,500 employees, 33 industrial sites, and sells 1.3 million square meters of flooring every day, for hospitals, schools, housing, hotels, offices, stores and sports fields. The company launched in 2015 an innovative project named Floor In Motion, whose goal is to put up a floor-sensor-based monitoring system for elderly. This system is designed for nursing homes with a potential development for individual use at home, and its first main application was decided to be a fall detector as a first step towards general monitoring. In this context, the main objective of this thesis is to study and propose solutions for event detection for elderly out of floor sensor temporal signals.

2 Motivations

2.1 A need for monitoring

Frailty and falls in an ageing population. According to the World Health Organization, the worldwide population of people above 60 years is growing faster than any other age group [196]. In fact, when looking at prospects, (Figure 1.1-a), the age pyramid tends



(a) Worldwide population pyramid

(b) Death rate due to falls in USA, year 2018

Figure 1.1: Prospect over the global population age (a) and death rate according to age in USA (b). Data source: (a) United Nations, Department of Economic and Social Affairs, Population Division [182], (b) Centers for Disease Control and Prevention, Injury Center, Injury Prevention & Control [36].

towards a "bell" shape instead of its present "triangle" shape, thus showing the growing ratio of elderly in the global population. This ageing population is prone to *frailty*, a notion that has received increased attention from the medical community in the recent years [124]. Frailty is commonly defined as the early signs of significant health deterioration in the elderly population, and Fried et al. [60] define it more specifically as a clinical syndrome in which at least three criteria are present among unintentional weight loss, self-reported exhaustion, weak grip strength, slow walking speed, and low physical activity.

Along with other external factors, frailty contributes to a higher risk of fall. Since frailty level is rising with age, this makes elderly all the more vulnerable to falls, with more than 30% of elderly above 60 falling at least once a year [196]. A fall, while not dangerous for a young healthy person, may lead to serious injury to an elderly person, and sometimes even death. Data shows that the fatal falls rate increases exponentially with age (Figure 1.1-b).

Consequences of a fall. The repercussions of falls among elderly are multiple [36]. First, there are obvious direct physical consequences, most of them being fractures (wrist, hip) or head injuries (that can lead to brain damage). Aside from pain and discomfort, physical injuries reduce mobility, sometimes for a long time or even irreversibly, which leads to higher dependency.

In addition, a fall, even without any serious physical injury, can trigger a fear of falling again. This effect leads to a reduction of social interactions and activities which significantly increases the frailty level, and hence the chances to fall again. The lack of exercise is particularly damaging, as it has been shown that physical activity reduces muscle weakness, increases mobility, improves neuronal health, limits frailty, and reduces overall risk of death [117].

Finally, falls also have a critical economic impact. These economic repercussions whether

direct (medication, rehabilitation...) or indirect (societal productivity losses), are raising all over the world [196].

How monitoring can help. A monitoring system can be defined as an assistive device whose main purpose is to give insight over one person's activity for health purpose, either as general tasks such as indicators of activity, or more crucial tasks such as alerts in dangerous situations. Monitoring systems for elderly care can serve two main objectives, that are, following frailty levels and alerting when the person is endangered, in particular falls.

The early detection of frailty is considered of major interest by the medical community for prevention purposes, and targeted therapy strategies have been shown to significantly reduce the consequences of frailty [185]. Besides, it has been shown that fear of falling is likely to be diminished by the patient's perception of being monitored by a robust fall detector [33]. Consequently, the monitoring physical activity of the elderly in nursing homes is key, particularly as it allows focusing resources on people who are particularly vulnerable to frailty and falls.

The other key element to monitoring systems is the external help (caregivers or family) reaction time after a fall. Not only an accurate fall detector may prevent severe injuries from leading to death, but it also limits the time a person can pass on the ground after a fall. Indeed, even uninjured, nearly one elderly out of two is unable to get up without assistance [112]. It has been shown that a "long lie" (defined by Lord et al. [112] as more than one hour spent on the floor after a fall) can lead to serious complications (hypothermia, dehydration...) but is also associated with increased frailty (half of those who go through a long lie die within six months).

Setting up a reliable monitoring system. During the last twenty years, there has been an increasing interest in the development of smart systems to monitor individuals in their daily life, with fall detection systems alone representing a vast and varied literature. In that context, Tarkett aims at proposing a floor-based system that can meet several practical constraints when it comes to a daily use in real conditions [161]. These important constraints include in particular the intrusiveness, which is defined as both the capacity to be hidden from the patient and the respect of her/his privacy. There are several other criteria that could be defined when it comes to industrial purposes and practical use. This raises the first question of how to locate Tarkett's system in this plethoric offer of monitoring systems. Figure 1.2 displays an illustration of a floor-based monitoring system.

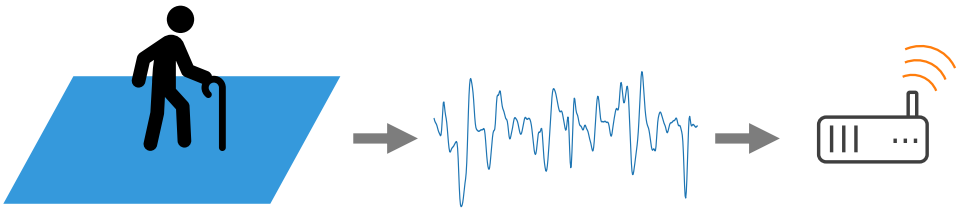


Figure 1.2: Schematics of a floor-based monitoring system.

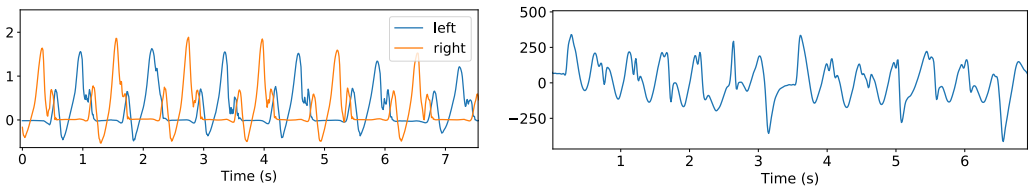
2.2 Processing time series for real world application

Understanding time series. The use of monitoring systems leads to the processing of a large amount of temporal signals. In fact, the past few years have witnessed a proliferation of sensor-based systems for health or medical purposes, hence raising the question of how to extract the best from data. Whether one-dimensional or multivariate, these signals are usually temporal. It can be for example electrocardiogram, electroencephalogram, acceleration of the body parts, video recordings, microphone, etc.

Depending on the task, dealing with temporal signals may raise several issues. First, signals may be very redundant or high-dimensional. For example, detecting events out of video might be easy since images contain a lot of information, but reveals to be computationally expensive and therefore demands some preprocessing steps that reduce signals into usable inputs for detection models. Secondly, the signals may be well fit for the task (e.g. electroencephalogram for anaesthesia monitoring, body acceleration for gait analysis) or much harder to interpret (e.g. radar for person monitoring, sound for fall detection). As a consequence, some tasks rely on simple heuristics to achieve their goal, and attempts to perform the same task with similar models with different signals may fail. Tarkett's floor-based system is intended for person monitoring but due to the nature of the signal and to hardware constraints, yields rather complex signals. An illustrative example is displayed in Figure 1.3, where walk signals are shown recorded with different sensors: the Tarkett's pressure sensor and a foot-fixed accelerometer (signal is taken from a public data set intended for gait analysis [177]). In that context, the general question of classification over time series becomes key to our problem.

Real-world applications. Another important aspect of this framework is the confrontation of controlled environment with real-world applications. This can be expressed in different ways.

First, many applications require to work in real time or at least in short time. This is obviously the case for fall detection systems but also any monitoring task that has crucial value in the early detection of events (e.g. anaesthesia control) or that need a short time response for practical use (e.g. medical diagnosis). Although many applications may benefit from powerful computers that can easily process the intended tasks, others are embedded



(a) Foot-attached accelerometer (b) Tarkett's floor sensor

Figure 1.3: Example of walk signals from different sensors. (a) displays a walk from a healthy subject recorded with two accelerometers attached on both feet. Each accelerometer records acceleration and angular velocity along four axes (X, Y, Z and the vertical axe), hence yielding 16-dimensional time series in total. We show the acceleration along the direction of the walk for both feet. (b) shows a walk from a healthy medical staff member of a nursing home, recorded with the floor sensor.

in much lower powered devices, hence introducing computational constraints. This is the case for example with cars that embed more and more sensors, and necessitate complex tasks (such as autonomous driving) to be performed at real-time and with limited hardware. Our case is not so different in the sense that the system comes as a whole i.e. the sensor and a processing unit, thus imposing hardware constraints, both in memory space and processing power.

Secondly, most traditional machine learning methods usually account for convenient hypotheses, a first one being that the learning procedure has access to a sufficiently large labeled data set which, in real applications, is already a crucial issue. When innovative systems are built, data collection is most of the time costly, whether in time or financially. Indeed, data may be fastidious to label due to complex signals to interpret that need expert eyes (e.g. medical data), or the large variety of signals that need to be labelled in order to encompass all possible unknown input signals (e.g. activities of daily living). This may lead to small amount of labeled data, or imbalanced class distributions (i.e. when a class is under-represented with respect to the others). Hence, data scarcity and class imbalance can become a key issue when one wants to build a reliable learning model.

Finally, to alleviate data scarcity or cost issues, data might be collected in a *controlled environment* or even synthesized, which is likely to induce a dissimilarity with the *operational* data, i.e. data one wishes to process with the final application. Since it may occur in a very wide range of topics, the general problem of dealing with differences between the training environment and the desired task has received an increased attention, and falls into a field of machine learning called *transfer learning*. In our framework, we come across a typical transfer situation with two similar (yet different) data sets, one being built in a controlled environment and the other being collected in real-life. For this reason, this work also intends to address the transfer learning problem.

3 Contributions

- Chapter 2 : Person monitoring: a review on systems and detection methods
  The first contribution of this thesis consists in an overview of existing monitoring systems designed for fall detection. This review is conducted along the processing chain from the signal generation to the final decision algorithm. It allows first to place Tarkett's system amid this vast range of systems, and, with the help of a set of criteria built for daily use systems, it highlights the benefits of floor-based systems. A part of this review is dedicated to time series classification, outlining the more frequent use of sophisticated supervised classifying methods over distance-based when signals are challenging.

- Chapter 3 : From floor sensor to fall detection
  This chapter explores a model based on the well-known random forest algorithm. The model is trained with an *experimental* fall data set built specifically for the fall detection task, in which events were performed by volunteers over a pilot site. As the model is intended to work in real-time, we propose a temporal aggregation of the random forest outputs so that the number of false detection is reduced. A data augmentation procedure is also proposed and significantly enhances the accuracy of the model. As we aim to produce a model intended for an embedded application,

we study a method to reduce the cost of feature computation while retaining most of the discriminative power and show that the model still performs well under this constraint.

- Chapter 4 : Transfer learning on decision tree
  After having set a first model for fall detection, we explore its performance over real data, i.e. data recorded in nursing homes, thus showing room for improvement. To that end, transfer learning procedures over decision trees are explored. These procedures adapt already learnt models to newly labeled data that may differ from training data. We propose several variants over seminal works that proposed two transfer algorithms. These variants address the previously mentioned class imbalance problem that we came across when designing our fall detector. The proposed algorithms show promising results over imbalanced data and we suggest improvements for more generalization.

- Chapter 5 : Elderly activity monitoring with neural nets
  The last chapter explores the perspectives of performing general monitoring with Tarkett's floor sensor. As the main drawback of this type of sensor is its inability to distinguish between users (the system delivers a one-dimensional signal for a whole area), we propose a method that aims at discriminating between elderly and caregivers. For that purpose we use a convolutional neural network, that we train to focus its attention on gait signals (since this is the most common recorded activity). Due to a limited amount of data, our method involves a pre-learning procedure that includes convolutional dictionary learning and a step detecting convolutional neural network. The model is well performing even over more complex signals than simple walks, thus potentially leading the way into the development of more complex tasks such as activity classification or individuals identification.

4 Publications

Chapters 3, 4 and 5 of this thesis each resulted in a publication.

- 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
  https://ieeexplore.ieee.org/document/8037597

- 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
  https://ieeexplore.ieee.org/document/8995296

- L. Minvielle and J. Audiffren. Nursenet: Monitoring elderly levels of activity with a piezoelectric floor. *Sensors*, 19(18), 2019
  https://www.mdpi.com/1424-8220/19/18/3851

2

Person monitoring: a review on systems and detection methods

Contents

Abstract

This chapter provides a general review over works that are related to the problematic previously described, that is to build a reliable fall detection system using time series derived from a floor sensor. To this end, the chapter is organised through the prism of fall detection systems, hence encompassing several ways of acquiring usable signals to perform monitoring. Systems are described according to the processing chain, from signal acquisition to final decision, and we focus on the matter of detection in temporal signals.

1 Introduction

Several surveys are available on fall detection methods, some of them being sensor-specific [7, 139, 210], giving short overviews [49, 130, 141, 199] or trying to encompass a great number of contributions [56, 86, 125, 183, 206]. In terms of fall detection system classification, most reviews use a division over the type of sensor that is used. However, they do not all use the same categorisation. The most common way to separate fall detection systems was given by Yu [206] in 2008 and taken up later by others [79, 125, 183], dividing them into three categories: wearable devices, ambience sensors and camera-based systems.

Among all explored surveys, we give a description of some that drew our attention in the way they tackle the subject. While first attempts are rather brief overviews of a field that was not as developed as it is today [141, 206], more extensive reviews were proposed in recent years. One of the most exhaustive was proposed in 2013 by Igual et al. [86], in which systems are grouped into *context-aware* (i.e. sensors deployed in the environment such as cameras, floor sensors and microphones) and *wearable* ones, this latter category being divided into external devices and smartphone-embedded sensors. Authors give extended summaries of all cited methods, and show the recent general trend in the field along with a list of main drawbacks a fall detection system can face.

El-Bendary et al. [56] explore in more depth the main causes and consequences of falls and describe systems developed for commercial purposes. To our knowledge it is the only survey that describes systems that are actually available for sale. However, they rather focus on prevention than fall detection systems, including for example systems built for wheelchairs.

On the matter of wearable systems, Pannurat et al. [139] proposed a thorough review where they classify systems according to the body location on which they are placed. They also indicate for each paper the level of variety of signals used for training, which is a valuable piece of information since the natural high diversity of events in real-world applications is challenging.

More recently, Khan and Hoey [91] proposed an original taxonomy based on the availability of data. They separate contributions between *sufficient training data* and *insufficient and no training data*, each category being divided according to the type of sensor, most often between wearable and video systems. This is the only encountered review that tackles the subject of data availability, which is crucial in such applications where the core objective is to detect a rare event among a wide variety of them.

Vallabh and Malekian [183] proposed in 2018 a rather exhaustive review in which they describe most used algorithmic methods, i.e. threshold-based and machine learning approaches, and propose a large tour of fall detection systems (wearable, ambient, and vision-based). One advantage of this review is that each category is accompanied with a comprehensive list of disadvantages, and they finally discuss personalized models.

Less exhaustive and rather focused on recent advances, the survey proposed by Xu et al. [199] gives a selection of the most cited papers before and after 2014 and highlights current trends along three key areas: sensors, algorithms, and performances. Although willingly not comprehensive, this survey gives a good view of recent evolution of fall detection systems, even introducing a new area for Wi-Fi and radar-based systems.

Figure 2.1 gives a schematic overview of fall detection systems along the three main axes

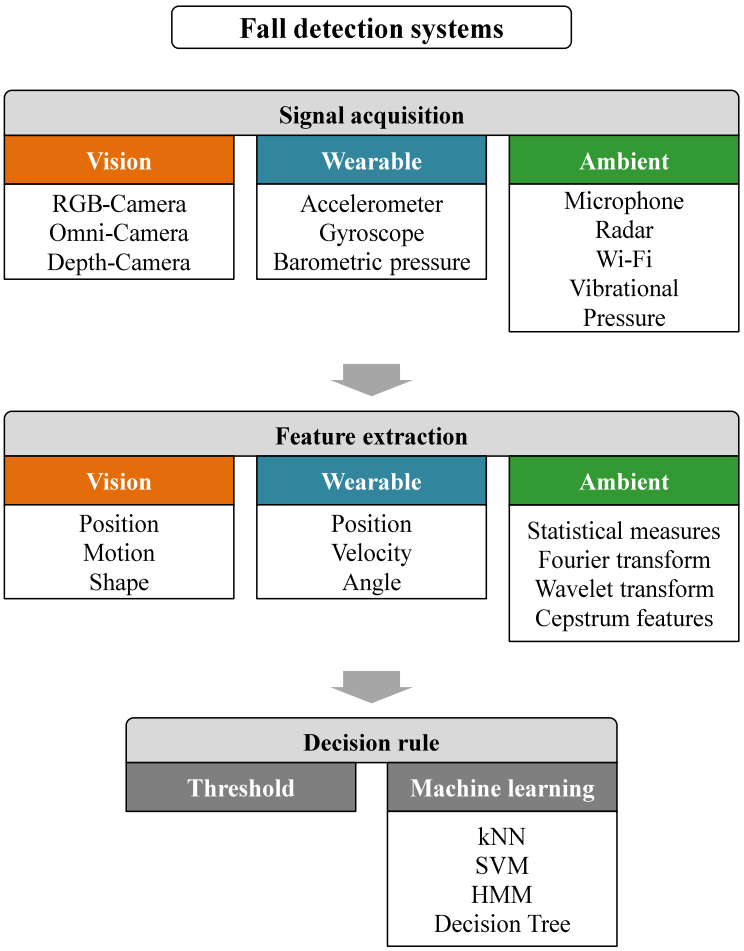


Figure 2.1: Summary schematic over fall detection systems.

that are described in further details through the rest of the chapter: signal acquisition, feature extraction and the decision making. In the following, we first present contributions in fall detection systems from a sensor point of view. Then, we expose the main used methodologies for the detection problem, from the variable extraction to the decision process, and we finally give summary tables of main contributions. We also consider literature that deals with the subject of event detection in time series (regardless of the fall detection problematic).

2 A sensor-oriented tour of monitoring systems

Fall detection systems are generally evaluated along the following criteria:

- coverage (Is the device covering the whole concerned area?) and occlusion (Can the patient be hidden by an object?)
- intrusiveness (Is the device hidden from the patient? In what extent does it respect of her/his privacy?)

- signal quality / information (Is the signal easy to exploit to perform the detection task?)
- robustness (Is the system effective under external perturbations? Does the system still work when the environment changes?)
- ease of installation / use
- scalability (Is it adaptable to any area?)

Based on the categorization system from Yu [206], which is the most used and actually fit to the current literature on the subject, systems are divided in vision-based, wearable, and ambient sensors.

2.1 Camera-based systems

Cameras have been widely used for fall detection systems. Image sequences provide a lot of exploitable information making detection or even recognition an easier task with than one-dimensional data. They allow users to detect multiple events simultaneously, the labelling is easy and with recorded data, it is possible to perform post verifications of the system outputs. Zhang et al. [210] dedicated a survey on vision-based methods, separating systems according to the devices that are used: single RGB camera, multiple RGB cameras, and depth cameras.

RGB cameras. Single RGB camera systems are inexpensive and easy to install. They can use regular field of view cameras [39, 100], which limits the coverage of the area. This can be solved using wide angle cameras, or even omnidirectional cameras [119]. These latter have a field of view of 360 degrees and a single narrow blind spot. They are usually made with shaped mirrors placed in front of and behind the lens. Compared to classical field of view cameras (e.g. usually less than 100 degrees), wide angle and omnidirectional cameras are more challenging in the fall detection process due to the distortion they imply [186] but offer the obvious advantage of having less blind spots. Using multiple cameras allows users to benefit from the additional depth information thus yielding better results in the detection process. Auvinet et al. [13] for example use visual hull, i.e. back-projection of the silhouette into space using camera parameters. However, such systems require calibration that leads to more difficult and expensive implementation. Besides, the processing steps such as background subtraction and tracking rely on RGB information that can highly vary under different lighting conditions.

Depth cameras. Depth cameras have also been used for fall detection systems since they provide additional valuable information. Most depth camera systems use stereo vision, Time of Flight (ToF) or structured light. Stereo vision uses two regular RGB cameras placed on the same plane to mimic the binocular vision. Like other multi-camera systems, stereo vision requires calibration, is computationally expensive and performs poorly under low-light conditions [154]. Non-RGB depth cameras on the other hand do not suffer from varying lighting conditions since they output the same image whatever the external light quantity. ToF cameras measure distance of objects by sending light pulses (usually infrared) that illuminate the scene. Distance is measured based on the time the light takes to travel from the camera to an object and back to the image sensor. Although accurate,

ToF cameras are expensive and are limited to low resolution images [154]. Structured light consists of emitting a known pattern over the scene, and computing depth of objects according to the pattern deformation. This principle is used by the Microsoft Kinect whose camera uses a sea of infrared dots to measure depth. Kinect also embeds a RGB camera and a built-in SDK for skeletal tracking with a 20-joints skeletal reconstruction. However, it seems that this latter feature is rarely used although being an interesting approach [162]. Since Kinect has become a cheap solution, it has been increasingly used for fall detection systems [55, 65, 154, 165]. Besides being way cheaper than ToF, it can acquire bigger resolution, however edges and far objects' depth is not well estimated.

Camera-based systems are well-performing solutions for fall detection when the monitored individual is in the system's field of view. They provide a large amount of information on the controlled area, making it relatively easy to detect a fall. However, this amount of information in the signal comes at high cost: these systems are indeed very intrusive, whether they are based on RGB or depth cameras, and can face occlusion or coverage issues depending on the area's organization.

2.2 Wearable sensors

Wearable-sensor-based systems are nowadays the most popular solution to detect falls and there is a vast literature over the subject, some surveys being entirely dedicated to the field [139]. Based on different technologies, they come as bands that one can tie to the concerned body part or in already equipped devices such as mobile phones and connected watches. These systems are affordable, easy to install, and give reliable signals.

Accelerometers and gyroscopes. In the context of fall detection, the most used sensors are accelerometers. They are inexpensive sensors that record the acceleration along one or three orthogonal axes of the body part on which they are placed. They make use of any body part location, but they are usually placed on the waist [26, 38], the chest [25, 88], the thigh [43, 111] and the wrist [48, 92]. Measuring orientation and angular velocity, gyroscopes are proposed as well but they are rarely used as the only sensor [23, 131] and instead are mostly seen as additional sensors to accelerometers [103].

Other wearables. Barometric pressure sensors [20, 175] measure the air pressure to estimate the height of the body part on which it is fixed. Pressure sensors seem to be used only as a support of accelerometer-based system, and not as a sole sensor, since they give less usable information than a tri-axial accelerometer. Doukas and Maglogiannis [52] make use of a microphone along an accelerometer, both attached to the user's foot.

Depending on the systems there may be only one sensor or several placed simultaneously on different body locations. Pannurat et al. [139] give an extensive review of wearable fall detection systems according to the body location. Research has been done into finding the location yielding the best results [24, 43] however it seems that there is no particular consensus on that matter.

Wearable systems are cheap, give good results and are quite easy to set up. However when dealing with elderly daily activity monitoring, they may suffer from one main drawback: patients may forget to wear those devices or even refuse to wear them, as they can be regarded as intrusive. In fact, when dealing with elderly, this risk is likely to increase and the very fact that systems are wearable makes them also easy to lose or avoid, which raises

the need for more reliable systems.

2.3 Ambient sensors

Although defined as non-wearable systems in Pannurat et al. [139], ambient sensing is most of the time a category defined to encompass the rest of sensing systems, i.e. all sensors except wearable and vision-based [125]. Their main purpose is to alleviate the intrusiveness problem. This field regroups the greatest variety of sensors: acoustic, vibrational, radar, Wi-Fi, and floor-based technologies.

Acoustic. Acoustic systems are based on one or multiple microphones (the sensing unit is called a microphone array in this latter case). For example, Xiaodan Zhuang et al. [198] use a single far-field microphone while Li et al. [104] use eight microphones arranged in a circular array and Li et al. [105] use the integrated array of four microphones of a Kinect. Acoustic systems are easy to install and require few processing power. However, these systems suffer from the variation due to distance between the device and the user, which may require additional calibration from one installation to another. Moreover, they are subject to unwanted noise and interference (e.g. environment, TV).

Vibration. Systems based on vibrational data make use of fall patterns collected by a vibration device placed in the room. For example, Alwan et al. [6] use a single piezoelectric sensor to record vibrations, and Werner et al. [193] use three to four accelerometer-based sensors placed on the floor at the middle of the room's edges. Vibration sensors are also used as additional sensors, as proposed by Zigel et al. [215] where authors use one accelerometer as a vibration sensor and a microphone, both placed on the floor near the wall. The vibration data is used for detecting and segmenting an event. When an event is detected, the sound data is added to determine if a fall occurred. These systems are cheap, easy to install and less obtrusive than acoustic. However, like acoustic systems, the output signal is subject to variation according to the device's location. Moreover, it also depends on the floor characteristics, and the size of the sensing area, which may require significant additional training for each new installation.

Radar. Now visible in the most recent studies [183, 199], radar-based systems are also proposed. These systems use a continuous-wave radar, which is based on the Doppler effect. When reflected by a moving object, the frequency of an emitted wave differs from the received one, hence allowing the detection of events such as falls. Liu et al. [110] use two radars placed at the edges of the room and Su et al. [166] use one radar located on the ceiling. Amin et al. [7] present a short review of radar-based methods and propose their own application with one radar placed on the ceiling of bathrooms in senior residence apartments. Radar systems are low-price and present promising results. However, like cameras, they suffer from occlusions and the difficulty to perform well when there are multiple persons in the room. Besides, this type of sensing raises the question of scalability since it depends on the distance between the sensor and the subject.

Wi-Fi. Wi-Fi systems also appear in one recent survey as a new trend in the fall detection domain [199]. The principle of Wi-Fi systems relies on the fact that given a Wi-Fi emitter and receiving antenna (which can be a simple laptop), the power is received via the direct path emitter-receiver, but also via reflection onto surrounding walls and any other objects. For a static environment, the received power is supposed to be constant, however when

there is movement (e.g. a walking person), then the power variates. Wi-Fi systems use the Channel State Information (CSI), which is an estimate of the relation between the transmitted and the received Wi-Fi signal, that comes as multiple one-dimensional signals. Recently, Wang et al. [189] proposed a system tested with three different experimental settings: a chamber equipped so that no reflection is possible on the walls, a large room (laboratory) and a smaller room (student's dormitory). The dormitory is equipped with one pair of emitter-receiver. In Wang et al. [187], authors use two receiving antennas and compute the difference of their CSI streams' phase. According to authors, compared to amplitude, phase difference is more sensitive and can better discriminate between similar activities. Although interesting, this technology seems to be in its early stages and suffers from issues such as the difficult availability of the needed metric to perform the sensing, external interferences, and a high dependency to the configuration in which it is installed, which limits its practical use [89].

Floor. Floor systems are based on sensors placed under the flooring. Current research shows that most floor systems are designed for gait analysis or activity of daily living [161]. There are however a few propositions specifically designed for fall detection. Tzeng et al. [179] use a pressure sensor as a trigger for an infrared camera system which performs the fall detection. Klack et al. [95] propose a pressure sensor based on piezoelectric effect, which comes as tiles. Although designed for monitoring and fall detection, authors do not present their fall detection system, most likely due to confidentiality issues. Rimminen et al. [149] propose a system called near field imaging (NFI), which consists of a matrix of 300 x 300 mm squares, one of them being fed by a low voltage signal, and the others used to measure the returning current [80]. Floor-sensor systems are not intrusive, do not suffer from occlusion and also have the advantage to be consistent since the sensing is equivalent at any location (unlike acoustic, vibration, and to some extent Wi-Fi and radar systems).

Among ambient-based systems, some may suffer from occlusion issues and lack of practical use (Wi-FI, radar) while others can face coverage issue, or at least calibration needs due to the distance between the sensing device and the patient (vibrational and acoustic systems). However floor sensors can deal with any area size without any coverage nor occlusion issues, which makes them good candidates for monitoring systems.

2.4 Conclusion

Fall detection systems rely on a very large variety of sensors, some of them being used since very recently. This vast variety, the great literature and the constant technical innovation over the subject may be explained by two reasons, the first being the growing demand for these systems and the second being the still on-going competition towards the design of ideal systems. Indeed, as of today, all systems present limitations to the implementation for real applications, where the sole purpose of good detection is not the only criterion. It should be noted that cost is not part our selected criteria. As far as we know there is no existent literature that proposes accurate data over system costs, apart from rare qualitative assessment. Besides, this matter is very relative to scale effect (when one produces at large volumes) and time (technology costs may vary rapidly).

The selected criteria rely on systems applicability, which is determinant. In that regard, there are significant differences between sensing methods, that are summarized in Table

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

Table 2.1: Sensors evaluation over key criteria for patient monitoring systems.

2.1. Wearable, vibration and floor sensors seem to best fit these requirements. However, if one considers the intrusiveness being key for daily patient monitoring, this rejects wearable sensors and leaves two families of sensors that are floor-related. While vibration sensors are easy to install but suffer calibration issues and show some issues with the area size, floor sensors are perfectly scalable and cover any area, although at a greater cost of installation, which makes them a good choice for long-term use.

3 Event detection in temporal signals

This section treats the subject of event detection in time series, which is a key element in activity monitoring from any sensing system. We focus on one-dimensional signals (or time series), which fits our scope for two reasons: our system is based on time series, and as we show in next section, vision-based methods tend to sum up image streams into one-dimensional signals thanks to simple descriptive features (e.g. body angle, velocity...). In the literature, event detection in time series has been used for a large variety of applications: pattern matching in financial data [197], seismic forecasting [123], detection of power quality disturbances [113], physiological data analysis [164], speech recognition [144], and many others. There are various ways of considering event detection depending on the type of data and the available knowledge on it. According to Abanda et al. [2], there are three main groups of time series classification methods: distance-based, model-based and feature-based. We briefly present the two first categories, i.e. non feature-based methods, in which we also include change point detection methods, and then we describe feature-based methods which are, according to us, the best fit to our problem.

3.1 Time series as sequences

There is a large part of literature that uses time series as raw instances and builds detection models directly on them. Depending on the prior knowledge, an event can be defined in different ways: an unexpected change in the signal, or a known pattern to retrieve, or a generation by a stochastic model.

Change point detection. Change point detection is a field generally related to the signal processing community. The main task of change point detection methods is to find changes in a signal, this latter being most of the time considered as multivariate time se-

ries. When given an unknown signal, the goal is to find the best segmentation according to a cost function that will evaluate the homogeneity within each segmented signals. This can be done through various ways, from maximum likelihood estimation to kernel-based detection [178]. These methods are mostly designed for detecting changes of regime in time series, without any specific a priori knowledge in general (aside from the number of change points). This reveals to be useful in unsupervised situations when one expect various types of signal but has no particular idea of what these signals look like. In our case, it could be applied to general activity monitoring when one wants to detect changes on which to perform further processing, however this is beyond our scope since we deal with annotated signals. For further details, the reader can refer to Truong et al. [178] who give a selective and technical survey over offline change point detection methods.

Distance-based methods. Distance methods compare a new instance with a set of labeled instances based on a similarity measure. Once a distance measure is chosen, it is generally exploited using the k-Nearest-Neighbour algorithm (k-NN). However the k-NN accuracy highly depends on the distance, which is a key element in the success of event detection [40]. There is a great variety of works over distances for time series and this is in fact the subject of entire reviews [2]. The most known (and used) distances are the classical Euclidian distance and the dynamic time warping distance, this latter allowing a non-linear mapping between time series [19]. More sophisticated ways are also introduced to define similarity measures over time series, such as the shapelets which are subsequences of series that are chosen as representatives of their class [202]. Work has also been carried out in the discovery of kernels for similarity measure, with a concern on kernels that are positive semi-definite or not, and the influence over classification results [2]. Although straightforward, distance-based k-NN methods are sensitive to noise in training data, and the main difficulty in this framework lies in the choice of the distance which determines the final task's performance. Recent work shows that time series classification methods tend to use distance to build feature vectors to be used in conventional classification schemes [2].

Model-based methods. Model-based classification relies on the assumption that series of the same class are generated by an underlying model. Hence, when an unknown series comes to the classifier's attention, it is tagged along the class of the best fitting model. In this field, the two best known families of methods are Hidden Markov Models (HMM) and Auto-Regressive Models (ARM). HMM are used to model randomly changing states. These states are hidden and the user has only access to observations. HMM are generally used to find the model parameters given observations, and then, given a sequence of observations (and model parameters) determine the most probable sequence of hidden states or the likelihood of observations. This can be used considering time series as the observation sequences. For example, Asadi et al. [11] propose a classification procedure where the main idea is to train a HMM for each class using ensemble method over all training sequences. Then, for any incoming series the likelihood over each HMM is computed, and the labelling is done according to the highest probability of generating the sequence. Auto-regressive models are stochastic models where a time series is explained by its past values and noise. They have been used for clustering as well as classification [41].

3.2 Time series as feature vectors

Other methods seek to transform time series into vectors of meaningful values, or at least measurements that allow a good representation for classification tasks.

Feature extraction. There is quite a huge number of possible features that one can compute over the signal. Fulcher et al. [62] propose a general analysis over about 900 time series and 9000 "operations" (i.e. features). Among other things, authors show that from the vast literature, they can cluster groups of features and even propose a list of 200 features which are supposed to be a summary of all feature extraction methods. In this summary we find a wide variety of features: autocorrelation, mutual information, stationary, long-range scaling, correlation dimension, wavelet transforms, linear and non-linear model fits, and measures from the power spectrum.

Fulcher and Jones [61] show later that when using the pool of features and a forward feature selection procedure, they can beat the traditional Euclidian-based 1-NN on most of the twenty data sets they tested, with a simple linear discriminant classifier. However, results of the linear classifier are more mixed when the comparison is done with the 1-NN associated with an enhanced dynamic time warping distance. Performances are indeed very data-dependant and authors suggest improvements can be made through the use of more sophisticated classifiers.

Given feature vectors, one can then use a large set of classification models. However, in the case of rare or unknown events, one may use other strategies.

Anomaly detection. When one class is scarce or missing in training, we usually refer to *anomaly detection* [142]. Sometimes associated with change point detection, this field includes methods that aim at detecting "abnormal" data which is generally rare or missing in training. In fact, there appears to be two main sets of methods. When no labels are available (unsupervised), the objective is to detect abnormal data without any previous knowledge, considering that the data set already contains "rogue" instances. This generally refers to *outlier detection*. The second set of methods, known as *novelty detection* encompasses approaches that seek to characterize "normal" data by training a model according to the only available class, and treat any unknown data as normal or abnormal depending on the model's output. This is considered as supervised learning since all training data is drawn from the "normal" class. We give here a few illustrative examples.

One of the most used algorithm in this field is the One-Class Support Vector Machine (OCSVM), which is an adaptation of the original SVM algorithm. A first approach was proposed by Schölkopf et al. [158]. The main idea is to map data into a feature space (corresponding to a selected kernel) and maximize the margin with the origin. The resulting hyperplane separates the space into two regions, the first one built around training data and the other one being its complementary.

Isolation Forest [109] consists of building a forest of isolation trees and determine a sample's degree of abnormality using the average path length needed to reach it. An isolation tree is built by successive binary splits with random choice in the feature and split value. Splits are done until the tree reaches a fixed depth or each training sample is isolated in a terminal node. The main idea is that abnormal instances are likely to be separated from others after fewer splits since they are different and less numerous. This method is designed for outlier detection since there is no "normal" model but rather an outlier-ness

measurement.

Breunig et al. [31] proposed an outlier detector based on a measure called Local Outlier Factor (LOF). Given a data point, the LOF gives a degree of how much this instance can be considered as an outlier. For that purpose, authors define a *local reachability density* and compute the LOF as the ratio between the density around the measured point and the densities around its neighbours. The neighbourhood is a fixed parameter and defined as the number of nearest points. Finally, one can use a threshold over the LOF to obtain an outlier detector.

There also has been a great deal of work with neural networks. Most of them are based on a reconstruction error, with a simple idea: train a neural network whose sole purpose is just to reconstruct data as it is given in input. Hence, when fed with "normal" data, the model is trained to reconstruct this type of data. When confronted to a new point, the reconstruction error becomes then a novelty score. It is usually done with hidden layers that contain few units so that the model has to learn a compressed representation of data. These models are called either Replicator Neural Networks (RNN) [75] or Autoencoder (AE) [174], the difference between them being the activation function of one hidden layer – a traditional sigmoid in the AE and a staircase-like in the RNN.

More recently, there has been an interest in anomaly detection using generative models. A Variational Autoencoder (VAE) is a probabilistic adaptation of autoencoder whose main goal is to learn a latent space that is sufficiently regular so that data generation (from the latent space) is possible. As VAEs learn distribution parameters, a reconstruction probability can be computed, i.e. the probability for a given unknown data that it has been generated from the latent variables learned with "normal" data [8]. Generative Adversarial Networks (GANs) are deep learning frameworks that are based on adversarial training. The general idea is to train two neural networks that will compete against one another. The first one – the generator (G), is trained to generate data that follows a "true" distribution (from which training data is sampled). The second one – the discriminator (D), is trained to discriminate between generated data and the true distribution. GAN-based anomaly detection methods make use of both networks by mixing the output of D and G to build an anomaly error [101].

Anomaly detection is a vast field of machine learning methods with many applications and still on-going challenges. Among these methods, novelty detection approaches are the most interesting to our framework since we aim to detect rare events (falls) among events that are easier to collect (any activity of daily living). These methods are inherently based on the a priori absence of abnormal data (fall events in our case), and the strength of these techniques resides not only on the model itself, whose choice among the large literature is not obvious, but also on a training set as representative as possible of the normal class. In the case of fall detection systems, as evidenced by the literature (more details in next section), the final purpose is rather to characterize falls than build a model of activity of daily living (ADL) and label as fall any "different" event. It is our belief that even if it implies to learn from very few fall events or simulated ones, it is still worth building from them than seeking to achieve a better ADL model, since ADL events may be a lot more varied than expected, combined with the fact that detecting a fall wrongly may be considered a higher error than missing one when it comes to industrial purposes [1].

[1] When implemented in a nursing home, a fall detection system sets off an alarm for each suspected fall

Supervised classifier. When one has at his disposal a reliable labeled set of data, a straightforward way to exploit it is to use classical supervised methods. Literature shows a use of the wide variety of available models to perform pattern recognition over time series. One can use, as distance-based methods do, the k-NN model, however it seems that time series classification methods rely on more sophisticated methods. In fact even distance-based times series classification methods tend to use distance for feature engineering and use more advanced supervised learning methods [2]. A widely used method is the Support Vector Machine (SVM), whose principle lies in the separation of data points with a hyperplane. The most interesting part is that this separation can be done not only in the original space but also in higher dimension spaces thanks to the use of kernels. Hence, the algorithm may find a linear separation in the new space, corresponding to a non-linear separation in the original space. It is used for example in gait classification by Begg et al. [16], who analyse performances with three kernels, showing that mapping data in higher dimension improve their results. Neural networks are also of use, with the example of Sternickel [164] who develop a model to recognize patterns in ECG time series. For this purpose authors use a neural network with a single hidden layer whose number of units is lead by the number of wavelets bands they use. As long as times series are transformed into a usable feature vectors, any classifier can be used. To prove the benefits of their work, Lines et al. [108] tested for example seven classifiers over 26 data sets. Results show that best performances are obtained among the most sophisticated classifiers of their selection, e.g. random forest and SVM, over simpler methods, e.g. 1-NN and decision tree. However among the most complex models results are mixed, showing that there is no particular best performing model and performances are rather data-related.

As we show in next section, fall detection systems also make a great use of classical supervised methods, but many other areas of applications seem to use such methods. Bolton and Hand [22] give a review on fraud detection methods, showing a large variety of used approaches and especially neural networks. Mahela et al. [113] present a comprehensive survey over power quality disturbances detection, in which numerous classifiers have been pointed out, among them the SVM and neural networks.

In the end, in the case of supervised learning for time series classification, it seems that there is no universal rule over the choice of the classifier, apart from general machine learning considerations, such as the complexity of the model, the size of the training set, the ease of implementation, the model's interpretability, the execution time etc.

4 Features for monitoring systems

In order to perform the classification task, one may feed directly the decision model with the signal. This can be done when this latter is considered to be sufficiently representative of human movements and the model is specifically designed for time series. In this case a few processing steps are sufficient to perform the detection. When regarding fall detection literature, since signals to be processed are rather complex, methods tend to use feature extraction, or at least a few transformations of the output signal. This turns out to be quite sensor-dependant since monitoring systems include a wide variety of signals. For this reason we distinguish here between image-based systems and the rest, which are mostly

of each connected room. With potentially dozens of residents, even the lightest false alarm rate can become unmanageable for caregivers.

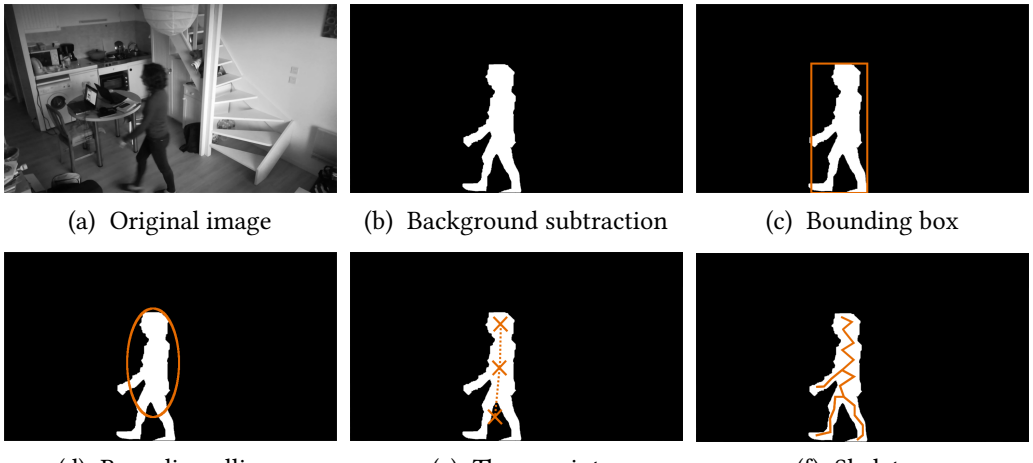
one-dimensional-signal-based systems.

4.1 Vision-based approaches

When using image streams as input, the temporal nature of the event we look for makes the signal hard to use as a direct input for a detection model. Indeed, using for example distance-based methods would lead to measure similarity between whole images (or even sequences) hence using models over a huge number of dimensions. Therefore, most vision systems use several processing steps to reduce image stream into usable and understandable signals.

