Automatic Non Linear Metric Learning - Application to Gesture Recognition

Samuel Charles Berlemont

N°d'ordre NNT : 2016LYSEI014

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
préparée au sein de
l'INSA LYON

Ecole Doctorale 512
Ecole doctorale InfoMaths

Spécialité de doctorat : Mathématiques

Soutenue publiquement le 11/02/2016, par :
Samuel Charles BERLEMONT

Automatic Non Linear Metric Learning Application to Gesture Recognition

---

Devant le jury composé de :

PELLERIN, Denis	Professeur	Université Grenoble Alpes	Président
CHATEAU, Thierry	Professeur	Université Blaise Pascal	Rapporteur
PAINDAVOINE, Michel	Professeur	Université de Bourgogne	Rapporteur
THOME, Nicolas	MC – HDR	Université P. et M. Curie	Examinateur
GARCIA, Christophe	Professeur	INSA de Lyon	Directeur de thèse
DUFFNER, Stefan	Maître de Conférences	INSA de Lyon	Co-directeur de thèse
LEFEBVRE, Grégoire	Ingénieur de Recherche, Docteur	Orange Labs	Co-encadrant

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec : Renée EL MELHEM Bat Blaise Pascal 3e etage secretariat@edchimie-lyon.fr Insa : R. GOURDON	M. Stéphane DANIELE Institut de Recherches sur la Catalyse et l'Environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 avenue Albert Einstein 69626 Villeurbanne cedex directeur@edchimie-lyon.fr
E.E.A.	**ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec : M.C. HAVGOUDOUKIAN Ecole-Doctorale.eea@ec-lyon.fr	M. Gérard SCORLETTI Ecole Centrale de Lyon 36 avenue Guy de Collongue 69134 ECULLY Tél : 04.72.18 60.97 Fax : 04 78 43 37 17 Gerard.scorletti@ec-lyon.fr
E2M2	**EVOLUTION, ECOSYSTEME, MICROBIOLOGIE, MODELISATION** **http://e2m2.universite-lyon.fr** Sec : Safia AIT CHALAL Bat Darwin - UCB Lyon 1 04.72.43.28.91 Insa : H. CHARLES Safia.ait-chalal@univ-lyon1.fr	Mme Gudrun BORNETTE CNRS UMR 5023 LEHNA Université Claude Bernard Lyon 1 Bât Forel 43 bd du 11 novembre 1918 69622 VILLEURBANNE Cédex Tél : 06.07.53.89.13 e2m2@ univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTE** **http://www.ediss-lyon.fr** Sec : Safia AIT CHALAL Hôpital Louis Pradel - Bron 04 72 68 49 09 Insa : M. LAGARDE Safia.ait-chalal@univ-lyon1.fr	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 avenue Jean Capelle INSA de Lyon 696621 Villeurbanne Tél : 04.72.68.49.09 Fax :04 72 68 49 16 Emmanuelle.canet@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHEMATIQUES** **http://infomaths.univ-lyon1.fr** Sec :Renée EL MELHEM Bat Blaise Pascal 3e etage infomaths@univ-lyon1.fr	Mme Sylvie CALABRETTO LIRIS – INSA de Lyon Bat Blaise Pascal 7 avenue Jean Capelle 69622 VILLEURBANNE Cedex Tél : 04.72. 43. 80. 46 Fax 04 72 43 16 87 Sylvie.calabretto@insa-lyon.fr
Matériaux	**MATERIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Saint Exupéry Ed.materiaux@insa-lyon.fr	M. Jean-Yves BUFFIERE INSA de Lyon MATEIS Bâtiment Saint Exupéry 7 avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72.43 71.70 Fax 04 72 43 85 28 Ed.materiaux@insa-lyon.fr
MEGA	**MECANIQUE, ENERGETIQUE, GENIE CIVIL, ACOUSTIQUE** **http://mega.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Saint Exupéry mega@insa-lyon.fr	M. Philippe BOISSE INSA de Lyon Laboratoire LAMCOS Bâtiment Jacquard 25 bis avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72 .43.71.70 Fax : 04 72 43 72 37 Philippe.boisse@insa-lyon.fr
ScSo	**ScSo*** **http://recherche.univ-lyon2.fr/scso/** Sec : Viviane POLSINELLI Brigitte DUBOIS Insa : J.Y. TOUSSAINT viviane.polsinelli@univ-lyon2.fr	Mme Isabelle VON BUELTZINGLOEWEN Université Lyon 2 86 rue Pasteur 69365 LYON Cedex 07 Tél : 04.78.77.23.86 Fax : 04.37.28.04.48

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Remerciements

Cette simple partie ne me permettra sûrement pas de remercier suffisamment tous ceux qui ont joué un rôle majeur durant ce long parcours qu'est une thèse, mais je vais quand même m'essayer à l'exercice.

Tout d'abord, je souhaite remercier tous mes encadrants, avec qui nous avons créé une équipe avec une dynamique excellente dans le cadre d'une thèse CIFRE. J'ai ainsi eu la chance d'être suivi quotidiennement chez Orange Labs par M. Grégoire Lefèbvre, dont l'intérêt, l'écoute, les qualités pédagogiques et les conseils particulièrement justes m'ont aidé à mener mes projets à bien, malgré les incertitudes et les échecs rencontrés en route. Un énorme merci également à mes directeurs de thèse Christophe Garcia et Stefan Duffner au laboratoire LIRIS, qui ont su me suivre malgré la distance et partager avec moi leur expérience inestimable.

Mes remerciements se dirigent ensuite vers tous les membres du Jury, en particulier à Messieurs Thierry Chateau et Michel Paindavoine pour le temps qu'ils ont pu accorder à relire mes travaux et leurs remarques très intéressantes et constructives. Merci aussi à M. Denis Pellerin d'avoir accepté de présider ce jury de soutenance, et à M. Nicolas Thome pour son intérêt pour mon travail et ses nombreuses suggestions.

Je tiens enfin à remercier toutes les personnes qui, sans directement participer à mes travaux, ont aidé à me faire avancer. Merci à tout mon entourage chez Orange Labs, notamment à l'équipe TAP, pour leur accueil chaleureux, leur bonne humeur quotidienne, et la richesse des expériences qu'ils ont partagées avec moi. Merci en particulier à Jean-Paul et Pyf sans qui je ne serais peut-être pas allé manger tous les jours. Une pensée également pour tous les thésards, en particulier Rick qui m'a montré l'exemple, et les stagiaires, avec l'éternel enthousiasme d'Andréa, les doutes partagés avec Julien, Vincent et Clémence.

Sur un plan plus personnel, merci à ma famille, parents et amis qui ont pu me supporter et m'encourager quand les résultats n'étaient pas au rendez-vous ; avec une petite mention spéciale à Céline, qui a été là pour m'inspirer et m'encourager.

Enfin, un grand merci à tous ceux que je n'ai pas pu nommer mais qui se reconnaîtront.

Abstract

As consumer devices become more and more ubiquitous, new interaction solutions are required. In this thesis, we explore inertial-based gesture recognition on Smartphones, where gestures holding a semantic value are drawn in the air with the device in hand.

Based on accelerometer and gyrometer data, three main approaches exist in the literature. The earliest methods suggest to model the temporal structure of a gesture class, with Hidden Markov Models for example; while another approach consists in matching gestures with reference instances, using a non-linear distance measure generally based on Dynamic Time Warping. Finally, features can be extracted from gesture signals in order to train specific classifiers, such as Support Vector Machines.

In our research, speed and delay constraints required by an application are critical, leading us to the choice of neural-based models. While Bi-Directional Long Short-Term Memory and Convolutional neural networks have already been investigated, the main issue is to tackle an open-world problem, which does not only require a good classification performance but, above all, an excellent capability to reject unknown classes.

Thus, our work focuses on metric learning between gesture sample signatures using the "Siamese" architecture (Siamese Neural Network, SNN), which aims at modelling semantic relations between classes to extract discriminative features, applied to the MultiLayer Perceptron.

Contrary to some popular versions of this algorithm, we opt for a strategy that does not require additional parameter fine tuning, namely a set threshold on dissimilar outputs, during training. Indeed, after a preprocessing step where the data is filtered and normalised spatially and temporally, the SNN is trained from sets of samples, composed of similar and dissimilar examples, to compute a higher-level representation of the gesture, where features are collinear for similar gestures, and orthogonal for dissimilar ones.

While the original model already works for classification, multiple mathematical problems which can impair its learning capabilities are identified.

Consequently, as opposed to the classical similar or dissimilar pair; or reference, similar and dissimilar sample triplet input set selection strategies, we propose to include samples from every available dissimilar classes, resulting in a better structuring of the output space. Moreover, we apply a regularisation on the outputs to better determine the objective function. Furthermore, the notion of polar sine enables a redefinition of the angular problem by maximising a normalised volume induced by the outputs of the

reference and dissimilar samples, which effectively results in a Supervised Non-Linear Independent Component Analysis. Finally, we assess the unexplored potential of the Siamese network and its higher-level representation for novelty and error detection and rejection. With the help of two real-world inertial datasets, the Multimodal Human Activity Dataset as well as the Orange Dataset, specifically gathered for the Smartphone inertial symbolic gesture interaction paradigm, we characterise the performance of each contribution, and prove the higher novelty detection and rejection rate of our model, with protocols aiming at modelling unknown gestures and open world configurations.

To summarise, the proposed SNN allows for supervised non-linear similarity metric learning, which extracts discriminative features, improving both inertial gesture classification and rejection.

Keywords: Gesture Recognition, MicroElectroMechanical Systems, Inertial Sensors, Machine Learning, Metric Learning, Similarity Metric, Artificial Neural Network, Siamese Network

Résumé

Alors que les appareils électroniques deviennent toujours plus omniprésents, de nouvelles solutions d'interfaces sont requises. Cette thèse explore la reconnaissance de gestes à partir de capteurs inertiels pour *Smartphone*. Ces gestes consistent en la réalisation d'un tracé dans l'espace présentant une valeur sémantique, avec l'appareil en main.

Trois principales approches sont identifiées dans la littérature pour l'exploitation des données accélérométriques et gyrométriques. Les premières méthodes cherchent à modéliser la structure temporelle de chaque classe de geste, avec, par exemple, les Modèles de Markov Cachés (*Hidden Markov Model*). Une autre approche consiste à faire correspondre les gestes avec des instances de référence à l'aide d'une mesure de distance non-linéaire, le plus souvent basée sur un alignement temporel dynamique (*Dynamic Time Warping*). La dernière stratégie propose d'extraire des valeurs caractéristiques des signaux gestuels afin d'entraîner des classifieurs spécifiques, tels que les Séparateurs à Vaste Marge (*Support Vector Machines*).

Dans notre contexte de recherche, les contraintes de rapidité d'exécution imposées par l'application jouent un rôle critique, nous orientant ainsi vers le choix de modèles neuronaux. Alors que les réseaux de neurones récurrents "longue mémoire à court terme" (*Long Short-Term Memory*) bidirectionnels et les réseaux de neurones convolutionnels ont déjà été explorés, notre problématique consiste en un "monde ouvert", qui nécessite non seulement de très bonnes performances en classification, mais aussi avant tout une grande capacité à rejeter des classes inconnues.

Ainsi, notre étude porte en particulier sur l'apprentissage de métrique entre signatures gestuelles grâce à l'architecture "Siamoise" (réseau de neurones siamois, *SNN*), qui a pour but de modéliser les relations sémantiques entre classes afin d'extraire des caractéristiques discriminantes. Cette architecture est appliquée au perceptron multicouche (*MultiLayer Perceptron*).

A l'inverse de versions plus populaires de cet algorithme, nous faisons le choix d'une stratégie qui ne requiert pas d'ajustement de paramètres supplémentaires au cours de l'apprentissage. Après une étape de prétraitements durant laquelle les données sont filtrées et normalisées en temps et en amplitude, le SNN est entraîné à partir d'ensembles d'échantillons, composés d'exemples similaires et dissimilaires, afin de produire une représentation supérieure, où les vecteurs caractéristiques sont colinéaires entre gestes similaires et orthogonaux entre gestes dissimilaires.

Alors que le modèle original du SNN a été principalement pensé pour des estimations de similarité, peu de modifications ont jusqu'alors été proposées afin d'exploiter l'ensemble des relations connues entre classes dans le cadre d'une problématique de classification. Les stratégies classiques de formation d'ensembles d'apprentissage sont essentiellement basées sur des paires similaires et dissimilaires, ou des triplets formés d'une référence et de deux échantillons respectivement similaires et dissimilaires à cette référence. Ainsi, nous proposons une généralisation de ces approches. Chaque ensemble d'apprentissage porte toutes les informations disponibles, avec une référence, un example positif, et un exemple négatif pour chaque classe dissimilaire, dans le but d'améliorer la structuration de l'espace de sortie du réseau.

Par ailleurs, plusieurs problèmes mathématiques dans la fonction d'erreur utilisée pouvant influer sur les capacités d'apprentissage du modèle sont indentifiés. Nous appliquons donc une régularisation particulière sur les sorties du réseau au cours de l'apprentissage afin de déterminer de manière plus précise la fonction objectif et limiter les variations de la norme moyenne des vecteurs caractéristiques obtenus.

Enfin, nous proposons une redéfinition du problème angulaire par une adaptation de la notion de sinus polaire. Il s'agit alors de maximiser un volume normalisé, induit par les vecteurs caractéristiques des échantillons pour un ensemble d'apprentissage, ce qui aboutit à une analyse en composantes indépendantes non-linéaire supervisée basée sur une nouvelle métrique de similarité pour un ensemble de vecteurs.

A l'aide de deux bases de données inertielles, la base d'activité humaine multimodale (*Multimodal Human Activity Dataset*) ainsi que la base Orange, spécifiquement collectée dans le cadre d'une interaction pour Smartphone basée sur des gestes symboliques inertiels, les performances de chaque contribution sont caractérisées et les meilleures capacités de détection et rejet de nouveauté du SNN sont prouvées au moyen de protocoles modélisant un monde ouvert, qui comprend des gestes inconnus par le système.

En résumé, le SNN proposé permet de réaliser un apprentissage supervisé de métrique de similarité non-linéaire, qui extrait des vecteurs caractéristiques discriminants, améliorant conjointement la classification et le rejet de gestes inertiels.

Mots-clés : Reconnaissance de Gestes, Systèmes MicroElectroMécaniques, Capteurs Inertiels, Apprentissage Automatique, Apprentissage de Métrique, Métrique de Similarité, Réseau de Neurones Artificiels, Réseau Siamois

French summary

0.1 Introduction

Le geste, et la reconnaissance de gestes en général, s'inscrivent dans un contexte d'intelligence ambiante (voir Fig.1.2), et plus précisément d'informatique ubiquitaire. En effet, selon Mark Weiser [Wei91], pionnier dans ce domaine, l'informatique ubiquitaire consiste à "penser les ordinateurs en considérant l'environnement humaine, afin qu'ils s'effacent dans le paysage".

Les interfaces homme-machine ont ainsi connu plusieurs stades d'évolution. Alors que les premières interfaces n'étaient accessibles qu'aux professionnels et experts, les interfaces graphiques ouvrirent la voie à l'informatique personnelle (*personal computing*) grâce à des concepts plus instinctifs, avec une manipulation d'objets concrets, tels que des dossiers, au moyen de nouveaux outils, tels que la souris. Cependant, ces solutions nécessitent toujours un effort important de la part de l'utilisateur.

Ainsi, nos travaux se trouvent à la frontière entre capteurs, intelligence artificielle et interfaces homme-machine. Il s'agit alors de développer de nouvelles solutions d'interaction tirant parti de la richesse de cette modalité naturelle qu'est le geste.

Nous étudions ici les gestes pouvant être intégrés dans une interface, gestes qui transmettent des informations à l'environnement. Ainsi, quatre catégories de gestes sont identifiées dans le continuum de Kendon, proposé par McNeill ([McN92]), pour la description des gestes : les gestes *iconiques*, *métaphoriques*, *déictiques*, et les *battements*. Il est d'ailleurs important de noter que ces catégories ne s'excluent pas mutuellement. Les gestes *iconiques* complètent l'image mentale du sujet discuté avec une illustration concrète de ce sujet, tandis que les gestes *métaphoriques* ne se limitent pas à une action ou objet concrets et peuvent représenter des concepts abstraits ou des expressions. Les gestes *déictiques* consistent en des mouvements de pointage, qui désignent aussi bien des objets physiques que des notions abstraites dans le temps et l'espace. Enfin, les *battements* sont de petits gestes utilisés par le locuteur comme procédé d'emphase sur certains mots ou phrases.

Par ailleurs, deux types de production de gestes existent : les gestes *statiques* sont relatifs à la posture de l'utilisateur, tandis que les gestes *dynamiques* sont caractérisés par la trajectoire d'un membre du corps.

Dans le cadre de nos travaux, nous cherchons à développer une solution d'apprentissage de métrique non linéaire, appliqué à la reconnaissance de gestes. Il s'agit alors de proposer une apprentissage de métrique directement à partir des données issues des capteurs inertiels présents dans les appareils équipés de capteurs MEMS (*MicroElectroMechanical Systems*). Nous nous concentrons alors sur la reconnaissance de gestes dynamiques et sémantiques, regroupant les types iconiques et métaphoriques avec des formes et des actions. Le domaine applicatif final visé concerne les terminaux mobiles,

dans un contexte de monde ouvert, et de personnalisation, où tous les gestes et tous les utilisateurs ne sont pas représentés au moment de l'apprentissage, ce qui introduit un nouveau problème de *rejet*, où le système doit identifier par lui-même si un geste réalisé a été appris.

Cinq propriétés sont recherchées (voir Fig.1.3). L'algorithme d'apprentissage doit être semi-supervisé, non-linéaire, évolutif par rapport au nombre d'exemples et leur dimension ; il doit également permettre une réduction dimensionnelle, et être applicable à l'ensemble de l'espace des gestes. Ces critères nous conduisent donc au choix du réseau de neurones siamois (*Siamese Neural Network*, SNN), qui apprend une métrique à l'aide d'informations de similarité, sans nécessité d'étiquetage des données, et qui extrait des vecteurs caractéristiques représentatifs de dimension ajustable.

Cette thèse est organisée comme suit. Après cette introduction, la section 0.2 présente les technologies d'acquisition de gestes, suivies d'un état-de-l'art des méthodes classiques employées pour la reconnaissance et classification de gestes. La section 0.3 aborde la notion de réseau de neurones siamois, avant de présenter nos contributions théoriques sur ce type de réseau. La section 0.4 analyse ces contributions à l'aide de tests de classification et de rejet basés sur deux bases de données de gestes 3D. Enfin, la section 0.5 conclut cette étude et présente les perspectives envisagées pour la suite de nos travaux.

0.2 Etat de l'art

Cette section présente un aperçu global de la reconnaissance de gestes en deux parties. Tout d'abord les principaux capteurs mis en jeu sont présentés, des capteurs inertiels MEMS aux méthodes de *motion capture* et de suivi de squelette, plus gourmandes en matériel. Dans un deuxième temps, nous présentons les différentes approches pour la reconnaissance de gestes, avec les méthodes statistiques, à base de classifieurs, et basées sur les réseaux de neurones artificiels.

0.2.1 Acquisition des données de geste

Capteurs inertiels

Les nouvelles applications des *Smartphones* doivent leur succès en grande partie à l'utilisation de nouveaux capteurs, en particulier les MEMS. Leur simplicité d'utilisation et leur faible coût de production basée sur les semi-conducteurs expliquent leur grand développement dans un grand éventail d'appareils, tels que les voitures pour les déclencheurs de déploiement d'airbag, ou les drones pour la navigation.

Trois capteurs sont les plus répandus : l'accéléromètre, le gyromètre et le magnétomètre. Pour notre étude, nous nous concentrons sur les capteurs les plus fiables, l'accéléromètre et le gyromètre, dont la combinaison améliore les résultats de classification (voir [BL]). Ces deux capteurs relèvent respectivement les accélérations et vitesses angulaires le long de trois axes, représentés Fig.2.1.

L'accéléromètre 3D est le résultat de la combinaison de trois accéléromètres uniaxiaux. Chaque accéléromètre 1D peut être vu comme un système masse-ressort, où le déplacement de la masse suite à l'application d'une force est directement corrélé à l'accélération selon le principe fondamental de la dynamique. Le gyromètre repose quant à lui sur le principe de Coriolis, qui veut qu'une masse oscillante cherche à continuer

d'osciller dans le même plan alors même que son support subit une rotation. L'objet applique donc une force sur son support, force dont la mesure permet de remonter à la vitesse angulaire.

Systèmes basés vision

Les applications de reconnaissance de gestes basés sur la vision par ordinateur s'appuient essentiellement sur l'identification et le suivi de squelettes. Une première stratégie consiste à suivre dans l'espace 3D des points identifiés au préalable sur différentes parties du corps afin de reconstruire le squelette ; tandis que la deuxième stratégie est plus facile à utiliser puisqu'elle reconstruit directement le squelette à partir d'images, sans nécessiter l'utilisation de marqueurs. L'application la plus célèbre de cette deuxième stratégie est le capteur Kinect, basé sur une source infrarouge, et deux capteurs d'images, de profondeur et de couleur. Ainsi, un motif est projeté sur la scène observée à l'aide de la source infrarouge, dont les déformations permettent une reconstruction en 3D de cette même scène. Enfin, la reconstruction du squelette s'opère par une étape d'identification des articulations. Ces opérations, très rapides, sont effectuées sur chaque image d'une résolution de 640×480 indépendamment, à raison de 30 images par seconde.

Prétraitements des données inertielles

Trois phases de prétraitements sont typiquement nécessaires dans le cadre d'une application de reconnaissance de gestes basée sur des capteurs inertiels : une phase de *calibration*, une phase de *filtrage*, et une phase de *normalisation*. En effet, en tant que système masse-ressort, l'accéléromètre présente le défaut d'interpréter le déplacement dû à la gravité comme une force verticale, dirigée vers le haut, appliquée sur le support. Ainsi, un accéléromètre ne présente de mesure nulle qu'en cas de chute libre. Il est alors nécessaire de gérer ce biais dans le signal, par exemple en opérant une rotation de ce signal afin d'assurer qu'une estimation de la direction verticale coïncide pour chaque échantillon [Pyl05]. Par ailleurs, une phase de filtrage permet de diminuer l'influence du bruit présent dans le signal, dû à de multiples facteurs, tels que le bruit électronique, mécanique, ou les mouvements involontaires des utilisateurs. Sont généralement employés des filtres passe-bas du premier ordre [Mli09], de butterworth du quatrième ordre [MMST00], ou un lissage au moyen d'une fenêtre glissante [HHH98] [AV10] [MHS01] [LC13]. Afin de rendre plusieurs dynamiques de production de gestes comparables, une phase de normalisation spatiale permet une réduction des différences d'amplitude entre plusieurs profils des données inertielles. Deux stratégies sont communément appliquées : une normalisation des distributions de probabilité vers une loi normale centrée réduite [MHS01] [MMST00] ; une mise à l'échelle linéaire "min-max", qui projette les relevés sur l'intervalle $[0; 1]$ [MKKK04][ZCL+11], ou une mise à l'échelle basée sur le relevé de norme maximale pour chaque échantillon [CCB+06] [HL10]. Enfin, une phase de normalisation temporelle peut être nécessaire, notamment afin de réduire la redondance dans les données avec une opération de seuillage [Mli09][SPHB08] ; ou pour les modèles ne permettant pas de gérer cette dimension, avec un ré-échantillonnage [KKM+05][CCB+06], ou une quantification vectorielle [HHH98][Kli09] [AV10][KKM+05].

0.2.2 Apprentissage et classification de données gestuelles

Bien que la reconnaissance de gestes représente un domaine relativement récent comparé à d'autres applications, cette discipline bénéficie de l'expérience accumulée pour d'autres problématiques telles que la reconnaissance de parole ou de visage. Trois

principales stratégies sont identifiées. La première stratégie consiste en une modélisation statistique de la dynamique du geste, notamment à l'aide de réseaux bayésiens [CCB+06], ou de modèles de markov cachés [HVL10][ZS09][LWJ+04][Pyl05][ZCL+11]. La seconde approche est basée sur des considérations géométriques. La distance élastique *Dynamic Time Warping* [BD13][CMC10][HL10][AV10][ZT09][ZDlT12] permet d'établir une mesure de similarité entre gestes de longueurs différentes, tandis que d'autres méthodes s'appuient sur d'autre caractéristiques géométriques tels que les *eigenvectors*, avec l'Analyse en Composantes Principales (PCA) [MP11][MHS01][YWC08], ou l'Analyse Discriminante Linéaire (LDA) [LWJ+04][MP11].

Enfin, la troisième approche consiste à produire un classifieur spécifique, notamment les classifieurs *Adaboost* [HVL10], les Séparateur à Vaste Marge (SVM) [HJZH08] [WPZ+09][CCB+06], ou les réseaux de neurones artificiels. Ces derniers sont représentés sous différentes architectures, telles que le Perceptron MultiCouches (MLP)[MHS01] [NH09][YWC08], le réseau de neurones récurrent "longue mémoire à court terme" bidirectionnel (BLSTM) [LBMG13][LBMG15], ou le réseau de neurones convolutionnel (CNN)[DBLG14].

Ainsi, à l'inverse des méthodes géométriques et statistiques qui présentent des difficultés de généralisation, les réseaux de neurones possèdent de bonnes caractéristiques grâce à leur non-linéarité. Cependant, les différents réseaux étudiés ne permettent pas de prendre en compte les informations de similarité entre échantillons, qui pourraient influencer la projection obtenue après apprentissage, ce qui justifie notre choix du réseau de neurones siamois.

0.3 Le réseau de neurones siamois

Dans la section précédente, nous avons présenté différents tupes de réseaux de neurones artificiels appliqués à la reconnaissance et classification de gestes. La plupart des réseaux peuvent à la fois extraire des caractéristiques, grâce aux couches de convolution pour les CNNs par exemple ; et réaliser une classification, généralement à l'aide d'une couche de sortie basée sur la fonction *Softmax*. Cependant, ces opérations se font automatiquement, de manière supervisée, sans possibilité de prendre en compte des informations sur les voisinages attendus pour l'espace caractéristique. Les régions de cet espace sont manuellement discrétisées et assignées à des classes. Ainsi, elles sont sémantiquement décorrélées, et les zones qui n'ont pas été attribuées ne portent pas la même quantité d'informations. L'architecture Siamoise, grâce à sa stratégie d'apprentissage semi-supervisée basée sur des relations entre plusieurs échantillons, permet une structuration différente de cet espace de sortie, de telle sorte que le sens assigné à une région évolue de manière continue.

Dans cette section, nous présentons d'abord les caractéristiques des différents composants formant le SNN, avant de définir nos contributions et leurs implications pour chacun de ces composants.

0.3.1 Etat de l'art

Le SNN tire son nom de sa stratégie d'apprentissage, qui met en oeuvre plusieurs copies identiques du même réseau en parallèle (voir Fig.1). Dans sa version originale, introduite par Bromley *et al.* [BGL+94], et Chopra *et al.* [CHL05], le réseau encode la similarité à l'aide d'une métrique contrôlée dans l'espace de sortie appliquée à des

paires d'échantillons. Une fonction d'erreur est alors définie, de telle manière que la similarité intra-classe soit forte, et la similarité inter-classe soit minime.

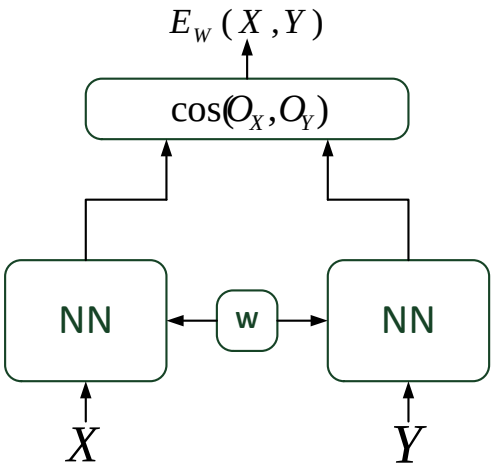


FIGURE 1 – Architecture du SNN originel. Deux réseaux partageant un même ensemble de poids W prennent deux échantillons $\mathbf{X}$ et $\mathbf{Y}$ en entrée, dans le but de calculer une erreur relative à une fonction de similarité définie sur l'espace de sortie (ici, la similarité cosinus), au moyen des vecteurs de sortie respectifs $\mathbf{O_X}$ et $\mathbf{O_Y}$.

Les principaux types de réseaux mis en jeu dans l'apprentissage Siamois sont les réseaux avec réglage en avant (*feed-forward neural network*), tels que les MLP et les CNN. Deux principaux composants peuvent alors varier lors de la définition d'un SNN : la stratégie de sélection des ensembles d'apprentissage, ainsi que la mesure de similarité utilisée dans l'espace de sortie, associée à la fonction objectif.

Sélection des ensembles d'apprentissage

Les ensembles d'apprentissage sont construits de telle manière à refléter la définition de la relation de similarité. Ainsi, l'apprentissage a lieu de manière stochastique à partir de ces relations de similarité entre ensembles d'échantillons. Deux principales approches sont identifiées : l'approche par paires, et l'approche par triplets. L'approche par paires est la stratégie la plus commune, retrouvée dans la version initiale du SNN : une relation de similarité est encodée à l'aide de deux échantillons, ainsi qu'une étiquette de similarité précisant si ces échantillons sont similaires ou dissimilaires. L'approche par triplets, proposée par Lefèbvre *et al.* [LG13], étend cette information en introduisant une représentation plus symétrique entre similarité et dissimilarité. Ainsi, un ensemble d'apprentissage est formé de trois échantillons, avec un échantillon de référence, un échantillon dit *positif*, similaire à la référence, et un échantillon *négatif*, dissimilaire de la référence.

Ainsi, les relations de similarité et leur définition ont un impact direct sur l'architecture du réseau lors de l'apprentissage, et peuvent être gérées différemment selon l'application finale de la métrique apprise.

Mesure de similarité et fonction objectif

Deux mesure de similarité, basées sur des considérations géométriques, sont typiquement utilisées dans l'espace de sortie : la similarité cosinus, et la distance euclidienne. Soit $\mathbf{X_1}$ et $\mathbf{X_2}$ deux vecteurs caractéristiques, ces mesures sont définies ainsi :

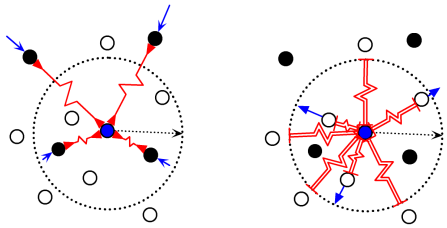


FIGURE 2 – Systèmes de ressorts. A gauche, deux échantillons similaires s'attirent, tandis que deux échantillons dissimilaires se repoussent à droite, avec une distance maximum représentée en pointillés à partir de laquelle les points sont suffisamment éloignés pour ne plus se repousser. Extrait de [HCL06].

— similarité cosinus

$$cos_{sim}(\mathbf{O_{X_1}}, \mathbf{O_{X_2}}) = 1 - \cos(\mathbf{O_{X_1}}, \mathbf{O_{X_2}})$$
$$\cos(\mathbf{O_{X_1}}, \mathbf{O_{X_2}}) = \frac{\mathbf{O_{X_1}}.\mathbf{O_{X_2}}}{\|\mathbf{O_{X_1}}\| \, . \, \|\mathbf{O_{X_2}}\|}$$

— distance Euclidienne

$$d(\mathbf{O_{X_1}}, \mathbf{O_{X_2}}) = \|\mathbf{O_{X_1}} - \mathbf{O_{X_2}}\|_2 \, .$$

Nous présenterons plus en détail les fonctions en relation avec la similarité cosinus, étudiée dans cette thèse. Trois approches basées sur la similarité cosinus sont identifiées. La première approche consiste à définir des cibles pour la valeur de la mesure cosinus, avec le *square error objective*. Soit un réseau avec un ensemble de poids W et deux échantillons $\mathbf{X_1}$ et $\mathbf{X_2}$, la cible $t_{\mathbf{X_1X_2}}$ est définie selon le label de similarité L entre les deux échantillons :

$$E_W(X_1, X_2, L) = (t_{\mathbf{X_1X_2}}(L) - \cos(\mathbf{O_{X_1}}, \mathbf{O_{X_2}}))^2, \tag{1}$$

Plusieurs variantes à existent, telles que le *triplet similarity objective* [LG13] ; la *triangular similarity metric* [ZIG+15], et la *deviance cost function* [YLLL14]. Il est également possible d'apprendre un classement de paires [YTPM11]. Il s'agit alors de maximiser la différence Δ entre des scores de similarité pour deux paires $(\mathbf{X_1}, \mathbf{X_2})$ et $(\mathbf{Y_1}, \mathbf{Y_2})$, lorsque la première paire est plus similaire que la seconde :

$$\Delta = \cos(\mathbf{O_{X_1}}, \mathbf{O_{X_2}}) - \cos(\mathbf{O_{Y_1}}, \mathbf{O_{Y_2}}). \tag{2}$$

Enfin, l'objectif probabiliste [NH10] repose sur la fonction logistique afin d'apprendre directement à évaluer la probabilité que deux échantillons soient similaires :

$$Pr(\text{"}Same\text{"}) = \frac{1}{1 + \exp(-(w.\cos(\mathbf{O_{X_1}}, \mathbf{O_{X_2}}) + b))}. \tag{3}$$

Les objectifs basés sur la distance euclidienne reposent essentiellement sur un principe de ressorts (voir Fig.2). Ainsi deux échantillons similaires, représentés en noir, sont connectés par des ressorts attractifs, tandis que des échantillons dissimilaires, en noir et blanc, sont connectés par des ressorts répulsifs. Cette répulsion est nulle lorsque la distance entre deux échantillons dissimilaires est supérieure à une marge m à définir au moment de l'apprentissage. Cette stratégie est bien illustrée avec la fonction objectif

de Hadsell *et al.* [HCL06] :

$$\begin{aligned} E_W(\mathbf{X_1}, \mathbf{X_2}) &= \|\mathbf{O_{X_1}} - \mathbf{O_{X_2}}\|_2 \\ Esim(E_W) &= \frac{1}{2}(E_W)^2 \\ E\overline{sim}(E_W) &= \frac{1}{2}(\max\{0, m - E_W\})^2 \end{aligned} \tag{4}$$

Ainsi, après avoir présenté les différents choix possibles pour les composants du SNN, nous présentons dans la partie suivante nos contributions pour chacun de ces composants.

0.3.2 Contributions

Dans le cadre de cette étude, nous basons l'architecture de notre SNN sur le MLP, afin de pouvoir suivre l'évolution des sorties de chacune des couches au cours de l'apprentissage, et caractériser plus facilement les propriétés de convergence des nouvelles fonctions objectifs proposées. L'objectif choisi est basé sur la similarité cosinus avec cibles, de telle manière que des échantillons similaires aient des vecteurs caractéristiques colinéaires, i.e. $t_{\mathbf{X_1X_2}} = 1$, et les échantillons dissimilaires aient des vecteurs caractéristiques orthogonaux, i.e. $t_{\mathbf{X_1X_2}} = 0$. Ce SNN opère une phase d'extraction de vecteurs caractéristiques, et est combiné à un classifieur k-plus-proches-voisins (K-NN) pour la phase de classification.

Nous proposons quatre principales contributions : avec une adaptation des ensembles d'apprentissage aux problèmes de classification multi-classes ; une relaxation de la similarité cosinus ; une redéfinition du problème angulaire avec une nouvelle métrique gérant un nombre arbitraire d'échantillons dissimilaires ; et enfin une exploration de nouvelles applications du SNN, notamment pour la détection de nouveauté et le rejet d'erreurs et éléments inconnus, effectué avec la stratégie classique d'un seuil unique pour toutes les classes.

Stratégie de sélection des ensembles d'apprentissage

Dans un problème de classification adapté au SNN, il est nécessaire au cours de l'apprentissage de traduire les étiquettes des échantillons en relations de similarité, ce qui introduit un biais. En effet, il est impossible de définir un ratio objectif entre le nombre de paires similaires et celui de paires dissimilaires à chaque mise à jour du réseau. Nous proposons donc une généralisation de la définition d'une relation de similarité dans un cadre de classification, à l'aide de *n-uplets*. Soit $C = \{C_1, .., C_K\}$ les classes représentées dans la base d'apprentissage ; et R_k, P_k, et N_l respectivement les échantillons de référence de la classe C_k, un échantillon positif de la même classe, et un échantillon négatif de la classe C_l, le n-uplet est alors défini ainsi :

$$T_k = \{\mathbf{X_{R_k}}, \mathbf{X_{P_k}}, \{\mathbf{X_{N_l}}, l = 1..K, l \neq k\}\}. \tag{5}$$

Cette représentation a pour avantage d'accorder un rôle symétrique à toutes les classes, et ainsi assurer une équité de représentations de toutes les relations présentes dans la base d'apprentissage.

La nouvelle fonction objectif intermédiaire est alors égale à :

$$E_W(T) = (1 - cos(\mathbf{O_R}, \mathbf{O_P}))^2 + \sum_l (0 - cos(\mathbf{O_R}, \mathbf{O_{N_l}}))^2. \tag{6}$$

Relaxation de la similarité cosinus

Soit $(\mathbf{X_1}, \mathbf{X_2})$ une paire de vecteurs caractéristiques dont les sorties doivent être mises à jour, et les fonctions $cos_{\mathbf{O^t_{X_1}}}$ et $cos_{\mathbf{O^t_{X_2}}}$, définies par :

$$\begin{aligned} cos_{\mathbf{O^t_{X_1}}} &: \mathbb{R}^n \to \mathbb{R}/\mathbf{X} \to \frac{1}{2}(1 - \cos(\mathbf{O^t_{X_1}}, \mathbf{X}))^2, \\ cos_{\mathbf{O^t_{X_2}}} &: \mathbb{R}^n \to \mathbb{R}/\mathbf{X} \to \frac{1}{2}(1 - \cos(\mathbf{X}, \mathbf{O^t_{X_2}}))^2. \end{aligned} \tag{7}$$

Ainsi, le gradient de la fonction objectif globale en $\left(\mathbf{O^t_{X_1}}, \mathbf{O^t_{X_2}}\right)$ peut être exprimée comme la concaténation des deux gradients des fonctions définies ci-dessus, respectivement évaluées en $\mathbf{O^t_{X_2}}$ et $\mathbf{O^t_{X_1}}$. La Figure 3 propose ainsi une étude de l'évolution de la norme de $\mathbf{O_{X_2}}$ après une mise à jour. Ainsi, à $\mathbf{O_{X_1}}$ fixé, le gradient $\nabla_{cos_{\mathbf{O^t_{X_1}}}}(\mathbf{O^t_{X_2}})$ est orthogonal à la surface équipotentielle en $\mathbf{O^t_{X_2}}$. Ce gradient est dont tangent au cercle de rayon égal à la norme de $\mathbf{O^t_{X_2}}$, la mise à jour $\mathbf{O^{t+1}_{X_2}}$ voit donc sa norme augmenter. Il est important de souligner que cette analyse ne tient pas compte de la non-linéarité introduite par le réseau, et repose uniquement dans l'espace de sortie.

Ainsi, afin de contrôler l'évolution de ces normes, nous proposons de subdiviser la similarité cosinus en trois cibles indépendantes, en contraignant à la fois le produit scalaire entre les deux vecteurs caractéristiques, et en appliquant une régularisation de la norme de ces derniers :

$$E_W(T_k) = (1 - \mathbf{O_R}.\mathbf{O_P})^2 + \sum_l (0 - \mathbf{O_R}.\mathbf{O_{N_j}})^2 + \sum_k (1 - \|\mathbf{O_k}\|)^2 \tag{8}$$

Redéfinition du problème angulaire : le SNN pour une analyse en composantes indépendantes

En dimension 2, il est possible de définir une mesure de dissimilarité entre deux vecteurs $\mathbf{a}$ et $\mathbf{b}$ comme une aire d'un polytope normalisée : $sin(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \wedge \mathbf{b}}{\|\mathbf{a}\|.\|\mathbf{b}\|}$. Ainsi, on cherche à définir une généralisation de cette mesure à une dimension n quelconque. Lerman *et al.* [LW09] proposent une fonction appelée *Sinus Polaire*, qui s'inspire de la définition du sinus. Soit une matrice $\mathbf{A} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix}$ composée de n colonnes linéairement indépendantes, avec $n < m$, où m est égal à la dimension de chaque colonne, le sinus polaire est définit comme :

$$PolarSine(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \frac{\sqrt{\det\left(\mathbf{A}^\top.\mathbf{A}\right)}}{\prod_{i=1}^n \|\mathbf{v_i}\|} \tag{9}$$

En posant $\mathbf{A_{norm}} = \begin{bmatrix} \frac{\mathbf{v}_1}{\|\mathbf{v_1}\|} & \frac{\mathbf{v}_2}{\|\mathbf{v_2}\|} & \cdots & \frac{\mathbf{v}_n}{\|\mathbf{v_n}\|} \end{bmatrix}$ et $\mathbf{S} = A^\top_{norm}.A_{norm}$, le sinus polaire peut alors être réinterprété comme :

$$PolarSine(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \sqrt{\det(S)}$$

avec $\mathbf{S}(i, j) = \cos(\mathbf{v_i}, \mathbf{v_j})$. Ainsi, la diagonale de la matrice $\mathbf{S}$ est toujours égale à 1 en tant que valeurs de cosinus entre deux vecteurs identiques, et forcer le déterminant de cette matrice à 0 revient à la faire tendre vers la matrice identité, ce qui force le critère d'orthogonalité recherché entre des éléments indépendants.

Nous proposons alors une adaptation de ce sinus polaire, qui est à l'origine numériquement instable lors d'une opération de descente de gradient. Ainsi, nous définissons la métrique sinus polaire, notée *psin*, comme :

$$psin(\mathbf{A}) = \sqrt[n]{\det\left(\mathbf{S}\right)} \tag{10}$$

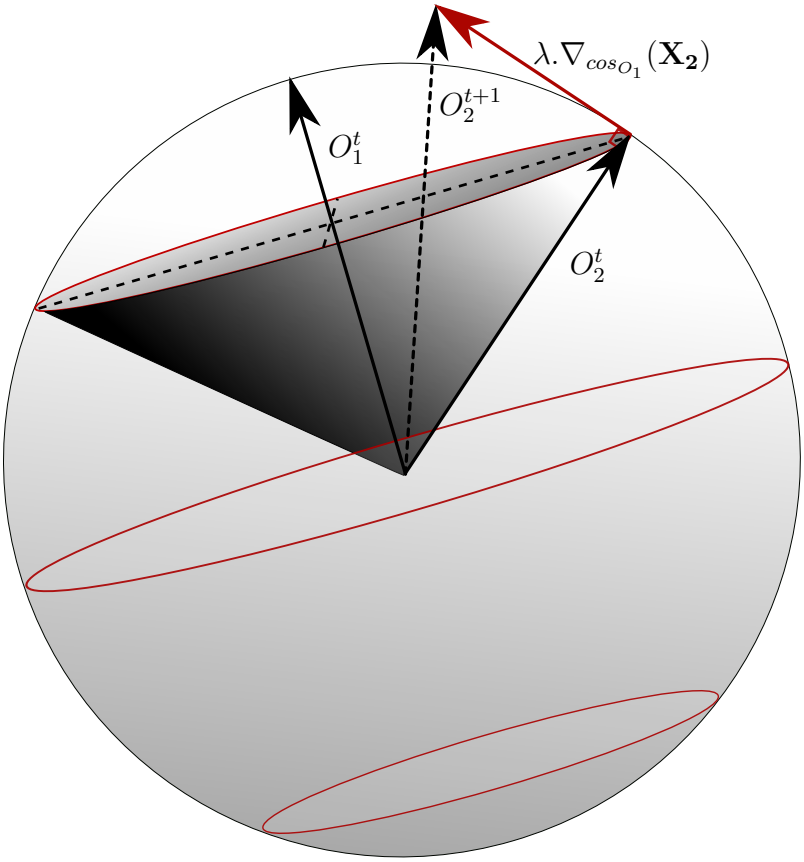


FIGURE 3 – Représentation 3D de l'effet d'une étape de mise à jour sur la norme des sorties pour une paire $(\mathbf{O_{X_1}}, \mathbf{O_{X_2}})$ à l'instant t. Les échantillons sont représentés avec la même norme dans un souci de simplification. Le cône gris correspond à la surface équipotentielle pour la fonction

L'introduction de la racine n-ième permet de rendre la valeur de cette métrique indépendante du nombre de vecteurs utilisés dans la matrice $\mathbf{A}$, tout en permettant une normalisation parfaite de l'erreur, égale à (voir Annexe A.14) :

$$\frac{\partial\left(psin(\mathbf{A})\right)}{\partial a_{ij}} = \frac{psin(\mathbf{A})}{n}.\frac{\varphi'(I_{ij})}{\|\mathbf{A_j}\|}.\left[\mathbf{A_{norm}}\mathbf{S^{-1}} - \mathbf{A_{norm}}\right]_{ij} \tag{11}$$

En effet, la matrice inverse présente dans le calcul de l'erreur est normalisée ainsi :

$$\begin{aligned}\det\left(\sqrt[n]{\det\left(\mathbf{S}\right)}\mathbf{S^{-1}}\right) &= \left(\sqrt[n]{\det\left(\mathbf{S}\right)}\right)^{n}\det\left(\mathbf{S^{-1}}\right)\\ &= \det\left(\mathbf{S}\right)\det\left(\mathbf{S^{-1}}\right)\\ &= \det\left(\mathbf{S}\mathbf{S^{-1}}\right)\\ &= 1.\end{aligned} \tag{12}$$

Il suffit alors de définir $\mathbf{A}$ comme la matrice dont les colonnes sont égales au vecteur caractéristique de l' échantillon de référence et des échantillons négatifs afin d'obtenir une nouvelle définition du problème angulaire, avec une parfaite symétrie pour un ensemble d'apprentissage T_k entre l'erreur relative à la similarité $Esim_W(T_k)$, et celle relative aux dissimilarités, $E\overline{sim}_W(T_k)$:

$$\begin{aligned}&E_W(T_k) = Esim_W(T_k) + E\overline{sim}_W(T_k)\\ &avec\\ &\begin{cases}Esim_W(T_k) = (1-\cos(\mathbf{O_{R_k}},\mathbf{O_{P_k}}))^2\\ E\overline{sim}_W(T_k) = (1-psin(\mathbf{O_{R_k}},\mathbf{O_{N_1}},\ldots,\mathbf{O_{N_K}}))^2\end{cases}\end{aligned} \tag{13}$$

Par ailleurs, il est intéressant de noter que cette nouvelle métrique permet de réaliser une analyse en composantes indépendantes non linéaire supervisée à l'aide du SNN.