Silhouette extraction. Systems built on video recording rely on a first step called background subtraction. This process consists of removing what are called *background* pixels from the successive images in order to keep the relevant ones, i.e. *foreground*, which in this case are supposed to be the monitored persons. This first step is key before being able to compute any features. The idea is, for each pixel to determine whether it belongs to the background or not. Most methods use an estimate of a background image, and at each time t, for each pixel of the image, depending on the form of the background estimate, a test is done to classify the pixel as background or foreground. The easiest method for background estimation is to take the mean value over several successive images. Then each pixel that falls within a threshold of its corresponding background value is set as background [154]. There are several other methods such as median, or Gaussian average filtering. Since these latter do not handle well the time variation of background changes, finer methods such as mixture models can be used [165]. Once the foreground is extracted, most methods use morphological operations to homogenize the silhouette, such as the opening operation [13, 118], which consists of an erosion and a dilation of the image. Some systems rather use tracking procedure and notably particle filtering [152], that consists of estimating the position given the successive observations with the use of a succession of prediction and corrections steps.

Silhouette into features. Once the silhouette is extracted for each time t, it is reduced into features that will feed the final the decision rule. The simplest ones are the silhouette's position and its velocity, which are computed by reducing the silhouette into a single centroid. However, to improve the level of information, one can extract shape and orientation features. Shape features are obtained through various methods. A first approach is to draw a geometrical object around the silhouette and extract the object's parameters as features. It can be done using a bounding box around the target, hence extracting the height and width [118], or using the parameters of an ellipse, i.e. center, orientation and semi-axes lengths [153]. Others use custom form parameters, such as three-point representation that estimates the position of the head, trunk, and legs of the monitored person [39]. A finer shape representation called a skeleton is also proposed in order to estimate a more complex body form. It is done for example using triangulation of the silhouette, which consists of filling the shape with triangles under some constraints, and connecting the centroids of the resulting triangles. The connected centroids form the final skeleton [205]. The Kinect comes with its own embedded skeletal reconstruction that can be used for feature computation [129]. Once the shape is summarized into a bounding box, an ellipse or a skeleton, one can compute its orientation, which adds relevant information [39, 100, 186]. Systems



(a) Original image (b) Background subtraction (c) Bounding box

(d) Bounding ellipse (e) Three-point (f) Skeleton

Figure 2.2: Example of preprocessing methods for feature extraction in video-based monitoring systems. From the original image (a), the background is subtracted, hence leaving the area of interest (silhouette) (b). The silhouette can then be reduced to various intermediate models (c, d, e and f) so that interpretable shape features can be extracted.

that dispose of depth information, whether by using multiple cameras or depth camera, use reconstruction methods to get the coordinates in the camera coordinate system [13]. To characterize the shape, histograms can also be used. The idea is, given the silhouette's centroid, to construct a polar coordinate system and divide it in a fixed number of areas, each area corresponding to a bin [205]. Figure 2.2 gives illustrations of cited methods from original image to shape or orientation features.

When regarding vision-based feature extraction, methodologies may vary a lot from one system to another, however the aim is generally to characterize the position, the velocity and the shape of the silhouette. In the end, most methods finally operate on a few variables that change over time.

4.2 One-dimensional signals characterisation

One-dimensional signals are generated by various types of sensor. Depending on the nature of the signal (e.g. acceleration, sound, Doppler frequency...), specific features may be computed. However, we can find many similarities across fields.

Features for accelerometers. The most straightforward features are found within wearable sensor systems. It can be explained by the fact that among one-dimensional signal systems, accelerometers and gyroscopes deliver the most explanatory signals to characterize human activity. In fact some systems do not even compute a single feature over output signals and use a light preprocessing as the only treatment before performing the decision rule [52, 131]. However most methods use basic computation of the signals, such as overall acceleration over the three axis, which is the two-norm of the acceleration signal [48]. It is called either *root sum of squares* [25, 26], *total sum vector* [90] or *signal magnitude vector* [88]. Methods also use estimation of the body angle [26], or its velocity by integrating the acceleration [26, 90]. Custom features may be found, such as the total sum vector

over the signals filtered by a high pass filter. Called the *dynamic sum-vector*, it is used to compute the *vertical acceleration* [90]. Authors also investigate fast acceleration changes by computing the difference of maximum and minimum acceleration values over a sliding window, and this for each axis. To characterize the intensity of the acceleration, some methods use the integration of acceleration over a fixed window called signal magnitude area [20].

More complex features. When it comes to other sensors, i.e. acoustic, vibration, radar, and Wi-Fi, more elaborate features are explored. These sensors are a lot more subject to external perturbations since they record any object or person in the covered area, and not just the monitored person (as wearable sensors do). To extract more information from the signals, some systems use time-frequency representations. This is usually done using spectrograms, which represent the signal in two dimensions (time and frequency), the most popular method being the Short-Time Fourier Transform (STFT), which is a Fourier transform using a window over the signal [52, 105, 110]. This technique implies a trade-off between the frequency and time resolutions, that depends on the chosen window size. Other methods, especially radar-based, use wavelet transform [7, 166]. This representation allows the user to tune the feature extraction process. Indeed, wavelet functions are derived from a *mother* function which is translated and dilated. Since there are multiple wavelet families and scales, the wavelet choosing procedure requires more actions than with STFT. Audio-based systems rely on acoustic-specific descriptors that are widely used for automatic speech recognition [46], such as the Mel Frequency Cepstral Coefficients (MFCC) [104, 110, 215] and Perceptual Linear Prediction Coefficients (PLPC) [198]. Both MFCC and PLPC are based on the *cepstrum* which is a Fourier-based representation that was initially designed to better fit the human auditive system. With vibrational data, physics-related features can be computed such as Shock Response Spectrum (SRS) which is a representation of the response of a mass-spring system to an input (using here the vibrational signal as the input) [215].

Unlike vision-based systems that rely on the same main descriptors, features used for one-dimensional signals are a lot more diverse. It is obviously explained by the great variety of one-dimensional signal sensors used for fall detection. Indeed, some fields have their own set of descriptors based on the nature of the signal and the difficulty to interpret it. However they all tend to describe the signal either using straightforward statistical measurements or frequency-related descriptors.

5 Detection strategy in fall systems

Once features are extracted from the signals, the decision rule can be constructed. Literature shows that fall detection systems are based either on threshold techniques or supervised machine learning methods.

5.1 Threshold-based methods

When descriptors possess an acceptable level of separability over data, a simple and fast way to construct the decision rule is to use thresholding methods over them. Most wearable methods rely on this type of decision rule. The simplest methods use directly thresholds over their whole pool of features [26, 88, 131] while others use multiple thresholding

stages with rules between them. For example, Bourke and Lyons [23] use a two-level method: they first evaluate angular velocity with a threshold to decide if there is a suspicion of a fall, and in the latter case, angle value and angular acceleration are computed before being compared to thresholds to make the final decision. This is equivalent to thresholding over all features but it is done in a way that critical features are computed only when needed.

One can also use thresholds combined with time, for example Bourke et al. [25] use the following rule: if the body angle does not exceed a certain value during N seconds after a suspected fall, then a fall is suspected to have occurred. Thresholds may also be used over activity measurement after a potential fall and according to its value, confirm or reject a suspected fall [193]. More sophisticated methods also use operations between features over a fixed time length, for example [90] use the difference between the maximum and minimum value of the total sum vector, and [43] also compute min-max thresholds, however on two time windows, intended for respectively event detection and fall confirmation. Finally, a mix of these approaches has been proposed by Degen et al. [48], who perform it in three steps: they first detect a high velocity with a simple thresholding, then the impact with a min-max threshold, and finally a long enough inactivity so that the system can confirm a fall has occurred.

According to literature, in most cases thresholds seem to be manually learnt, i.e. besides human observation, there is no particular metric on which the threshold selection relies on. In the other case, they are based on metrics such as the extrema of the features in the data base of a given class [26]. Threshold methods are easily set up and seem to give good result. However, they are a lot more suited to vision-based and wearable systems since they give very interpretable signals. When confronted with more difficult signals, machine learning methods seem more adapted.

5.2 Statistical models

When it comes to statistical models, there is no particular learning method that emerges from fall detection systems literature. Like previously mentioned for time series classification, the choice of a method is done over more general considerations such as data availability, computation etc. Since they lack fall events (and yet dispose of numerous non-fall events), some systems rely on anomaly detection, and others, due to the nature of the signal, use very specific methods. However, most methods rely on classical supervised learning models.

Audio-related models. As seen in previous section, audio systems use specific methods. For example, Li et al. [105] improve their previous fall detection system [104] by adding blind source separation (BSS) in the process. BSS is achieved with Non-negative Matrix Factorization (NMF) over the short-time Fourier transform of the signal, which allows to identify and isolate the contribution of a suspected fall in the audio signal. The isolated signal is finally classified with MFCCs features and a k-NN. Works have also been proposed inspired by speaker recognition methods [198]. Traditional speaker recognition methods transform acoustic signals into feature vectors according to classical audio features (MFCC, PLPC), and the most standard approach consists of fitting a Gaussian Mixture Model for each type of event and using the maximum likelihood to determine in which Gaussian Mixture an unlabeled event falls into. More advanced approaches use a Universal Background

Model (GMM UBM), which is a single GMM trained with all data. Then, for each new audio entry, UBM parameters are adapted using maximum a posteriori [35]. These new parameters are called supervectors. Out of an upper bound of the Kullback-Leibler divergence, one can define a kernel for these supervectors that satisfies the Mercer's condition and then use a SVM to perform the final classification [35].

Anomaly detection. Among used statistical methods for fall detection, a few approaches use anomaly detection techniques in which models are trained over activity of daily living events. For example, Wang et al. [189] use two stages to detect fall events from Wi-Fi signals. The first one is an unsupervised anomaly detector (i.e. outlier detector): it uses a metric called Local Outlier Factor (LOF) which is, for each data point, the ratio between the average local density of its neighbours to its own local density. This step detects any human behaviour that differs from staying still. The second stage consists of passing a Gaussian-kernel-based One-Class SVM (OCSVM) on the extracted anomaly. This whole method (LOF + OCSVM) is in fact a double anomaly detection, the first being unsupervised and the second being supervised. Another similar approach is proposed by Yu et al. [205], who use a combination of two thresholding procedures and the output of a OCSVM. The thresholding rules are respectively intended to detect a large human movement and the duration of an abnormal position, which is detected by the OCSVM. The OCSVM is not learnt with a full training batch but rather in an online fashion (Online OCSVM). They do so not only to speed up the training but more importantly to adapt the model to new labeled events.

Classical supervised learning. Aside from methods that are either domain-specific (audio) or designed for a lack of labelled fall events (anomaly detection), the rest of statistical-based methods rely on traditional supervised learning. Although familiar with threshold methods, some wearable systems use supervised learning, for example Doukas and Maglogiannis [52] use a SVM over the raw signal and simple features. They compare three kernels, finally choosing the Radial Basis Function. Several methods use a mix of a light thresholding procedure followed by an evaluation with a supervised model. The first is often intended to detect an event and the second to classify it as fall or non-fall. For example, Zigel et al. [215] (vibration sensor) use a threshold over the energy of the vibration signal as a first step to detect an event, and a Quadratic Discriminant Analysis (QDA) over additional features (MFCC, SRS...) for the fall detection procedure. Similarly, Stone and Skubic [165] (Kinect) use a threshold over the vertical state to detect a suspected fall, which is then confirmed (or not) with a decision tree over different features. Liu et al. [110] (radar) filter events with a quantity called "energy burst", which is the sum of Short-Time Fourier Transform values over a predefined frequency range. Events are then classified with a supervised algorithm. Authors compare performances with a Support Vector Machine (SVM) and a k-Nearest-Neighbour (k-NN), the latter giving better results. Su et al. [166] (radar) also use a prescreening stage which consists of thresholding the output of one preselected wavelet transform. The wavelet function and scale are chosen according to fall / non-fall discriminating power. Then, they compute additional wavelet-related features and use them as a feature vector for a k-NN. Finally, Wang et al. [187] (Wi-Fi) use as well a double detection technique: they first detect a *fall-type* event through thresholds of the phase of the CSI signal, and then apply a kernel-SVM on the selected signals.

It appears that when the signal is more challenging, fall detection methods tend to use

more sophisticated algorithms. Indeed, we note that almost all wearable-sensor systems are based on simple thresholding. This is mainly due to the fact that the signal is very interpretable and not subject to external perturbations nor occlusions. However, when using other sensing methods that deliver more complex signals (radar, Wi-Fi, vibration, pressure...), then using machine learning becomes a necessity. In their survey, Xu et al. [199] show that machine learning tends to be increasingly used among fall detection systems. They show at the same time that the part of wearable systems is decreasing, which is obviously correlated since there is then a growing part of more complex signals to deal with.

6 Conclusion

From the literature, we observe that there are many ways of acquiring a relevant signal to detect falls in a monitored area. However, when dealing with real applications such as elderly monitoring in nursing homes, a fall detection system must meet specific criteria. We showed (see Table 2.1) that from a sensor point of view, proposed systems are very unequal when evaluated through this set of criteria.

Methods mentioned in this chapter are regrouped in three tables: Table 2.2 for camera systems, Table 2.3 for wearable sensors, and Table 2.4 for ambient systems. Within each table, systems are ordered by year. Whenever available and necessary, we also indicate the number of sensors and their position. Performance results are also shown (when available) but are present as mere indications and not as the main comparison argument. As pointed out there are more decisive criteria than performance, and, furthermore, results are very data-dependant. As a matter of fact, most studies use their own data collection protocol and a system may perform differently from one to another. Pannurat et al. [139] give a comprehensive table that shows the variety of events in each class (i.e. fall or activity of daily living) for each reviewed fall detection method, showing the great variety of protocols across all proposed systems. In addition, these protocols are conducted in *controlled environment*, which gives little clue about how well systems perform in real situation.

From these tables we observe the clear distinction of methods from one sensor family to another, notably feature extraction procedures which are very dependant on the signal's nature. Concerning the classification task, likewise, there is not one particular approach that addresses the fall detection issue, but rather a whole set of methods that depend on the type of signal that has to be processed. However, whenever possible, i.e. when the assumption is made that one has a reliable labeled data set of each class (falls and activity of daily living), time series classification methods tends to use supervised algorithms over statistical feature vectors. This is even more the case when more complex signals are used, where simple thresholding methods would fail.

In this context, the Tarkett's floor sensor seems to meet the main criteria for a daily use in nursing homes, and this fall detection systems overview leads us to consider an approach based on statistical features associated with a supervised model, whose details are given in next chapter.

Study	Year	Sensor	Features	Detection	Results
Lee and Mihailidis [100]	2005	RGB camera (ceiling)	Position, spatial orientation and velocity	Threshold	Sens 94% Spec 81%
Miaou et al. [118]	2006	Omni-camera (ceiling)	Height, width and weight	Threshold	Sens 90% Spec 86%
Rougier et al. [153]	2007	RGB camera (wall)	Shape and motion (ellipse)	Threshold	Sens 88% Spec 87.5%
Vishwakarma et al. [186]	2007	Omni-camera	Shape and motion (Angle, gradients)	Threshold	Acc 74% Sens 62% Spe 84% (indoor and multiple persons)
Hazelhoff et al. [76]	2008	Multiple RGB cameras (2)	PCA-based feature extraction	Multi-frame Gaussian classifier	Sens 91-100% (realistic activity)
Auvinet et al. [13]	2011	Multiple RGB cameras (2-6) with high FOV	Shape (Vertical volume distribution ratio)	Threshold	Sens 80.6% Spec 100% (3 cameras, no occlusion)
Rougier et al. [154]	2011	Kinect	Position and motion	Threshold	Acc 98.7%
Yu et al. [205]	2013	RGB camera (wall)	Shape and position (ellipse, histogram)	Online One-Class SVM	Sens 100% Spec 97%
Gasparrini et al. [65]	2014	Kinect	Position (Height)	Threshold	N/A
Chua et al. [39]	2015	RGB camera (wall)	Shape (three-point representation)	Threshold	Acc 90.5%
Stone and Skubic [165]	2015	Kinect	Shape and motion (Vertical state, Min vertical velocity...)	Threshold and Decision Tree	Sens 71-98% (far & not occluded)
Nizam et al. [129]	2017	Kinect	Velocity and position (Kinect joints)	Decision Tree	Acc 93%

Table 2.2: Summary table of literature review: camera systems.

Study	Year	Sensor	Features	Detection	Results
Degen et al. [48]	2003	Accelerometer (Wrist)	Acceleration (norm), velocity (two estimations)	Threshold	Sens 65%
Nyan et al. [131]	2006	Gyroscope (3, Trunk)	None (direct use of the signal)	Threshold	Sens 100% Spec 92.5 - 97.5%
Jeon et al. [88]	2007	Accelerometer (Chest)	Acceleration (norm) and velocity	Threshold	Acc 96.52%
Bourke and Lyons [23]	2008	Gyroscope (1, Trunk)	Acceleration (norm) angle, angular velocity	Threshold	Sens 100%
Doukas and Maglogiannis [52]	2008	Accelerometer + Microphone (Foot)	Acceleration (raw), sound peak frequency, sound relative amplitude	SVM	N/A
Bourke et al. [25]	2008	Accelerometer (Chest)	Acceleration (norm)	Threshold	Sens 90% Spec 99%
Kangas et al. [90]	2009	Accelerometer (Waist)	Acceleration (norm), dynamic sum vector, vertical acceleration, velocity	Threshold	Sens 97.5% Spec 100%
Dai et al. [43]	2010	Accelerometer (Thigh)	Acceleration (norm) and vertical acceleration	Threshold	Sens 97.33% Spec 92.3%
Bourke et al. [26]	2010	Accelerometer (1, Waist)	Acceleration (norm), velocity, angle	Threshold	Sens 100% Spec 100%
Bianchi et al. [20]	2010	Accelerometer + Barometric (Waist)	Acceleration (norm), signal magnitude area, angle, differential pressure	Threshold	Sens 97.5% Spec 96.5%

Table 2.3: Summary table of literature review: wearable sensors.

Study	Year	Sensor	Features	Detection	Results
Zigel et al. [215]	2009	Vibration and sound	Length, energy, duration, SRS, MFCC	Threshold and QDA	Sens 97.5% Spec 98.6%
Xiaodan Zhuang et al. [198]	2009	Microphone	PLPC and energy	GMM and SVM	Sens 70% Recall 64%
Rimminen et al. [149]	2010	Electric Near Field	Number of activated tiles, longest dimension on ground, and sum of magnitudes	Markov Chain	Sens 90.7% Spec 90.6%
Werner et al. [193]	2011	Vibration (3-4 sensors)	Peak-to-peak, average, wieghted average, duration	Threshold	Sens 87.7% Spec 97.7%
Liu et al. [110]	2011	Radar (2, ground)	Energy burst (with STFT) and MFCC	k-NN (k=3)	AUC 0.96
Li et al. [104]	2012	Array of 8 microphones (circle)	MFCC	k-NN	Sens 76% Spec 69% (SNR 6dB, SIR 4dB)
Li et al. [105]	2014	Array of 4 microphones (Kinect)	Energy ratio (using blind source separation and STFT)	k-NN	Sens 93% Spec 78% (SNR and SIR at 0dB)
Su et al. [166]	2014	Radar (ceiling)	Wavelet transform	k-NN (k=1)	Sens 97.1% Spec 92.2%
Amin et al. [7]	2016	Radar (ceiling)	Wavelet transform	k-NN (k=1)	N/A
Wang et al. [189]	2017	Wi-Fi	Standard deviation, derivative...	One-Class SVM	Prec 78% Spec 79% (dormitory)
Wang et al. [187]	2017	Wi-Fi (two links)	Same as [189] + 2 customs	SVM	Sens 91% Spec 92%

Table 2.4: Summary table of literature review: ambient systems.

3

From floor sensor to fall detection

Contents

Abstract

This chapter describes a fall detection system based on the combination of a floor sensor and a predictive model. The sensing device, which is a piezoelectric sensor placed directly under the flooring of the monitored area, outputs a one-dimensional signal to the fall detection algorithm. This latter includes feature extraction and a decision rule built on temporal aggregation of the outputs of a random forest model. For industrial optimization reasons, we propose a simple feature selection procedure and observe its impact on performance, along with a model improvement through data augmentation. This system has clear benefits when regarding main fall detection system issues, and provides good results on an experimental data set. Parts of this work have been published in Minvielle et al. [121].

Tarkett's objective is to propose a high performing system based on a floor sensor. To that end, the company developed a sensor that comes as a thin layer that is invisible to the user since it is installed under the flooring. The sensor configuration is such that it can cover any area and shows valuable implementation advantages. The resulting fall detection system is meant to be independent for each installation. This means that in a nursing home, each room has its own processing unit that is connected to a central alarm system. The processing unit is made as small as possible so that it can be easily installed in any configuration. Therefore, some computing constraints are at stake, whether in terms memory or processing power. In that context, we aim at proposing a simple yet performing detection model.

First, we describe the floor sensor on which the Tarkett system is based on, with its advantages and limitations, from the core sensing material to the final output signal. The experimental data set that was acquired for the fall detection task is then described, followed the main contribution of this chapter. A predictive model based on data augmentation and time aggregation of a supervised model is proposed, along with a feature selection method that addresses the hardware constraint, and performance results are finally given.

1 From floor sensor to signal processing

1.1 Floor sensors

Many floor sensor technologies have been proposed, most of them for gait analysis purposes. More details on gait analysis are given in Section 1.2 on page 113 , and we give here a short overview of main floor technologies. Floor technologies here encompass systems that are integrated in the flooring. It means that vibration devices for example, are out of this field. They are usually placed directly under the flooring and are invisible to the patient. One can create a floor sensor through different means. In literature, we identify four main different technologies: capacitive sensors, contact sensors, piezoresistive, and piezoelectric sensors.

Capacitive. The capacitive effect is the property of certain material to retain electric charges. Between two electrodes, a capacity depends on the distance and the permittivity between these electrodes. Hence one can use these two factors to create a sensor. A first way to create a capacitive sensor is to put an insulating material between two electrodes parallel with the ground. Then, when a force is applied on the sensor, the distance between the two electrodes decreases and the capacity increases. However, to the best of our knowledge there is no particular floor technology that uses this approach.

All capacitive floor technologies seem to use the second approach, which consists of setting up the electrodes on the same horizontal plane (see Figure 3.1). The idea is that when current is fed to one of the electrodes, an electric field is created between the two electrodes. This electric field depends on the current intensity but also on the dieletric properties of the medium between electrodes. Hence, the capacity is constant when there is just air above the measuring electrode. However, when a charged object (such as a human foot) is approaching the field, the capacity changes. Capacitive sensors are used as matrices of tiles, these latter being of various size. Each tile measures the capacity of what lies above, hence allowing the patient's monitoring through an array of binary values [80, 184].

These systems have the advantage of detecting motionless persons since it uses the field deformation. Besides, it allows users to detect a human presence even if one does not touch the measuring tile, since being in the tile's neighbourhood is sufficient. However capacitive sensors need a constant power supply. Besides, they are sensitive to humidity, making them unsuitable for elderly care as a great part of fall happen in the bathroom [196].

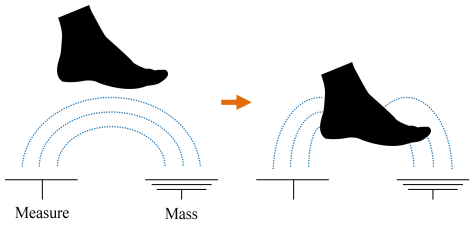
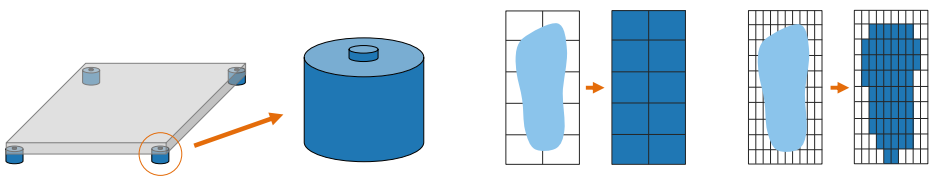


Figure 3.1: Principle of capacitive sensor. An electrode is used to receive current while the others will serve as presence detectors.

Contact sensors. Some works proposed the use of load cells, which are cylindrical devices that are usually placed at the four corners of each tile. Working as pressure sensor, they use various technologies to measure the applied force, the most used being the strain gauge [3, 132]. Binary switch sensors are simple sensors that output a two-value signal depending if a weight is sensed or not. Installed as matrices under the floor, since there is just a spatial information and no pressure signal, the higher the resolution the better. For example Yun et al. [207] use a large resolution of more than 700 sensors per square meter, while Suutala et al. [171] use 100 sensors per square meter. Figure 3.2 shows schematics of contact sensors.

Load cells floors seem to be designed for clinical purposes. They require a heavy hardware since the floor is divided in tiles of several cells, hence making them a poor choice for large areas. As of today they seem to be better suited for small-scale systems such as connected shoe soles [83]. Binary switches are cheap sensors but have the obvious disadvantage to require a high resolution to perform monitoring tasks, hence posing a hardware issue.



(a) Load cell (b) Binary switch output

Figure 3.2: Contact sensors. Load cells (a) are usually installed under each corner of tile and work as pressure sensors. On the other hand, binary switches require a high resolution to perform well. Figure (b) shows examples of outputs of a foot on low (left) and high (right) resolution systems.

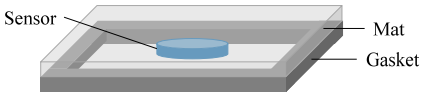


Figure 3.3: Example of a piezoresistive floor system. When a force stresses the material, its resistance decreases.

Piezoresistive. The piezoresistive effect is for a material to have its electric resistance modified depending on the mechanical constraint that it receives. More precisely, the piezoelectric sensor is fed with a current that is constantly measured at the output. When a force is applied to the material, its resistance is lowered and the measured current increases, hence acting as a pressure sensor. For example, Tanaka et al. [173] use several 40 x 40 cm^2 mats, each of them containing a piezoelectric sensor (see Figure 3.3). However, they use a threshold over the mats output, hence processing an array of binary signals (4x4 in their experiment).

Piezoresistive sensors can be adapted to various resolutions, however, like capacitive sensors, they need a constant power supply.

Piezoelectric. A piezoelectric material emits charges when stressed or squeezed. Pressure sensor can be obtained in a variety of ways, using for example crystals such as quartz, or dieletric materials called electrets. For example Suutala and Röning [170] use a polypropylene film coated with metallic electrodes, resulting in a very thin film (less than 100 µm) than can be installed directly between the concrete and whatever flooring [135]. The main difference with piezoresistive material is that when stressed by a constant force, this latter will see its resistance increase as long as the force is present. However in the same situation, when using a piezoelectric material, the output voltage increases then returns to its initial value. Hence, while the piezoresistive material measures a pressure, the piezoelectric material measures pressure *variations*. Figure 3.4 shows a schematic view of a piezoelectric floor sensor.

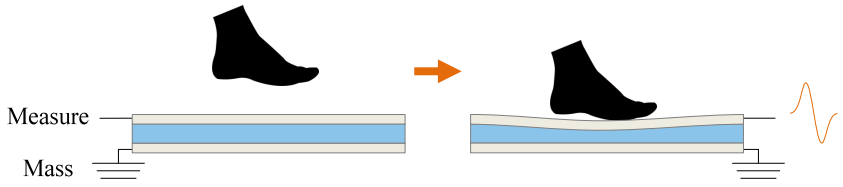


Figure 3.4: Piezoelectric sensor principle. When deformed, the piezoelectric material emits charges, hence a current can be measured in output.

Film sensors are the most interesting in terms of application. They are flexible and since they come as bands, they can be easily manufactured for large scales. Besides, they are insensitive to humidity and do not require power supply. For theses reasons, this technology was adopted. The film is the same as the one proposed by Paajanen et al. [135], however the installation is different. The following section describes the sensor in more details.

1.2 Tarkett's technology

In this subsection we present the floor sensor that is used for the installation in nursing homes. The piezoelectric principle (on which the device is based) is outlined, then the unit is described in more depth.

1.2.1 The piezoelectric sensor

The piezoelectric principle, when simplified, can be explained by the relation $d = \frac{Q}{F}$, with d being the piezoelectric coefficient, Q the charge and F the force received by the material. A piezoelectric material will emit charges under deformation but can also shrink (or expand) when submitted to an electric field. Hence, such a material can fit multiple purposes such as speakers or motors when using its deformation under a electric field, to microphones and sensors when using the electric charge emitted when submitted to a force. Tarkett's system is based on the latter use, and the piezoelectric material acts as a pressure sensor. It is done using an electret, which is a dieletric material in a quasi permanent polarization state. An electret can be obtained by applying an electric field to a polymer. Indeed when a polymer comes as a foam, its cavities can be electrically charged, and the polymer becomes piezoelectric. Tarkett's sensor is composed of several insulating layers of polyethylene terephthalate (PET) and aluminum, and the piezoelectric layer is made of a polypropylene foam (PP) (see Figure 3.5) resulting in a total width of about 0.3 mm.

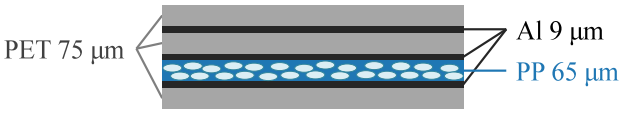
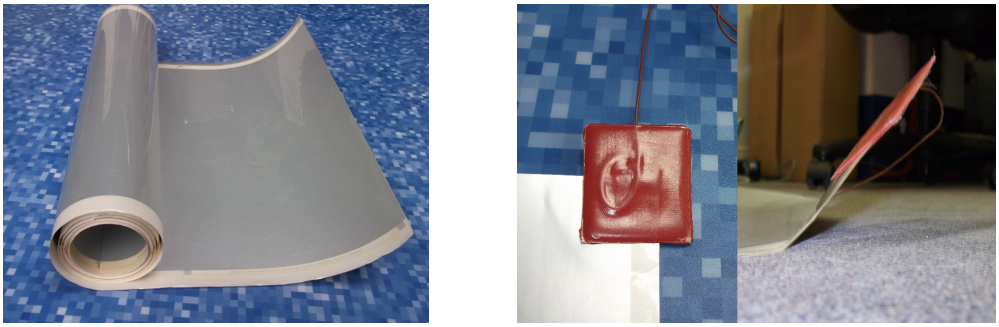


Figure 3.5: Section of Tarkett's piezoelectric sensor. The PP foam is coated with aluminium layers that serve as measurement and mass for the sensor and the PET is used as insulator.

1.2.2 Set up



(a) Roll of sensor (b) Connector

Figure 3.6: Roll of sensor (a) and connector (b).

(a) Sensor set up (b) Room equipped with the fall detection system

Figure 3.7: Pressure sensor installation. The sensor can be installed directly onto the concrete and covered by the flooring (a). Figure (b) shows an example of installation in a room. In this example, there are three areas. The bedroom area is composed of four connected bands that output a single signal. This signal constitutes one of the three inputs that feed the processing unit.

The sensor comes in bands of 60 cm wide and is placed directly under the flooring. The bands are initially rolls of about 100 meters that can be cut every 30 centimeters. Once cut, bands can be connected together without limitation in their number before the signal is recovered by the processing unit. This is done with small flat connectors that link bands together to form an area before passing the resulting signal to the unit. Pictures of the sensor and connectors are given in Figure 3.6, and schematic installation is displayed on Figure 3.7. The processing unit has 8 entry channels, allowing each unit to process a large surface by connecting bands. This allows a great flexibility of the system to any scale while keeping a certain ease of installation.

The processing unit is designed to process the signal from end to end (from basic filtering to event detection) and to communicate with servers (sending signal recordings) and the nursing home (sending information such as activity reports or alarms). When conveyed to the unit, the signal is first passed through an analog charge amplifier that convert the charge signal into voltage, and is then converted into numerical values. The unit is equipped with a 32-bit 500 MHz processor, accompanied with 256 MB of RAM, and 500 MB of local storage.

1.2.3 Limitations

It should be noted that compared to the theoretical piezoelectric principle law, the sensor presents some variability in the piezoelectric coefficient d. Indeed, although d seems to remain constant when put under humidity or high temperature (less than 40° Celsius), it can variate depending on where the impact is located on the floor. A difference of 9% between max and min value of d has been observed in controlled experiments [160]. Besides, when implemented the system can suffer perturbations due to the way sensor, electrodes and upper layers have been installed. These elements that come with the implementation in real conditions can result in alterations to the resulting signals when compared to similar devices used in laboratory conditions [161].

1.3 Signal processing

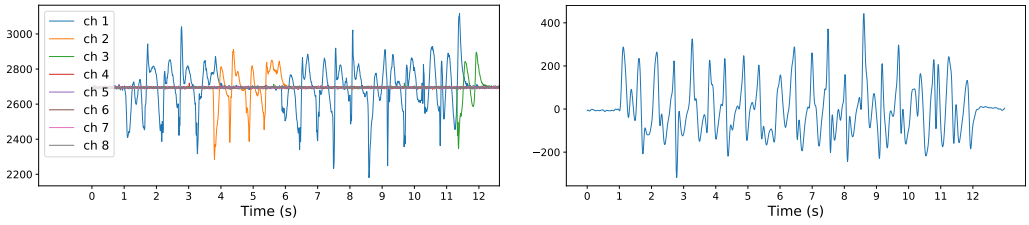
The preprocessing of data is done as follows. Let $C^k(t)$ denotes the signal produced by the channel k ($k \in [1, \ldots, K]$), where $t = 1, \ldots, T$ represent the data points recorded with a frequency of 100 Hz.

1. The linear trend of each channel is removed using a least-squares model.
2. Each channel $C^k(t)$ is filtered with a low-pass Butterworth filter with a 10 Hz cutoff frequency, fifth order, and zero lag. This step aims to limit the amount of electronic noise present in the signal, as the 10 Hz cutoff frequency is the reference in gait related signals [9].
3. Each channel whose signal maximum amplitude is small is then set to zero, as the channel is assumed to only accounts for noise.
4. The resulting signal $\mathbf{s}$ is obtained as the sum of all the channels:

$$\mathbf{s} = -\sum_{k=1}^{K} C^k(t)$$

The minus sign here has no influence on any further processing but comes from internal aspect of the system. Indeed, as the processing unit contains an inverting amplifier, to better reflect the physical aspect of the pressure sensor the final processed signal is inverted back.

Figure 3.8 shows an example of a raw signal and its resulting preprocessing.



(a) Raw signal (b) Preprocessed signal

Figure 3.8: Example of raw (a) and preprocessed (b) signal resulting from a person walking on the equipped pilot site.

2 Experimental data set

2.1 Protocol

A pilot site was equipped with the sensor and a data set was created using 28 volunteers. Following a protocol, people were falling starting from specific positions towards different directions. The non-fall events were made as varied as possible, recording walks (with

one or more persons), some with a cane, movements with a chair, sitting, jumping, running, picking objects. 742 acquisitions were made, including 409 falls (i.e. 55% of the acquisitions). Recorded signals last about 20 seconds in average. All acquisitions were simultaneously video-recorded for labelling.

2.2 Data

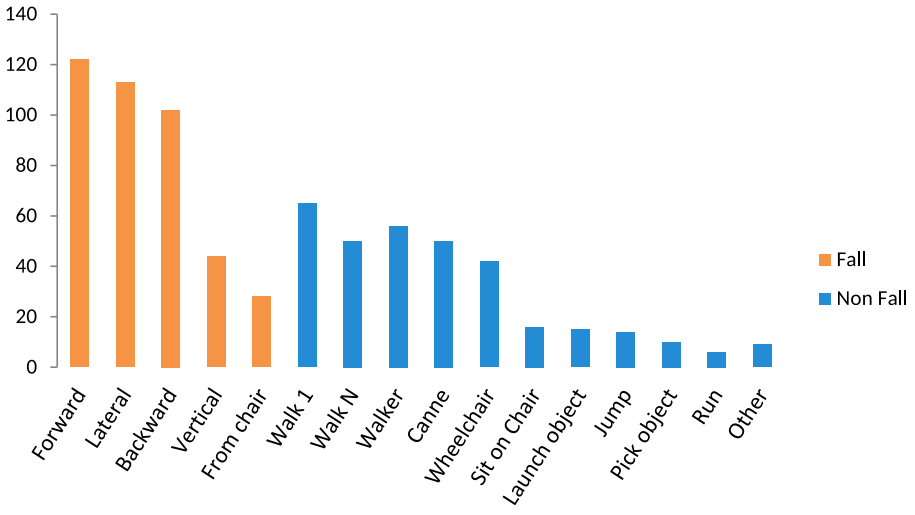


Figure 3.9: Experimental database events

Figure 3.9 shows the variety of events for each class (fall and non-fall). Among fall events, volunteers performed falls from a standing position and falls from a chair. Falls from a chair were all performed in a similar way, however falls from a standing position were done with various beginning and ending positions. Table 3.1 shows the variety of fall events amongst falls that started in standing position.

The signal coming from the processing unit is sampled at $f_s = 100$ Hz. As an example, a fall acquisition signal is shown on Figure 3.10. In our database, we observe that the falls last 1.2 seconds in average (*i.e.* 120 samples at 100 Hz).

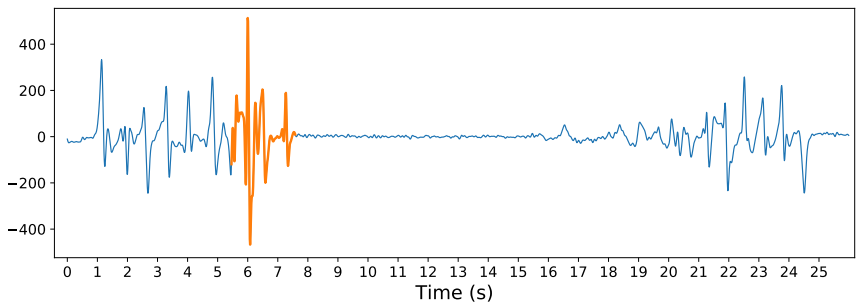


Figure 3.10: Example of a fall acquisition signal. The volunteer arrives onto the smart floor, walks a few steps, falls, stays put for a while, gets up and leave the floor. The orange part is the ground truth of the fall location.

	Beginning posture			
Ending posture	Forward	Lateral	Backward	Vertical
Forward	65	7	0	7
Lateral	30	62	0	5
Backward	27	44	102	32

Table 3.1: Experimental data set: diversity of falls events. The column indicates the beginning posture of the volunteer while the row indicates the ending position.

3 A predictive model for fall detection

This section describes the proposed model for fall detection from floor signals. As previously shown (see Chapter 2), supervised model trained with feature vectors have proven to be efficient when it comes to complex signals. In light of this, the adopted solution is provided by a combination of a statistical feature vector computed over the one-dimensional signal and a random forest algorithm, which is a high-performing model, easy to handle, train, and implement. The model is enhanced with data augmentation and in order to limit the false alarm rate, we use time aggregation of the classifier outputs. We also question the feature space for implementation in the embedded monitoring system.

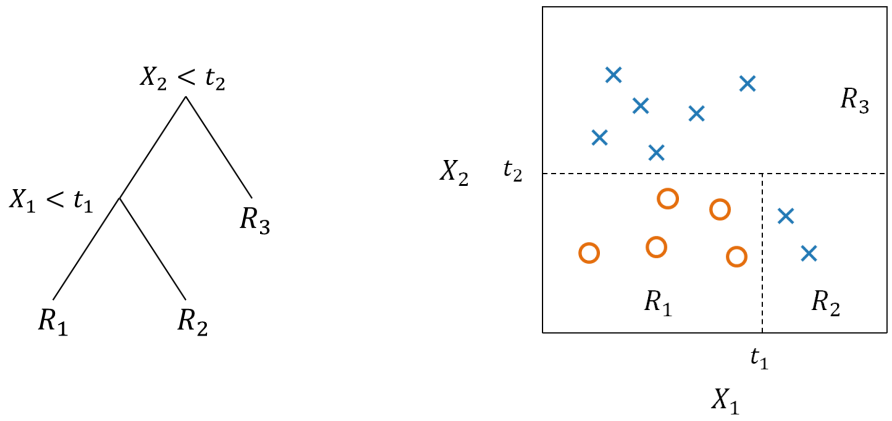
Supervised learning models seek to estimate a function f whose goal is to predict from an unknown input $x \in \mathcal{X}$ a corresponding output $y \in \mathcal{Y}$, $\mathcal{X}$ being the feature space and $\mathcal{Y}$ the label space. For a binary classification over real-value feature vectors we have $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \{-1, 1\}$. The estimation of f is usually done through the minimization of a loss function $\mathcal{L}$ that measures the error between the prediction and its expected value. With a labeled set $\{(x_i, y_i), i = 1, \ldots, n\}$ we can then obtain a model $\hat{f}$ such that $\hat{f} = \arg\min_f \sum_{i=1}^{n} \mathcal{L}(f(x_i), y_i)$. This framework is the most used for regression and classification tasks, however random forests are trained differently.

3.1 The random forest model

A random forest (RF) is a powerful model proposed by Breiman [29] and based on the aggregation of multiple decision trees. It can be used for both regression and classification problems. Due to its good performances and the ease of use and implementation, the RF model became quite popular and is now implemented in various packages. Before defining a RF, we must first describe what a decision tree is.

3.1.1 Definition of a decision tree

Decision trees as they were defined by Breiman et al. [30] do not use a global optimization scheme but rather greedy heuristics all along the training. Given a feature space $\mathcal{X}$ of Q dimensions ($\mathcal{X} = \mathbb{R}^Q$), a decision tree (DT) is the division of $\mathcal{X}$ into J non-overlapping regions $R_1, R_2, ..., R_J$, also called *terminal nodes* or *leaves*. Figure 3.11 displays these two representations of a DT. The most used methodology to build a tree is the CART algorithm (Classification and Regression Tree) which operates as follows [30]. The tree is created by recursive binary splits, meaning that we successively split the feature space into two parts. At each split (or node), the algorithm chooses a feature and a corresponding threshold.



(a) Tree (b) Regions

Figure 3.11: Example of a decision tree (a) and the corresponding regions in a two-dimensional feature space (b) with two classes. Crosses and circles represent the training samples.