Hypothèses émises sur nos contributions

Afin d'évaluer nos contributions, cinq hypothèses sont émises :

1. l'hypothèse H_1, concernant les **cibles pour paires dissimilaires**, suggère qu'un critère d'orthogonalité ($t_{cos} = 0$) pour les paires négatives est plus stable qu'un autre critère également communément utilisé de répulsion maximale ($t_{cos} = -1$) ;
2. l'hypothèse H_2, portant sur les **n-uplets**, veut que ces derniers permettent une meilleure représentation des relations entre classes, aboutissant à une projection plus discriminante ;
3. l'hypothèse H_3 propose que notre procédé de **régularisation de la norme** permet un meilleur *contrôle de l'évolution* de la norme des sorties ;
4. l'hypothèse H_4 considère que notre **métrique sinus polaire** est plus *apte à séparer les classes* que d'autres fonctions basées sur la similarité cosinus ;
5. l'hypothèse H_5 avance le plus fort **potentiel de rejet** du SNN comparé à d'autres méthodes de l'état de l'art à l'aide de son *apprentissage discriminant*.

0.4 Expériences et résultats

Nos expériences sont réalisées sur deux bases de données : la base Multimodale d'Activités Humaines (MHAD), nous permettra d'évaluer nos contributions théoriques

sur le SNN, en cherchant à valider en particulier les hypothèses H_1 à H_4 ; tandis que la base Orange, collectée sur place, nous permet d'établir une comparaison de nos contributions avec l'état de l'art, et d'étudier la validité de l'hypothèse H_5 sur le potentiel de rejet de notre SNN.

0.4.1 Expériences sur la base MHAD

La base MHAD de Berkeley [OCK+13] est constituée de 11 actions (*jumping, jumping jacks, bending, punching, waving 1/2 hands, clapping, throwing, sit down/stand up, sit down, stand up*), réalisées par un ensemble de 12 participants, à raison de 5 répétitions par utilisateur et par action. Bien que plusieurs capteurs ont été déployés lors de la collecte des données, combiant des caméras, appareils Kinect, microphones et *motion capture*, notre étude se base sur les données inertielles de 6 accéléromètres attachés aux extrémités des membres et aux hanches.

Après des tests préliminaires, l'accéléromètre attaché au poignet droit des utilisateurs est plus particulièrement sélectionné car les données issues de ce capteur montrent les résultats les plus discriminants. Les données inertielles sont soumises à un filtrage passe-bas, un seuillage et un rééchantillonnage afin d'obtenir des données de dimension fixe avec une dimension temporelle égale à 45.

L'architecture du SNN correspond à un MLP à trois couches, avec une couche d'entrée égale à 135, une couche cachée de 45 neurones, et une couche de sortie de 90 neurones. Une activation sigmoïde est choisie pour tous les neurones du réseau.

Les tests de classification suivent un protocole *leave-one-out*, avec les échantillons de 11 participants utilisés en apprentissage, et le dernier utilisé en test, afin de pouvoir également évaluer les capacités de généralisation de chaque modèle. Plusieurs configurations sont étudiées, avec une comparaison des deux objectifs pour les paires négatives (H_1) ; une comparaison entre les trois stratégies de sélection d'ensembles d'apprentissage (paires, triplets, n-uplets) ; et une comparaison entre trois fonctions objectives, notées *cos* (cf. Equation 6), *scal* (cf. Equation 8), et *psine* (cf. Equation 13).

Les résultats sont regroupés dans le Tableau 4.1. Il est intéressant de noter tout d'abord qu'une cible égale à 0 pour les paires négatives produit les meilleurs résultats pour les deux fonctions objectif *cos* et *psine*, et pour la fonction *scal* associée à la stratégie n-uplets, ce qui valide notre hypothèse H_1. La constatation inverse peut être faite pour la fonction *scal* associée aux stratégies paires et triplets, ce qui peut s'expliquer par la subdivision de la correction angulaire en trois sous-objectifs. Ainsi, une cible non-atteignable permet d'augmenter l'amplitude relative de l'erreur destinée à la correction du produit scalaire, par rapport aux erreurs dues à la régularisation des normes. Par ailleurs, les meilleures configurations sont atteintes avec la stratégie n-uplets pour toutes les fonctions objectif, avec le meilleur score obtenu par l'association de notre métrique sinus polaire (*psine*) avec notre stratégie n-uplets, validant ainsi nos hypothèses H_2 et H_4.

L'hypothèse H_3 est évaluée en suivant l'évolution des normes moyennes des sorties du réseau au cours de l'apprentissage (voir Figures 4.2, 4.3, 4.4). De manière surprenante, la non-linéarité apportée par le réseau inverse la tendance théorique d'évolution des normes, qui diminuent. Cette diminution peut néanmoins poser un problème pour les longs apprentissages, avec des normes qui tendent vers zéro. Ainsi, les versions stables des fonctions *cos* et *psine*, résultant d'un objectif d'orthogonalité pour les paires négatives, associées aux différentes stratégies de sélection des ensembles d'apprentissage, voient leur amplitude moyenne des vecteurs de sortie diminuer très rapidement

durant les premières époques. Cette diminution ralentit au cours de l'apprentissage, bien qu'elle ne s'arrête pas complètement. A l'inverse, la fonction *scal* avec régularisation des normes montre une correction de cette évolution, avec une amplitude moyenne stabilisée dans les cas stables, ce qui valide notre hypothèse H_3.

Après avoir validé nos quatre premières hypothèses sur nos contributions théoriques, la dernière hypothèse sur le côté applicatif du SNN pour la détection et le rejet d'erreurs de classification et d'éléments nouveaux est étudiée dans la partie suivante, à l'aide de la base Orange.

0.4.2 Expériences sur la base Orange

La base Orange est constituée de deux jeux de données inertielles provenant d'un Smartphone. Un seul de ces deux jeux sera présenté dans ce résumé. Ainsi, 40 échantillons sont prélevés pour un unique utilisateur sur un ensemble de 18 gestes (voir Fig. 4.8). Ces 18 classes sont séparées en deux groupes : seules 14 classes sont apprises par notre modèle Siamois, tandis que les 4 classes restantes sont utilisées durant la phase de test en tant que données inconnues. Ainsi, cette base couvre plusieurs types de gestes, avec de simples translations dans les plans horizontal (*Flick North*,*Flick South*,*Flick East*,*Flick West*) et vertical (*Up*, *Tap*) ; et des gestes symboliques tels que des lettres (*Z*), *N*, *Alpha*) ; des formes (*Clockwise*, *Counter Clockwise*, *Heart*) ; ou des actions (*Pick*, *Throw*). Les classes de gestes écartées durant l'apprentissage sont sélectionnées dans le but de présenter des similarités avec celles qui sont connues du modèle : le *8* est très proche des deux gestes circulaires ; *Infinity* se rapproche également de ces gestes circulaires ainsi que du geste *Alpha* ; et les gestes *V* et *W* sont similaires à plusieurs translations, tout en présentant la même dynamique en trois parties que les lettres *Z* et *N*.

Ces données, regroupant les relevés accélérométriques et gyrométriques, sont soumises aux mêmes prétraitements que celles de la base MHAD. L'architecture choisie pour le SNN est comparable à celle utilisée sur la base MHAD, avec une couche d'entrée de dimension égale à 270, afin de gérer les 45 relevés pour les deux capteurs 3D pour chaque échantillon.

Comme expliqué plus haut, cette base nous permet de tester l'hypothèse H_5 selon laquelle l'utilisation du SNN implique une meilleure identification et un rejet de meilleur qualité des éléments inconnus. Ainsi, 5 échantillons sont sélectionnés pour les 14 classes de gestes présentées en apprentissage, tandis que 16 autres échantillons sont sélectionnés pour ces mêmes classes en test, complétés par les 40 échantillons disponibles des classes inconnues. Ce test représente donc un paradigme de personnalisation intégré dans une interface naturelle où l'utilisateur ne spécifie pas à quel moment il réalise un geste, et c'est alors au système qu'incombe la tâche de déterminer si une action doit être déclenchée, avant même de choisir laquelle. On dispose ainsi de 70 échantillons de gestes en apprentissage ; et de 224 échantillons connus et 160 échantillons inconnus en test, pour un total de 384 échantillons de test, dont 41.6% sont inconnus. Chaque test est répété 10 fois, afin de suivre les scores moyens et leur écart-type. D'après les résultats obtenus pour les hypothèses H_1 à H_4 sur la base MHAD, les choix suivant sont faits pour les SNNs étudiés : une cible d'orthogonalité est choisie pour les paires négatives, avec une stratégie de sélection des ensembles d'apprentissage basée sur les n-uplets. Trois différentes configurations, nommées respectivement "SNN-cos", "SNN-scal" et "SNN-psine" afin de reprendre les notations de la section précédente, sont comparées, et diffèrent donc uniquement par leur fonction objectif. Une comparaison

est également faite avec deux méthodes de l'état-de-l'art, sélectionnées pour leur similarité à l'approche SNN. Ainsi, les différentes configurations du SNN sont également comparées à une méthode basée sur un DTW associé à un K-NN en tant que mesure de similarité non linéaire ; ainsi qu'au classifieur MLP, en tant que méthode neuronale voisine.

Deux critères de comparaison sont suivis. Le premier consiste à étudier l'évolution du taux de classification en fonction du taux de rejet lorsque le seuil de rejet varie. Le taux de classification est défini comme le rapport entre le nombre d'échantillons acceptés et correctement classifiés, et le nombre total d'échantillons acceptés ; tandis que le taux de rejet correspond au rapport entre le nombre d'éléments rejetés et le nombre d'éléments total. La performance d'un modèle peut alors être évaluée à l'aide de l'aire sous sa courbe. Par ailleurs, on propose également de suivre les ratios des différents types de rejet en fonction du taux de rejet. Ainsi, trois différents types sont identifiés : le faux rejet correspond aux éléments correctement classifiés qui ont été rejetés ; le rejet des erreurs de classification est formé des éléments mal classifiés, qui ont cependant été correctement rejetés ; et le rejet des éléments inconnus concerne l'identification des échantillons provenant des classes non présentées en apprentissage.

Les résultats de ces tests sont présentés dans les Figures 4.10, avec un agrandissement de la zone importante dans la Figure 4.11. Il est important de noter qu'aucun modèle ne peut présenter un taux de classification égal à 100% tant que le taux de rejet n'est pas suffisant pour écarter les 41.6% de données inconnues, d'où la présence de la zone bleu ciel inatteignable, et de la courbe dorée qui représente une performance parfaite. On peut immédiatement noter la supériorité des trois configurations SNN, qui présentent des performances autour de 95% au seuil critique de 41.6% de rejet, performances supérieures de 2% au DTW et de 4% au MLP. Le DTW tire parti de sa mesure de similarité non-linéaire, à l'inverse du MLP, qui n'est pas correctement équipé pour le rejet d'éléments inconnus en tant que simple classifieur. Parmi les trois configurations du SNN, le SNN-scal montre des résultats supérieurs aux deux autres modèles SNN-cos et SNN-psine. Ce phénomène peut être expliqué par le fait que la subdivision du cosinus en trois objectifs et la régularisation sur les normes permet de limiter un phénomène de surapprentissage, qui est un problème particulièrement sensible dans ce cas où les tests sont réalisés sur un unique utilisateur. Une analyse plus détaillée du rejet peut être réalisée à l'aide de la Figure 4.12. On peut remarquer que les méthodes SNN présentent la plus faible surface pour le taux de faux rejet moyen, avec un rejet dont la qualité ne commence à se dégrader qu'à partir du seuil critique des 41.6%. Cela met en évidence le plus fort potentiel de sélections des méthodes SNN comparées aux deux autres, où la dégradation est plus progressive avec l'augmentation du taux de rejet. Le SNN-scal montre un taux de faux rejet nul pour les faibles taux de rejet, tandis que les SNN-psine et SNN-cos rejettent toujours des échantillons correctement classifiés, autre indice du phénomène de surapprentissage.

En conclusion, nous avons montré la validité de l'approche SNN dans ses différentes configurations pour la classification et le rejet grâce aux bases de données MHAD et Orange. Nous avons pu confirmer et quantifier l'impact de nos contributions. Ainsi, notre stratégie de sélection adaptée aux problèmes de classification permet d'améliorer l'apprentissage, tandis que l'objectif basé sur le produit scalaire avec régularisation des normes permet de contrôler l'évolution des variations des normes des vecteurs caractéristiques, au prix d'une plus grande difficulté de correction des angles pour des bases de données plus complexes ; et la métrique sinus polaire se révèle être une généralisation efficace de la fonction sinus afin de mesurer les dissimilarités entre un

nombre arbitraire de vecteurs, au coût d'une complexité plus élevée. Enfin, le potentiel du SNN a été évalué sur une base de données réaliste de relevés de capteurs MEMS sur Smartphone, montrant des résultats concluants en identification et rejet d'erreurs et éléments inconnus, comparés à ses deux méthodes voisines basées sur le DTW et sur le MLP.

0.5 Conclusion et perspectives

Ainsi, dans cette thèse, nous avons développé une solution de reconnaissance de gestes symboliques inertiels dans le but de proposer des interfaces plus riches, dans le cadre d'un monde ouvert où l'on cherche à gérer de multiples utilisateurs et la possibilité de rencontrer des gestes inconnus. Pour cela, nous avons travaillé sur l'apprentissage automatique de métrique non linéaire à l'aide du paradigme Siamois. Nous avons donc proposé :

1. une adaptation du SNN aux problèmes de classification avec une généralisation de la stratégie de sélection des ensembles d'apprentissage ;
2. une relaxation de la similarité cosinus afin de mieux contrôler l'évolution de la projection ;
3. une redéfinition du problème angulaire avec une nouvelle métrique de similarité, la métrique Sinus Polaire, qui peut être interprétée comme un apprentissage stochastique d'une analyse en composantes indépendantes non linéaire ;
4. l'utilisation du SNN pour la détection de nouveauté et le rejet d'erreurs et éléments inconnus, avec des performances compétitives par rapport à des méthodes voisines telles que celles basées sur le DTW et MLP.

Plusieurs perspectives sont envisagées, notamment pour le réseau Siamois lui-même, puis pour son application à la reconnaissance de gestes. La principale évolution du réseau Siamois résulterait d'une exploration d'autres architectures de réseaux, notamment avec les réseaux convolutionnels, qui ont déjà montré leur adéquation pour la reconnaissance de gestes [DBLG14], et plus généralement en cherchant à introduire la dimension temporelle, avec par exemple des réseaux récurrents tels que le BLSTM [LBMG15]. Par ailleurs, notre étude de la problématique de la reconnaissance de gestes tirerait un grand bénéfice d'une augmentation du volume des bases d'apprentissage et de test afin de pouvoir proposer une application grand public. D'autres stratégies peuvent être explorées : en apprentissage, il serait intéressant de travailler sur un apprentissage hiérarchique sous forme d'arbre de décisions, avec un classifieur spécialisé qui permettrait de résoudre les confusions plus subtiles entre des classes similaires qui sont confondues ; en test, d'autres stratégies de rejet pourraient être expérimentées, avec par exemple la possibilité de définir un seuil de rejet spécifique à chaque classe. Enfin, une étude peut être réalisée sur la projection des éléments nouveaux par le réseau, avec une éventuelle possibilité d'identifier et de labelliser ces éléments sans nécessité de réapprentissage, afin d'aboutir à une création dynamique de nouvelles classes dans un processus de personnalisation.

Contents

Acronyms

AP Affinity Propagation. 31

BPTT BackPropagation Through Time. 44
BW Baum Welch. 27

CEC Constant Error Carousel. xxviii, 43, 44
cHMM Continuous Hidden Markov Model. 25, 27, 29, 44, 78, 81
CNN Convolutional Neural Network. x, xi, 41–45, 47, 48, 53, 62, 78, 81–83, 93
CPCA Common Principal Component Analysis. 33
CPC Common Principal Component. 33
CTW Canonical Time Warping. 31, 32
HHMM Hierarchical Hidden Markov Model. 29

DCT Discrete Fourier Transform. 20, 35
DDTW Derivative Dynamic Time Warping. 30
dHMM discrete Hidden Markov Model. 25, 27
DTW Dynamic Time Warping. xix, xx, xxviii, 22, 29–32, 41, 44, 75, 76, 78, 79, 81, 83, 84, 89, 92

FDSVM Frame-Based Support Vector Machine. 44, 78
FFT Fast Fourier Transform. 20, 35

GTW Generalized Time Warping. 31, 32

HMM Hidden Markov Model. xxviii, 22, 23, 25–29, 34, 41

K-NN K-Nearest-Neighbour. xiii, xix, 6, 29, 34, 55, 62, 68, 75, 79, 94, 95

BLSTM Bi-directional Long Short-Term Memory Recurrent Neural Network. x, xx, 44, 45, 78, 81–83
LDA Linear Discriminant Analysis. x, 7, 22, 29, 33, 34
LSTM Long Short-Term Memory Recurrent Neural Network. xxviii, 43, 44, 78, 93

MEMS MicroElectroMechanical Systems. xxi, xxvii, 5, 11–13, 15

MFCC Mel Frequency Cepstral Coefficients. 53

MHAD Berkeley Multimodal Human Action Database. xvi–xix, xxi, xxxi, 65, 66, 68, 75, 76, 89

MLP MultiLayer Perceptron. x, xi, xiii, xvii, xix, xx, 8, 33, 37, 38, 40, 41, 45, 48, 50, 55, 56, 62, 75, 76, 78, 79, 84, 89, 92–94

NB Naive Bayes. 23

PCA Principal Component Analysis. x, 7, 22, 29, 32–34, 78, 81, 83

PC Principal Component. 33

PDTW Piecewise Dynamic Time Warping. 30

RBM Regularized Boltzmann Machines. 49

RNN Recurrent Neural Network. 43

SFNN Single layer Feed-Forward Neural Network. 48

SNN Siamese Neural Network. viii, x, xi, xiii, xiv, xvi–xx, xxiii, xxvii–xxix, 8, 47, 61–63, 65, 67–82, 84, 86, 89, 91–95

SOM Self-Organizing Maps. 48

SVM Support Vector Machine. x, 7, 23, 34, 75, 76, 78

VCR Video Cassette Recording. 23

WPD Wavelet Packet Decomposition. 35

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Human-Machine Interfaces started with expert modalities, such as punch cards, cables, or text-based environments with a single keyboard, where instructions were transmitted to the machine through specific commands only accessible to professionals and experts. Emerged then the Graphical User Interfaces along with personal computing, where users would manipulate more instinctive objects, opening files and documents in hierarchical structures called directories, copying and pasting; with the help of new input techniques such as the mouse. While this new paradigm allowed for a simpler interaction and a wider public with functions hidden behind known patterns, some basics concepts have to be learned to understand the interaction tools. Moreover, these actions are still very dependent of the visual modality through the screen.

For this reason, interactions still evolve today with computing capabilities and new hardware. Multiple fields such as cognitive science and design are integrated to provide users with new interfaces, mimicking the already established cognitive models in a "Natural Interface", which allows for a direct manipulation of the device functions.

In this chapter, we first provide an overview of Gesture Recognition and its applications and limits as a tool for these new concepts. In a second part, we define the frame and contributions of this study, called "Automatic non-linear metric learning". Applied to 3D-sensors Gesture Recognition, its main goal is learning a projection and feature space characteristic of this type of signals from similarity information.

1.1 Gesture Recognition

While linguistics were the first means of Human-Machine Interactions until the 1970s, through the first programming languages closer to natural language such as FORTRAN, a new interest was then oriented towards a spoken communication with machines, developing the field of speech recognition, analysis and synthesis. However, gestures are another instinctive way of conveying messages, information or emotions. In 1980, Richard A. Bolt explores for the first time a multi-modal interface with "Put That There", consisting in "a physical facility where the user's terminal is a room into which one steps". The user, sat in a chair, combines speech commands to create or move objects, with spatial and selection information through "gestures", defined as an extension of the arm in the direction of the screen. The gestures are interpreted using a magnetic-based space position and orientation sensing device attached to the wrist of the user (see Fig. 1.1).

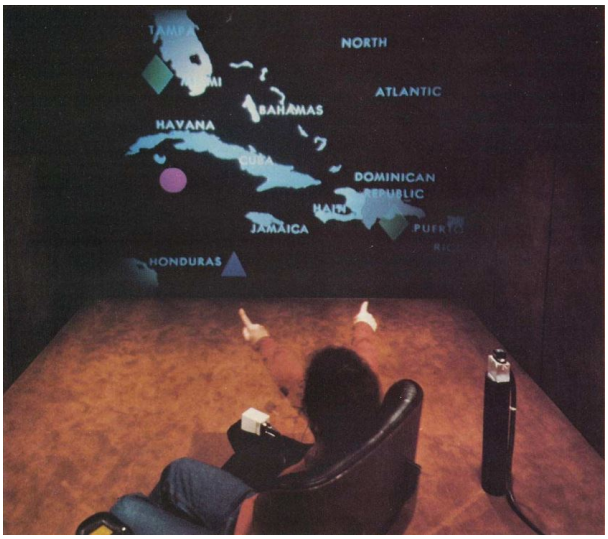


Figure 1.1 – User talking and pointing to items on the screen with the Put-That-There interface. Extracted from [Bol80].

In this context, Cadoz [Cad94] studies how the gesture modality can be the richest interaction medium. Indeed, gestures present a duality in a sense that, contrary to vision or hearing, they allow for practical interactions with the environment as well as sensing through touch and pressure. Thus, three different functional roles of the gestural channel are identified: the epistemic role, contributing to the discovery of the environment through temperature, pressure and the tactile sense; the ergotic role, with material actions generating modifications and transformations of that environment; and the semiotic role, communicating meaningful information towards that environment.

The term "gesture" covers a wide variety of movements, with a focus drawn on face, through expressions and emotions; and arms. In order to define this notion, McNeill [McN92] proposes a survey based on *Kendon's continuum*, from the work of Adam Kendon, to clarify the distinctions between different actions which might be called "gestures". Five main categories are identified: gesticulations, speech-framed gestures, emblems, pantomime and signs. *Gesticulations* are the most common and natural gestures: they correspond to movements with a specific meaning accompanying speech, describing for example the shape of an object. *Speech-framed gestures* differ from gesticulations in the sense that they hold a specific, independent role in the sentence negating the need for an additional spoken component, for instance, the physical representation of an action. *Emblems* are culturally specific gestures which hold their own meaning, from head shakes or hand movements for signifying approval, to "rude" gestures. A *pantomime* is closely related to the speech-framed gestures as it conveys a narrative line through a gesture or sequence of gestures, but does so without any need of speech. Finally, the *signs* replace words in the context of a "sign language".

As a means of communication, the relation between speech and gestures was more closely analysed to describe the different types of gestures encountered in a multi-modal speech and gesture frame, for a semiotic purpose. Hence, the identified types mainly fall within the first two categories, which are gesticulations and speech-framed gestures. Four types are described: iconic, metaphoric, deictic, and beats gestures. *Iconic* gestures complete the mental image of the spoken subject with a concrete illustration of that subject, while *metaphoric* gestures do not limit to a concrete action or object, to represent abstract concepts and expressions. *Deictic* gestures are the ones involved in

the Put-That-There interface, with pointing movements which can designate physical objects as well as abstract notions in time and space. *Beats* are small gestures used by the speaker in order to emphasise certain words or sentences. It is important to note that these categories are not mutually exclusive, and are used to characterize a gesture rather than classifying it.

To conclude, gestures are a main channel in our everyday communications, supporting the current speech as well as communicating concrete and abstract ideas by themselves. However, they are the result of language development and are not trivially transposed for a "natural" interface. Two roles are identified: while manipulation allows for a direct transformation of virtual objects, developing the ergotic role of the gestural channel, control consists in executing a specific command triggered by the corresponding gesture, in a predefined vocabulary, playing the semiotic role. Here, gesture recognition is defined as the process transmitting information about the gestures performed by a user to a device.

In the following, we present the thesis context for gesture recognition, the practical applications of a framework interpreting gesture data and the known issues associated with such an application.

1.1.1 Context

Nowadays, the urge for more intuitive interactions with machines grows as computers are always more present in our everyday lives. Personal computing has reached a peak, and devices become smaller and smaller.

Ubiquitous computing was foreseen by Mark Weiser [Wei91] as "a way of thinking about computers in the world [which] takes into account the natural human environment and allows the computers themselves to vanish in the background". It is gradually becoming a reality through the emergence of the Internet of Things (IoT), where everyday objects may provide enhanced services thanks to their connectivity, communicating between them and added sensors. Thus, the ultimate goal behind ubiquitous computing is to completely detach the user from the device.

Ambient Intelligence is a concept closely related to ubiquitous computing, and consists in a technological evolution allowing the development of omnipresent, connected and communicating small sensors, which transmit data without any user intervention. Its aim is to "enrich specific places (room, building, car, street) with computing facilities which can react to peoples' needs and provide assistance" [AM07].

This concept is visible in its early stages today through "smart" accessories or clothing, more generally called "wearable devices". The most famous example of a device which has become common is the Smartphone. Bringing the personal computer in our pocket, these devices deliver much more features thanks to numerous applications. Indeed, everyday objects such as the agenda, calculator or calendar are replaced, while an Internet connection allows for the use of a mail client and data synchronization between devices among other services. However, the integration of new sensors is the main reason of the development of the Smartphone potential, with, for instance, geolocalisation, thanks to the GPS, and orientation, with the magnetometer.

As such, Ambient Intelligence also implies a proactive dimension, with context awareness. Brooks [Bro03] identifies the five basic narrative questions which should serve as guidelines: Who, What, When, Where and Why. While the processes of spatial and temporal localisation are well-mastered, the main issues reside in the "Who", "What", and "When" questions. "Why" is the most challenging question, requiring

a deep understanding of relationships between events or actions, and the meaning they hold for the user. However, "Who" and "What" are currently under great study thanks to Pattern Analysis techniques, with for instance user identification and activity recognition. Hence, Ambient Intelligence is at a crossroads of multiple fields, linking sensors, Machine Learning and Human-Computer Interactions (see Fig.1.2).

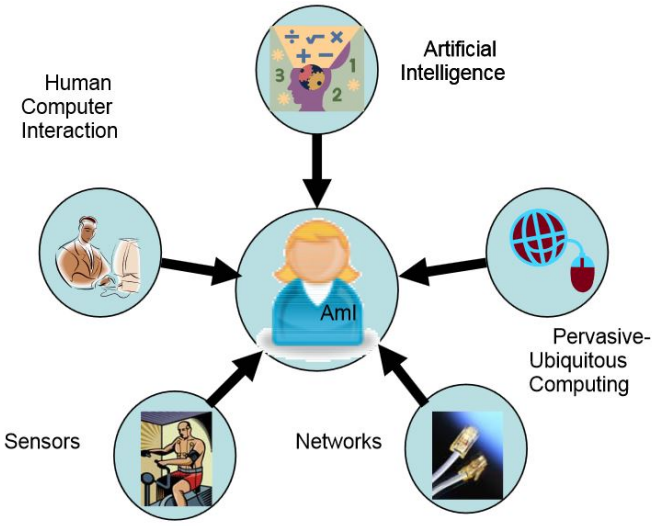


Figure 1.2 – Relation between Ambiant Intelligence (AmI) and other areas in Computing Science. Extracted from [AM07].

In that context, as computing and everyday objects merge and become always more popularized, it is necessary to provide new interaction experiences to users, in order to ease the manipulation of a technology evermore present, for experts and beginners alike. As a natural and spontaneous communication channel, the gesture is the right candidate for a new communication channel with the machine.

1.1.2 Applications

Two kinds of gestures can be considered for different applications. On the one hand, static gestures correspond to a specific state, described by a unique set of features, with, in the context of Smartphones, a "phone-to-ear" posture for instance. On the other hand, dynamic gestures are more complex, since they are described by a time-series of inertial signals, such as the "picking-up" movement when the user is ready to take a call. While their path information allows for tasks such as pointing, they can also hold a meaning through a symbolic association, whether representing a shape or an action. Thus, in the rest of this thesis, the term *gesture* targets in particular the symbolic, dynamic gestures, whose path in space is associated to a meaning, such as letters, geometric shapes or activities.

Interaction evolutions were always paired with available technology. 2D gesture recognition was developed first. Based on Optical Character Recognition techniques, pen strokes or finger movements on a sensing surface are tracked to compute robust features such as form, speed and acceleration, with very little noise, enabling writing or symbolic drawing recognition. Then, 3D gestures were analysed. The first systems were heavily reliant on vision-based systems, with body or hand tracking, body suits, or instrumented gloves. However, these systems are not user-friendly in the sense that they require a cumbersome equipment and heavy calibrations. Vision-based gesture

recognition picked up considerably with the apparition of the Kinect device, which was the first affordable 3D motion sensing device available to individual customers.

In recent years, new sensors called MicroElectroMechanical Systems (MEMS) were popularized thanks to their small sizes and low production costs. Nowadays, MEMS-based inertial sensors like the accelerometer and gyrometer can provide spatial awareness to any device, effectively increasing the accessible space dimensionality. These sensors may be embbeded in a device such as a Smartphone, allowing for the analysis of gestures realized by users in the 3D space, with the phone in hand; or directly worn by the user, for body and activity tracking.

Other applications for gesture recognition can be emphasised, with for instance remote control of everyday and household devices such as the television, interactions with virtual objects or interfaces in an Augmented Reality context, avatar control or object manipulation for gaming, or even authentication with spatial signatures.

Thus, these new sensors are the basis for a new user experience, with a new freedom, in particular to those with a visual or physical impairment, thanks to a detachment from the typical, omnipresent visual interface which requires a coordination between the eye and the hand.

In the next section, we will present the main issues impeding 3D-sensor-based gesture recognition for these domains of application.

1.1.3 Known issues

Inertial gesture recognition is subject to multiple drawbacks which limit its impact. Indeed, two approaches can be identified. The user-dependent approach only targets one user, while the user-independent one should not limit the number of users. As an interface, a gesture-based framework falls into the second category. It is necessary to design an open-world system, where the model used captures the diversity of the different productions of the same gestures, and provide a sufficient generalisation.

It is essential to mention that differences between users are amplified by the inertial sensors. Indeed, while a human being would mainly associate the shape of a gesture to its meaning, with small possible variations for speed, inertial sensors cannot provide the equivalent of this mental shape, as only accelerations and angular velocities are available. Thus, inherent differences in gesture expressiveness, with varying speeds and amplitudes, are directly impacting the recognition step, which has to process signals with variable amplitudes and durations.

Moreover, in an open-world application, a personalisation paradigm has to be considered, allowing users to define their own set of gestures. So, the inertial limits must also be taken into account during this vocabulary definition. To what extent different gestures for human beings are different for the machine, sensor-wise? Is it possible to provide generalisation for an undefined number of users with arbitrary preferences and styles? Furthermore, within a realistic framework, any personalisation process could not ask for a comprehensive training set, as it would require a deterring effort from the user to perform each gesture to be learned many times.

This necessary instinctive and seamless experience introduces new problems. During a natural manipulation of the device where the user does not directly signal its intention to activate the system, gestures such as involuntary shakes or non-machine-informational gestures destined to other targets must be filtered by the system thanks to a rejection strategy.

Thus, our study focuses on this aspect to devise a framework which would include classification as well as rejection processes, in particular thanks to a training based on an automatic, non-linear metric learning through similarity considerations. However, we do not tackle here the problem of temporal segmentation, which is already performed for the testing process. In the next section, we define the notion of metric, to develop the thesis subject and contributions.

1.2 Automatic non linear metric learning

Given a set of features representing a phenomenon, any analysis or classification task is greatly simplified when a specific metric is available to gauge its similarity to a known example, for instance through a K-NN classifier [CH67]. However, this metric, supposed to output respectively low and high values to similar and dissimilar events, may not be directly accessible because of unknown dependencies or non-linearities. To that end, metric learning exploits known pairwise constraints in order to devise a more coherent comparison metric.

Proposition 1 *A metric d in a space $\mathcal{X}$ is defined by four properties.* $\forall\ \mathbf{X_1}, \mathbf{X_2}, \mathbf{X_3} \in \mathcal{X}$, *the following conditions are satisfied:*

1. *non-negativity :* $d(\mathbf{X_1}, \mathbf{X_2}) \geq 0$,
2. *the coincidence axiom :* $d(\mathbf{X_1}, \mathbf{X_2}) = 0 \Rightarrow \mathbf{X_1} = \mathbf{X_2}$,
3. *symmetry :* $d(\mathbf{X_1}, \mathbf{X_2}) = d(\mathbf{X_2}, \mathbf{X_1})$,
4. *triangle inequality* $d(\mathbf{X_1}, \mathbf{X_3}) \leq d(\mathbf{X_1}, \mathbf{X_2}) + d(\mathbf{X_2}, \mathbf{X_3})$.

However, multiple axioms can be dropped in practice, leading to:
— pseudo-metrics, where the coincidence axiom is replaced by the identity axiom $d(\mathbf{X_1}, \mathbf{X_2}) = 0$;
— quasimetrics, dropping the symmetry axiom;
— semi-metrics, dropping the triangle inequality;
— premetrics, which only verify non-negativity and identity.

Metric learning techniques can be characterised depending on five criteria (see Fig.1.3), which are:
— the learning paradigm (fully, weakly or semi supervised);
— the form of the metric (linear, nonlinear, local);
— the scalability with relation to the number of examples or the dimension of the data;
— the optimality of the solution (local, global);
— and whether the metric performs a dimensionality reduction.

Three main approaches exist for global metric learning: Mahalanobis metric learning [Mah36]; linear similarity learning; and kernel methods which introduce a non-linearity.

The Mahalanobis distance between two vectors $\mathbf{X_1}$ and $\mathbf{X_2}$ is related to the Euclidean distance between both vectors in a projection space defined by a matrix $\mathbf{P}$. Thus, given the matrix $\mathbf{A} = \mathbf{P}^T\mathbf{P}$, the distance $d(\mathbf{X_1}, \mathbf{X_2})$ between these two vectors is defined as:

$$d(\mathbf{X_1}, \mathbf{X_2}) = \sqrt{(\mathbf{X_1} - \mathbf{X_2})^T \mathbf{A} (\mathbf{X_1} - \mathbf{X_2})}. \tag{1.1}$$

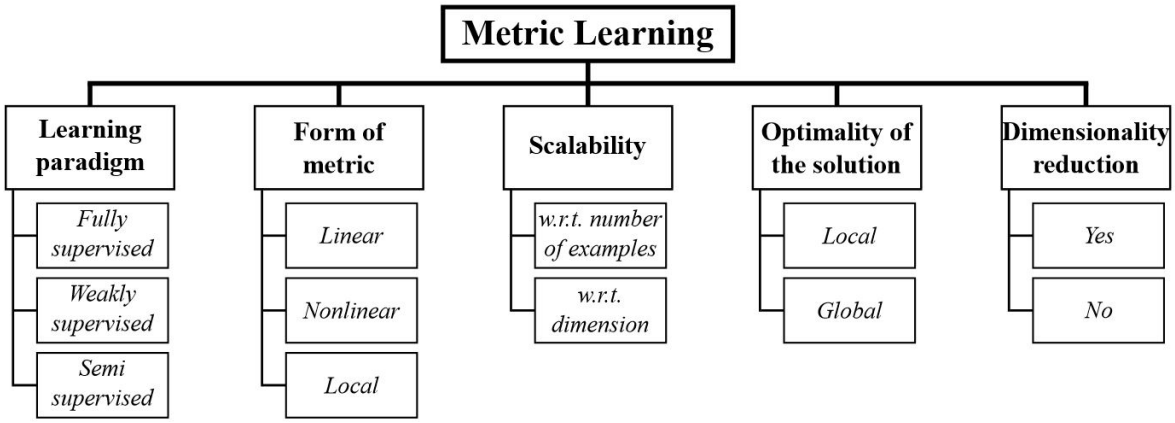


Figure 1.3 – Five key properties of metric learning algorithms. Extracted from [BHS13].

The matrix $\mathbf{A}$ is symmetric positive semi-definite, satisfying the conditions of a pseudo-metric. Eigenvector-based methods such as Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) can be seen as a Mahalanobis distance learning, as they result in a projection matrix, respectively maximising the variance in the subspace, and the ratio of the inter-class variance over the intra-class variance. Among the most popular Mahalanobis distance learning algorithms, we can cite the original Malahanobis Metric Learning (MMC) algorithm [XJRN02]; Neighbourhood Component Analysis, optimising the expected leave-one-out error of a stochastic nearest neighbour classifier in the projection space [GHRS04]; Large Margin Nearest Neighbours [WS09], locally minimising the distance with "target neighbours" and maximising the distance with "impostors" over the training set ; or Information-Theoretic Metric Learning [DKJ+07] which introduces a Bergman divergence with a LogDet divergence regularisation, generated by the convex function $\phi(\mathbf{A}) = -\log\det\mathbf{A}$ defined over the cone of positive-definite matrices.

Given a square matrix $\mathbf{M}$ and a normalization term $N(\mathbf{X_1}, \mathbf{X_2})$, linear similarity methods aim at learning distance functions $K_M(\mathbf{X_1}, \mathbf{X_2})$ of the form:

$$K_M(\mathbf{X_1}, \mathbf{X_2}) = \frac{{\mathbf{X_1}}^T \mathbf{M} \mathbf{X_2}}{N(\mathbf{X_1}, \mathbf{X_2})} \tag{1.2}$$

No assumption is made on the matrix $\mathbf{M}$, which can allow for more flexibility. The cosine distance $d_{cos}(\mathbf{X_1}, \mathbf{X_2}) = 1 - \cos(\mathbf{X_1}, \mathbf{X_2})$ can be seen as a particular case, where the matrix $\mathbf{M}$ is equal to the identity matrix, and the normalization term is the product of the norms of $\mathbf{X_1}$ and $\mathbf{X_2}$. One good representative of this approach is the Generalised Cosine Learning Algorithm [QGCL08], which aims at learning a cosine similarity in a projection space implied by a definite semi-positive projection matrix.

Finally, kernel methods consists in introducing non-linear forms of metrics. While this can consist in a kernelisation of linear methods, with Kernel Principal Component Analysis [SSM98], or with the Gradient-Boosted Large Margin Nearest Neighbour algorithm [KTS+12], non-linear methods can also take advantage of machine learning techniques such as Support Vector Machine (SVM), with the Support Vector Metric Learning algorithm, alternating between training an SVM model with respect to a Mahalanobis distance and learning a Mahalanobis distance minimising the validation error of the SVM model; or neural network, with for instance the Siamese strategy

[BGL+94][CHL05].

In the following, we first present the thesis subject, aiming at an Automatic non linear metric learning based on neural networks, before introducing our contributions to this field.

1.2.1 Thesis subject

This thesis focuses on non linear metric learning through neural networks applied to gesture recognition. Indeed, these models, known for their execution computation speed, are ideal in a context of embedded computing. This execution speed is however not the same during training, which may be time consuming. Thus, we focus on the Siamese Neural Network (SNN), which proposes a training strategy based on known similarity relationships between pairs of samples, allowing for the non-linearity of the neural network, as well as a possibility for dimension reduction through an adjustable output layer. The SNN is trained to reflect these relationships through a controlled metric, such as the Euclidean or cosine distances, applied to the output space. Hence, this model also provides representative feature vectors, on which the output metric is based.

While the SNN acts as a regular neural network during the recognition step to compute these new features, its training is different from other networks as it implies the use of identical networks in parallel in order to process and learn from the similarity information between pairs of samples.

As a consequence, the SNN-based metric learning differs from other classical neural networks seeing as the original class labels assigned to the data are not necessary anymore. Class labels are ultimately translated into similarity relationships, and are only used for the classification process, which intervenes subsequently with an independent classifier.

In the following, we expose our contributions to the Siamese Neural Network.

1.2.2 Contributions

In this thesis, we propose three contributions to the cosine metric, MultiLayer Perceptron(MLP)-based SNN.

Firstly, while most applications make use of a limited information during training, with relationships between pairs or triplets of samples, we first propose to integrate the simultaneous known relationships between sets of samples through a selection of tuples in order to produce a more structured output space. This approach looks particularly sensible with a view to operating a classification, as classes present an inherent, rich relationship information between sets of samples.

Secondly, the analysis for the cosine similarity applied to the output space shows many possibilities for numerical instabilities. In this sense, we suggest a modified similarity metric, where the norm of the outputs is controlled through a regularization applied to the outputs themselves. This also leads to a simplification of the metric in the new projection space, which only relies on the scalar product.

Furthermore, we investigate the analysis of relationships between sets of samples, introducing a polar sine-based dissimilarity metric, which can handle in a uniform, simultaneous way the dissimilarities between samples belonging to different classes. This new similarity metric is then proven to perform a Supervised Non-Linear Independent

Component Analysis, aiming at decorrelating the set of samples from every available class.

Finally, we explore the potential of Siamese networks for novelty detection and rejection, aiming at harnessing the high-level representation capacities for separating different gestures.

Thus, this thesis explores the potential of a neural network-based similarity metric learning applied to gesture recognition, with a view to enhancing user experience through the enrichment of new, more natural interaction modalities. This process relies on the Siamese Neural Network, trained thanks to relationship information between sets of samples, whose discrimination capacity is proved in a rejection context.

This thesis is organized as follows. After this introduction, Chapter 2 presents a review of gesture acquisition technology and equipment, followed by a literature of state-of-the-art methods for gesture recognition and rejection. Chapter 3 studies more closely the Siamese Neural Network through its previous occurrences, before presenting the theoretical background behind our contributions. Chapter 4 analyses these contributions thanks to classification and rejection tests over two 3D-sensor databases. Finally, conclusions are drawn in Chapter 5, and perspectives are presented.

Chapter 2

Classical Methods for Gesture Classification

The following chapter presents a global overview for gesture recognition in two parts. Firstly, the main sensors involved are studied, from the inertial sensors, like the MicroElectroMechanical Systems, to the more equipment demanding motion capture and skeleton tracking. Secondly, we survey the different approaches to gesture recognition, with statistical, classifier, and artificial-neural-network-based methods.

2.1 Gesture data acquisition

For the rest of this document, we use a notion of gesture which appeared with modern technology, meaning a specific motion associated to an idea or used to control a device. Thus, this section presents the multiple sensors and modalities involved in gesture acquisition, with inertial sensors which describe the relative movement of a single point; to the global body tracking thanks to vision-based methods such as motion capture.

2.1.1 Inertial system

New Smartphone applications draw their success mostly from the use of sensors, enablers of the "smart" trait of these new kind of phones. These sensors are known under the name "MicroElectroMechanical Systems" (MEMS). Their ease of use and cheap integrated production costs based on semi-conductors, printed onto circuit boards using photo-lithography, explain their wide spread in a lot of devices, such as cars for airbag deployment detection, or drones for navigation.

Three sensors are typically present in our everyday devices: the accelerometer, the gyrometer and the magnetometer. The latter is usually dismissed, since its use is not natural in a seamless interface. Indeed, preliminary tests proved that this sensor is less reliable, as it requires constant calibrations to compensate the variations and interferences induced by the local electromagnetic fields. That is why we focus on accelerometers and gyrometers in the following, whose combination improves classification results, as proved in [BL], and explain how these sensors operate so as to extract characteristic features describing the movements of the device. These sensors record respectively the accelerations and angular rates along the three axes, as represented in Fig.2.1.

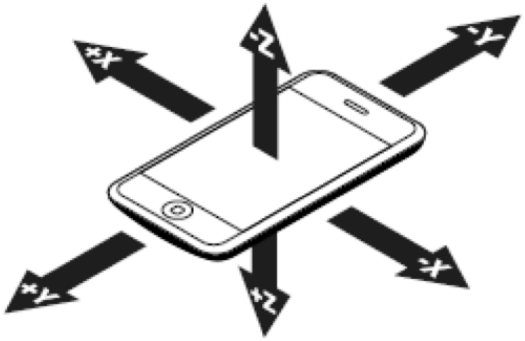


Figure 2.1 – Representation of the three axis of MEMS sensors, relative to the Smartphone device.

2.1.1.1 Accelerometers

A 3-axis accelerometer is actually constituted of three linear accelerometers (see Fig. 2.2), which record the acceleration of the system along a predetermined axis. Although many accelerometer types exist, from mechanical to piezoelectric, the most common accelerometer is based on capacitive effects.

Two imbricated, oppositely charged semi-conductors combs form the capacitive accelerometer. While one comb is mobile, the other is fixed to the device, which allows them to act as a mass-spring system where the variation of the total capacitance, and thus the current going through the sensor, is directly related to the acceleration of the system.

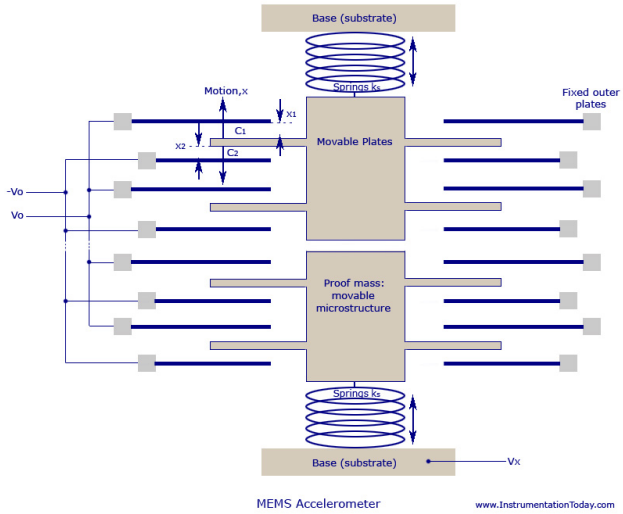


Figure 2.2 – Schematics for a one-dimensional MEMS accelerometer. Extracted from [ins].

Let x be the amplitude of the motion of the movable comb, and d the distance between the forks of the combs at rest. When respective voltage amplitudes V_0 and $-V_0$ are applied to the set of combs, the output voltage V_x measured is directly related to the displacement of the mobile comb. After a linearisation (cf. [And08] for details):

$$V_x = \frac{x}{d} V_0. \tag{2.1}$$

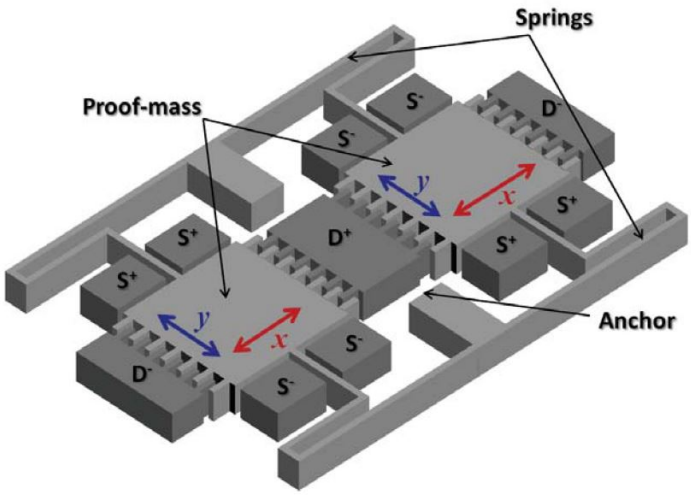


Figure 2.3 – Schematics for a one-axis vibrating structure MEMS gyrometer. Extracted from [GK03].

Given Hook's law for springs, k_s the spring constant of the system, and m the mass of the movable structure, the acceleration a is defined by:

$$a = \frac{k_s}{m}x = \frac{k_s d}{mV_0}V_x. \tag{2.2}$$

This formula is a simplification, which is only valid in precise circumstances that are not met in real use, as it would require a high quality factor, limiting the effective bandwidth of the oscillating system.

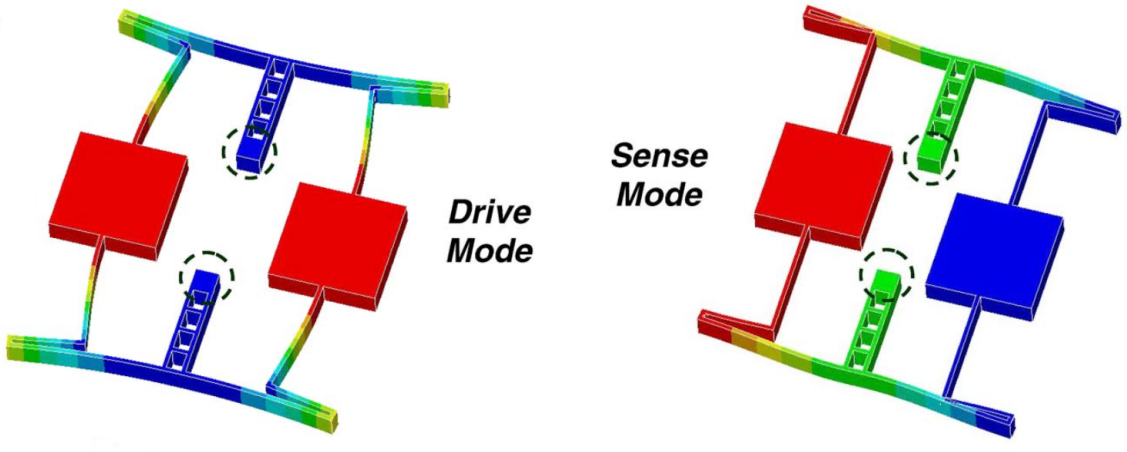
2.1.1.2 Gyrometers

Gyrometers only appeared later in consumer devices. They describe the angular velocity of the device relative to two or three axis depending on the sensor model. In the following, we describe the principles involved in the angular rate measurement for an example of a one-axis gyrometer (see Fig. 2.3, 2.4), before presenting another concept for a single driving mass 3D-axis gyrometer (see Fig. 2.5).

Each gyrometer contains at least one constantly oscillating component, whose displacement is perturbed by the Coriolis effect when the device is rotated. In a reference frame rotating at an angular velocity $\mathbf{\Omega}$, a mass m moving with velocity $\mathbf{v}$ is subjected to a force perpendicular to the vibration direction and to the rotation axis, the force F is defined as

$$\mathbf{F} = 2m\mathbf{v} \wedge \mathbf{\Omega}. \tag{2.3}$$

Thus, the Tuning Fork Gyroscope (TFG), as a one-axis gyrometer represented in Figure 2.3, is formed by a symmetric system of two resonating proof-masses. These masses can resonate along two modes. The drive mode (see Fig. 2.4a) is directed along specific axis (x) and plane (xy) thanks to capacitive combs (D^-, D^+), whose aspect is similar to the accelerometer MEMS. While the system vibrates along the drive mode, any rotation along the axis perpendicular to the plane of oscillation provokes a resonating displacement of the two masses, in opposition of phase, along the direction y, called the sense mode (see Fig. 2.4b). The amplitude of that displacement is measured thanks to symmetric capacitive sensors (S^-, S^+), and is directly linked to the angular



(a) Drive mode. (b) Sense mode.

Figure 2.4 – Resonance modes of the tuning fork gyrometer. Extracted from [ZSZA08].

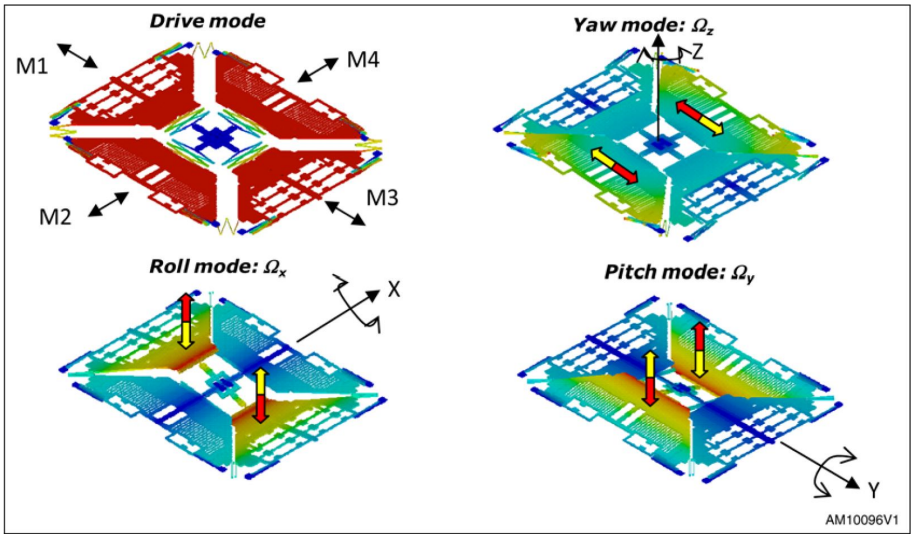


Figure 2.5 – Single driving structure of 3-axis digital gyroscopes. Extracted from [STM].

rate of the rotation. Moreover, if a linear acceleration is applied along an axis, both parts will move in the same direction, which prevents any capacitance differential, and prevents the system from being influenced.

Other gyrometers can sense rotations along three simultaneous axis with a single mass, which reduces their size and their power consumption.

The single driving mass gyrometer, represented in Figure 2.5, is indeed able to cover simultaneously the three angular degrees of freedom, corresponding to every possible rotation relative to the three axes: two in-plane axes, whose rotations are called *roll* and *pitch*, and one out-of-plane axis, whose rotation is called *yaw*. The gyrometer is formed of four parts M_1, M_2, M_3 and M_4, which oscillate along a drive mode: each part vibrates at the same frequency along inward/outward axes. Three other modes exist for the three rotations. Each mode sees two opposite parts oscillating in opposition of phase according to a direction symbolised by the red and yellow arrows. Thus, when an angular rate is applied to an axis, its amplitude can be computed from the opposite displacements of the parts involved in that mode, which create a capacitance differential between these two parts.

2.1.1.3 Flaws of MEMS sensors

MEMS present inherent flaws, which makes their readings uncertain.

Primarily, they can be influenced by physical phenomena. As a spring-mass system, the accelerometer is only able to measure the acceleration resulting of forces applied on its frame. This means that a free-falling accelerometer in a vacuum will show a zero-output, whereas it is accelerating due to gravity. When an accelerometer is placed on a horizontal surface, the system will only read the acceleration resulting from the ground reaction force: the mass is still submitted to gravity, which brings it towards the Earth. This displacement is interpreted by the system as a vertical acceleration in a direction opposite to gravity. This acceleration component introduces a constant bias which is difficult to track, as it is impossible to distinguish it from the acceleration due to the movement of the device.

Moreover, MEMS readings are subjected to non-linearities [Kaa], which can only be neglected under specific conditions, depending not only on parameters such as the sensitivity and quality of the sensor, but also on humidity and temperature. Materials and engineering play an equally important part in reducing the bias drifts present in every inertial MEMS sensor. The accelerometer mass and spring constant may vary with use, while wear may change the quality factor determining the oscillation modes of a gyrometer.

This quality factor has also a large impact in reducing the measurements noise. This noise has multiple origins [BMKW99]: electronic, from the readout circuit, and mechanical, from Brownian motion, which cannot be neglected for MEMS because of the reduced mass and dimensions of the vibrating elements.

Since inertial sensors present multiple flaws which render them unreliable, many gesture recognition applications involve vision-based systems for position tracking.

2.1.2 Vision-based system

Vision-based gesture recognition applications essentially rely on skeleton identification. Multiple strategies are used to generate and track the skeleton. The first method consists in following known points on different body parts in 3D space to reconstruct the skeleton, while the second strategy is more easy to use as it does not involve the specific markers.

2.1.2.1 Marker-based motion capture

Marker-based motion capture relies on specific, camera-identifiable markers disposed on the body or body part to be followed. This type of application is very equipment-intensive, as it requires a specific room setup. Multiple high-speed cameras must be placed around the room, covering different angles of the scene, and have to be calibrated in order to triangulate the markers placed on the subject, which should not be occluded.

The typical motion capture cameras only sense the infra-red spectrum. These cameras also include infra-red emitters, and this specific light is reflected on the markers on the surface of the body. Markers are placed following bio-mechanical considerations, as they have to be located as close to the real skeleton as possible in order to represent accurately the movements. The triangulation process corresponds to a square-error minimisation problem, where the camera orientations are known, and multiple views are combined to provide an estimate of the 3D space position of the marker. This

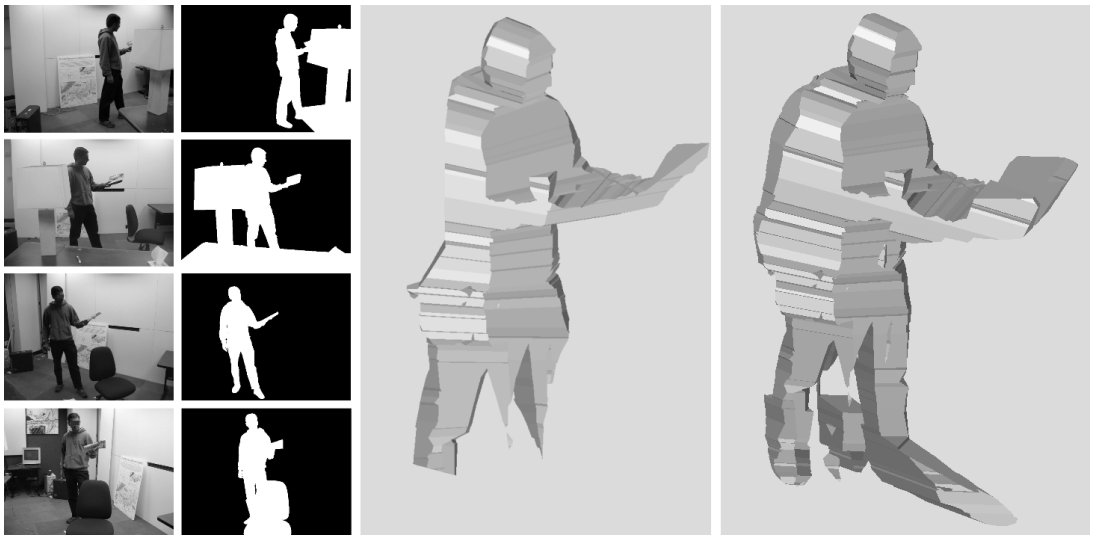


Figure 2.6 – Representation of the visual hull, result of 3D space carving. Extracted from [GSFP06].

process can accurately be executed within a one millimeter error confidence. Occluded markers are reconstructed using machine-learning methods where the set of visible markers serve as features.

2.1.2.2 Markerless motion capture

Many strategies exist for body tracking. For example, as for marker-based motion capture, multiple cameras can be arranged around a room, and the multiple points of view will allow a 3D-carving of voxels (volumetric pixels) by projecting multiple cones on the silhouette of the body, resulting in a visual hull (see Fig. 2.6).

It is also possible to track a body using a single camera. The most famous example of this strategy is the Kinect device (see Fig. 2.7). Originally produced for the Xbox 360 gaming console by Microsoft, a PC version, Kinect for Windows, was released in 2012, with new functionalities. Thanks to its low cost, many motion capture applications and research fields now rely on this device. This technology was licensed to Microsoft by a company called PrimeSense.

Kinect relies on an infrared light source, a depth image sensor, and a color image sensor to propose multiple features for gesture interaction. In particular, it is able to track up to six people, and determine 25 skeletal joints per person. While the details of the depth map production were not revealed, it is safe to assume from the patent [Pri] that the system relies on an infrared projector which projects a specific pattern onto the scene, while a camera captures the image of the projected pattern to reconstruct a 3D map of the scene, based on the grey levels of the captured image and the distortion of the pattern.

The body part inference and tracking process relies on the Kinect camera, which gives 640×480 images at 30 frames per second with depth resolution of a few centimeters. Using a set of 100,000 poses labelled with 31 distinct body parts, Shotton *et al.* [SFC+11] generate synthetic data using computer graphics techniques to render depth and body part images from texture mapped 3D meshes. The final dataset is formed of 900,000 training images, which serve to train 3 randomised decision forests of depth 20.

In order to extract body joints, body parts are first identified, and then serve to

Figure 2.7 – Photograph of a Kinect device.

Figure 2.8 – Depth image, corresponding inferred most likely body part labels and joints proposals. Extracted from [SFC+11].

infer the final corresponding joints (see Fig. 2.8). The body part classifier is based on randomised decision forests, whose split nodes compare simple depth image features $f_\theta(I, \mathbf{x})$ to a learned threshold in order to assign a class to each pixel of each image independently. Given the pixel $\mathbf{x}$ in image I; $\theta = (\mathbf{u}, \mathbf{v})$ a set of two 2D offsets ; and the depth at any pixel $\mathbf{p}$ $d_I(\mathbf{p})$, the feature becomes:

$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right) \tag{2.4}$$

During training, a random subset of 2000 example pixels by image is formed so as to ensure an even distribution across body parts. Then, multiple splitting candidates $\phi = (\theta, \tau)$, with τ the associated threshold, are drawn at random, and used to partition the set of pixel examples depending on their feature exceeding the threshold. The splitting candidate giving the highest gain in information is selected, and the algorithm is repeated on the produced left and right subsets. The final body part classification is operated from three trees trained to depth 20 with one million images.

Finally, joint position estimations are based on the mean shift algorithm [CM02] with a weighted Gaussian kernel. The density estimator per body part c is defined as:

$$f_c(\hat{\mathbf{x}}) \propto \sum_{i=1}^{N} w_{ic} \exp\left(-\left\|\frac{\hat{\mathbf{x}} - \widehat{\mathbf{x_i}}}{b_c}\right\|^2\right) \tag{2.5}$$

where $\hat{\mathbf{x}}$ is a coordinate in 3D world space, N is the number of image pixels, $\widehat{\mathbf{x_i}}$ is the reprojection of image pixel $\mathbf{x_i}$ into world space given depth $d_I(\widehat{\mathbf{x_i}})$, b_c is a learned per-part bandwidth, and w_{ic} is a pixel weighting equal to:

$$w_{ic} = P(c|I, \mathbf{x_i}).d_I(\mathbf{x})^2 \tag{2.6}$$

Starting points for part c are then selected thanks to a learned threshold λ_c. The final identified modes are then shifted along an offset η_c optimised on a validation set in order to "push them back" into the body, at the joints location instead of just the surface of the body.

To conclude this section, a lot more information is available for vision-based methods, relative to inertial based ones. However, this advantage is obtained at the price of a very heavy and pricey equipment, from high-speed cameras to infra-red sensors followed in a controlled and calibrated environment. Conversely, inertial sensors propose a practical and affordable solution for mobile devices, at the cost of a much greater unreliability.

Indeed, it is necessary to apply some specific transformations to inertial sensor data in order to limit the influence of their flaws identified above. In the following, we will present these data processing steps.

2.2 Inertial gesture data processing

As shown above, inertial gesture data are the combination of two independent sensors, the accelerometer and gyrometer. Thus, a gesture sample corresponds to two time-series of three dimensional measurement, which present inherent flaws introducing, noise, drift and biases in the raw measurements. Moreover, these sensors may not be perfectly synchronised.

Before any feature computation or classification task, it is mandatory to operate four steps to extract the essential information for gesture recognition: calibration, in order to remove the bias induced by gravity on the accelerometer data; filtering, to improve the signal-to-noise ratio; normalisation, dealing with amplitude gaps between repetitions; and vectorisation, to reduce temporal discrepancies between different repetitions of the same gesture and compress the information.

Thus, these preprocessing steps not only improve data alignment for one-user cases, but also contribute to reducing the differences which naturally exist between different users in open-world applications.

2.2.1 Calibration

The first step consists in calibrating the accelerometer signals, in order to filter out the constant due to the reactive force to the gravitational pull of the Earth on the device. Indeed, in the case where the device is held in the hand, two same gestures performed with different wrist angles see their reference frame shifted, and result in two completely different accelerometer readings.

Cho *et al.* [CCB+06] suggest an approximate gravity removal by substracting the mean vector of accelerations to each measurement.

In [Pyl05], Pylvänäinen suggest an additional calibration for Smartphone readings, relying on the fact that any gravity estimation $\mathbf{g}_T(D)$ over a dataset D in a 3D-space parametrised with the axes xyz, should point toward the negative y-axis, which corresponds to the position where the phone is held vertically, screen towards the face of the subject. The gravity estimation is performed the same way as above, using the mean vector of accelerations of the gesture duration. Since the sensors axis form an orthogonal base for the acceleration space, this operation translates to a rotation or roto-inversion of the initial frame thanks to an orthonormal matrix R. The axes of this matrix $R = \begin{pmatrix} \mathbf{r_1}^\top \\ \mathbf{r_2}^\top \\ \mathbf{r_3}^\top \end{pmatrix}$ are determined by means of a Gram-Schmidt orthogonalisation: the first axis to be determined is the one defined by the unit vector aligned with the gravity estimation:

$$\mathbf{r_2}^\top = -\frac{\mathbf{g_T(D)}}{\|\mathbf{g_T(D)}\|}. \tag{2.7}$$

Given $(\mathbf{u}, \mathbf{v}) \in \mathcal{R}^n \times \mathcal{R}^n)$, the orthogonal projection operator $\mathbf{proj_u(v)}$ is defined as:

$$\mathbf{proj_u(v)} = \frac{\mathbf{u.v}}{\|\mathbf{u}\|^2}\mathbf{u}. \tag{2.8}$$

The last two remaining vectors are then calculated using corrected projections of the previous unit vectors $\hat{x}, \hat{z}$, directing the axes x and z :

$$\begin{aligned} \mathbf{r_1} &= -\frac{\hat{\mathbf{x}} - \mathbf{proj_{r_2}}(\hat{\mathbf{x}})}{\|\hat{\mathbf{x}} - \mathbf{proj_{r_2}}(\hat{\mathbf{x}})\|}, \\ \mathbf{r'_3} &= \hat{\mathbf{x}} - \mathbf{proj_{r_2}}(\hat{\mathbf{z}}), \\ \mathbf{r_3} &= -\frac{\mathbf{r'_3} - \mathbf{proj_{r_1}}(\mathbf{r'_3})}{\|\mathbf{r'_3} - \mathbf{proj_{r_1}}(\mathbf{r'_3})\|}. \end{aligned} \tag{2.9}$$

2.2.2 Filtering

Filtering is a common signal processing step destined to remove an unwanted component from a signal. In our case, the goal is to remove the noise caused by the sensor itself, or short, quick and unwanted movements of the subject.

Low-pass and Butterworth filters are classically used as the noise to be filtered is high-frequency. Indeed, both filters are defined by a cutoff frequency, from which higher frequencies are progressively attenuated.

The first-order low-pass filter [Mli09] corrects every sample of a signal by considering its recent past. Thus, for a signal $\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$, the sample $\mathbf{X_i}$ is adjusted thanks to a parameter $\beta \in [0; 1]$, which determines the weight given to the past samples in the filtered signal $\mathbf{X^f}$. The filter function $H_{low}(z)$ after z-transform, where z^{-1} correspond to a time-shift, is equal to:

$$H_{low}(z) = \frac{1-\beta}{1-\beta z^{-1}} \tag{2.10}$$

which easily translates in temporal form to:

$$\mathbf{X^f}_t = \beta \mathbf{X}_t + (1-\beta)\mathbf{X}_{t-1} \tag{2.11}$$

The Butterworth filter is often used up to the 4th order [MMST00]. Its frequency response is characterised by a flat pass-band and linear reject-band. The coefficient of the z-transform can be found thanks to the normalised Laplace transform, depending on the order $n \in [1, 4]$ of the filter:

$$H_{butter}(s) = \frac{1}{P_n(s)} \tag{2.12}$$

with

P_1	$1 + s$
P_2	$1 + 1.414s + s^2$
P_3	$(1 + s)(1 + s + s^2)$
P_4	$(1 + 0.765s + s^2)(1 + 1.848s + s^2)$

A bilinear transform is then applied to determine the coefficient of the time-discrete filter, using a first-order approximation of $s = \frac{1}{T}\ln(z)$, with T the period related to the sampling rate, to give the following replacement formula:

$$s \equiv \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}. \tag{2.13}$$

Another acceleration time series smoothing process implies using a mean filter [HHH98] [AV10] [MHS01] [LC13], where the information is temporally compressed over a sliding averaging window. For a window of k samples

$$\mathbf{X^f}_t = \frac{1}{k}\sum_{i=0}^{k-1} \mathbf{X}_{t+i}. \tag{2.14}$$

Finally, other applications apply a Fast Fourier Transform (FFT) or Discrete Cosine Transform (DCT) [HJZH08], before cutting the high-frequency coefficients, which is equivalent to a low-pass filter. However, this method may introduce signal oscillations in the temporal domain of the data near discontinuities, known as the Gibbs phenomenon.

After the noise attenuating phase, it is necessary to reduce the amplitude differences between multiple repetitions of a gesture and different dynamics from multiple users, process which is explained in the following section.

2.2.3 Spatial Normalisation

The second most important source of discrepancies between users comes from spatial amplitude. A fast user will produce a higher overall acceleration while performing the same gesture as a more laid-back user. A specific normalisation step is thus necessary. Since no a-priori exists on the performing user, this preprocessing phase should be user independent, and not rely on multiple repetitions of the same gesture.

One first normalisation strategy choice consists in an alignment of the probability distribution of the data towards a normal distribution with a zero mean and unit variance for every sensor component [MHS01] [MMST00].

Another common process scales every sensor component of the filtered time series $\mathbf{X^f} = \left[\mathbf{X^f}_1, \ldots, \mathbf{X^f}_n\right]$ independently and in a linear manner [MKKK04][ZCL+11]. The scaled time series becomes $\mathbf{X^s} = [\mathbf{X^s}_1, \ldots, \mathbf{X^s}_n]$, with

$$\mathbf{X^s}_i = \frac{\mathbf{X^f}_i - \min_j(\mathbf{X^f}_j)}{\max_j(\mathbf{X^f}_j) - \min_j(\mathbf{X^f}_j)} \tag{2.15}$$

While the min-max scaling conveniently projects every component onto the $[0; 1]$ segment, it also discards the relative scalings between different sensor components. For that reason, another strategy may be applied, where the components are scaled by a unique factor depending only on the gesture. This factor is commonly chosen as the maximum norm of the time-series concatenation of the different sensors readings [CCB+06] [HL10]. Given the gesture data $\mathbf{X} = \{\mathbf{X_1}, \ldots, \mathbf{X_n}\}$:

$$\mathbf{X^s}_i = \frac{\mathbf{X}_i}{\max_j(\|\mathbf{X}_j\|)} \tag{2.16}$$

While amplitude discrepancies may be the result of broader gestures, they are often the product of generally more dynamic styles, where gestures are performed more quickly. As such, amplitude normalisation is often completed with a temporal scaling in order to improve the alignment of samples produced with different styles.

2.2.4 Temporal Normalisation

Directed by specific kinetics, human motion is highly redundant, which suggest to reduce the actual number of measurements describing the gesture. Moreover, the size of gestures performed with different speeds will differ greatly. A step of temporal normalisation is destined to erase those differences, by thresholding, resampling and quantisation.

Thresholding proposes to exclude the measurements which do not bring more discriminative information about the gesture, for models to identify more clearly the meaningful part of the signal. Two types of thresholding filters exist. The first one is called the "idle thresholding filter", used to remove samples with low importance. Mlich [Mli09] imposes a threshold on the absolute value of the acceleration samples. Given a threshold t and the value of the acceleration due to gravity g, a sample acceleration vector $\mathbf{A_i} = \left\{a_{xi}, a_{y_i}, a_{zi}\right\}$ is removed if:

$$|\|\mathbf{A_i}\| - g| < t \tag{2.17}$$

Schlomer *et al.* [SPHB08] suggest a "directional equivalence filter" where vectors too close to their predecessor are removed from the final gesture data. For a defined

threshold ϵ, the thresholded signal $\mathbf{X_{th}} = \{\mathbf{X_{th1}}, \ldots, \mathbf{X_{thm}}\}$ presents the following characteristic:

$$\forall i \in [2; m], \|\mathbf{X_{thi}} - \mathbf{X_{thi-1}}\| \geq \epsilon \tag{2.18}$$

Resampling allows a standardisation of different gestures durations. Kela *et al.* [KKM+05] linearly interpolate or extrapolate the gesture data if any sequence is respectively too short or too long. Cho *et al.* [CCB+06] suggest to impose a predetermined length of accelerations signals, and resample the signal in order to get a predetermined unit length U between successive points.

The second resampling strategy complements the first one. A vector quantisation step may be applied on gesture data in order to discretise the data. Vector quantisation is a lossy compression methods which consists in mapping the vector space to a set of exemplars, forming a "codebook", which permits to encode every sequence as a one-dimensional exemplar label vector.

Codebooks are often produced as the result of a clustering process of the data. The most popular clustering algorithm is the k-means algorithm, also called Lloyd's algorithm [HHH98][Kli09] [AV10][KKM+05]. This algorithm performs an unsupervised clustering, which updates a predefined number of cluster centres towards the centres of mass of the dataset. Vector references are drawn randomly at first, and assigned a set of data samples on a lowest distance criterion. This process creates a set of clusters, whose means are computed to serve as new references. Thus, the set of references converges towards the highest densities of the data distribution, justifying the representative character of the obtained references. However, this method is strongly affected by the initial references choice, and only converges towards a local minimum. Moreover, it is not possible to determine the exact number of representatives needed to accurately describe the data without losing any crucial information.

However, other codebooks come from a simple even discretisation of the whole sensors reading space. While this method requires less computation, it is plagued by the often greater size of the representatives set, as enough granularity is needed to efficiently differentiate two quantised gestures.

Thus, after this overview of gesture detection and capture, using descriptions whether inertial or vision-based, we will see in the next section how these sensors have been used for gesture recognition.

2.3 Gesture Data Learning and Classification

Due to the recent development of the inertial sensors, which are the main enablers for widespread and accessible gesture recognition, this field is relatively young compared to other problems, such as speech or face recognition. However, gesture recognition benefits from the experience accumulated in these disciplines, and offers a wide range of approaches. Three main strategies can be identified.

The first one consists in a statistical modelling of the gesture dynamics, with models whose main representative is the Hidden Markov Model (HMM).

The second approach is based on geometric considerations. As an elastic distance, the Dynamic Time Warping (DTW) allows for a direct similarity between gestures, while other methods are based on some discriminative geometric features, extracted thanks to a Principal Component Analysis (PCA), or a Linear Discriminant Analysis (LDA) for instance.

Finally, the third approach consists in building a specific classifier, such as Adaboost, Support Vector Machines (SVM), or artificial neural networks. As the theoretical base for our study, the artificial neural network-based classifiers are presented in a more detailed manner in a last section.

2.3.1 Statistical methods

Statistical methods are suited for time series in particular. They aim at producing a generative model from the analysis of the underlying distribution and dynamics of samples, also called "observations".

Two main approaches are classically followed: Bayesian-based approaches, and their Hidden Markov Models (HMMs) specialisation.

2.3.1.1 Naïve Bayes

The Naïve Bayes (NB) classifier gets its name from its original assumption that every component of the signal is statistically independent during the use of the Bayes theorem. Thus, for a sample $\mathbf{x} = \{x_1, \ldots, x_n\}$, the probability of a class C_m given the sample is equal to:

$$\begin{aligned} P(C_m|\mathbf{x}) &= \frac{P(C_m)P(\mathbf{x}|C_m)}{P(\mathbf{x})} \\ &\propto P(C_m)\prod_{i=1}^{n} P(x_i|C_k). \end{aligned} \tag{2.19}$$

The distributions of all the continuous variables are considered as Gaussian. Every distribution is estimated based on the training dataset. The final choice is made on a maximum likelihood criterion.

Rehm *et al.* [RBA08] apply the Naïve Bayes classifier on three gesture datasets. The first dataset is composed of the 10 digits, the second, formed by 7 gestures, is inspired by the "Berlin Dictionary of German Everyday Gestures", and the third one is created for Video Cassette Recording (VCR) control. 16 features are extracted from the accelerometer signals, such as the length of the signal, the minimum and maximum for each axis, the median and mean for each axis, and the gradient for each axis. The Digits dataset is composed of the 10 digits recorded 10 times by 7 users. The NB classifier shows a poor performance of 58.1% on a 10-fold cross-validation in a user-independent setting, while it can show very inconsistent results in the user-dependent case, ranging from 90.2% to 100% for different users. The other two datasets are user-dependent, and prove this inconsistency with a 88.6% accuracy for the German emblems problem, and 99.6% for the VCR control.

2.3.1.2 Bayesian Networks

Bayesian Networks [Jen96], or Belief Networks, are a type of probabilistic graphical models. Composed of nodes and directed arcs, respectively representing random variables and their relationships, the network takes the form of a directed acyclic graph.

Cho *et al.* [CCB+06] use Bayesian networks in order to compute one gesture model per class (see Fig. 2.9). Firstly, gestures are described as a set of primitives, defined as a portion of acceleration signals whose values increase or decrease monotonously. These primitives are thus separated by feature points, called "end-points" EP, which are local minima or maxima in the acceleration space. Secondly, these primitives are

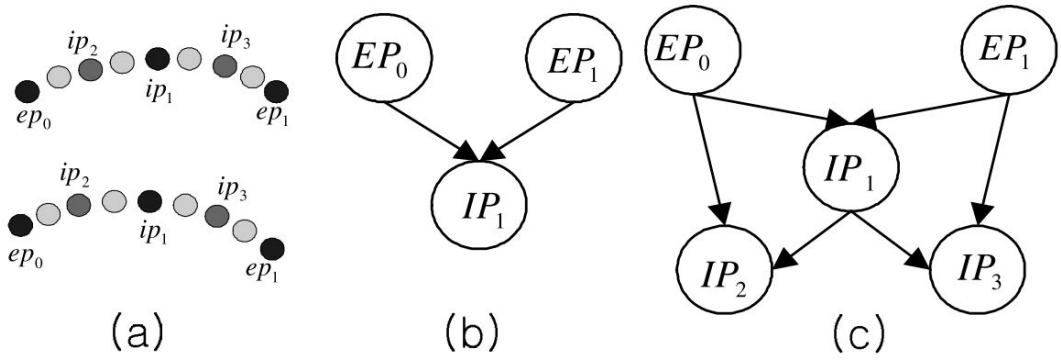


Figure 2.9 – Example of a recursive construction of a primitive model. Gesture primitives, formed by two end-points (EP) which are local extrema for the acceleration signal, are recursively decomposed in inter-primitive points (IP), located at equal distance from their parents). Extracted from [CCB+06].

recursively decomposed into "primitive models", composed of point models, or "inter-primitive" points IP, which are located at equal distance from their two parents; and their relationships.

These relationships correspond to 3D conditional Gaussian distributions, for the three acceleration axes of multiple repetitions of the same gesture. Given a multivariate random variable X depending on $X_1, \ldots, X_n$, i.e. given a node X and its parents $Pa(X) = X_1, \ldots, X_n$, and μ and Σ its respective conditional mean and covariance matrix, then conditional probability distribution for a realisation $x_1, \ldots, x_n, x$ is equal to:

$$P(X = x | X_1 = x_1, \ldots, X_n = x_n) = (2\Pi)^{-d/2} |\Sigma|^{-1/2} \exp\left([x-\mu]^\top \Sigma^{-1} [x-\mu]\right) \quad (2.20)$$

with d the dimension of X.

$\mu = \mu_{X|Pa(X)}$ can be computed from the example-learned mean and covariance matrix for each random variable thanks to a linear regression [Mur98]. Given concatenation of the random variables $Y = \begin{pmatrix} X \\ Pa(X) \end{pmatrix}$; with $Pa(X) = \left(X_1^\top, \ldots, X_n^\top\right)$, $\mu_Y = \begin{pmatrix} \mu_X \\ \mu_{Pa(X)} \end{pmatrix}$, $\Sigma_Y = \begin{pmatrix} \Sigma_X & \Sigma_{XPa(X)} \\ \Sigma_{Pa(X)X} & \Sigma_{Pa(X)} \end{pmatrix}$, and a realisation $y = (x, Pa(x))$ with $Pa(x) = \left(x_1^\top, \ldots, x_n^\top\right)$, we can compute:

$$\begin{aligned} \mu &= \mu_{X|pa(X)} \\ &= \mu_X + \Sigma_{XPa(X)} \Sigma_{Pa(X)}^{-1} (Pa(x) - \mu_{Pa(X)}) \end{aligned} \quad (2.21)$$

and

$$\Sigma_{XPa(X)} = \Sigma_X - \Sigma_{XPa(X)} \Sigma_{Pa(X)}^{-1} \Sigma_{Pa(X)X}. \quad (2.22)$$

The training phase consists in computing the single means and covariance matrices for each node using the set of training examples. One model is trained for each class.

Classification with Bayesian Networks

A 3D-accelerometer gesture sequence $\mathbf{G} = G(1), \ldots, G(T)$ is classified on a maximum likelihood criterion on the set of trained models. Given λ_m the model corresponding the class m, the model $\lambda^*(\mathbf{G})$ assigned to $\mathbf{G}$ is obtained with:

$$\lambda^*(\mathbf{G}) = \arg\max_m P(\lambda_m | G(1), \ldots, G(m)) \quad = P(\lambda_m) P(G(1), \ldots, G(m) | \lambda_m). \quad (2.23)$$

Primitives must be recursively matched on all possible segmentations of a gesture in order to calculate $P(G(1), \ldots, G(m)|\lambda_m)$. Thus, for a specific segmentation $\gamma_i = (t_0, t_1, \ldots, t_n)$, the likelihood corresponds to the product of the conditional probabilities of all nodes given their respective parents. The final likelihood should operate a sum over all possible segmentations, which leads to an exponential complexity. Thus, only estimates are computed, approximating the model likelihood with the likelihood of the most probable segmentation. Moreover, the conditional probabilities of end-points are initially ignored to only identify the most probable primitives segmentations, whose likelihoods are then re-scored considering the end-points.

Bayesian Networks for gesture recognition
Cho *et al.* [CCB+06] propose to use Bayesian Networks to model gestures captured with a Gesture interactive cell-phone. They obtain a classification score of 96.3% based on a 100-user, 14-gesture database divided into four-folds. This approach is still limited, as its recursive intra-primitive point selection is not able to discriminate efficiently between similar gestures such as "0" and "6".

While this Bayesian network models the general statistical shape of the gesture, Hidden Markov Models allow to consider the temporal output dynamics.

2.3.1.3 Hidden Markov Models

A temporal signal may be assumed to follow a statistical Markov process, where the measured observations are the result of emissions produced following the specific statistical laws of multiple "states", accessible thanks to a set of transitions. However, these states may not be directly observable, which is why we refer to Hidden Markov Models (HMM).

An HMM is a generative model, formally defined by the parameters $\{S, A, B, \pi\}$, with S the set of hidden states, $A = a_{ij}$ the transition probabilities matrix between these states, B the emission probability function/density for each state, and π the probabilities for each state to be an initial state.

Different state transition architectures lead to different signal modelling. An HMM is said to be ergodic if all its states can operate transitions between them. In the case of a temporal signal, it is safe to assume that no turnabouts happen in the state space, leading to the most oftenly "bakis", or left-to-right, model, where transitions only exist towards forward states (see Fig.2.10).

Two types of HMM exist, depending on the observations type: the discrete HMM (dHMM), and the continuous HMM (cHMM). The dHMM corresponds to a fixed-size set of potential observations, whose probability distribution is consequently discrete. This kind of approach thus requires the step of vector quantisation. Conversely, the cHMM emissions follow continuous probability distributions. This aspect allows for a better representation of the acceleration time-series, which varies in a continuous manner. The model is more capable to handle short incoherences, as an unseen sample for a dHMM results in a zero likelihood. However, as seen below, model updates and sequence likelihood computation require additional steps compared to the simpler dHMM.

Two problems have to be tackled in order to use HMMs for recognition purposes: the evaluation and the training problems.

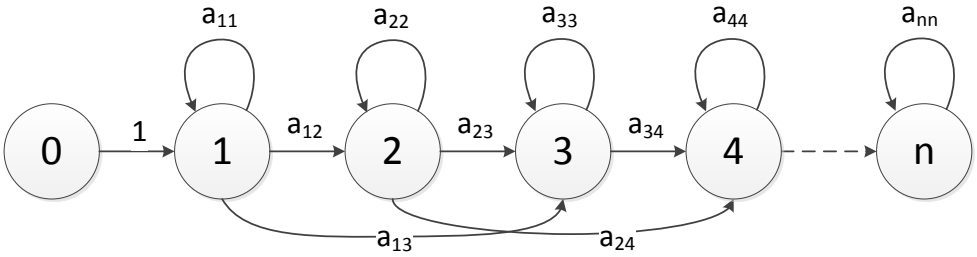


Figure 2.10 – Example of a left-to-right HMM, with two forward connections.

Classification with HMMs

As for the Bayesian network seen above, a sequence $\mathbf{G} = G(1), \ldots, G(T)$ is classified on a maximum likelihood criterion on the set of trained models, with one model per class, where λ_m the model for the class m. Classification is then performed by selecting the model λ_{m^*} such that

$$m^* = \arg\max_m P(\lambda_m | G(1), \ldots, G(T)) \quad = P(\lambda_m) P(G(1), \ldots, G(T) | \lambda_m). \tag{2.24}$$

It is thus necessary to be able to evaluate $P(\mathbf{G}|\lambda_m)$.

Two algorithms based on the same concept are commonly used: the "Forward" and "Backward" algorithm. The Forward algorithm computes the likelihood $\alpha_t(i)$ that the model ends up in the state ω_i at time t, after producing the subsequence $G(1), \ldots, G(t-1)$. Thus, the likelihood $P(G(1), \ldots, G(m)|\lambda_m)$ is computed as the sum over all the states $S = \omega_1, \ldots, \omega_N$ of the alpha values at time T:

$$P(G(1), \ldots, G(m)|\lambda_m) = \sum_{i=1}^{N} \alpha_T(i). \tag{2.25}$$

This value is easily computed thanks to a recursive algorithm which efficiently drops the $O(N^\top)$ complexity to $O(N^2 T)$. Given π_j the probability for ω_j to be an initial state, and $b_j(G(1))$ the emission probability of the first observation by this same state, then we define:

$$\forall j \in [1; N], \alpha_1(j) = \pi_j b_j(G(1)). \tag{2.26}$$

It is then possible to establish a recursive relation between α_t and α_{t+1}. The likelihood to get to state ω_j at time $t+1$, producing the sequence $G(1), \ldots, G(t+1)$, is equal to probability to be in any state ω_i at time t and producing the sequence $G(1), \ldots, G(t)$ with likelihood $\alpha_t(i)$, then operate a transition to ω_j with probability a_{ij}, and emitting the observation $G(t+1)$ from ω_j with probability $b_j(G(t+1))$:

$$\alpha_{t+1}(j) = b_j(G(t+1)) \sum_{i=1}^{N} \alpha_t(i) a_{ij}. \tag{2.27}$$

Similarly, the Backward algorithm computes the likelihood $\beta_i(t)$ that the model ends up in the state ω_i at time t, after producing the subsequence $G(t+1), \ldots, G(T)$. We define:

$$\forall j \in [1; N], \beta_T(j) = 1, \tag{2.28}$$

$$\beta_t(j) = \sum_{i=1}^{N} \beta_{t+1}(i) a_{ji} b_i(G(t+1)). \tag{2.29}$$

HMM training

The training problem consists in updating the model parameters to maximize the likelihood of training samples. The most commonly used algorithm is called the "Baum Welch" (BW) algorithm, or "Forward-Backward", which corresponds to a Generalised Expectation Maximisation. BW relies on the computation of the alpha and beta values defined in the Forward and Backward algorithm for each available training sample to redefine and update iteratively the transition, emission and starting states probabilities. For instance, it is possible to compute for each sample the likelihood of a transition from the state i to state j at a time t, and the reestimation of the transition probabilities consists in summing these values over every time step, and normalising the result to ensure $\sum_j \hat{\alpha}_i j = 1$.

Thus, two new intermediate variables are necessary. $\xi_t(i,j)$ is the probability of being in ω_i at time t and ω_j at time $t+1$, producing the entire sequence $\mathbf{G}$. In terms of α and β, we define:

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(G(t+1))\beta_{t+1}(j)}{P(\mathbf{G}|\lambda)} = \frac{\alpha_t(i)a_{ij}b_j(G(t+1))\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(G(t+1))\beta_{t+1}(j)}. \tag{2.30}$$

$\gamma_t(i)$ is the probability of being in state ω_i at time t, producing the entire sequence $\mathbf{G}$. This can be interpreted as the probability to be in the state ω_i at time t and any state ω_j at time $t+1$, given $\mathbf{G}$ and λ, we define:

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j). \tag{2.31}$$

Using these new variables, we can define the general re-estimation formulas for $\hat{\pi}$, $\hat{A}$, and $\hat{B}$. π_i corresponds to the probability to be in state ω_i at $t=1$, given $\mathbf{G}$ and λ_m,

$$\hat{\pi}_i = \gamma_1(i). \tag{2.32}$$

a_{ij} is the probability to be in state ω_i at any time and operate a transition to state ω_j, which is equal to the following normalised sum

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{k=0}^{N}\sum_{t=1}^{T-1}\xi_t(i,k)} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}. \tag{2.33}$$

The emission probabilities are updated differently, depending on the discrete or continuous character of the HMM. We detail two cases: the discrete HMM, with discrete emission probability tables, and the most common continuous HMM, with Gaussian mixture emission probabilities.

Firstly, in the case of a dHMM, $b_j(l)$ is the probability to observe the l^{th} codebook vector ν_l from state ω_j, which is equal to the following normalised sum:

$$\hat{b}_j(l) = \frac{\sum_{\substack{t=1 \\ s.t. G(t)=\nu_l}}^{T-1}\gamma_t(j)}{\sum_{t=1}^{T-1}\gamma_t(j)}. \tag{2.34}$$

In the case of a Gaussian-mixture cHMM, the K-Gaussian-mixture emission probability distributions $b_j(G(t))$ at state ω_j, with means μ_{jk}, covariance matrices Σ_{jk}, and

coefficients c_{jk}, are defined as:

$$b_j(G(t)) = \sum_{k=1}^{K} c_{jk} N(G(t), \mu_{jk}, \Sigma_{jk}). \tag{2.35}$$

It is necessary to define a generalisation of $\gamma_t(i)$, to take into account the part played by every Gaussian in the mixtures. Thus, we define $\gamma_t(j,k)$ as the probability of being in state ω_j at time t with the k^{th} mixture component accounting for $G(t)$:

$$\gamma_t(j,l) = \gamma_t(j) \frac{c_{jl} N(G(t), \mu_{jl}, \Sigma_{jl})}{\sum_{k=1}^{K} c_{jk} N(G(t), \mu_{jk}, \Sigma_{jk})}. \tag{2.36}$$

Thus, re-estimations $\hat{b}_j$ of b_j imply three parameter re-estimations, one for each $\hat{c}_{jl}$, $\hat{\mu}_{jl}$, $\hat{\Sigma}_{jl}$.

As the coefficient reflecting the proportion played by the l^th Gaussian at the state ω_j at any time t, c_{jl} is re-evaluated as the normalised sum over all the time steps:

$$\hat{c}_{jl} = \frac{\sum_{t=1}^{\top} \gamma_t(j,l)}{\sum_{k=1}^{K} \sum_{t=1}^{\top} \gamma_t(j,k)}. \tag{2.37}$$

The means and covariance matrices are updated in a similar manner:

$$\hat{\mu}_{jl} = \frac{\sum_{t=1}^{\top} \gamma_t(j,l) G(t)}{\sum_{t=1}^{\top} \gamma_t(j,k)} \tag{2.38}$$

and

$$\hat{\Sigma}_{jl} = \frac{\sum_{t=1}^{\top} \gamma_t(j,l)(G(t) - \mu_{jk})(G(t) - \mu_{jk})^\top}{\sum_{t=1}^{\top} \gamma_t(j,k)}. \tag{2.39}$$

The HMM for gesture recognition

Hofmann *et al.* [HHH98] use an alphabet size of 120 samples of joint finger angles and accelerations for the TUB-SensorGlove. After a study of HMM configurations and training set size, they show a highest recognition rate of 95.6% for both a 5-state ergodic HMM and a 3-state left-to-right HMM, using a two-fold cross-validation on a dataset of 100 gestures repeated 20 times by two persons. Kallio *et al.* [KKM03] study the influence of the number of training samples, showing that curvilinear movements need more training vectors. A recognition rate of over 95% is attained with 16 samples for each of the 16 gestures for a single user, using a 5-state HMM with a codebook size equal to 8.

Zhu *et al.* [ZS09] tackle online gesture recognition step with hierarchical HMMs complimented by gesture spotting. Gesture start and end points are determined thanks to a feed-forward neural network, trained to detect gesture and non-gesture sequences using twelve features extracted from one-second sliding windows over the accelerometer and gyrometer data, with both means and variances for each axis of both sensors. This

segmentation is then fed to a Hierarchical HMM (HHMM), generalisation of the HMM where every state is itself a HHMM, with one upper level HMM, and one lower level HMM per class. The lower level HMMs compute time-series features extracted on sliding windows of 20 samples from the gesture data, and a majority voting to classify 150-sample gestures. This result is updated with an online Bayesian filtering, utilising the context information in the upper level HMM. A detailed explanation for the HHMM training was proposed by Fine *et al.* [FST98].

Lukowicz *et al.* [LWJ+04] rely on the PadNET sensor network with three 3D accelerometers positioned on both wrists to recognize 8 workshop activities for one subject, showing a 95.51% overall accuracy with a leave-one-out protocol on 10 repetitions of each gesture. Emission probabilities are modelled using a 3D Gaussian distribution for each state, while the number of state is actually manually adjusted for each class model. Pylvänäinen [Pyl05] test the performance of the Gaussian HMM on a dataset of 10 gestures, with 20 samples per gesture for 7 participants. Thanks to a cross-validation using 3 samples for training and 17 samples for testing, Gaussian cHMM show a 96.76% accuracy in the single-user case, and 99.76% accuracy for the multi-user case, difference explained by the author with a potential overfitting in the single-user case. Finally, Zhang *et al.* [ZCL+11] use multi-stream HMMs on accelerometer and electromyography sensors at the bottom of a decision tree detecting static or dynamic gestures, short or long durations and hand orientation. The multiple sensors are handled by two separate, independent HMMs, with a fusion corresponding to the sum of their log-likelihoods: given the weights of each stream/sensor w_k so that $\sum_{k=1}^{K} w_k = 1$, and λ the parameters of the global model, λ^k the parameters of the model specific to the k^{th} stream, and $\mathbf{G}^k$ the portion of the gesture data relative to that stream, we get:

$$\log(P(\mathbf{G}|\lambda)) = \sum_{k=1}^{K} w_k \log(P(\mathbf{G}^k|\lambda^k)). \tag{2.40}$$

Probabilistic approaches allow for a generic framework based on known statistical tools. However, they remain limited due to the high statistical independences assumptions on which they rely. The mid/end-point modelling of the Bayesian Network does not provide enough detail for every gesture to be identified independently. HMMs present other drawbacks: it is still difficult to determine the correct probability distribution for the HMM states, whose number has to be determined experimentally. Moreover, these methods require extensive preprocessing steps in order to ensure a distribution alignment between multiple repetitions of the same gesture. Probabilistic methods depend highly on orientation and temporal variations. For example, temporal delays are problematic for HMMs, whose likelihood drops exponentially with the number of transitions needed to the same state, forcing a transition to another state, whose emission distribution may not be adapted anymore.