This part is done according to the minimization of an impurity measure of the split. For each node n, the algorithm works greedily by considering all features $\{X_q, q = 1 \dots Q\}$ and splitting values τ to maximize the information gain (IG). For a couple (X_q, τ), IG is the difference between the impurity (denoted I) at node n and the weighted impurity of the children nodes (denoted l and r for left and right):

$$\mathrm{IG}(X_q, \tau) = I(n) - \frac{N_l}{N_n} I(l) - \frac{N_r}{N_n} I(r) \ , \tag{3.1}$$

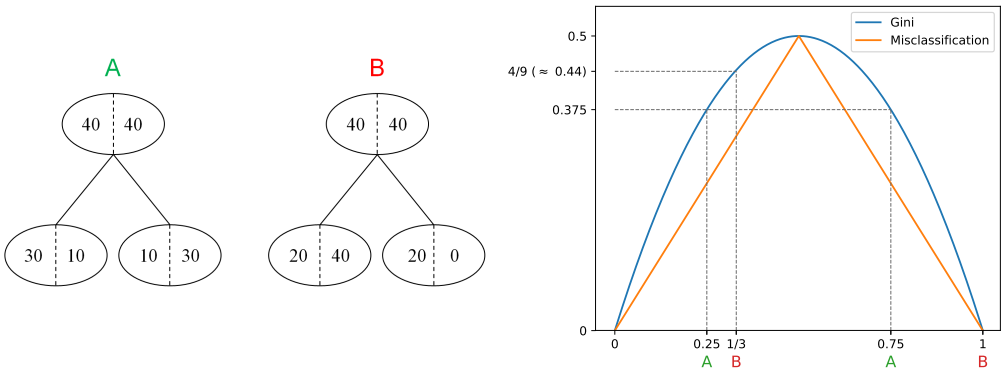
with N_n the number of training sample in the node n, N_l and N_r the number of training samples in left and right child nodes. There are several impurity measures, however the most commonly used is the Gini index, which is defined as follows. Let p_{nk} be the class proportion of the class k in node n, the Gini index is

$$G(n) = \sum_k p_{nk}(1 - p_{nk}) \ . \tag{3.2}$$

Hence defined, the Gini index gives a level of how much the split isolates a class from others. It is preferred to the classical misclassification error (i.e. proportion of misclassified examples in the node) since it tends to make pure nodes. An illustrative example is given in Figure 3.12.

Splits are done repeatedly until a node is pure (i.e. $G(n) = 0$) and it becomes then a *leaf*. One can also specify a maximum depth to reach. The depth of a tree is the longest path between the tree root (i.e. the first node) and the leaves. Once the tree is trained, for every unknown observation x that falls into the region R_j, we predict the output corresponding to the majority class in R_j. The decision function of a DT is then defined as

$$f(x) = \sum_{j=1}^{J} c_j \mathbb{1}(x \in R_j) \ , \tag{3.3}$$



(a) Two possible splits

(b) Gini and misclassification measures as a function of the proportion in one of the classes

Figure 3.12: Example of two possible splits (A and B) in binary classification. We show the possible outcomes (a) and Gini index and misclassification error are displayed for the two-class problem (b). When using the misclassification error, $\mathrm{IG}(A) = \mathrm{IG}(B) = 0.25$. However if we use the Gini index, then $\mathrm{IG}(A) = 0.125$ and $\mathrm{IG}(B) \approx 0.17$. While the misclassification error considers A and B with the same interest, the Gini index makes the algorithm chose the split B. In fact, we remark that the Gini curve, compared to the misclassification error, tends to have values reaching the upper left and right corners. The more the impurity tends to reach these corners, the more it will penalize "impure" nodes.

with

$$c_j = \arg\max_k p_k \ . \tag{3.4}$$

3.1.2 Definition of a random forest

A random forest (RF) is an aggregation of several decision trees with two major principles [29].

First, each tree is grown from a dataset obtained by *bootstrap* of the initial training dataset. Given a set of data, performing a bootstrap on it consists of sampling the same number of elements with replacement. If done several times, say N_T, then one obtains as much training data sets and hence trains N_T decision trees. This process is called *bagging* (for bootstrap aggregation) and aims at reducing the variance of the final prediction function [74].

Second, when growing a tree, each split is done on a random subset of m features rather than the whole pool of features. This aims at decorrelating the trees. The m value is commonly set at $\sqrt{Q}$, where Q is the total number of features [74]. The final decision function is then defined as the majority vote over all trees. As each tree represents a *vote* for a class k, the output of a RF can be seen as an estimate of the probability for x to be of class k. This estimate is defined as:

$$f_k(x) = \frac{1}{N_T} \sum_{i=1}^{N_T} \mathbb{1}(d_i(x) = k) \ , \tag{3.5}$$

where d_i is the decision function of decision tree number i and N_T the total number of trees. The final decision function is then:

$$f(x) = \arg\max_k f_k(x) \ . \tag{3.6}$$

The RF model presents some advantages considering our issues, the first point concerning overfitting. Overfitting happens when a model "sticks" too closely to training data (i.e. it has a high variance) and hence has trouble generalizing. High variance may come from the model complexity (a more complex model leads to higher variance) and the feature space size (the more variables, the higher the variance) [74]. For example, Begg et al. [16] use SVM for gait signals classification and note a significant decline in test performances when adding more features into the model. However, a RF reduces its variance through the averaging of multiple decision trees, but also with the inherent feature selection process which decreases correlation between trees.

Besides, assuming that one has access to a large number of features (without taking computational cost into account), a RF model makes an additional feature selection procedure less necessary. Indeed, as the model chooses the most separating dimensions along the training (subject to the random pool of $\sqrt{Q}$ features), this means that even when given a large amount of features, a RF may still perform well since non-relevant variables are likely to be ignored. In fact it would take a significantly low ratio between relevant and "useless" features to see a deterioration in results [74].

Finally, this type of model is quite easy to use and tune, and once trained, implementation is quite straightforward with a possible parallelization (each tree can be traversed independently of all others), hence making it an interesting choice for industrial purposes.

3.2 Temporal signal into augmented feature vectors

3.2.1 Feature extraction

For the classification problem, we consider a window of size T over the signal, with T previously set to 250 (*i.e.* 2.5 seconds at 100 Hz). Instead of dealing with the signal itself, we chose to represent instances in a feature space. Indeed, the signal itself contains a lot of information (here it would have been represented in a T-dimension space), and part of it is not relevant for the classification. Besides, using each sample as a dimension would lead to both redundancy between dimensions, and great variance within them. Therefore, we compute 29 features over three views of the signal:

- the signal itself
- the first derivative of the signal, as not only the pressure variations but also their speed may matter in discriminating such events
- the absolute value of its Fourier transform, assuming that frequency response between a fall and other events are different

It creates then a 87-dimension feature space. These features are statistical measures over the selected window. They come as simple metrics (e.g. minimum, maximum, average, variance...), to more elaborate (e.g. normalised momentum, Shannon energy...), inspired

from previously cited works. All features are detailed in Appendix A. It should be noted that some of them are by definition computed over the derived signal (one of them allows us to derive n times). Hence, when the input for feature calculation is an already derived signal, it means that the feature in fact computes a measure over the second order (or $n+1$ order) derivative of the original signal.

3.2.2 Data augmentation in time series

As we select a window position on which we calculate the features, we use an additional step in the training process in order to improve our model robustness. This is done in a way that also extends the database. This process can be referred to as *data augmentation*. Data augmentation is used notably to address class imbalance problems with over sampling technique, the most popular being SMOTE [37]. It has also been used to enlarge the training data set for neural networks, for example with image warping in the context of handwriting character classification [200]. However, unlike augmentation techniques that create new data points from real ones, our process consists of selecting more.

Indeed, extracting a window out of a fall acquisition makes us naturally choose the window in which the sub-signal of the fall is *centred*. However, doing so may introduce a bias in the training data hence leading the algorithm to detect centred falls in the window. Therefore, in order to improve robustness of our solution, we select r windows randomly chosen over time, making sure that if we are dealing with a fall acquisition, the fall signal is encompassed in the different windows. By definition, this process adds data points that are not generated from already existing points (as previously described methods do). However it outputs events that are viewed at different times which inherently creates some redundancy, this is why it can be viewed as a data augmentation procedure.

When $r = 1$ then there is no augmentation at all and it is a classical training procedure. Choosing r was done by training and testing our classifier over varying values of r. In our experiments, we also vary the number of features as well to evaluate the influence of r when less features are available. Results are shown in Section 4 (Figure 3.18).

3.3 Macro-decision construction

As said before, our signal is sampled at frequency $f_s = 100$ Hz and we selected a T-sample long window as the instance to be classified. Every $1/f_s = 0.1$ second, the 87 features are computed over the window. The feature vector is passed through the RF giving as an output a number N_T of *votes* (we can consider a tree output as a vote for the class *fall*). Let $N_f(t)$ denote the number of trees voting for the *fall* class at time step t. If divided by the number of trees, $N_f(t)$ is the previously mentioned $f_k(t)$ with k corresponding to the *fall* class (see Equation 3.6). Considering solely this output over time may be unsuccessful in terms of false alarms as a raise of $N_f(t)$ can trigger an alarm, as short as the raise duration may be. For that reason, we use a *buffer* to encompass successive micro-decisions (i.e. $N_f(t)$) into a macro-decision. The results of the RF are gathered over time and we construct our decision as following. We introduce two new parameters: a buffer size B_s and a threshold T_h, with $0 < T_h < 1$. We collect the RF votes during B_s samples, meaning that we have now $B_s \times N_T$ votes at each time step t. Let us denote the function g as

following:

$$g(t) = \frac{\sum_{u=t-B_s+1}^{t} N_f(u)}{B_s \times N_T} \ . \tag{3.7}$$

Then we define our new binary classification function:

$$d(t) = \begin{cases} 1, & \textit{if } g(t) > T_h \\ 0, & \textit{otherwise} \end{cases} \ . \tag{3.8}$$

A remark to be made is that when using a classification model over temporal signals (or any time series), one could build a model with "on-the-fly" decision. In our case, it would mean trigger a fall alarm at any time t such that $\frac{N_f(t)}{N_T}$ exceeds a preset threshold, which is equivalent to the use of a buffer of size 1. Hence, extending this buffer size with corresponding threshold gives us control over the required detection duration and prevents short peaks in N_f (e.g. falling objects or rapid movements) from being detected as falls.

Figure 3.13 shows the responses of a classical model and a model with outputs aggregation. The influence of the parameters B_s and T_h over the classification performance is tested in Section 4 (Figure 3.19).

3.4 Feature selection for operational design

To this day, the algorithm is implemented in an embedded system with a specific architecture, hence bringing limitations in terms of both memory and computing power. A first solution may be to reduce the frequency of the decision. However, time step between each diagnosis was not to be modified, hence bringing another proposition based on feature space reduction. Reducing the number of features decreases both the need for computational power and memory when performing the decision process.

The decision tree algorithm comes with an interesting measure called *variable importance* [29]. Given a variable (or feature) X_q, the importance is defined as the sum of the weighted impurity decreases for all nodes in which X_q is used. For a tree T, it is defined as follows:

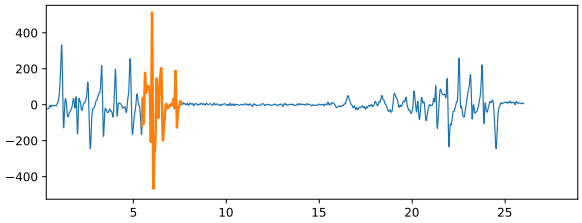
$$I(T, X_q) = \sum_{t \in T} p(t) \Delta i(t) \mathbb{1}(v(t) = X_q) \ ,$$

where $p(t)$ is the proportion of samples reaching the node t, $v(t)$ is the variable used at node t and $\Delta i(t)$ is the decrease of impurity $i(t)$. $\Delta i(t)$ is the difference between the impurity in node t and its two children nodes. Hence defined, the feature importance for a random forest consists of averaging over all trees:

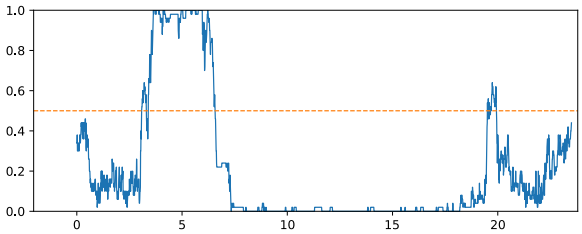
$$I(X_q) = \frac{1}{N_T} \sum_{n=1}^{N_T} I(T_n, X_q) \ ,$$

with N_T the number of trees. Therefore, ranking features according to their importance highlights those that are the most used and efficient for separating our data.

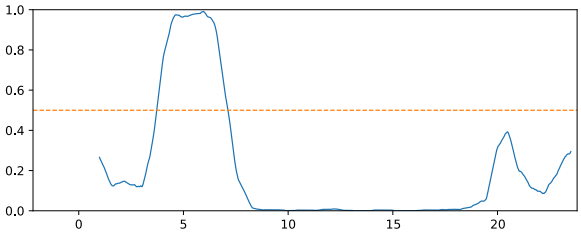
It is then possible to build a selection procedure based on the variable importance. Inspired by the *recursive feature elimination* proposed for SVMs Guyon et al. [73], we use the same scheme with however several trainings to retrieve the importances (steps 1 and



(a) Processed fall signal



(b) Original response of a RF



(c) New response using a buffer

Figure 3.13: Decision responses to a signal in function of time (in seconds). Figure (b) shows a response of a classical RF algorithm to a fall signal (a), while Figure (c) displays the response using a buffer. The orange line is an example of a threshold at 0.5. In this example, the threshold would have detected two falls while the buffer alleviates this problem.

2 in the below description). Indeed, as the RF training mechanism use random subsets of variables to optimize the splits, two trainings with the exact same input data will lead to differences between the final variable importances. Hence, averaging importances over several trainings insures a representative ranking of variables. The variable reduction is done as following:

1. Using the pool of Q features, several trainings are done while recording the variable importances.

2. The average of importances over trainings is computed and we select the variable X_q* such that $I(X_q*)$ is minimum.

3. X_q* is removed from the pool of features and we go back to step 1.

The features importances of a RF model trained with the biggest pool of features are shown on Figure 3.15. Derivative and Fourier transform of the signal seem to be the most responsible of the data classification. It should be noted that the ranked-2 feature (i.e. S-Max-n-derivative) is computed over the raw signal, however it intrinsically uses the derivative

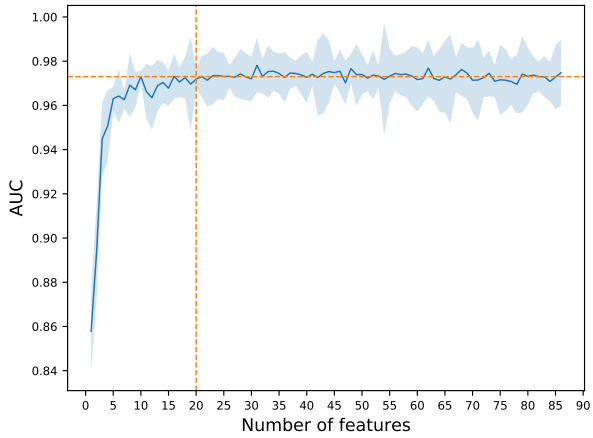


Figure 3.14: Area under ROC curve of a RF model trained with reduced number of features using the *recursive feature elimination* method. The model is trained and tested using k-fold cross validation. It seems that in this context, the first 20 "best" features (orange vertical line) allow to reach the maximum performance. The orange horizontal line is displayed for visual help (the value is arbitrarily chosen).

of order n (see Appendix A), with $n = 3$ here. To observe the influence of features availability on the model performances, RFs were trained with a diminishing pool of features following the selection process and results are shown on Figure 3.14.

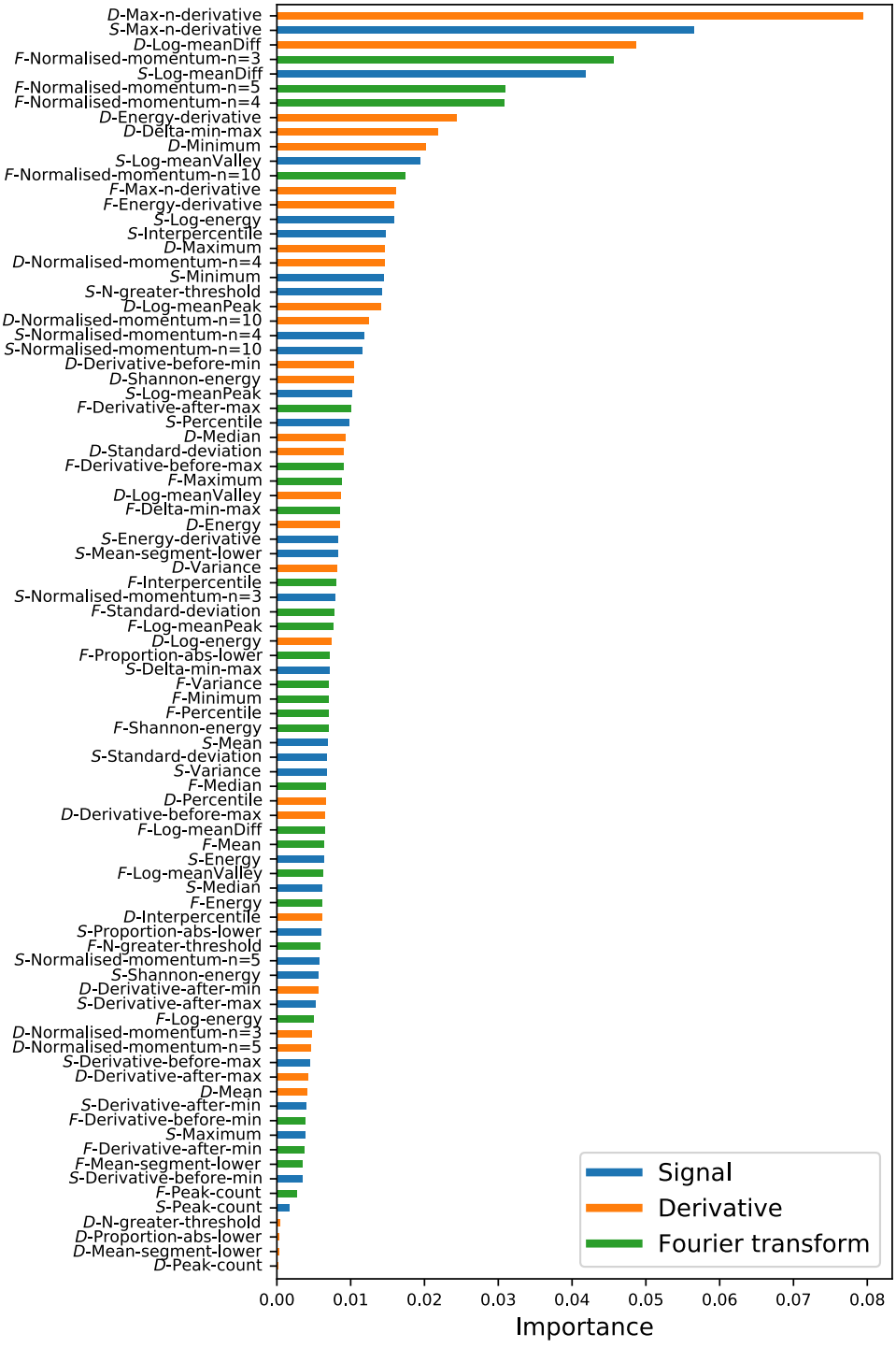


Figure 3.15: Mean feature importance for a RF trained 100 times. Each feature is noted as "$S/D/F$-Name" with S, D, F referring to either the signal, the derivative or the Fourier transform, and Name to one of the 29 described functions (see Appendix A).

4 Performance results

4.1 Evaluation

Let us first briefly describe the performance indicators we use throughout this work. When developing binary classification models, we use various evaluation metrics that are computed from the confusion matrix, which contains the account of tested instances, with four possible outcomes: true positive (TP), true negative (TN), false positive (FP) and false negative (FN). We then can compute the accuracy, true positive rate (TPR) and the false positive rate (FPR) as follows:

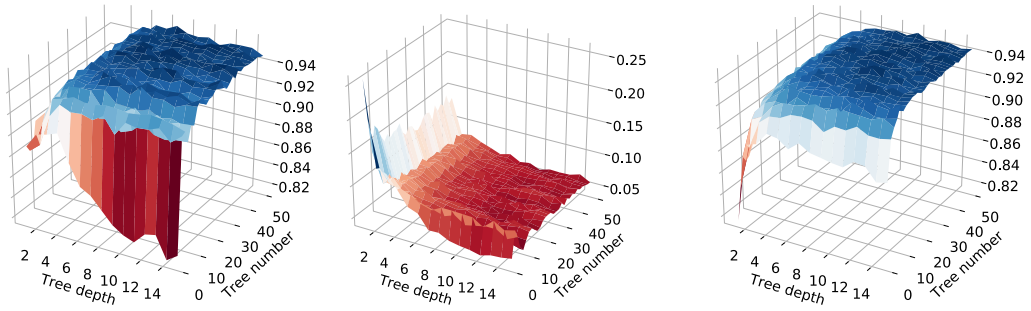
$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{Total predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$
$$\text{TPR} = \frac{\text{True positives}}{\text{Total positives}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{FPR} = \frac{\text{False positives}}{\text{Total negatives}} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

While the accuracy measures the proportion of well classified instances, TPR and FPR give ratios that depends on the type of error that we investigate.

A well used indicator for a binary classifier is the receiver operating characteristic (ROC) curve which is the plot of TPR values in function of FPR values when a prediction threshold is varied. The area under the ROC curve (AUC) is a usable value that allows us to compare different classifying models.

4.2 Parameters investigation

Random forest hyper-parameters. We investigate two parameters: the number of DT in the RF and the maximum tree depth. Using k-folds, RFs are trained for varying number of DT and maximum allowed depth, and we measure the FPR, TPR and AUC. These results are visible at Figure 3.16.



(a) True positive rate (b) False positive rate (c) Area under ROC curve

Figure 3.16: Investigating tree depth and number of trees in a RF. RFs are trained with different number of trees (from 2 to 50) and maximum allowed depth (from 1 to 15) and we computed the mean TPR (a), FPR (b) and AUC (c). In this case it seems that 30 trees and a maximum depth of 10 are sufficient to achieve the maximum performance.

We remark that maximum performance is reached from a maximum allowed depth of 10. In fact, this is the maximum depth at which decision trees will train on this data set. This means that with no limitation on their depth, in average decision trees will have maximum depth of 10. Figure 3.17 illustrates this by displaying the actual depth of DT as function of the maximum allowed depth.

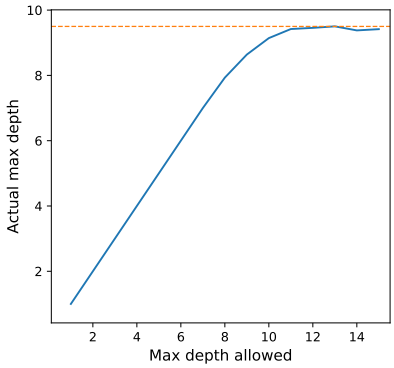
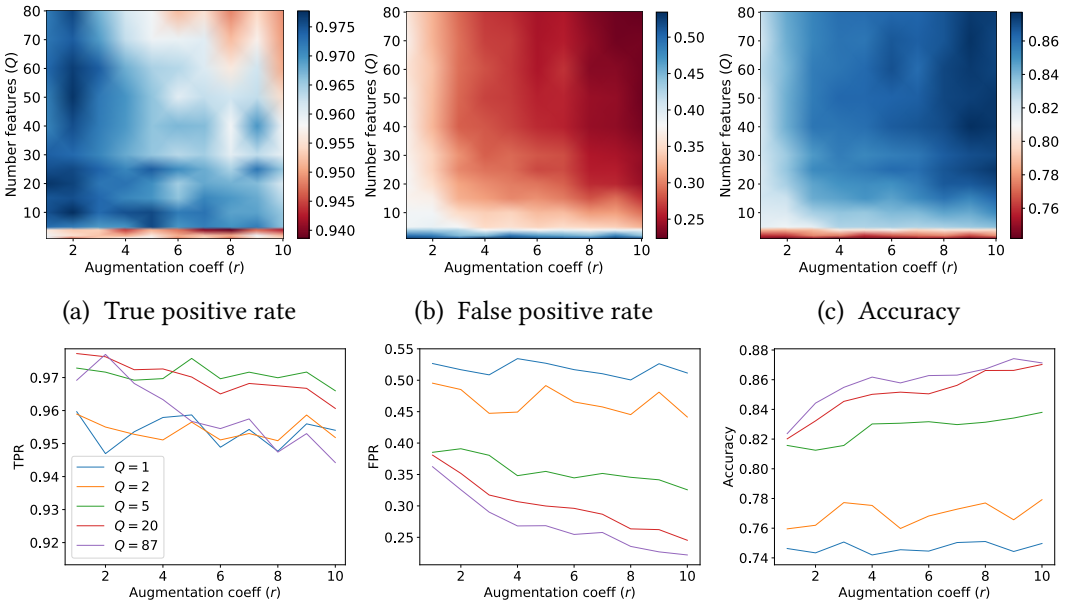


Figure 3.17: Actual mean depth over 50 trees according to maximum allowed depth. Mean depth reaches a limit of 9.5 (orange line).

Data augmentation. RFs are trained over augmented data sets and tested over the regular data set (i.e. not augmented). Tests are performed over signals that are not used for the data augmentation process. We also test the model with restricted number of features. Figure 3.18 shows the results. When using a limited number of features, we observe that augmenting the training data set along the proposed method allows the model to reach performances that were previously obtained (i.e. with a larger feature space). Besides, the main improvement it brings is that the model outputs less false positives. This can mean that augmenting the training set with our proposed method brings significant additional information over non-fall events. A possible explanation to it is that in our experimental data set, despite the variety of falls that were performed, non-fall events may be more diverse in terms of signal pattern and as well in the feature space. Hence, augmenting the *non-fall* class rather than the *fall* class may be more effective in the resulting performances.

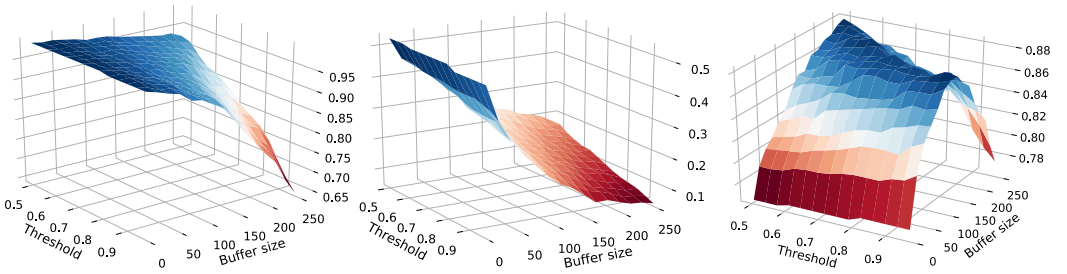
Macro decision. The time aggregation parameters were also tested. Using k-folds, RFs were trained and tested with varying B_s and T_h. Figure 3.19 shows the results. A first observation to be made is that we directly see the benefit of using a buffer of size greater than one over our signals. Secondly, as expected, a low threshold or a high buffer size decreases TPR and FPR. Indeed, when considering a fall event, the higher the buffer, the more the macro-decision encompasses negative (i.e. *non-fall*) micro-decisions, hence lowering the output of $g(t)$. There is hence a trade-off between B_s and T_h. In fact we can "see" this trade-off when looking at the accuracy of the model – it seems to reach its highest values on an affine relation between the two parameters (see Figure 3.20). Finally, we may find various couples of (B_s, T_h) that result with nearly the same TPR (or FPR) but a lower FPR (or a higher TPR). This means that the buffer may bring improvement to the model performances. This part is addressed in more details in next section.



(a) True positive rate (b) False positive rate (c) Accuracy

(d) True positive rate (e) False positive rate (f) Accuracy

Figure 3.18: Data augmentation. We show TPR, FPR and accuracy over two varying parameters: the augmentation coefficient r (ranging from 1 to 10) and the number of features Q (ranging from 2 to 87). Top row figures (a, b and c) show the global behaviour of these scores while the second row (d, e and f) shows an insight over five selected number of features. When r increases, the TPR seems to be either stable or slightly decreasing (for high number of features). However the FPR is significantly decreasing when training data is augmented, hence increasing the accuracy of the model.



(a) True positive rate (b) False positive rate (c) Accuracy

Figure 3.19: Macro-decision. We display TPR (a), FPR (b) and Accuracy (c) on a grid of varied B_s (5 to 250) and T_h (0.5 to 1). Increasing the buffer size or the threshold lowers both TPR and FPR. Hence, there is a trade-off between detection rate and false alarm rate that can be controlled by those two parameters. When looking at the accuracy, we find that the best values are obtained with what seem to be an affine relation between B_s and T_h.

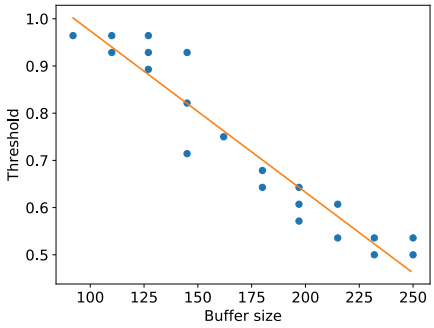


Figure 3.20: Linear fit over maximum accuracy points for (B_s, T_h). These points are the 10% with highest accuracy. On this set of points, the mean $\mathrm{Accuracy}$ is 87.8% ($\pm$ 0.2), mean TPR is 92.4% ($\pm$ 1.1) and mean FPR is 17.9% ($\pm$ 1.6).

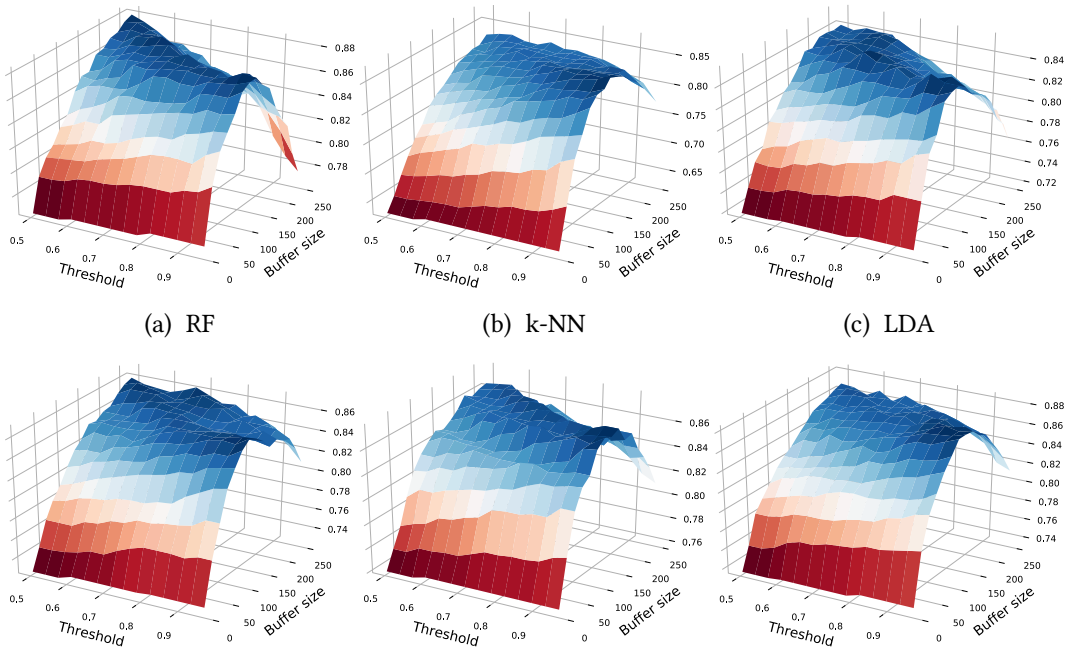
4.3 Comparison and final results.

The RF algorithm was compared to other state-of-the-art methods. The classifiers are: Logistic Regression (LR), Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), k-Nearest Neighbours (k-NN) and Multi-Layer Perceptron (MLP). The SVM was used with a Gaussian kernel, the k-NN with k=5, and the MLP with two hidden layers of respectively 5 and 2 neurons. The RF was set with 50 trees of full depth. Classifiers were trained and tested using k-fold cross-validation and feature scaling, and we used various values of B_s and T_h. Figure 3.21 shows the results of time aggregation over all models, including RF, with varying buffer size. Figure 3.22 shows detailed results for a fixed buffer size.

Macro-decision function reveals to be useful for any model since it is designed to address classification over temporal signals. The maximum scores for each model over the grid of B_s and T_h are given in the left side of Table 3.2.

We observe that while all methods have close results, parametric statistical methods (LR, LDA, k-NN) perform slightly worse than non-parametric methods that contain variable selection (RF), feature space modification (SVM) or feature construction (MLP). To show in more details the result of time aggregation of classifier outputs, we select for each method the set of (TPR, FPR) such that $\mathrm{FPR} < 10\%$ and give the maximum and minimum TPR over the set (see right side of Table 3.2). Results show that when tuning properly B_s and T_h, one can maintain a reasonable FPR while improving the TPR. Hence, this combination proved to be useful for eliminating false alarms that would have been physically too short to be falls.

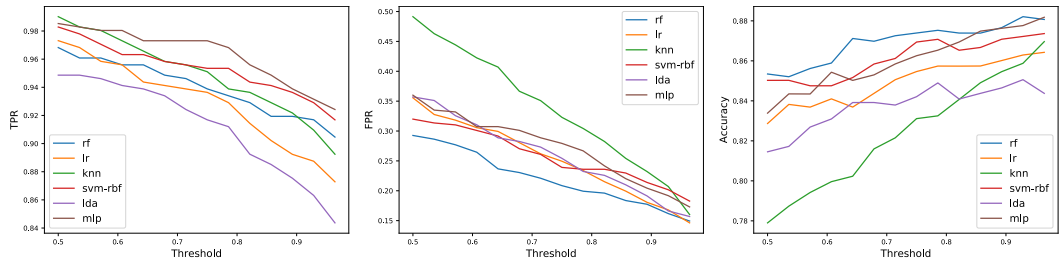
Figure 3.23 shows example of well classified and ill classified signals and the "real-time" response of the model (compared with the unprocessed RF output). The first examples illustrate the good accuracy of the model while signals are a bit challenging. However, as shown, the ideal fall event is in fact one of the possible fall type that can happen. Indeed, since they have various causes [112], falls represent a varied set of events, from the brutal fall (caused for example by slipping) to the "softer" fall (caused by for example by muscle weakness) where the subject tries to cushion the fall. As a consequence, others activity of daily living events may look like falls. This illustrates the ambiguity that can happen when observing signals with the naked eye, thus showing the non-easy task that fall detection



(a) RF (b) k-NN (c) LDA

(d) LR (e) SVM (f) MLP

Figure 3.21: Fall classification performances of various state-of-the-art models using data augmentation and time aggregation. The displayed score is the accuracy. Parameter r is set to 5, Q is set to 20. Parameters B_s is ranging from 5 to 250 and T_h from 0.5 to 1. We observe that all models have same behaviour to time aggregation of their outputs, but the highest accuracy is obtained with the RF. It is reached with $B_s = 127$ and $T_h = 0.93$. With this parameters, RF scores are: Accuracy = 88.2%, TPR = 91.7%, FPR = 16.2%.

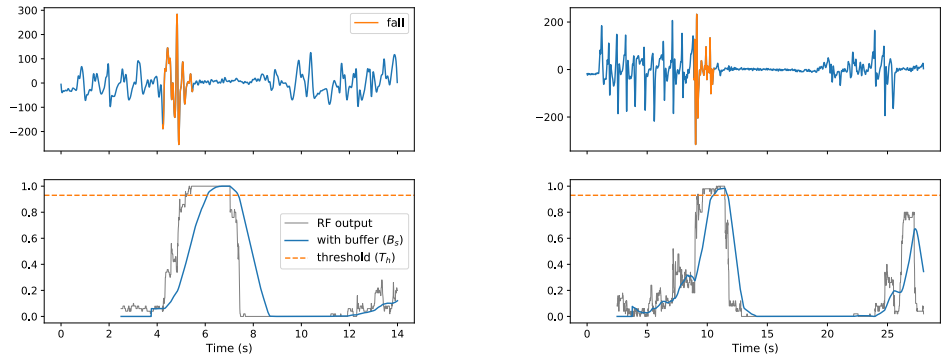


(a) True positive rate (b) False positive rate (c) Accuracy

Figure 3.22: Comparing state-of-the-art algorithms with TPR (a), FPR (b) and Accuracy (c). Models are trained and tested with k-folds, using $r = 5$. Tests are done with $B_s = 127$ and varying T_h (0.5 to 1). RF can outperform other models in accuracy due to a significantly lower FPR.

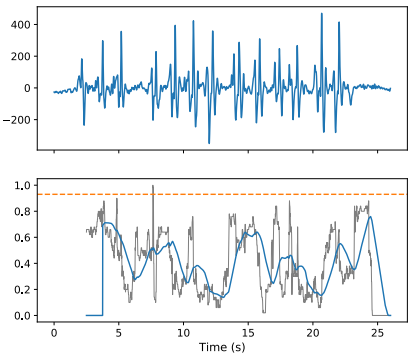
can be.

To conclude, the algorithm designed for this application gives very good results on our data considering standard supervised methods. Besides, it comes with a certain ease of implementation and to some extent, a rather good interpretation of the outputs (which is limited

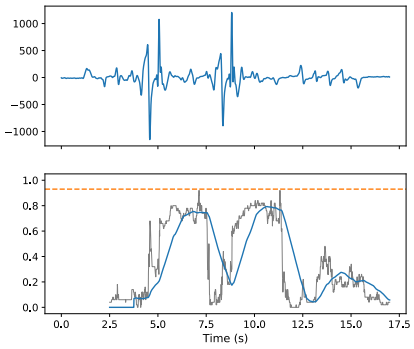


(a) Fall – *true positive*

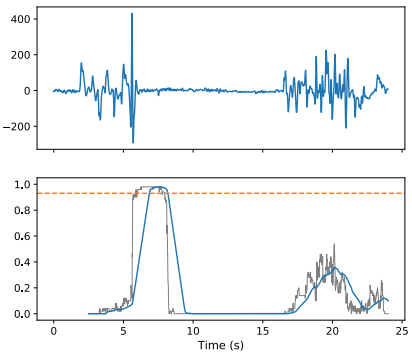
(b) Fall – *true positive*



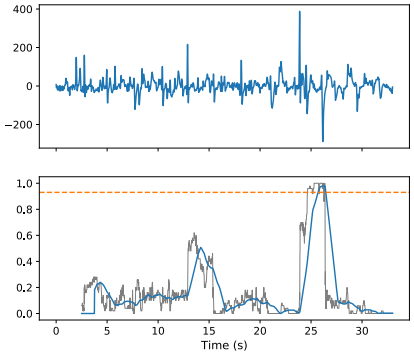
(c) Walk with a cane – *true negative*



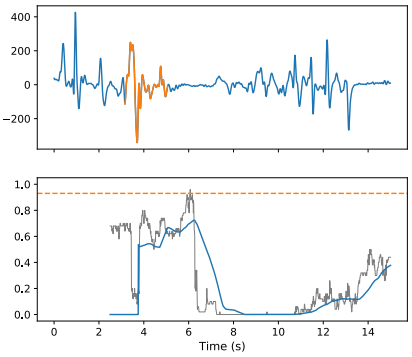
(d) Jumps – *true negative*



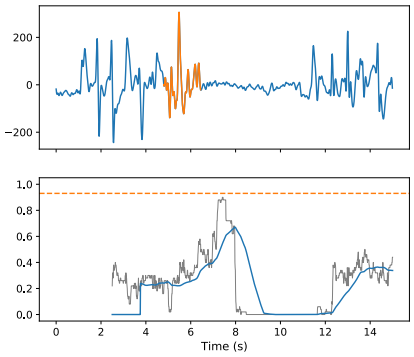
(e) Sitting on a chair – *false positive*



(f) Walk (two persons) – *false positive*



(g) Fall – *false negative*



(h) Fall – *false negative*

Figure 3.23: Examples of events classification over the experimental fall data set.

Model	Accuracy	TPR	FPR	$\mathrm{TPR}_{\mathrm{min}}^{\mathrm{FPR<10}}$	$\mathrm{TPR}_{\mathrm{max}}^{\mathrm{FPR<10}}$
			$r = 5,\ Q = 20$		
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	$\mathbf{16.0 \pm 4.7}$	63.1 ± 4.2	83.1 ± 2.5
SVM	87.6 ± 3.2	90.0 ± 4.5	$\mathbf{15.5 \pm 6.8}$	69.2 ± 2.1	82.9 ± 3.2
MLP	$\mathbf{88.2 \pm 1.5}$	$\mathbf{92.4 \pm 1.2}$	17.3 ± 4.1	71.4 ± 4.5	85.1 ± 2.1
RF	$\mathbf{88.2 \pm 1.5}$	$\mathbf{91.7 \pm 3.5}$	16.2 ± 6.2	63.8 ± 6.8	84.3 ± 7.9
			$r = 10,\ Q = 30$		
LR	87.5 ± 2.5	93.4 ± 2.3	20.0 ± 4.2	66.5 ± 2.4	83.1 ± 1.9
LDA	86.4 ± 1.3	89.5 ± 2.3	17.5 ± 2.6	56.7 ± 4.2	76.8 ± 3.7
k-NN	87.8 ± 4.1	90.7 ± 4.3	16.0 ± 4.7	63.8 ± 5.3	80.7 ± 4.5
SVM	88.3 ± 3.4	88.0 ± 4.2	$\mathbf{11.3 \pm 4.0}$	68.7 ± 5.7	85.3 ± 4.7
MLP	$\mathbf{89.8 \pm 1.8}$	$\mathbf{95.4 \pm 1.2}$	17.2 ± 3.6	72.1 ± 4.8	88.3 ± 1.6
RF	$\mathbf{89.9 \pm 2.5}$	$\mathbf{93.6 \pm 2.5}$	$\mathbf{15.0 \pm 3.0}$	62.6 ± 3.8	85.1 ± 2.9

Table 3.2: Performance results of state-of-the-art models over varying B_s (1 to 250) and T_h (0.5 to 1). Left side: highest $\mathrm{Accuracy}$ over the whole set parameters, and its corresponding TPR and FPR. For each score type, the two best models are stressed with bold font. Right side: minimum and maximum TPR for FPR lower than 10%. We note that maximum TPRs are significantly greater than their corresponding minimum. We also tested varying r and Q and display two configurations (upper part and lower part of the Table). Results show a slight improvement between the two configurations.

when regarding MLPs). To finish, the feature reduction analysis allowed operational team to make the algorithm *fit* into the embedded system. Our algorithm is embedded in a system with 16 kB RAM, 256 kB program memory and a CPU speed of 40 MIPS (million instruction per second).

5 Conclusion

This chapter presented a fall detection system based on a pressure floor sensor. It comes with many advantages when regarding fall detection requirements. From this system an experimental data set could be extracted in a controlled environment, and this was done trying to encompass a large variety of events. Motivated by operational grounds (hardware limitations), we proposed a feature space reduction procedure that may be compensated by a data augmentation method, this latter aiming at making the model more robust and performing. We also put up a macro-decision framework that reduces the number of false positives, and relies on two parameters that can be easily tuned. In light of these results, a model was implemented in the embedded processing unit. This model relies on all the presented facets of the proposed solution, i.e. , the feature extraction procedure, the proposed data augmentation scheme and the time aggregation of the RF outputs. The feature space used in the implemented model was finally reduced following the proposed procedure.