Given the high variations between repetitions, another strategy consists in comparing directly the geometry of the gesture signals.

2.3.2 Geometric-based methods

Metric-based methods consist in classifying the sample thanks to a decision based on exemplars, usually, a k-Nearest-Neighbour (K-NN).

Three common approaches are identified: the DTW-based; PCA-based; and the LDA-based ones.

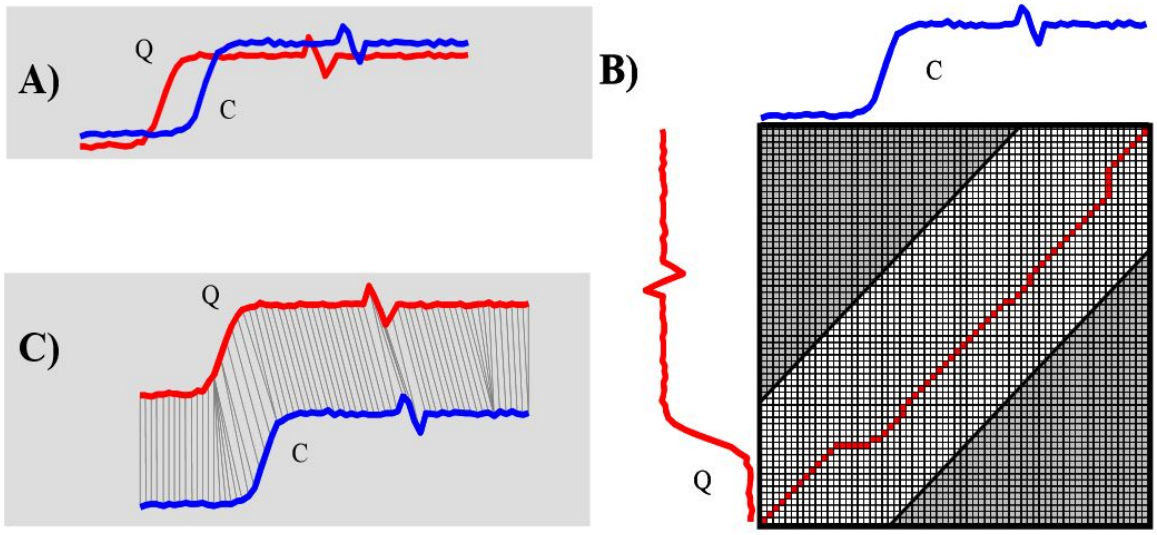


Figure 2.11 – Extracted from [RK04]. Example of DTW computation between two similar, out-of-phase signals Q and C, represented in A. B corresponds to the warping matrix, with the minimum-weight warping path between the two signals. C is the representation of the resulting alignment.

2.3.2.1 Dynamic Time Warping

The DTW metric is an elastic metric between two time series of different sizes. Given two time-series $\mathbf{Q} = \{Q(1), \ldots, Q(M)\}$ and $\mathbf{T} = \{C(1), \ldots, C(N)\}$, this algorithm consists in finding the minimum weight reconstruction path followed to transform $\mathbf{Q}$ into $\mathbf{C}$ (and reciprocally). The DTW metric computation requires an $M \times N$ matrix D (see Fig. 2.11), where D_{ij} represents the warping path cost between $Q(i)$ and $C(j)$. This cost can be found using dynamic programming, following the recursive relation

$$D_{ij} = d(Q(i), C(j)) + min\{D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1}\} \tag{2.41}$$

with $d(Q(i), C(j))$ the Euclidean distance between the two sample vectors $Q(i)$ and $C(j)$.

In order to limit the computations, multiple heuristics may be applied on the warping matrix, limiting the path possibilities (see Fig. 2.11.B).

Barczewska *et al.* [BD13] compare the DTW to two variants, the Derivative Dynamic Time Warping (DDTW), and Piecewise Dynamic Time Warping (PDTW), applied to forefinger gesture recognition. DDTW is the same DTW algorithm, applied to the estimated derivatives of the signals, while PDTW operates an averaging steps over fixed-size frames before applying the DTW. Tests were carried out using 10 gesture classes for 9 users, with 15 training and 9 test gestures per user and per class. In the user-dependent case, one repetition was used as exemplar among 15 training gestures on a minimum-DTW criterion with the others. For the user-dependent case, this step was repeated using the exemplar for each user, and selecting the one most similar to the others. The results show little difference between the 3 methods for the user dependent case, with an accuracy of about 94%, while the user dependent case proves the superiority of the original DTW, with a 85.4% accuracy, against 83.6% for DDTW, and 62.6% for PDTW. This study emphasizes the choice of the most efficient exemplar.

Likewise, Choe *et al.* [CMC10] propose a method to reduce the computation cost of the DTW, which requires as many comparisons as training exemplars, with a modifica-

tion of the k-means algorithm. Thus, a clustering step is applied to each gesture class training data in order to extract the most representative exemplars: a set of gesture signals is used for initialisation, and clusters are formed based on a minimum criterion using the DTW metric.

While the application above limits intra-class distance variations, Hartmann *et al.* [HL10] suggest considering the inter-class distances as well to optimize the DTW prototypes, using four approaches: the minimal interclass to maximal intraclass distance; the center point distance, based on the difference between distribution centre points; the Kullback-Leibler Divergence; and the Error Function Integral.

Akl *et al.* [AV10] propose an Affinity Propagation clustering step to determine the most relevant exemplars. Affinity propagation (AP) [FD07] is a process involving passing two kind of messages between data points: responsibility $r(i,k)$, which indicates how well the point k is suited to be an exemplar for the point i, taking into account the other potential exemplars; and availability $a(i,k)$, which reflects how appropriate it would be for i to pick j as an exemplar, taking into account the support from other points that k should be an exemplar. Exemplars are selected depending on the value of k which maximises $a(i,k)+r(i,k)$. AP is an iterative process, repeated until convergence of the exemplars, and initialised with two matrices: $\mathbf{A}$ corresponds to the availability matrix, initially equal to the zero-matrix; and $\mathbf{R}$ is the responsibility matrix, initialised thanks to similarity measures $s(i,k)$ between every pair of points (i,k) in the dataset. Thus, the initial values $r(i,k)$ are computed as:

$$r(i,k) = s(i,k) - \max_{k' \ s.t. \ k' \neq k} \{s(i,k')\}\,. \tag{2.42}$$

$\mathbf{A}$ and $\mathbf{R}$ are updated as:

$$\begin{aligned} r(i,k) &= s(i,k) - \max_{k' \ s.t. \ k' \neq k} \{a(i,k') + s(i,k')\}\,, \\ a(i,k) &= min\left\{0, r(k,k) + \sum_{i' \ s.t. \ i' \notin \{i,k\}} max\{0, r(i',k)\}\right\}, \\ a(k,k) &= \sum_{i' \ s.t. \ i' \neq k} max\{0, r(i',k)\}\,. \end{aligned} \tag{2.43}$$

Akl *et al.* compute the similarity $s(i,j)$ between the gestures $\mathbf{G}_i = \left\{\mathbf{a^i}_x, \mathbf{a^i}_y, \mathbf{a^i}_z\right\}$ and $\mathbf{G}_j = \left\{\mathbf{a^j}_x, \mathbf{a^j}_y, \mathbf{a^j}_z\right\}$ as the negative squared norm of the vector whose components are the DTW distances between the respective acceleration signals components:

$$s(i,j) = -(DTW(\mathbf{a^i}_x, \mathbf{a^j}_x)^2 + DTW(\mathbf{a^i}_y, \mathbf{a^j}_y)^2 + DTW(\mathbf{a^i}_z, \mathbf{a^j}_z)^2). \tag{2.44}$$

In the user-dependent case, a number of exemplars per gesture class is chosen among the candidates obtained thanks to AP, and classification is carried out on a nearest-neighbour basis. In the-user independent case, an additional step of Compressive Sensing is added, reconstructing the signal from the exemplars with a high sparsity constraint. The label of the gesture class which is the most represented in the reconstruction is selected as the final classification.

Zhou *et al.* [ZS09][ZDlT12] propose two generalisations of the DTW similarity metric, namely Canonical Time Warping (CTW) and Generalised Time Warping (GTW). A reformulation of the DTW computation is proposed: given two samples $\mathbf{X}$ and $\mathbf{Y}$ of respective sizes n_x and n_y; given the matrices $\mathbf{W_x} \in \{0,1\}^{m \times n_x}$ and $\mathbf{W_y} \in \{0,1\}^{m \times n_y}$

needing to be inferred to align $\mathbf{X}$ and $\mathbf{Y}$, and $\|.\|_F$ the Frobenius norm, the DTW can be expressed as:

$$J_{dtw}(\mathbf{W_x}, \mathbf{W_y}) = \left\|\mathbf{X}\mathbf{W_x}^\top - \mathbf{Y}\mathbf{W_y}^\top\right\|_F^2. \tag{2.45}$$

CTW adds a linear transformation $(\mathbf{V_x}^\top, \mathbf{V_y}^\top)$ to this least-squares form, allowing alignments between signals with different dimensionality. The final computation J_{ctw} is defined as:

$$J_{ctw}(\mathbf{W_x}, \mathbf{W_y}, \mathbf{V_x}, \mathbf{V_y}) = \left\|\mathbf{V_x}^\top\mathbf{X}\mathbf{W_x}^\top - \mathbf{V_y}^\top\mathbf{Y}\mathbf{W_y}^\top\right\|_F^2. \tag{2.46}$$

Optimising J_{ctw} is performed by alternating between solving for $\mathbf{W_x}$, $\mathbf{W_y}$ with DTW; and $\mathbf{V_x}$, $\mathbf{V_y}$ using Canonical Correlation Analysis.

Zhou *et al.* [ZDlT12] also propose GTW for multi-modal alignment of human motion. Based on multi-set canonical correlation analysis, GTW extends DTW by incorporating a more flexible temporal warping parametrised by a set of monotonic basis functions.

In the next section, we present the gesture recognition approaches based on Principal Component Analysis.

2.3.2.2 Principal Component Analysis

Principal Component Analysis (PCA) is an eigenvector-based multivariate analysis performed by eigenvalue decomposition of the data covariance matrix or singular value decomposition of the data matrix.

Given the N training samples $\{\mathbf{G_k}, k \in [1; N]\}$ available, the covariance matrix $\mathbf{C}$ is defined as:

$$\mathbf{C} = \frac{1}{n-1}\sum_{k=1}^{N} \mathbf{G_k}^\top.\mathbf{G_k}. \tag{2.47}$$

As a symmetric matrix, $\mathbf{C}$ can be orthogonally diagonalised in a basis of principal directions $\mathbf{Q}$, with their respective eigenvalues on the diagonal matrix $\mathbf{L}$ i.e.

$$\mathbf{C} = \mathbf{Q}^\top\mathbf{L}\mathbf{Q}. \tag{2.48}$$

In the case of singular value decomposition, we can write $\mathbf{G}$, the matrix whose columns are equal to the samples $\mathbf{G_k}$, as the product of a matrix $\mathbf{U}$ of left singular vectors, a diagonal matrix $\mathbf{S}$, and $\mathbf{V}$ the matrix of right singular vectors, i.e.

$$\mathbf{G} = \mathbf{U}\mathbf{S}\mathbf{V}^\top. \tag{2.49}$$

Then, $\mathbf{C}$ can be rewritten as:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{V}\mathbf{S}\mathbf{U}^\top\mathbf{U}\mathbf{S}\mathbf{V}^\top = \frac{1}{n-1}\mathbf{V}\mathbf{S^2}\mathbf{V}^\top. \tag{2.50}$$

Thus, the principal components are equal to the right singular vectors, and the eigenvalues of the covariance matrix can be computed as $\lambda_i = \frac{s_i^2}{n-1}$.

It is then possible to use these eigenvalues and eigenvectors, which best reflect the variance of the data, to minimize the reconstruction error of the data on a lower-dimensional space for dimensionality reduction for example.

Marasovic *et al.* [MP11] limit their study of 7 gestures to the x and y accelerometer axes as the gestures are supposed to be performed in that plane. They apply PCA to a set of 37 features as a dimension reduction step, keeping 80% of the total variance.

Tests are carried out on a limited dataset of one user, and 6 repetitions per gesture. 50% of the data is used for training, and 50% is used for testing, which allows for 20 different test configuration, and 60 test examples per gesture, for a global recognition rate of around 86% for the three 1,3 and 5-NN classifiers.

Mantyjarvi *et al.* [MHS01] also use PCA as feature extraction to whiten the data, and decorrelate every axis. The projected data is then normalised to a zero-mean and a unit variance. Finally, a sliding window of length 256 points and shift of 64 points is applied on the data in order to perform a wavelet transform. These features are then submitted to a specific classifier, here a MultiLayer Perceptron (MLP).

Yang *et al.* [YWC08] operate a Feature Subset Selection for dimension reduction in an activity recognition application, using the Common PCA (CPCA), generalisation of PCA for multiple sets. CPCA assumes that the eigenvectors are shared between the sets while the eigenvalues may vary depending on the set. Eight features are computed for each accelerometer axis over windows of the acceleration data, namely the mean, the correlation between axes, the energy, the interquartile range, the mean absolute deviation, the root mean square, standard deviation and variance. Thus, one feature subset is created for each class. Common Principal Components (CPCs) are obtained by performing a PCA on each set, selecting the minimum common number of Principal Components (PC) so that their cumulative contribution exceeds a certain threshold for each subset, before performing a second PCA on the set formed by all the identified PCs. These CPCs are then mapped to a higher dimensional space using a Gaussian Kernel, where they are clustered. The final selected features are the ones from the gesture data closest to the cluster centroids.

In the following, another method based on eigenvalues, called the Linear Discriminant Analysis, is developed.

2.3.2.3 Linear Discriminant Analysis

Contrary to the PCA, which is an unsupervised method, the Linear Discriminant Analysis (LDA) considers the labels assigned to the samples to produce a set of directions which best separate the different classes, minimizing intra-class distances and maximizing between-class distances. This process is well-known under the name "Fisher's Linear Discriminant". Classes are then considered to follow Gaussian distributions with the same covariance matrices. Two scatter matrices are defined: $\mathbf{S_W}$, the within-class scatter matrix; and $\mathbf{S_B}$, the between-class scatter matrix. Given N classes $\{C_i, i \in [1; N]\}$, with samples $\left\{x_j^i, i \in [1; N], j \in [1; N_i]\right\}$, their respective means μ_i, and the global mean μ, then:

$$\mathbf{S_W} = \sum_{i=1}^{N} \sum_{j=1}^{N^i} (\mathbf{x_j^i} - \mu_i)(\mathbf{x_j^i} - \mu_i)^\top, \tag{2.51}$$

and

$$\mathbf{S_B} = \sum_{i=1}^{N} N_i (\mu_i - \mu)(\mu_i - \mu)^\top. \tag{2.52}$$

LDA consists in finding a projection matrix $\mathbf{W}$ that maximises the Fisher's discriminant:

$$S(W) = \frac{\det(\mathbf{W}^\top.\mathbf{S_B}.\mathbf{W})}{\det(\mathbf{W}^\top.\mathbf{S_W}.\mathbf{W})}. \tag{2.53}$$

It is important to note that the rank of $\mathbf{S_B}$ is equal to $N-1$. As a consequence, only $N-1$ of its eigenvalues are non-zero, and the matrix $\mathbf{W}$ is a $d \times (N-1)$ matrix, with d the dimension of the data.

The solution to this problem is the matrix whose columns are the eigenvectors of $\mathbf{S_W^{-1}S_B}$, and maximum linear separation is achieved along the eigenvectors associated to the highest eigenvalues.

Lukowicz [LWJ+04] combine the HMM and LDA classifications to improve gesture classification. In their study presented above (2.3.2.2), Marasovic *et al.* [MP11] compare PCA to LDA associated with a K-NN classifier on their set of 37-feature gesture samples, showing a similar score for both methods (87.6% accuracy for LDA, 86.7% accuracy for PCA).

Geometric-based approaches have the advantage to detect similarities between different signals with different reference frames and durations more easily. They do not require any intensive parametrisation, and permit to approach the actual class distribution as the number of exemplars increases. However, the comprehensive comparisons to the exemplars may render these methods unusable in a context where a high number of user has to be considered.

That is why the next approach is based on specific trained classifiers, which do not need any comparison in order to assign a class label to a signal.

2.3.3 Classifier-based methods

Instead of identifying a rule of similarity over exemplars in order to assign a label to a sample, other methods aim at extrapolating directly that label from the data with a trained decision rule. In the following are presented the SVM and Adaboost classifiers, with neural network-based methods addressed in a more detailed manner in Section 2.3.4.

2.3.3.1 Support Vector Machines

Support Vector Machines (SVM) were introduced by Boser *et al.* [BGV92]. This algorithm tries to maximize the margin between training patterns and a decision boundary in a binary classification problem. For a predefined function ϕ projecting vectors $\mathbf{x}$ in ϕ-space (kernel trick), $\mathbf{w}$ and a bias b, the hyperplan $D(\mathbf{x})$ is defined as,

$$D(\mathbf{x}) = \mathbf{w}.\phi(\mathbf{x}) + b. \tag{2.54}$$

The distance of a pattern $\mathbf{x}$ to this hyperplan is equal to $\frac{D(\mathbf{x})}{\|\mathbf{w}\|}$. Given training samples $\{(\mathbf{x}_k, y_k), k \in [1;p]\}$ with $y_k = 1$ if $\mathbf{x_k} \in class\ A$, and $y_k = -1$ if $\mathbf{x_k} \in class\ B$, then, considering the margin M between class boundaries and the training pattern exists,

$$\forall k, \frac{y_k D(\mathbf{x}_k)}{\|\mathbf{w}\|} \geq M. \tag{2.55}$$

With the fixed scale $M\|\mathbf{w}\| = 1$, the margin problem consists in solving the quadratic problem exposed in the equation 2.56,

$$\min_{\mathbf{w},b} \|\mathbf{w}\|^2, s.t.\ \forall k,\ y_k D(\mathbf{x}_k) = 1. \tag{2.56}$$

This problem can be resolved using the Lagrangian in the dual space. Given the kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x_i}).\phi(\mathbf{x_j})$, and the fact that $\mathbf{w}$ can be expressed as a linear combination of the training vectors:

$$\mathbf{w} = \sum_{i=1}^{p} \alpha_i y_i \mathbf{x_i}, \tag{2.57}$$

then the following Lagrangian has to be minimised

$$\begin{aligned} &J(\alpha, b) = \sum_{k=1}^{p} \alpha_k (1 - b y_k) - \frac{1}{2} \alpha^\top . \mathbf{H} . \alpha \\ &subject\ to\ \alpha_k \geq 0, \forall k \in [1; p]\,, \end{aligned} \tag{2.58}$$

where $\mathbf{H}$ is a $p \times p$ matrix so that $H_{kl} = y_k y_l K(\mathbf{x}_k, \mathbf{x}_l)$. The choice of the bias b requires an additional strategy, where it can be fixed a-priori, or optimised too.

Classification with SVMs

This algorithm can be relaxed in the case where no hyperplane is able to separate the data. the Soft Margin method [CV95] relies on additional non-negative variables ξ_k, which measure the errors in the separation. Given a constant C, the problem 2.56 is then modified accordingly:

$$\begin{aligned} &\min_{\mathbf{w}, b} \|\mathbf{w}\|^2 + C \sum_{k=1}^{p} \xi_k, \\ &s.t.\ \forall k,\ y_k D(\mathbf{x}_k) \geq 1 - \xi_k,\ and\ \xi_k \geq 0. \end{aligned} \tag{2.59}$$

Classification for SVMs can be operated in two ways for multi-class problems. Let N the number of classes. The one-versus-all approach takes one class as the positive class, all the others as negatives, for a total of N classifiers. The final decision is based on the maximum margin over all the classifiers. The one-versus-one approach computes the decisions of the $\frac{N(N-1)}{2}$ possible pairs of classifiers, and selects the class which has been attributed the highest number of votes.

The SVM for gesture recognition

He *et al.* [HJZH08] compare the performance of SVMs on different sets of features extracted from the accelerometer signals: DCT coefficients; FFT coefficients; and a combination of Wavelet Packet Decomposition (WPD), whose low frequency coefficients extract the primary information of the accelerometer signals while removing the high frequency noise, and FFT. Tests are carried out on a five-cross-validation over a dataset of 17 different gestures performed once by 67 subjects. With a classification based on the "Max-Wins" strategy (one-versus-one case), the features based on DCT show an average accuracy of 85.16%, FFT 86.92%, and FFT + WPD 87.36%. These score show that SVMs are suited for gesture data, and that most of the information is actually contained in the low frequency domain.

Wu *et al.* [WPZ+09] extract statistical descriptors (mean, energy, entropy, standard deviation, correlation) over a sliding frame for each gesture. Both approaches for multi-class SVM classification are studied and compared on a dataset of 12 gestures repeated 28 times by 10 individuals. The user-dependent case, tested with a 4-fold cross validation over each user's data, show an average accuracy of 99.38%; while the user-independent case, tested with a leave-one-out cross validation, reaches an accuracy of 95.21%.

Finally, Cho *et al.* [CCB+06] enhance their Bayesian Network-based 96.3% average recognition rate to 96.9% thanks to confusion pair discrimination, in particular between the gestures "0" and "6", using an SVM.

Thus, the SVM tries to determine frontiers between classes based on the most relevant samples from the dataset, in order to form a single, unified decision process. On the contrary, Adaboost, described in the following section, relies on multiple "weak" classifiers, unusable by themselves, so as to build a strong super-classifier based on the fusion of multiple decisions.

2.3.3.2 Adaboost-based classifiers

Adaboost, meaning "Adaptative Boosting", combines the decisions of multiple "weak learners", whose performance is only needed to be better than a random classifier. The classical Adaboost algorithm is applied to binary problems, with one positive and one negative class. Thus, every sample in the training set $\mathbf{X} = \{x_i | i \in [1; n]\}$ from the training set is assigned an integer $y_i \in \{-1, +1\}$.

Adaboost is an iterative algorithm which specifically targets the misclassified samples from the previous iteration. Given the iterations $1, \ldots, T$, Adaboost relies on a distribution $D_t = \{D_t(i) | i \in [1; n]\}$ over the training samples, initialised with a uniform distribution, updated to give more importance to the misclassified samples. Let h_t be the weak classifier selected at the t^{th} iteration, minimizing the error:

$$\epsilon_t = \sum_{i \ s.t. \ h_t(x_i) \neq y_i} D_t(i). \tag{2.60}$$

If the accuracy error is higher than 50%, it is possible to reverse the weak learner classification condition.

The coefficient α_t assigned to that weak classifier is defined as:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right), \tag{2.61}$$

which, given the normalisation factor Z_t, allows the distribution update as follows:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}. \tag{2.62}$$

The final classification, or hypothesis, $H(x)$ is operated on a maximum voting over the iterated weak learners, i.e.

$$H(x) = sign(\sum_{t=1}^{\top} \alpha_t h_t(x)). \tag{2.63}$$

This algorithm can be derived from an iterative classifier ranking, adding one classifier at each iteration. Given the linear combination of classifiers at the t^{th} iteration, we define C_{t-1} as follows:

$$C_{t-1}(x) = \sum_{j=1}^{t-1} \alpha_j h_j(x), \tag{2.64}$$

and the total error E_t of the extended classifier corresponds to the exponential loss

$$E_t = \sum_{i=1}^{n} \exp^{-y_i (C_{t-1}(x_i) + \alpha_t h_t(x_i))}. \tag{2.65}$$

Adaboost for gesture recognition

Hoffman *et al.* [HVL10] apply the pairwise Adaboost algorithm in order to classify 25 gestures recorded with the accelerometer-based Nintendo Wiimote device, and its gyrometer-based WiiMotion Plus attachment. 41 features inspired by 2D gestures drawn on screens are extracted from the data. The set of weak learners consists in a minimum distance criterion to the D_t-weighted averages of the training samples of the considered pair. Tests performed on 13 gestures made by 17 participants show that Adaboost is outperformed by a linear classifier, with respective accuracies of over 99% and 95% in the user-dependent case; and 98% and 93% in the user-independent case.

In the following section, neural network-based classifiers are developed more in-depth, as the theoretical background for our study.

2.3.4 Neural network-based methods

Designed to mimic our understanding of biological neuron processes, an artificial neural network takes advantage of connections between simple processing units called "artificial neurons" in order to extrapolate a more complex function.

Two main types of artificial neural networks can be identified. The feed-forward networks take the whole samples as an input, and activate neurons of successive layers. Opposite to those networks, the recurrent neural networks compute time-series, with recurrent connections that simulate a memory of previous states.

Two classical feed-forward networks and one recurrent network are presented, respectively the MultiLayer Perceptron and the Convolution Neural Network; and the Bi-directional Long Short-Term Memory Recurrent Neural Network.

2.3.4.1 MultiLayer Perceptron

A MultiLayer Perceptron (MLP) is a feed-forward network composed of multiple computational neural layers whose behavior mirrors our understanding of brain neurons. The artificial neuron (see Fig. 2.12) is a mathematical function which acts in two steps. First, the stimuli from the input connections, symbolizing the biological neuron dendrites, are integrated. For the j^{th} neuron of the L^{th} layer N_j^L, receiving the activations $a_i^{L-1}, i \in [1; n]$ weighted by the connection weights ω_{ij}^L and a bias ω_{bj}^L, whose activation is constant and equal to 1, the integrated input net_j^L is defined as:

$$net_j^L = \sum_{i=1}^{n} \omega_{ij}^L a_i^{L-1} + \omega_{bj}^L. \tag{2.66}$$

This input is then processed by the activation function φ, which will determine the output signal $a_i^L = \varphi(I_i^L)$, also called *activation*, which mirrors the electrical response of the real neuron. The most common activation may be linear, proportional to the input value; or non-linear. In that case, the sigmoid is historically used:

$$sigmoid(x) = \frac{1}{1 + \exp(-x)}. \tag{2.67}$$

However, as this function does not allow for negative values, it may be replaced by the hyperbolic tangent:

$$tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}. \tag{2.68}$$

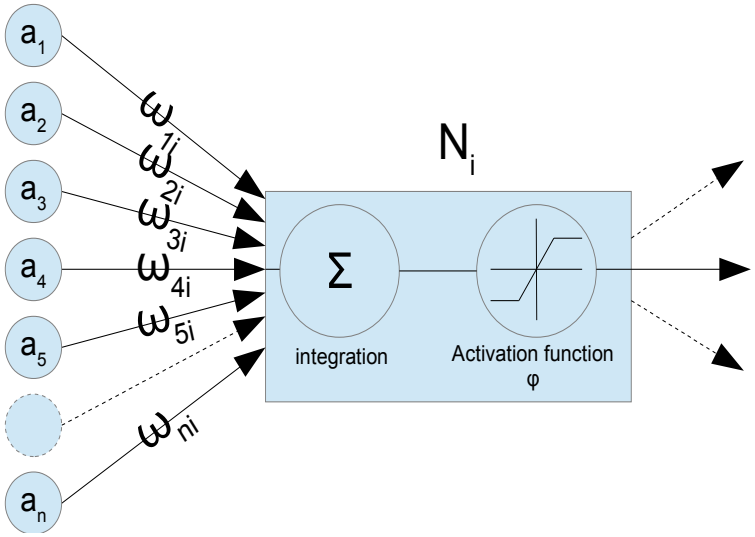


Figure 2.12 – Decomposition of the successive computations operated by an artificial neuron.

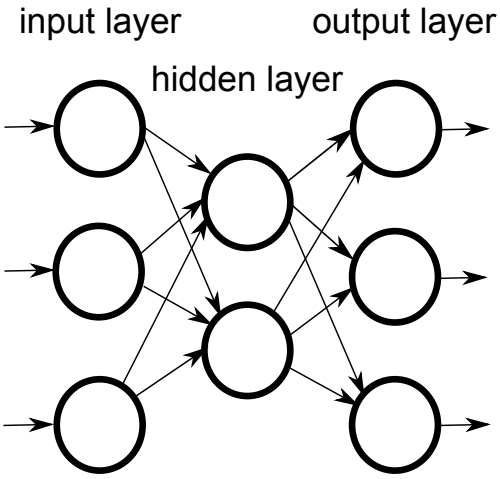


Figure 2.13 – Representation of a MLP with one hidden layer, and the forward connections between neurons.

The MLP architecture is comprised of multiple layers (see Fig.2.13), with connections between the neurons of two successive layers. Thus, the information only flows forward, hence the name "feed-forward neural network". The MLP is formed of three types of layers. The input layer L_{in} is directly activated by the features of the sample $\mathbf{x} = \{x_1, \ldots, x_n\}$ to be recognised, with one artificial neuron for each dimension:

$$\forall i \in [1; n], a_i^{L_{in}} = x_i. \tag{2.69}$$

The hidden layers hold the computational power of the network. The consecutive hidden layers perform successive operations on the propagated information, and have a high potential in mirroring any function. Finally, the output layer, formed of one neuron for each training class, performs the classification. The output neuron with the strongest activation determines the winning class for that sample.

MLP Training: the BackPropagation algorithm

The MLP training is performed using the backpropagation algorithm. Following a gradient descent logic, for each training sample, the network is activated, then the discrepancy between the activations of the output layer neurons and the target output is computed. This discrepancy is usually computed with either of two functions: the squared error or the cross-entropy.

The squared error objective aims at reducing the squared norm between the outputs and their corresponding targets, i.e. 1 for the output of the neuron associated to the known label of the input sample, and 0 (or -1) for the other neuron outputs. Given K the number of classes; and the target t_{kl} for the output neuron k when the network has been activated by the sample $\mathbf{x_n}$, the least-square error $E_W^{squared}$ over the training dataset, with $\mathbf{W} = \{\omega_{ji}\}$, is defined as:

$$E_W^{squared} = \sum_{n=1}^{N} \sum_{k=1}^{K} (t_{kn} - y_{kn})^2 \tag{2.70}$$

The second main error criterion is based on the cross-entropy between the estimate and the target distributions for the model. The name of this function comes from the Kullback-Leibler cross-entropy method for measuring the number of bits needed to identify an event from a set of probabilities. In this case, the event is the actual label of the gesture. E_W^{cross} is defined as:

$$E_W^{cross} = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{kn} log(y_{kn}). \tag{2.71}$$

The cross-entropy function is almost always accompanied with a change of the activation function for the output layer. Every output neuron activation is transformed so as to represent a probability distribution thanks to the "softmax" function, where the activations a_i^O of the output layer are equal to:

$$a_i^O = \frac{\exp(net_i^O)}{\sum_j \exp(net_j^O)}. \tag{2.72}$$

This leads to a very simple error computation formula, developed in Appendix A.1.

The only parameters to be updated during training are the connection weights $\mathbf{W}$, and the biases $\mathbf{B}$. The most famous technique for MLP training, called the "Backpropagation" algorithm, relies on gradient descent on the error function. Thus, for the epoch t, given the learning rate λ^t, the weights are corrected following the formula

$$\mathbf{W_{t+1}} = \mathbf{W_t} - \lambda^t \frac{\partial E}{\partial \mathbf{W}}(\mathbf{W_t}) \tag{2.73}$$

In particular, for each ω_{ij}, at the t^{th} epoch, we have to compute $\frac{\partial E}{\partial \omega_{ij}}(\omega_{ij}^t)$ in order to update ω_{ij}^t. Thanks to the chain rule, and $\delta_j = \frac{\partial E}{\partial net_j}$ for a neuron n_j, we can rewrite the equation 2.73 as follows:

$$\frac{\partial E}{\partial \omega_{ij}}(\omega_{ij}^t) = \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial \omega_{ij}} = \delta_j a_i. \tag{2.74}$$

The delta values for the output layer O can be obtained through an immediate calculation:

$$\begin{aligned}\delta_j^O &= \frac{\partial E}{\partial net_j^O} = \frac{\partial E}{\partial a_j^O}\frac{\partial a_j^O}{\partial net_j^O} \\ &= \varphi'(net_j^O)\frac{\partial E}{\partial a_j^O}.\end{aligned} \tag{2.75}$$

Still using the chain rule for any hidden layer L, the delta values δ_i^L can be expressed as a function of the deltas of the next layer. Thus, while the signal only flows forward during activation, the error is propagated backwards in the network during training in order to update each connection weight, following the "delta rule":

$$\begin{aligned}\delta_i^L &= \sum_j \frac{\partial E}{\partial net_j^{L+1}}\frac{\partial net_j^{L+1}}{\partial a_i^L}\frac{\partial a_i^L}{\partial net_i^L} \\ &= \varphi'(net_i^L)\sum_j \omega_{ij}\delta_j^{L+1}.\end{aligned} \tag{2.76}$$

Algorithm 1 presents a summary of the MLP BackPropagation algorithm.

Algorithm 1 MLP BackPropagation algorithm : stochastic gradient descent

procedure FEED-FORWARD THE TRAINING SAMPLE $\mathbf{X} = x_1, \ldots, x_n$
  Initialize the input layer $\forall i \in [1; n]\,, a_i^{L_{in}} = x_i$
  for all other layers L of the network **do**
    for all neurons n_{Li} in L **do**
      $net_j^L = \sum_{i=1}^{n} \omega_{ij}^{L,t} a_i^{L-1} + \omega_{b,j}$
      $a_j^L = \varphi(net_j^L)$
    end for
  end for
end procedure
procedure COMPUTE THE DELTA VALUES FOR THE OUTPUT LAYER O
  for all neurons n_{Oj} in O **do**
    $\delta_{Oj}^p = \varphi'(net_{Oj}^p).\frac{\partial E}{\partial a_{Oj}^p}$
  end for
end procedure
procedure APPLY THE DELTA RULE
  for all hidden layers L **do**
    for all neurons n_i^L in L **do**
      $\delta_i^L = \varphi'(net_i^L).\sum_j \omega_{ij}.\delta_j^{L+1}$
    end for
  end for
end procedure
procedure UPDATE THE WEIGHTS AND BIASES
  for all layers L of the network other than the input layer **do**
    for all neurons n_i^L in L **do**
      $\omega_{ij}^{L,t+1} = \omega_{ij}^{L,t} - \lambda^t a_i^{L-1}.\delta_j^L$
    end for
  end for
end procedure

The MLP for gesture recognition

As seen above, Mantyjarvi *et al.* [MHS01] extract 24 features from the normalised signals whitened with a Principal Component Analysis to train a 3-layer MLP. The data was created using a pair of accelerometers attached to the users' belt. 6 participants carry out one of four activities such as walking up or down the stairs. A cross-validation based on the leave-one-out strategy, with best accuracies reaching 83-90%.

Niezen *et al.* [NH09] compare HMM, DTW and artificial neural networks performances based on an 8-gesture dataset with 10 repetitions per gesture. The test protocols may be lacking on the fact that they rely on a leave-one-out strategy, where only one sample is used for testing, while the other 79 are used for training. Moreover, no details on the number of users was given. The final results show an accuracy of 90% for the MLP, compared to 96.25% for DTW and HMM strategies.

Finally, Yang *et al.* [YWC08] adopt the MultiLayer Feed Forward neural network as an activity classifier, comprising 8 common activities such as standing, sitting, walking, running, vacuuming, scrubbing, brushing teet and working at computer. Training was performed using the Resilient BackPropagation algorithm, which only takes into account the sign of the gradient. Features are extracted from the original data in order to recognize static and dynamic gestures, which are handled separately. The MLP is trained using the features explained above (cf. paragraph 2.3.2.2). Tests were performed following the one-subject-out strategy, on the data from 7 subjects repeating each activity 45 times. An average recognition rate of 94.52% was reached for 9 selected features, and 94.64% for the whole 24 features for the dynamic classifier.

One of the restriction of the MLP classifier consists in its lack of tools for handling a second dimension, whether for images, or for time-series of feature vectors, which required a resampling or a specific feature extraction in all the studies above. That is why the Convolutional Neural Network (CNN), originally designed to process images, was proposed.

2.3.4.2 Convolutional Neural Networks

Inspired by studies on the visual cortex, the Convolutional Neural Network (CNN) is based on a mathematical model of *receptive fields*, where cells are sensitive to small, over-lapping subregions of the visual fields.

The typical CNN consists of multiple stacks of convolutional and sub-sampling layers 2.14, completed with fully connected layers which are activated and operate the classification in the same manner as the MLP.

A convolutional layer processes the correlation between multiple filters/kernels and sub-regions of the input field. In the case of a 2D input of dimensions $n \times m$ convoluted with a $u_i \times v, u \leq n, v \leq m$ sliding filter $\mathbf{f_i}$, the resulting convolution map dimension may be equal to $(n-1) \times (m-1)$, or preserve the original dimension thanks to a preprocessing step of zero padding at the frontiers of the input. One convolution map is produced for every kernel. Applying a sliding filter ensures a global spatial independence for the detected features.

Given the input $\mathbf{I} = \{I(i,j), i \in [1;n], j \in [1;m]\}$, the filter $\mathbf{f} = \{\omega(k,l), k \in [1;u], l \in [1;v]\}$ and its bias b, the convolutional map $C = \{c(i,j)\}$ is computed

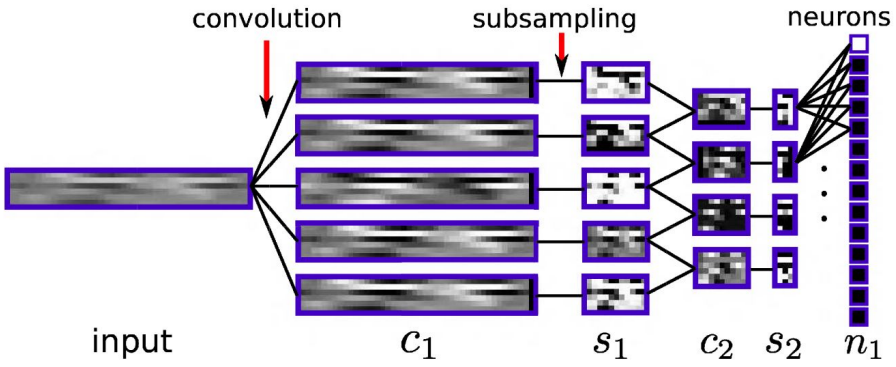


Figure 2.14 – Representation of the architecture and activation maps of a Convolutional Neural Network. Extracted from [DBLG14]. Convolutional layers c_i are followed by sub-sampling layers s_i, which allows a progressive dimensionality reduction and feature computation for the last output layer operating the classification.

according to the formula:

$$c(i,j) = \phi_c \left(\sum_{\substack{k\in[1;u] \\ l\in[1;v]}} \omega_{k,l} I(i+k, j+l) + b \right). \tag{2.77}$$

The convolutional layer is then followed by a sub-sampling layer, which reduces the dimension of the input. Sub-sampling ensures a local independence to small changes, and can generally be performed in two ways, averaging or max-pooling. The averaging map s_a is produced thanks to a $p_a \times q_a$ kernel, with weight and bias ω_a and b_a, which computes a ϕ_a squashed weighted average over sliding, non-overlapping windows. s_a is defined as:

$$s_a(x,y) = \phi_a \left(\omega_a \sum_{\substack{k\in[1;p_a] \\ l\in[1;q_a]}} I_{conv}(xp_a + k, yq_a + l) + b_a \right). \tag{2.78}$$

The max-pooling layer creates a new map based on the ϕ_m-squashed value of the maximum activation in non-overlapping $p_m \times q_m$ regions of the input:

$$s_a(x,y) = \phi_m \left(\max_{\substack{k\in[1;p_m] \\ l\in[1;q_m]}} I_{conv}(xp_m + k, yq_m + l) \right). \tag{2.79}$$

CNN training

CNN training is performed usually using an adapted version of the BackPropagation algorithm.

In the convolutional layers, the delta-rule can be modified in order to take into account the successive convolutions performed for each filter. Given ϕ the activation function of the layer L, $a^L_{k,l} = \phi(net_{k,l})$, $p \times q$ the dimensions of the layer $L-1$, $u \times v$ the dimensions of the layer L, and $\delta^L_{kl} = \frac{\partial E}{\partial net^L_{k,l}}$, then we can write

$$\frac{\partial E}{\partial \omega_{i,j}} = \sum_{\substack{k\in[0;p-u] \\ l\in[0;q-v]}} \frac{\partial E}{\partial net^L_{k,l}} \frac{\partial net^L_{k,l}}{\partial \omega_{k,l}} = \sum_{\substack{k\in[0;p-u] \\ l\in[0;q-v]}} \delta^L_{kl} \phi'(net^L_{k,l}) a^{L-1}_{i+k,j+l}. \tag{2.80}$$

Moreover, after a zero-padding on the top and left sides of the input,

$$\begin{aligned}\delta_{ij}^{L-1} = \frac{\partial E}{\partial net_{i,j}^{L-1}} &= \sum_{\substack{k\in[0;u-1]\\ l\in[0;v-1]}} \frac{\partial E}{\partial a_{i-k,j-l}^{L}} \frac{\partial a_{i-k,j-l}^{L}}{\partial net_{i,j}^{L-1}} \\ &= \sum_{\substack{k\in[0;u-1]\\ l\in[0;v-1]}} \delta_{i-k,j-l}^{L} \omega_{k,l}.\end{aligned} \tag{2.81}$$

The CNN for gesture recognition

Duffner *et al.* [DBLG14] devised a CNN composed of alternated convolution and sub-sampling layers in order to process temporally normalised gestures, which are classified with a fully-connected softmax layer. The sensors data are treated as 6×45 images, with the concatenated 3D measurements of a Smartphone accelerometer and gyrometer over 45 time-samples. This "images" are successively analysed using temporal, feature, and combined convolutions, with respective filters 3×1, 1×3 and 3×3. Experiments are carried out over two datasets of 14 symbolic gestures. The first dataset targets the single-user case, with 40 samples by gesture, while the second dataset comprises 22 users, with 5 samples per user. Temporal convolutions prove to be the most suited in each of the four test configurations, with a 96.5% accuracy in the single-user case, 93.7% in the multi-user closed-world case, 91.5% in the multi-user open-world, and 73.5% for the most challenging configuration where the network is trained from the samples of a single user, and is tested on the other users data.

Although the CNN is able to consider the temporal dimension of the gesture, which is used as a 2D image, another family of neural networks, called "recurrent neural networks", is specifically designed to process temporal series, such as the Bi-directional Long Short-Term memory neural network.

2.3.4.3 Bi-directional Long Short Term Memory

The Long-Short Term Memory (LSTM) Recurrent Neural Network was introduced by Hochreiter and Schmidhuber [HS97]. Recurrent Neural Networks (RNN) are a specific family of neural networks which can handle time-series thanks to recurrent connections in the hidden layers. These connections allow for a short-term memory, where the state of a neuron at time t is influenced by its previous state at time $t+1$. However, this recurrence introduces the problem of the exponential vanishing gradient during the BackPropagation through time (BPTT). As a solution, Hochreiter proposes a new architecture based on "LSTM cells", later improved to its current standard by Graves *et al.* [GS05]. The LSTM block formed of a Constant Error Carousel (CEC), with input, output and forget gates, and a peephole connection (see Fig. 2.15).

The CEC shows a constant error flow: given the error $e(t)$ at time t, the linear activation function f and the recurrent connection weight ω, the following relation is respected:

$$e(t) = \frac{\partial f}{\partial \omega}(net(t)).\omega(t+1) = 1. \tag{2.82}$$

Thus, the CEC stores the error, while the input, output and forget gates respectively control the "write", "read" and "reset" operations on this error through multiplications.

The LSTM network is composed of one input and one output layers, while the hidden layer may be formed of LSTM cells as well as "conventional" hidden units providing inputs to gate units and memory cells. During an activation of a memory

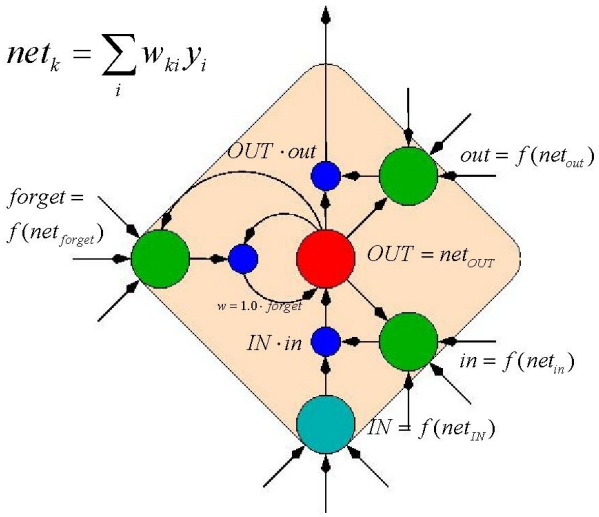


Figure 2.15 – Extracted from [Sch]. Representation of the LSTM cell. The CEC is marked in red, while the output, input and forget gates are in green. The peephole connection corresponds to the connection between the CEC, whose outputs are delayed by one time step, and the gates. The blue nodes show the multiplications operated by the gates.

cell c_j, the net input net_{c_j}, the internal state s_{c_j}, and the output activation a_{c_j} are computed as:

$$\begin{aligned} net_{c_j}(t) &= \sum_i \omega_{ic_j} a^i(t-1), \\ s_{c_j}(t) &= s_{c_j}(t-1) + a^{in}(t) f(net_{c_j}(t)), \\ a_{c_j}(t) &= a^{in}(t) f(s_{c_j}(t)). \end{aligned} \tag{2.83}$$

Learning is performed thanks to any the "BackPropagation Through Time" (BPTT) algorithm.

The LSTM network classification can be operated on a majority-voting over every activation of the output neurons for every time step.

The Bi-directional LSTM (BLSTM) is a variant of the LSTM network. Two networks are trained in parallel, the "forward" and "backward" networks. This allows for a consideration of the past and the future of a time-series sample at every time step. A fusion of both networks outputs is operated with an output layer whose size is equal to the number of classes, as for the MLP. Thus, the BLSTM offers one classification for each time step, and a max-voting operation on these decisions determines the final class.

The BLSTM for gesture recognition

Lefebvre *et al.* [LBMG15] introduce the same dataset used later by Duffner *et al.* [DBLG14] to test the BLSTM performance on a 14 symbolic gestures problem, with 22 users. The inertial data is preprocessed and used directly as the training features. The study shows that the BLSTM outperforms state-of-the-art methods such as DTW, FDSVM or cHMM in the multi-user configurations, with an accuracy of 95.57% for a closed world problem where every user is represented in the training dataset; and 92.57% in an open world configuration, where tests are carried out on the samples of 5 users completely unknown to the network. BLSTM are not suited to the single-user case, mostly due to the lack of sufficient data. However, CNNs achieve a higher accuracy in every configuration in the later study by Duffner *et al.* [DBLG14].

2.4 Conclusion

Artificial Neural Networks are also well-known for their adaptation and generalisation potential, which is well complimented by a very flexible training algorithm allowing for a great variety of architectures. Neural-based approaches show state-of-the-art results in the typical gesture recognition frameworks where every class is known and no rejection is considered. While results depend highly on the right choice of parameters, such as the network size, they can then be obtained directly from preprocessed inertial signals, without the need for extensive and subjective feature extraction.

However, it is important to note that the MLP, CNN and BLSTM networks do not fulfil all the requirements of an open-world application. Indeed, they are limited in the sense that they cannot adapt to new classes, and operate in a closed-world paradigm. Any added class would require an entirely new model due to the rigid output layer architecture which can only handle a predefined number of classes. Moreover, they cannot take advantage of relationship informations and expected neighbourhoods in the feature space. As such, it is necessary to consider an intermediary solution, where the artificial neural networks advantages are captured in order to compute a higher level and more discriminative representation, which is then submitted to a simplified classification process.

Thus, in order to harness these performances while achieving a high discrimination between known and unknown gestures, we propose to learn a similarity metric based on the Siamese artificial neural network.

Chapter 3

The Siamese Neural Network (SNN)

In the previous chapter, we presented different types of neural networks applied to gesture recognition and classification. Most networks both extract features, thanks to their convolutional and max-pooling layers in the case of CNNs for example; and classify, generally with a "Softmax"-based output layer. However, in a fully supervised manner, they do so automatically, without any possibility to take into account prior knowledge about expected neighbourhoods in the feature space. Since regions of the output space are manually discretised, defined and assigned to classes, they are completely semantically decorrelated, and undefined regions do not hold any meaning.

The Siamese architecture, thanks to its semi-supervised training strategy involving multiple samples, whose relations are known, allows for a different structuring of the output space, where the meaning assigned to a region varies in a continuous manner.

In this chapter, we will first detail the different existing variants of the Siamese architecture components, before defining our contributions and their implications for each of these components.

3.1 State of the art

More generally, a SNN learns a non-linear similarity metric, and essentially differentiates itself from classical networks thanks to its specific training strategy involving sets of samples tagged as similar or dissimilar.

Although SNNs perform the same task, their capabilities essentially depend on four points: the network architecture, the training sets selection strategy, the objective function for similar and dissimilar pairs, and the training algorithm.

3.1.1 Architecture

The name "Siamese" comes from the necessity to use multiple identical, weight-sharing parallel networks.

Indeed, during the training step, the network is composed of multiple identical sub-networks, forming low-level input fields joined at their ends by an output "contrastive loss layer", whose objective function E_W is devised as a distance computation between features from each of the input samples (see Fig.3.1).

Siamese networks mainly involve two types of networks: CNNs and single or multi-layer feed-forward perceptrons.

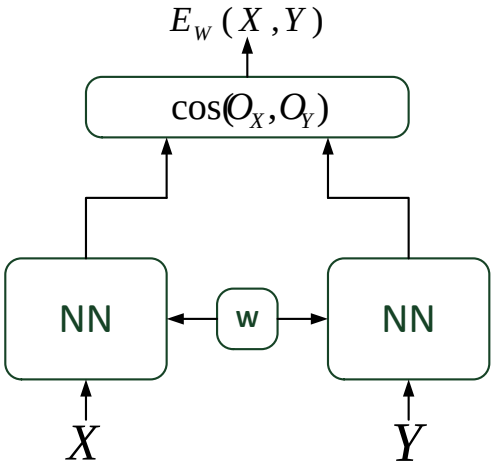


Figure 3.1 – Architecture of the original Siamese Neural Network. Two identical neural networks (NN) with shared weights W take simultaneously two input samples X and Y to compute the error relative to a cosine-based objective function, thanks to the respective outputs O_X and O_Y.

Three specific studies were crucial for the development of this model, and define the typical architecture, with two identical CNN sub-networks during training. Bromley *et al.* [BGL+94] introduce the Siamese architecture in 1994, with a signature verification system handling time-series of hand-crafted features extracted from signals captured with a LCD device. In 2005, Chopra, Hadsell and LeCun [CHL05] formalise the Siamese architecture, applying CNNs on raw images for face verification, before adapting it for a dimensionality reduction technique [HCL06].

Every study working with Siamese CNNs is employing this two-sub-network architecture. Yi *et al.* [YLLL14] suggest the "Deep Metric Learning" (DML) method for person re-identification, where every input image is divided in three parts in order to train three parallel CNN-based Siamese networks. Chen *et al.* [CS11] extract speaker-specific information with a pair of identical CNNs auto-encoders where half of the encoding layer error is regularised with a Siamese objective for every update. Finally, Sun *et al.* [SWT14] combine face identification and verification with a softmax layer added above the feature extraction layer.

The other main types of network involved in the Siamese architecture is based on feed-forward perceptrons.

The first applications were essentially linear projections, modelled as Single layer Feed-Forward Neural Networks (SFNNs) with linear activations functions. With their S2Net, Yih *et al.* [YTPM11] apply the two-sub-network SFNN Siamese architecture to compute a similarity score between texts represented by term-vectors. Likewise, Bordes *et al.* model [BWCB11] relations between entities in Knowledge Bases with one SFNN-based Siamese network for each relation, whose weights may not be shared depending on the symmetry of that relation. Masci *et al.* [MBBS14] introduce a multi-modal similarity-preserving hashing based on coupled SFNN Siamese networks.

However, other strategies are needed to compute a non-linear similarity metric. Lefebvre *et al.* [LG13] tackle face verification and suggest a MLP Siamese network based system, trained from features extracted with SOMs. Hu *et al.* [HLT14] redis-

cover this architecture using hand-crafted features for face verification, under the name *Discriminative Deep Metric Learning (DDML)*, while Nair *et al.* [NH10] pre-train their network as Regularised Boltzmann Machines (RBMs) for face verification.

Consequently, the Siamese architecture is commonly applied to non-recurrent feed-forward networks. It engages multiple weight-sharing sub-networks, which process sets of samples whose similarity is previously known. While the multiple-network architecture allows for a comparison based training, the latter depends highly on the similarities modelling.

Thus, in the following, we will expose the different strategies proposed to model these relationships.

3.1.2 Training Set Selection

As said above, during training, multiple samples are simultaneously forwarded to each sub-network, in order to devise a non-linear metric based on the respective projections of each input. The Siamese network is thus trained to project multiple samples coherently. The resulting application of the network depends on the kind of knowledge about similarities one implements. In problems such as face or signature verification [BGL+94][CHL05][LG13][SWT14][YLLL14][ZIG+15], the similarity between samples depend on their origin, and the network allows to determine the genuineness of a test sample with a binary classification. In cases involving mapping learning robust to specific transformations [HCL06], similar samples differ by slight rotations or translations. However, similarities can be more abstract concepts, such as same documents in different languages [YTPM11].

As such, the final metric is the result of the modelling of similarity relations. The most common representation consists in a binary relation based on pairs: given two samples $\mathbf{X_1}$ and $\mathbf{X_2}$, the $(\mathbf{X_1}, \mathbf{X_2})$ pair similarity is determined by a tag, which takes two different values whether the relation is similar or dissimilar.

However, knowledge about semantic similarities can take more complex forms. Lefebvre *et al.* [LG13] expand the information about expected neighbourhoods, and suggest a more symmetric representation: by considering a reference sample $\mathbf{H}$ for each known relation, it is possible to define triplets $(\mathbf{H}, \mathbf{H}+, \mathbf{H}-)$, with $\mathbf{H}+$ forming a genuine pair with the reference $\mathbf{H}$, while $\mathbf{H}-$ is the member of an impostor pair. Similarities are then represented as much as dissimilarities.

To conclude, similarity relationships may be handled differently, depending on the knowledge about the data and the final application for the learnt metric.

With these different knowledge representations presenting multiple samples to a set of weight-sharing sub-networks, it is necessary to study new objective functions in order to define how semantic relations will be reflected in the output space.

3.1.3 Objective Function

The contrastive loss layer objective function is destined to compute a similarity metric between the higher-level features extracted from multiple input patterns. Thus, this discriminative distance is trained to get smaller for similar patterns, and higher for dissimilar ones. It takes two forms, respectively bringing together and pushing away features from similar and dissimilar pair of patterns.

Two main similarity measures, imported from other common applications, are used: the cosine similarity metric, where similar samples are collinear, and the Euclidean similarity metric, where the Euclidean distance between similar samples is small. Other metrics, such as statistics-based ones, are more fringe.

3.1.3.1 Cosine-Based Objective Functions

A cosine objective function aims at learning a non-linear cosine similarity metric, whether it is expressed specifically, in the form of multiple targets, or relatively, by pair scores ranking. Given two samples $\mathbf{X_1}$ and $\mathbf{X_2}$, and the cosine of the angle between the two vectors implied by these samples $\cos(\mathbf{X_1}, \mathbf{X_2}) = \frac{\mathbf{X_1}.\mathbf{X_2}}{\|\mathbf{X_1}\|.\|\mathbf{X_2}\|}$, the cosine similarity metric is defined by:

$$cos_{sim}(\mathbf{X_1}, \mathbf{X_2}) = 1 - \cos(\mathbf{X_1}, \mathbf{X_2}). \tag{3.1}$$

Square Error Objective

One approach comes from the original use of the square error objective function for the MLP. Given a network with weights W and two samples $\mathbf{X_1}$ and $\mathbf{X_2}$, a target $t_{\mathbf{X_1X_2}}$ is defined for the cosine value between the two respective output vectors $\mathbf{O_{X_1}}$ and $\mathbf{O_{X_2}}$. In [BGL+94], Bromley *et al.* set this target to 1 if for a similar pair, and -1 otherwise. Given Y the similarity label, the error estimation E_W for any pair is thus defined as:

$$E_W(X_1, X_2, Y) = (t_{\mathbf{X_1X_2}}(Y) - \cos(\mathbf{O_{X_1}}, \mathbf{O_{X_2}}))^2. \tag{3.2}$$

Triangular Similarity Metric

Zheng *et al.* [ZIG+15] imply these same targets. Given Y the numerical label for the $(\mathbf{X_1}, \mathbf{X_2})$ pair, acting as the target $t_{\mathbf{O_{X_1}O_{X_2}}}$ and respectively equal to 1 and -1 for similar and dissimilar pairs; and $\mathbf{C}(\mathbf{X_1}, \mathbf{X_2}, \mathbf{Y}) = \mathbf{O_{X_1}} + Y.\mathbf{O_{X_2}}$ the target vector for the pair, the triangular inequality imposes:

$$\|\mathbf{O_{X_1}}\| + \|\mathbf{O_{X_2}}\| - \|\mathbf{C}\| \geq 0. \tag{3.3}$$

After adding norm constraints to prevent a degeneration towards a null projection, the final objective function becomes:

$$\begin{aligned} E_W(X_1, X_2, Y) &= \|\mathbf{O_{X_1}}\| + \|\mathbf{O_{X_2}}\| - \|\mathbf{C}(\mathbf{X_1}, \mathbf{X_2}, \mathbf{Y})\| + \frac{1}{2}(1 - \|\mathbf{X_1}\|)^2 + \frac{1}{2}(1 - \|\mathbf{X_2}\|)^2 \\ &= \frac{1}{2}\|\mathbf{O_{X_1}}\|^2 + \frac{1}{2}\|\mathbf{O_{X_2}}\|^2 - \|\mathbf{C}(\mathbf{X_1}, \mathbf{X_2}, \mathbf{Y})\| + 1. \end{aligned} \tag{3.4}$$

Triplet Similarity Objective

Lefebvre *et al.* [LG13] generalise the Square Error Objective by using simultaneously targets for genuine and impostor pairs. Samples outputs from similar classes are collinear while outputs from different classes tend to orthogonality, which translates as a target equal to 1 for similar pairs and 0 for dissimilar ones. Let $(\mathbf{R}, \mathbf{P}, \mathbf{N})$ be a triplet, with a reference sample $\mathbf{R}$, a positive sample $\mathbf{P}$ forming a similar pair with $\mathbf{R}$, and a negative sample $\mathbf{N}$, forming a dissimilar pair with $\mathbf{R}$, we get:

$$E_W(\mathbf{R}, \mathbf{P}, \mathbf{N}) = (1 - cos(\mathbf{O_R}, \mathbf{O_P}))^2 + (0 - cos(\mathbf{O_R}, \mathbf{O_N}))^2. \tag{3.5}$$

Deviance Cost Function

Inspired by the common loss functions such as square or exponential losses, Yi *et al.* [YLLL14] opt for the binomial deviance. Since their Siamese architecture does

not necessarily share weights between sub-networks, let B_1 and B_2 be the respective functions associated to both sub-networks, and $B_1(\mathbf{X_1})$ and $B_2(\mathbf{X_2})$ be the projections of the samples of a pair, we get:

$$E_W(X_1, X_2, Y) = \ln(\exp^{-2Y.cos(B_1(\mathbf{X_1}),B_2(\mathbf{X_2}))} + 1). \tag{3.6}$$

Two Pair Objective

Yih *et al.* [YTPM11] consider two pairs of term-vectors, namely $(\mathbf{f_{p1}}, \mathbf{f_{q1}})$ and $(\mathbf{f_{p2}}, \mathbf{f_{q2}})$, where the first pair is known to have a higher similarity than the second. The main objective is then to maximise the difference between their similarity scores:

$$\Delta = \cos(\mathbf{O_{f_{p1}}}, \mathbf{O_{f_{q1}}}) - \cos(\mathbf{O_{f_{p2}}}, \mathbf{O_{f_{q2}}}). \tag{3.7}$$

with the use of a scaling factor γ, destined to penalise more on the prediction errors, combined with the logistic loss:

$$E_W(\Delta) = \log(1 + \exp(-\gamma\Delta)). \tag{3.8}$$

Probability Driven Objective

Nair *et al.* [NH10] add a single neuron to their architecture whose activation function processes directly the cosine value between a pair of sample outputs. Its activation becomes the probability for two faces to have the same identity. Given the pair of samples $(\mathbf{X_1}, \mathbf{X_2})$ and w, b scalar learnable parameters:

$$Pr(\text{"}Same\text{"}) = \frac{1}{1 + \exp(-(w.\cos(\mathbf{O_{X_1}}, \mathbf{O_{X_2}}) + b))}. \tag{3.9}$$

In the next section, we will present the objective functions based on the distance implied by the Euclidean norm in the output space of the Siamese network.

3.1.3.2 Euclidean-Based Objective Functions

Chopra *et al.* [CHL05] introduce an objective metric based on the Euclidean distance, with a $(\mathbf{X_1}, \mathbf{X_2})$ pair energy E_W equal to:

$$E_W(\mathbf{X_1}, \mathbf{X_2}) = \|\mathbf{O_{X_1}} - \mathbf{O_{X_2}}\|_2 . \tag{3.10}$$

The goal of the network is to assign a lower energy for any similar pair than for a dissimilar pair. The existence of an intrinsic universal **margin** in the data structure is necessary to ensure the identification of similar examples. Given one genuine training pair $(\mathbf{X_1}, \mathbf{X_2})$ with an energy E_W^G, and one impostor training pair $(\mathbf{X_1}, \mathbf{X_2'})$ with an energy E_W^I, they express this requirement under the condition:

$$\exists m > 0 \ , E_W^G + m < E_W^I. \tag{3.11}$$

Given Q the upper bound of E_W, a binary label Y, respectively equal to zero and one for similar and dissimilar pairs, and specific partial loss functions, L_G for similar/genuine pairs, and L_I for dissimilar/impostor pairs, they define:

$$\begin{aligned} L_G(E_W) &= \frac{2}{Q}(E_W)^2, \\ L_I(E_W) &= 2Qe^{(-\frac{2.77}{Q}E_W)}, \\ L(W, Y, \mathbf{X_1}, \mathbf{X_2}) &= (1 - Y)L_G(E_W) + (Y)L_I(E_W). \end{aligned} \tag{3.12}$$

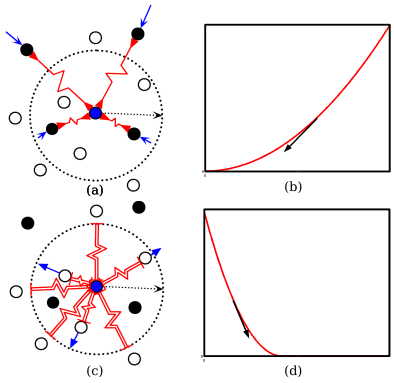


Figure 3.2 – Spring system between similar examples (solid circles) and dissimilar examples (hollow circles). (a) *Attract-only* springs (b) Loss function and gradient associated with similar pairs. (c) *Repulse-only* springs with margin m. (d) Loss function and gradient associated with dissimilar pairs. Extracted from [HCL06].

Hadsell *et al.* [HCL06] pursue this idea further. While they retain the similarity metric based on the Euclidean distance between labelled pairs of samples, they suggest a different partial hinge-loss-based function for dissimilar pairs, which can be interpreted as a spring system (see. Fig.3.2). Similar sample projections are always brought closer together, but the loss function relative to dissimilar samples is set to zero if the distance between their projections is higher than a manually defined margin m,

$$\begin{aligned} L_G(E_W) &= \frac{1}{2}(E_W)^2 \\ L_I(E_W) &= \frac{1}{2}(\max\{0, m - E_W\})^2. \end{aligned} \tag{3.13}$$

Sun *et al.* [SWT14] opt for the exact same loss function for their face verification module. Masci *et al.* [MBBS14] generalise this objective in order to couple multiple modalities in the same feature space. Given two distinct siamese-based embeddings ξ and η respectively representing the modalities X and Y, with $\mathbf{x}$ and $\mathbf{y}$ the corresponding data, they define a cross-modal loss with an energy:

$$E_{W,XY} = \|\xi(\mathbf{x}) - \eta(\mathbf{y})\|_2 \tag{3.14}$$

Bordes *et al.* [BWCB11] choose the L_1 norm with a similar objective function. One similar and one dissimilar pairs, respectively called x and x^{neg}, and sharing one sample, are selected. This strategy is equivalent to the triplet representation from Lefebvre *et al.* [LG13]. However, since the sub-networks do not share their weights, the position of the mutual example in both pairs is important. The network is updated only when a unit margin is not respected between the similarity scores $f(x)$ and $f(x^{neg})$:

$$E_W(x, x^{neg}) = \max(0, 1 - f(x^{neg}) + f(x)). \tag{3.15}$$

After presenting the objective functions based on the two main metrics, respectively the cosine and Euclidean metrics, we develop the statistical objective functions, where similarities between sets of samples depend on their means and correlation matrices.

3.1.3.3 Statistical Objective Function

In order to discern different speakers using a CNN auto-encoder, Chen *et al.* [CS11] suggest to regularise half of the encoding layer with a Siamese objective. Mel Frequency Cepstral Coefficients (MFCC) are extracted from T_B sliding temporal windows over each sample, whose input thus corresponds to a set of feature vectors $\mathbf{X} = \{x_t\}_{t=1}^{T_B}$. The new representation $CS(\mathbf{X}) = \{CS(x_t)\}_{t=1}^{T_B}$ of the sample becomes a set of higher level feature vectors, and the Siamese network is trained following an incompatibility measure based on the first and second-order statistics of this new representation, respectively D_m and D_S. Let $\mu^{(i)}$ and $\Sigma^{(i)}$ be the mean and covariance matrix for the sample $i, i \in [1, 2]$ of a pair, and $\|.\|_F$ the Frobenius norm:

$$\mu^{(i)} = \frac{1}{T_B}\sum_{t=1}^{T_B} CS(x_{it}) \ , \ \Sigma^{(i)} = \frac{1}{T_B - 1}\sum_{t=1}^{T_B}\left[CS(x_{it}) - \mu^{(i)}\right]\left[CS(x_{it}) - \mu^{(i)}\right]^\top , \quad (3.16)$$

then,

$$D_m = \left\|\mu^{(\mathbf{1})} - \mu^{(\mathbf{2})}\right\|_2^2 \quad , \quad D_S = \left\|\mathbf{\Sigma^{(1)}} - \mathbf{\Sigma^{(2)}}\right\|_F^2 . \quad (3.17)$$

Given λ_m and λ_S the tolerance bounds of incompatibility scores in terms of D_m and D_S estimated from the training data, the final Siamese objective function is very similar to the one proposed by Chopra *et al.* [CHL05]:

$$L(W, Y, \mathbf{X_1}, \mathbf{X_2}) = (1 - Y)(D_m + D_S) + (Y)(\exp(\frac{-D_m}{\lambda_m}) + \exp(\frac{-D_S}{\lambda_S})). \quad (3.18)$$

Although this objective functions involves the same CNN for both samples, it is also sollicited T_B times for each sample of a pair.

We have presented the multiple objective functions which reflects the similarity relationships between sets of samples in the output space. It is now necessary to explain how to benefit from the information brought by every sample of these training sets for a model update.

3.1.4 Training Algorithm

As for most of the neural networks, the main algorithm to train networks in a Siamese architecture is the stochastic gradient descent, using the backpropagation algorithm. However, two main methods exist in order to perform the gradient descent for a set of training samples.

The first approach [LG13] consists in applying the exact same backpropagation algorithm as for a single sample. This amounts to activating one single network with every member of a training set except one, so as to recover the states of the output layer for each of them. It is then possible to activate the network with the last sample, considered as the reference in the training set, and compute the objective function gradient only in relation to that single sample.

The second approach is the most popular, and more mathematically correct. Let us define a training set of samples $T_S = \{\mathbf{X_p}, p \in [1..n]\}$ and the same notations as in Section 2.3.4.1, with an added exponent denoting which sample each value corresponds to. We activate n identical networks in parallel with each sample, storing their activation states. It is possible to compute the output layer δ values for every sub-network

directly: for the j^{th} neuron of the output layer O of the p^{th} sub-network, noted $\mathcal{N}_p$, we compute δ^p_{Oj} as follows:

$$\delta^p_{Oj} = \frac{\partial E}{\partial net^p_j}. \tag{3.19}$$

This allows for independent error backpropagations in each sub-network using the classical *delta rule*. In opposition to the first approach, the final update equation takes into account every sample from the training set, by using the chain rule for composite functions:

$$\begin{aligned} \omega^{t+1}_{ij} &= \omega^t_{ij} - \lambda \frac{\partial E}{\partial \omega_{ij}}(\omega^t_{ij}) = \omega^t_{ij} - \lambda \sum_p \frac{\partial E}{\partial net^p_j}.\frac{\partial net^p_j}{\partial \omega_{ij}} \\ &= \omega^t_{ij} - \lambda \sum_p a^p_i.\delta^p_j. \end{aligned} \tag{3.20}$$

Algorithm 2 presents a summary of the BackPropagation generalisation for Siamese networks.

Algorithm 2 Siamese Generalised Training algorithm

for all $\mathbf{X_p}$ in $T_S = \{\mathbf{X_p}, p \in [1..n]\}$ **do**
    Forward pass $\mathbf{X_p}$ in its corresponding sub-network $\mathcal{N}_p$
end for
for all $\mathbf{X_p}$ in T_S **do**
    procedure COMPUTE THE DELTA VALUES FOR THE OUTPUT LAYER O OF $\mathcal{N}_p$
        for all neurons n_{Oj} in O **do**
            $\delta^p_{Oj} = \varphi'(net^p_{Oj}).\frac{\partial E}{\partial a^p_{Oj}}$
        end for
    end procedure
    procedure APPLY THE DELTA RULE TO $\mathcal{N}_p$:
        for all hidden layers L **do**
            for all neurons n_{Li} in L **do**
                $\delta^p_{Li} = \varphi'(net^p_i).\sum_j \omega_{ij}.\delta^p_{(L+1)j}$
            end for
        end for
    end procedure
end for
procedure UPDATE THE SHARED WEIGHTS
    $\omega^{t+1}_{ij} = \omega^t_{ij} - \lambda \sum_p a^p_i.\delta^p_j$
end procedure

Thus, Siamese networks differ from classical networks essentially by their training strategy involving multiple samples. However, since the Siamese architecture is devised so as to produce a similarity metric in its projection space, it is necessary to use another model on the projected data to obtain the final label of a sample for classification tasks.

3.1.5 Siamese Network for Verification/Classification

Siamese networks may be used straightforwardly for their verification capacities, keeping the identical parallel subnetworks in order to rate the similarity of a pair of

samples. As seen above, Nair *et al.* [NH10] directly train the network to give the probability for two sample images to share the same identity. Masci *et al.* apply the Hamming distance on their Siamese-extracted binary hash in order to rank the relevance of matches with a reference image. Hu *et al.* [HLT14] and Zheng *et al.* [ZIG+15] consider a threshold on the value of the trained metric, respectively Euclidean and cosine-based, to assess the similarity or dissimilarity between pairs of samples. Zheng *et al.* [ZDI+15] also apply the Triangular Similarity Metric Learning objective function to the MLP architecture for face identification and dimensionality reduction. The networks are trained from mini-batches composed of $N(N-1)/2$ pairs, with N the number of training examples, with inputs formed of different face descriptors such as Gabor wavelets, Local Binary Patterns, and Over-complete Local Binary Patterns. Identification is then performed thanks to a K-NN classifier using the cosine function to measure the pairwise distance with the nearest neighbour, i.e. $K = 1$.

The learned metric can also serve as a higher-level representation, combined with other classifiers to assign a label to a single sample, whether this label is binary for verification, or multi-valued for recognition. This classifier can be exemplar-based, depending on direct comparisons with the projected training samples, or trained from this new representation.

For their verification step, Bromley *et al.* [BGL+94] only use one single sub-network, and the output for the pattern to be recognised is compared to a multivariate normal density model trained from the features of six examples of a person's signature. Similarly, Chopra *et al.* [CHL05] compute the likelihood for a test face image to be genuine, $\rho_{genuine}$, by evaluating the normal density of the test image on the model of the concerned subject. The probability that the given image corresponds to the subject is defined as:

$$Prob(genuine) = \frac{\rho_{genuine}}{\rho_{genuine} + \rho_{impostor}}, \tag{3.21}$$

with $\rho_{impostor}$, the average $\rho_{genuine}$ value for all the training impostor images, which allows for a consideration of the Gaussian model global performance. Sun *et al.* [SWT14] combine the Siamese architecture with a classification network, through an added softmax layer trained with a cross-entropy objective function. In this case, the Siamese objective becomes a regularisation over the error of the penultimate layer, which has to accommodate two objectives simultaneously. This approach has the benefit of a single training for both identification/classification and verification. However, they operate their final face verification thanks to a Joint Bayesian Model trained from the PCA-compressed projected training samples, and do not comment on the accuracy of the softmax identification layer.

Chen *et al.* [CS11] apply a one-nearest-neighbour to evaluate their speaker comparison system with a metric based on the same statistics used during training. Given a pair of speaker models $\{\mathcal{SM}_i = \{\mu_i, \Sigma_i\}, i = 1; 2\}$, they define the speaker distance metric

$$d(\mathcal{SM}_1, \mathcal{SM}_2) = tr\left[(\Sigma_1^{-1} + \Sigma_2^{-1})(\mu_1 - \mu_2)(\mu_1 - \mu_2)^\top\right] \tag{3.22}$$

Yih *et al.* [YTPM11] propose the same classification method to pair Wikipedia articles written in English with their Spanish counterpart, while Lefebvre *et al.* recognise faces within a pool of 68 subjects.

To sum up, Siamese networks are semi-supervised models, trained to learn previously known similarity relationships from sets of samples. These relationships are embedded in a specific objective function, which characterises the final projected higher-level features. Most Siamese networks are however used exclusively in a verification

mindset, with a binary classification problem. While Euclidean-based objectives are more flexible and more suited to mapping-learning, they require an additional margin tuning which can be detrimental to the global performance, while cosine-based strategies are not appropriate for multi-class problems combining classification and rejection, which are both necessary for our gesture recognition system.

3.2 Contributions on the SNN

As seen in the last section, it is necessary to develop the potential for Siamese networks in order to tackle multi-class configurations with novelty detection. In the following, we propose multiple contributions aiming at increasing the discriminative potential of the network and improve its convergence, so as to be able to combine classification and rejection in a new feature space.

After defining our base choices for our Siamese network, we will present our contributions. Firstly, we suggest a more general way to define similarities in multi-class problems. Indeed, the pair or triplet strategy is not sufficient to define every relationship between the different classes, which leads to biases in the training set selection. Moreover, a mathematical analysis developed below proves that cosine-based objectives can lead to numerical instabilities and uncontrolled behaviours, leading us to a regularisation of the original metric, for a simpler and more computationally efficient version. We also propose to modify the negative part of the objective function thanks to the *polar sine* notion, which consists in a function that estimates a similarity between sets of vectors, and whose value is maximal for a set of mutually orthogonal vectors. This approach can be interpreted as an Independent Component Analysis. We finally conclude with a rejection strategy applied to the learnt representation.

3.2.1 Architecture and General Choices

In the following, we choose a cosine-based objective function for its ease of use, and a MLP-based Siamese architecture, with hyperbolic tangents activation functions for every layer. We use the work of Bromley *et al.* [BGL+94], and Lefebvre *et al.* [LG13] as references, with a base objective function depending on the square error objective. Since a target angle equal to -1 for a dissimilar pairs induces an instability, as this target cannot be met for more than two classes, we opt for a target for equal to 0. However, we do not use the common two-sub-network architecture, since the latter is greatly modified by a our specific training set selection strategy, presented below.

3.2.2 Training Set Selection Strategy

Every training set selection strategy for a Siamese network consists in defining a subjective number of similar and dissimilar pairs, deemed representative of the global relationships within the data. This generally induces a bias, since it is not possible to ensure a perfect coverage for every relationship. This is why we first propose a unified approach for multi-class problems.

Let $C = \{C_1, .., C_K\}$ be the set of classes represented in the training data, $\mathbf{O_{R_k}}$ the output vector of the reference sample $\mathbf{X_{R_k}}$ from the class C_k presented to the model for update, $\mathbf{O_{P_k}}$ the output of a different sample $\mathbf{X_{P_k}}$ from the same class, and $\mathbf{O_{R_l}}$ the output of a sample $\mathbf{X_{N_l}}$ from another class C_l.

In order to keep symmetric roles for every class and optimise the efficiency of every update, we propose here to minimise an error criterion for training tuples $T_k = \{\mathbf{X_{R_k}}, \mathbf{X_{P_k}}, \{\mathbf{X_{N_l}}, l = 1..K, l \neq k\}\}$ involving one reference sample from the class C_k, one positive sample and one negative sample from every other class.

Our proposal for the error estimation $E_W(T)$ becomes:

$$E_W(T) = (1 - cos(\mathbf{O_R}, \mathbf{O_P}))^2 + \sum_l (0 - cos(\mathbf{O_R}, \mathbf{O_{N_l}}))^2. \tag{3.23}$$

Thus, our network architecture dynamically changes depending on the training subsets sizes, and involves as many sub-networks as classes. Every sample is taken once as a reference, while the others are drawn at random. This facilitates the selection of representative training sets, and gives a global approach which does not require any additional parameter. Moreover, this strategy acts as a mini-batch learning, which limits the number of updates required before convergence.

In the next section, we present our second contribution, which consists in a regularisation of the objective function. This regularisation aims at controlling the behaviour of the output vector norms, which are shown to be uncontrolled thanks to a mathematical analysis.

3.2.3 Norm Regularisation

In order to improve convergence, we also study the behaviour of a weight update over the projected samples. In the following, we will analyse the cosine metric as a function of two vectors of dimension n,

$$cos_{X_1,X_2} : \mathbb{R}^{2n} \to \mathbb{R}/(\mathbf{X_1}, \mathbf{X_2}) \to \frac{1}{2}(1 - \cos(\mathbf{X_1}, \mathbf{X_2}))^2. \tag{3.24}$$

Let $(\mathbf{O_1}, \mathbf{O_2})$ be a pair of outputs to be updated. Given the functions

$$\begin{aligned} cos_{O_1} &: \mathbb{R}^n \to \mathbb{R}/\mathbf{X} \to \frac{1}{2}(1 - \cos(\mathbf{O_1}, \mathbf{X}))^2 \\ cos_{O_2} &: \mathbb{R}^n \to \mathbb{R}/\mathbf{X} \to \frac{1}{2}(1 - \cos(\mathbf{X}, \mathbf{O_2}))^2 \end{aligned} \tag{3.25}$$

respectively evaluated at the points $\mathbf{O_2}$ and $\mathbf{O_1}$, the cos_{X_1,X_2} directional derivative at $(\mathbf{O_1}, \mathbf{O_2})$ can be expressed as the concatenation of the two directional derivatives $\nabla_{cos_{O_1}}(O_2)$ and $\nabla_{cos_{O_2}}(O_1)$.

We will show here that every stochastic gradient descent will increase the norms for both samples. Indeed, if we consider the function cos_{O_1}, we have an update of $\mathbf{O_2}$ which follows:

$$\mathbf{O_2}^{t+1} = \mathbf{O_2}^t - \lambda . \nabla_{cos_{O_1}}(\mathbf{O_2}). \tag{3.26}$$

Besides, the line directed by the vector $\frac{\mathbf{O_2}}{\|\mathbf{O_2}\|}$ belongs to the equipotential for the $\cos_{O_1}$ function. By definition, we can conclude that the directional derivative $\nabla_{cos_{O_1}}(\mathbf{O_2})$ is orthogonal to $\mathbf{O_2}$. Thanks to Pythagoras' theorem, we can conclude:

$$\begin{aligned} \left\|\mathbf{O_2^{t+1}}\right\|^2 &= \left\|\mathbf{O_2^t}\right\|^2 + \lambda^2 . \left\|\nabla_{\mathbf{cos_{O_1}}}(\mathbf{O_2})\right\|^2 \\ \Rightarrow \left\|\mathbf{O_2^{t+1}}\right\| &> \left\|\mathbf{O_2^t}\right\|. \end{aligned} \tag{3.27}$$

We propose a graphical representation of this phenomenon in 3D in Figure 3.3. The two vectors $\mathbf{O_1^t}$ and $\mathbf{O_2^t}$ are represented with the same norm for visualisation simplification, thus belonging to a sphere. The equipotential for cos_{O_1} forms a cone, centred on $\mathbf{O_1^t}$, and directed by $\mathbf{O_2^t}$. It is then easier to see the higher norm of the updated $\mathbf{O_2^{t+1}}$ after a step of gradient descent. It is important to note that the reasoning was based on the output vectors components, and does not take into account the additional non-linearity induced by the weights of the network.

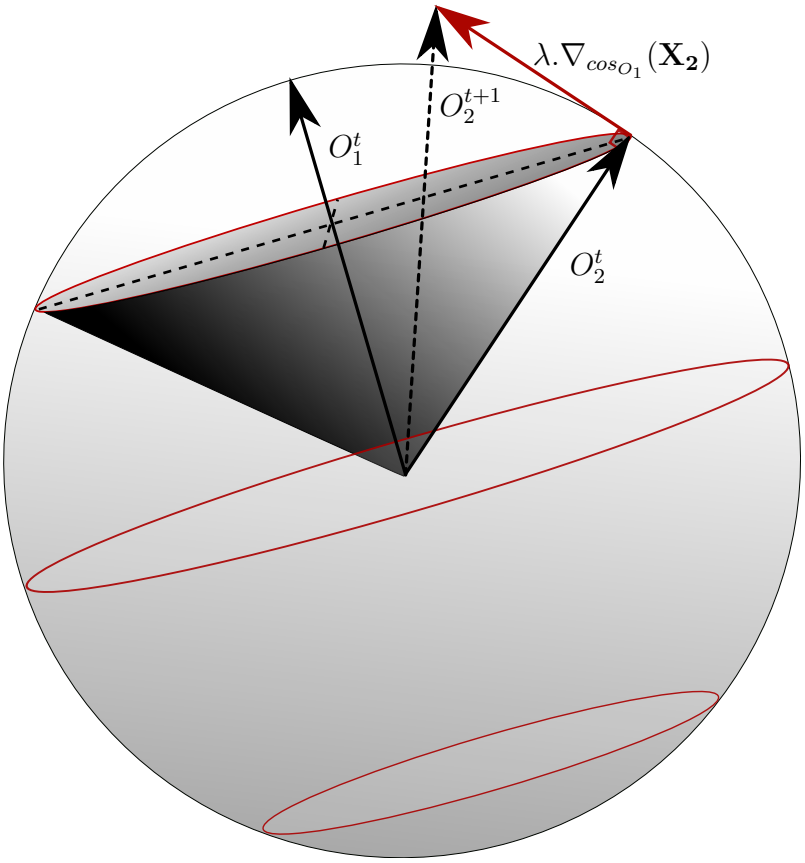


Figure 3.3 – Representation of the effect of an update step on the norm of the projections for a pair of outputs $(\mathbf{O_1^t}, \mathbf{O_2^t})$ at the epoch t in 3D. The centre of the sphere corresponds to the origin of the output space. The grey cone represents the equipotential surface for the function cos_{O_1}. For simplification, $\mathbf{O_1^t}$ and $\mathbf{O_2^t}$ were given the same norm.

Uncontrolled norms complicate training, possibly leading to a progressive divergence. Moreover, in the case of hyperbolic tangent activation functions, the output space is a higher-dimensional cube of dimension n, which restricts the norms to a maximum of $\sqrt{n}$. Thus, to stabilise the convergence of the model, we propose to add constraints on the norms of every output, forcing them to one. These conditions ensure that the norms of the outputs will not keep on increasing with each update, thus preventing any saturation of the outputs.

We can define $E_W(T_s)$ the error for a training subset T_s as:

$$\begin{aligned} E_W(T_s) =& (1 - cos(\mathbf{O_R}, \mathbf{O_P}))^2 + \sum_l (0 - cos(\mathbf{O_R}, \mathbf{O_{N_l}}))^2 \\ &+ \sum_k (1 - \|\mathbf{O_k}\|)^2. \end{aligned} \tag{3.28}$$

Given $cos(\mathbf{O_1}, \mathbf{O_2}) = \frac{\mathbf{O_1}.\mathbf{O_2}}{\|\mathbf{O_1}\|.\|\mathbf{O_2}\|}$, we also propose to replace the cosine distance for each pair by the scalar product $\mathbf{O_1}.\mathbf{O_2}$ between two sample outputs $\mathbf{O_1}$ and $\mathbf{O_2}$. Since their norms are set to one for our training subsets, these conditions ensure that the cosine distance between these outputs is still equal to the original target.

Thus, we define the final error estimation over all the chosen training subsets $T_s, s \in [\![1, \tau]\!]$ E_W as:

$$E_W = \sum_{s \in [\![1,\tau]\!]} E_W(T_s), \tag{3.29}$$

with

$$\begin{aligned} E_W(T_s) =& (1 - \mathbf{O_R}.\mathbf{O_P})^2 + \sum_l (0 - \mathbf{O_R}.\mathbf{O_{N_j}})^2 \\ &+ \sum_k (1 - \|\mathbf{O_k}\|)^2. \end{aligned} \tag{3.30}$$

While this formulation is more suited to handle angular updates, it combines many independent targets, which may reveal impractical for an increasing number of classes. Moreover, the mean square error objective has specific drawbacks. Indeed, in the derivative form, the cosine error is pondered by a factor $(target - cosine\ value)$, which tends to zero as the model converges. Thus, we propose a new error function, which will preserve the targets, while answering to both of these problems.

3.2.4 Angle Problem Reformulation

While the cosine allows for a correlation estimation between two vectors in any Euclidean space of finite dimension, it is sensible to consider another function which would measure dissimilarities, like the sine in 2D. In the following, we propose a reformulation of the objective function based on a higher-dimensional dissimilarity measure, the polar sine. We will then show that this new analysis leads to a non-linear discriminant analysis.

Polar Sine Definition

In 2D (see Fig.3.4), given the origin O and two vectors $\mathbf{a} = \mathbf{OA}$ and $\mathbf{b} = \mathbf{OB}$, and $V(V(\mathbf{a}, \mathbf{b})$ the area of the polytope formed by those two vectors, we have the relation:

$$sin(\mathbf{a}, \mathbf{b}) = \frac{V(\mathbf{a}, \mathbf{b})}{\|\mathbf{a}\| \, . \, \|\mathbf{b}\|} \tag{3.31}$$

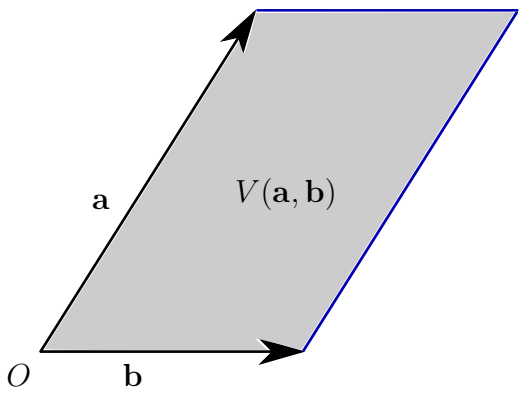


Figure 3.4 – Area of a 2D polytope implied by a vertex of two vectors $\mathbf{a}$ and $\mathbf{b}$

Inspired by that formula, Lerman *et al.* [LW09] define the *polar sine* for a set $V = \{v_1, \ldots, v_n\}$ of n-dimensional linearly independent vectors:

$$PolarSine(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \frac{\Omega}{\Pi}, \tag{3.32}$$

where

$$\Omega = \left|\det \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix}\right|, \tag{3.33}$$

and

$$\Pi = \prod_{i=1}^{n} \|\mathbf{v_i}\| . \tag{3.34}$$

Furthermore, another definition for the polar sine exists for a set $V_n = \{v_1, \ldots, v_n\}$ of m-dimensional vectors, with $n < m$. Given $\mathbf{A} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix}$ and its transpose $A^\top$, we define:

$$\Omega = \sqrt{\det\left(\mathbf{A}^\top.\mathbf{A}\right)}. \tag{3.35}$$

Also, when the norms are incorporated wisely in the matrix $\mathbf{A} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix}$, the Polar Sine is equivalent to the square root of the determinant of the matrix

$$\mathbf{A_{norm}} = \begin{bmatrix} \frac{\mathbf{v}_1}{\|\mathbf{v_1}\|} & \frac{\mathbf{v}_2}{\|\mathbf{v_2}\|} & \cdots & \frac{\mathbf{v}_m}{\|\mathbf{v_m}\|} \end{bmatrix}. \tag{3.36}$$

Thus, the Polar Sine corresponds effectively to the square root of the determinant of a matrix of cosine values, where the component of the i^{th} line and j^{th} column $S_{norm}(i, j)$, with $\mathbf{S} = A^\top_{norm}.A_{norm}$:

$$\begin{aligned} \mathbf{S}(i, j) &= \cos(\mathbf{v_i}, \mathbf{v_j}), \\ PolarSine(\mathbf{v}_1, \ldots, \mathbf{v}_n) &= \sqrt{\det\left(\mathbf{S}\right)}. \end{aligned} \tag{3.37}$$

Adaptation of the Polar Sine for learning dissimilarities

The polar sine corresponds to a measure of a regularised hyper-volume, which, as seen above, only depends on the angles between every vector of the set, and reaches its maximum value when its edges are orthogonal. In that sense, the polar sine will reach its maximum value when every vector of the set is orthogonal to every other.

However, the error computation of the polar sine introduces a matrix inversion (see Appendix A.2.3), which provokes a highly unstable behaviour. Moreover, the derivative of the determinant of the matrix $\mathbf{S}$ is still dependent to the value of this determinant, which, as the product of n eigenvalues, is initially all the lower as the number of vectors used increases. For example, if initial eigenvalues, comprised between 0 and 1, are equal to 0.5, then the value of the Polar Sine is equal to 0.5^n, making the error too small to actually allow a correction. Inspired by the equation A.2.2 where the division by the cosine value is compensated, we can deduce that the correct function should also normalise the inverse of the matrix $\mathbf{S}$.

As a consequence, we propose a redefinition of the Polar Sine for learning angles. In the following, we call this adaptation the *Polar Sine Metric*:

$$psin(\mathbf{A}) = \sqrt[n]{\det\left(\mathbf{S}\right)}. \tag{3.38}$$

By using the n^{th} root of the determinant of the matrix $\mathbf{S}$, i.e. $\sqrt[n]{\det\left(\mathbf{S}\right)}$, the new value becomes independent from its dimension, which is equal to the number of samples

in the set. Furthermore, the error for the Polar Sine Metric (see Appendix A.14) is computed as:

$$\frac{\partial\left(psin(\mathbf{A})\right)}{\partial a_{ij}} = \frac{psin(\mathbf{A})}{n}.\frac{\varphi'(I_{ij})}{\|\mathbf{A_j}\|}.\left[\mathbf{A_{norm}S^{-1}} - \mathbf{A_{norm}}\right]_{ij} \tag{3.39}$$

Thus, the reader can see that every component of the matrix inverse $\mathbf{S^{-1}}$ is multiplied by the Polar Sine Metric value, i.e. the n^{th} root of the determinant, which leads to a perfect normalisation, as

$$\begin{aligned}\det\left(\sqrt[n]{\det\left(\mathbf{S}\right)}\mathbf{S^{-1}}\right) &= \left(\sqrt[n]{\det\left(\mathbf{S}\right)}\right)^n \det\left(\mathbf{S^{-1}}\right)\\ &= \det\left(\mathbf{S}\right)\det\left(\mathbf{S^{-1}}\right)\\ &= \det\left(\mathbf{SS^{-1}}\right)\\ &= 1.\end{aligned} \tag{3.40}$$

With two comparable similarity estimators, whose values are comprised between 0 and 1, it is now possible to redefine the objective function for our training sets $T_k = \{\mathbf{O_{R_k}}, \mathbf{O_{P_k}}, \{\mathbf{O_{N_l}}, l = 1..K, l \neq k\}\}$:

$$\begin{aligned}&E_W(T_k) = Esim_W(T_k) + E\overline{sim}_W(T_k),\\ &with\\ &\begin{cases}Esim_W(T_k) = (1 - \cos(\mathbf{O_{R_k}}, \mathbf{O_{P_k}}))^2\\ E\overline{sim}_W(T_k) = (1 - psin(\mathbf{O_{R_k}}, \mathbf{O_{N_1}}, \ldots, \mathbf{O_{N_K}}))^2.\end{cases}\end{aligned} \tag{3.41}$$

Optimising the Polar Sine Metric corresponds to assigning a target equal to 0 to the cosine between every pair of different vectors drawn in $T_k \setminus \{O_{P_k}\}$. This actually holds more information than our original objective function which aimed at assigning zero-cosine-values only for pairs between the reference and negative outputs, and may increase the efficiency of a single update, as every vector in the matrix $\mathbf{A}$ plays the same role, contrary to our previous tuple-based error function. Finally, this approach is easily scalable to any number of classes.

The Siamese Network as a Supervised Non-Linear Independent Component Analysis

While the cosine metric describes the *intra-class* similarities, the polar sine metric is an indicator for similarities between classes, with a maximum value when every class is decorrelated from the other.