There are however some points that need to be noted. First, as a remark, in the context

of real-time embedded algorithms, feature selection regarding computational resources (and not only their separability power) may be useful. Though we did not explored it in this thesis, *budget learning* is precisely addressing this subject and may be of interest to the reader. Second, this model relies on an experimental data set, hence bringing out some concerns. Indeed, real-world data may differ from the experimental one. It is highly possible in this case that the original algorithm performs poorly when used in its final environment. Besides, as falls are rare events, adding data to the training process raises the question of dealing with an imbalanced data set.

4

Transfer learning on decision tree

Contents

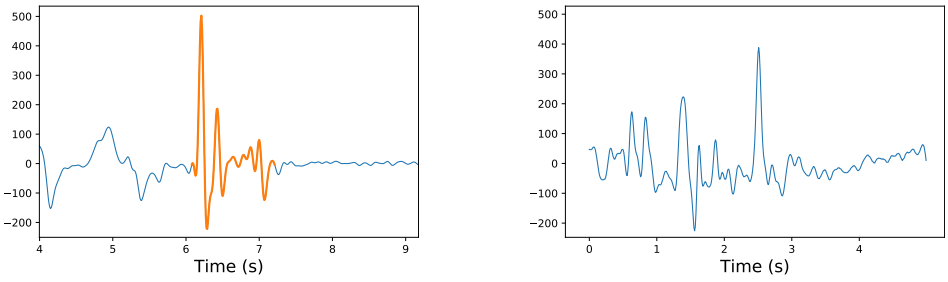
Abstract

This chapter presents work on model-based transfer learning adapted to imbalanced data. From existing transfer methods over decision trees, we express a risk in the case of class imbalance, and propose several extensions that address this problem. Performances are evaluated with several data sets, including synthetic data. These latter are generated so that we control at the same time the source-to-target transformation and the imbalance level. Finally, models are evaluated with fall data and public data sets. Results show the benefits of our approach and suggest several perspectives for further research. Part of this work has been published in Minvielle et al. [122].

1 Introduction

1.1 Training set vs. operational data

Previous chapter presented a fall detection system trained on data that was acquired with a group of volunteers performing various activities on a pilot site, which is a room installation in the company's facilities. However, signals taken from a real-world application, in our case nursing homes, may differ from the ones used for training the learning models. First, users are very different. While volunteers are aged from 25 to 45 years old, the targeted users are elderly. Hence, they behave differently in their various daily activities and most of all their falls are very likely to be different from younger persons. They might also have weights ranging differently, whether being lighter or heavier. Besides, when willingly performed, a fall cannot be as representative as a real one for the simple reason that the risk makes the volunteer behave differently. This situation tends to result in a dissimilarity between the data used for the training, and the "real" data. As an illustrative example, Figure 4.1 shows signals of experimental and real-world falls recorded with similar sensing systems.



(a) Faked fall (b) Real fall

Figure 4.1: Fall signals from experimental (a) and real-life (b) data sets. While the faked fall signal (a) seems to be easily recognizable, the real one (b) is more challenging.

This dissimilarity in the original representation of the signal is likely to have repercussions in the feature space. As an illustrative example, Figure 4.2 compares boxplots of experimental and real-life data sets along the most important features given in previous chapter, and Figure 4.3 displays data projection with Principal Component Analysis. We observe from Figure 4.2 that on most dimensions, data points have undergone transformations,

either drifted along the dimension or spanned and shrinked, which results in complex transformation when observing principal components projection (Figure 4.3).

Besides, we note that since experimental fall data was obtained in a controlled environment, it allowed to build a balanced set. That is, there are nearly as much fall events than non-fall ones, the data set containing 45% falls. This situation is quite ideal for learning supervised classifiers, however real data is a lot more challenging to acquire. Fall detection systems based on the experimental data set were installed in nursing homes and data was labelled with either confirmation of caregivers or expert analysis of signals. Real falls are both rare and hard-to-label events for practical reasons, whereas activity of daily living events are easy to record.

For all the aforementioned reasons, the resulting real fall data set presents dissimilarities with the experimental set and it appears to be largely imbalanced.

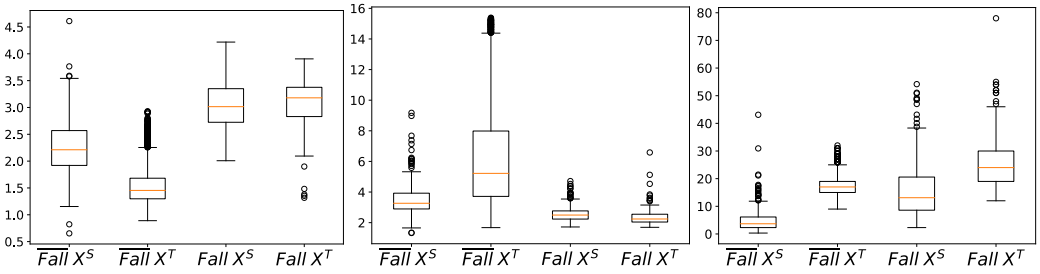


Figure 4.2: Boxplots of experimental and real-world fall data sets. Data is displayed along the first 3 features according to our feature selection procedure over a RF trained with the experimental data set (see Figure 3.15). Experimental data is referred to as X^S (source) and real-world data as X^T (target). The box extends from the lower to upper quartile values of the data (Q_1 and Q_3), with a line at the median value. Minimum and maximum values are set to $Q_1 - 1.5(Q_3 - Q_1)$ and $Q_3 + 1.5(Q_3 - Q_1)$ and plotted as the tips of the whiskers, while the rest (dots) is considered outliers. For source and target we display the boxes for both classes *Fall* and *Non-fall* ($\overline{Fall}$ in the legend).

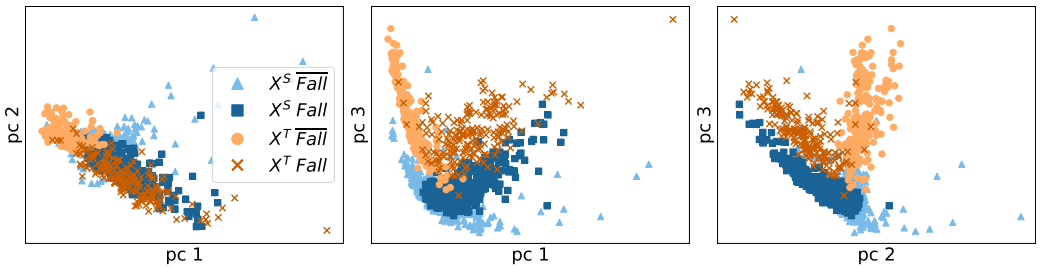
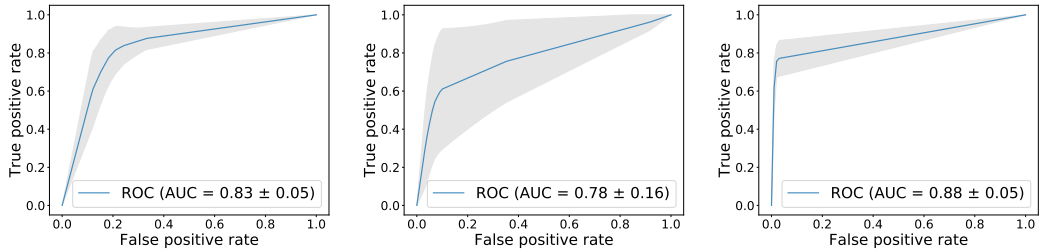


Figure 4.3: Principal component analysis (PCA) is run over a data set made of experimental (source X^S) and real-world (target X^T) fall data samples. For visualization purposes, subsets of data are shown since they are a lot more numerous in the original data sets. Data is projected onto the three first eigen vectors and we display two-dimensional views.

1.2 Knowledge from different domains

This situation is actually a known issue in machine learning. Indeed, in the context of classical supervised learning, one usually assumes that training data and data on which the model is intended to perform its task come from the same distribution and share the same feature space. However, especially in industrial applications, it often happens to be a strong assumption, and the performance of a trained model tends to degrade on real-world data. In order to cope with this problem, a possibility is to collect supplementary real-world data to re-train the model. Nevertheless, this may be difficult and/or expensive, thus leading to few available data on which re-training a model would give poor results.

There is however a set of methods referred to as *transfer learning* that aims at building a model using knowledge from different domains. This growing field of interest covers methods that adapt data, features or models coming from the training domain – referred to as *source domain*, to the operational domain – referred to as *target domain* [190]. As a motivating example, Figure 4.4 shows a model trained on source and target data (which are here our experimental and real-world fall data sets). This is the typical framework that illustrates the need for transfer methods. While the source model (*Source*) obtains fairly good results on source data, it fails to give similar performance on target data. When the model is trained with target data instead (*Target*), it outperforms *Source* but suffers from lack of data for a efficient training. There is then a significant "gap" between performances on target data and performances on source data. Transfer methods aim at reducing this gap.



(a) *Source* tested on source (b) *Source* tested on target (c) *Target* tested on target

Figure 4.4: Transfer learning motivation with ROC curves (and corresponding AUC) of two reference models. A decision tree model, denoted *Source*, is trained with source data. This model is then tested over source data (a) and target data (b). Another decision tree model, denoted *Target* is trained with target data and also tested with target data (c). Training and testing procedures were performed using k-fold cross validation with $k = 10$. Note that the feature spaces are the same for source and target data, i.e. , experimental signals and real signals went through the same feature extraction procedure.

Among changes that can occur between source and target distributions is the class imbalance, i.e. when classification labels are not represented equally in the data set [77]. Indeed, when the source data set is acquired in optimal conditions, either by synthetic procedure or data collection in a very controlled environment, it usually results in a balanced data set. On the other hand, target data is likely to be collected as it comes and its labelling may be costly, hence resulting in an imbalanced data set. This may be the case for example in fraud detection, spam classification, and obviously fall detection in time series.

In fact, Tarkett's experimental fall data set is well balanced (55% falls) while the data set made of events collected in nursing homes is highly imbalanced (7% falls). Unfortunately in all these cases, the classification task focuses on a specific label that is simultaneously valuable and rare.

Recently, Segev et al. [159] proposed two novel transfer learning methods for decision tree. However, both approaches have not been designed nor tested in the context of class imbalance problems. In fact, they present some limitations when a class is scarce in the target domain. After a review of related works, we describe these two seminal transfer methods. Then, limitations to these models are given and we propose several adaptations to them in the context of imbalanced data. These variants are tested on synthetic and real data sets, including Tarkett's fall data. Finally, for the sake of reproducibility, in a joint work with Mounir Atiq and Sergio Peignier, the data generator [140] and all algorithms (original methods and our variants) [12] are made available (Python language).

2 Related works

2.1 Transfer learning

Transfer learning has been a growing field of interest over the past few years. In fact, methods that fall into that field were not initially called that way (sample selection bias, covariate shift, domain adaptation...) and existed before they were grouped into this field, hence making a history hard to achieve. The term *transfer learning* appears in the 2000s with an increase of approaches, and a first comprehensive survey is proposed in 2010 [138], followed later by equally exhaustive updates that encompass the large variety of approaches [190, 214].

Before giving a definition of transfer learning, we need first to define a domain and a task.

Definition 4.1 (Domain). *A domain $\mathcal{D}$ is defined by two parts, a feature space denoted $\mathcal{X}$ and a marginal probability distribution $P(X)$ where $X \in \mathcal{X}$. We denote $\mathcal{D} = \{\mathcal{X}, P(X)\}$.*

Definition 4.2 (Task). *A task $\mathcal{T}$ is defined by two parts, a label space denoted $\mathcal{Y}$ and a prediction function f learnt from samples $\{(x_i, y_i), i = 1...n\}$ where $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. We denote $\mathcal{T} = \{\mathcal{Y}, f\}$.*

We are now able to define the general framework of transfer learning.

Definition 4.3 (Transfer learning). *Let $\mathcal{D}^S = \{\mathcal{X}^S, P^S(X^S)\}$ be a source domain, $\mathcal{T}^S = \{\mathcal{Y}^S, f^S\}$ a source task, $\mathcal{D}^T = \{\mathcal{X}^T, P^T(X^T)\}$ a target domain, and $\mathcal{T}^T = \{\mathcal{Y}^T, f^T\}$ a target task. In the case where $\mathcal{X}^S \neq \mathcal{X}^T$ or $\mathcal{T}^S \neq \mathcal{T}^T$, transfer learning uses the knowledge from $\mathcal{D}^S$, $\mathcal{T}^S$ and $\mathcal{D}^T$ to improve the task $\mathcal{T}^T$.*

In other words, transfer learning aims at building a model that benefits from knowledge of different domains. There are two main taxonomies of transfer learning approaches. The first one, defined by Pan and Yang [138], is based on the availability of labeled data in the source or target domain. When labeled data is available in the target domain, it is referred to as *inductive transfer*. Otherwise (i.e. when no labels are available in target), it is called *transductive transfer*. A second taxonomy is proposed by Weiss et al. [190] where authors use a simpler categorization based on the equality (or not) between feature spaces,

namely *homogeneous transfer learning* when $\mathcal{X}^S = \mathcal{X}^T$ and *heterogeneous transfer learning* otherwise (the interested reader can refer to Day and Khoshgoftaar [47] for a review over heterogeneous transfer). Although different, those taxonomies are compatible and we can label all methods according to both. In the literature, approaches that address the transfer learning problem are usually divided in three categories, whether it focuses on the training instances, the features or the model itself [1].

In the case of our fall detection problematic, both experimental and real-word data sets share the same feature space ($\mathcal{X}^S = \mathcal{X}^T$), hence placing ourselves in the homogeneous transfer learning framework. Note that label spaces are also equal ($\mathcal{Y}^S = \mathcal{Y}^T$). Besides, we have access to labelled instances from the target domain. As we aim to take advantage of these labels, we are in an inductive transfer learning framework. We shortly describe the three main categories aforementioned, restricting ourselves to the *homogeneous transfer* framework. Note that methods used for transductive transfer (no labels available in target) are also applicable to the inductive case.

Transfer with instances. *Instance-based* techniques basically rely on reweighting methods, that is, using weights over training instances so that the discrepancy between the two domains is reduced. These tasks are originally referred to as *sample selection bias* or *covariate shift*, and target instances are usually unlabeled. The general idea is to estimate weights as the ratios between marginals of the two domains, i.e. $w(x_i^S) = P^T(x_i^S)/P^S(x_i^S)$. The weights $\hat{w}(x_i^S)$ are then used in the learning objective function to retrain the final model. Various methods propose solutions to this estimation problem. For example, Huang et al. [84] propose to estimate these weights with Kernel Mean Matching, which consists on minimizing the distance between the two domains in a reproducing kernel Hilbert space, with source instances being weighted.

Weights can also be adjusted iteratively. For example, Dai et al. [44] propose an extension of the original AdaBoost algorithm [59]. Considering that some part of the source data is usable because similar to target data, authors put up a framework in which weak learners are trained over the mixed data (source and target) and tested over target. As each weak learner is tested, instance weights are updated according to their predicted value. Hence source instances that are less similar to target have their weight decreased. Since some target data need to be labeled, this methods falls in the field of inductive transfer learning. Most instance-based methods use the covariate shift hypothesis, that is, $P^S(X) \neq P^T(X)$ and $P^S(Y|X) = P^T(Y|X)$ which is a strong assumption since the target domain, even if similar to source, is likely to have a different conditional distributions as well as a different marginal distributions.

Transfer with features. *Feature-based* methods transform feature space into new representations where source and target are close. These methods usually consider the hypothesis of transductive transfer. Often referred to as *domain adaptation*, there have been different approaches in this field. For example, Daumé III [45] perform *feature augmentation*. Authors put up a feature space of dimension three times bigger than the original where the original vectors are stacked with 0-value vectors placed differently depending if the instance comes from source or target. This redundant method allows the model to

[1] Benchmark codes have been proposed by Zhuang [212] over various transfer learning techniques (with documentation [213]) and domain adaptation codes have been proposed by Zhao [211].

use shared (or not) features for the intended task.

Other methods use *feature alignment*, such as Fernando et al. [58] who propose to extract subspaces from source and target using principal component analysis, and then use a linear function to project source data onto the target subspace.

More recently, works using optimal transport have been proposed. The main idea is to formulate domain adaptation as a transport problem between source and target distributions [42]. Using a relaxation of the original transport problem, authors are able to find a *transportation plan* T between source and target that preserves conditional distributions ($P^S(y|x) = P^T(y|T(x))$) and use it to transport labeled source samples to finally retrain a model.

There have also been works over theoretical guarantees in domain adaptation, in particular on the question of bounding the risk over the target domain. This risk is usually bounded by the sum of the risk over the source domain, a divergence measure between source and target domain, and a term that measures the quality of the hypothesis space [18, 116]. More recent studies propose bounds adapted to the PAC-Bayesian theory [66]. Feature-based methods usually make no use of labelled target data, and instead try to find the best representation for labelled source instances and retrain from there.

Transfer on model. The last type of approach makes use of the model trained on source to learn a new model over target data. This set of methods is also known as *parameter-based* or *model-based* transfer learning, and generally require labeled data in the target distribution (inductive transfer). There are different strategies that tackle this problem. A well-known idea is to learn the target model while including in the objective function a pre-learnt model over source, so that knowledge from source domain can be transferred to the target model. For example, Yang et al. [201] proposed the adaptive-SVM which uses target data to regularize a SVM model already trained on source data. For this purpose, authors add a *delta* function to the previously trained decision function and solve the corresponding minimization problem along target data. In the optimization procedure, the source function is left untouched while the delta function is learnt. As a consequence, the final prediction function is close to the original one while fitting the new data. Since the objective function can be easily tuned, there are several variants of this framework in the literature [214]. Instead of using the the whole source decision function in the target function, some "subtler" methods insure the proximity between source and target models with direct use of the models' parameters. For example, Tommasi et al. [176] include in the target objective function the ℓ_2 distance between source and target model parameters. The learning process consists then of joint minimization of this distance and the error of the target model along labelled target data.

Usually, model-based methods require only target data, which is why model-based transfer learning is particularly useful when source data is not available. This may be the case with privacy issues or data storage limitations.

Negative transfer. Although transfer learning can be very effective with different but related domains, if source and target domains do not share enough similarities, the new model can be badly impacted. In this case, models trained with only source data or only with target data may outperform the one derived from the transfer procedure. This sit-

uation is referred to as *negative transfer*. Hence, when performing inductive transfer, it is highly recommended to compare any performance to models trained with source data only, and target data only.

Transfer for event detection in nursing homes. In the Tarkett's case, the main motivation is to build a model that can adapt to real-world data but also, when possible, to more precise situations such as a specific nursing home or even individual rooms. This process was decided to be done without any access to the experimental data (source), but rather with a fixed model that is known to perform well on source data and newly labeled data from the real world. Since the adopted model is a random forest, this motivated the use for decision trees transfer methods.

This chapter focuses on two model-based transfer procedures on decision trees, with no available source data and a few labeled target data, with the hypothesis of same feature spaces. This setting can be referred to as model-based inductive and homogeneous transfer learning.

2.2 Class imbalance learning

In classification applications, the imbalance problem may arise in different situations [77]. First, class distributions may be unequal, i.e. one class is largely under-represented compared to others. This may be due to a difficulty in the acquisition of a certain class in the data set (e.g. fraud detection, or rare event in time series). Second, error costs may be different depending on the class. This is the case when one aims at recognizing (or not) a class that has an arbitrary high importance, as for example in medical diagnosis. Those two situations may also happen at the same time, i.e. we may have a rare event in the training data set and at the same time it may be crucial to detect it.

To overcome these biases, most methods use either sampling techniques or training procedures that include the cost matrix. Most popular sampling methods use training data to build a balanced set to avoid the learning procedure to be biased. In this framework, the most straightforward approaches are oversampling and undersampling [77]. Sampling methods can also perform data augmentation through the creation of synthetic instances [37]. They have yet some limitations such as discarding potential useful data, high variance, or greater learning time. Cost-sensitive methods aim at integrating costs of different errors into the model. This can be done through boosting [168], or directly within the learning procedure for example with neural networks [98]. According to Weiss and Khoshgoftaar [191], transfer learning under class imbalanced domain is a challenging task that may lead to negative transfer. Authors define *domain class imbalance*, which differs from classical class imbalance in the fact that there is a *variation* in class probability between source and target data. It gathers situations where source data might be balanced whereas target is imbalanced or the other way around. In this work, we focus on the case where source data is balanced. It is our belief that in most practical cases, one may have access to a balanced data set for training a model that will perform on data that is potentially imbalanced.

2.3 Decision tree for data stream

In the context of model-based transfer learning, some methods have been designed for various kinds of model including decision trees (DTs), and by extension random forests (RFs). Regarding DTs, research has been done in the context of evolving data especially for data streams. First works were introduced using Hoeffding trees, originally designed to learn from "regular" stream and later improved for drifting data [21, 85] or for imbalanced drifting data [96]. These works use the term *concept drift*, which is a change in data distribution when performing learning and can be viewed as inductive transfer within streams. Concerning RFs, Gomes et al. [69] proposed an adaptive RF (ARF), which includes sampling methods and specific operators to cope with potential drifts. In practice, the ARF trains a "background tree" as soon as a concept drift is detected, and it replaces its current model when the drift effectively occurs. Concept drift techniques can be effectively used to update random forests once they are already adapted to the target domain. However, it might not be possible to gather enough data to build representative background trees.

Broadly speaking, data stream methods are not dedicated to the adaptation of a model from a source to a target domain but rather to keep up with continuous changes in data, hence deviating from the transfer learning hypothesis (the target domain is not supposed to change and we have access to a certain fixed batch of data).

The problem of adapting a RF trained on a source domain to a target domain has been investigated recently by Segev et al. [159] who propose two functions (called STRUT and SER) that aim at adapting each tree from the original RF to improve its performance with respect to target data.

3 Transfer learning on decision tree

In our application framework (i.e. fall detection in nursing homes), we use a random forest model trained over an experimental data set. However, since neither this source model or the target model yields good performance over target data, transfer on decision tree is a well-adapted approach to our problem. Besides, the fact that there is no use of source data for model-based transfer procedures makes them even more interesting for privacy or storage limitations. In this section, we recall the SER and STRUT methods introduced by Segev et al. [159] in order to highlight our transfer learning contribution on decision tree with imbalanced data.

Decision tree parameters. As described in Section 3.1 on page 65, a decision tree (DT) is defined by the division of the feature space into non-overlapping regions. Once the DT is trained, a new sample is then evaluated according to the region it falls in, or, from the tree point-of-view, through a path of successive thresholds. The parameters of a DT can be summarized as the set of splits (or nodes), each split being a couple (X_q, τ) of a feature and a threshold value. As model-based transfer methods aim at modifying a model's parameters according to new available labelled data, it raises the question of how to modify a DT. Given a set of splits, one straightforward approach is to add or remove some of them, hence either sub-dividing the feature space into smaller regions or merging regions. These operations respectively consist of *growing* a subtree from an existing leaf and *pruning* a node (hence removing the current node and all its children). Another approach would be

to act on the splits themselves and modify their parameters.

These two approaches are proposed by Segev et al. [159] in the form of two algorithms. Structure Expansion-Reduction (SER) modifies the tree structure in order to create finer or coarser representations, while Structure Transfer (STRUT) adapts the threshold values in order to cope with shifts.

Framework and notations. Given a source domain $\mathcal{D}^S$ and a target domain $\mathcal{D}^T$, we assume that we have access to the DT model trained over $\mathcal{D}^S$, and to a few labeled samples from $\mathcal{D}^T$, however we do not have access to source data. The target set is denoted S^T. Given the DT trained on source data and one of its node v, S_v^T denotes the subset of S^T that reaches v. In the following, we also denote by *source tree* the decision tree that is trained on source data and that we aim to adapt to target data.

3.1 Expansion and reduction of the tree (SER)

Structure Expansion-Reduction (SER) is a recursive algorithm that applies two transformations relatively to target data, in two successive steps: an *expansion* step followed by a *reduction* step.

- The *expansion* step consists of expanding any terminal node (or leaf) v that is reached by target data into a subtree. This subtree is computed by growing a full binary decision tree using CART algorithm from S_v^T.
- The second transformation, the *reduction*, relies on the computation of the *leaf error* which, for any node v and its corresponding set S_v^T, is defined as the misclassification error at the current node:

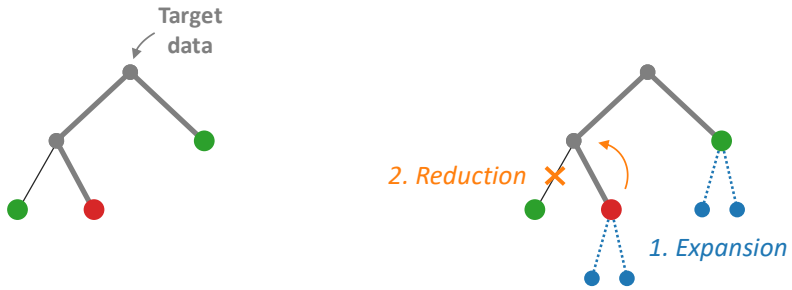
$$\varepsilon_{\text{leaf}}(v, S_v) = \frac{1}{|S_v^T|} \sum_{k=1}^{|S_v^T|} \mathbb{1}(y_k \neq y_v) \ ,$$

  with y_k the label of the k-th element of S_v and y_v prediction label in v. It cuts any node that has a leaf error lower to the *subtree error*. The "cutting" procedure is called *pruning*. The subtree error relative to a subset S_v^T is the sum of all leaves errors weighted by proportion of S_v^T that reaches each leaf. In other words it is the error of the tree whose root is the node v:

$$\varepsilon_{\text{subtree}}(v, S_v^T) = \frac{1}{|L|} \sum_{l \in L} \varepsilon_{\text{leaf}}(l, S_l) \ ,$$

  where L is the set of leaves of the subtree that starts at node v.

When does pruning occur ? In the original paper, these errors are described as the empirical errors of the subtree (whose root is v) and the leaf (considering v as a leaf). We remark that as the expansion step grows a full tree, this results in the creation of pure leaves (when regarding target data). Therefore, $\varepsilon_{\text{subtree}}$ is in fact always zero, and this step becomes a pruning procedure that conserves the tree coherence for target data. Indeed, if a node v is not reached by target data, then $\varepsilon_{\text{leaf}} = 0$ and the pruning condition is respected. This is according to us the only case where pruning occurs, and it leads to the



(a) Original decision tree (b) SER procedure

Figure 4.5: Schematics of SER. Target training data is first passed trough the original tree (a). Then, the algorithm expands all reached leaves into new trees and prunes nodes according to $\varepsilon_{\text{leaf}}$ and $\varepsilon_{\text{subtree}}$ (b). Classes are pictured with green and red leaves.

same condition developed in STRUT algorithm (see next section), which is motivated by the willing to obtain a tree that fits new data by removing unreachable parts of the tree.

Summarizing schematics of SER are displayed on Figure 4.5, and its pseudo-code is presented in Algorithm 3.1.

Algorithm 4.1 SER

procedure SER(v, S_v)
  % Expansion
  if v is a leaf **then**
    $v \leftarrow$ BuildTree(S_v)
    return v
  end if
  % Recursive calls
  SER($v_r, S_{v,r}$)
  SER($v_l, S_{v,l}$)
  % Reduction
  if $\varepsilon_{\text{leaf}}(v, S_v) \leq \varepsilon_{\text{subtree}}(v, S_v)$ **then**
    $v \leftarrow$ Prune(v)
  end if
  return v
end procedure

Notations: BuildTree denotes the procedure that grows a full tree from a terminal node, following the CART algorithm.

3.2 Transfer on the tree structure (STRUT)

The Structure Transfer (STRUT) algorithm goes through the decision tree from the top to the bottom. Given a node v, if v is unreachable by target data, it is pruned into a leaf. Otherwise, there are two possible situations. Either it is a leaf and its output is then updated, or it is a regular node and the algorithm recomputes its threshold according to

the set S_v^T that falls into it. Unlike a classical split procedure, here the measure that is optimized is different and there is no choosing in the feature (the algorithm updates the threshold while keeping the same variable).

The threshold selection procedure is done as follows. Given a node v, Q_l^S and Q_r^S denote respectively the class proportions of source data in left child node and right child node after the original split. $Q_l^T(\tau)$ and $Q_r^T(\tau)$ denote respectively the class proportions of target data in left child node and right child node after the new split τ. As authors [159] aim at finding a new threshold while keeping similar label distributions between source and target, they define a *divergence gain* (denoted DG) that measures the similarity between the original label distributions (Q_l^S, Q_r^S) and the new ones (Q_l^T, Q_r^T). DG is defined as:

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

$S_{v,r}^T$ and $S_{v,l}^T$ are the subsets of S_v^T that falls respectively into the right and left child node of v, and JSD is the Jensen-Shannon divergence, defined as:

$$\mathrm{JSD}(P, Q) = \frac{1}{2}\left(\mathrm{KL}(P, M) + \mathrm{KL}(Q, M)\right) \ ,$$

with KL denoting the Kullback-Leibler divergence:

$$\mathrm{KL}(P_1, P_2) = \sum_k P_1(k) \log\left(\frac{P_1(k)}{P_2(k)}\right) \ ,$$

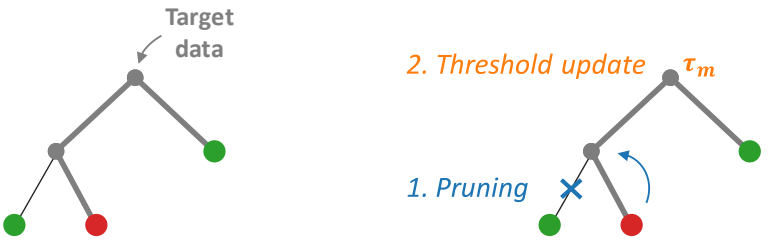
and M the mean distribution, defined as $M = \frac{1}{2}(P+Q)$. The Jensen-Shannon divergence is a symmetrized version of KL that measures the dissimilarity between the two distributions P and Q. Note that while KL can have infinite value (when the two distributions do not share the same support), JSD is positive and bounded by 1 (with $\log$ being the base 2 logarithm) [107]. Hence defined, DG measures for a node v the similarity between previously learned source distribution and the newly set target distribution. We note that in practice, Q_l^S, Q_r^S, Q_l^T and Q_r^T are vectors that contain the proportions of each class at a given node. Hence we can write them $Q_l^S = \left[Q_{l,1}^S, \ldots, Q_{l,K}^S\right]$, K being the number of classes.

The threshold selection procedure relies on an optimization problem that can be summarized as follows: the goal is to maximize DG while insuring a local maximum of the *information gain* (IG), defined here as the Gini gain (definition given by Equation (3.2)). At node v, we denote $\Phi_v = \{\phi_1, \ldots, \phi_N\}$ the set of all ordered distinct feature values of instances of S_v^T. Then, $\mathrm{T}_v = \{\tau_1, \ldots, \tau_{N-1}\} = \{\frac{\phi_1+\phi_2}{2}, \ldots, \frac{\phi_{N-1}+\phi_N}{2}\}$ represents the set of all possible thresholds considered by the *threshold selection*.

The selected τ denoted τ_m is then defined as:

$$\tau_m = \underset{\tau \in \mathrm{T}_v}{\arg\max}\, \mathrm{DG}(S_v^T, \tau, Q_l^S, Q_r^S) \quad \text{s.t.} \quad \begin{cases} \mathrm{IG}(\tau_{m-1}) < \mathrm{IG}(\tau_m) \\ \mathrm{IG}(\tau_m) > \mathrm{IG}(\tau_{m+1}) \end{cases}$$

As pointed out by authors, between source and target, different labels can also swap relatively to the node threshold. To take into account this possible event during the *threshold selection*, the optimization problem has to be solved a second time swapping left and right



(a) Original decision tree (b) STRUT procedure

Figure 4.6: Schematics of STRUT. Like SER, target data is first passed through the original tree (a). Then, the algorithm prunes unreachable node and updates thresholds of reached nodes according to new data (b). Classes are pictured with green and red leaves.

for Q^S and Q^T which gives another threshold $\overline{\tau_m}$. Finally, the threshold selection procedure ends by taking the threshold maximizing DG between the two optimal thresholds : τ_m and $\overline{\tau_m}$. Schematics of the STRUT algorithm are displayed in Figure 4.6 and its pseudo-code is presented in Algorithm 4.2.

Algorithm 4.2 STRUT

procedure STRUT(v, S_v)
  if $|S_v| = 0$ **then**
    % Prune unreachable node
    $v \leftarrow$ Prune(v)
    return v
  else if v is a leaf **then**
    $v \leftarrow$ UpdateLeafValue(S_v)
    return v
  else
    % Recompute threshold
    $v \leftarrow$ ThresholdSelection(S_v, Q_l^S, Q_r^S)
    STRUT($v_r, S_{v,r}$)
    STRUT($v_l, S_{v,l}$)
  end if
end procedure

To sum up, SER seems to be well adapted when decision tree partitioning needs to be refined (expansion step) or on the contrary to be more coarse (reduction step). On the other hand, STRUT is especially designed for drifts and label swaps. Together, both procedures capture pretty well the possible transformations a decision tree may need to be transferred along target data. However, when exposed to class imbalance, these methods can suffer undesirable effects.

4 Transfer learning on decision tree with class imbalance

In this section, we present the main issues of using SER and STRUT in the case of imbalanced data and propose an adaptation for each one of them. For that we first define a framework that we named *homogeneous class imbalance* in order to get an idea of the misbehaviour that we may encounter with the pruning procedure in SER and STRUT.

4.1 Leaf loss risk under homogeneous class imbalance

Indeed, as SER and STRUT both use pruning procedure, it is likely that classes that are the least represented in the target set will have their corresponding leaves being pruned in the DT.

Homogeneous class imbalance. The underlying object while transferring models is the relation between the two distributions $P^S(x, y)$ and $P^T(x, y)$. Let $P^S(y) = \int P^S(x, y)dx$ and $P^T(y) = \int P^T(x, y)dx$ denote respectively the source and target marginal distributions over $\mathcal{Y}$.

Considering imbalanced data, we study the specific effect of class proportion changes between source and target. To this end, we consider a simple case when it comes to label distribution changes.

Definition 4.4 (Homogeneous class imbalance)**.** *Given a source domain $\mathcal{D}^S = \{\mathcal{X}, P^S(X^S)\}$ and a target domain $\mathcal{D}^T = \{\mathcal{X}, P^T(X^T)\}$, homogeneous class imbalance occurs when:*

$$P^T(x|y) = P^S(x|y) \tag{4.1}$$

$$P^T(y|x) = \frac{\lambda_y P^S(y|x)}{\int \lambda_y P^S(y|x)dy} \ , \tag{4.2}$$

with $\lambda_y = \frac{P^T(y)}{P^S(y)}$ being the ratio between target and source proportion class for each class y.

Here, the term *homogeneous* is not in any way related to *homogeneous transfer learning* but rather means that the class proportion change is homogeneous throughout the feature space since it depends only on λ_y. This framework is simple in the fact that conditionals of x given y between source and target are considered to be the same, which is generally not the case in transfer learning situations. However we show later that this gives interesting insights over transfer procedures on DTs.

Leaf loss risk. We formulate the pruning risk as the significant minority class leaf loss risk, i.e. the risk for a minority class leaf to be pruned even if it is still representative for the target domain.

Definition 4.5 (Significant leaf)**.** *Let f^S be a decision tree learnt over a training set sampled from the source domain $\mathcal{D}^S$, and k_{min} the minority class in the target domain $\mathcal{D}^T$. Let l be a leaf of f^S of class k_{min}. l is still significant for $\mathcal{D}^T$ if:*

$$\forall k \neq k_{min}, \quad P^T(y = k_{min} \,|\, x \in l) > P^T(y = k \,|\, x \in l) \ . \tag{4.3}$$

Definition 4.6 (Leaf loss risk)**.** *Let $S^T = \{(x_i, y_i), i = 1 \dots n\}$ denote the target training set, where $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ are independent and identically distributed. n_k denotes the*

number of instances of class k. Let consider l, a minority class leaf still significant for the target domain. When using an algorithm that prunes leaves that are not reached by target data, the risk of losing l is then quantified by:

$$R_L(l) = P^T(x \notin l \,|\, y = k_{min})^{n_{k_{min}}} \ . \tag{4.4}$$

$R_L(l)$ is obtained as follows. The risk for l to be pruned is expressed as the probability of not having a single sample falling in l:

$$R_L(l) = P^T\left(\bigcap_{i=1}^{n_{k_{min}}} (x_i \notin l \,|\, y = k_{min})\right) \ .$$

Since the (x_i, y_i) are independent, then the x_i are conditionally independents given y_i, which gives:

$$R_L(l) = \prod_{i=1}^{n_{k_{min}}} P^T(x_i \notin l \,|\, y = k_{min}) \ ,$$

and, given that the (x_i, y_i) are identically distributed:

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

In balanced conditions, $n_{k_{min}}$ is large enough so the probability tends to 0 and the pruning risk becomes negligible. However when $n_{k_{min}}$ decreases, it leads to higher quantity of pruned leaves. Let us now consider the case where the only transformation that occurs between source and target is an homogeneous class imbalance. Under this assumption and simply using (4.1), (4.3) and (4.4) can be reformulated as follows:

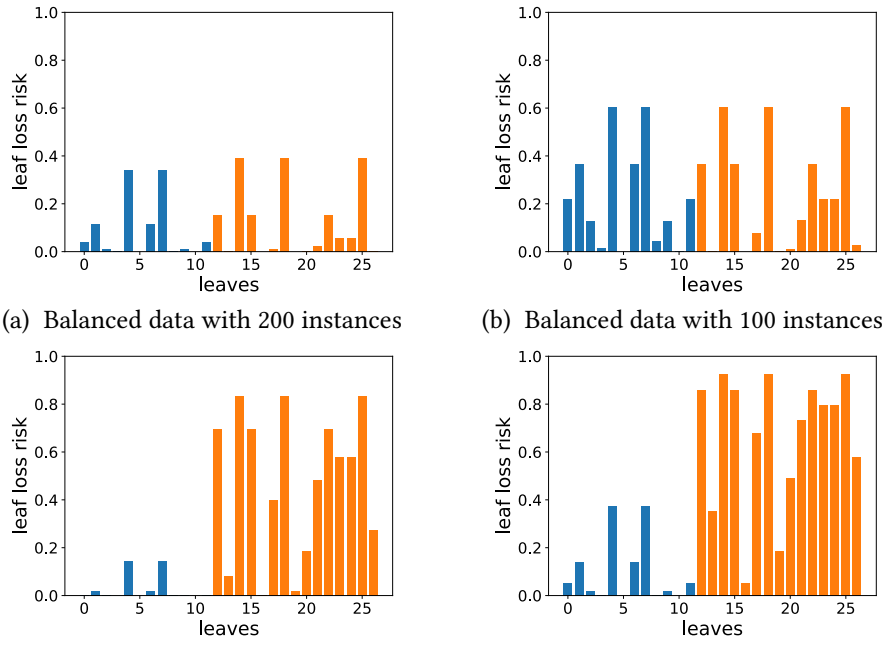
$$\forall k \neq k_{min}, \quad \lambda_{k_{min}} P^S(y = k_{min} \,|\, x \in l) > \lambda_k P^S(y = k \,|\, x \in l) \tag{4.5}$$

$$R_L(l) = P^S(x \notin l \,|\, y = k_{min})^{n_{k_{min}}} \tag{4.6}$$

Although this result relies on a theoretical assumption that is not necessarily verified in practice, computing this value gives us valuable information about the impact of using pruning algorithms such as SER and STRUT on minority class leaves. Figure 4.7 shows examples of leaf loss risk computation using (4.5) and (4.6) under various conditions.

4.2 Divergence gain

The divergence gain (DG) defined for the STRUT procedure measures the similarity between previous class distributions (Q_l^S, Q_r^S) and the new ones (Q_l^T, Q_r^T). As previously said, STRUT algorithm recomputes thresholds from the top to the bottom of the tree using DG (and IG). Hence at a given node, when the threshold is recomputed, label distributions according to source data (Q_l^S and Q_r^S) change. However, as we consider that we do not have access to the source data during the transfer procedure, it means that we cannot recompute Q_l^S and Q_r^S according to the new threshold. As a consequence, when we go deeper into the tree while performing STRUT, Q_l^S and Q_r^S that are used in the optimization problem are obsolete (they correspond to old threshold values) and are therefore more likely to mislead the algorithm, especially if distributions were swapped as described



(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

Figure 4.7: Leaf loss risk for one DT under different target data conditions, with two classes. Leaves are in abscissa. Blue ones are **majority class** leaves and orange ones correspond to **minority class**. With balanced target data (a), the risk of losing minority leaves is similar to the risk of losing majority class leaves. When decreasing the number of target data (b), the risk increases equally regardless of the leaf class. However when dealing with imbalanced data (c), the leaf loss risk on minority leaves is significantly higher, and in the same imbalance conditions while decreasing the number of target data (d), the risk is even worse.

previously. This is due to the fact that at current node v, computing Q_l^S and Q_r^S implies to know where source data falls according to all thresholds that lead to this node (i.e. the thresholds of all parents nodes of v).

Besides, more broadly speaking, the optimization procedure prompts the model to choose a threshold so that new class distributions do no differ too much from the old ones. This implies the assumption that at each node, class distributions have not much evolved between source and target. However in the case of domain class imbalance the situation is quite the opposite: we need to transfer a source model built on balanced data with a target distribution that is imbalanced, hence we expect different distributions in the end. To show the influence of DG in the performances, a version of STRUT *without using* DG is proposed and tested along with all other algorithms.

These two issues, i.e. the leaf loss risk and the optimization procedure in STRUT, lead us to adapt the initial SER and STRUT algorithms to target data class imbalance.

4.3 Expansion and reduction for class imbalance

As previously pointed out, the main concern comes from the representative leaf loss risk of these transfer algorithms when there are too few target data of a class. In fact, for a given leaf l in the source tree, there are three possible events relatively to target training data:

1. l is reached by target training data and keeps the same majority class as source;
2. l is reached by target training data and the majority class changes (i.e. it differs from the source majority class);
3. l is not reached by any target training data.

In case 1, the minority class leaf is conserved. We focus our attention on the two other situations. Considering SER algorithm, these two possibilities match the two transfer steps. Case 2 corresponds to an *expansion* of the tree and case 3 to a *reduction.* Therefore, in order to limit minority leaf loss events, we set up a version of SER that applies the *reduction* step conditionally to the class. For this purpose, we propose two configurations:

1. The algorithm avoids automatically to prune any source leaf of the minority class but allows the *expansion* phase on them. This version is named SER_{R} (for "SER without reduction over minority class leaves").
2. The algorithm does not prune the minority class leaf if it is still *significant* (as defined above) and its leaf loss risk (as defined previously) is above a given threshold set to 0.5. This version is named SER_{LL} (LL refers to the leaf loss risk).