Given a set number of sources $\{\mathbf{G_1}, \ldots, \mathbf{G_n}\}$, which correspond to the examples for each of n classes, we transform these initial inputs into maximally independent, multi-dimensional components $\{\mathbf{O_1}, \ldots, \mathbf{O_n}\}$ thanks to the non-linear SNN neuron network. Indeed with a measure of independence $psin(\mathbf{O_1}, \ldots, \mathbf{O_n})$ fully defined by the Polar Sine Metric. Moreover, the similarity target defined by $Esim_W$ (see Equation 3.41) reinforces at the same time the correlation between samples from a common source. This approach is also flexible in the sense that the size of the output layer is adjustable, which allows for the selection of the number of components.

To conclude, our Siamese network, combined with this new objective function, presents all the properties of a Supervised, Stochastic Non-Linear Independent Component Analysis, with multiple advantages, apart from non-linearity. Indeed, the number

of components is adjustable. Furthermore, with a large enough training dataset, it is possible to devise strategies for the choice of the number of components, thanks to a classification score maximisation scheme on a validation set.

After explaining our contributions about the parametrisation and training of the SNN, we present one original use we make of the SNN, which aims at improving novelty detection and rejection based on the higher-level representation obtained with this network.

3.2.5 Dealing with Rejection

Our original hypothesis is that, as Siamese networks learn a structured output space in relation to similarities between samples, they are more apt to produce coherent results when presented unknown samples, or samples which are too different from the training set.

Any classifier can be used on the extracted feature vectors. We choose a K-NN classification based on the cosine similarity metric in order to prove the validity and reliability of the learned SNN projection. Indeed, while the K-NN classifier does not scale efficiently for larger datasets, it stays relevant for the domain of gesture recognition.

Preliminary test results for rejection are available in Annex B. The network, with one 45-neuron hidden layer and one 3-neuron output layer, is trained using three of the four horizontal possible translation gestures, recorded on a Smartphone for a single user. The respective projections of the training and test sets are represented in Figures B.1 and B.2. It is important to note that the samples from the unknown class, in black, are separated from the other classes, who show a tendency to orthogonality.

Thus, our rejection criterion consists in a single threshold, common to all classes, on the distance to the closest known sample. Error detection should prove all the better as the within-class similarity is strong, and novelty detection is facilitated by high inter-class decorrelation and dissimilarity.

3.3 Conclusion

Siamese Neural Networks are specific networks trained to reflect similarities within a training set. Its architecture, based on CNNs or MLPs, consists of multiple identical, weight-sharing parallel networks, simultaneously activated by sets of samples. These parallel networks are joined at their ends by a contrastive loss layer, which computes a given objective in the output space, devised as a representation of the similarities between samples.

The two main types of objective functions are built around the cosine metric, with angular constraints, or the Euclidean distance, with norm constraints. Similarities are modelled using labelled pairs of samples, or triplets forming one similar pair and one dissimilar pair. While the Euclidean distance allows for more flexibility, it still depends on a margin parameter.

Thus, we propose three contributions for a Siamese network based on a MLP, with a view to classification problems. We generalise the similarity representation to groups of samples, where every class is taken into account, which benefits from a batch effect during training, and simplifies greatly an otherwise biased pair selection strategy. Moreover, we contribute to the parameter-free cosine objective function by applying

a regularisation on the norms of the outputs, whose behaviour is not controlled, as seen in section 3.2.3. Finally, we propose a reformulation of the dissimilarities objective thanks to a new similarity metric for sets of samples, based on the polar sine. This approach can be interpreted as a non-linear discriminant analysis, which promotes our goal to take advantage of the Siamese characteristics for error and novelty rejection.

Five hypotheses are made about our contributions.

— The first hypothesis H_1 corresponds to the target choice for similar and dissimilar pairs. Indeed, we surmise that an orthogonality objective for the cosine value between negative samples pairs leads to a better, more stable convergence.

— Hypothesis H_2 is related to the training set selection strategy: we conjecture that a tuple-based set selection strategy allows for a better representation of the relationships between classes, thus leading to a better structuring of the output space.

— With hypothesis H_3, we suggest a better convergence and a more stable norm evolution behaviour thanks to the proposed norm regularisation scheme.

— H_4 concerns the angular reformulation, in the sense that the Polar Sine Metric-based objective, through a non-linear discriminant analysis, is more efficient at separating classes than other objective functions.

— Finally, we hypothesise with H_5 the stronger rejection capabilities of the SNN from its discriminant learning leading to a more discriminant distance.

In the next chapter, we aim at analysing and confirming or invalidate each of these hypotheses thanks to tests performed on two real-life inertial datasets, the Multimodal Human Activity Dataset, and the Orange Dataset.

Chapter 4

Experiments and Results

This chapter aims at characterising our contributions for Siamese networks, in particular for their classification and discrimination potential. Experiments are carried out on two different datasets. Firstly, tests based on the inertial sensors of the Berkeley Multimodal Human Action Database (MHAD) act as a comparison between the different SNN configurations, identifying the strength and weaknesses of every contribution. Secondly, a real inertial smartphone-oriented dataset, called the Orange Dataset, collected by ourselves at Orange Labs, serves as the basis for our study of symbolic gesture classification and rejection, with a comparison to the main state-of-the-art methods. It is important to note that every test result corresponds to the best identified configuration on the corresponding test set. Indeed, the lack of sufficient data prevents from forming a representative validation set, which would usually be used to determine a score in more realistic conditions.

4.1 Multimodal Human Activity Dataset

In this section, we aim at comparing the different choices for the SNN configuration on a public dataset. The dataset and experimentation protocol encompass a more general use of the inertial sensor for activity recognition, based on the Multimodal Human Action Database (MHAD).

In this case, the SNNs should demonstrate their generalisation capabilities for recognizing longer sequences, representing actions performed with different styles by multiple users.

In the following, we first introduce the Berkeley MHAD. Then, the comparison protocols are presented, along with the different SNN variants explored. Evaluation criteria are proposed, based on classification results, output projection analyses and algorithm complexities. Finally, results are investigated and conclusions are drawn.

4.1.1 Database Introduction

The Multimodal Human Activity Database (MHAD) [OCK+13] was generated as part of the project "A Bio-Inspired approach to Recognition of Human Movements and Movement Styles". This database comprises the recordings from 12 participants performing 11 different actions (see Fig.4.1). These actions were designed to cover diverse dynamics combinations for different body extremities, and were not specifically

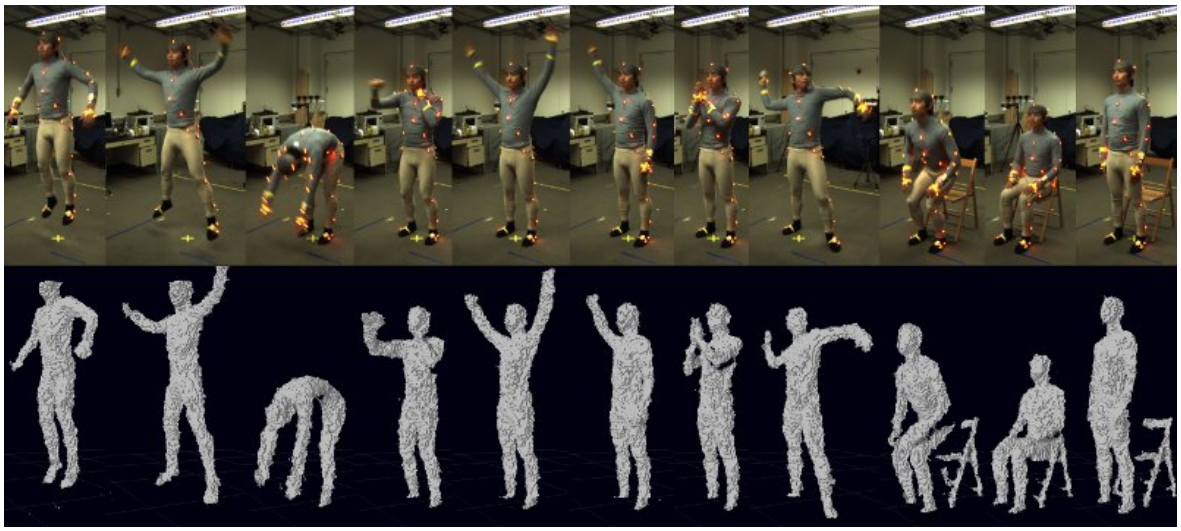


Figure 4.1 – Snapshots from all the actions available in the Berkeley MHAD are displayed together with the corresponding point clouds obtained from the Kinect depth data. Actions (from left to right): *jumping*, *jumping jacks*, *bending*, *punching*, *waving two hands*, *waving one hand*, *clapping*, *throwing*, *sit down/stand up*, *sit down*, *stand up*. Extracted from [OCK+13].

detailed to the subjects in order to collect a representative range of styles for each action.

The MHAD gathers 5 repetitions by user and by action, for a total of about 660 action sequences, amounting to about 82 minutes of total recording time. Three different groups of actions can be identified. The first group shows movement in both upper and lower extremities (*jumping in place*, *jumping jacks*, *throwing*). The second group consists of actions with high dynamics in upper extremities (*waving one hand*, *waving two hands*, *clapping hands*, *punching*, *bending*), while the third group proposes actions with high dynamics in lower extremities (*sit down*, *stand up*, *sit down then stand up*).

While multiple sensors were deployed during the data collection process, combining stereo cameras, Kinect devices, microphones and Motion Capture, we focus our study on the inertial sensors, consisting in six wireless three-axis accelerometers placed on the bodies of the participants, with one on each wrist, one on each foot, and one on each hip. The accelerometers captured the data at a frequency of about 30Hz.

Classically, gesture data are preprocessed in 3 steps: amplitude scaling and filtering, shared for both neural and DTW approaches; and specific temporal normalisations. A general amplitude scaling is performed, where each component of every sample forming a gesture record is divided by the maximum norm over all the samples of this gesture. Furthermore, this ensures additionally that every input value is comprised between -1 and $+1$, which is recommended for an efficient neural network training. Then, a low-pass filter with a parameter $\beta = 0.7$ is applied to the inertial signals so as to filter out the involuntary small shakes and electronic noise. Finally, gesture data are temporally normalised, with specific strategies suited to each method to compare. In the case where the model can handle time series, a vectorisation is applied, limiting the local approximation error by a factor 0.1. Otherwise, a common fixed size equal to 45 six-dimensional inertial samples is set for each gesture record thanks to a resampling based on a fixed curvilinear length between the new samples. This operation is performed thanks to linear interpolation or extrapolation at fixed coordinates after an estimation of the curvilinear length $L(\mathbf{G})$ of the whole gesture $\mathbf{G} = \{\mathbf{G}_0, \ldots, \mathbf{G}_n\}$, which is then

described by a 270-feature vector $\hat{\mathbf{G}}$.

Preliminary tests direct our focus to the A_1 accelerometer, attached to the user's right wrist, as it allows for the best classification rates over this dataset.

4.1.2 Protocols

This experiment aims at characterizing the behaviour of different SNN configurations. The same architecture is selected for every variant, with a 135-neuron input layer, one 45-neuron hidden layer, and one 90-neuron output layer. The hyperbolic tangent is classically chosen as the activation function for every neuron, and the learning rate is set to 0.001.

Multiple aspects are explored. Firstly, the influence of the different choices for the cosine target between two output vectors from dissimilar inputs is analysed. Secondly, our contributions are evaluated in relation to the literature, with the different training set selection strategies and choices for the cost function.

Negative Target Selection (Hypothesis H_1)
Two different types of target are commonly used for the cosine value of dissimilar output vectors. A target equal to -1 ([BGL+94]) will push these vectors apart as much as possible, creating an unstable balance as this target cannot be met for more than two classes. The second choice tends to impose an orthogonality between these vectors, with a target equal to 0 ([LG13]). While this approach allows for a better convergence as the minimum for the error function(0) can actually be attained, and it limits the space occupied by the samples. However, it is important to note that this objective requires an output space dimension superior or equal to the number of classes represented in the training set, at the risk of overstraining the system, causing instability.

Training Set Selection Strategy : pairs, triplets vs. tuples (Hypothesis H_2)
We compare three different selection strategies, and assess their relative complexities and computation costs. Let N_c be the number of classes, and N_s the number of samples in the dataset.

The first strategy ([BGL+94]) is the most common and consists in selecting a specific number of pairs of samples, labelled as "similar" or "dissimilar". Here, we propose to form one similar pair with every sample of the dataset, where the second element of the pair is drawn at random. Dissimilar pairs are selected so as to represent every dissimilarity relationship available for every sample of the dataset, for a total of $N_c - 1$ dissimilar pairs per sample. The total number of updates for one epoch is then equal to $N_c \times N_s$, with $2(N_c \times N_s)$ activations.

The second strategy [LG13] favours a symmetry between similarities and dissimilarities, forming triplets, with one reference, one similar and one dissimilar samples per set. As for the first strategy, every dissimilarity relationship is represented. As a consequence, a triplet is formed for each dissimilarity, for a total of $N_c - 1$ positive and negative pairs per sample. As such, the total number of updates is equal to $(N_c - 1) \times N_s$, with three activations per update.

Finally, the third strategy consists in our contribution, which generalizes the triplets approach with tuples. Every relationship, whether similar or dissimilar, is represented once for each sample. This limits the number of updates per epoch, with only N_s updates, for $[(N_c + 1) \times N_s]$ activations.

A1	pairs		triplets		tuples	
targets	t-11	t01	t-11	t01	t-11	t01
cos	0.901 ± 0.105	**0.916 ± 0.100**	0.881 ± 0.100	**0.916 ± 0.086**	0.880 ± 0.094	0.915 ± 0.102
scal	0.883 ± 0.087	0.869 ± 0.092	0.871 ± 0.110	0.869 ± 0.108	0.880 ± 0.078	0.886 ± 0.079
psine		0.910 ± 0.098		0.910 ± 0.084		**0.918 ± 0.091**

Table 4.1 – Recognition rates for the MHAD-based protocol.

Cost Function Choice : cosine vs. scalar product with norm regularisation vs. Polar Sine Metric (Hypotheses H_3, H_4)
The study focused on the cost function allows for a comparison between the initial cosine-based error function, referred as *cos*, and our two contributions, namely the norm regularisation, resulting in the scalar product-based cost function referred as *scal*; and the Polar Sine Metric-based cost function referred as *psine*.

These three points lead to a total of 15 configurations for the SNN, with three training set strategies, subdivided into two different negative targets for the cosine and scalar product-based SNNs, while the Polar Sine metric-based one can only accommodate an orthogonality between dissimilar output vectors (see Table 4.1). Each SNN is complimented by a K-NN classifier, with $K = 1$. Every test is performed on the MHAD, using the accelerometer named *A1*, attached to the right wrist of the user. Indeed, preliminary tests show that this sensor is the most relevant as it holds most of the discriminative information about each activity performed. A leave-one-out strategy is selected, with every sample from 11 users selected for training, and the samples from the last user for the testing phase.

4.1.3 Evaluation Criteria

The main evaluation criterion used in this study is the accuracy of the tests based on each of the 15 different configurations identified in the section above.

The projection resulting of the different cost functions is further analysed thanks to a study of the evolution of the mean output norms during training.

We also comment on the trade-offs between the reduced number of updates and the associated increased error computation complexities: complexity and convergence speeds are analysed based on the evolution during training of the classification score on the test set.

Finally, a comparison with state-of-the-art results using the same leave-one-out protocol is shown, using successively the A_1 accelerometer only; and, in order to broaden the frame of the study, the M_{20} motion capture sensor only, sensor providing the 3D coordinates in space of the right hand of each user. The same SNN architecture is used, and a focus is made on the three objective function variants with a tuple-based training set selection strategy.

4.1.4 Results

Classification accuracy scores are presented in Table 4.1.

In the following, we successively analyse the hypotheses presented in the section 4.1.2.

Negative Target Selection

For the cosine-based cost function, an orthogonality between negative vectors, with a target equal to 0, produces better overall results for every training set selection strategy than the repulsion scheme, with a target equal to -1. Indeed, classification scores may differ from 1% for a pair-based selection strategy, to 3% for triplets and tuples-based strategies. This confirms our initial hypothesis that an orthogonality objective, as a reachable target, produces a much more stable behaviour favourable during training.

However, this tendency is reversed for the scalar-based cost function with norm regularisation, with scores giving a slight advantage to the -1 target for negative relationships for pairs and triplets-based strategies. This phenomenon can be explained by the decomposition of the target angle in three sub-targets. The norm objectives have to be met perfectly for the scalar product objective to propose the right angle correction. These norms dynamically change with each update, and norms correction may be favoured over angle corrections. Thus, a target equal to -1 amplifies the scalar product error, and gives an extra edge to the amplitude of the corresponding objective error over the norm objectives error, speeding up the convergence.

Nevertheless, this tendency is reversed with the use of tuples, with a 0.6% advantage for the orthogonality objective. Indeed, as the norm objectives are met, the batch effect resulting from the representation of every class increase the global amplitude of the error dedicated to the angle correction.

Thus, hypothesis H_1 is validated for the cosine and Polar Sine Metric-based objective functions.

Training Set Selection Strategy : pairs, triplets vs. tuples

The training set selection strategy does not impact significantly the classification scores for the cosine-based cost function, with similar scores around 91.5% for the three strategies combined with a zero negative target. It is however interesting to notice a reduced standard deviation for the triplet set strategy with a cosine-based objective, due to the increased representation of the similarity relationships which improves the intra-class distances.

Once again, the cosine-based cost function shows better results than the scalar-based one, for the same reason negative targets improve the scores for this method. Indeed, the subdivision of the cosine function in three subtargets slows down the convergence considerably. This aspect is developed later in our study. Nevertheless, the tuple contribution gives the best results for the scalar-based objective.

Finally, the Polar Sine Metric-based SNN shows a correct performance in every case, comparable to the cosine-based SNN. While the pair and triplet strategies give the advantage to the cosine-based cost function, the potential of the Polar Sine Metric approach is revealed with tuples, which cover every relationship available during training. This method shows then the best accuracy, equal to 91.8%, with a reduced standard deviation of 9.1% over the cosine-based one, whose accuracy is equal to 91.5% with a standard deviation of 10.2%.

Thus, we can conclude that the Polar Sine Metric-based objective shows challenging results, as it corrects the cosine values more reliably than the scalar product-based one.

This analysis validates hypothesis H_2, with a tuple-based strategy producing better or similar results as other strategies for the three objective functions. Moreover, H_4 is validated as well, as the Polar Sine Metric-based objective combined with a tuple-based strategy produces the best classification score. However, the resulting projections differ for every cost functions, as seen below.

Projection Analysis

As exposed in the previous chapter, an update based on gradient descent with the cosine function leads to instabilities in the norms of the outputs. Indeed, for a model where the gradient descent is simply performed on the components of the output vectors, we proved that these norms increase with each update. Figures 4.2, 4.3 and 4.3 reflect the evolution of the mean norms of the output and their corresponding standard deviations for the different negative targets and training set strategies, respectively combined with a cosine-based, scalar product-based and Polar Sine Metric-based cost function. Thus, the mean amplitude of the output norms is followed with the number of epochs performed during training. Six configurations are studied, corresponding to the three objective functions for two negative targets choices, 0 and -1.

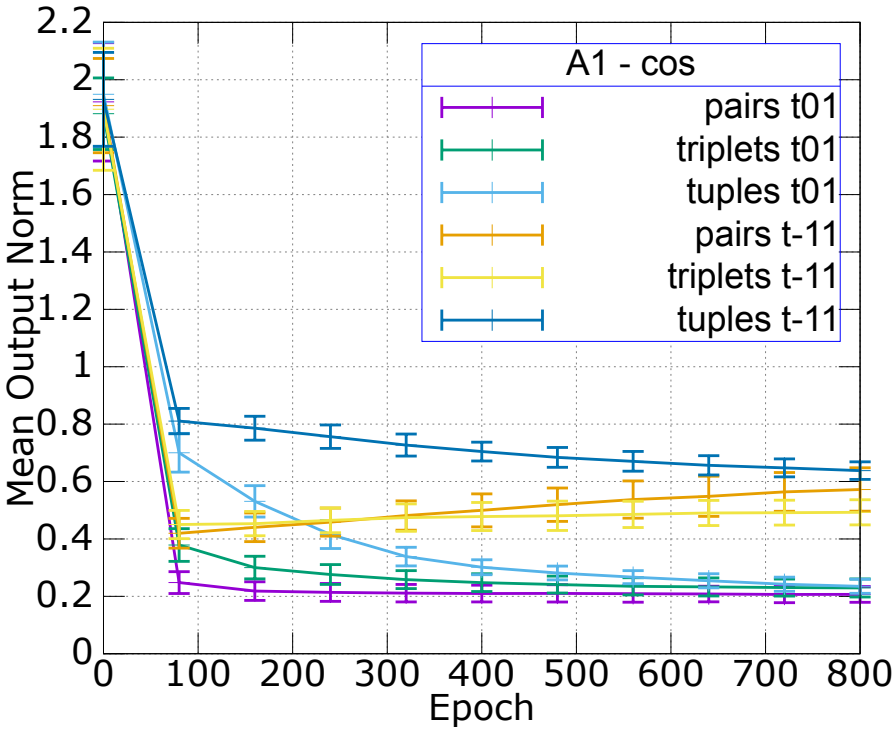


Figure 4.2 – Evolution of the mean output norm for cosine-based SNNs.

Firstly, -1 negative targets are commented. Indeed, they introduce an instability, already noted above, which leads to the expected increasing norms for the pairs and triplets strategies, for both cosine and scalar product-based cost functions, which is all the more important as the number of presented pairs during training is high. As norms amplitudes increase, two phenomena may appear. First, the angular error gradient also has to increase in order to maintain the angle error correction rate. Moreover, increasing norms lead to instabilities, as the output space is contained in a hypercube. Indeed, the maximum norm in a corner of a n-dimensional hypercube is equal to $\sqrt{n}$, while it is equal to one in the middle of a "face".

Unexpectedly, the non-linearities of the neural network, where the gradient descent is ultimately performed on the connection weights, reverse the norm evolution direction for the zero negative target. For both cosine (see Fig.4.2) and Polar Sine Metric-based (see Fig.4.4) cost functions, norms decrease steadily with each update, with a speed depending on the amplitude of the error. Thus, while this decrease slows down as the model converges, it persists, which may lead to numerical instabilities after a while for

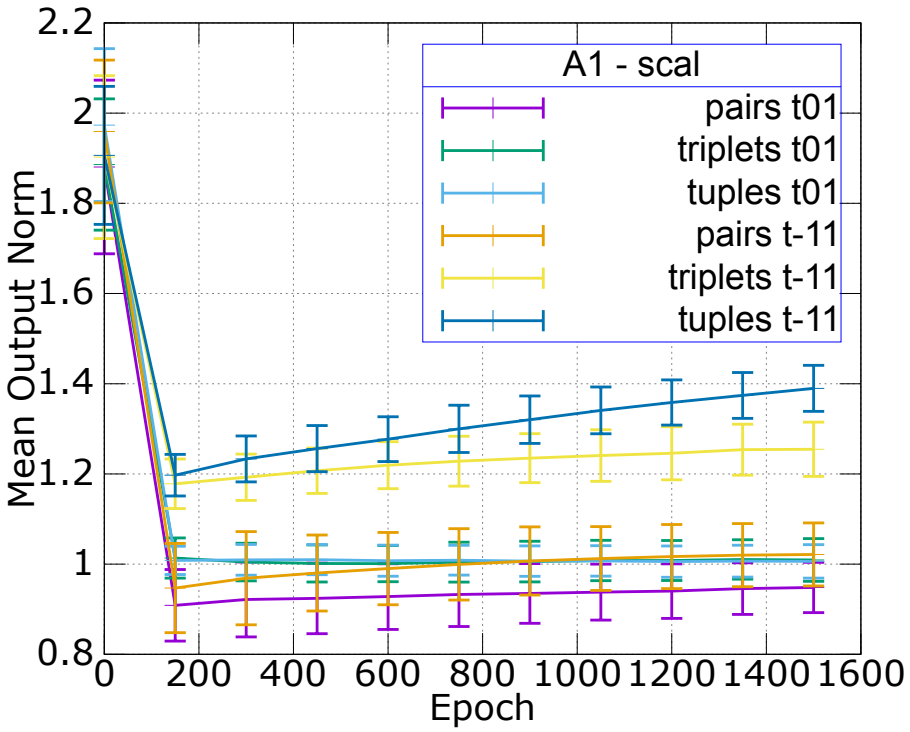


Figure 4.3 – Evolution of the mean output norm for scalar product-based SNNs.

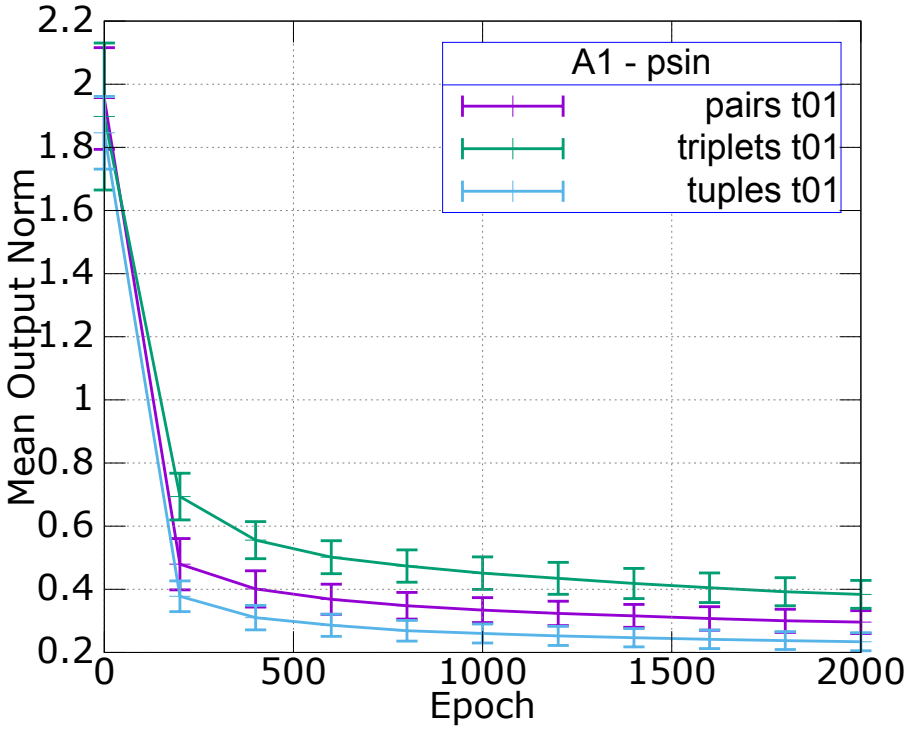


Figure 4.4 – Evolution of the mean output norm for Polar Sine Metric-based SNNs.

long trainings, with mean norms close to zero. However, it is important to note that this norm decrease facilitates training for the hypercube output space created by the hyperbolic tangent activation functions. As a consequence, overall norm values inferior to one enable every direction for any angle update with gradient descent.

The scalar product-based strategy proves more reliable in that case (see Fig.4.3), although it implies a longer training with 1600 epochs. Indeed, it is devised to counteract this norm variation phenomenon, and proves to be efficient, with mean output norms close to one, and small standard deviations for the triplets and tuples strategies. While the unit mean norm is not reached for the pairs strategy, it still stabilizes the norm evolution, validating hypothesis H_3, except for tuple and triplet objectives combined with a -1 negative target whose mathematical instabilities lead to increasing norms.

As a conclusion, angle updates with SNN networks imply a compromise between efficient angle corrections and output norms stabilisation, although this phenomenon does not produce major adverse effects in the case of a relatively small number of classes and similarity relationships.

Computation Cost and Complexity Analysis

Finally, we assess the strengths and weaknesses of the three proposed training set selection strategies in terms of computations. As seen above, we have to separate two kinds of complexities. The first one, is closely related to the number of relationships present in the cost function, recapitulated in Table 4.2. The second one, presented in Table 4.3, consists in the memory load and the number of backpropagations and updates necessary to present to the model every relationship available given a reference sample.

Firstly, we can observe that the tuple-based strategy proposes updates which are more balanced in terms of relationship representations. While the cosine and scalar-based cost functions only present the relationships considering a reference class, the Polar Sine Metric-based cost function makes every negative relationship update equivalent, as every relationship known in the dataset is available.

Hence, we observe a second trade-off between the required number of updates and the computation complexity of each update. In this case, the Polar Sine Metric-based SNN requires twice the number of epochs of the cosine-based SNN, with 1600 epochs against 800, to reach its optimum configuration, as seen in Figures 4.5 and 4.7. It can also be noted that the scalar product-based SNN classification score still increases after 1200 epochs (see Fig.4.6), but at a rate which prohibits a realistic training time to reach the same score as the other two cost functions.

The cosine and scalar-based cost function with the tuples strategy imply an error computation complexity which is linear in terms of output dimension N_O and number of classes N_c, i.e. a complexity in $O(N_O.N_c)$. While the Polar Sine Metric cost function involves a matrix inversion, the size of the latter stays quite limited, as it cannot exceed the number of classes available for the classification task. Moreover, this matrix is symmetric, positive-definite, which can ease memory usage and computations with specific, optimised matrix algorithms, such as the Lower-Upper (LU) decomposition with Cholesky factorisation, easing the determinant computation, limiting the dimensionality of the problem as well as improving numerical stability. Thus, the global complexity of the error resides in the matrix multiplication between the inverse cosine matrix $\mathbf{S^{-1}}$ and the normalised output matrix $\mathbf{A_{norm}}$, as seen in Appendix A.2.3, for a final complexity in $O(N_O.N_c^2)$. As such, the Polar Sine Metric-based method can stay competitive for the most common small to medium scale problems. One approach for

	pairs	triplets	tuples
cos	1	2	N_c
scal	1	2	N_c
psine	1	2	$\frac{N_c(N_c-1)}{2}+1$

Table 4.2 – Number of relationships represented in the cost function per update given a reference sample, with N_c the number of classes.

	pairs	triplets	tuples
number of parallel networks/update	2	3	N_c+1
number of backpropagations/update	$2N_c$	$3N_c$	N_c+1
number of updates	N_c	N_c-1	1

Table 4.3 – Number of parallel networks, backpropagations and updates necessary for each training set selection strategy to present every available similarity relationship given a reference sample, with N_c the number of classes.

larger scale problems would be a decomposition into smaller sets of relationships in order to limit the amplitude of N_c.

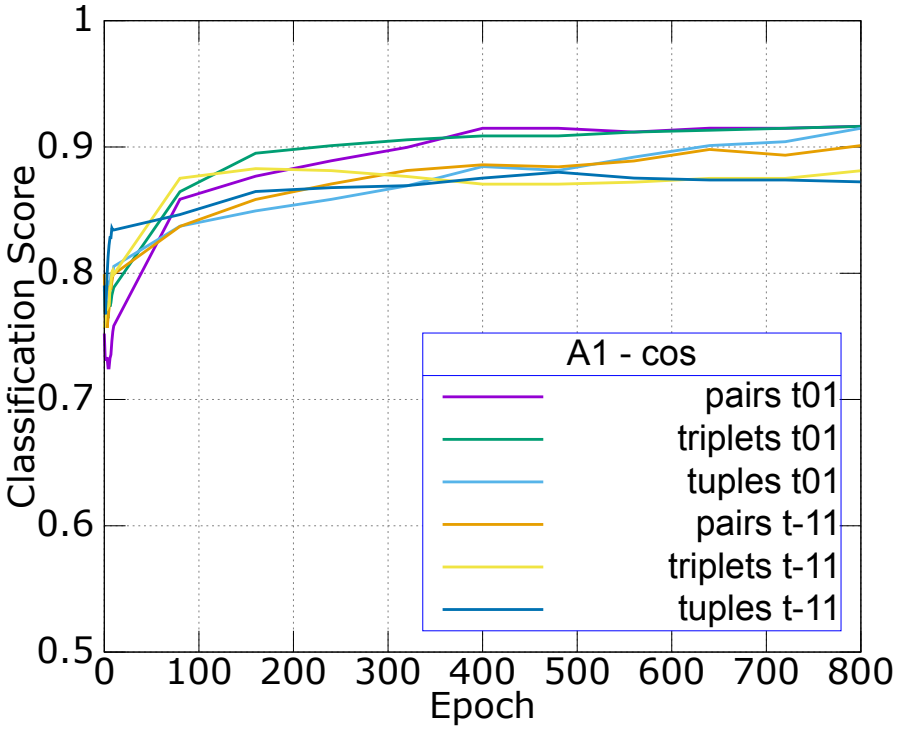


Figure 4.5 – Evolution of the Classification score for the cosine-based SNN.

For information purposes, Table 4.4 reports the average training times for one update, for each training set selection strategy and cost function studied above. These times were obtained with an Intel(R) Core(TM) i7-4800MQ CPU @ 2.70 GHz. Computation parallelism was not implemented, except for the matrix computations which were performed with the LAPACK library [ABD+90].

We can see that the order of magnitude identified above are respected, with a triplets strategy taking about 20% more time than the pair strategy, while the tuples

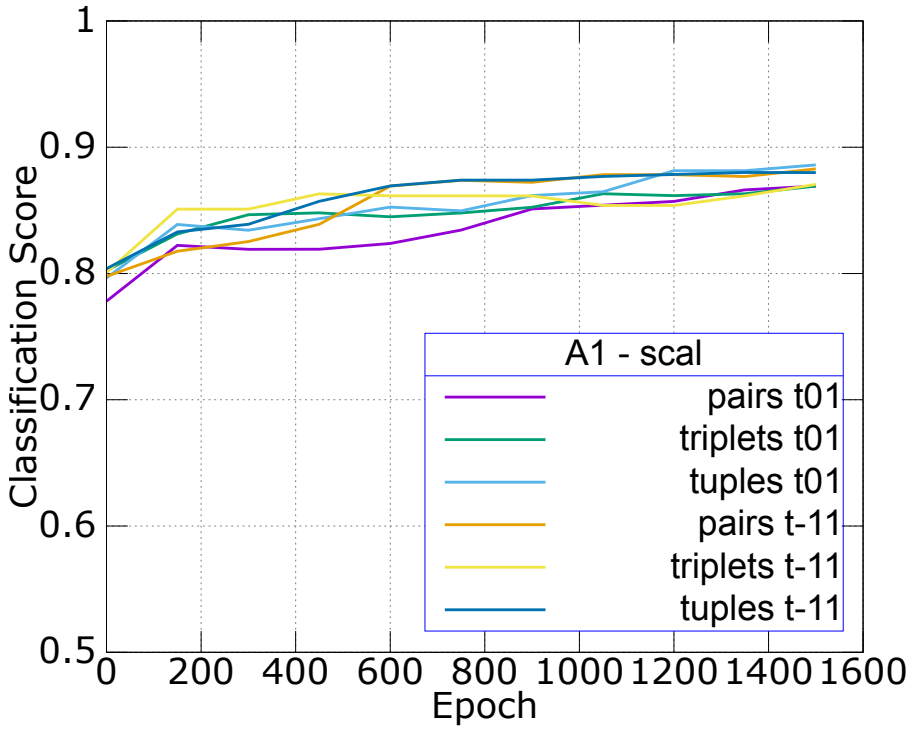


Figure 4.6 – Evolution of the Classification score for the scalar product-based SNN.

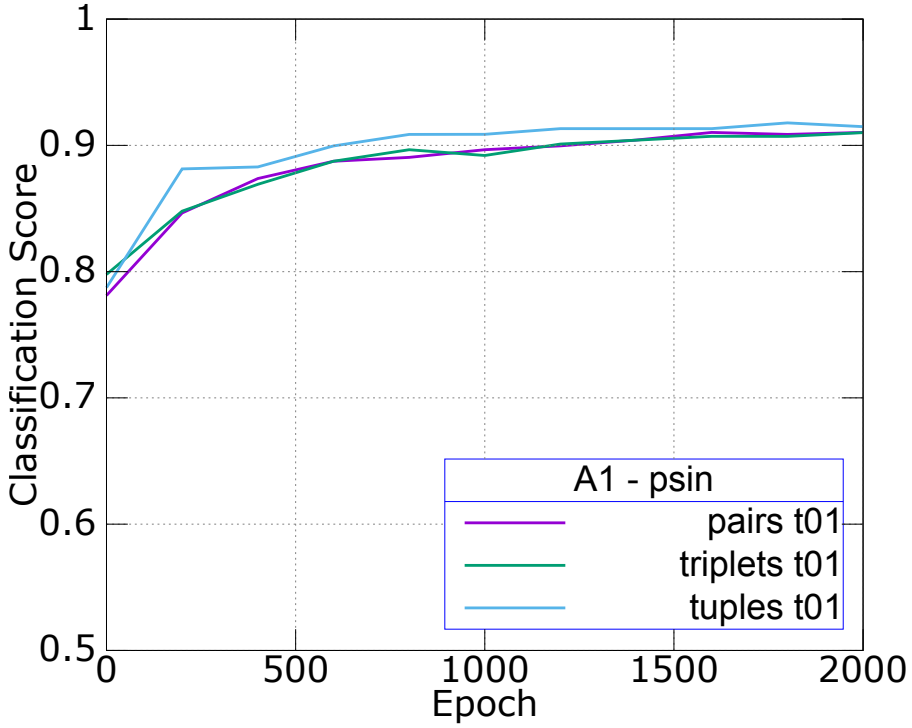


Figure 4.7 – Evolution of the Classification score for the Polar Sine Metric-based SNN.

A1	pairs	triplets	tuples
cos	$0.300823 \pm 1.21945.10^{-2}$	$0.353983 \pm 1.01395.10^{-2}$	$1.60702 \pm 9.48366.10^{-2}$
scal	$0.317392 \pm 8.24548.10^{-3}$	$0.386716 \pm 8.50045.10^{-3}$	$1.71631 \pm 5.94483.10^{-2}$
psine	$0.273950 \pm 9.81730.10^{-3}$	$0.328152 \pm 1.95091.10^{-3}$	$2.05181 \pm 1.17176.10^{-1}$

Table 4.4 – Time for one update iteration (in milliseconds) for the different cost functions and training set selection strategies on the MHAD protocol.

strategy increases the update time by more than 400%, which is still interesting with a total number of updates divided by the number of classes, equal to 11. Finally the Polar Sine Metric-based times does not reflect the increased complexity, as the LAPACK library was designed for efficient matrix computations and parallelism.

Thus, thanks to tests on the MHAD, every choice for the SNN variants was evaluated, from cosine targets, to training set strategies and cost functions. The first notable results reside in the negative target, which offers a much more stable convergence and better scores when it is set to zero, meaning output vectors from dissimilar samples tend to orthogonality. Moreover, a tuple-based training set selection strategy shows a comparable performance to pairs and triplets for the cosine-based cost function, while it improves the scores for the scalar and Polar Sine Metric-based ones, validating our approach. While the scalar product-based cost function stays behind with a recognition rate of 88.6% compared to the other methods around 91.5-91.6%, it was proved to efficiently stabilise the norm instabilities identified in these other methods. Finally, the complexities for every variant were studied.

Lastly, we provide a comparison between our results and the literature in order to prove the viability of Siamese networks as classifiers.

Comparison with state-of-the-art results

Thus, we compare the performances of each SNN objective function variant, with a tuple-based training selection strategy, whose choice is the result of the validation of our hypothesis H_2. The three tested Siamese configurations are denominated *SNN-cos*, *SNN-scal* and *SNN-psine*, depending on their respective objective functions based on the cosine, the scalar product, and the Polar Sine Metric. These results are available in Table 4.5. For our analysis, we focus on the classification rates obtained from isolated sensor data, with the accelerometer A_1 and motion capture sensor M_{20}.

Indeed, Cumin *et al.* [CL16] propose to tackle the challenge of human action recognition using multiple data sources, and test their approach on the MHAD, using the same leave-one-out protocol as the one we adopted. Although they reach a classification rate of 99.85% thanks to decision fusion methods from three classifiers, namely MLP, DTW and SNN with K-NN ($K = 1$), results are available for single sensors. Chen *et al.* [CJK15] use a fusion of accelerometer and depth maps data, for a final mean classification rate of 99.54%. However, a SVM approach based on A_1 is available.

The SNN-based approaches show globally superior results on the inertial sensor A_1, with a lowest score of 88.6% for the SNN-scal, compared to a best score of 86.7% for the SVM approach. This proves that a SNN-based approach is challenging. This conclusion is verified for the M_{20} sensor. While the MLP show an increased classification rate from 79.0% with A_1 to 91.3% with M_{20}, higher than the results for SNN-cos and SNN-scal, respectively equal to 90.6% and 91.0%. However, our SNN-psine ap-

	A_1	M_{20}
DTW [CL16]	0.790 ± 0.107	0.888 ± 0.076
MLP [CL16]	0.818 ± 0.099	0.913 ± 0.067
SVM [CJK15]	0.867	
SNN-cos	0.915 ± 0.102	0.906 ± 0.080
SNN-scal	0.886 ± 0.079	0.910 ± 0.065
SNN-psine	0.918 ± 0.091	0.924 ± 0.068

Table 4.5 – Comparison between state-of-the-art methods on the MHAD.

proach, implementing the Polar Sine Metric and tuple-based set selection strategies contributions, show the best result of 92.4%.

After this characterisation of each SNN variant, we focus on the classification performance of the SNN compared to state-of-the-art methods, and assess the rejection capabilities of such a model (hypothesis H_5), thanks to a dataset specifically gathered to target the problem of inertial symbolic gesture classification and rejection with embedded devices such as Smartphones.

4.2 Inertial Symbolic Gesture Classification and Rejection

In this section, we study the behavior of the SNN for inertial 3D symbolic gesture recognition, with novelty detection and rejection. The context of the experiment is the case of a Smartphone user gesture-based interface, where shortcuts to applications may be provided with gestures performed in the air, with the device in hand. This problem presents all the main characteristics of an open-world problem, where it is not possible to collect data from every targeted user. It is then mandatory to detect and isolate unbound or not-learned gestures to prevent any unwanted trigger during the interaction.

Indeed, rejection is only rarely taken into account by existing methods, or is taken care of by another model specifically trained for this task. Here, once the SNN is trained, the output layer provides a feature vector representing a similarity measure of a set of samples, measure whose consistency should allow for a more selective rejection.

In the light of the state-of-the-art, we assess the classification and rejection potential of the SNN when building a non-linear similarity metric between sets of samples.

At the beginning of the study, no public dataset was available for 3D symbolic gesture recognition to our knowledge. Thus, in the following, we first introduce the gesture database gathered to test our algorithms for gesture recognition. Secondly, two protocols designed to cover the main open-world aspects are developed. Then, criteria are proposed to evaluate the performance, combining both classification and rejection. Finally, conclusions are drawn from the comparison results.

4.2.1 Database Introduction

Two datasets were formed, based on the accelerometer and gyrometer data from the Android Samsung Nexus S device, sampled at 40Hz. The first dataset, named DB1, comprises 40 repetitions of 18 different classes performed by a single individual,

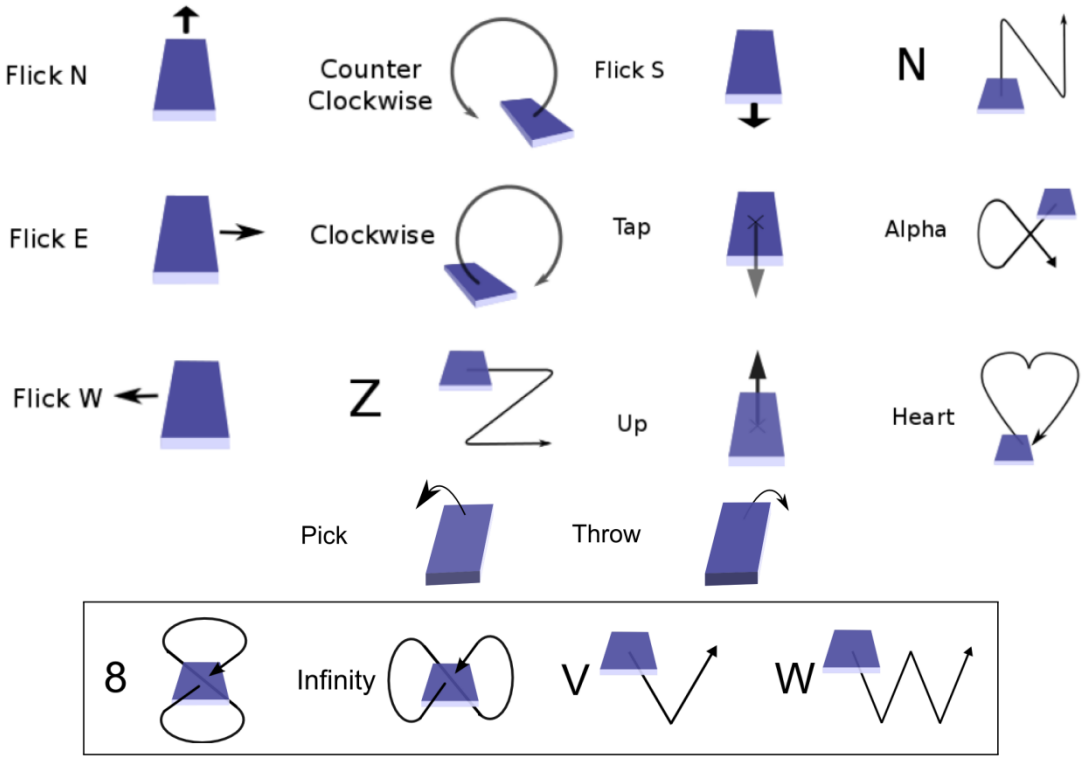


Figure 4.8 – Representations of the gestures in the Orange Dataset.

for a total of 720 records. Indeed, DB1 is the representation of a personalisation context, where the model is fitted to one user. Conversely, DB2 gathers 5 repetitions of 14 gesture classes performed by 22 individuals, for a total of 1540 records. DB2 corresponds to an open-world testing with multiple users. The 14 classes in DB2 encompass gestures with different complexities (see Fig.4.8). They are composed of linear gestures, with horizontal (*flick North*, *South*, *East*, *West*) and vertical (*flick Up*, *Down*) translations; curvilinear gestures (*clockwise* and *counter-clockwise* circles, *alpha*, *heart*, *N* and *Z* letters, *pick* gesture towards, and *throw* gesture away from the user). The 4 additional classes in DB1 are chosen to be close to some of the gestures cited above: the number *8* is close to both *clockwise* and *counter-clockwise* circles; the *infinity* gesture presents the same confusions in an horizontal plane, and with the gesture *alpha*; finally the letters *V* and *W* can easily be confused with vertical translations as well as the letter *N*.

The same preprocessing steps as in 4.1.1 were applied to the data, that is, an amplitude scaling-based on the maximum norm over the samples of a gesture; a low-pass filter with a parameter $\beta = 0.7$; and a 45-length resampling of the inertial samples based on the curvilinear length. An example of accelerometer signals for the gesture *Flick East* for one user, before and after preprocess, is available in Annex C. The signals from the three axes x, y and z, corresponding to the Figure 2.1, are successively represented, with a scale where $1000 = 1g$, given $g = 9.8m.s^{-2}$ the acceleration due to gravity.

4.2.2 Protocols

Two sets of experiments are performed on both datasets. The first one consists in assessing the classification performance of the SNN, while the second sets aims at studying the rejection potential of this model, according to hypothesis H_5. As seen

in Section 4.1.4, the tuple training strategy with a zero negative target provides the best results. As such, three variants of the SNN are tested, called "SNN-cos", "SNN-scal" and "SNN-psine", depending on their respective cost functions 3.23, 3.30 and 3.41. The same architecture is selected for every variant, with a 270-neuron input layer, one 45-neuron hidden layer, and one 90-neuron output layer. The hyperbolic tangent is classically chosen as the activation function for every neuron. The training set strategy is based on tuples, where each sample from the training set is selected once as a reference, with one additional sample from every class selected as the positive and negatives examples.

In the following, state-of-the-art models selected for comparison and test protocols are exposed.

4.2.2.1 Classification tests

The classification tests serve as a direct comparison with the different state-of-the-art strategies for gesture recognition, in a controlled framework with a closed gesture vocabulary. Every performed gesture is assumed to belong to this vocabulary, and a label is immediately assigned.

As the most common framework, classification tests are the most direct and efficient comparison with the literature. Every gesture recognition approach is represented in our comparative study: statistical-based methods with continuous Hidden Markov Models (cHMM) [Pyl05]; geometric-based methods with Dynamic Time Warping (DTW) [Pet10] and Principal Component Analysis (PCA); classifier-based methods with the classical Support Vector Machine and Frame-based Descriptor Support Vector Machines (FDSVM) [WPZ+09]; and neural-based methods with the MultiLayer Perceptron (MLP), Bi-directional Long Short-Term Memory Neural Network (BLSTM) [LBMG15] and Convolutional Neural Network (CNN) [DBLG14].

The best parameters and configurations for each method were identified thanks to preliminary experiments:

- cHMMs follow a left-to-right architecture, are composed of 9 states with 3 forward connections using a Gaussian emission distribution, while classification is operated on a maximum-likelihood criterion;
- the DTW-based method is associated to a 5-nearest-neighbour classifier, with the vectorised gesture data;
- the PCA-based method is associated to a 1-nearest-neighbour classifier, using the 90 eigenvectors linked to the highest eigenvalues for reconstruction;
- the SVM-based method is applied to the 270 fixed-size vectors with a linear kernel;
- the FDSVM-based method extracts 19 features over 9 time-windows, for a total of 1026-dimensional feature vectors;
- the MLP classifies the 270-feature vectors on a maximum activation criterion: inputs are processed by a 45-neuron hidden layer with a hyperbolic tangent activation function, and a 14-neuron "softmax" output layer;
- the BLSTM-based model is composed of two 100-neuron LSTM networks, with a majority vote over the classifications at each time step, and works on the vectorised gesture data;
- the CNN is applied to the 270-feature vectors presented as a 45×6 image, using temporal convolutions with a 10-10-65-65-14 network, with two stacks of

convolutional and subsampling layers, followed by a fully-connected "softmax" output layer.

The classification tests rely on 4 protocols, named **C1** to **C4**, covering different real application settings. Each protocol is repeated 10 times so as to take into account the influence of the training and testing data selection. These 10 configurations are shared for every method during the tests. The 14 common classes to both DB1 and DB2 datasets were selected for these experiments.

— **C1**, based on DB1, covers the closed-world application with a single user and a controlled set of gestures, with 5 randomly selected samples per class for training, and 16 samples for testing;
— **C2**, based on DB2, corresponds to a multi-user closed-world application. Every user is represented in the training data, with 2 samples per class and per user used for training, and the 3 remaining samples for testing;
— **C3**, based on DB2, represents an open-world case, where every potential user cannot be identified. Thus, the training phase is based on every sample from 17 users, while testing is performed on the samples of the 5 remaining users;
— **C4** is the most challenging open-world scenario, with one user used as a reference, while tests are carried out on the 21 remaining users. While this protocol underlines the specificities in the gesture style of each user, it also assesses the generalisation capacities of each classifier.

These protocols also serve as a basis for the rejection study, where unknown classes are introduced.

4.2.2.2 Rejection tests

In a second phase, we center our study towards the rejection capacities of the SNN (hypothesis H_5). Two approaches are selected to compare the performance of this model. The DTW-based model stands out as the best immediate comparison with another similarity metric, while the MLP, as the neural-based classifier counterpart of the SNN, is the most natural choice to evaluate their performances as neural networks. With a K-NN classifier, a single distance threshold is applied on the nearest neighbour for the DTW and SNN-based strategies, which effectively reflects the relative projection of the test and training samples. Classically, the rejection criterion of the MLP consists of a threshold on the maximum activation, also interpreted as a posterior probability.

The goal of this experiment is to characterize the performance of the abovementioned models for two types of rejection: incorrect classifications, where classification errors have to be identified; and unknown classes. In the first case, the main challenge is to isolate the misclassified samples before rejecting correctly classified ones, while the second kind of rejection concerns the "rest of the world" paradigm, which aims at evaluating a model performance in isolating elements it was not trained for from the rest of the known classes.

Thus, two protocols were designed for these two tasks.

— **R1** tests the rejection quality for misclassified samples, whose outputs are too far from the references. It is similar to **C4**: based on DB2, the samples from one user form the training set, while the data from the 21 remaining users form the test set. This allows for testing the generalisation potential of the trained model, as well as its capacity for rejecting samples whose variations from the reference individual are too important.

— **R2** is similar to **C1**: based on DB1, 5 samples for the 14 classes are used for training, and 16 for testing. However, every samples from the 4 remaining classes ("V", "W", "infinity", "8") are introduced in the testing set. This test embodies a realistic personalisation paradigm, used in a natural user interface where the user does not specify when they make a gesture, and the system has to determine whether to trigger an event even before selecting the corresponding one.

4.2.3 Evaluation Criteria (classification/reject)

Two different evaluations are performed in order to characterize the performance for each test.

For the protocols **C1** to **C4**, we report the accuracy of the tests, i.e. the ratio between the number of correctly classified samples and the total number of samples. Moreover, a confusion matrix is studied in detail for every protocol in order to identify the main error origins. For each protocol, the SNN variant showing the best results is selected for these confusion matrices.

For the rejection protocols **R1** and **R2**, we propose an evaluation of the rejection quality based on graphs similar to the Receiver Operating Characteristic: we follow the classification rate as a function of the rejection rate. The classification rate is defined as the ratio between accepted and correctly classified samples and the total number of accepted samples, while the rejection rate is defined as the ratio between the number of rejected samples and the total number of samples. Thus, a high rejection quality will see an early rising curve, which is why the total performance can be estimated thanks to the area under the curve. While each model can reach a 100% classification rate even for a 0% rejection rate for protocol **R1**, this value cannot theoretically be reached for protocol **R2** until the 41.7% rejection rate mark, which is the ratio of unknown samples in the test set. We also propose to further study the rejection quality for protocol **R2** with a graphical representation of the ratio of the different types of rejected samples as a function of the rejection rate. This allows for a visualisation of unwanted rejection, as well as the relative importance of misclassification and unknown classes rejections.

4.2.4 Results

In this section, results for the classification tests are presented, followed by the rejection protocols.

4.2.4.1 Classification tests

The different SNN variants perform in relatively the same way for each protocol in the classification tests, showing that the simpler, clean database allows each method to reach their full potential, with efficient angle updates. In the following, the results for each protocol is studied into more detail, and compared to the state-of-the-art.

Protocol C1

The general performance comparisons between the main models for gesture recognition are presented in Table 4.6. As results between the SNN-scal and the SNN-psine versions are quite similar and often in favour of the first version, the proposed confusion matrices are extracted from the tests using the SNN-scal.

	C1	C2	C3	C4
DTW	**0.995** ± 0.005	0.945 ± 0.002	0.911 ± 0.014	0.791 ± 0.016
PCA	0.992 ± 0.007	0.962 ± 0.008	0.923 ± 0.012	**0.793** ± 0.019
cHMM	0.992 ± 0.010	0.858 ± 0.013	0.817 ± 0.017	0.765 ± 0.023
SVM	0.961 ± 0.009	0.949 ± 0.008	0.913 ± 0.002	0.674 ± 0.037
FDSVM	0.964 ± 0.019	0.954 ± 0.006	0.924 ± 0.013	0.693 ± 0.041
MLP	0.978 ± 0.010	0.954 ± 0.006	0.905 ± 0.010	0.744 ± 0.040
BLSTM	0.868 ± 0.007	0.956 ± 0.005	0.926 ± 0.029	0.654 ± 0.010
CNN	0.979 ± 0.005	0.958 ± 0.009	**0.934** ± 0.015	0.787 ± 0.034
SNN-cos	0.988 ± 0.008	**0.969** ± 0.007	**0.934** ± 0.013	0.775 ± 0.032
SNN-scal	0.990 ± 0.009	0.968 ± 0.009	0.933 ± 0.014	0.766 ± 0.029
SNN-psine	0.987 ± 0.009	0.968 ± 0.006	**0.934** ± 0.011	0.776 ± 0.025

Table 4.6 – Recognition rates for symbolic gesture recognition protocols.

For protocol **C1**, PCA, DTW and cHMM methods present the best accuracies, with an advantage for the DTW at 99.5% accuracy. Indeed, neural-based methods are challenged by the relatively low number of training samples. The BLSTM shows a clear case of undertraining with an accuracy of 86.8%, while the CNN reaches a higher accuracy of 97.9%, thanks to its architecture which can offset some variations such as temporal misalignments. The different versions of the SNN show comparable results, with the highest scores for neural-based methods, which proves the coherence of the learnt projections. Overfitting is the main concern for this protocol, and it is easier to find the best configuration for the SNN-scal as convergence is slower. As shown in Table 4.7, the few errors come from gestures who show the same initial dynamic. For instance, the gesture "Alpha" is confused with "N" as both gestures share the same dynamics, with three vertical strokes. These vertical components are also found in the gesture "Clockwise". Other confusions come from gestures contained within others, such as "Z" and "FlickE", where "Z" actually starts with a translation to the right of the user. These errors might be the sign that the temporal normalisation introduces a bias. Indeed, the typical Smartphone accelerometer is not needed for precision tasks, and its range is thus limited. It is then possible to notice some time windows where the sensors saturate, producing the maximum output value, time windows which are reduced to a single sample with a the thresholding scheme. Complex gestures such as the ones above may be deformed, with one critical stroke being neglected.

Protocol C2

The SNN-cos shows the best accuracy for protocol **C2** of 96.9%, closely followed by both SNN-cos and SNN-psine with 96.8%, proving that the SNN performs well even when multiple, different gesture dynamics are involved. The third best score is obtained by the PCA-based method, as it can handle variations thanks to the projection on its principal eigenvectors, which orient the reconstruction towards known configurations. The majority of the other methods sees their accuracy gravitate around 95.5%.

A closer study of the confusion matrix for the SNN-scal in Table 4.8 shows small confusions between "N" and "Up", and "Alpha" and "Heart", which are indeed similar gestures. Moreover, a confusion between "Up" and "Pick" appears, which seems understandable as both gestures require a vertical movement upwards. As movements away from the user, the gestures "Throw" and "Flick North" show the highest error rate. An analysis of the source of these errors shows that the majority of these samples belong to

–	alpha	ccw	cw	flicke	flickn	flicks	flickw	heart	n	pick	tap	throw	up	z	sum
alpha	13								3						16
ccw		16													16
cw			16												16
flicke				16											16
flickn					16										16
flicks						16									16
flickw							16								16
heart								16							16
n			1						15						16
pick										16					16
tap											16				16
throw												16			16
up													16		16
z				1										15	16
sum	13	16	17	17	16	16	16	16	18	16	16	16	16	15	224

Table 4.7 – Confusion matrix for protocol **C1** for SNN-scal

–	alpha	ccw	cw	flicke	flickn	flicks	flickw	heart	n	pick	tap	throw	up	z	sum
alpha	43												1		44
ccw		42				1						1			44
cw	1		42	1											44
flicke				44											44
flickn					43								1		44
flicks						44									44
flickw							44								44
heart	1							43							44
n									44						44
pick										44					44
tap											44				44
throw					4							40			44
up										2	1		41		44
z	1													43	44
sum	46	42	42	45	47	45	44	43	44	46	45	41	43	43	616

Table 4.8 – Confusion matrix for protocol **C2** for SNN-cos

a unique user. Thus, this phenomenon underlines the fact that some users may have a really specific way of performing gestures, which, combined with the imprecision of the sensors, may result in a great difficulty to manage them with a single, general model not specifically trained for these singletons.

Protocol C3

The protocol **C3** amplifies the difficulties encountered with **C2**. The SNN-psine and SNN-cos take advantage of the bigger training dataset with an accuracy of 93.4%. In this configuration, neural-based methods such as CNN, BLSTM and SNN perform better than other statistical and geometric methods, which justifies the need for a high non-linearity in order to generalize to unseen users. In that case, the SNN (see Table 4.9) shows a high symmetric confusion between "Pick" and "Up" for the same reasons as for **C2**. It also handles badly the gesture "Throw". Indeed, this gesture, which consists in an arc away from the user, brought about fears of actually throwing the device, fears causing in turn the highest disparities between users. This result is all the more visible in **C4**.

–	alpha	ccw	cw	flicke	flickn	flicks	flickw	heart	n	pick	tap	throw	up	z	sum
alpha	22			1					1			1			25
ccw		25													25
cw			25												25
flicke				23		1						1			25
flickn					24								1		25
flicks						25									25
flickw	1	1					23								25
heart								25							25
n		1	1						23						25
pick										20		1	4		25
tap											25				25
throw				3	5						2	15			25
up		1								2			22		25
z		1												24	25
sum	23	29	26	27	29	26	23	25	24	22	27	18	27	24	350

Table 4.9 – Confusion matrix for protocol **C3** for SNN-psine

Protocol C4

Finally, the protocol **C4** presents the highest challenge for these methods, with a single user data for training. As a consequence, all the flaws identified above are reinforced, with results dropping respectively to 79.3% and 79.1% for the PCA and DTW-based methods. Geometric methods have the advantage over neural-based ones, which have to balance under and overfitting for this very limited training set. The BLSTM-based method was not developed on this protocol in [LBMG15], since the undertraining underwent in **C1** would prove all the more detrimental here. However, the architecture of the CNN allows for a better absorption of different gesture dynamics, following with an accuracy of 78.7%, while the SNN-cos and SNN-psine take a 1% advantage over the SNN-scal, with respective accuracies of 77.5% and 77.6% against 76.5%. The confusion matrix in Table 4.10 confirms the hypothesis of amplified flaws. The gesture "Counter-Clockwise" is easily confused with a "Z". Indeed, a reasoning on the horizontal plane show that both gestures are composed of three horizontal strokes, towards the right, the left, and right again. The "Up" and "Pick" gestures are a source of errors, as the difference between the two gestures essentially resides in the end of the movement where the "Pick" gesture adds a horizontal movement. The same approach can be used to understand the confusion between the "Up" and "Flick North". Indeed, during a "Flick North", the user involuntarily introduces a vertical movement as the arm advances forward, which can be mistaken for the vertical movement of "Up". The "Throw" gesture reveals all its complexity, representing 25% of the total number of errors. It is however important to notice that the classification of easier, horizontal gestures, such as translations, is satisfactory. Thus, most of the complexity and errors come from more elaborated gestures, for instance "Counter-Clockwise" or "Throw", and from vertical gestures, with "Up", "Pick" or "Tap".

Thus, many limitations were identified, from unique users to easy confusions between gestures where one can be identified as a part of the other. While a different strategy combining multiple classifiers may simplify the task of handling these outliers, rejection remains as a solution to isolate and prevent these errors.

–	alpha	ccw	cw	flicke	flickn	flicks	flickw	heart	n	pick	tap	throw	up	z	sum
alpha	82	1	1				1		3	1	7				105
ccw		69				5			3		2			26	105
cw	6	2	88	7					1		1				105
flicke	1	1	3	97	2		1								105
flickn				4	101										105
flicks						104							1		105
flickw	5				3	4	93								105
heart	1		1	1	1	1		99			1				105
n	4		2			1			91		6	1			105
pick						2	1			71		2	29		105
tap					1				4		84	16			105
throw	1		1	11	21				4	4	2	4	3		105
up			2		13	4	1			6	1		78		105
z	7	6			1		2		1		5			83	105
sum	107	79	107	12	143	121	99	99	107	82	127	59	111	109	1470

Table 4.10 – Confusion matrix for protocol **C4** for SNN-psine

4.2.4.2 Rejection tests (Hypothesis H_5)

Protocol R1

Protocol **R1** consists in studying the rejection behaviour for the most challenging case described above, with a training set comprising the data from a single user. Results are presented in Figure 4.9. While the DTW-based method takes profit from the temporal information while recognizing gestures, it is outperformed for error identification after the 36% rejection rate landmark, where the non-linear projection of the SNN-scal allows for a more efficient rejection. The SNN-cos and SNN-psine show a slightly inferior score, while staying more efficient than the MLP-based method. This difference can be explained by the potential overfitting of the SNN-cos and SNN-psine. Thanks to its regularisation, the SNN-scal learns every class in a more uniform manner, with a more consistent intra-class distance over all classes, which in turn favours the rejection strategy based on a single threshold.

Protocol R2

Protocol **R2** shows the better capacity of the SNN to isolate unknown samples. As seen in Figure 4.10, in a case based on a single user where overfitting does not threaten generalisation over the testing set in the same manner, every version of the SNN proves to be superior to the DTW and MLP-based methods. Around the 41.7% landmark 4.11, which is the theoretical minimum rejection rate which would allow the rejection of every unknown sample, the SNN variants present a classification rate of 94.4%, while the DTW and MLP-based methods get lower respective scores of 92% and 88%.

A more detailed analysis of the rejection can be conducted thanks to Figure 4.12. The evolution of three types of rejection is followed as the rejection rate increases. Rejected misclassifications and samples from unknown classes form the correct rejection, while rejected samples which would have been correctly classified form the wrong rejection. It is very interesting to observe that the SNN-based methods present the lowest area for the mean wrong rejection rate, with a steady rejection rate that only starts to degrade after the 41.7% landmark. This shows their greater selection ability compared to the other two methods where the degradation is a lot more spread with the increase of the rejection rate. The gap with the perfect rejection model, where the area under the perfect rejection line would be dedicated entirely to unknown classes rejection, is also a lot smaller for the both SNN variants. Indeed, the non-linearity of

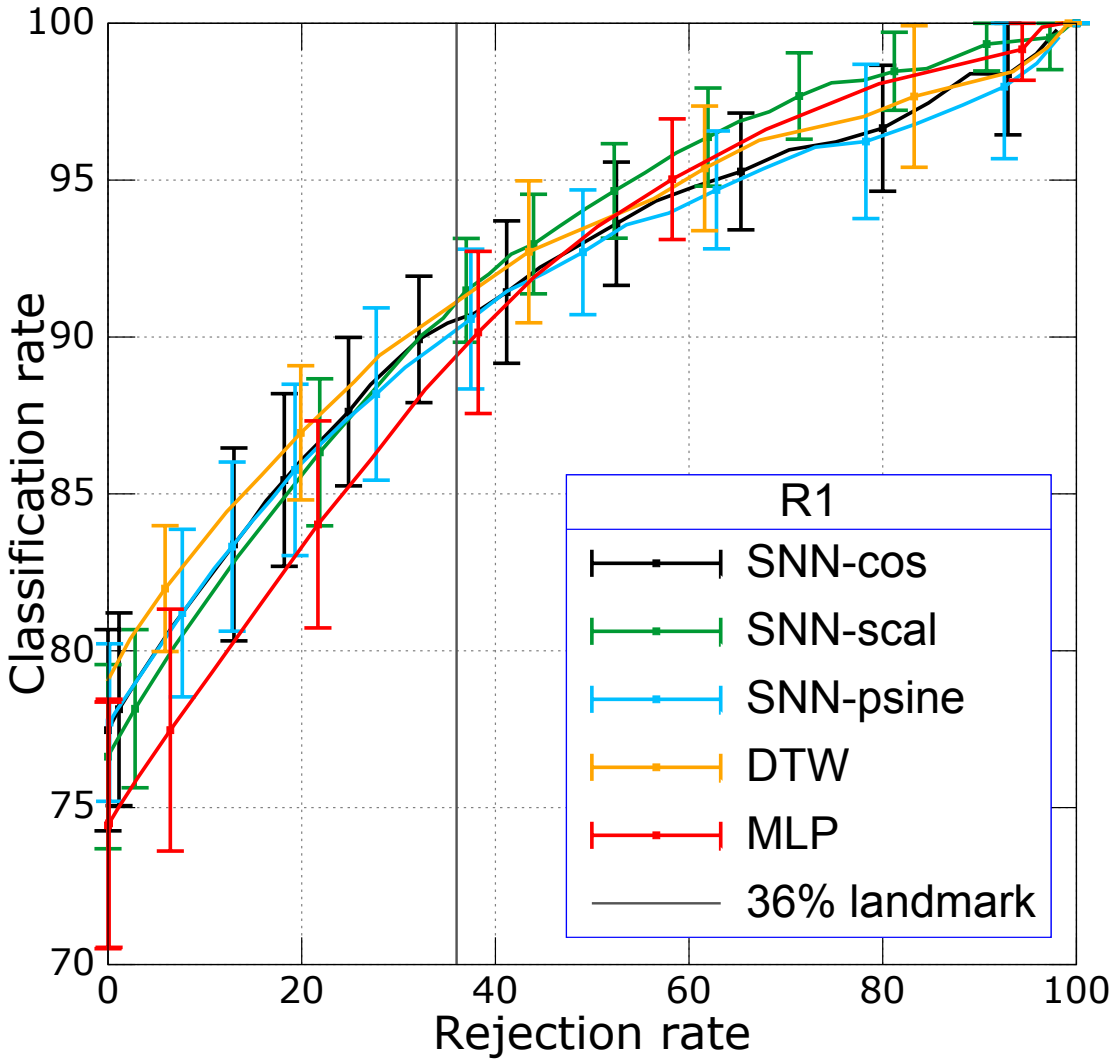


Figure 4.9 – SNN-scal, SNN-psine, DTW and MLP comparison on **R1**.

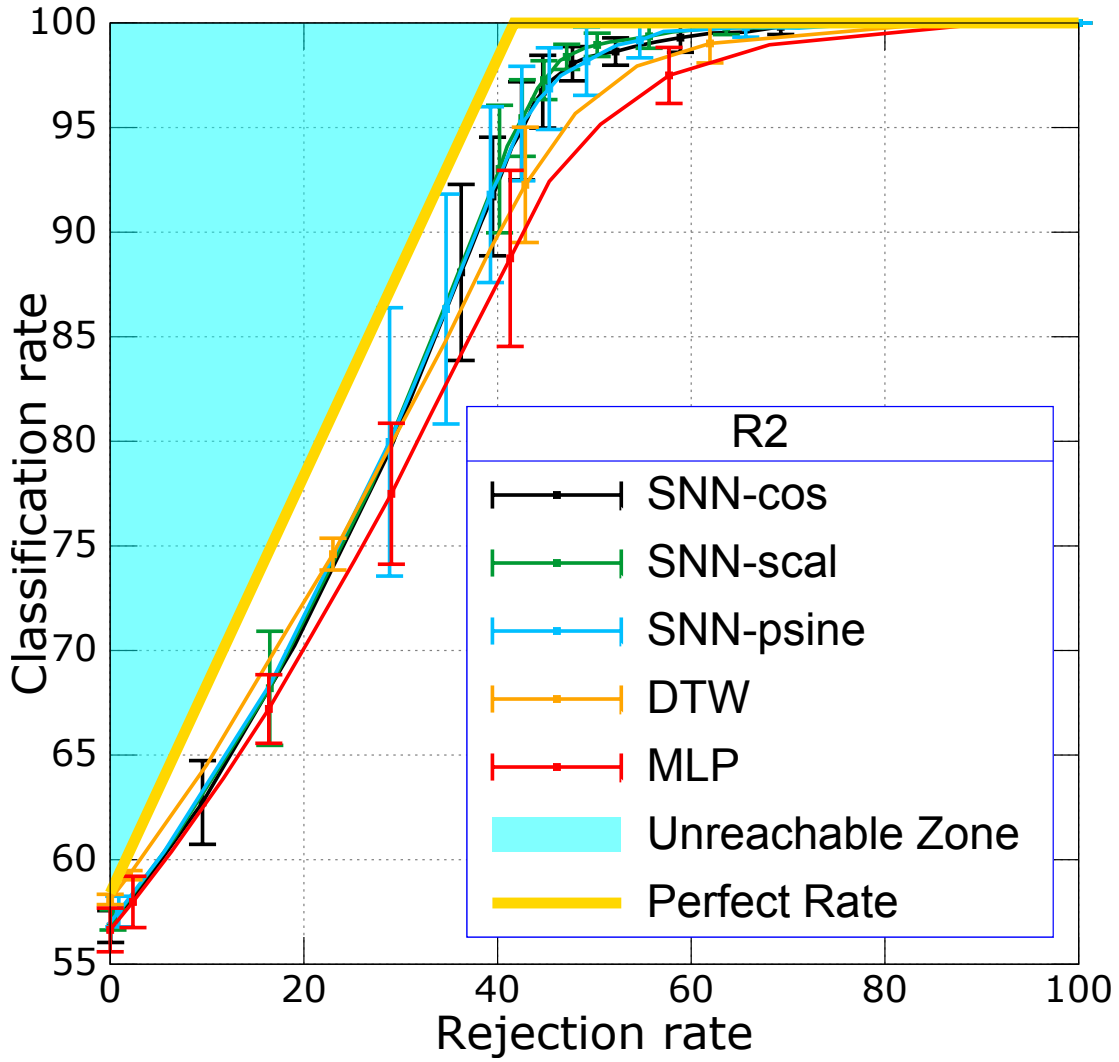


Figure 4.10 – SNN-scal, SNN-psine, DTW and MLP comparison on **R2**.

the projections allows for a more even intra to extra class ratio, favouring the single threshold rejection strategy. The SNN-scal show a zero-incorrect-rejection for smaller rejection rates, while the SNN-psine and SNN-cos still reject correctly classified samples, which is another sign of their limited error selection, where more complex classes cannot be learnt as well before overfitting.

To conclude, we have shown the validity of the SNN-based methods for classification and rejection, with a single-user most favourable case. Thus, these methods can easily be considered for a realistic personalisation paradigm, where the interface is suited to one user in particular, and is also required to handle unseen gestures. Moreover, the SNN-scal shows state-of-the-art result on a more general closed-world case, where multiple known users may use the device with a single model. The SNN-scal strategy proves to be more versatile in a rejection problem, with its norm regularisation preventing an overfitting, which allows a uniform training of every class. Conversely, the SNN-cos and SNN-psine present more accurate classification scores, to the detriment of rejection, with a training less adapted to a single threshold rejection. This drawback

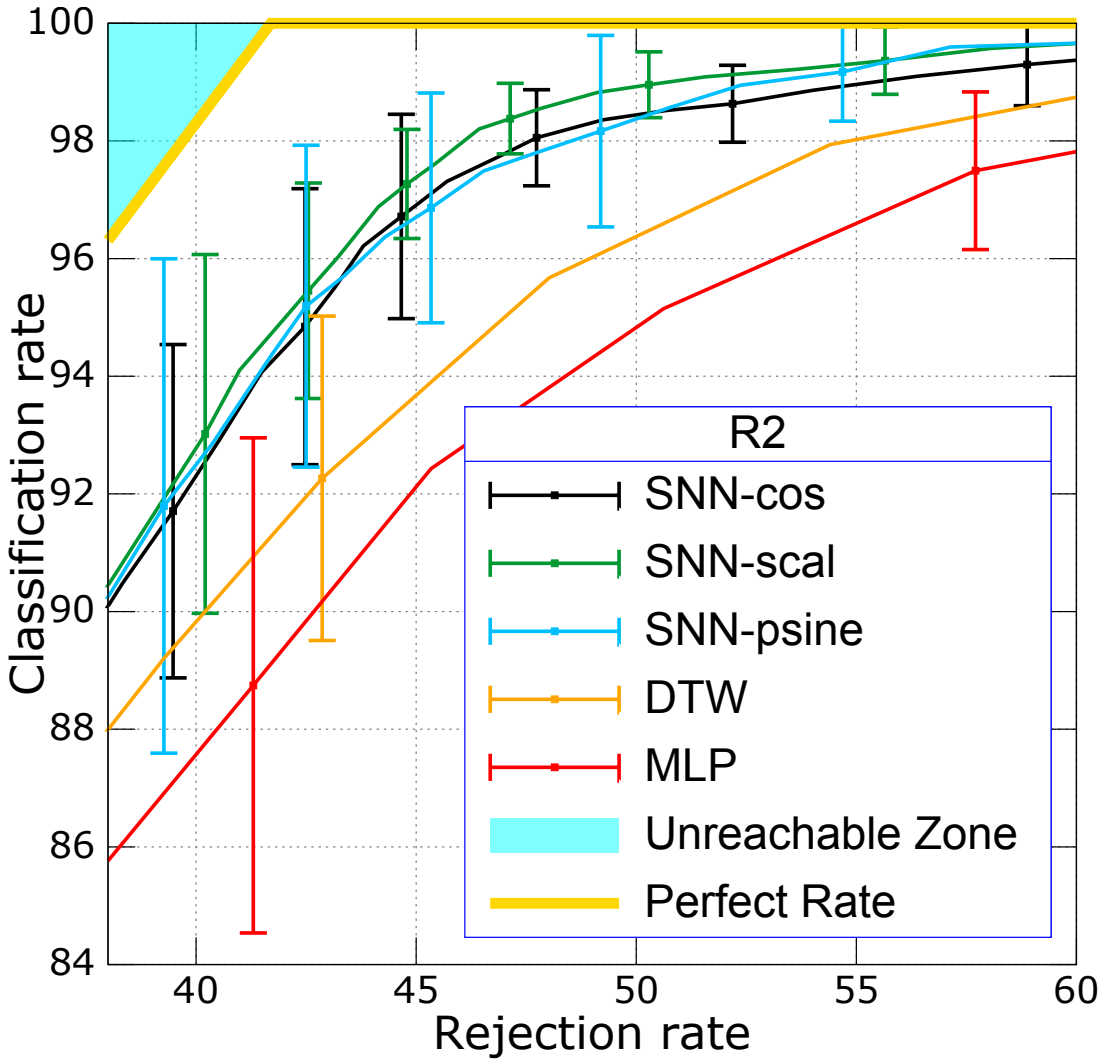
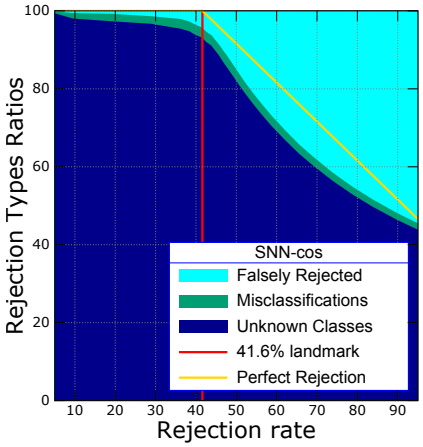
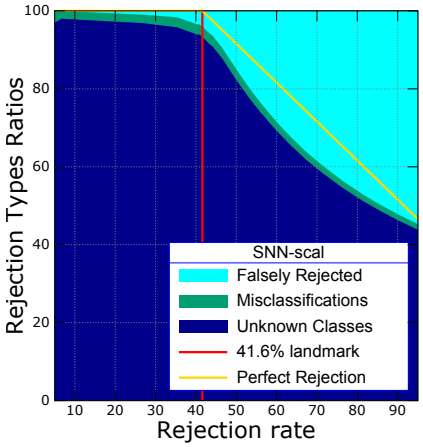


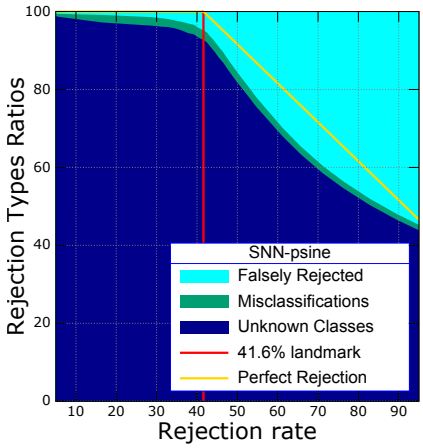
Figure 4.11 – Close-up of Fig.4.10 around the 41.7% landmark.



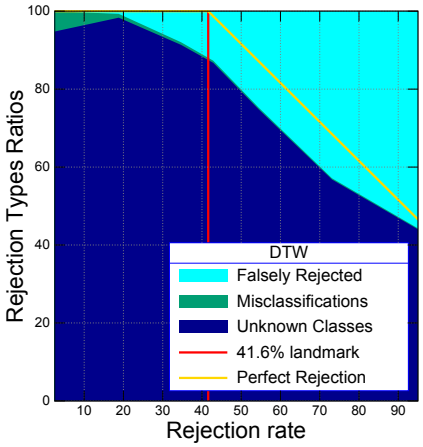
(a) SNN-cos rejection details on **R2**.



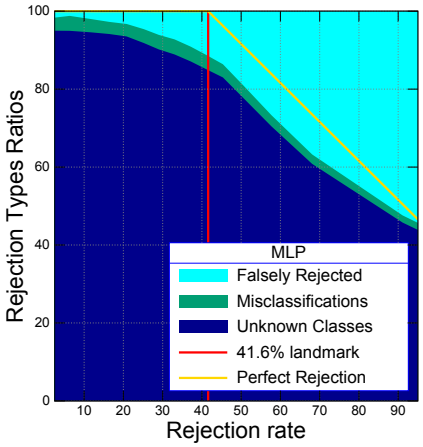
(b) SNN-scal rejection details on **R2**.



(c) SNN-psine rejection details on **R2**.



(d) DTW rejection details on **R2**.



(e) MLP rejection details on **R2**.

Figure 4.12 – SNN-scal, SNN-psine, DTW and MLP rejection details on **R2**

may be corrected with class-specific thresholds, in order to take into account different intra-class distances.

4.3 Conclusion

We have shown in this chapter the potential of the SNN in its different configurations thanks to the MHAD and Orange Dataset. We have proved that a training based on multiple samples covering the whole ensemble of known relationships can improve and speed up training. Moreover, we have studied the impact of our two variants, with the scalar-based cost function with norm regularisation, and the Polar Sine Metric-based cost function. On the one hand, while the scalar-based cost function allows for a control in the output vector norm variations, its angular correction divided in three subtargets limits its potential for complex datasets. On the other hand, the Polar Sine Metric-based cost function proves to be a valid and reliable generalisation of the sine function to measure dissimilarities, and provides a standardised approach to handle variable numbers of relationships, at the cost of a higher complexity of the error computation. The potential of the SNN was also studied for realistic Smartphone-based gesture recognition problems, with tests performed on the Orange Dataset, covering multiple realistic scenarios in classification and rejection. The SNN-based methods show competitive, state-of-the-art results in classification, especially with higher size training sets; as well as in rejection, outperforming its respective neural and elastic metric counterparts, the MLP and DTW-based methods, in novelty detection and rejection.

Chapter 5

Conclusion and Perspectives

This last chapter presents a summary of our contributions and the associated tests destined to assess their behaviour and performances. Then, limitations of our research are developed. Finally, multiple avenues for research are proposed for the development of the potential of the SNN for Similarity Metric Learning; and for its application for Gesture Recognition.

5.1 Conclusion

In this work, we present the problem of inertial gesture recognition as a potential solution for low-cost, richer interfaces and interactions in an ever-growing ubiquitous computing context. Indeed, recognising gestures or activities from inertial sensors is not straightforward, as this process is disrupted by the sensors limitations and variations in realisation between users. However, while this task has already been approached with classical machine learning methods, every necessary component for a complete interface was not considered. Thus, we focus our study on classification and rejection methods applied to gesture recognition, in order to fill the void in the interaction where the user would perform a gesture not destined to the machine and it is up to the model to determine whether an action should be triggered or not. Hence, we propose to perform an automatic non linear metric learning on accelerometer and gyrometer data, with an artificial neural network based approach inspired by the Siamese paradigm. While classical neural networks are specifically trained for classification, the Siamese Neural Network operates an automatic feature extraction thanks to a training based on multiple samples and their similarity relationships.

Firstly, we propose an adaptation of the Siamese strategy to a multi-class classification context for a stochastic training. Beyond the typical pairs and triplets of samples labelled as similar or dissimilar, a generalised training set selection strategy is suggested, which leverages one sample from every class, effectively simplifying the constitution of these training sets by balancing the representation of every relationship. Secondly, following the choice of the cosine similarity metric for the output space, mathematical flaws are identified, in particular concerning the uncontrolled norms of the outputs during training, leading to a second contribution. Simplifying the original cosine function, angle updates are decomposed in three independent targets for a pair of vectors, namely a target on the scalar product between these vectors, and a target on their respective norms. Finally, we go into multi-class relationships definitions in depth, and proposed a unified similarity function, the Polar Sine Metric, which holds

and represents all the available information about dissimilarities in the training set, and leads to a supervised, stochastic non-linear Independent Component Analysis learning.

Two sets of experiments are then performed in order to assess the performances of the different cost functions. The first set of experiments is based on the Multimodal Human Activity Dataset, with 11 actions performed by 12 subjects. This datasets allows for a comparison between cosine-based, scalar-based and Polar Sine Metric-based SNN variants, as well as comparisons with methods based on different modalities. We show that an orthogonality objective for negative pairs of samples is more stable and produces better classification rates than a repulsion scheme with a target equal to -1 for the cosine value. Moreover, target functions including the actual cosine, i.e. the cosine and Polar Sine Metric, possess a higher discrimination potential than the scalar-based objective function, which suffers from an angle definition divided in three subtargets. However, a norm evolution study during training shows the uncontrolled behaviour of the mean output norms, which tend to decrease with each update; while the regularisation term introduced in the scalar-based objective proves to stabilise that behaviour, which may be beneficial for longer trainings. Finally, a comparison between pairs, triplets and tuples training set strategies show that the tuple-based strategy conciliates the best results for every objective variant, with a roughly same score for each training strategy with a cosine-based objective. The tuple-based training set selection strategy comes to the price of a higher number of epochs to reach the best configuration, with fewer, more computational updates.

Then, a second, Smartphone based dataset, composed of 18 symbolic gestures from 22 users, was gathered in order to represent specifically the embedded device interaction paradigm at the centre of our study. Both classification and rejection problems are tackled. Multiple classification schemes are proposed, reflecting the different plausible interaction frameworks, from the single user in a "closed world", with a limited and defined gesture vocabulary, to the open world, where the number of users is not specified and the model has to generalise. We then show that our SNN approach, coupled with a one-nearest-neighbour classifier, proves to be competitive against state-of-the-art methods. In a rejection problem, we bring to light the higher novelty detection capacities of the SNN network over its neural and distance based counterparts, respectively the MLP and DTW.

5.2 Research Limitations

The principal limitation of this study consists in the lack of a dataset large enough to represent a real generalisation problem. Indeed, gesture recognition as an interface solution would need to encompass the different representations of each gesture, which cannot be perfectly represented with a limited number of users.

Thus, once again, it is important to bear in mind that the tests performed aimed at identifying the best performances for each model in order to assess their potential. As such, it is not a realistic assessment of a real-world application behaviour. This limitation was induced in particular by the relatively small amount of data available, which prevented the constitution of the conventional validation set.

A potential solution would be to expand the current Orange Dataset, or identify a future, larger dataset closely related to our problematic.

5.3 Perspectives

Two main perspectives are envisioned. First, continued research is possible on the SNN. Secondly, different strategies can be applied to improve gesture recognition based on our results.

5.3.1 Continued research on SNNs

In this section, we propose leads for different solutions to the identified problems, in particular norm regularisation, designed to compensate for the drawbacks of our contributions identified in this study. Secondly, some evolutions are suggested for the SNN.

Thus, the norm regularisation scheme proposed with the scalar based cost function presents the drawback of compromising between output norm and angle correction. Hence, a solution would be to integrate the angle correction directly in the norm regularisation, since the proposed model is one epoch late for norm correction: the norm to be regularised should be the one of the updated sample, rather than the one from the current projection. In Appendix A.3 is presented a possible solution which takes into account the angle correction in order to improve training. It is also be possible to consider a two-stage training, which would require to duplicate every update. Indeed, the first update, while temporary, is performed normally, and the second one takes into account the norm of the updated outputs to propose a more accurate regularisation for the initial network. Finally, a control over the norms may be performed manually with the help of an added linear layer. Indeed, any multiplication or division of the weights would not impact the final cost.

Moreover, the Polar Sine Metric proposes a matrix approach to describe relationships, but relies on a determinant to compute the final dissimilarity for a set of samples. This matrix definition may also be used to generalize the cosine cost function, including between-negatives relationships, and performing a gradient descent on the Frobenius Norm of difference between the identity $\mathbf{I}$ and the cosines matrix $\mathbf{S}$ for a training set T:

$$Cost_{Frob}(T) = \|\mathbf{I} - \mathbf{S}\|_F \tag{5.1}$$

The SNN structure itself may be modulated to adapt better. First, the Siamese network is above all a training strategy based on the relationships between sets of samples. Thus, it is possible to consider different architecture for the SNN than the proposed one based on a MLP. Indeed, the Siamese training strategy may be seen as a batch training, where the error is added for the different samples constituting the training set. As such, any classifier taking fixed inputs should be adaptable to this strategy, with Siamese CNN for instance.

In the context of temporal series such as inertial sensors data, a great development would reside in the temporal dimension of the SNN network. However, the adaptation of the Siamese strategy is not straightforward for temporal networks, such as the recurrent or LSTM neural networks, as the network output dimensions are not determined, which complicates the definition of a similarity criterion in the output space. Indeed, the objective function relies on a pre-defined, controlled similarity metric, learned in the produced output space, yet a temporal network would introduce a new dimension to the outputs, related to the length of the input signal. As such, it is necessary to

consider similarity metrics between two matrices which only share one dimension. One intermediary solution would be to normalize the inputs in time, and suggest a similarity criterion based on matrices formed by the successive outputs. This would be possible for the cosine metric, which only depends on the definition of a scalar product. For example, given the matrices $\mathbf{A}$ and $\mathbf{B}$, a common scalar product consists in the value of the trace of the product between one matrix and the conjugate of the other:

$$< \mathbf{A}, \mathbf{B} >= tr(\mathbf{A}.\mathbf{B}^H) \tag{5.2}$$

This scalar product also defines the Frobenius norm, which can be used to generalize the Euclidean based SNNs. One first correlation estimation between matrices of different sizes $\mathbf{X_1}$ $\mathbf{X_2}$ sharing one dimension is the RV-coefficient [Esc06], defined as:

$$RV(\mathbf{X}, \mathbf{Y}) = \frac{tr\left(\mathbf{X_1}\mathbf{X_1}^\top\mathbf{X_2}\mathbf{X_2}^\top\right)}{\sqrt{tr\left[(\mathbf{X_1}\mathbf{X_1}^\top)^2\right] tr\left[(\mathbf{X_2}\mathbf{X_2}^\top)^2\right]}}, \tag{5.3}$$

which would obviously come with an obvious additional computational cost.

To conclude, while different SNN architecture may be adapted with fixed inputs, different similarity measures have to be explored in order to propose a true temporal SNN, which would be beneficial to a more impartial inertial data analysis. After presenting multiple leads for cost function and SNN architecture evolutions, we focus on Gesture Recognition to propose research subjects based on Similarity Metric Learning.

5.3.2 Suggestions for Gesture Recognition

Two approaches are suggested to improve Non-Linear Similarity Metric Learning based Gesture Recognition. The first one consists in training a specific classifier based on the SNN projection. The second leverages the identification of multiple gesture styles in order to consider different training strategies.

On the one hand, it would be beneficial to envision using different classifiers. Indeed, while the K-NN classifier shows a lot of advantages, it may not capitalise on the whole potential of the trained projection. For example, it is possible to train a second network using the SNN outputs to produce a global neural classifier. Any classifier can effectively be considered, which would however require larger datasets, as the final classifier training should be performed on samples other than the ones used for the SNN training to limit any overfitting issue. In case of identified confusions for the trained classifier, classification may also be reinforced thanks to the intrinsic, directly available K-NN classifier. The first classifier which comes to mind would be the MLP, which could unify the training of a single network for feature extraction and classification. Two strategies may arise, with separate trainings for the SNN and MLP, or a unified scheme, where the Siamese objective is introduced as a regularisation over the error of one layer in the network.

On the other hand, specific training strategies may be devised to improve the SNN training. Indeed, a global training on every class available favours the extraction of the most significant traits necessary to distinguish the most different gestures. However, this is detrimental to different yet similar classes, which require a much higher effort to be separated. Thus, we suggest a multi-stage training, which would reveal the more similar classes. Then, it could be possible to identify "super-classes", which would differentiate between different "families" of gestures. This approach would then consist

in a decision tree, where each node is a SNN network and with a K-NN classifier, trained specifically on the samples of this super-class so as to extract the most salient features for this particular sub-problem. This approach has the potential of proposing an adaptive approach, where network-nodes can be added or updated as new classes are added to the interface.