The SER_{R} approach may seem harsh since it conserves leaves that were defined with source data and not target. However if we consider that doing transfer with imbalanced data biases the model towards the majority class, this method aims to compensate this bias with another one towards the minority class. The SER_{LL} approach is more refined since the pruning is still allowed on some conditions.

Since they differ by just one condition, both methods are gathered in one algorithm whose pseudo-code is given in Algorithm 4.3.

4.4 Structure transfer with a generalized divergence

As explained previously, class ratio changes are not directly compatible with the use of the divergence in the STRUT algorithm. The original algorithm without this criteria is proposed as an alternative. In this configuration, it simply consists of updating thresholds by recomputing the maximum Gini gain over target data while keeping the same feature. Throughout the rest of the paper, this version is referred to as STRUT_{IG} (for information gain).

Another concern is that our transfer methods need to be able to handle at the very least the homogeneous class imbalance transformations. Equation (4.2) in a discrete formulation

Algorithm 4.3 SER_{R}, SER_{LL}

procedure SER*(v, S_v, opt)
  % Expansion
  if v is a leaf **then**
    $v \leftarrow \text{BuildTree}(S_v)$
    return v
  end if
  % Recursive calls
  SER*$(v_r, S_{v,r}, opt)$
  SER*$(v_l, S_{v,l}, opt)$
  % Reduction
  if $\text{LeafError}(v, S_v) \leq \text{TreeError}(v, S_v)$ **then**
    % Controlled pruning
    if $c(v) == c_m$ **then**
      % Majority class in node v is the minority class
      if $opt ==$ 'LL' **then**
        % Check for leaf loss conditions
        if *leaf is significant* **and** $R_L > 0.5$ **then**
          % No pruning
        else
          $v \leftarrow \text{Prune}(v)$
        end if
      end if
      if $opt ==$ 'R' **then**
        % No pruning
      end if
    else
      $v \leftarrow \text{Prune}(v)$
    end if
  end if
  return v
end procedure

Notations: c_m is the overall minority class, $c(v)$ is the majority class in node v

becomes:

$$P^T(y = k \mid x) = \frac{\lambda_k P^S(y = k \mid x)}{\sum_i \lambda_i P^S(y = i \mid x)} \tag{4.7}$$

The idea of our STRUT extension is to use Equation (4.7) in the divergence optimization and replacing Q_l^S and Q_r^S as if the homogeneous class imbalance condition was satisfied. For each class k, the class proportions according to source data become:

$$Q_{l,k}^S = \frac{\lambda_k Q_{l,k}^S}{\sum_i \lambda_i Q_{l,i}^S} \qquad Q_{r,k}^S = \frac{\lambda_k Q_{r,k}^S}{\sum_i \lambda_i Q_{r,i}^S} \tag{4.8}$$

Algorithm 4.4 STRUT_{IG}, STRUT_{HI}

procedure STRUT$^*(v, S_v, \lambda, opt)$
  if $v = root$ **then**
    % Compute λ_k
    for each class k **do**:
      $\lambda_k = \frac{p_k(S^T)}{p_k(S^S)}$
    end for
  end if
  if $|S_v| = 0$ **then**
    % Prune unreachable node
    $v \leftarrow \text{Prune}(v)$
    return v
  else if v is a leaf **then**
    $v \leftarrow \text{UpdateLeafValue}(S_v)$
    return v
  else
    % Recompute threshold
    if $opt ==$ 'IG' **then**
      % Selecting threshold according to max IG only
      $v \leftarrow \text{ThresholdSelection}_{IG}(S_v)$
    end if
    if $opt ==$ 'HI' **then**
      % Selecting threshold with modified source proportions
      $(Q_l^S, Q_r^S) \leftarrow \text{HomogeneousImb}(Q_l^S, Q_r^S)$
      $v \leftarrow \text{ThresholdSelection}(S_v, Q_l^S, Q_r^S)$
    end if
    STRUT$^*(v_r, S_{v,r}, \lambda, opt)$
    STRUT$^*(v_l, S_{v,l}, \lambda, opt)$
  end if
  return v
end procedure

Notations: HomogeneousImb is the function that uses Equation (4.8) to adapt the source distributions

This extension of STRUT is referred to as STRUT_{HI} (for homogeneous imbalance). We note that in this configuration, if we consider a transfer procedure in which class proportions are conserved between source and target, then for each class k we have $\lambda_k \approx 1$. This leads to ${Q_{l,k}^S}^* \approx Q_{l,k}^S$ and ${Q_{l,k}^S}^* \approx Q_{l,k}^S$ which goes back to the original STRUT algorithm. Like SER_{R} and SER_{LL}, STRUT_{IG} and STRUT_{HI} are presented within the same pseudo-code given by Algorithm 4.4.

A Python implementation of SER, STRUT and all the proposed variants is available here [12] (joint work with Mounir Atiq and Sergio Peignier).

5 Data and experimental setup

This section presents the data sets on which algorithms are tested (synthetic data, public data and fall data), as well as the experimental framework. Note that since we aim to describe decision tree transfer procedures, all experiments are run on a decision tree model, knowing that these results may be extrapolated to tree ensemble models such as random forests. However, since our work is initially motivated by the use of random forests, additional tests with a RF model are included in the experiments with Tarkett's fall data sets.

5.1 Synthetic data

In order to easily control data transformations between source and target, we generate several data sets with three-dimensional binary labeled clusters in a bounded space. Each synthetic source data set consists of a mixture of several multivariate Gaussian clusters, each one being assigned to a single class. For simplicity the clusters have diagonal covariance matrices, meaning that for each cluster dimensions are not correlated. The initial parameters μ_i and σ_i of all Gaussian distributions are randomly drawn from Uniform distribution: $\mu_i \in [-70, 70]$ and $\sigma_i \in [5, 15]$ for $i = 1, 2, 3$. Source data consists of $N_{source} = 200$ samples, with balanced class distribution (i.e. 50% of each class) and drawn out of $N_{clust} = 10$ clusters. Note that to obtain "mixed" clusters and a complex separation between classes, one has to tune at the same time the Uniform bounds and N_{clust}. Hence, these parameters were arbitrarily set so that the classification task is achievable although not to easy.

After drawing source samples, we apply imbalance conditions and transformations to the generator and redraw target samples. Data imbalance is controlled with weights over clusters. Indeed, each cluster is assigned a weight that determines the probability of data to be drawn from that cluster, hence allowing us to control the class distributions and therefore create homogeneous class imbalance conditions. The imbalance ratio is ranging from 2% to 50% and is combined with transformations of tree kinds:

1. **Drift:** change in Gaussian clusters means, resulting in "drifts" in the feature space.
2. **Squeeze / Stretch**: change in Gaussian clusters variances, resulting in either "stretching" or "squeezing" data.
3. **Add / Remove**: mix between adding and removing clusters to the generator.

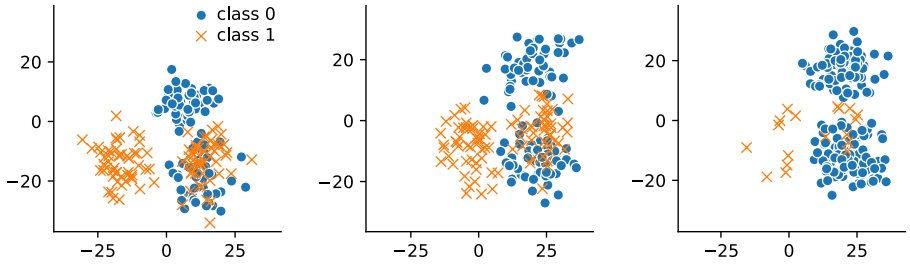
Figure 4.8 shows a simplified example of synthetic data generation.

These synthetic Gaussian controlled scenarios have the advantage of generating as much data as needed for the evaluation process to ensure good performance measure. In our experiments we use $N_{target} = 1000$ samples. A Python implementation of the generator (joint work with Sergio Peignier and Mounir Atiq) is available at [140].

5.2 Real-world public data sets

5.2.1 Public data

Two public data sets are used to analyse the performances of these procedures.



(a) Source data... (b) ...after a drift... (c) ...and homogeneous imbalance

Figure 4.8: Example of synthetic data generation in two dimensions, with $N_{clust} = 4$. Data is first generated with $\mu \in [-20, 20]$, $\sigma \in [20, 50]$ (a), then undergoes a drifting procedure (b) and finally an homogeneous class imbalance (c). Note that after each transformation, data is resampled from the generator.

Magic Gamma Telescope (MGT). This data set comes from simulated registrations of high energy gamma particles in a Cherenkov gamma telescope. Photons are collected in patterns as "shower image" and the goal is to separate images formed by gamma signals from images created by other types of rays. There are 19,020 instances described by 10 features. Those features are real numbers that give characteristics of the shower image. The data set is available on UCI repository [53]. Used by Weiss and Khoshgoftaar [191, 192], it is not a dedicated data set to test transfer learning methods, however it has a natural imbalance (35% positive class) and a large amount of data so that it can be tested under great imbalance conditions. To separate source data from target data, we chose to divide along the feature that has the greatest variance. Source and target data are each set to 500 instances.

Office-Caltech (OC). The Office-Caltech data set contains images from Amazon website and office environment images taken under different conditions (with a webcam or a higher quality camera). There are in all 10 classes available in four domains: *amazon*, *caltech* *dslr*, *webcam*. Features are generated by SURF algorithm categorized in 800 dimensions available from [1]. We use the first two domains as source and *webcam* as target. This reflects the general framework where one can have access to data acquired in a controlled environment (here images for sales purposes) but faces challenging data when it comes to real environment (here low quality images taken in different conditions of exposition etc.) The *amazon* set contains 958 instances ($\approx$ 100 instances per class), the *caltech* set has 1123 instances ($\approx$ 100 instances per class) and the *webcam* one has 295 instances ($\approx$ 30 instances per class). We use a one-versus-all classification scheme and average results over all classification tasks.

For public data sets, the imbalance ratio is controlled with downsampling. Source training data set is always made balanced (50% of each class), and target training set is made imbalanced with ratios going from 5% to 50% of minority class. All experiments were conducted through k-fold cross-validation with $k = 5$.

5.3 Fall data

The transfer methods are finally tested over Tarkett's fall data sets, one for each domain. The experimental data set (described in previous chapter) is used as source and the target set is built with real fall data (described in the introduction of this chapter). We recall that the experimental data set contains 742 events with 45% falls.

Real fall events were recorded using our fall detection system installed in nursing homes, hence gathering a significant amount of data. However for previously mentioned reasons fall labels are hard to obtain, hence resulting in a data set containing 174 fall events and 2619 non-fall events (falls represent about 6% of the total). In these experiments, we do not seek to manipulate the imbalance ratio but rather investigate the number of available data, which is an important concern as real data acquisition is expensive in several respects. For this purpose, when using k-fold validation, instead of using the largest set for training (and the smallest for testing), the opposite is done, hence allowing to observe the effects of data scarcity. The number of folds k is set to different values ranging from 5 to 40, resulting in respectively 35 and 4 fall events in the training target data set.

For this data set, additional experiments are run with a random forest model of 10 decision trees. In this case, the procedure is exactly the same as for the decision tree model experiments, the only difference being that when the *Source* model (i.e. the RF trained over source data) is transferred with one of the algorithms (SER, STRUT, or a variant), the algorithm is run over all its trees, hence transferring the whole RF.

5.4 Performance measures

These experiments on synthetic and real data compare the original SER and STRUT algorithms with their proposed variants under class imbalance and rarity of the minority class in target data set. While dealing with these two conditions, choosing a performance measure that does not penalize the minority class, unlike accuracy for example, is a known issue [17]. As ROC curves consider all the trade-offs between false positive rate and true positive rate, they are commonly computed to extract performance assessment metrics in class imbalance situations [77]. Hence, the area under the ROC (ROC AUC) curve is generally used as a reference metric in the context of transfer learning with imbalanced conditions [5]. For these reasons, in this work, performance results of different presented models are also compared using the ROC AUC measure.

In all experiments, models trained only with source data or trained with only target data are also tested so they can be used as reference. Indeed, when performing transfer learning, one should make sure the transferred model outperforms the original one (trained on source data) and the model trained with the few available target data. In all results those models are respectively referred to as *Source* and *Target*.

I	C	*Source*	*Target*	**SER**	**SER$_R$**	**SER$_{LL}$**	**STRUT**	**STRUT$_{IG}$**	**STRUT$_{HI}$**
					I. Drift				
5%	0	17 ± 5	5 ± 2	14 ± 5	18 ± 6	15 ± 5	5 ± 2	4 ± 2	8 ± 3
	1	17 ± 5	4 ± 2	6 ± 2	12 ± 3	6 ± 2	2 ± 1	2 ± 1	4 ± 2
50%	0	16 ± 4	13 ± 3	21 ± 6	23 ± 7	21 ± 6	12 ± 3	8 ± 2	12 ± 3
	1	16 ± 5	13 ± 4	22 ± 5	25 ± 6	22 ± 5	11 ± 3	7 ± 2	11 ± 3
					II. Squeeze / Stretch				
5%	0	22 ± 3	6 ± 2	19 ± 5	24 ± 4	19 ± 5	5 ± 2	4 ± 2	9 ± 3
	1	23 ± 3	4 ± 2	6 ± 2	13 ± 3	7 ± 2	2 ± 1	2 ± 1	4 ± 2
50%	0	22 ± 3	16 ± 3	29 ± 3	32 ± 5	29 ± 3	15 ± 3	8 ± 3	15 ± 3
	1	23 ± 3	16 ± 3	30 ± 4	33 ± 5	29 ± 4	14 ± 3	8 ± 3	14 ± 3
					III. Add / Remove				
5%	0	19 ± 3	5 ± 2	15 ± 5	19 ± 5	15 ± 5	5 ± 2	5 ± 2	8 ± 2
	1	18 ± 2	4 ± 2	6 ± 2	11 ± 3	6 ± 2	2 ± 1	2 ± 1	4 ± 1
50%	0	19 ± 3	14 ± 3	24 ± 5	26 ± 5	24 ± 5	13 ± 3	8 ± 3	13 ± 3
	1	18 ± 3	13 ± 3	24 ± 5	27 ± 5	24 ± 5	11 ± 3	7 ± 2	11 ± 3

Table 4.1: Mean number of leaves for each class with different algorithms tested on synthetic data. First column indicates the imbalance ratio in target, while the second gives the class. When the imbalance ratio is 5%, the minority class is class 1.

.

6 Results

6.1 Synthetic data

6.1.1 Leaf loss risk evaluation

To verify our assumption about the leaf loss risk in original STRUT and SER, we measure the mean number of minority class leaves on resulting trees of both algorithms and compare it with *Source* model, *Target* model and our proposed variants. Results are given in Table 4.1.

For comparison purpose, we also give the results over a balanced target set. We note that in this case, for all transformations the resulting number of leaves is balanced between the two classes, whatever the algorithm. However STRUT and their variants significantly reduce the overall number of leaves. This is a consequence of SER's ability to grow new trees and create new partitions while STRUT cannot.

When confronted with imbalanced target set, the number of minority class leaves for SER and STRUT decision trees is always lower than in the *Source* decision trees, thus illustrating the risk we defined. Considering SER variants, SER_R significantly compensates the loss of minority class leaves, whereas SER_{LL} has practically no impact on this loss. Indeed, SER_R has a strict rule concerning minority class leaves while SER_{LL} is based on a risk estimation. The alternate algorithms of STRUT do not limit the minority leaf loss, which is expected since they are not designed for this purpose. However we notice that $STRUT_{IG}$ particularly decreases the overall number of leaves, which is due to the fact that unlike other algorithms, it has no restriction to conserve previous proportions, hence leading to higher pruning in the tree.

I	*Source*	*Target*	**SER**	**SER$_R$**	**SER$_{LL}$**	**STRUT**	**STRUT$_{IG}$**	**STRUT$_{HI}$**
				I. Drift				
2%	0.66 ± 0.09	0.54 ± 0.05	0.56 ± 0.06	**0.59 ± 0.07**	0.57 ± 0.06	0.55 ± 0.08	0.55 ± 0.06	**0.57 ± 0.08**
5%	0.66 ± 0.09	0.58 ± 0.07	0.61 ± 0.07	**0.63 ± 0.08**	0.61 ± 0.07	0.6 ± 0.1	**0.62 ± 0.09**	0.61 ± 0.08
10%	0.66 ± 0.09	0.63 ± 0.08	0.66 ± 0.08	**0.67 ± 0.08**	0.66 ± 0.08	0.64 ± 0.11	**0.67 ± 0.1**	**0.67 ± 0.09**
				II. Squeeze / Stretch				
2%	0.6 ± 0.04	0.52 ± 0.02	0.52 ± 0.02	**0.54 ± 0.03**	**0.53 ± 0.03**	**0.53 ± 0.05**	**0.53 ± 0.04**	0.52 ± 0.04
5%	0.6 ± 0.04	0.54 ± 0.03	0.55 ± 0.03	**0.56 ± 0.04**	0.55 ± 0.04	**0.56 ± 0.05**	**0.56 ± 0.05**	0.55 ± 0.05
10 %	0.6 ± 0.04	0.56 ± 0.04	0.57 ± 0.04	**0.58 ± 0.04**	**0.58 ± 0.04**	0.57 ± 0.06	0.59 ± 0.06	0.58 ± 0.05
				III. Add / Remove				
2%	0.57 ± 0.12	0.54 ± 0.07	0.57 ± 0.08	**0.6 ± 0.09**	0.58 ± 0.08	0.56 ± 0.1	**0.57 ± 0.1**	**0.57 ± 0.09**
5%	0.57 ± 0.12	0.59 ± 0.09	0.62 ± 0.09	**0.64 ± 0.09**	0.62 ± 0.09	0.61 ± 0.12	**0.63 ± 0.12**	0.62 ± 0.1
10%	0.57 ± 0.12	0.64 ± 0.1	0.67 ± 0.09	**0.68 ± 0.09**	0.67 ± 0.09	0.65 ± 0.12	**0.68 ± 0.12**	0.67 ± 0.1

Table 4.2: Mean ROC AUC over synthetic data tests. Best results within the STRUT and SER family methods are highlighted with bold font.

.

6.1.2 Performance results

Each of the three transformations (described in Section 5.1) is applied with various "strength" to data, to simulate various levels of change between source and target domain. For each level of transformation, several repetitions are conducted and results are averaged. Table 4.2 gives for each transformation the mean ROC AUC over all levels and focuses on three imbalance ratios (2%, 5% and 10%). Figure 4.9 shows the average ROC AUC for one level of each transformation. More results are given in Appendix B showing several levels of transformation between source and target domain.

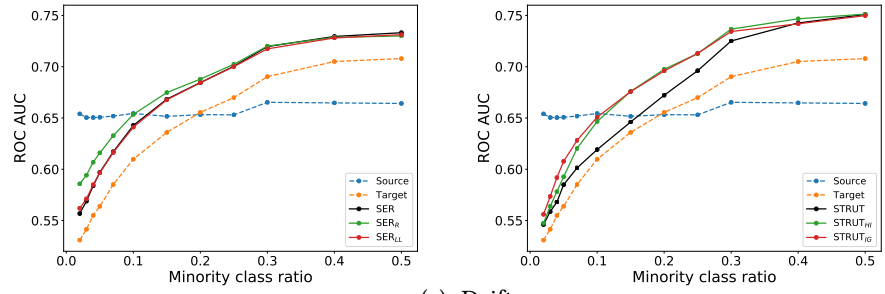
First of all, we notice that when the class imbalance is strong (i.e. when ratio decreases), transfer methods are exposed to negative transfer since they are outperformed by *Source*. However all methods give better results than *Target*. In fact – and this is illustrated in Appendix B, when transformations get stronger, then *Source* is also outperformed, hence suggesting that transfer is a good choice.

Concerning SER and its variants, a first remark is that SER$_{LL}$ does not improve transfer and is generally very close to SER in scores. However, when there is class imbalance, SER$_R$ gives better results than SER, and as the imbalance worsens, it gets even better, thus showing the importance of conserving the minority class leaves that were destined to be pruned. These results suggest that even if SER$_{LL}$ is a more subtle adaptation than SER$_R$, a more radical method is worth considering.

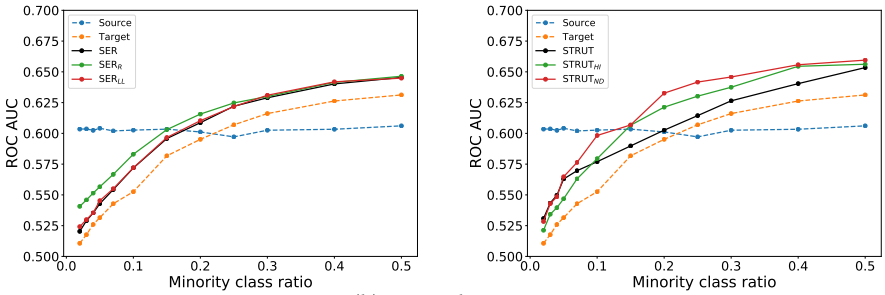
Regarding STRUT variants, STRUT$_{IG}$ and STRUT$_{HI}$ outperform the original STRUT algorithm for all transformation types. STRUT$_{HI}$ is in general slightly better, thus indicating here that taking into account the class proportion change instead of simply reusing the classical information gain may be a better choice.

6.2 Real public data

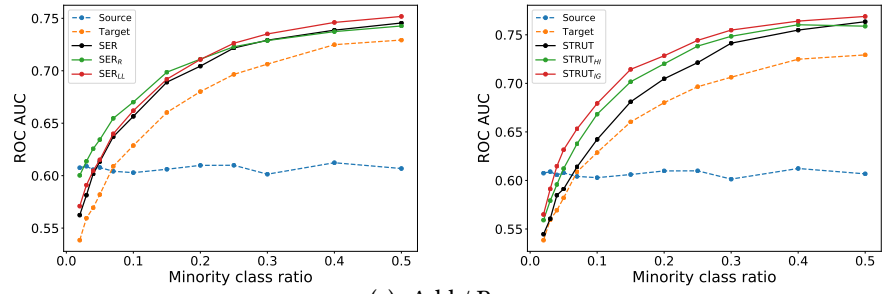
Figure 4.10 provides ROC AUC scores over public data sets (MGT and OC) . Here the results are more mixed.



(a) Drift



(b) Stretch / Squeeze

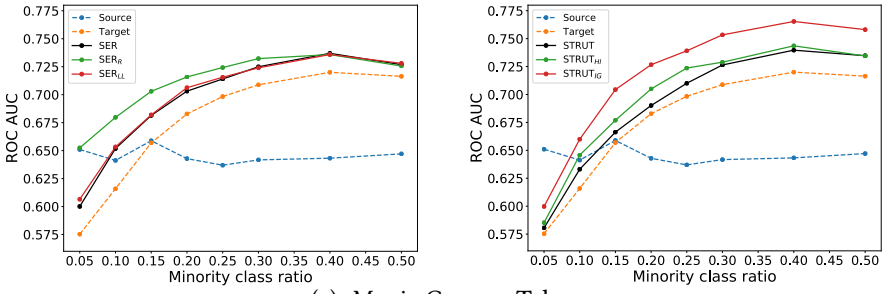


(c) Add / Remove

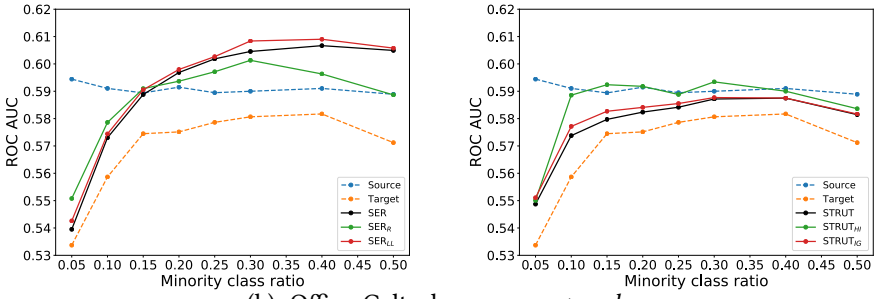
Figure 4.9: ROC AUC over synthetic data. Results are given with varying minority class ratio.

SER_R performs better than SER on MGT especially with high imbalance and insures a valid transfer (i.e. it yields better results than *Source* and *Target*), but is less performing over OC data when the imbalance ratio is above 15%. For both data sets, SER_{LL} is either giving same performance as SER or slightly better. Concerning STRUT variants, both $STRUT_{IG}$ and $STRUT_{HI}$ either outperform or give similar result as STRUT on both data sets. However while $STRUT_{IG}$ gives the best scores over MGT, it is $STRUT_{HI}$ that is the more performing over OC.

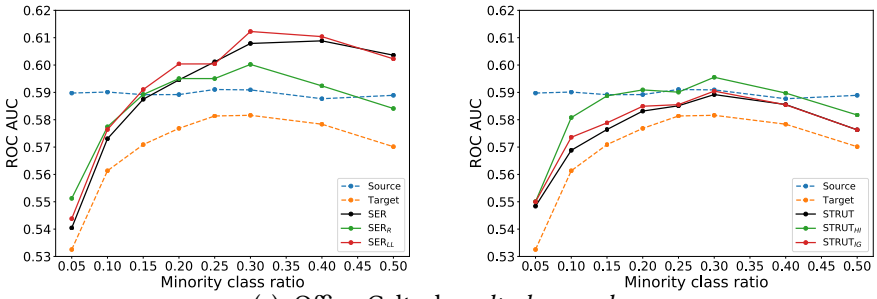
Giving a general conclusion over real data is more delicate, since results are in fact data-dependant. What we can say however is that given our proposed methods, there is at least one of them that outperforms the original one.



(a) Magic Gamma Telescope



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



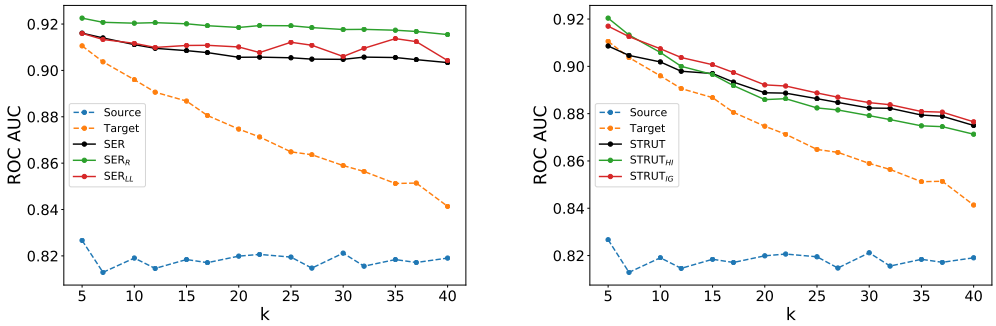
(c) Office-Caltech: *caltech* → *webcam*

Figure 4.10: ROC AUC over public data sets. Performance is measured with varying minority class ratio.

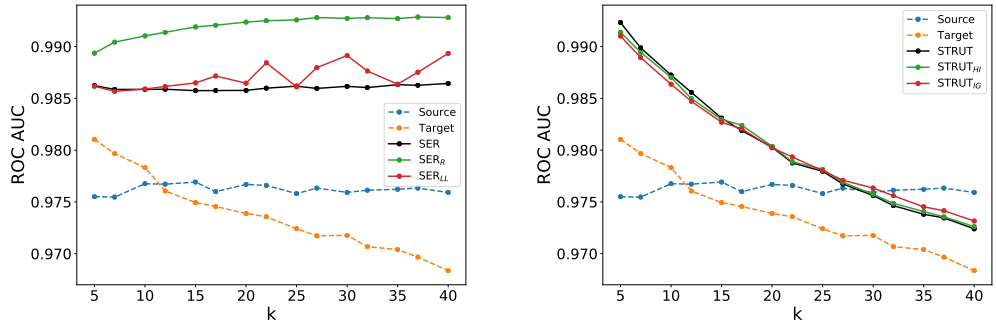
6.3 Fall data

Figure 4.11 displays ROC AUC on our fall data framework (experimental vs. real-world data).

Let us first focus our attention on the decision tree model results. The first important remark to be made is that when the target training set gets smaller (i.e. when k increases), *Target* score significantly decreases toward *Source* score. It clearly illustrates the key role that real-data gathering has in real-world applications. We point out that whatever k, transfer is always valid, highlighting that in the case of our fall data sets, using knowledge from both experimental and real-world data has value in terms of final performance. Concerning proposed algorithms, SER_R outperforms SER along all k values, while SER_LL gives similar or slightly better results. As for STRUT variants, STRUT_IG outperforms the original STRUT when given enough data (i.e. low values of k) and gives similar results otherwise. STRUT_HI is also outperforming with low k, however when target training set



(a) Decision tree model



(b) Random forest model with 10 decision trees

Figure 4.11: ROC AUC over the real-world fall data set (*experimental* $\rightarrow$ *real-world*) for a decision tree (a) and a random forest of 10 decision tree (b). We recall that the imbalance ratio in target is left unchanged (6%) and we make the number of folds k vary. The higher k, the less data is available in the target training set.

gets scarce it yields lower results than the original algorithm.

Results over the RF model are similar in terms of performance of the original STRUT and SER compared with their variants. However, scores are all higher than with a DT model, which is expected, but we also notice that *Source* is more competitive with the other algorithms, since it beats *Target* (and even the STRUT algorithms family) when data is scarce. This suggests that a random forest is more robust to changes in data than a decision tree.

6.4 Conclusion over performances

SER. While SER_{LL} is quite close to the original SER, SER_{R}, which is more drastic in its behaviour, can give significantly better results but also worse, notably in one real data example. It may be explained by the fact that unlike synthetic data that is generated with simple transformations, real data is likely to involve more complex transformations, hence suggesting that a choice has to be made depending on the data one deals with.

STRUT. STRUT variants outperform the original algorithm on all data sets under all conditions. The only exception to that is STRUT_{HI} on the fall data set, when data is scarce in training. Similarly to SER variants, no particular STRUT variant is the overall best

through all experiments, also showing a certain dependency of the algorithms on the data transformations.

Finally, results with the fall data framework suggests that performance on decision trees can extrapolate to random forests.

7 Conclusion

In this chapter, we propose adaptations of two transfer learning methods on decision trees to address the imbalance class problem. The key intuition of our adaptations consists of taking into account imbalanced target data by defining the minority class leaf loss risk in the context of homogeneous class imbalance. Results show that proposed variants performance are data-dependant but, as a group, can outperform the original algorithms, i.e. , at least one of them beats the original. This suggests several areas for improvement.

Since SER_R is more a rough alternative to SER than SER_{LL} (that can be viewed as an extension), future work can consider refining this latter alternative. For example, all experiments were conducted with a fixed threshold, although it may possible to learn it from data and adjust it accordingly.

Besides, as these methods are designed for decision trees, they are applicable to tree ensembles such as random forests. Since methods are quite data-dependant and there is no clear "winner", an interesting continuation of this work would be to build a meta-model that incorporates different variants and uses a selection method (according to performances over target data) to finally transfer a whole tree ensemble. This point is of great interest in order not only to maximize performance with selection but also to adapt any tree ensemble to a target domain without prior knowledge on this latter.

Finally, one can face situations where source and target feature spaces are different. This is particularly the case in industrial applications where there is a continuous need for feature engineering. Thus, future work could investigate an adaptation of these methods to heterogeneous transfer learning.

5

Elderly activity monitoring with neural nets

Contents

Abstract

In this chapter, we propose a classification method motivated by the person monitoring in nursing homes, that discriminates between signals generated by medical staff and those generated by elderly. For that purpose, we have developed a model that includes several steps of learning procedure trained over a challenging data set of activity signals. Based on neural networks, the model first uses signal embedding based on atoms of a pre-learned dictionary and focuses the network's attention on step-related signals. We show that it is able to avoid the main limitation of floor sensors by recognizing relevant signals and ignoring events related to the medical staff. This work has been published in Minvielle and Audiffren [120].

As mentioned in the beginning of this thesis, the elderly population is prone to frailty, and maintaining a minimum level of activity is profitable to restrain frailty level from increasing [117, 185]. In this context, what may be required from a "good" monitoring system is not only to detect falls but also to monitor the general activity of elderly so that preventive actions can be taken. Floor-sensor-based systems resolve several issues concerning patient daily monitoring by being unobtrusive while providing exploitable signals. However, the main drawback of this type of sensor is its *a priori* inability to distinguish between individuals, unlike wearable devices for example that are unique to each wearer. This is particularly problematic in nursing homes, where most of the activity is generated by medical staff, especially in common areas with high traffic. Besides, even individual rooms, where identifying patients would be easier, are also subject to regular visits of caregivers or external persons (families etc.), hence making monitoring tasks more delicate.

To solve this problem, we introduce a system named Non-invasive Unit Recognition System for the Elderly (NurseNet), which combines Tarkett's piezoelectric floor sensor and a deep learning-based algorithm to distinguish elderly from staff activity. The system relies on a data set that is mainly composed of gait signals, but not restricted to that type which makes it particularly challenging.

1 Related work

1.1 Gait sensing

A large variety of sensors has been used to record walks and the quantification of gait is hardly new [10, 126, 172]. These systems are quite similar to those used for fall detection (see Chapter 2). The most popular systems lie within the wearable family, as evidenced by the large literature on the subject [10, 50], with accelerometers largely dominating the field, though there are also gait-dedicated sensors such as foot switches or pressure soles. These latter are rather designed for "one-off" diagnoses than for daily monitoring. In fact, gait sensing literature mainly covers clinical applications and most systems are not particularly suited for daily monitoring. This is the case for several wearable systems but also video-based systems [54] and some of the ambient sensing system field, such as radar [133] or Wi-Fi [89, 188], which are usually not designed for daily use. Among ambient systems, vibration sensors are less obtrusive and easier to use [57, 137] but suffer the implementation problems mentioned earlier in this thesis. Compared with fall detection, gait sensing systems make a greater use of floor-covering solutions [161], by using binary switch sensors [171, 207], capacitive sensors [78, 148, 184], and either piezoresistive [173]

or piezoelectric [151, 170] pressure sensors. When implemented, these systems have different resolutions, from only one sensor that covers a whole area to a grid of 1 cm $\times$ 1 cm tiles. However, high resolution leads to implementation issues and higher costs. Depending on the sensor and the resolution, the emitted signals can provide sufficient information for complex tasks, but can also suffer from external perturbations [161].

1.2 Gait analysis

The main families of analyses developed in previous works include: (A) the use of biomechanical features, such as single/double stance time [157]; (B) the use of descriptors derived from mathematical models [51]; and more recently, (C) the use of machine learning-based algorithms that perform end-to-end learning on gait signals [208]. Several walk detection systems rely on gait decomposition into several phases, trying to detect two to eight phases depending on the sensor and the precision level needed for their application [172]. These methods can be based on heuristic rules (e.g. thresholds [70]), signal processing techniques (e.g. Pan–Tompkins peak detection [203]), or probabilistic approaches (e.g. HMM-based algorithms [14]).

However, most methods are designed to characterize gait in a controlled environment where no challenging type of signal can infer (i.e. , signals that may look like what we want to detect). Hence, when dealing with daily activities, walk detection becomes a harder task [15, 28]. Another segment of step detection aims at recognizing a step signal as a whole (i.e. , without considering any phase in it) in order to perform walk detection within more challenging environments or to use walk signals for patient monitoring. These methods are generally based on machine learning techniques and seem more suited to such tasks [102, 209], but to the best of our knowledge, no previous work has classified activities using a floor piezoelectric sensor combined with machine learning techniques.

1.3 Convolutional dictionary learning

In recent years, dictionary learning and sparse coding techniques have been successfully applied in a wide range of topics, including image classification [114], image restoration [4], and signal processing [115]. The main idea of this representation is to linearly decompose a signal into few atoms (sparse coding) and, instead of using an already existing dictionary (wavelets for example), the dictionary is *learnt* along with the signal decomposition. More recently, its *convolutional* counterpart, known as convolutional dictionary learning (CDL) and convolutional sparse coding (CSC), gained a renewed interest with the emergence of efficient solving methods [64]. In the convolutional setting, atoms can then be seen as *shift-invariant* filters or patterns and the sparse activations encode the temporal or spatial locations where these patterns occur.

Previous works have applied *classical* dictionary learning to walk-related problems. For example, Poschadel et al. [143] used the accelerometer signal reconstructed from sparse coding to perform gait classification between healthy and movement-impaired individuals. Zhang et al. [209] also aimed at gait classification by applying specific techniques to produce a dictionary that is both representative and discriminative. Following the same idea, we use *convolutional* dictionary learning to build the first layer of NurseNet in order to construct a relevant representation. To the best of our knowledge, no previous work has used CDL for gait recognition analysis.

1.4 Region proposal network

Recently, convolutional neural networks (CNN) have been successfully used to detect objects in images [34, 68, 134], hence superseding previously existing techniques [99]. Among them, recent networks such as YOLO [145] and Faster R-CNN [146] have been shown to find and classify multiple objects in images with both high accuracy and small time cost. However, significantly less attention has been given to the application of CNN on one-dimensional signals, despite such signals being widely represented, especially in medical recordings [167]. Previous works generally transformed one-dimensional signals into two 2D representation (seen as images) using spectrograms [155] or directly used CNN without transforming the data [94].

In this work, we use the principle of the region proposal network proposed in Faster R-CNN [146] to train the intermediate layers of our model and draw its attention onto relevant parts of the signal. To our knowledge, this is the first work to combine dictionary learning, pre-training, and final training of a CNN to classify activities from time series. As shown in next sections, the combination of these three steps appears to be necessary due to the irregular nature of the signal.

2 Data set

2.1 Material

The objective of this work is to provide a tool to monitor the physical activity of elderly individuals in nursing homes. To this end, a partner retirement house was chosen to collect the signals using the following setting. A corridor and a nearby common room were equipped with the floor sensor (see Figure 5.1), and signals were recorded and labeled (event types and footsteps beginnings and ends for some of the walk signals) by an expert with the help of two cameras. The areas, a corridor and a common room, were chosen for their accessibility (i.e. ease of installation of the system and event labelling) and the great frequency of people passing through. Indeed, the corridor is surrounded by the dining room, a caregiver office, and another corridor to another aisle of the nursing home, making it an important crossing point. Signals were labeled according to the type of event (walking, rolling a cart, pushing a chair, etc.), the status of the person (elderly patient or caregiver), and the number of persons activating the sensor area.

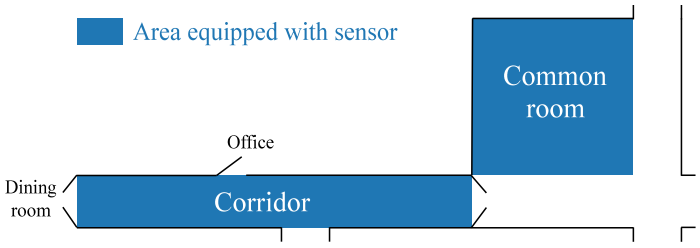


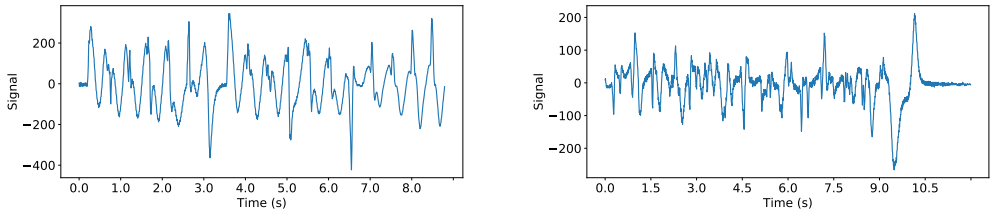
Figure 5.1: System installation in the nursing home.

2.2 Activity signals

The final data set contains 146 labeled signals with a labelling resolution of one second for event labels (e.g walk, rolling a cart, etc.) and ten milliseconds for the footsteps of the concerned signals. Table 5.1 shows the number of events of each main class. It should be noted that the majority of signals are generated by caregivers. Consequently, it is crucial to identify them as accurately as possible in order not to alter a potential elderly activity monitoring system. Figure 5.2 shows two examples of signals recorded in the nursing home, one of a staff member, the other from a resident.

Event	Number of instances
Walking, 1 person, staff	43
Walking, 1 person, elderly	31
Walking, multiple persons	30
Wheelchair (manual)	16
Wheelchair (pushed by another)	15
Wheelchair (electric)	2
Walking with a cart	7
Other events	2

Table 5.1: Number of instances in the dataset. Signals have an average duration of 10 s. Note that in the *walking, multiple persons* category, all the walkers were either all elderly or all medical staff.



(a) Walking of a staff member (SNR $\approx$ 23 dB) (b) Walking of an elderly person (SNR $\approx$ 18 dB)

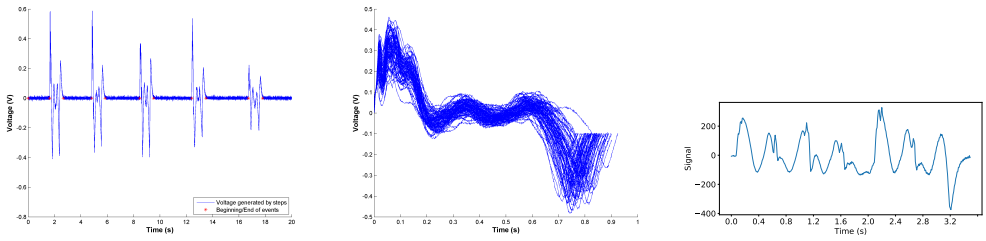
Figure 5.2: Example of signal of a young healthy person from the staff (a) and an elderly individual (b). The first observation is that the signal generated by the elderly footsteps has a significantly smaller amplitude, leading to a lower signal-to-noise ratio. We recognize some similarities between these signals as step patterns, as can be seen in (b). However as a whole, the (b) signal seems less repeatable than (a). This is probably due to the fact that the walking of an elderly person can be less regular and less pronounced than a young healthy person whose feet leave and hit the floor at a regular pace.

2.3 Laboratory conditions vs. reality

Previous works conducted within Tarkett research team using the piezoelectric floor sensor showed examples of footstep recordings that proved to be very reliable in their interpretability [161]. For instance, different phases of a footstep could be seen: heel, toe, weight transfer, and foot removal. Moreover, these footsteps were recorded multiple times

with different persons, showing that the generated signal was repeatable, illustrating it with a superimposition of several footsteps of one person (see Figure 5.3). However, when implemented, the output was of significantly lower quality. Multiple factors may account for this difference. Serra et al. [161] recorded their signals in a *controlled environment*: a single band was directly linked to an acquisition unit dedicated to laboratory prototyping. Hence, the signal was not altered by any additional component that might be present in a processing system, and the acquisition unit has a specific amplifier that provides a highly accurate signal. These laboratory conditions lead to high-quality outputs, and events such as walks could be easily decomposed.

In real conditions, due to scale and cost issues, the signal is collected on multiple large surfaces and sent through a processing unit designed to perform multiple tasks including the amplification of the signal, which may alter the quality of the data. Besides, with electronic components comes a noise that is constantly active. Moreover, in addition to previously-acknowledged limitations (i.e. , non-homogeneousness piezoelectric coefficient across space), the system appeared to be sensitive to humidity. Indeed, in this setting, the signal was collected through crimps hooked to the sensor and linked to a printed circuit. This connection was vulnerable to humidity that may be in the surfaces in contact with the system (e.g., adjacent walls), which may even lead to corrosion. Besides, this humidity may enter in contact with the edges of the sensor, thus creating an additional resistance between the two sides of the device. Depending on the characteristics of the humidity, these two phenomenons may significantly alter the output of the unit. Figure 5.3 illustrates the difference between signals acquired in laboratory conditions and in real environment (nursing home).



(a) Example of footsteps recording in the **laboratory** environment. Experiments were done such that each event is only one step on the sensor

(b) Superimposition of 100 steps from the same person recorded in the **laboratory** environment

(c) Example of a signal of a walk of a young healthy person recorded in the nursing home

Figure 5.3: When considering signals recorded in laboratory conditions (a and b), we notice a regularity and good repeatability of events that are not of the same quality when looking at signals taken in the nursing home (c). Figure (a) and (b) were taken with author's agreement [161].

Following sections present our proposed neural network named NurseNet, which is used to identify and characterize activity signals between medical staff and elderly individuals. We first describe the general structure of NurseNet and its main training (Section 3). Then, we describe the data embedding that is used to encode the signals (Section 4), and finally, the subnetwork used as the intermediate layers of NurseNet is explained (Section

5). Most of the strategies developed in the training process aim at circumventing our main issues, which are the challenging signals that aim to classify, and the alterations of the signal due to the real conditions.

3 Classification model

3.1 General classifier

This section details the architecture of NurseNet and its inner workings. The main assumption that guided our work is that regardless of the type of activity recorded on the floor sensor, it is very likely to be mostly made of walks. A significant part of the training process aims at training the network to recognize gait-related signals and in particular gait of the medical staff. The structure of NurseNet is presented in Figure 5.4. It is a convolutional neural network composed of:

- One 1D convolutional layer with 32 filters of size 60, with a stride of 10, followed by batch normalization (BatchNorm), with the activation function hyperbolic tangent (tanh).
- One 1D convolutional layer with 16 filters of size 1, with a stride of 1, followed by BatchNorm, with the activation function tanh.
- One 1D convolutional layer with 8 filters of size 1, with a stride of 1, followed by BatchNorm, with the activation function tanh.
- One 1D convolutional layer with 1 filter of size 5, with a stride of 1, with the activation function Rectified Linear Unit (ReLU) [127]. It is followed by a Maxpool layer of size 5.
- One fully-connected layer, with an output of dimension 64, with activation function ReLU.
- One fully-connected layer, with an output of dimension 16, with activation function ReLU.
- One fully-connected layer, with an output of dimension 1, with activation function sigmoid.

The output of the network is the probability for a signal to be generated from staff activity. Note that the input of NurseNet is not the raw signal, but instead a transformation of it (see Section 4).

3.2 Sub-networks

NurseNet can be seen as the combination of two sub-networks. The first one, named the step proposal network (SPN), is made of the first three convolutional layers. SPN is a full CNN inspired by the region proposal network, which is the first part of the Faster R-CNN algorithm [146]. At the end of this sub-network, the output is constituted of features computed on each window of the signal embedding.

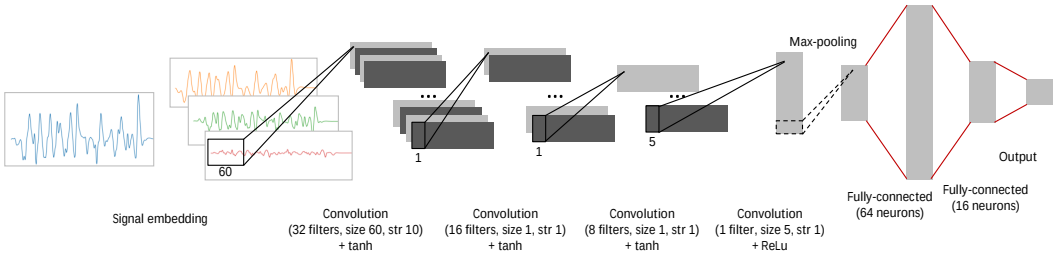


Figure 5.4: Architecture of the overall network.

The second part of the network is made of a convolutional layer, a Max Pooling operation and three classical fully-connected layers. The role of this last part of the network is to use the previously extracted features to classify signals as *staff* or *elderly*.

3.3 Training

As mentioned before, the training of NurseNet is rather complex and involves multiple stages. This is due to the small signal-to-noise ratio, the perturbations discussed earlier, and the limited size of the data set. The first phase of the training is to learn the embedding of the data. This is detailed in Section 4. Then, SPN is trained separately on a different task, in order to take advantage of the gait structure (see Section 5). The resulting weights are used to initialize the first three layers of NurseNet, and these layers are frozen during the rest of the training process to limit the number of free parameters, henceforth reducing overfitting [204]. Finally, the rest of the network is initialized using i.i.d. centered Gaussian variables with a standard deviation of 0.2. For the training phase, regularization is done using dropout (with $p = 0.5$) after each fully-connected layer, except the last one [163].

Given an input signal $\mathbf{s}_i \in \mathbb{R}^T$, let $\hat{y}_i$ denote the output of $\mathbf{s}_i$ through the embedding procedure followed by the classification network, and y_i the corresponding label ($y_i = 0$ if and only if $\mathbf{s}_i$ is generated by elderly activity). The associated loss function we used is the binary cross entropy (or log loss):

$$\mathcal{L}(\mathbf{s}_i, \hat{y}_i) = -\big(\mathbb{1}_{y_i=1} \log(\hat{y}_i) + \mathbb{1}_{y_i=0} \log(1 - \hat{y}_i)\big) \ . \tag{5.1}$$

The weights of NurseNet are optimized using backpropagation and gradient descent with a learning rate of 10^{-5}, decreasing geometrically ($\times 0.9$) every 10 epochs, and the momentum strategy proposed by Nesterov [128]. More details on the training of neural nets are given in Appendix C.

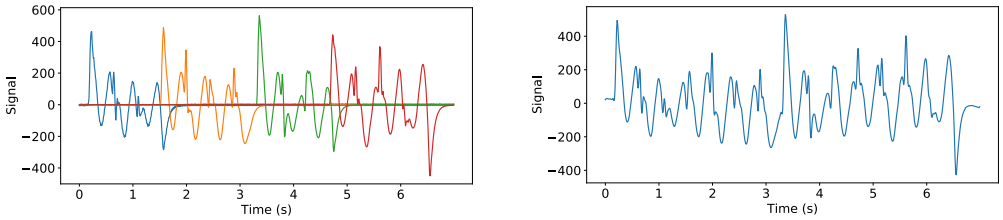
4 Signal embedding

The first step of our classification algorithm is the data embedding, which consists successively of preprocessing the data and encoding the resulting signal using convolution with atoms of a pre-learned dictionary. The preprocessing phase is made of a classical tool that aims at denoising and cleaning the data. However, our approach of data embedding

is rather new: while we use convolutional sparse coding to learn the custom dictionary, which is a standard approach since sparsity produces relevant atoms (see [156] and the references therein), the actual embedding of the data is done with simple convolution with the atoms, which yields non-sparse signals.

The reasoning behind this approach is as follows. It has been shown that the first layers of a CNN tend to learn general feature extractors: for example, in image processing, the first layers usually exhibit features similar to Gabor filters and color blobs [97]. The aim of our data embedding phase is to replace the first layers of the CNN with some already trained filters, here the atoms of the dictionary. Since the training is done on a small data set, reducing the number of layers in the network may improve the results. Hence, in order to mimic the behavior of the first layers of the CNN, regular convolution is used instead of convolutional sparse coding. All data embedding details are given below.

Preprocessing. It is known that preprocessing improves the performance of CNN networks [136], particularly when dealing with a small labeled data set. The preprocessing of data is done exactly the same as described in Section 1.3 on page 63, with the only difference being that we do not recover the opposite of sum but the sum, since the signal was already inverted in the processing unit used for data collection. Figure 5.5 shows an example of the resulting signal.



(a) Raw signal (b) Preprocessed signal

Figure 5.5: Example of raw (a) and preprocessed (b) signals resulting from medical staff walking on the sensor. The person is crossing the corridor from one end to the other, hence activating the channels one after the other. The walk signal is fairly regular with recognizable patterns (steps), however, each beginning / ending contact with a channel results in particular responses of the sensor which alters the regularity of the final signal.

Signal embedding. Data augmentation is a key part of training complex neural networks, particularly when only small labeled datasets are available [181]. However, these techniques are difficult to apply in our setting, as it is unclear that commonly-used transformations would preserve the nature and structure of the data. For instance, a re-scaled step signal may be confused with a fall signal, as the amplitude of the applied force is the main difference between the two signals. In our case, we are dealing with a small data set, which is composed of complex signals, thus making end-to-end learning a difficult task. To circumvent this difficulty and for the aforementioned reasons, data is first transformed using convolution with atoms learned from convolutional dictionary learning in order to extract key elements and features of the signal. This first phase is intended to improve the quality of the CNN input to balance out the limited amount of available data.

4.1 Sparse convolutional dictionary learning

4.1.1 General framework

In order to learn relevant atoms to our step description process, we use a convolutional dictionary learning (CDL) approach on a subset of walk signals. CDL is defined as the learning of a set of atoms to best input signals (e.g. images or one-dimensional signals) in a shift-invariant way while insuring a sparse representation. Given a vector $\mathbf{s}$ that contains data to be represented, the objective is to find M atoms $\mathbf{d}_m$ and activation signals $\mathbf{x}_m$ such that the reconstruction is accurate, i.e. $\mathbf{s} \approx \sum_m \mathbf{x}_m * \mathbf{d}_m$, with $*$ denoting the convolution. CDL general problem is formalized as:

$$\begin{aligned} \underset{\mathbf{x}_m, \mathbf{d}_m}{\arg\min} \frac{1}{2} \left\| \sum_{m=1}^{M} \mathbf{x}_m * \mathbf{d}_m - \mathbf{s} \right\|_2^2 + \lambda \sum_{m=1}^{M} \|\mathbf{x}_m\|_1 \\ \text{s.t.} \quad \|\mathbf{d}_m\|_2 \leq 1 \quad \forall m \ . \end{aligned} \tag{5.2}$$

While the ℓ_1-norm over the activations $\mathbf{x}_m$ insures sparsity, the norm constraint over the atoms prevents them from "attracting" all the energy of the signal. Indeed, scaling atoms $\mathbf{d}_m$ by $\alpha > 1$ and activations by $1/\alpha$ do not change the reconstruction error (first term of (5.2)) but multiply the regularization term (second term of (5.2)) by $1/\alpha$. Therefore, without the unit constraint, $\mathbf{x}_m$ tend towards 0 and the norm of $\mathbf{d}_m$ explodes.

The standard procedure of dictionary learning is to use Alternate Minimization, that is, alternating between a sparse coding step (i.e. , updating the sparse representation of the data according to the current dictionary) and a dictionary update (i.e. , updating the dictionary according to the current sparse representation). This means that (5.2) is solved successively with $\mathbf{d}_m$ fixed (sparse coding) or $\mathbf{x}_m$ fixed (dictionary update). We used the method of Bristow et al. [32], which is derived from the Alternating Direction Method of Multipliers (ADMM) [27]. Adapted to sparse coding and dictionary update, this method proved to be efficient among the existing ones [64]. We briefly recall their resolution of the two CDL subproblems that are the convolutional sparse coding and the dictionary update.

4.1.2 Solving CDL with ADMM.

Convolutional sparse coding. The convolutional sparse coding problem is expressed as:

$$\underset{\mathbf{x}_m}{\arg\min} \frac{1}{2} \left\| \sum_{m=1}^{M} \mathbf{x}_m * \mathbf{d}_m - \mathbf{s} \right\|_2^2 + \lambda \sum_{m=1}^{M} \|\mathbf{x}_m\|_1 \ , \tag{5.3}$$

where $\mathbf{d}_m$ are fixed and the objective is to find activations $\mathbf{x}_m$.

In their method, Bristow et al. [32] use the ADMM, which solves problems of the form:

$$\underset{x,y}{\arg\min} f(x) + g(y) \quad \text{s.t.} \quad Ax + By = c \tag{5.4}$$

by using the augmented Lagrangian multipliers. Following ADMM framework (5.4), the CSC problem (5.3) can be rewritten:

$$\underset{\mathbf{x}_m, \mathbf{t}_m}{\arg\min} f(\mathbf{x}_m) + g(\mathbf{t}_m) \quad \text{s.t.} \quad \mathbf{x}_m = \mathbf{t}_m \ ,$$

with

$$f(\mathbf{x}_m) = \frac{1}{2}\left\|\sum_{m=1}^{M}\mathbf{x}_m * \mathbf{d}_m - \mathbf{s}\right\|_2^2 \quad \text{and} \quad g(\mathbf{t}_m) = \lambda\sum_{m=1}^{M}\|\mathbf{t}_m\|_1 \ .$$

ADMM solution is obtained by iterating over the steps:

$$\mathbf{x}_m^{(i+1)} = \arg\min_{\mathbf{x}_m} f(\mathbf{x}_m) + \frac{\rho}{2}\sum_{m=1}^{M}\|\mathbf{x}_m - \mathbf{t}_m^{(i)} + \mathbf{u}_m^{(i)}\|_2^2 \tag{5.5}$$

$$\mathbf{t}_m^{(i+1)} = \arg\min_{\mathbf{t}_m} g(\mathbf{t}_m) + \frac{\rho}{2}\sum_{m=1}^{M}\|\mathbf{x}_m^{(i+1)} - \mathbf{t}_m + \mathbf{u}_m^{(i)}\|_2^2 \tag{5.6}$$

$$\mathbf{u}_m^{(i+1)} = \mathbf{u}_m^{(i)} + \mathbf{x}_m^{(i+1)} + \mathbf{t}_m^{(i+1)} \ . \tag{5.7}$$

While step (5.7) is straightforward, step (5.6) admits the closed-form solution:

$$\mathbf{t}_m^{(i+1)} = \mathcal{S}_{\lambda/\rho}\left(\mathbf{x}_m^{(i+1)} + \mathbf{u}_m^{(i)}\right) \ ,$$

where $\mathcal{S}_\gamma$ denotes the soft-thresholding function:

$$\mathcal{S}_\gamma(\mathbf{x}) = \text{sign}(\mathbf{x}) \odot \max(0, |\mathbf{x}| - \gamma) \ ,$$

with $\odot$ denoting the element-wise multiplication (sign$(\cdot)$ and $|\cdot|$ are applied element-wise) [32, 194].

The solution of (5.5) is less straightforward. Bristow et al. [32] propose to use two convenient properties: the Parseval theorem and the convolution in Fourier's domain. Step (5.5) is rewritten in Fourier's domain:

$$\widehat{\mathbf{x}}_m^{(i+1)} = \arg\min_{\widehat{\mathbf{x}}_m} \frac{1}{2}\left\|\sum_{m=1}^{M}\widehat{\mathbf{x}}_m \odot \widehat{\mathbf{d}}_m - \widehat{\mathbf{s}}\right\|_2^2 + \frac{\rho}{2}\sum_{m=1}^{M}\|\widehat{\mathbf{x}}_m - \widehat{\mathbf{t}}_m^{(i)} + \widehat{\mathbf{u}}_m^{(i)}\|_2^2 \ , \tag{5.8}$$

and the element-wise product $\widehat{\mathbf{x}}_m \odot \widehat{\mathbf{d}}_m$ can be rewritten as the matrix product $\widehat{\mathbf{D}}\widehat{\mathbf{x}}$, where $\widehat{\mathbf{D}} = [\text{diag}(\widehat{\mathbf{d}}_1), \ldots, \text{diag}(\widehat{\mathbf{d}}_M)]$. The solution of (5.8) is then:

$$\widehat{\mathbf{x}}_m^{(i+1)} = \left(\widehat{\mathbf{D}}^T\widehat{\mathbf{D}} + \rho\mathbf{I}\right)^{-1}\left(\widehat{\mathbf{D}}^T\mathbf{s} + \rho\left(\widehat{\mathbf{t}}_m^{(i)} + \widehat{\mathbf{u}}_m^{(i)}\right)\right) \ ,$$

and $\mathbf{x}_m^{(i+1)}$ is finally retrieved using the inverse Fourier transform over $\widehat{\mathbf{x}}_m^{(i+1)}$.

Dictionary update. The second step of the main framework of CDL is the dictionary update, which consists of learning the dictionary. The problem is formalized as:

$$\arg\min_{\mathbf{d}_m} \frac{1}{2}\left\|\sum_{m=1}^{M}\mathbf{x}_m * \mathbf{d}_m - \mathbf{s}\right\|_2^2 \quad \text{s.t.} \quad \|\mathbf{d}_m\|_2 \leq 1 \quad \forall m \ . \tag{5.9}$$

This problem can also be solved with ADMM [32]. For that purpose, the method uses an indicator function $\mathcal{I}_{\mathcal{C}}$ where $\mathcal{C}$ is the constraint set $\{\mathbf{x} : \|\mathbf{x}\|_2 \leq 1\}$. The dictionary update problem can be rewritten:

$$\arg\min_{\mathbf{d}_m} \frac{1}{2} \left\| \sum_{m=1}^{M} \mathbf{x}_m * \mathbf{d}_m - \mathbf{s} \right\|_2^2 + \mathcal{I}_{\mathcal{C}}(\mathbf{d}_m) \ ,$$

and, following the ADMM framework, it becomes:

$$\arg\min_{\mathbf{d}_m, \mathbf{t}_m} f(\mathbf{d}_m) + g(\mathbf{t}_m) \quad \text{s.t.} \quad \mathbf{d}_m = \mathbf{t}_m \ ,$$

with

$$f(\mathbf{d}_m) = \frac{1}{2} \left\| \sum_{m=1}^{M} \mathbf{x}_m * \mathbf{d}_m - \mathbf{s} \right\|_2^2 \quad \text{and} \quad g(\mathbf{t}_m) = \mathcal{I}_{\mathcal{C}}(\mathbf{t}_m) \ .$$

Then, similarly to the CSC problem, ADMM iterates over three steps, (minimizing the augmented Lagrangian with $\mathbf{d}_m$ and $\mathbf{t}_m$ and maximizing it with the dual $\mathbf{u}_m$).

4.2 Learning step atoms

Previous works have shown that many walk alterations can be detected and quantified by studying the characteristics and variety of steps [150]. Following this idea, dictionary learning (DL) has been successfully used to quantify and evaluate gait [143, 209]. However, it is worth noting that DL is generally used on signals derived from accelerometers, which have been proven to be a robust and reliable way to record walks. In our setting, the recording is of lower quality, and nearly-identical steps, such as the ones produced by the walking of young healthy individual [63], are recorded with significant variations between them (see Figure 5.5). Besides, a large number of signals in the data set contains elements that are not related to steps (e.g., pushing a wheelchair or a cart). Finding a suitable dictionary is therefore significantly more challenging, and this is why atoms were created by selecting signals that are almost entirely walk-related.

We used the implementation of Wohlberg [195] of the algorithm, available in a Python package named SPORCO (SParse Optimization Research COde). The number of atoms M was considered a hyperparameter of our CDL problem. Our experiments showed that $M = 3$ achieved the best trade-off between the quality of the reconstruction while maintaining the sparsity of the representation. Using less atoms led to either poor reconstruction or non-sparse representation, whereas using more atoms significantly decreased the reconstruction quality. These three atoms are 0.7 second long, which corresponds to an upper bound of the duration of a step in the walking of a healthy young individual [81]. Figure 5.6 shows the resulting dictionary.

4.3 Signal encoding

As discussed above, while we use a sparsity constraint on the dictionary learning process, the actual embedding of the input data $\mathbf{s}$ into $\mathbf{S}$ is done with standard convolution:

$$\mathbf{S} \doteq \left(\mathbf{s} * \mathbf{d}_m \right)_{1 \leq m \leq 3} \ , \tag{5.10}$$

resulting in a three-channel encoding of the signal ($\mathbf{S} \in \mathbb{R}^{T \times 3}$, where T is the size of the input signal $\mathbf{s}$). Figure 5.7 shows an example of the data encoding process.

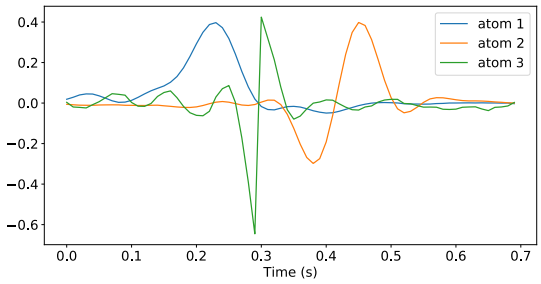
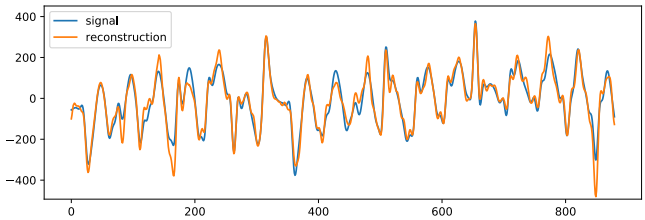
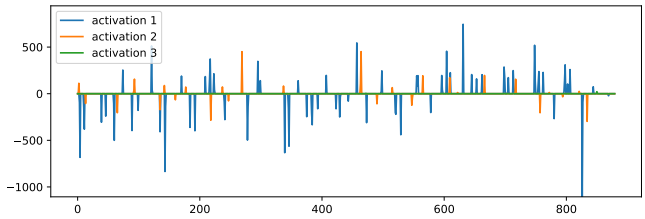


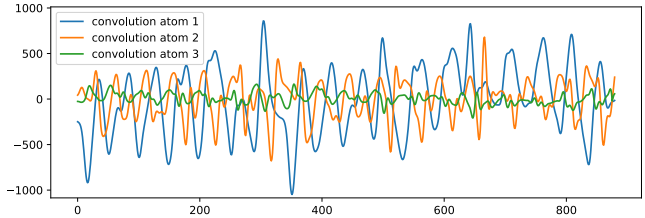
Figure 5.6: Dictionary learned using Equation (5.2). Note that the amplitude of the atoms is small compared to the signal due to the fact that they are normalized in the training process.



(a) Original signal and its reconstruction



(b) Signal activations



(c) Embedding

Figure 5.7: Signal embedding. A signal and its reconstruction by convolutional sparse coding are displayed (a). The activations (b) show the "presence" of each atom in the original signal. As the embedding step, we use the convolution of the signal with each atom (c). It is interesting to note that since atom 1 is mildly similar to a Gaussian kernel, the result of the convolution of the signal with atom 1 is close to a smoothed version of the original signal.

5 Step detection with region proposal network

This section discusses the idea behind the first sub-network of NurseNet and presents its dedicated training process.

5.1 Region proposal network

As discussed before, this sub-network is inspired by the region proposal network (RPN). RPN's role is to help the main network of Faster R-CNN [146] focus only on interesting parts of an image. To this end, a sliding window is passed on a feature map of the input image. On this sliding window, boxes of different scales and ratios (in total K) are used as region proposals, and these regions are evaluated through two layers. The first one outputs the probability of a box to be an object and the other one gives corrected coordinates of the box. This method reduces the number of false examples that may mislead the training process and greatly accelerates the item search. In our case, since we are dealing with one-dimensional signals (hence significantly smaller than images), we were not especially seeking high speed execution. However in this approach, steps are considered as a key element to distinguish between medical staff and elderly activities. Therefore, by training a network to identify and characterize steps, the idea is to ensure that most of the information that is passed to the second sub-network of NurseNet is relevant. Moreover, as this step recognition part brings a separate training, it adds a constraint to the overall model, hence reducing the chances of overfitting (see [147] for a discussion related to constraints and overfitting).

The architecture of the CNN that is used to train the first sub-network of NurseNet is presented in Figure 5.8. This CNN is made of:

- One 1D convolutional layer with 32 filters of size 60, with a stride of 10, followed by BatchNorm, with the activation function hyperbolic tangent (tanh).
- One 1D convolutional layer with 16 filters of size 1, with a stride of 1, followed by BatchNorm, with the activation function tanh.
- One 1D convolutional layer with 8 filters of size 1, with a stride of 1, followed by BatchNorm, with the activation function tanh.
- One 1D convolutional layer with 3 filters of size 1, with a stride of 1, with the activation function sigmoid.

5.2 Training a neural net to detect steps

Step boxes and the associated loss function. The training of this CNN was done as follows. For this task, we selected a subset of signals that only included easily-identifiable walks of the medical staff. Each of these signals was manually segmented by an expert using external sensor information, such as the video recordings of the equipped area. More precisely, boxes (denoted $\mathbf{b}_j$) were manually labeled for each step of each embedded signal. Figure 5.9 gives an example of a step signal and delimiting boxes after labelling. The

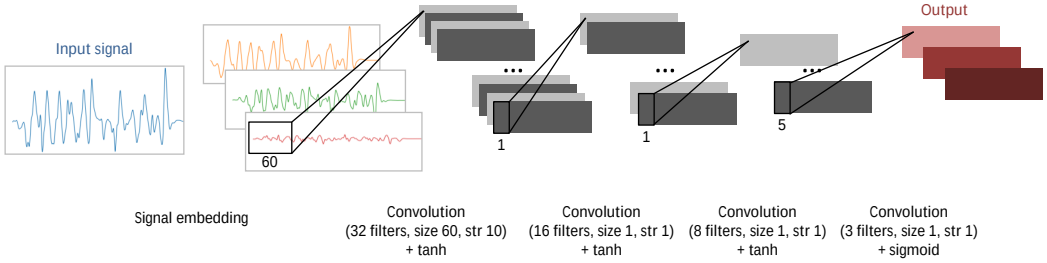
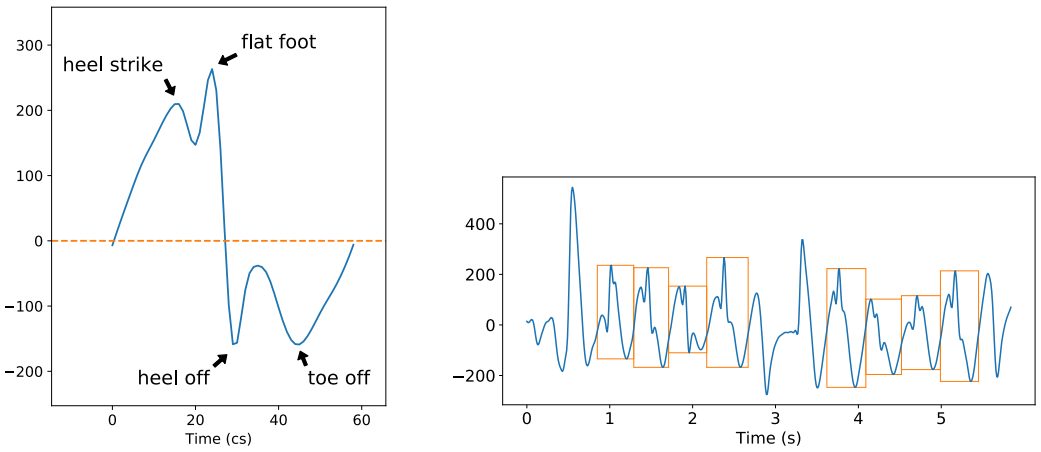


Figure 5.8: Architecture of the first part of the network, the SPN. Note that the last layer, which gives the output, is not present in the general NurseNet algorithm (Figure 5.4).



(a) Step signal

(b) Step labels over a walk signal

Figure 5.9: Example of a typical step (a) and true steps delimiting boxes in a walk signal (b). In this example of a step signal (a), we can separate the different phases of the step from heel strike to toe off. However, signals are in fact subject to variations, making the step detection task challenging (b). During labelling, only parts of the signal that can be undoubtedly connected to a step are selected.

purpose of the SPN is to produce boxes $\hat{b}$ with a large Intersection over Union (IoU) score, which is defined as following:

$$\mathrm{IoU}(\hat{b}) \doteq \max_j \frac{|\mathbf{b_j} \cap \hat{b}|}{|\mathbf{b_j} \cup \hat{b}|}, \tag{5.11}$$

meaning that SPN is trained to output a set of boxes in which each box is nearly equal to at least one of the ground-truth boxes. More precisely, the output of the network is a matrix $\mathbf{W} \in \mathbb{R}^{T \times K}$, with T being the signal length and K the number of different box sizes. Here, $\mathbf{W}_{t,k}$ is interpreted as the probability that the box b_t^k starting at time t and of size 40, 50, or 60 (for respectively $k = 1$, 2, or 3) has a large IoU score. Theses sizes are chosen according to typical step durations of young healthy individuals. In line with Ren et al. [146], we define positive instances of boxes, i.e. , b_t^k such that $\mathrm{IoU}(b_t^k) > \sqrt{0.7}$, and negative instances, i.e. , b_t^k such that $\mathrm{IoU}(b_t^k) < \sqrt{0.3}$. It should be noted that in

the original paper, authors used 0.7 (resp. 0.3) for the upper threshold (resp. the lower threshold). However, in their case, IoU was computed between 2D boxes, instead of 1D in our case. All other boxes were considered neutral boxes and did not participate in the training process. The loss function $\mathcal{L}$ over a signal $\mathbf{s}$ is then defined as:

$$\mathcal{L}(\mathbf{s}, \mathbf{W}) = \sum_{t} \sum_{k \in [1,2,3]} \mathbb{1}_{\mathrm{IoU}(b_t^k) > \sqrt{0.7}} \log(\mathbf{W}_{t,k}) + \mathbb{1}_{\mathrm{IoU}(b_t^k) < \sqrt{0.3}} \log(1 - \mathbf{W}_{t,k}). \quad (5.12)$$

Training process. The weights of each layer were initialized using i.i.d. centered Gaussian variables with a standard deviation of 0.2. Regularization was done using dropout (with $p = 0.5$) after each BatchNorm layer. Equation (5.12) was optimized using gradient descent with a learning rate of 10^{-3}, decreasing geometrically ($\times 0.9$) every 10 epochs, and the Nesterov momentum strategy. The learning was monitored and stopped before over-fitting.

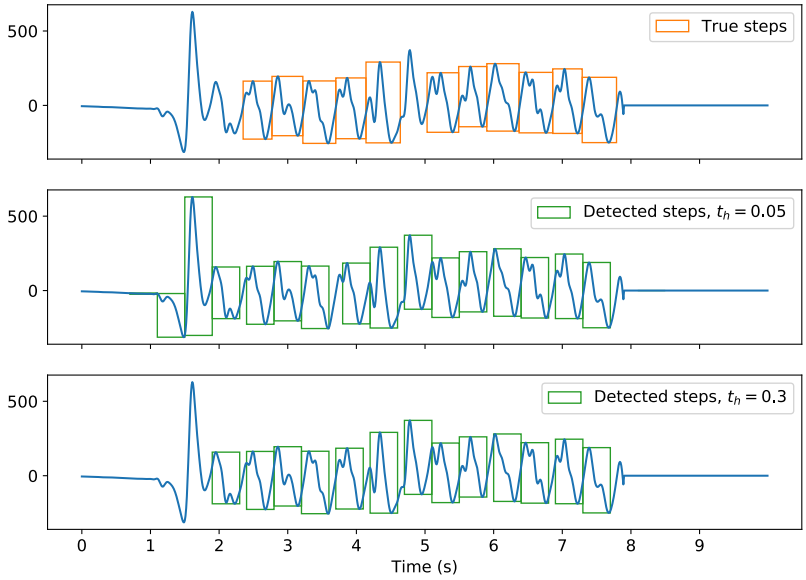


Figure 5.10: Test of SPN over a walk signal, with top figure showing the true labels. Results are displayed after applying a threshold over the output score of SPN and removing overlapping boxes (middle and bottom figures). With $t_h = 0.3$ results are satisfying, with only two steps more than the true labels.

Results of step detection. Once SPN is run over a signal, there are multiple stages towards the final detection of steps. First, a threshold t_h is passed over scores $\mathbf{W}_{t,k}$. From the remaining set of boxes, overlapping boxes are removed, making sure that the ones with the highest scores are kept. The average precision (AP) can then be computed in function of a threshold over the IoU (see Appendix C for more details). Figure 5.10 shows an example of the result of SPN over a test signal. The step detector achieved good results: for an IoU threshold of 0.7 (resp 0.9), it achieved an AP of 85.2% (resp. 55.3%).

6 Results

In this section, we thoroughly evaluate the performance and behavior of NurseNet on multiple types of signals. First, the accuracy of NurseNet is studied over each subgroup of our database, and we take a closer look at misclassified signals. Then, a complete ablation analysis is done to show the relative improvement added by each part of the algorithm, and our approach is compared to a random forest model trained with different feature vectors. The algorithm is tested and run with a Python implementation on a laptop with a 2.4-GHz Intel Core i7 processor. The training of the several stages (i.e. , CDL, SPN, and NurseNet) takes about an hour, however, the complete inference process is fast (nearly instantaneous for a 10-s signal).

6.1 Performance evaluation

The algorithm is evaluated by performing stratified random splits of the dataset into training and testing sets using 5 folds, resulting in similar distributions of labels in all sets. These splits are done with similar distributions of the three following sublabels: *single walk* – where one individual is walking on the sensor, *multiple walks* – where strictly more than one individual is walking at the same time on the sensor, and *others* – where one non-walk-related event occurs (e.g., pushing a wheelchair or a cart). The demographics are detailed in Table 5.2.

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

Table 5.2: Repartition of signals across main labels (*staff* / *elderly*) and sublabels.

NurseNet achieves a global AUC of 0.93 for the classification between staff and non-staff activities. This high performance is evenly distributed on all labels, as shown by Figure 5.11. Table 5.3 presents two examples of confusion matrices obtained by NurseNet for different thresholds, and Figure 5.12 shows examples of misclassified signals. These signals are all hard to discriminate, which is primarily due to the variety of events that we aim to classify – guessing the age group of multiple persons walking or persons pushing wheelchairs is by nature a difficult task, and secondly to the event itself – contrary to the previous example of a walk signal generated by an elderly person (Figure 5.2), the signal of 5.12(b) was recorded from a healthier individual, hence a lot more challenging for the classification task.

6.2 Ablation Analysis

This subsection aims to perform an ablation analysis on NurseNet and compare it to other versions of itself (in which some parts are "deactivated"). To achieve this result, three other versions of NurseNet are trained.

1. **No embedding**. A version without the embedding of the input signal, i.e. the model is fed directly with the raw signal (instead of its convolution with step atoms).

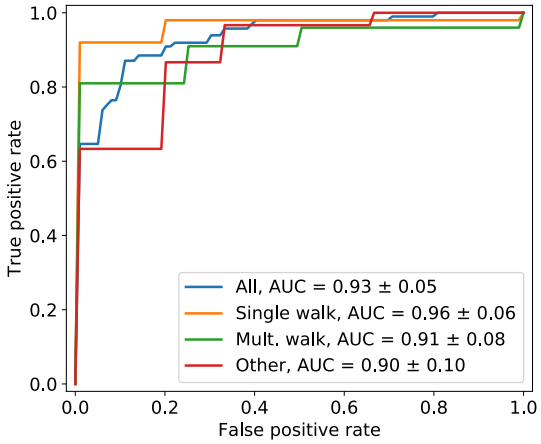


Figure 5.11: ROC obtained when classifying *staff*/*elderly* for different activities (single individual walk, multiple individuals walks, other) using NurseNet over stratified folds. Mean AUCs over the folds are displayed in the legends.

	Staff			**Elderly**		
	Single walk	**Multiple walks**	**Other**	**Single walk**	**Multiple walks**	**Other**
			$\tau = 0.5$			
Classified Staff	41	15	14	4	1	7
Classified Elderly	2	4	5	27	10	16
			$\tau = 0.7$			
Classified Staff	38	13	13	3	1	5
Classified Elderly	5	6	6	28	10	18

Table 5.3: Confusion matrix for different thresholds τ, i.e. , activity is classified as *staff* if the network produces an output greater than or equal to τ. Among staff activities, we note that the *staff* vs. *elderly* classification is the hardest for the *multiple walks* and *other* groups.

2. **No SPN**. A version without the SPN pre-training part (but embedding is still used), meaning that the network is trained without "freezing" the first layers.

3. **No embedding & No SPN**. A version in which no SPN is used nor embedding, i.e. the whole network is trained from scratch on the final classification task.

Finally, NurseNet is compared to three models based on the random forest (RF) algorithm. All three models are RF made of 30 fully grown decision trees trained with CART. These models differ in the way the signal is processed before a vector is passed to the RF.

1. **RF + raw**. The signal is passed directly as the input vector for the RF (with a 10-second padding for homogeneity), thus resulting in a vector of $10 \times 100 = 1000$ dimensions.

2. **RF + embedding**. The 10-s padded signal is decomposed using the embedding with the atoms learned in Section 4, thus resulting in a vector of $10 \times 100 \times 3 = 3000$ dimensions.

(a) Medical staff walking

(b) Elderly walking

(c) Medical staff walking (two persons)

(d) Elderly walking (two persons)

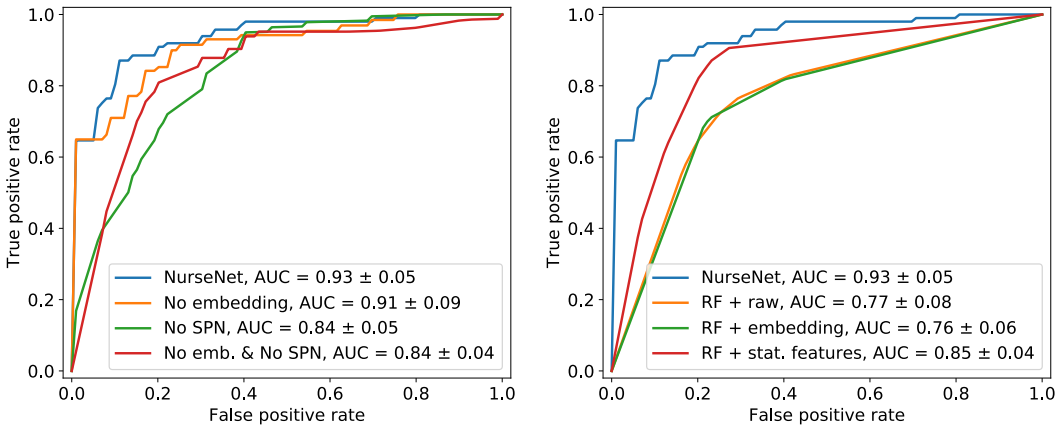
(e) Medical staff pushing a wheelchair

(f) Elderly using an electric wheelchair

Figure 5.12: Examples of signals misclassified by NurseNet with their output score (the score can be seen as the probability, according to the model, that the signal belongs to the *staff* class). Left column (resp. right column) shows signals generated by medical staff (resp. elderly) and each row corresponds to a sublabel (i.e. *single walk*, *multiple walks*, and *other*). We note that misclassified *staff* signals present irregularities, while misclassified *elderly* signals have unusually high amplitudes.

3. **RF + stat. features**. The 10-s padded signal is transformed into a vector of statistical features, following the procedure described in section 3.2 on page 68, hence resulting in a vector of 87 dimensions.

ROC curves and corresponding AUCs of all the described models are presented in Figure 5.13. Regarding the ablation analysis, results show that while the embedding part slightly improves results of the complete model, the SPN brings a larger improvement, hence illustrating the advantages of our training method. We note that NurseNet without embedding nor SPN, which is a CNN directly applied to the raw signal, achieves the surprisingly high AUC of 0.84. When compared to random forests, NurseNet significantly outperforms all RF-based models, the most performing of these latter being the one trained with our set of statistical features, which is quite expected since a supervised model such as a RF is meant to work on "relevant" variables and not on the raw signal. We note that not only NurseNet but also its ablation versions achieve higher AUCs than RF + stat. fea-



(a) NurseNet ablation analysis (b) NurseNet and several RF-based models

Figure 5.13: Classifying *staff*/*elderly* with ablation over NurseNet model (a) and comparing it with random-forest-based models (b). The corresponding AUCs are displayed in the legends. Note that NurseNet outperforms all the other algorithms.

tures, thus suggesting that the features learnt by the first layers are well adapted to the classifying task.

7 Discussion

Classification across all labels. It is interesting to note that NurseNet performs well across all sublabels. This might seem counter-intuitive at first as the algorithm is pretrained towards detecting steps and some signals contain non-gait-related activities (e.g. pushing a cart). A possible explanation comes from the fact that all signals contain some elements of walks. In the example where a staff member is pushing a cart, while not obvious to the untrained eye, it is possible that some characteristics of the medical staff walk are encoded in the signal. Therefore, the features learned by the SPN can be relevant to correctly assess the floor signals.

Multiple training. Interestingly, the CNN without the dictionary encoding nor SPN pretrainings gives same performances as the CNN with the sole dictionary encoding. This may be explained as follows. Both the dictionary encoding and the SPN are two complementary stages of step detection: the dictionary extracts features linked to step characteristics, and the SPN is trained to use these features to segment the signal into steps. On the one hand, when the CNN is used with only dictionary embedding, the training task becomes more complex as the network has to learn how to use these features to recognize steps while trying to classify these signals. On the other hand, the regular CNN algorithm can use any non-step-related features to perform the classification. Consequently, its objective function is significantly less constrained and has a larger number of reasonable local minima, making it easier to train. However, it should be noted that when properly guided toward a step-related solution (NurseNet), the CNN appears to perform even better, which emphasizes the interest of our approach.

NurseNet vs. shallow. On this classification task, the random forest model trained

over the raw or embedded signals gives low performance (AUC 0.77 and 0.76). This may be explained by the complex nature of the signal: while NurseNet (as well as other deep network algorithms) can learn relevant features to describe the signal, shallow algorithms tackle a significantly harder task. We observe that with properly designed features, the RF achieves better results (AUC 0.85). However, it is still outperformed by NurseNet (AUC 0.93).

In-depth classification. As seen in Section 6, NurseNet is able to separate medical staff and elderly activities with very high accuracy (AUC 0.93). Future work may focus on separating different categories of activity as each one of them entails different indicators regarding elderly activity monitoring. However, this task requires a significantly larger data set, and therefore, the manual labelling of far more numerous sequences of data.