Appendix A

Derivation calculations

A.1 Cross-entropy with Softmax output

Let $\mathbf{X}$ the output of the MLP for the sample $\mathbf{G_x}$, net_{Xj} the input of the neuron j activated with the sample $\mathbf{G_x}$, a_i^O the output of the i^{th} neuron of the output layer O, and t_i the corresponding target, equal to 1 for the neuron associated to the class of $\mathbf{G_x}$, and 0 for the other neurons.

$$E_W^{cross}(\mathbf{X}) = -\sum_{k=1}^{K} t_k log(a_k^O).$$

$$a_i^O = \frac{\exp(net_i^O)}{\sum\limits_k \exp(net_k^O)}.$$

$$\begin{aligned} \frac{\partial a_i^O}{\partial net_{Xj}} &= \delta_{ij}\frac{\exp(net_i^O)}{\sum\limits_k \exp(net_k^O)} - \exp(net_i^O)\frac{\exp(net_j^O)}{\left(\sum\limits_k \exp(net_k^O)\right)^2} \\ &= a_i^O(\delta_{ij} - a_j^O) \end{aligned} \tag{A.1}$$

$$\begin{aligned} \frac{\partial E_W^{cross}}{\partial net_{Xj}} &= \frac{\partial}{\partial net_{Xj}}\left(-\sum_{k=1}^{K} t_k log(a_k^O)\right) \\ &= -\sum_{k=1}^{K}\frac{t_k}{a_k^O}\frac{\partial a_k^O}{\partial net_{Xj}} \\ &= -\sum_{k=1}^{K} t_k(\delta_{jk} - a_j^O) \\ &= a_j^O(\sum_{k=1}^{K} t_k\delta_{ik}) - \sum_{k=1}^{K} t_k\delta_{jk} \\ &= a_j^O - t_j \end{aligned} \tag{A.2}$$

A.2 Computations for the Siamese Neural Network variants

A.2.1 Intermediate results

Let $\mathbf{X}$ the output of the MLP for the sample $\mathbf{G_x}$, and net_{Xj} the input of the neuron j activated with the sample $\mathbf{G_x}$

For another sample G_y with an output $\mathbf{Y}$

$$\frac{\partial(\|\mathbf{Y}\|)}{\partial net_{Xj}} = 0 \tag{A.3}$$

$$\begin{aligned}\frac{\partial(\|\mathbf{X}\|^2)}{\partial net_{Xj}} &= \Sigma_k \frac{\partial \varphi(I_{Xk})^2}{\partial net_{Xj}} \qquad = 2.\varphi(net_{Xj}).\varphi'(net_{Xj}) \\ &= 2.O_{Xj}.\varphi'(net_{Xj})\end{aligned} \tag{A.4}$$

$$\begin{aligned}\frac{\partial \|\mathbf{X}\|}{\partial net_{Xj}} &= \frac{1}{2.\|\mathbf{X}\|}.\frac{\partial(\|\mathbf{X}\|^2)}{\partial net_{Xj}} \qquad = \frac{1}{\|\mathbf{X}\|}.\varphi(net_{Xj}).\varphi'(net_{Xj}) \\ &= \frac{1}{\|\mathbf{X}\|}.O_{Xj}.\varphi'(net_{Xj})\end{aligned} \tag{A.5}$$

$$\begin{aligned}\frac{\partial(\mathbf{X}.\mathbf{Y})}{\partial net_{Xj}} &= \Sigma_k \frac{\partial \varphi(I_{Xk}).\varphi(I_{Yk})}{\partial net_{Xj}} \\ &= O_{Yj}.\varphi'(net_{Xj})\end{aligned} \tag{A.6}$$

A.2.2 Derivation of the cosine function

$$cos(\mathbf{X},\mathbf{Y}) = \frac{\mathbf{X}.\mathbf{Y}}{\|\mathbf{X}\|.\|\mathbf{Y}\|} \tag{A.7}$$

$$\begin{aligned}\frac{\partial cos(\mathbf{X},\mathbf{Y})}{\partial net_{Xj}} &= \frac{1}{\|\mathbf{X}\|^2.\|\mathbf{Y}\|^2}.\left[\frac{\partial \mathbf{X}.\mathbf{Y}}{\partial net_{Xj}}.\|\mathbf{X}\|.\|\mathbf{Y}\| - (\mathbf{X}.\mathbf{Y}).\frac{\partial(\|\mathbf{X}\|.\|\mathbf{Y}\|)}{\partial net_{Xj}}\right] \\ &= \frac{1}{\|\mathbf{X}\|.\|\mathbf{Y}\|}.\frac{\partial(\mathbf{X}.\mathbf{Y})}{\partial net_{Xj}} - \frac{\mathbf{X}.\mathbf{Y}}{\|\mathbf{X}\|^2.\|\mathbf{Y}\|^2}.\left[\|\mathbf{Y}\|.\frac{\partial\|\mathbf{X}\|}{\partial net_{Xj}} + \|\mathbf{X}\|.\frac{\partial\|\mathbf{Y}\|}{\partial net_{Xj}}\right] \\ &= \frac{\varphi'(net_{Xj})}{\|\mathbf{X}\|}.\left[\frac{O_{Yj}}{\|\mathbf{Y}\|} - cos(\mathbf{X},\mathbf{Y}).\frac{O_{Xj}}{\|\mathbf{X}\|}.\right]\end{aligned} \tag{A.8}$$

A.2.3 Derivation of the Polar Sine Metric for $n < m$

Let $A = \begin{bmatrix} O_R & O_{N_1} & \dots & O_{N_l}\end{bmatrix} = \{a_{ij}\}$, and $A_{norm} = \begin{bmatrix}\frac{O_R}{\|\mathbf{O_R}\|} & \frac{O_{N_1}}{\|\mathbf{O_{N_1}}\|} & \cdots & \frac{O_{N_l}}{\|\mathbf{O_{N_l}}\|}\end{bmatrix} = \{A^n_{ij}\}$. Let's assume that the outputs O_{N_i} and O_R are not collinear, which is plausible given our end goal.

Let $S = A^T_{norm}.A_{norm}$, then, given $\forall i, \|\mathbf{v_i}\| \neq 0$ and $\forall i,j$, v_i and v_j are linearly independent, we know that $\det(A) \neq 0$. Thus, S is a symmetric definite positive matrix, which guarantees that $det(S) = det(A_{norm})^2 \neq 0$ and the existence of $\mathbf{S^{-1}}$.

Given, for a square matrix $\mathbf{M}$, and its adjugate $adj(\mathbf{M}) = det(\mathbf{M}).\mathbf{M}^{-T}$

$$\frac{\partial \det(\mathbf{M})}{\partial m_{ij}} = adj(\mathbf{M})_{ij} \tag{A.9}$$

We get for $\mathbf{S}$, with $\mathbf{S^{-1}} = \mathbf{S^{-T}}$

$$\begin{aligned}\frac{\partial \det(\mathbf{S})}{\partial a_{ij}} &= \Sigma_k\Sigma_l \frac{\partial \det(\mathbf{S})}{\partial s_{kl}}.\frac{\partial s_{kl}}{\partial a_{ij}} \\ &= \det(\mathbf{S}).\Sigma_k\Sigma_l {s^{-1}}_{kl}.\frac{\partial s_{kl}}{\partial a_{ij}}\end{aligned} \tag{A.10}$$

or,

$$\forall k,\forall l,\ s_{kl} = \cos(\mathbf{A_k},\mathbf{A_l}) \tag{A.11}$$

Thus,

$$\frac{\partial s_{kl}}{\partial a_{ij}} = \delta_{kj}.\frac{\partial s_{jl}}{\partial a_{ij}} + \delta_{lj}.\frac{\partial s_{kj}}{\partial a_{ij}} \tag{A.12}$$

and

$$\begin{aligned}\frac{\partial \det(\mathbf{S})}{\partial a_{ij}} &= \det(\mathbf{S}).\Sigma_k\Sigma_l s^{-1}_{kl}.\left[\delta_{kj}.\frac{\partial s_{jl}}{\partial a_{ij}} + \delta_{lj}.\frac{\partial s_{kj}}{\partial a_{ij}}\right] \\ &= 2\det(\mathbf{S}).\sum_k s^{-1}_{kj}\frac{\partial \cos(\mathbf{A_k},\mathbf{A_j})}{\partial a_{ij}} \\ &= 2\det(\mathbf{S}).\frac{\varphi'(I_{ij})}{\|\mathbf{A_j}\|}.\sum_k s^{-1}_{kj}\left[\frac{A_{ik}}{\|\mathbf{A_k}\|} - cos(\mathbf{A_k},\mathbf{A_j}).\frac{A_{ij}}{\|\mathbf{A_j}\|}.\right] \\ &= 2\det(\mathbf{S}).\frac{\varphi'(I_{ij})}{\|\mathbf{A_j}\|}.\left[\sum_k s^{-1}_{kj}\frac{A_{ik}}{\|\mathbf{A_k}\|} - \frac{A_{ij}}{\|\mathbf{A_j}\|}\sum_k s^{-1}_{kj}\ s_{jk}\right] \\ &= 2\det(\mathbf{S}).\frac{\varphi'(I_{ij})}{\|\mathbf{A_j}\|}.\left[\sum_k s^{-1}_{kj}\frac{A_{ik}}{\|\mathbf{A_k}\|} - \frac{A_{ij}}{\|\mathbf{A_j}\|}\right]\end{aligned} \tag{A.13}$$

Finally,

$$\begin{aligned}\frac{\partial\left(psin(\mathbf{A})\right)}{\partial a_{ij}} &= \frac{\partial\sqrt[n]{\det\left(\mathbf{S}\right)}}{\partial a_{ij}} \\ &= \frac{\sqrt[n]{\det(\mathbf{S})}}{2n\det\left(\mathbf{S}\right)}.2\det(\mathbf{S}).\frac{\varphi'(I_{ij})}{\|\mathbf{A_j}\|}.\left[\sum_k s_{kj}^{-1}\frac{A_{ik}}{\|\mathbf{A_k}\|}-\frac{A_{ij}}{\|\mathbf{A_j}\|}\right] \\ &= \frac{psin(\mathbf{A})}{n}.\frac{\varphi'(I_{ij})}{\|\mathbf{A_j}\|}.\left[\mathbf{A_{norm}S^{-1}}-\mathbf{A_{norm}}\right]_{ij}\end{aligned} \tag{A.14}$$

A.3 Gradient Projection Norm Regularisation

The goal here is to replace the regularisation on the norm, set to one for every sample output. We want to correct the gradient so as to project the updated outputs onto the norm-1 sphere, which corresponds to a regularization equal to

$$R(\mathbf{O_X}) = (1-\|\mathbf{O_X}-\lambda\nabla(\mathbf{O_X})\|)^2 \tag{A.15}$$

Thus, we have to compute the second order partial derivatives in order to evaluate the gradient corresponding to this regularization.

$$\begin{aligned}\frac{\partial\det(\mathbf{S})}{\partial a_{ij}\partial a_{mn}} &= \det(\mathbf{S})\Sigma_k\frac{\partial}{\partial a_{mn}}\left[a_{ik}.s_{kj}^{-1}\right] \\ &= \det(\mathbf{S})\left[\delta_{im}.s_{nj}^{-1}+\Sigma_k a_{ik}.\frac{\partial s_{kj}^{-1}}{\partial a_{mn}}\right]\end{aligned} \tag{A.16}$$

with

$$\begin{aligned}\frac{\partial s_{kj}^{-1}}{\partial a_{mn}} &= \Sigma_o\Sigma_p\frac{\partial s_{kj}^{-1}}{\partial s_{op}}.\frac{\partial s_{op}}{\partial a_{mn}} \\ &= -\Sigma_o\Sigma_p(\mathbf{S^{-1}}\frac{\partial\mathbf{S}}{\partial s_{op}}\mathbf{S^{-1}})_{kj}.\frac{\partial s_{op}}{\partial a_{mn}} \\ &= -\Sigma_o\Sigma_p\left[\Sigma_r s_{kr}^{-1}.(\frac{\partial\mathbf{S}}{\partial s_{op}}\mathbf{S^{-1}})_{rj}\right].\frac{\partial s_{op}}{\partial a_{mn}} \\ &= -\Sigma_o\Sigma_p\left[\Sigma_r s_{kr}^{-1}.\delta_{ro}.s_{pj}^{-1}\right].\frac{\partial s_{op}}{\partial a_{mn}} \\ &= -\Sigma_o\Sigma_p\left[s_{ko}^{-1}.s_{pj}^{-1}\right].\frac{\partial s_{op}}{\partial a_{mn}} \\ &= -\Sigma_o\Sigma_p\left[s_{ko}^{-1}.s_{pj}^{-1}\right].\left[\delta_{on}a_{mp}+\delta_{pn}a_{mo}\right] \\ &= -\Sigma_p s_{kn}^{-1}.s_{pj}^{-1}.a_{mp}-\Sigma_o s_{ko}^{-1}.s_{nj}^{-1}.a_{mo}\end{aligned} \tag{A.17}$$

Thus,

$$\begin{aligned}\frac{1}{\det(\mathbf{S})}\frac{\partial\det(\mathbf{S})}{\partial a_{ij}\partial a_{mn}} &= \left[\delta_{im}.s_{nj}^{-1}+\Sigma_k a_{ik}.\frac{\partial s_{kj}^{-1}}{\partial a_{mn}}\right] \\ &= \delta_{im}.s_{nj}^{-1}+\Sigma_k a_{ik}\Sigma_p s_{kn}^{-1}.s_{pj}^{-1}.a_{mp}-\Sigma_k a_{ik}\Sigma_o s_{ko}^{-1}.s_{nj}^{-1}.a_{mo} \\ &= s_{nj}^{-1}\left[\delta_{im}-(\mathbf{AS^{-1}A}^T)_{im}\right]-(\mathbf{AS^{-1}})_{in}.(\mathbf{AS^{-1}})_{mj}\end{aligned} \tag{A.18}$$

We end up with an impractical formula, with the apparition of the pseudo inverse of A. The part on the left, depending on s_{nj}^{-1}, is equal to zero ONLY when the matrix $\mathbf{A}$

is square, i.e. when the dimension of the output layer is equal to the number of classes. This is unsatisfactory, as the convergence is improved as this dimension is increased.

Appendix B

Preliminary rejection tests and projections

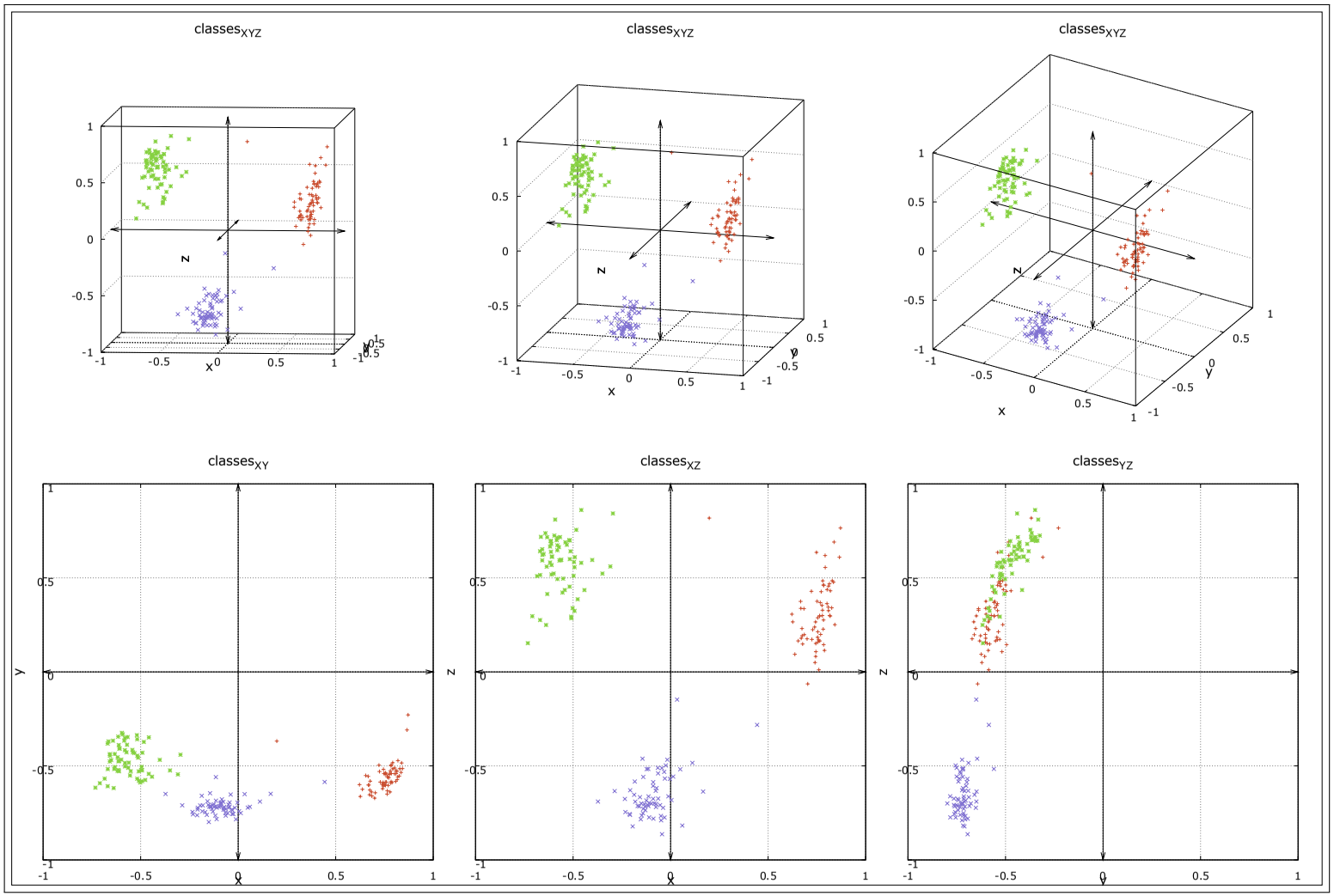


Figure B.1 – 3D plot of the training outputs distribution after training the Siamese network on 3 flick classes. Three different representations of the output space are available, with different rotations, and projections on the XY horizontal plane, and XZ and YZ vertical planes.

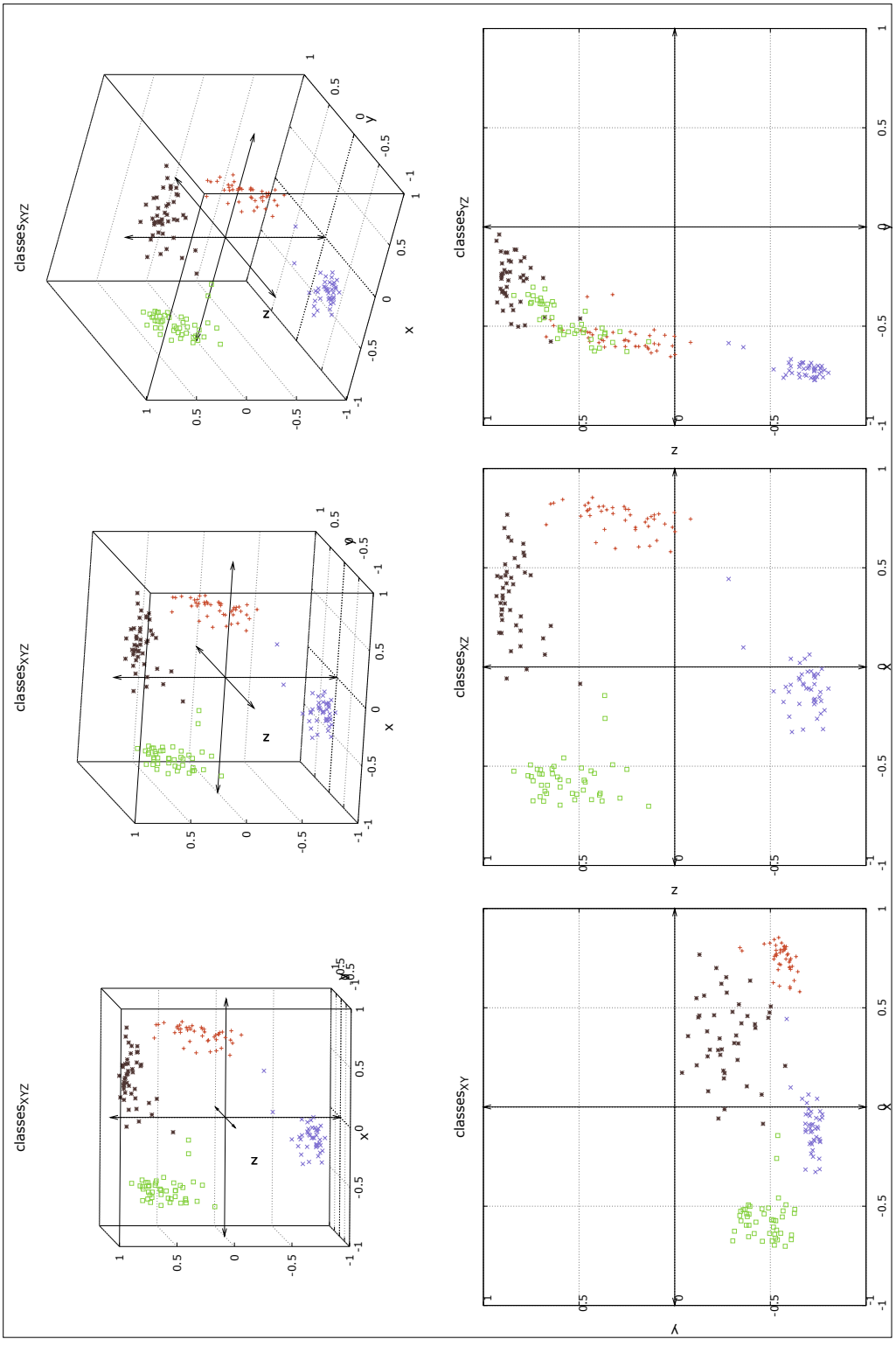


Figure B.2 – 3D plot of the test outputs distribution after training the Siamese network on 3 flick classes, with outputs from one unknown class in black. Three different representations of the output space are available, with different rotations, and projections on the XY horizontal plane, and XZ and YZ vertical planes.

Appendix C

Preprocess visualisation

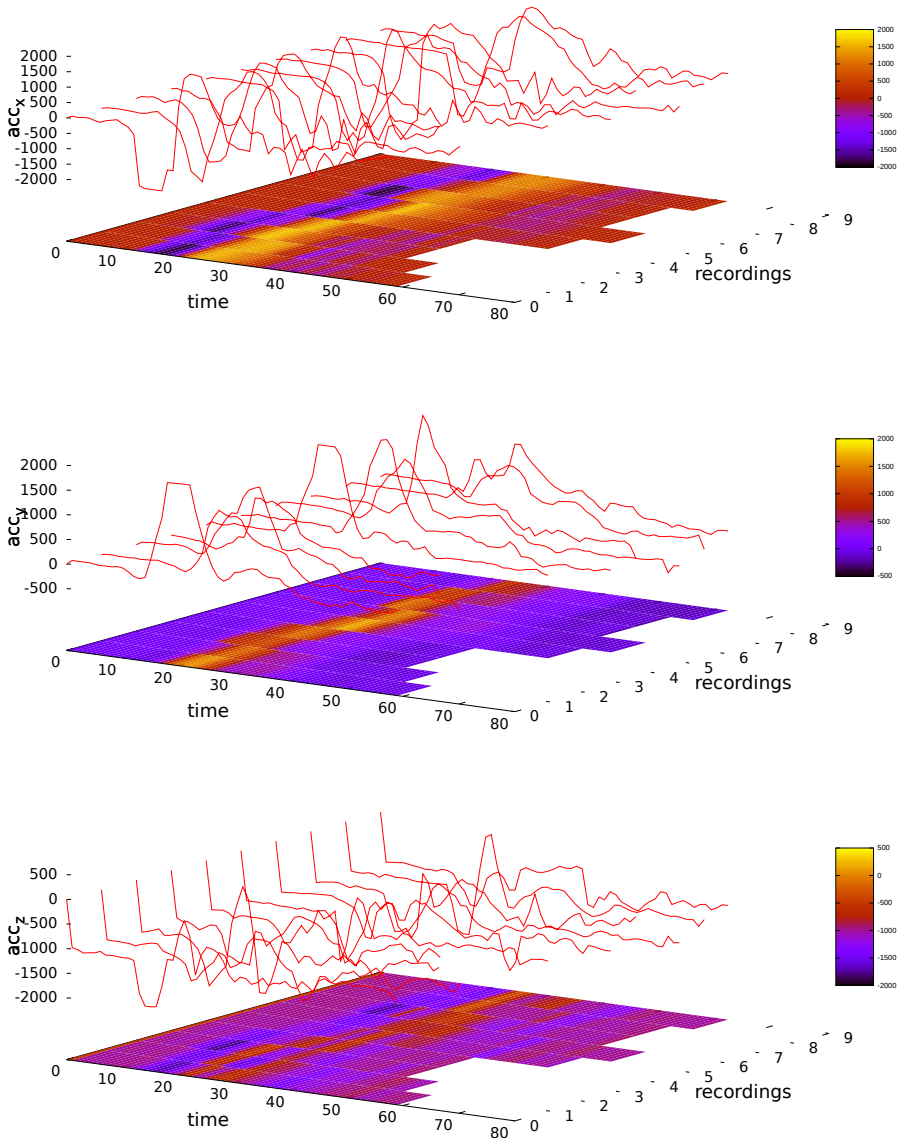


Figure C.1 – Raw accelerometer signals for 10 recordings of the "Flick East" gesture for one user.

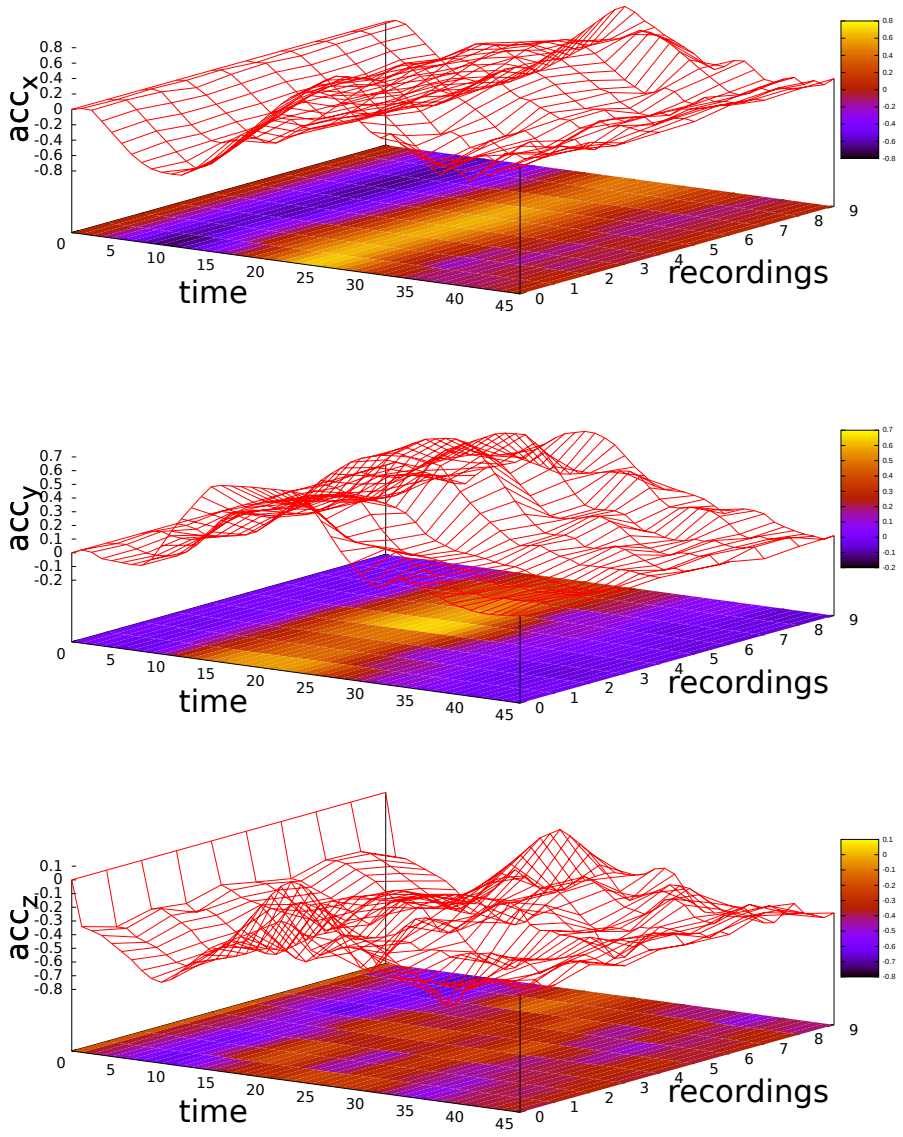


Figure C.2 – Accelerometer signals for 10 recordings of the "Flick East" gesture for one user after preprocess.

Bibliography

[ABD+90] E. Angerson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammarling, J. Demmel, C. Bischof, and D. Sorensen. LAPACK: A portable linear algebra library for high-performance computers. In *Proceedings of Supercomputing '90*, pages 2–11, November 1990. 73

[AM07] Juan Carlos Augusto and Paul McCullagh. Ambient intelligence: Concepts and applications. *Computer Science and Information Systems*, 4(1):1–27, 2007. xxvii, 3, 4

[And08] Matej Andrejasic. Mems accelerometers. In *University of Ljubljana. Faculty for mathematics and physics, Department of physics, Seminar*, 2008. 12

[AV10] A. Akl and S. Valaee. Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, #x00026; compressive sensing. In *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 2270–2273, March 2010. ix, x, 20, 22, 31

[BD13] Katarzyna Barczewska and Aleksandra Drozd. Comparison of methods for hand gesture recognition based on Dynamic Time Warping algorithm. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 207–210. IEEE, 2013. x, 30

[BGL+94] Jane Bromley, Isabelle Guyon, Yann Lecun, Eduard Säckinger, and Roopak Shah. Signature Verification using a "Siamese" Time Delay Neural Network. In *NIPS Proc*, 1994. x, 8, 48, 49, 50, 55, 56, 67

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992. 34

[BHS13] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013. xxvii, 7

[BL] Samuel Berlemont and Grégoire Lefèbvre. Fusion de caractéristiques inertielles pour la reconnaissance de gestes. In *XXIVe Colloque GRETSI*. viii

[BMKW99] Jonathan Bernstein, Raanan Miller, William Kelley, and Paul Ward. Low-noise MEMS vibration sensor for geophysical applications. *Microelectromechanical Systems, Journal of*, 8(4):433–438, 1999. 15

[Bol80] Richard A. Bolt. "Put-that-there": Voice and gesture at the graphics interface. *ACM SIGGRAPH Computer Graphics*, 14(3):262, 262–270, 270, July 1980. xxvii, 2

[Bro03] Kevin Brooks. The context quintet: narrative elements applied to context awareness. In *Human Computer Interaction International Proceedings*, volume 2003, 2003. 3

[BWCB11] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, 2011. 48, 52

[Cad94] Claude Cadoz. Le geste canal de communication homme/machine: la communication instrumentale. *Technique et Science Informatiques*, 13(1):31–61, 1994. 2

[CCB+06] Sung-Jung Cho, Eunseok Choi, Won-Chul Bang, Jing Yang, Junil Sohn, Dong Yoon Kim, Young-Bum Lee, and Sangryong Kim. Two-stage Recognition of Raw Acceleration Signals for 3-D Gesture-Understanding Cell Phones. Suvisoft, October 2006. ix, x, xxviii, 19, 21, 22, 23, 24, 25, 36

[CH67] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967. 6

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005. x, 8, 48, 49, 51, 53, 55

[CJK15] Chen Chen, R. Jafari, and N. Kehtarnavaz. Improving Human Action Recognition Using Fusion of Depth Camera and Inertial Sensors. *IEEE Transactions on Human-Machine Systems*, 45(1):51–61, February 2015. 75, 76

[CL16] Julien Cumin and Grégoire Lefebvre. A priori Data and A posteriori Decision Fusions for Human Action Recognition. In *ResearchGate*, February 2016. 75, 76

[CM02] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002. 18

[CMC10] BongWhan Choe, Jun-Ki Min, and Sung-Bae Cho. Online Gesture Recognition for User Interface on Accelerometer Built-in Mobile Phones. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Kok Wai Wong, B. Sumudu U. Mendis, and Abdesselam Bouzerdoum, editors, *Neural Information Processing. Models and Applications*, volume 6444, pages 650–657. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. x, 30

[CS11] Ke Chen and Ahmad Salman. Extracting Speaker-Specific Information with a Regularized Siamese Deep Network. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 298–306. Curran Associates, Inc., 2011. 48, 53, 55

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 35

[DKJ+07] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007. 7

[Esc06] Yves Escoufier. Operator related to a data matrix: a survey. In *Compstat 2006-Proceedings in Computational Statistics*, pages 285–297. Springer, 2006. 94

[FD07] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007. 31

[FST98] Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine learning*, 32(1):41–62, 1998. 29

[GHRS04] Jacob Goldberger, Geoffrey E. Hinton, Sam T. Roweis, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2004. 7

[GK03] J. Green and D. Krakauer. New iMEMS Angular Rate Sensing Gyroscope. http://www.analog.com/library/analogDialogue/archives/37-03/gyro.html, 2003. xxvii, 13

[GS05] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005. 43

[GSFP06] Li Guan, Sudipa Sinha, Jean-Sébastien Franco, and Marc Pollefeys. Visual Hull Construction in the Presence of Partial Occlusion. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 413–420, United States, June 2006. xxvii, 16

[HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006. xii, xiii, xxvii, xxviii, 48, 49, 52

[HHH98] Frank G. Hofmann, Peter Heyer, and Günter Hommel. Velocity profile based recognition of dynamic gestures with discrete Hidden Markov Models. In Ipke Wachsmuth and Martin Fröhlich, editors, *Gesture and Sign Language in Human-Computer Interaction*, number 1371 in Lecture Notes in Computer Science, pages 81–95. Springer Berlin Heidelberg, 1998. ix, 20, 22, 28

[HJZH08] Zhenyu He, Lianwen Jin, Lixin Zhen, and Jiancheng Huang. Gesture recognition based on 3d accelerometer for cell phones interaction. In *IEEE Asia Pacific Conference on Circuits and Systems, 2008. APCCAS 2008*, pages 217–220, November 2008. x, 20, 35

[HL10] B. Hartmann and N. Link. Gesture recognition with inertial sensors and optimized DTW prototypes. In *2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*, pages 2102–2109, October 2010. ix, x, 21, 31

[HLT14] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Discriminative Deep Metric Learning for Face Verification in the Wild. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1875–1882. IEEE, 2014. 48, 55

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997. 43

[HVL10] M. Hoffman, P. Varcholik, and J.J. LaViola. Breaking the status quo: Improving 3d gesture recognition with spatially convenient input devices. In *2010 IEEE Virtual Reality Conference (VR)*, pages 59–66, March 2010. x, 37

[ins] instrumentationtoday.com. MEMS Accelerometer-Acceleration Transducer,Sensor,Working,Technology. http://www.instrumentationtoday.com/mems-accelerometer/2011/08/. xxvii, 12

[Jen96] Finn V. Jensen. Bayesian networks basics. *AISB quarterly*, pages 9–22, 1996. 23

[Kaa] Ville Kaajakari. Noise in micromechanical systems. http://www.kaajakari.net/~ville/research/tutorials/mech_noise_tutorial.pdf. 15

[KKM03] S. Kallio, J. Kela, and J. Mäntyjärvi. Online gesture recognition system for mobile interaction. volume 3, pages 2070 – 2076 vol.3, October 2003. 28

[KKM+05] Juha Kela, Panu Korpipää, Jani Mäntyjärvi, Sanna Kallio, Giuseppe Savino, Luca Jozzo, and Sergio Di Marca. Accelerometer-based gesture control for a design environment. *Personal and Ubiquitous Computing*, 10(5):285–299, August 2005. ix, 22

[Kli09] Marco Klingmann. *Accelerometer-Based Gesture Recognition with the iPhone*. 2009. ix, 22

[KTS+12] Dor Kedem, Stephen Tyree, Fei Sha, Gert R. Lanckriet, and Kilian Q. Weinberger. Non-linear metric learning. In *Advances in Neural Information Processing Systems*, pages 2573–2581, 2012. 7

[LC13] Myeong-Chun Lee and Sung-Bae Cho. A Recurrent Neural Network with Non-gesture Rejection Model for Recognizing Gestures with Smartphone Sensors. In *Pattern Recognition and Machine Intelligence*, pages 40–46. Springer, 2013. ix, 20

[LG13] Grégoire Lefebvre and Christophe Garcia. Learning a bag of features based nonlinear metric for facial similarity. In *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, pages 238–243. IEEE, 2013. xi, xii, 48, 49, 50, 52, 53, 56, 67

[LW09] Gilad Lerman and J. Tyler Whitehouse. On d-dimensional d-semimetrics and simplex-type inequalities for high-dimensional sine functions. *J. Approx. Theory*, 156(1):52–81, January 2009. xiv, 60

[LWJ+04] Paul Lukowicz, Jamie A. Ward, Holger Junker, Mathias Stäger, Gerhard Tröster, Amin Atrash, and Thad Starner. Recognizing Workshop Activity Using Body Worn Microphones and Accelerometers. In Alois Ferscha and Friedemann Mattern, editors, *Pervasive Computing*, number 3001 in Lecture Notes in Computer Science, pages 18–32. Springer Berlin Heidelberg, April 2004. x, 29, 34

[Mah36] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, April 1936. 6

[MBBS14] Jonathan Masci, Michael M. Bronstein, Alexander M. Bronstein, and Jurgen Schmidhuber. Multimodal Similarity-Preserving Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):824–830, April 2014. 48, 52

[McN92] David McNeill. *Hand and Mind: What Gestures Reveal about Thought*. University of Chicago Press, August 1992. vii, 2

[MHS01] J. Mantyjarvi, J. Himberg, and T. Seppanen. Recognizing human motion with multiple acceleration sensors. In *2001 IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 747–752 vol.2, 2001. ix, x, 20, 21, 33, 41

[MKKK04] J. Mäntyjärvi, J. Kela, P. Korpipää, and S. Kallio. Enabling fast and effortless customisation in accelerometer based gesture interaction. In *MUM '04 Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia Pages 25-31*, pages 25–31, 2004. ix, 21

[Mli09] J. Mlich. Wiimote Gesture Recognition. pages 344–349, 2009. ix, 20, 21

[MMST00] V.-M. Mantyla, J. Mäntyjärvi, T. Seppanen, and E. Tuulari. Hand gesture recognition of a mobile device user. volume 1, pages 281 –284 vol.1, 2000. ix, 20, 21

[MP11] T. Marasovic and V. Papic. Accelerometer-based gesture classification using principal component analysis. In *2011 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–5, September 2011. x, 32, 34

[Mur98] Kevin P. Murphy. *Inference and learning in hybrid bayesian networks*. University of California, Berkeley, Computer Science Division, 1998. 24

[NH09] G. Niezen and G.P. Hancke. Evaluating and optimising accelerometer-based gesture recognition techniques for mobile devices. In *AFRICON, 2009. AFRICON '09.*, pages 1–6, September 2009. x, 41

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814, 2010. xii, 49, 51, 55

[OCK+13] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal, and Ruzena Bajcsy. Berkeley mhad: A comprehensive multimodal human action database. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pages 53–60. IEEE, 2013. xvii, xxviii, 65, 66

[Pet10] Eric Petit. GRASP: Gesture Recognition Engine, IDDN.FR.001.030023.000.S.P.2010.000.31500. Interdeposit Certification, 2010. 78

[Pri] PrimeSense. Depth mapping using projected patterns. Classification aux États-Unis 382/103; Classification internationale G06K9/20; Classification coopérative G06K2209/40, G01B11/2513, G06K9/2036, G06T7/0057; Classification européenne G06T7/00R3, G06K9/20E, G01B11/25D. 16

[Pyl05] T. Pylvänäinen. Accelerometer based gesture recognition using continuous HMMs. Pattern Recognition and Image Analysis(VIII):639–646, 2005. ix, x, 19, 29, 78

[QGCL08] Ali Mustafa Qamar, Eric Gaussier, Jean-Pierre Chevallet, and Joo Hwee Lim. Similarity learning for nearest neighbor classification. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 983–988. IEEE, 2008. 7

[RBA08] Matthias Rehm, Nikolaus Bee, and Elisabeth AndrÃ©. Wave Like an Egyptian: Accelerometer Based Gesture Recognition for Culture Specific Interactions. In *Proceedings of the 22Nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 1*, BCS-HCI '08, pages 13–22, Swinton, UK, UK, 2008. British Computer Society. 23

[RK04] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data*. Citeseer, 2004. xxviii, 30

[Sch] J\"urgen Schmidhuber. Long Short-Term Memory: Tutorial on LSTM Recurrent Networks. xxviii, 44

[SFC+11] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *In In CVPR, 2011. 3*, 2011. xxvii, 16, 17

[SPHB08] T. Schlömer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a Wii controller. pages 11–14, 2008. ix, 21

[SSM98] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10:1299–1319, 1998. 7

[STM] STMicroelectronics. Everything about STMicroelectronics 3-axis digital MEMS gyroscopes. http://www.st.com/web/en/resource/technical/document/technical_article/DM00034730.pdf. xxvii, 14

[SWT14] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. *CoRR*, abs/1406.4773, 2014. 48, 49, 52, 55

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991. vii, 3

[WPZ+09] J. Wu, G. Pan, D. Zhang, G. Qi, and S. Li. Gesture recognition with a 3-d accelerometer. 5585(Ubiquitous Intelligence and Computing):25–38, 2009. x, 35, 78

[WS09] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009. 7

[XJRN02] Eric P. Xing, Michael I. Jordan, Stuart Russell, and Andrew Y. Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002. 7

[YLLL14] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Deep metric learning for person re-identification. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 34–39. IEEE, 2014. xii, 48, 49, 50

[YTPM11] Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 247–256. Association for Computational Linguistics, 2011. xii, 48, 49, 51, 55

[YWC08] Jhun-Ying Yang, Jeen-Shing Wang, and Yen-Ping Chen. Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern Recognition Letters*, 29(16):2213–2220, December 2008. x, 33, 41

[ZCL+11] Xu Zhang, Xiang Chen, Yun Li, V. Lantz, Kongqiao Wang, and Jihai Yang. A Framework for Hand Gesture Recognition Based on Accelerometer and EMG Sensors. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 41(6):1064–1076, November 2011. ix, x, 21, 29

[ZDI+15] Lilei Zheng, Stefan Duffner, Khalid Idrissi, Christophe Garcia, and Atilla Baskurt. Siamese multi-layer perceptrons for dimensionality reduction and face identification. *Multimedia Tools and Applications*, pages 1–19, August 2015. 55

[ZDlT12] Feng Zhou and Fernando De la Torre. Generalized time warping for multi-modal alignment of human motion. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1282–1289. IEEE, 2012. x, 31, 32

[ZIG+15] Lilei Zheng, Khalid Idrissi, Christophe Garcia, Stefan Duffner, and Atilla Baskurt. Triangular similarity metric learning for face verification. In *11th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2015)*, 2015. xii, 49, 50, 55

[ZS09] C. Zhu and Weihua Sheng. Online hand gesture recognition using neural network based segmentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009*, pages 2415–2420, October 2009. x, 28, 31

[ZSZA08] M.F. Zaman, A. Sharma, Zhili Hao, and F. Ayazi. A Mode-Matched Silicon-Yaw Tuning-Fork Gyroscope With Subdegree-Per-Hour Allan Deviation Bias Instability. *Journal of Microelectromechanical Systems*, 17(6):1526–1536, December 2008. xxvii, 14

[ZT09] Feng Zhou and Fernando Torre. Canonical time warping for alignment of human behavior. In *Advances in neural information processing systems*, pages 2286–2294, 2009. x

Author's publications and Software Copyright

[Ber15] Samuel Berlemont. Siamese Neural Networks version 1, IDDN.FR.001.420012.000.S.P.2015.000.10400. Interdeposit Certification, 2015.

[BL] Samuel Berlemont and Grégoire Lefebvre. Fusion de caractéristiques inertielles pour la reconnaissance de gestes. In *XXIVe Colloque GRETSI*. 11

[BLDG15] Samuel Berlemont, Grégoire Lefebvre, Stefan Duffner, and Christophe Garcia. Siamese neural network based similarity metric for inertial gesture classification and rejection. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 1, pages 1–6, May 2015.

[DBLG14] Stefan Duffner, Samuel Berlemont, Grégoire Lefebvre, and Christophe Garcia. 3d gesture classification with convolutional neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 5432–5436. IEEE, 2014. x, xx, xxviii, 42, 43, 44, 78

[LBMG13] Grégoire Lefebvre, Samuel Berlemont, Franck Mamalet, and Christophe Garcia. BLSTM-RNN Based 3d Gesture Classification. In *Artificial Neural Networks and Machine Learning - ICANN 2013*, number 8131 in Lecture Notes in Computer Science, pages 381–388. Springer Berlin Heidelberg, September 2013. DOI: 10.1007/978-3-642-40728-4_48. x

[LBMG15] Grégoire Lefebvre, Samuel Berlemont, Franck Mamalet, and Christophe Garcia. Inertial Gesture Recognition with BLSTM-RNN. In Petia Koprinkova-Hristova, Valeri Mladenov, and Nikola K. Kasabov, editors, *Artificial Neural Networks*, volume 4, pages 393–410. Springer International Publishing, Cham, 2015. x, xx, 44, 78, 83

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

FOLIO ADMINISTRATIF

THÈSE SOUTENUE DEVANT L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

NOM : BERLEMONT
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 11/02/2016

Prénoms : Samuel Charles

TITRE : Apprentissage Automatique de Métrique Non Linéaire / Application à la Reconnaissance de Gestes
Automatic Non Linear Metric Learning / Application to Gesture Recognition

NATURE : Doctorat

Numéro d'ordre :

Ecole doctorale : Infomaths

Spécialité : Mathématiques Appliquées

RESUME : Cette thèse explore la reconnaissance de gestes à partir de capteurs inertiels pour Smartphone. Ces gestes consistent en la réalisation d'un tracé dans l'espace présentant une valeur sémantique, avec l'appareil en main. Notre étude porte en particulier sur l'apprentissage de métrique entre signatures gestuelles grâce à l'architecture "Siamoise" (réseau de neurones siamois, SNN), qui a pour but de modéliser les relations sémantiques entre classes afin d'extraire des caractéristiques discriminantes. Cette architecture est appliquée au perceptron multicouche (MultiLayer Perceptron). Les stratégies classiques de formation d'ensembles d'apprentissage sont essentiellement basées sur des paires similaires et dissimilaires, ou des triplets formés d'une référence et de deux échantillons respectivement similaires et dissimilaires à cette référence. Ainsi, nous proposons une généralisation de ces approches dans un cadre de classification, où chaque ensemble d'apprentissage est composé d'une référence, un exemple positif, et un exemple négatif pour chaque classe dissimilaire. Par ailleurs, nous appliquons une régularisation sur les sorties du réseau au cours de l'apprentissage afin de limiter les variations de la norme moyenne des vecteurs caractéristiques obtenus. Enfin, nous proposons une redéfinition du problème angulaire par une adaptation de la notion de « sinus polaire », aboutissant à une analyse en composantes indépendantes non-linéaire supervisée. A l'aide de deux bases de données inertielles, la base MHAD (Multimodal Human Activity Dataset) ainsi que la base Orange, composée de gestes symboliques inertiels réalisés avec un Smartphone, les performances de chaque contribution sont caractérisées. Ainsi, des protocoles modélisant un monde ouvert, qui comprend des gestes inconnus par le système, mettent en évidence les meilleures capacités de détection et rejet de nouveauté du SNN. En résumé, le SNN proposé permet de réaliser un apprentissage supervisé de métrique de similarité non-linéaire, qui extrait des vecteurs caractéristiques discriminants, améliorant conjointement la classification et le rejet de gestes inertiels.

MOTS-CLÉS : Reconnaissance de Gestes, Systèmes MicroElectroMécaniques, Capteurs Inertiels, Apprentissage Automatique, Apprentissage de Métrique, Métrique de Similarité, Réseau de Neurones Artificiels, Réseau Siamois

Laboratoire (s) de recherche : LIRIS

Directeur de thèse: Christophe GARCIA

Président de jury :

Composition du jury :

Rapporteur:	Thierry CHATEAU	Professeur (Université Blaise Pascal)
Co-encadrant:	Stefan DUFFNER	Maître de Conférences (INSA Lyon)
Directeur:	Christophe GARCIA	Professeur (INSA Lyon)
Co-encadrant:	Grégoire LEFEBVRE	Docteur, Ingénieur de Recherche (Orange Labs)
Rapporteur:	Michel PAINDAVOINE	Professeur (Université de Bourgogne)
Examinateur:	Denis PELLERIN	Professeur (Université Joseph Fourier)
Examinateur:	Nicolas THOME	Maître de Conférences - HDR (Université Pierre et Marie Curie)