Individual identification. While NurseNet achieves very high accuracy for classifying elderly versus medical staff, it is not able at the moment to recognize individuals. This is due to the fact that the data set only includes a handful of signals for each individual. However, we believe that with significantly more data, this method can be improved to distinguish between individuals, although at a lower accuracy than the *staff*/*elderly* separation. Future work may aim at improving NurseNet in that regard.

Conclusion and perspectives

This thesis explores contributions to a floor-based system in the context of elderly monitoring. To this end, we have first recalled the existent literature over fall detection systems and time series classification, thus reflecting the great variety of sensors and methods, and the difficulties of choosing one from another. We proposed a set of criteria that allows to have a grasp at the implementability of fall detection systems, hence showing some limitations among this abundant offer, and the benefits of floor systems when considering practical use. Moreover, this study highlighted the more frequent use of supervised classifiers for time series classification. Motivated by the interesting properties of Tarkett's floor sensor and the fall detection application, we put up a simple yet practical supervised model that achieves state-of-the-art performances over an experimental data set. This model is based on the well known random forest and we proposed improvements through time-aggregation and data augmentation. A feature reduction procedure was also proposed to answer hardware constraints, thus allowing the integration of the model into an embeddable version. It was finally implemented in the processing unit in nursing homes and allowed to collect a few real data that were used for the study of a new model. These new data being somehow different from the experimental data and the fall being a rare event, we proposed class-imbalance adapted transfer learning procedures over decision trees. These methods showed promising results, hence highlighting the importance of preserving the minority class when transferring decision trees. Along with these findings we made available a Python implementation of the seminal transfer algorithms and their proposed variants. Finally, still motivated by elderly monitoring in nursing homes, we proposed a method that allows the floor-based system to distinguish elderly persons from others. We showed that even with a small data set, using a convolutional neural network with constrained pre-learning steps can achieve state-of-the-art performances.

These contributions all led to interesting perspectives. First, the monitoring system was designed with embedded processing, hence imposing hardware constraints that were addressed with a simple feature selection procedure. Although effective in terms of performance, this procedure only takes as input the variable importance, without any consideration over the variable computational complexity. This kind of method lies in a currently rarely explored field that is referred to as *budget learning*, and might lead to interesting improvements for hardware-limited applications or under power consumption constraints. Moreover, as feature engineering can always enhance learning models, the feature spaces may change from a model's version to another, thus suggesting the exploration of *heterogeneous* transfer learning with decision trees. We also noted that one of the proposed transfer methods could be improved and more importantly, that although these methods are easily applicable to any tree ensemble (by simply applying the procedure to all trees), future work may consider building a selective meta-method that can automatically build the "best" tree ensemble for target data. Finally, our work on deep-learning based model suggested natural continuations such as detecting activity types (from all possible activities of daily living), and even identifying individuals.

A

Features for time series classification

In this appendix, we give the details of statistical features used for the floor signals, and give a short correlation analysis.

1 Signal into feature vector

As described in Section 3.2 on page 68, in order to transform the preprocessed signal into a feature vector, we compute statistical features over a fixed window. Let $\mathbf{s}$ denote a preprocessed signal, and we denote by $\mathbf{s}[t] \in \mathbb{R}$ its value at time t. The signal extracted over a window of size T is then denoted $\{\mathbf{s}[t]\}_{t=1}^{T}$.

We first denote by μ the mean over $\mathbf{s}$, which for a discrete signal is defined as

$$\mu = \frac{1}{T}\sum_{t=1}^{T} \mathbf{s}[t] \ .$$

The *central moment* at order n of a random variable X is defined as $\mathbb{E}[(X-\mu)^n]$. We denote by μ_n its discrete version, expressed as

$$\mu_n = \frac{1}{T}\sum_{t=1}^{T} (\mathbf{s}[t]-\mu)^n \ .$$

Numerically, the derivative of $\mathbf{s}$ is computed with the discrete difference

$$\mathbf{s}'[t] = \mathbf{s}[t+1] - \mathbf{s}[t] \quad \forall t \in [\![1, T-1]\!] \ ,$$

and the derivative at order n is obtained in the same manner:

$$\mathbf{s}^{(n)}[t] = \mathbf{s}^{(n-1)}[t+1] - \mathbf{s}^{(n-1)}[t] \ ,$$

with $\mathbf{s}^{(1)} = \mathbf{s}'$.

We also use the time stamps of minimum and maximum values of $\mathbf{s}$:

$$t_{\min} = \arg\min_t \mathbf{s} \quad \text{and} \quad t_{\max} = \arg\max_t \mathbf{s} \ .$$

We denote by $\mathcal{C}$ the set of segments (or sub-signals) $\mathbf{c}_k$ of $\mathbf{s}$ that are above a threshold over the absolute value of $\mathbf{s}$:

$$\mathcal{C} = \{\mathbf{c}_k\}_{k=1}^{K} = \left\{ \{\mathbf{s}[t]\}_{t=a_k}^{b_k} \mid |\mathbf{s}[t]| > \alpha, 1 \leq a_k < b_k \leq T \right\} \ , \tag{A.1}$$

Name	Formula
Maximum	$\max_t \mathbf{s}$
Minimum	$\min_t \mathbf{s}$
Delta-min-max	$\max_t(\mathbf{s}) - \min_t(\mathbf{s})$
Median	$\text{median}(\mathbf{s})$
Mean	$\mu = \frac{1}{T}\sum_{t=1}^{T} \mathbf{s}[t]$
Variance	$\sigma^2 = \mu_2 = \frac{1}{T}\sum_{t=1}^{T} (\mathbf{s}[t] - \mu)^2$
Standard deviation	σ
Normalised momentum	$\frac{\mu_n}{\sigma^n}$
Energy	$\frac{1}{T}\sum_{t=1}^{T} \mathbf{s}^2[t]$
Log-energy	$\frac{1}{T}\sum_{t=1}^{T} \log\left(1 + \mathbf{s}^2[t]\right)$
Shannon-energy	$\frac{1}{T}\sum_{t=1}^{T} \mathbf{s}^2[t] \log\left(1 + \mathbf{s}^2[t]\right)$
Max-n-derivative	$\max_t \left(\mathbf{s}^{(n)}\right)$
Energy-derivative	$\text{Energy}(\mathbf{s}')$
N-greater-threshold	$\sum_{t=1}^{T} \mathbb{1}\{t \mid \mathbf{s}[t] > \alpha\}$
Peak-count	$\sum_{t=1}^{T} \mathbb{1}\{t \mid s[t] > \alpha\ ,\ \mathbf{s}'[t] > \beta\ ,\ -\mathbf{s}'[t-1] < -\beta\}$
Derivative-before-max	$\mathbf{s}'[t_{\text{max}} - 1]$
Derivative-after-max	$\mathbf{s}'[t_{\text{max}}]$
Derivative-before-min	$\mathbf{s}'[t_{\text{min}} - 1]$
Derivative-after-min	$\mathbf{s}'[t_{\text{min}}]$
Proportion-abs-lower	$\frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\left\{t \mid \lvert\mathbf{s}[t]\rvert < \alpha\right\}$
Mean-segment-lower	$\frac{1}{K}\sum_{k=1}^{K}(b_k - a_k)$ cf. (A.1)
Percentile	$\text{percentile}(\mathbf{s}, \alpha)$ (with α the percentage)
Interpercentile	$\text{percentile}(\mathbf{s}, \alpha) - \text{percentile}(\mathbf{s}, 1-\alpha)$
Log-mean-Peak	$\log\left(1 + \frac{1}{N}\sum_{k=1}^{N} \{\lvert\mathbf{s}[t]\rvert \mid \lvert\mathbf{s}[t]\rvert > \text{percentile}(\lvert\mathbf{s}\rvert, \alpha)\}\right)$
Log-mean-Valley	$\log\left(1 + \frac{1}{N}\sum_{k=1}^{N} \{\lvert\mathbf{s}[t]\rvert \mid \lvert\mathbf{s}[t]\rvert < \text{percentile}(\lvert\mathbf{s}\rvert, \alpha)\}\right)$
Log-mean-Diff	$\text{Log-meanPeak}(\mathbf{s}) - \text{Log-meanValley}(\mathbf{s})$

Table A.1: Features

and the length of a segment $\mathbf{c}_k$ is then $b_k - a_k$.

The normalised momentum is computed with $n \in \{3, 4, 5, 10\}$ which results in total in 29 features. These features are computed over the preprocessed signal $\mathbf{s}$, its first derivative $\mathbf{s}$', and the absolute value of its Fourier transform (computed with the FFT algorithm). In the following figures, the features are denoted *X*-Name with *X* denoting the signal, derivative or Fourier transform (resp. *S*, *D*, *F*), and Name denoting one of the 29 previously described functions. For example, the feature "Maximum function computed over the absolute value of Fourier transform of the signal" is denoted *F*-Maximum.

2 Correlation

We compute the correlation over our set variables using the standard Pearson correlation, which, between two feature vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^N$, is expressed as

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)}{\sigma_x \sigma_y} ,$$

with $\mu_x, \mu_y, \sigma_x, \sigma_y$ the empirical means and standard deviations of $\mathbf{x}$ and $\mathbf{y}$. Values of ρ range from -1 to 1. As we do not care about either positive or negative correlation but rather its absolute value, we compute $|\rho(\mathbf{x}, \mathbf{y})|$, hence ranging from 0 to 1, 1 meaning total correlation (positive or negative), 0 meaning no linear correlation. Figure A.1 displays the correlation matrix of all variables, showing that generally speaking the variables are rather decorrelated. In this case, the variable importance is still usable (the more we have correlated variables, the less variable importance is meaningful) for interpretation. In general, even with numerous correlated features the recursive feature elimination is still pertinent [72].

A remark to be made is that two features – Delta-min-max and Log-mean-Diff, are built as the difference of other features. This does not imply multicollinearity nor correlation between variables, but rather adds potentially usable variables to the pool of features, that can have a physical sense in terms of signal interpretation. Correlation matrices of these features (and the ones they rely on) are displayed in Figure A.2, showing that despite being built from difference of other variables, a variable is not necessarily correlated to them.

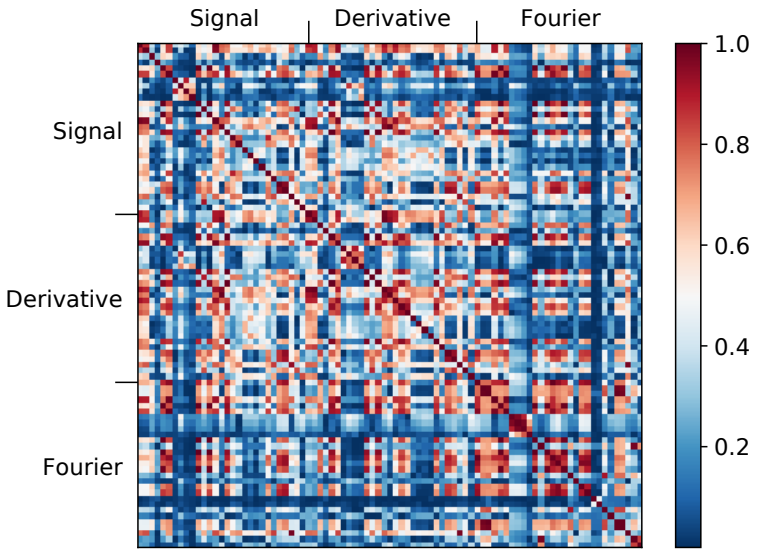
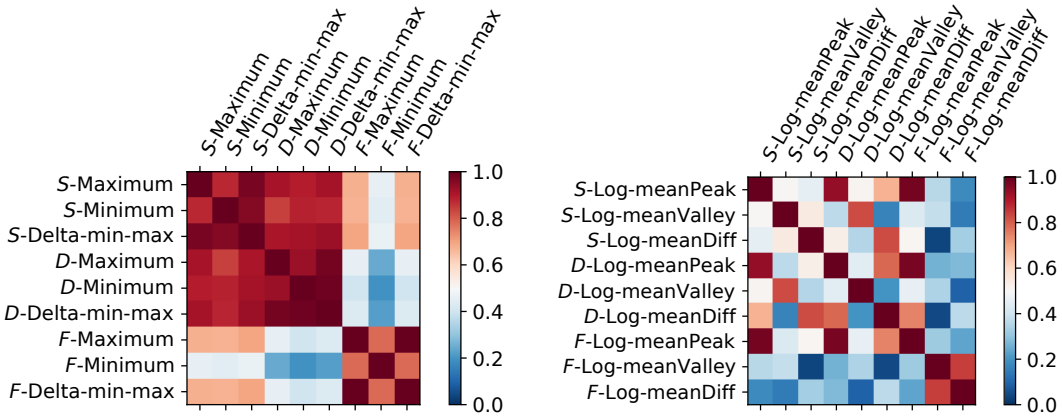


Figure A.1: Correlation matrix with the whole set of 87 variables, ordered along the three groups: signal, derivative and Fourier transform (each group contains the 29 variables previously described)



(a) Delta-min-max and its relative variables

(b) Log-meanDiff and its related variables

Figure A.2: Correlation matrices for the variables built with existing ones, along the three representation (signal, derivative, Fourier transform).

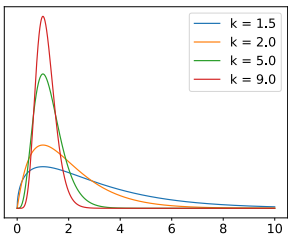
B

Transfer learning: results over synthetic data

This appendix gives additional results to the transfer methods described in Section 6 on page 105. These results are taken from the experiments over synthetic data with varied "levels" of transformation between source and target domain. We first give some more details about how we simulate these transformation levels. We recall that data are generated with points drawn from multivariate Gaussian distributions. We generated $N_{clust} = 10$ clusters in three dimensions, and sampled $N_{source} = N_{target} = 200$ samples to generate the training source and target sets. In source, all 10 clusters have random mean and variances drawn from Uniform distributions with values respectively in $[-70, 70]$ and $[5, 15]$.

The transformation towards target is done as follow: the clusters parameters are changed and data points are sampled from these new clusters. The most straightforward modifications that can be done to a multivariate Gaussian distribution consist in slightly changing its inner parameters (mean and variance), hence leading to "drifts", or "squeezes" and "stretches" of the clusters in the feature space. Secondly, another method to change data is to remove or add clusters to the initial pool (and redrawing means and variances from the Uniform distributions for the new ones).

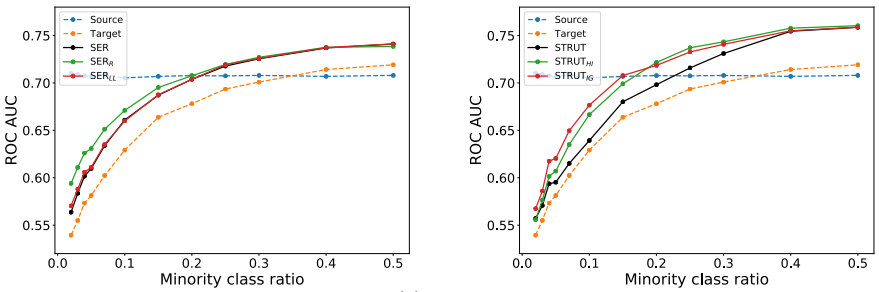
1. **Drifts**: Given a cluster, a drift is applied as an additive value to *each* of the cluster's means $\{\mu_i\}_{i=1,2,3}$, meaning that each μ_i is modified. The additive value (i.e. the drift) is randomly sampled from a Gaussian distribution with mean μ_D and a fixed variance $\sigma_D = 2$. The parameter μ_D hence guides the "degree" of transformation applied from source to target. When performing a drift, all clusters means are modified.
2. **Squeeze / Stretch**: The concept for Squeeze / Streches is very similar as Drifts since we apply a multiplicative value to each of the cluster's variances $\{\sigma\}_{i=1,2,3}$. The multiplicative value is drawn from a Gamma distribution (parameters k and θ). Parameter θ is fixed to $\frac{1}{1-k}$ so that the mode (the most represented value) is always 1. The lower k, the higher the variance, hence guiding the "degree" of transformation.



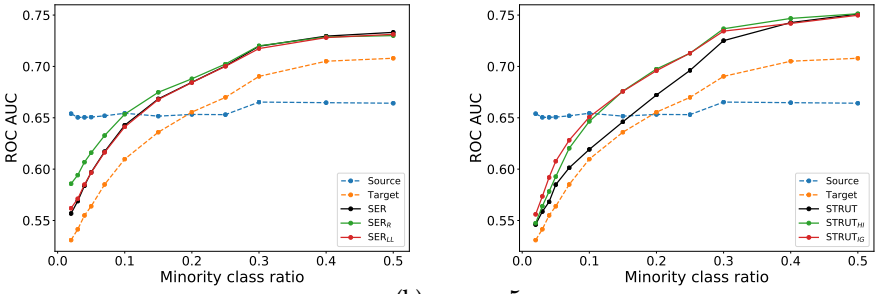
3. **Add / Remove**: The last transformation consists of choosing numbers N_R and N_A of clusters to remove and add. These numbers are drawn from a Gaussian distribution centred on $\mu_{AR} \times N_{clust}$ with fixed variance $\sigma_{AR} = 0.5$. The parameter μ_{AR} is then the mean proportion of clusters that will be removed and added, hence guiding the "dregree" of transformation.

As previously described, these transformations are all combined with imbalance in the target set. These transformations are performed independently, i.e. there is no combination between them. Figure B.1, B.2 and B.3 display the results over various degree for each transformation. The general conclusion is the same as in the chapter, that is, the proposed methods outperform the original ones under class imbalance.

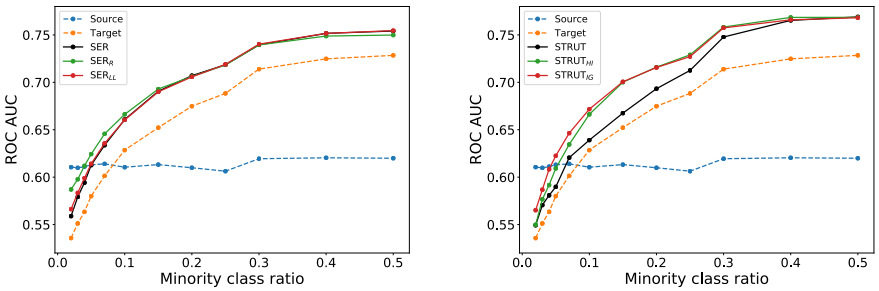
However, some additional remarks can be made. First, we note that as the transformation degree increases, *Source* performance drastically drops, hence illustrating the dissimilarity between domains. It shows that the source domain is no longer sufficient to explain the target domain to the learning model. With sufficient balance (and data) in the target training set, it becomes then more interesting to learn a new model to classify data. Secondly, it seems that the performance gap between the original algorithms and our proposed variants does not depend on the transformation degree but only on the class imbalance ratio. Indeed, these methods were not thought as general improvements but as adaptations to the imbalance case.



(a) $\mu_D = 2$

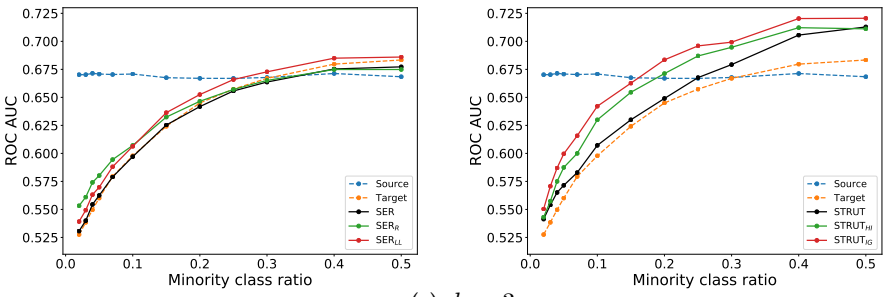


(b) $\mu_D = 5$

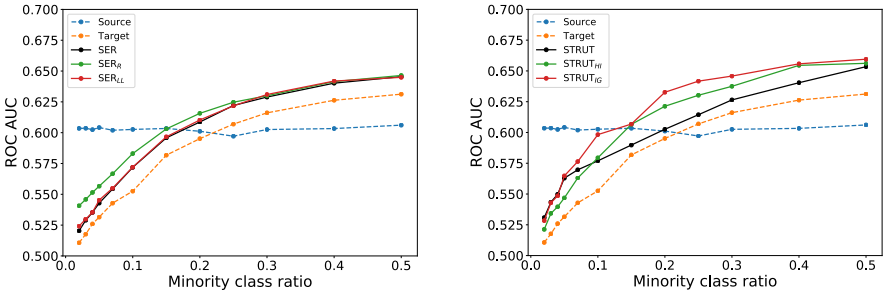


(c) $\mu_D = 10$

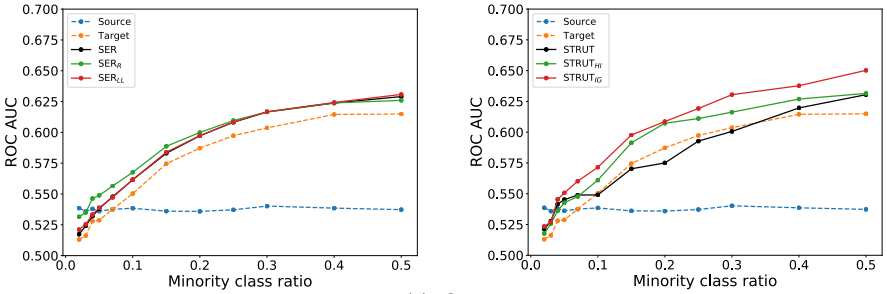
Figure B.1: Drift. ROC AUC over synthetic data depending on the minority class ratio (ratio given on the scale 0-1).



(a) $k = 3$

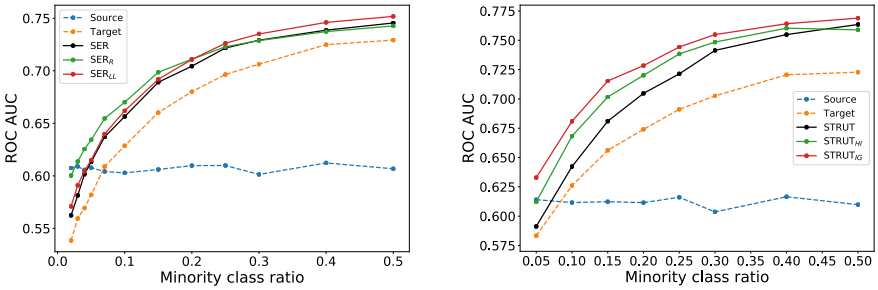


(b) $k = 2$

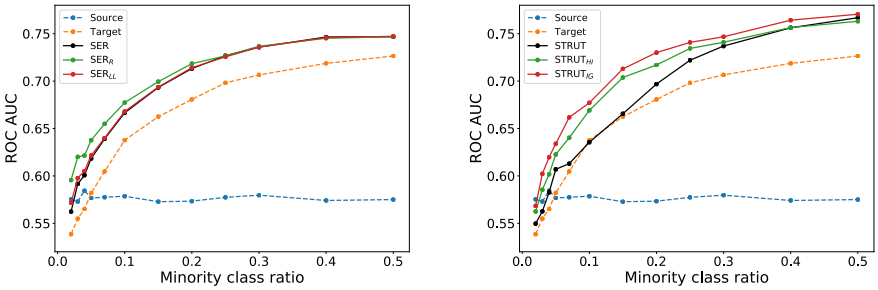


(c) $k = 1.1$

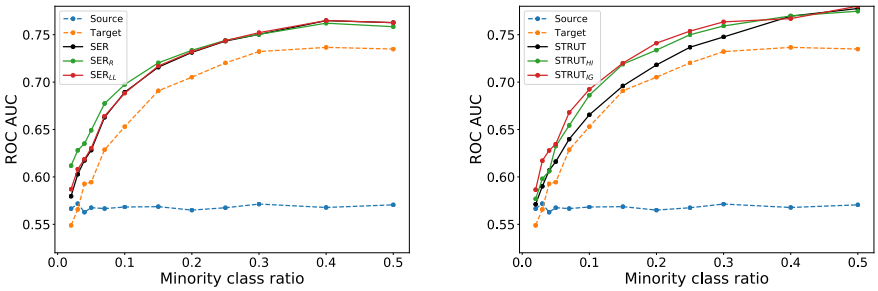
Figure B.2: Squeeze / Stretch. ROC AUC over synthetic data depending on the minority class ratio (ratio given on the scale 0-1).



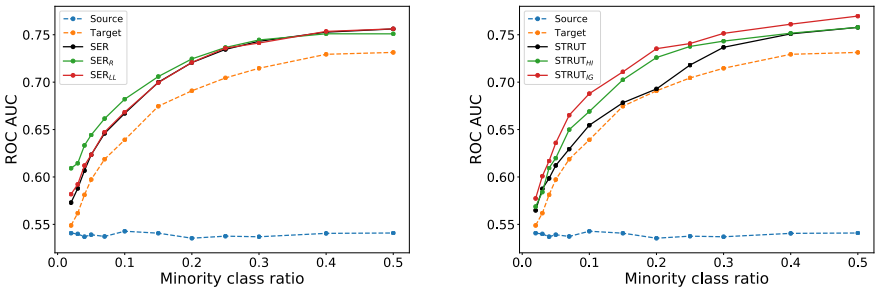
(a) $\mu_{AR} = 1/4$



(b) $\mu_{AR} = 1/3$



(c) $\mu_{AR} = 1/2$



(d) $\mu_{AR} = 3/2$

Figure B.3: Add / Remove. ROC AUC over synthetic data depending on the minority class ratio (the ratio is given on the scale 0-1).

C

Deep learning models

This appendix gives more details on the neural network described in Section 3 on page 117, in particular through a short description of convolutional neural networks and the tools used for their training.

1 Feedforward neural networks

Framework. The core model of deep learning models is the feedforward neural network. The principle of this model is to learn a predictive function Φ that will apply successive transformations to the input vector $\mathbf{x}$ to finally output $\mathbf{y}$ as the estimate of the class of $\mathbf{x}$. In multiclass classification with K classes, $\mathbf{y}$ is generally a vector in $\mathbb{R}^K$ in which each element $y_k \in \mathbb{R}$ is the probability estimate of belonging to class k). In its basic form, the model is organised in *layers* of *neurons* (or *units*), and each neuron applies to an input vector $\mathbf{z} \in \mathbb{R}^N$ the following transformation:

$$f(\mathbf{z}) = a(\mathbf{w}^\intercal \mathbf{z} + b) \ ,$$

with a denoting the *activation* function, $\mathbf{w} \in \mathbb{R}^N$ being a *weight* vector, and $b \in \mathbb{R}$ denoting the *bias*. Each layer l has an activation function a_l that is the same for all neurons, and the output of a whole layer can then be written

$$\mathbf{z}_l = \phi_l(\mathbf{z}_{l-1}) = a_l(\mathbf{W}_l \mathbf{z}_{l-1} + \mathbf{b}_l) \ ,$$

with $\mathbf{W}_l \in \mathbb{R}^{n_{l-1} \times n_l}$ denoting the weight matrix of the layer l, and n_l the number of neurons in layer l. This form of neural network is called *fully-connected*, in the sense that each neuron of the layer l is connected to all neurons of its two surrounding layers (see Figure C.1).

In a more general view, for a given input $\mathbf{x}$, the final output $\mathbf{y}$ of a L-layer neural network is the composition of all layers applied to $\mathbf{x}$:

$$\mathbf{y} = \Phi(\mathbf{x}) = \phi_L\Big(\phi_{L-1}\big(\dots \phi_1(\mathbf{x})\big)\Big) \ ,$$

with each layer l having its own parameters: number of neurons, activation function, weights and biases.

Activation function. In this setting, each layer applies a linear transformation on the input vector by multiplying it with the weight vector (and adding the bias), followed by a

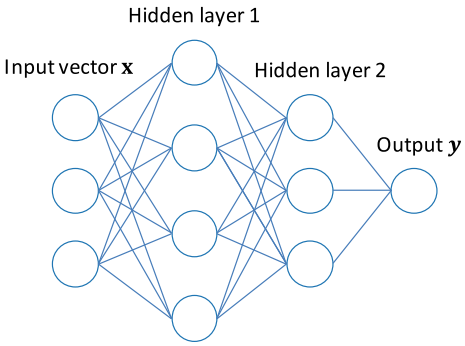


Figure C.1: Schematics of a fully-connected feedforward neural network with two hidden layers and a one-dimensional output vector (e.g. for a binary classification).

non-linear transformation performed by the activation function a. There are several types of activation functions, the most common ones being the sigmoid (denoted σ), the Rectified Linear Unit (denoted ReLU), and the hyperbolic tangent (denoted tanh):

$$\begin{aligned} \sigma(x) &= \frac{1}{1+e^{-x}} \\ \mathrm{ReLU}(x) &= \max(0, x) \\ \tanh(x) &= \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \ . \end{aligned}$$

These functions are displayed on Figure C.2. Originally, the first activations functions were designed to mimic the neuron behaviour, that is, a binary response depending on the level of the input. While this is easily done with a step function (0 if under a threshold, and 1 otherwise), this function does not allow complex representations of data (all neurons output either 0 or 1) and has a null gradient everywhere (and undefined on the threshold), which makes it a bad choice for learning parameters.

In this context, the sigmoid function was proposed as a smoothed version of the step function, which is differentiable and has value between 0 and 1. The hyperbolic tangent is very similar, as it is a rescaled sigmoid. The main differences lie in the values taken by the tanh (-1 to 1) and the derivative at 0, which is greater for the tanh, thus making weights move away from potentially small initial values. The ReLU was also proposed since it allows a larger scale of values to be taken ($\mathrm{ReLU}(x) \in \mathbb{R}^+ \quad \forall x \in \mathbb{R}$) and has very easily computed gradient (1 if $x > 0$ and 0 otherwise), thus resulting in faster learning. Besides, ReLU has the nice property of not having a "vanishing gradient" (i.e. very low gradient values) contrary to the sigmoid, which may "slow" the learning for high input values.

Although the choice of activation functions can be decisive, aside common sense (e.g. for classification, since sigmoid outputs values between 0 and 1, it is usually used in the last layer), there is no general rule for selecting them, and according to literature, this relies more on experimental skills [71].

Convolutional neural networks. Convolutional neural networks (CNN) are feedforward models in which the layers do not apply the same inner transformation to the input. Instead, the matrix product is replaced by a convolution between the input and a set of

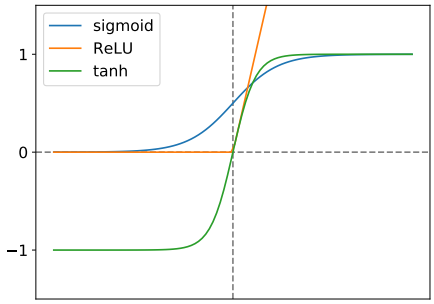


Figure C.2: Common activation functions for feedforward neural networks.

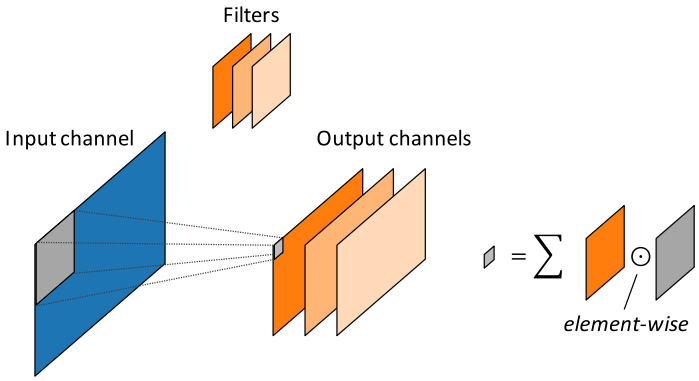


Figure C.3: Convolutional 2D layer.

filters. Like fully-connected networks, an activation function is then applied before passing the output to the next layer. Although applicable to one-dimensional signals, CNN are usually used for image recognition, and therefore applied with 2D signals. Each filter produces an output channel whose size depends (but not exclusively, see later) on the filter size. Figure C.3 displays a convolutional layer.

Convolutional layers are usually followed by *pooling* layers, the most used one being the MaxPooling layer. These layers are responsible for a size reduction of the inputs. Similarly to a convolutional layer, a MaxPooling layer consists of passing a small filter that, instead of being multiplied with the input, performs the *maximum* operation on it (see Figure C.4).

These operations, the convolutional and the pooling layers, can be performed with various *strides*. Indeed, in discrete convolution, a new parameter arises which is the step size when the filter is passed along the input channel. In classical convolution, this step is of size 1, however one can choose to bring it to larger values so that the output is even more downsized. Figure C.5 displays the effect of different strides.

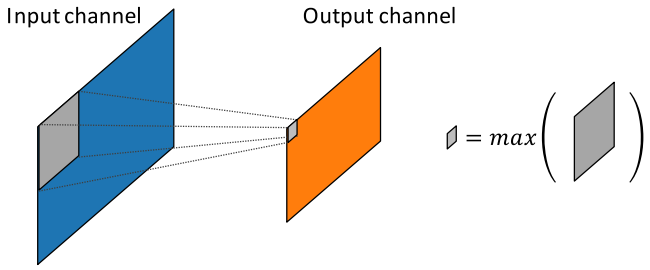
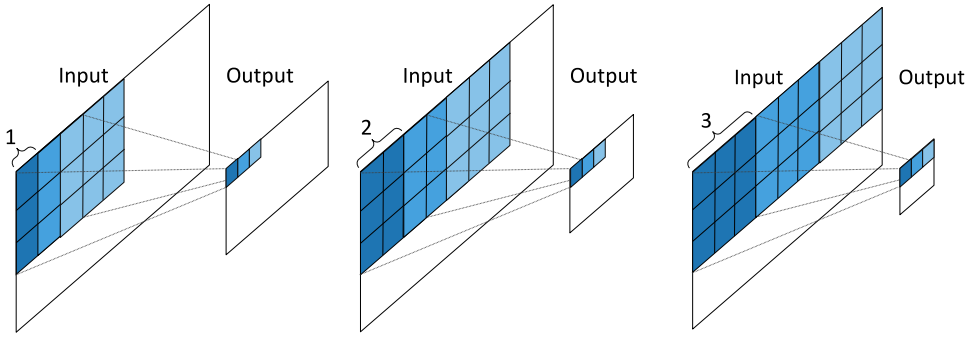


Figure C.4: MaxPooling layer.



(a) Stride 1 (b) Stride 2 (c) Stride 3

Figure C.5: Schematics of the stride in convolutional layers.

2 Training the neural network

Gradient descent. Most neural networks are trained with stochastic gradient descent (SGD) [71]. It is a slightly different version of the classical gradient descent in which at each step, rather than using the whole training set to estimate the gradient of the objective function, only a subset of examples is used. The SGD usually refers to a subset of size one, while using a larger subset is referred to as *minibatch* gradient descent. This variant revealed to be useful for very large training sets since computing the gradient becomes faster. However in our framework, since we deal with rather small data sets, the training of both parts of our model (the step detection network and the final classification network) can be done while computing the gradient with the whole training set. We then use a classical gradient descent scheme.

Let us denote by Θ the set of all parameters of the neural network. In a fully-connected network, Θ would encompass all the weights and biases of all layers. Then, given a loss function $\mathcal{L}$ associated with our task (e.g. the binary cross entropy for classification), the objective function to minimize is written

$$J(\Theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\mathbf{x}_i, \mathbf{y}_i, \Theta) \ .$$

The gradient over the whole training set can be written

$$\nabla_\Theta J(\Theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\Theta \mathcal{L}(\mathbf{x}_i, \mathbf{y}_i, \Theta) \ .$$

The gradient descent (GD) algorithm consists in successively computing the gradient and updating Θ as follows:

$$\Theta^{(q+1)} = \Theta^{(q)} - \alpha \nabla_\Theta J(\Theta^{(q)}) \, ,$$

with q denoting the iteration number and α the learning rate.

Adapting the learning rate. Since the objective function for neural network is not convex, it contains local minima or saddle points in which a gradient descent algorithm may be stuck. However, reaching a local minimum that is not the global minimum does not prevent the model from yielding good results on the testing set [71]. To ensure that we end up in a "good" minimum of J in a reasonable time, the learning rate α can be tuned. In fact, the learning rate can be decisive when using gradient descent methods. A low value of α can lead to very slow convergence, or make the algorithm stay in the first encountered local minimum, while a too large value may make the algorithm "bounce" in the parameter space without settling in any minimum.

As the gradient descent approaches the minimum of J, we wish to "slow" the progression of the descent so that the algorithm may converge. It is then common to use a *decay* for the learning rate [71]. There are various ways of choosing α, and several works proposed automatic computation, such as Adam [93]. These methods were proposed for SGD were the gradient is expected to be very noisy. In our case, we used a simple geometric decay, meaning that every M epochs of the learning procedure, α is multiplied by a constant $\gamma < 1$. We combined this decay with a momentum strategy, and this was used for both NurseNet and SPN optimization procedures.

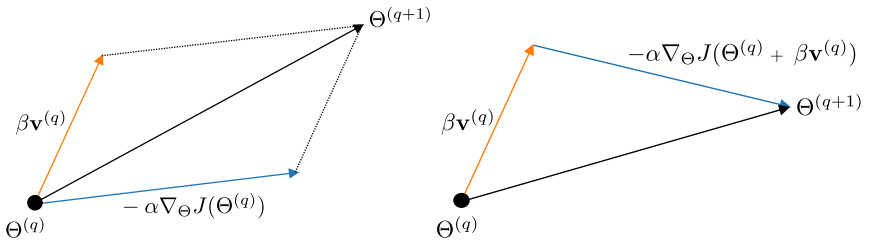
Momentum. Among strategies that allow to improve the gradient descent the method of *momentum* has been fairly used [71]. It is called that way due to a physical interpretation that when going downhill, a mass retains its velocity. The method introduces a vector $\mathbf{v}$ that aims to retain the negative gradient with a decaying factor β. In this case, the gradient update is replaced by

$$\begin{aligned} \mathbf{v}^{(q+1)} &= \beta \mathbf{v}^{(q)} - \alpha \nabla_\Theta J(\Theta^{(q)}) \\ \Theta^{(q+1)} &= \Theta^{(q)} + \mathbf{v}^{(q+1)} \, , \end{aligned}$$

where $0 < \beta < 1$. Momentum speeds up the movement along strong gradients hence accelerating the descent and helping the algorithm to avoid local minima. A variant was proposed by Sutskever et al. [169] inspired by Nesterov's accelerated gradient [128]. The Nesterov momentum is sightly different in the way that the gradient is evaluated ahead of the current position in the parameter space. The update becomes

$$\begin{aligned} \mathbf{v}^{(q+1)} &= \beta \mathbf{v}^{(q)} - \alpha \nabla_\Theta J(\Theta^{(q)} + \beta \mathbf{v}^{(q)}) \\ \Theta^{(q+1)} &= \Theta^{(q)} + \mathbf{v}^{(q+1)} \, . \end{aligned}$$

This variant is thought as a correction of the classical momentum method in the way that if the momentum makes the descent algorithm point in the wrong direction or overshoot, looking ahead allows to correct the trajectory. Figure C.6 illustrates the momentum strategies.



(a) Classical (b) Nesterov

Figure C.6: Schematics of momentum strategies.

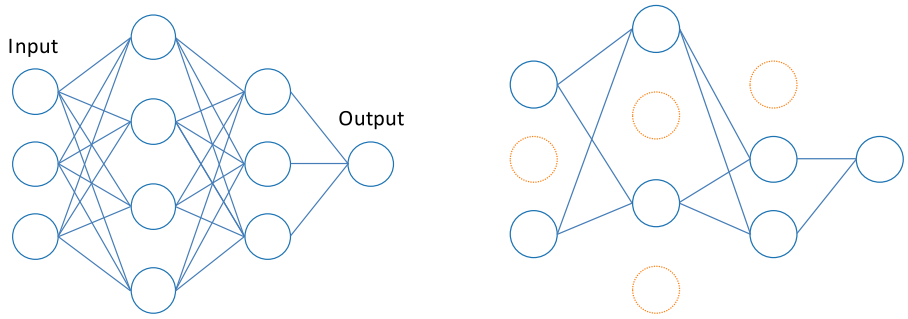
Early stopping. Most learning techniques use optimization as their core training procedure. In our method for example, we used gradient descent to train SPN and NurseNet. As the optimization algorithm is progressing through iterations (or *epochs*), if parameters are well set, the training error is expected to continuously decrease. However, the validation error may reach a minimum and increase as iterations continue, thus leading to overfitting. Since neural networks are complex models, they are all the more prone to overfitting. Therefore, it is crucial to track the validation error along training and stop the latter at the right moment. Techniques that follow this principle are referred to as *early stopping*. They are usually quite simple heuristics whose main goal is to stop training as soon as validation error rises. In practice, rather than using rules that rely on the variation of the validation error, one can simply train a first time the model while monitoring training and validation errors and retrain the model with the right amount of iterations [71].

During our optimization procedures, we used this straightforward method and recorded the "necessary" number of iterations while monitoring errors.

Batch normalization. Batch normalization [87] is a recent innovation that aims at improving the stability and performance of deep neural networks. The principle is to normalize the output of a layer by the mean and the variance of the activations, that is, given a vector of activations $\mathbf{z}$, replacing it with $\mathbf{z}' = \frac{\mathbf{z}-\mu}{\sigma+\epsilon}$, with μ and γ being the mean and variance vectors over the units in the layer. The main motivation is to prevent activations to "shift around", that is, no matter how the activation changes, we maintain a controlled mean and variance. However, in order not to force the layer to have zero-mean and variance one, the procedure adds two new trainable parameters γ and β to the layer, so that after normalization, the activations are replaced with $\mathbf{z}'' = \mathbf{z}'\gamma + \beta$, thus allowing the network to learn by itself the "useful" mean and variance for the given layer. Although the contributions of batch normalization are still a subject of research in the deep learning community [106], it has been largely used in many state-of-the-art deep learning models.

In our model, batch normalization is applied after the first three convolution layers.

Dropout. Batch normalization if often combined with dropout [71]. Dropout is a regularization method that consists of zeroing a random subset of units during training, thus limiting overfitting [163]. The random selection is usually done according to a Bernouilli distribution of parameter $p = 0.5$, and it is applied to all layers except the output one. With $p = 0.5$, it means that during training, half of the neurons (in average) are inactive, and the selection of neurons that are zeroed is changing at each pass (see Figure C.7). This



(a) Original network (b) with dropout

Figure C.7: Schematics of dropout regularization.

can be seen as training a very large ensemble of subnetworks that can be formed with less neurons at each layer, each one of them being trained with a single step of gradient [71].

It should be noted that even if dropout may be applied with convolutional layers, in our model it is used on the fully-connected layers only.

3 Convolutional neural network for object detection

The CNN architecture is effective for two main reasons. First, in signals such as time series or images, we find many locally correlated groups forming patterns that may be of interest. Due to its inner filters, a CNN can grasp such patterns. Besides, these patterns are likely to be repeated, which makes the convolution operation ideal to localize them in every part of the input signals. When applied, CNNs are responsible for localizing patterns with several level of features and are finally combined with fully-connected layers that perform the final classification.

Region proposal for object detection. Unlike image classification where the goal is to guess the class of the whole input image, the main goal of object detection is to *localize and classify* objects in the image. To that end, many attempts were based on the exhaustive sliding window strategy, which yields performing detection results, although at a great computational cost [82]. To overcome this issue, recent advances were driven by *proposal* models [67, 145, 146]. The main idea is that, instead of exploring and running a classification procedure over a very large set of window shapes and positions, a first model is passed over the image and "proposes" to the classifier positions (and shapes) with high probability of object presence. This procedure, called region proposal, is an ongoing research topic and has received various solutions[1], one of them being the well-known selective search [180]. In fact, it has been noted that when exploring test time of object detection methods, the region proposal might represent the largest part, hence becoming the bottleneck in many algorithms. One of the latest advances in this topic consists of using a CNN as the proposal model. It was proposed by Ren et al. [146] in a general object detection model called Faster R-CNN, in which the subpart responsible for the region proposal is simply called region proposal network (RPN).

[1] A comprehensive review of region proposal models has been proposed by Hosang et al. [82].

Region proposal network. One major improvement Faster R-CNN brings to object detection is that RPN and the classifier *share the same first convolutional layers*, thus drastically reducing the marginal cost of region proposal during test time. As explained in Chapter 5, RPN evaluates K bounding boxes (called *anchors*) of various sizes and shapes using the main convolutional layers and a mini-network made of two sibling layers, one layer giving the probability estimate of the box to be an object, the other layer yielding 4 coordinates of the box (center coordinates, width and height). This is done with a sliding window on which anchors are centred. Authors use $K = 9$, resulting from 3 scales and 3 aspect ratios. For the training procedure, they assign positive value to anchors that have $\text{IoU} > 0.7$ and negative if $\text{IoU} < 0.3$. The loss function they minimize is the *multi-task* loss proposed by Fast R-CNN [67], which is the weighted sum over two losses:

$$\mathcal{L}(p_i, t_i) = \mathcal{L}_{\text{cls}}(p_i, p_i^*) + \lambda \mathcal{L}_{\text{reg}}(t_i, t_i^*) \ ,$$

with i denoting the box index, p_i the output prediction of box i being an object, p_i^* the ground-truth label, t_i the vector of 4 coordinates of the predicted box and t_i^* the ground-truth box coordinates. $\mathcal{L}_{\text{cls}}$ is the log-loss over the class *object*, and $\mathcal{L}_{\text{reg}}$ a custom regression loss for the box coordinates [67].

Authors propose a training method called *alternating training* that entails four steps designed to train RPN and the classifier while enforcing their convolutional layers to be shared. Step 1 consists of training end-to-end RPN, and step 2 uses RPN proposals to train the classification network. At this stage, RPN and the classifier have the same convolutional architecture but do not share the same filters since weights have been trained separately. The procedure adds then two more steps. Step 3 consists of initializing the convolutional layers of RPN with the classifier's filters and fixing them so that just the layers unique to RPN are refined. Finally, step 4 fine-tunes the layers that are unique to the classifier while fixing the convolutional layers. At this point, the convolutional layers are shared across both networks, hence forming a unified network [146].

Inspired by this work, our classification network NurseNet is based on a simpler version of RPN which is trained to "propose" boxes with high probability of containing a *step* signal. This part of NurseNet is called step proposal network (SPN). Like RPN, we chose several box sizes according to a typical step size. The main differences with RPN is that we do not use a regression layer over the boxes since our purpose is not to perform a precise step detection.

Then, the output layer of SPN is removed, leaving only the convolutional layers, and we add fully-connected layers that are finally trained over the classification task while keeping the SPN part fixed. Like RPN, SPN tells the fully-connected part of NurseNet "where to look".

Average precision. The average precision (AP) is a common metric used in object detection literature. It is the average of the precision along all recall values. Precision and

recall are defined as:

$$\text{Precision} = \frac{\text{True positives}}{\text{Positive outputs}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{Recall} = \frac{\text{True positives}}{\text{Total positives}} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (= \text{TPR})$$

AP is the area under curve (AUC) of the precision-recall curve. To obtain this curve, the methodology is the following:

1. The object detector outputs boxes over the testing data. Each box has an associated score (i.e. a confidence value).
2. All output boxes with a high score (e.g. above 0.5) are considered positive outputs according to the detector. These prediction boxes are ranked along their score.
3. The boxes with $\text{IoU} > T_h$ with ground truth boxes are considered true positive and false positive otherwise.
4. The precision and recall values are computed as we take into account output boxes from the highest to lowest score.
5. The AUC over the precision-recall curve is computed.

Bibliography

[1] Domain adaptation project. URL https://people.eecs.berkeley.edu/~jhoffman/domainadapt/#datasets_code. 103

[2] A. Abanda, U. Mori, and J. A. Lozano. A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33(2):378–412, 2019. 42, 43, 46

[3] M. D. Addlesee, A. Jones, F. Livesey, and F. Samaria. The orl active floor [sensor system]. *IEEE Personal Communications*, 4(5):35–41, Oct 1997. 59

[4] M. Aharon, M. Elad, and A. Bruckstein. rmk-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006. 113

[5] S. Al-Stouhi and C. K. Reddy. Transfer learning for class imbalance problems with inadequate data. *Knowledge and Information Systems*, 48(1):201–228, Jul 2016. 104

[6] M. Alwan, P. Rajendran, S. Kell, D. Mack, S. Dalal, M. Wolfe, and R. Felder. A smart and passive floor-vibration based fall detector for elderly. volume 1, pages 1003 – 1007, 01 2006. 40

[7] M. G. Amin, Y. D. Zhang, F. Ahmad, and K. C. D. Ho. Radar signal processing for elderly fall detection: The future for in-home monitoring. *IEEE Signal Processing Magazine*, 33(2):71–80, March 2016. 36, 40, 49, 55

[8] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1), 2015. 45

[9] E. K. Antonsson and R. W. Mann. The frequency content of gait. *Journal of biomechanics*, 18(1):39–47, 1985. 63

[10] B. Ao, Y. Wang, H. Liu, D. Li, L. Song, and J. Li. Context impacts in accelerometer-based walk detection and step counting. *Sensors*, 18(11), 2018. 112

[11] N. Asadi, A. Mirzaei, and E. Haghshenas. Creating discriminative models for time series classification and clustering by hmm ensembles. *IEEE Transactions on Cybernetics*, 46(12):2899–2910, Dec 2016. 43

[12] M. Atiq. Transfer algorithms on decision trees with class imbalance. https://github.com/atiqm/Transfer_DT. 87, 101

[13] E. Auvinet, F. Multon, A. Saint-Arnaud, J. Rousseau, and J. Meunier. Fall detection with multiple cameras: An occlusion-resistant method based on 3-d silhouette vertical distribution. *IEEE Transactions on Information Technology in Biomedicine*, 15(2): 290–300, March 2011. 38, 47, 48, 53

[14] J. Bae and M. Tomizuka. Gait phase analysis based on a hidden markov model. *Mechatronics*, 21(6):961–970, 2011. 113

[15] P. Barralon, N. Vuillerme, and N. Noury. Walk detection with a kinematic sensor: Frequency and wavelet comparison. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1711–1714, 2006. 113

[16] R. K. Begg, M. Palaniswami, and B. Owen. Support vector machines for automated gait classification. *IEEE Transactions on Biomedical Engineering*, 52(5):828–838, May 2005. 46, 68

[17] M. Bekkar, H. Djema, and T. Alitouche. Evaluation measures for models assessment over imbalanced data sets. *Journal of Information Engineering and Applications*, 3: 27–38, 01 2013. 104

[18] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1):151–175, 2010. 89

[19] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994. 43

[20] F. Bianchi, S. J. Redmond, M. R. Narayanan, S. Cerutti, and N. H. Lovell. Barometric pressure and triaxial accelerometry-based falls event detection. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(6):619–627, 2010. 39, 49, 54

[21] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank. Fast perceptron decision tree learning from evolving data streams. In *Advances in Knowledge Discovery and Data Mining*, pages 299–310, Berlin, Heidelberg, 2010. Springer. 91

[22] R. J. Bolton and D. J. Hand. Statistical fraud detection: A review. *Statistical Science*, 17(3):235–249. 46

[23] A. K. Bourke and G. M. Lyons. A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. *Medical engineering & physics*, 30(1):84–90, 2008. 39, 50, 54

[24] A. K. Bourke, J. V. O'Brien, and G. M. Lyons. Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm. *Gait & Posture*, 26(2):194–199, 2007. 39

[25] A. K. Bourke, P. W. Van de Ven, A. E. Chaya, G. M. OLaighin, and J. Nelson. Testing of a long-term fall detection system incorporated into a custom vest for the elderly. In *Conf Proc IEEE Eng Med Biol Soc*, volume 2008, pages 2844–2847, 2008. 39, 48, 50, 54

[26] A. K. Bourke, P. Van de Ven, M. Gamble, R. O'Connor, K. Murphy, E. Bogan, E. McQuade, P. Finucane, G. Olaighin, and J. Nelson. Evaluation of waist-mounted tri-axial accelerometer based fall-detection algorithms during scripted and continuous unscripted activities. *Journal of biomechanics*, 43(15):3051–3057, 2010. 39, 48, 49, 50, 54

[27] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011. 120

[28] A. Brajdic and R. Harle. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 225–234, 2013. 113

[29] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 65, 67, 70

[30] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984. 65

[31] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000. 45

[32] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 120, 121, 122

[33] S. Brownsell and M. S. Hawley. Automatic fall detectors and the fear of falling. *Journal of Telemedicine and Telecare*, 10(5):262–266, 2004. 21, 31

[34] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European conference on computer vision*, pages 354–370, 2016. 114

[35] W. M. Campbell, D. E. Sturim, D. A. Reynolds, and A. Solomonoff. Svm based speaker verification using a gmm supervector kernel and nap variability compensation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I, May 2006. 51

[36] Centers for Disease Control and Prevention, Injury Center, Injury Prevention & Control. Fatal Injury Data. URL `https://www.cdc.gov/injury/wisqars/index.html`. 20, 30

[37] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. 69, 90

[38] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy. Wearable sensors for reliable fall detection. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 3551–3554. IEEE, 2005. 39

[39] J.-L. Chua, Y. C. Chang, and W. K. Lim. A simple vision-based fall detection technique for indoor video surveillance. *Signal, Image and Video Processing*, 9(3):623–633, Mar 2015. 38, 47, 53

[40] T. chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164 – 181, 2011. 43

[41] M. Corduas and D. Piccolo. Time series clustering and classification by the autoregressive metric. *Computational Statistics & Data Analysis*, 52(4):1860 – 1872, 2008. 43

[42] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39 (9):1853–1865, Sep. 2017. 89

[43] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan. Perfalld: A pervasive fall detection system using mobile phones. In *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 292–297. IEEE, 2010. 39, 50, 54

[44] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 193–200, New York, NY, USA, 2007. Association for Computing Machinery. 88

[45] H. Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics. 88

[46] N. Dave. Feature extraction methods lpc, plp and mfcc in speech recognition. *International Journal For Advance Research in Engineering And Technology(ISSN 2320-6802)*, Volume 1, 07 2013. 49

[47] O. Day and T. M. Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4(1):29, 2017. 88

[48] T. Degen, H. Jaeckel, M. Rufer, and S. Wyss. Speedy: A fall detector in a wrist watch. In *ISWC*, pages 184–189, 2003. 39, 48, 50, 54

[49] Y. S. Delahoz and M. A. Labrador. Survey on fall detection and fall prevention using wearable and external sensors. *Sensors*, 14(10):19806–19842, 2014. 36

[50] M. O. Derawi. Accelerometer-based gait analysis, a survey. *Nor Informasjonssikkerhetskonferanse NISK*, 2010. 112

[51] S. L. Dockstader, M. J. Berg, and A. M. Tekalp. Stochastic kinematic modeling and feature extraction for gait analysis. *IEEE Transactions on Image Processing*, 12(8): 962–976, 2003. 113

[52] C. Doukas and I. Maglogiannis. Advanced patient or elder fall detection based on movement and sound data. In *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*, pages 103–107. IEEE, 2008. 39, 48, 49, 51, 54

[53] D. Dua and E. Karra Taniskidou. UCI machine learning repository, 2017. 103

[54] A. Dubois and F. Charpillet. A gait analysis method based on a depth camera for fall prevention. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4515–4518, 2014. 112

[55] A. Dubois and F. Charpillet. Measuring frailty and detecting falls for elderly home care using depth camera. *Journal of ambient intelligence and smart environments*, 9 (4):469–481, 2017. 39

[56] N. El-Bendary, Q. Tan, F. C. Pivot, and A. Lam. Fall detection and prevention for the elderly: A review of trends and challenges. *International Journal on Smart Sensing & Intelligent Systems*, 6(3), 2013. 36

[57] J. Fagert, M. Mirshekari, S. Pan, P. Zhang, and H. Y. Noh. Characterizing left-right gait balance using footstep-induced structural vibrations. In *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2017*, volume 10168, page 1016819. International Society for Optics and Photonics, 2017. 112

[58] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *2013 IEEE International Conference on Computer Vision*, pages 2960–2967, Dec 2013. 89

[59] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997. 88

[60] L. P. Fried, C. M. Tangen, J. Walston, A. B. Newman, C. Hirsch, J. Gottdiener, T. Seeman, R. Tracy, W. J. Kop, G. Burke, et al. Frailty in older adults: evidence for a phenotype. *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences*, 56(3):M146–M157, 2001. 20, 30

[61] B. D. Fulcher and N. S. Jones. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, 2014. 44

[62] B. D. Fulcher, M. A. Little, and N. S. Jones. Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of The Royal Society Interface*, 10(83). 44

[63] A. Gabell and U. Nayak. The effect of age on variability in gait. *Journal of gerontology*, 39(6):662–666, 1984. 122

[64] C. Garcia-Cardona and B. Wohlberg. Convolutional dictionary learning: A comparative review and new algorithms. *IEEE Transactions on Computational Imaging*, 4(3): 366–381, 2018. 113, 120

[65] S. Gasparrini, E. Cippitelli, S. Spinsante, and E. Gambi. A depth-based fall detection system using a kinect® sensor. *Sensors*, 14(2):2756–2775, 2014. 39, 53

[66] P. Germain, A. Habrard, F. Laviolette, and E. Morvant. Pac-bayes and domain adaptation. *Neurocomputing*, 379:379 – 397, 2020. 89

[67] R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 149, 150

[68] 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. 114

[69] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9):1469–1495, Oct 2017. 91

[70] R. C. González, A. M. López, J. Rodriguez-Uría, D. Álvarez, and J. C. Alvarez. Real-time gait event detection for normal subjects from lower trunk accelerations. *Gait and Posture*, 31(3):322 – 325, 2010. 113

[71] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. 144, 146, 147, 148, 149

[72] B. Gregorutti, B. Michel, and P. Saint-Pierre. Correlation and variable importance in random forests. *Statistics and Computing*, 27(3):659–678, 2017. 137

[73] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422, 2002. 70

[74] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer. 67, 68

[75] S. Hawkins, H. He, G. Williams, and R. Baxter. Outlier detection using replicator neural networks. In *Data Warehousing and Knowledge Discovery*, pages 170–180, Berlin, Heidelberg, 2002. Springer. 45

[76] L. Hazelhoff, J. Han, and P. H. N. de With. Video-based fall detection in the home using principal component analysis. In *Advanced Concepts for Intelligent Vision Systems*, pages 298–309, 2008. 53

[77] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, Sep. 2009. 86, 90, 104

[78] R. Henry, L. Matti, and S. Raimo. Human tracking using near field imaging. In *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*, pages 148–151, 2008. 112

[79] F. Hijaz, N. Afzal, T. Ahmad, and O. Hasan. Survey of fall detection and daily activity monitoring techniques. In *2010 International Conference on Information and Emerging Technologies*, pages 1–6. IEEE, 2010. 36

[80] R. H. Hnery, L. Matti, and S. Raimo. Human tracking using near field imaging. In *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*, pages 148–151, Jan 2008. 41, 58

[81] M. K. Holden, K. M. Gill, M. R. Magliozzi, J. Nathan, and L. Piehl-Baker. Clinical gait assessment in the neurologically impaired: reliability and meaningfulness. *Physical therapy*, 64(1):35–40, 1984. 122

[82] 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. 149

[83] H. Hu, J. Zheng, E. Zhan, and L. Yu. Curve similarity model for real-time gait phase detection based on ground contact forces. *Sensors*, 19(14), 2019. 59

[84] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Scholkopf. Correcting sample selection bias by unlabeled data. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, page 601–608, Cambridge, MA, USA, 2006. 88

[85] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 97–106, New York, NY, USA, 2001. Association for Computing Machinery. 91

[86] R. Igual, C. Medrano, and I. Plaza. Challenges, issues and trends in fall detection systems. *Biomedical engineering online*, 12(1):66, 2013. 36

[87] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, page 448–456, 2015. 148

[88] A. Jeon, J. Kim, I. Kim, J. Jung, S. Ye, J. Ro, S. Yoon, J. Son, B. Kim, B. Shin, et al. Implementation of the personal emergency response system using a 3-axial accelerometer. In *2007 6th International Special Topic Conference on Information Technology Applications in Biomedicine*, pages 223–226. IEEE, 2007. 39, 48, 49, 54

[89] H. Jiang, C. Cai, X. Ma, Y. Yang, and J. Liu. Smart home based on wifi sensing: A survey. *IEEE Access*, 6:13317–13325, 2018. 41, 112

[90] M. Kangas, I. Vikman, J. Wiklander, P. Lindgren, L. Nyberg, and T. Jämsä. Sensitivity and specificity of fall detection in people aged 40 years and over. *Gait & Posture*, 29 (4):571 – 574, 2009. 48, 49, 50, 54

[91] S. S. Khan and J. Hoey. Review of fall detection techniques: A data availability perspective. *Medical engineering & physics*, 39:12–22, 2017. 36

[92] S. Khojasteh, J. Villar, C. Chira, V. González, and E. De La Cal. Improving fall detection using an on-wrist wearable accelerometer. *Sensors*, 18(5):1350, 2018. 39

[93] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014. 147

[94] S. Kiranyaz, T. Ince, and M. Gabbouj. Real-time patient-specific ecg classification by 1-d convolutional neural networks. *IEEE Transactions on Biomedical Engineering*, 63 (3):664–675, 2015. 114

[95] L. Klack, C. Möllering, M. Ziefle, and T. Schmitz-Rode. Future care floor: A sensitive floor for movement monitoring and fall detection in home environments. In *Wireless Mobile Communication and Healthcare*, pages 211–218, 2011. 41

[96] B. Krawczyk and P. Skryjomski. Cost-sensitive perceptron decision trees for imbalanced drifting data streams. In *Machine Learning and Knowledge Discovery in Databases*, pages 512–527, Cham, 2017. Springer International Publishing. 91

[97] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 119

[98] M. Kukar, I. Kononenko, et al. Cost-sensitive learning with neural networks. In *ECAI*, volume 98, pages 445–449, 1998. 90

[99] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015. 114

[100] T. Lee and A. Mihailidis. An intelligent emergency response system: preliminary development and testing of automated fall detection. *Journal of Telemedicine and Telecare*, 11(4):194–198, 2005. 38, 47, 53

[101] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, 2019. 45

[102] M. Li, S. Tian, L. Sun, and X. Chen. Gait analysis for post-stroke hemiparetic patient by multi-features fusion method. *Sensors*, 19(7), 2019. 113

[103] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In *2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, pages 138–143, June 2009. 39

[104] Y. Li, K. C. Ho, and M. Popescu. A microphone array system for automatic fall detection. *IEEE Transactions on Biomedical Engineering*, 59(5):1291–1301, May 2012. 40, 49, 50, 55

[105] Y. Li, K. C. Ho, and M. Popescu. Efficient source separation algorithms for acoustic fall detection using a microsoft kinect. *IEEE Transactions on Biomedical Engineering*, 61(3):745–755, March 2014. 40, 49, 50, 55

[106] X. Lian and J. Liu. Revisit batch normalization: New understanding and refinement via composition optimization. In *Proceedings of Machine Learning Research*, volume 89, pages 3254–3263, 2019. 148

[107] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991. 94

[108] J. Lines, L. Davis, J. Hills, and A. Bagnall. A shapelet transform for time series classification. pages 289–297, 08 2012. 46

[109] F. T. Liu, K. M. Ting, and Z. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, Dec 2008. 44

[110] L. Liu, M. Popescu, M. Skubic, M. Rantz, T. Yardibi, and P. Cuddihy. Automatic fall detection based on doppler radar motion signature. In *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 222–225, May 2011. 40, 49, 51, 55

[111] I. C. Lopes, B. Vaidya, and J. J. Rodrigues. Towards an autonomous fall detection and alerting system on a mobile and pervasive environment. *Telecommunication Systems*, 52(4):2299–2310, 2013. 39

[112] S. R. Lord, C. Sherrington, H. B. Menz, and J. C. T. Close. *Falls in Older People: Risk Factors and Strategies for Prevention*. Cambridge University Press, 2 edition, 2007. 21, 31, 77

[113] O. P. Mahela, A. G. Shaik, and N. Gupta. A critical review of detection and classification of power quality events. *Renewable and Sustainable Energy Reviews*, 41:495 – 505, 2015. 42, 46

[114] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach. Supervised dictionary learning. In *Advances in neural information processing systems*, pages 1033–1040, 2009. 113

[115] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010. 113

[116] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *COLT 2009 - The 22nd Conference on Learning Theory*, 02 2009. 89

[117] J. S. McPhee, D. P. French, D. Jackson, J. Nazroo, N. Pendleton, and H. Degens. Physical activity in older age: perspectives for healthy ageing and frailty. *Biogerontology*, 17(3):567–580, 2016. 21, 30, 112

[118] S. . Miaou, Pei-Hsu Sung, and Chia-Yuan Huang. A customized human fall detection system using omni-camera images and personal information. In *1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare, 2006. D2H2.*, pages 39–42, April 2006. 47, 53

[119] S. G. Miaou, P. H. Sung, and C. Y. Huang. A customized human fall detection system using omni-camera images and personal information. *Conference Proceedings - 1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare, D2H2 2006*, 2006:39–42, 2006. 38

[120] L. Minvielle and J. Audiffren. Nursenet: Monitoring elderly levels of activity with a piezoelectric floor. *Sensors*, 19(18), 2019. 112

[121] 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. 57

[122] 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. 84

[123] A. Morales-Esteban, F. Martínez-Álvarez, A. Troncoso, J. Justo, and C. Rubio-Escudero. Pattern recognition to forecast seismic time series. *Expert Systems with Applications*, 37(12):8333 – 8342, 2010. 42

[124] J. E. Morley, B. Vellas, G. A. Van Kan, S. D. Anker, J. M. Bauer, R. Bernabei, M. Cesari, W. Chumlea, W. Doehner, J. Evans, et al. Frailty consensus: a call to action. *Journal of the American Medical Directors Association*, 14(6):392–397, 2013. 20, 30

[125] M. Mubashir, L. Shao, and L. Seed. A survey on fall detection: Principles and approaches. *Neurocomputing*, 100:144–152, 2013. 36, 40

[126] A. Muro-De-La-Herran, B. Garcia-Zapirain, and A. Mendez-Zorrilla. Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications. *Sensors*, 14(2):3362–3394, 2014. 112

[127] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 117

[128] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, volume 269, pages 543–547, 1983. 118, 147

[129] Y. Nizam, M. N. H. Mohd, and M. M. A. Jamil. Human fall detection from depth images using position and velocity of subject. *Procedia Computer Science*, 105:131 – 137, 2017. 2016 IEEE International Symposium on Robotics and Intelligent Sensors, IRIS 2016, 17-20 December 2016, Tokyo, Japan. 47, 53

[130] N. Noury, a. Fleury, P. Rumeau, a.K. Bourke, G. Laighin, V. Rialle, and J. Lundy. Fall detection - Principles and Methods. *Conf Proc IEEE-EMBS*, pages 1663–1666, 2007. 36

[131] M. Nyan, F. Tay, A. Tan, and K. Seah. Distinguishing fall activities from normal activities by angular rate characteristics and high-speed camera characterization. *Medical Engineering and Physics*, 28(8):842 – 849, 2006. 39, 48, 49, 54

[132] R. J. Orr and G. D. Abowd. The smart floor: A mechanism for natural user identification and tracking. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '00, pages 275–276, New York, NY, USA, 2000. ACM. 59

[133] M. Otero. Application of a continuous wave radar for human gait recognition. In *Signal Processing, Sensor Fusion, and Target Recognition XIV*, volume 5809, pages 538 – 548. International Society for Optics and Photonics, SPIE, 2005. 112

[134] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al. Deepid-net: Deformable deep convolutional neural networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2015. 114

[135] M. Paajanen, J. Lekkala, and K. Kirjavainen. Electromechanical film (emfi) — a new multipurpose electret material. *Sensors and Actuators A: Physical*, 84(1):95 – 102, 2000. 60

[136] K. K. Pal and K. Sudeep. Preprocessing for image classification by convolutional neural networks. In *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pages 1778–1781. IEEE, 2016. 119

[137] S. Pan, T. Yu, M. Mirshekari, J. Fagert, A. Bonde, O. J. Mengshoel, H. Y. Noh, and P. Zhang. Footprintid: Indoor pedestrian identification through ambient structural vibration sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):89, 2017. 112

[138] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010. 87

[139] N. Pannurat, S. Thiemjarus, and E. Nantajeewarawat. Automatic fall monitoring: a review. *Sensors*, 14(7):12900–12936, 2014. 36, 39, 40, 52

[140] S. Peignier. Transfer learning synthetic data generator. https://github.com/SergioPeignier/TLSyntheticDataGenerator. 87, 102

[141] J. T. Perry, S. Kellog, S. M. Vaidya, J.-H. Youn, H. Ali, and H. Sharif. Survey and evaluation of real-time fall detection approaches. In *2009 6th International Symposium on High Capacity Optical Networks and Enabling Technologies (HONET)*, pages 158–164. IEEE, 2009. 36

[142] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Processing*, 99:215 – 249, 2014. 44

[143] N. Poschadel, S. Moghaddamnia, J. C. Alcaraz, M. Steinbach, and J. Peissig. A dictionary learning based approach for gait classification. In *2017 22nd International Conference on Digital Signal Processing (DSP)*, pages 1–4, 2017. 113, 122

[144] V. Radha and C. Vimala. A review on speech recognition challenges and approaches. *World of Computer Science and Information Technology Journal (WCSIT)*, 2(1):1–7, 2012. 42

[145] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 114, 149

[146] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 114, 117, 124, 125, 149, 150

[147] R. M. Rifkin. *Everything old is new again: a fresh look at historical approaches in machine learning*. PhD thesis, MaSSachuSettS InStitute of Technology, 2002. 124

[148] H. Rimminen, J. Lindström, R. Sepponen, et al. Positioning accuracy and multi-target separation with a human tracking system using near field imaging. *International Journal on Smart Sensing and Intelligent Systems*, 2(1):156–175, 2009. 112

[149] H. Rimminen, J. Lindström, M. Linnavuo, and R. Sepponen. Detection of falls among the elderly by a floor sensor using the electric near field. *IEEE Transactions on Information Technology in Biomedicine*, 14(6):1475–1476, Nov 2010. 41, 55

[150] J. L. Robinson and G. L. Smidt. Quantitative gait evaluation in the clinic. *Physical Therapy*, 61(3):351–353, 1981. 122

[151] R. V. Rodríguez, R. P. Lewis, J. S. Mason, and N. W. Evans. Footstep recognition for a smart home environment. *International Journal of Smart Home*, 2(2):95–110, 2008. 113

[152] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau. Monocular 3d head tracking to detect falls of elderly people. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6384–6387, Aug 2006. 47

[153] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau. Fall detection from human shape and motion history using video surveillance. In *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, volume 2, pages 875–880, May 2007. 47, 53

[154] C. Rougier, E. Auvinet, J. Rousseau, M. Mignotte, and J. Meunier. Fall detection from depth map video sequences. In *Proceedings of the 9th International Conference on Toward Useful Services for Elderly and People with Disabilities: Smart Homes and Health Telematics*, page 121–128, 2011. 38, 39, 47, 53

[155] J. Rubin, S. Parvaneh, A. Rahman, B. Conroy, and S. Babaeizadeh. Densely connected convolutional networks and signal quality analysis to detect atrial fibrillation using short single-lead ecg recordings. In *2017 Computing in Cardiology (CinC)*, pages 1–4. IEEE, 2017. 114

[156] R. Rubinstein, M. Zibulevsky, and M. Elad. Double sparsity: Learning sparse dictionaries for sparse signal approximation. *IEEE Transactions on signal processing*, 58(3): 1553–1564, 2009. 119

[157] I. d. C. N. Sacco and A. C. Amadio. A study of biomechanical parameters in gait analysis and sensitive cronaxie of diabetic neuropathic patients. *Clinical Biomechanics*, 15(3):196–202, 2000. 113

[158] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000. 44

[159] 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. 87, 91, 92, 94

[160] R. Serra. *Développement et caractérisation d'un système de sol piézoélectrique intelligent. Application à la détection des chutes.* PhD thesis, Université de Strasbourg, 2017. 62

[161] R. Serra, D. Knittel, P. Di Croce, and R. Peres. Activity recognition with smart polymer floor sensor: Application to human footstep recognition. *IEEE Sensors Journal*, 16(14):5757–5775, July 2016. 21, 31, 41, 62, 112, 113, 115, 116

[162] S. Sinha and S. Deb. Depth sensor based skeletal tracking evaluation for fall detection systems. *International Journal of Computer Trends and Technology*, 9(7):350–354, 2014. 39

[163] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 118, 148

[164] K. Sternickel. Automatic pattern recognition in ecg time series. *Computer Methods and Programs in Biomedicine*, 68(2):109 – 115, 2002. ISSN 0169-2607. 42, 46

[165] E. E. Stone and M. Skubic. Fall detection in homes of older adults using the microsoft kinect. *IEEE Journal of Biomedical and Health Informatics*, 19(1):290–301, Jan 2015. 39, 47, 51, 53

[166] B. Y. Su, K. C. Ho, M. J. Rantz, and M. Skubic. Doppler radar fall activity detection using the wavelet transform. *IEEE Transactions on Biomedical Engineering*, 62(3): 865–875, March 2015. 40, 49, 51, 55

[167] K. Suder, F. R. Drepper, M. Schiek, and H.-H. Abel. One-dimensional, nonlinear determinism characterizes heart rate pattern during paced respiration. *American Journal of Physiology-Heart and Circulatory Physiology*, 275(3):H1092–H1102, 1998. 114

[168] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358 – 3378, 2007. 90

[169] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1139–1147, 17–19 Jun 2013. 147

[170] J. Suutala and J. Röning. Methods for person identification on a pressure-sensitive floor: Experiments with multiple classifiers and reject option. *Information Fusion*, 9 (1):21 – 40, 2008. Special Issue on Applications of Ensemble Methods. 60, 113

[171] J. Suutala, K. Fujinami, and J. Röning. Gaussian process person identifier based on simple floor sensors. In *Smart Sensing and Context*, pages 55–68, Berlin, Heidelberg, 2008. Springer. 59, 112

[172] J. Taborri, E. Palermo, S. Rossi, and P. Cappa. Gait partitioning methods: A systematic review. *Sensors*, 16(1), 2016. 112, 113

[173] O. Tanaka, T. Ryu, A. Hayashida, V. G. Moshnyaga, and K. Hashimoto. A smart carpet design for monitoring people with dementia. In *Progress in Systems Engineering*, pages 653–659, Cham, 2015. Springer International Publishing. 60, 112

[174] B. B. Thompson, R. J. Marks, J. J. Choi, M. A. El-Sharkawi, Ming-Yuh Huang, and C. Bunje. Implicit learning in autoencoder novelty assessment. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, volume 3, pages 2878–2883, 2002. 45

[175] M. Tolkiehn, L. Atallah, B. Lo, and G.-Z. Yang. Direction sensitive fall detection using a triaxial accelerometer and a barometric pressure sensor. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 369–372. IEEE, 2011. 39

[176] T. Tommasi, F. Orabona, and B. Caputo. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. pages 3081–3088, 12 2010. 89

[177] C. Truong, R. Barrois-Müller, T. Moreau, C. Provost, A. Vienne-Jumeau, A. Moreau, P.-P. Vidal, N. Vayatis, S. Buffat, A. Yelnik, D. Ricard, and L. Oudre. A Data Set for the Study of Human Locomotion with Inertial Measurements Units. *Image Processing On Line*, 9:381–390, 2019. doi: 10.5201/ipol.2019.265. 23, 32

[178] C. Truong, L. Oudre, and N. Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020. 43

[179] H.-W. Tzeng, M.-Y. Chen, and J.-Y. Chen. Design of fall detection system with floor pressure and infrared image. In *2010 International Conference on System Science and Engineering*, pages 131–135. IEEE, 2010. 41

[180] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, Sep 2013. 149

[181] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, page 216–220, 2017. 119

[182] United Nations, Department of Economic and Social Affairs, Population Division. World Population Prospects: The 2019 Revision. URL https://population.un.org/wpp/. 20, 30

[183] P. Vallabh and R. Malekian. Fall detection monitoring systems: a comprehensive review. *Journal of Ambient Intelligence and Humanized Computing*, 9(6):1809–1833, Nov 2018. 36, 40

[184] M. Valtonen, J. Maentausta, and J. Vanhala. Tiletrack: Capacitive human tracking using floor tiles. In *2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–10, March 2009. 58, 112

[185] B. Vellas. Implementing frailty screening, assessment, and sustained intervention: the experience of the gérontopôle. *The journal of nutrition, health & aging*, 19(6): 673–680, 2015. 21, 31, 112

[186] V. Vishwakarma, C. Mandal, and S. Sural. Automatic detection of human fall in video. In *Pattern Recognition and Machine Intelligence*, 2007. 38, 47, 53

[187] H. Wang, D. Zhang, Y. Wang, J. Ma, Y. Wang, and S. Li. Rt-fall: A real-time and contactless fall detection system with commodity wifi devices. *IEEE Transactions on Mobile Computing*, 16(2):511–526, Feb 2017. 41, 51, 55

[188] W. Wang, A. X. Liu, and M. Shahzad. Gait recognition using wifi signals. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 363–373, 2016. 112

[189] Y. Wang, K. Wu, and L. M. Ni. Wifall: Device-free fall detection by wireless networks. *IEEE Transactions on Mobile Computing*, 16(2):581–594, Feb 2017. 41, 51, 55

[190] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016. 86, 87

[191] K. R. Weiss and T. M. Khoshgoftaar. Investigating transfer learners for robustness to domain class imbalance. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 207–213, Dec 2016. 90, 103

[192] K. R. Weiss and T. M. Khoshgoftaar. Comparing transfer learning and traditional learning under domain class imbalance. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 337–343, Dec 2017. 103

[193] F. Werner, J. Diermaier, S. Schmid, and P. Panek. Fall detection with distributed floor-mounted accelerometers: An overview of the development and evaluation of a fall detection system within the project ehome. In *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 354–361, May 2011. 40, 50, 55

[194] B. Wohlberg. Efficient convolutional sparse coding. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7173–7177, 2014. 121

[195] B. Wohlberg. Sporco: A python package for standard and convolutional sparse representations. In *Proceedings of the 15th Python in Science Conference, Austin, TX, USA*, pages 1–8, 2017. 122

[196] World Health Organization. WHO Global Report on Falls Prevention in Older Age. *Community Health*, page 53, 2007. URL https://www.who.int/ageing/publications/Falls_prevention7March.pdf. 20, 21, 29, 30, 31, 59

[197] H. Wu, B. Salzberg, and D. Zhang. Online event-driven subsequence matching over financial data streams. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, page 23–34, New York, NY, USA, 2004. ISBN 1581138598. 42

[198] Xiaodan Zhuang, J. Huang, G. Potamianos, and M. Hasegawa-Johnson. Acoustic fall detection using gaussian mixture models and gmm supervectors. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 69–72, April 2009. 40, 49, 50, 55

[199] T. Xu, Y. Zhou, and J. Zhu. New advances and challenges of fall detection systems: A survey. *Applied Sciences*, 8(3):418, 2018. 36, 40, 52

[200] L. S. Yaeger, R. F. Lyon, and B. J. Webb. Effective training of a neural network character classifier for word recognition. In *Advances in neural information processing systems*, pages 807–816, 1997. 69

[201] J. Yang, R. Yan, and A. G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 188–197. ACM, 2007. 89

[202] L. Ye and E. Keogh. Time series shapelets: A new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 947–956, 2009. 43

[203] H. Ying, C. Silex, A. Schnitzer, S. Leonhardt, and M. Schiek. Automatic step detection in the accelerometer signal. In *4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*, pages 80–85. Springer, 2007. 113

[204] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014. 118

[205] M. Yu, Y. Yu, A. Rhuma, S. M. R. Naqvi, L. Wang, and J. A. Chambers. An online one class support vector machine-based person-specific fall detection system for monitoring an elderly individual in a room environment. *IEEE Journal of Biomedical and Health Informatics*, 17(6):1002–1014, Nov 2013. 47, 48, 51, 53

[206] X. Yu. Approaches and principles of fall detection for elderly and patient. In *HealthCom 2008-10th International Conference on e-health Networking, Applications and Services*, pages 42–47. IEEE, 2008. 36, 38

[207] J. Yun, W. Woo, and J. Ryu. User identification using user's walking pattern over the ubifloorii. In *Computational Intelligence and Security*, pages 949–956, Berlin, Heidelberg, 2005. Springer. 59, 112

[208] C. Zhang, W. Liu, H. Ma, and H. Fu. Siamese neural network based gait recognition for human identification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2832–2836, 2016. 113

[209] Y. Zhang, P. O. Ogunbona, W. Li, B. Munro, and G. G. Wallace. Pathological gait detection of parkinson's disease using sparse representation. In *2013 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8. IEEE, 2013. 113, 122

[210] Z. Zhang, C. Conly, and V. Athitsos. A survey on vision-based fall detection. In *Proceedings of the 8th ACM international conference on PErvasive technologies related to assistive environments*, page 46. ACM, 2015. 36, 38

[211] X. Zhao. Awesome domain adaptation. https://github.com/zhaoxin94/awsome-domain-adaptation, 2019. 88

[212] F. Zhuang. Transfer learning toolkit. https://https://github.com/FuzhenZhuang/Transfer-Learning-Toolkit, 2019. 88

[213] F. Zhuang, K. Duan, T. Guo, Y. Zhu, D. Xi, Z. Qi, and Q. He. Transfer learning toolkit: Primers and benchmarks. *arXiv preprint arXiv 1911.08967*, 2019. 88

[214] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *arXiv preprint arXiv 1911.02685*, 2019. 87, 89

[215] Y. Zigel, D. Litvak, and I. Gannot*. A method for automatic fall detection of elderly people using floor vibrations and sound—proof of concept on human mimicking doll falls. *IEEE Transactions on Biomedical Engineering*, 56(12):2858–2867, Dec 2009. 40, 49, 51, 55

université PARIS-SACLAY

ECOLE DOCTORALE N°574
Mathématiques Hadamard (EDMH)

Titre : Classification d'événements à partir de capteurs sol – application au suivi de personnes fragiles

Mots clés : Détection de chute, Apprentissage par transfert, Arbre de décision, Forêt aléatoire, Réseau de neurones convolutif

Résumé : Cette thèse porte sur la détection d'événements dans des signaux issus de capteurs sols pour le suivi des personnes âgées. Au vu des questions pratiques, il semble que les capteurs de pression situés au sol soient des candidats prometteurs pour les activités de suivi, notamment la détection de chute. Les signaux étant complexes, il convient d'utiliser des modèles sophistiqués. Afin de concevoir un détecteur de chutes, nous proposons une approche basée sur les forêts aléatoires, tout en répondant aux contraintes matérielles à l'aide d'une procédure de sélection des variables. Les performances sont améliorées à l'aide d'une méthode d'augmentation des données ainsi qu'à l'agrégation temporelle des réponses du modèle. Nous abordons ensuite la question de la confrontation de notre modèle au monde réel, avec des méthodes d'apprentissage par transfert qui agissent sur le modèle de base des forêts aléatoires, c'est-à-dire les arbres de décision. Ces méthodes sont des adaptations de travaux antérieurs aux nôtres et sont conçues pour aborder le problème de déséquilibre des classes, la chute étant un événement rare. Nous les testons sur plusieurs ensembles de données, montrant ainsi des résultats encourageants pour la suite, et une implémentation Python est mise à disposition. Enfin, motivés par la question du suivi des personnes âgées tout en traitant un signal unidimensionnel pour une grande zone, nous proposons de distinguer les personnes âgées des individus plus jeunes grâce à un modèle de réseau de neurones convolutifs et un apprentissage de dictionnaire. Les signaux à traiter étant principalement constitués de marches, la première brique du modèle est entrainée pour se focaliser sur les pas dans les signaux, et la seconde partie du modèle est entraînée séparément. Cette nouvelle approche de la classification de la marche permet de reconnaître avec efficacité les signaux issus de personnes âgées.

Title : Event classification from floor sensor – application to elderly monitoring

Keywords : Fall detection, Transfer learning, Decision tree, Random forest, Convolutional neural network

Abstract : This thesis addresses the subject of event detection in temporal signals for elderly monitoring by the use of a floor pressure sensor. We first show that most proposed systems do not meet main practical issues and that floor systems constitute promising candidates for monitoring tasks. Since complex signals require sophisticated models, we propose a random-forest-based approach that detects falls with state-of-the-art accuracy and meets hardware constraints with a feature selection procedure. The model performance is improved with data augmentation and time aggregation of the random forest outputs. Then, we address the issue of confronting our model to the real world with transfer learning methods that act on the core model of random forests, i.e. decision trees. These methods are adaptations of seminal work and are designed to tackle the class imbalance problem as falls are rare events. Methods are tested on several data sets, showing interesting potential continuation, and a Python implementation is made available. Finally, motivated by the issue of elderly monitoring while dealing with one-dimensional signals for a large areas, we propose to distinguish elderly persons from younger individuals with a model based on convolutional neural network and convolutional dictionary learning. Since signals are mainly made of walks, the first part of the model is trained to recognize steps, and the last part of the model is trained with all previous layers frozen. This novel approach to gait classification allows to isolate elderly-generated signals with very high accuracy.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France