Deep learning for human motion analysis

Natalia Neverova

UNIVERSITÉ DE LYON

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

nº d'ordre NNT : 2016LYSEI029

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

Ecole Doctorate ED 512
Informatique et Mathématique de Lyon
(INFOMATHS)

Spécialité de doctorat : Informatique

Soutenue publiquement le 08/04/2016, par :
Natalia Neverova

Deep Learning for Human Motion Analysis

Devant le jury composé de :

JURIE, Frédéric	Professeur, Université de Caen	Rapporteur
ODOBEZ, Jean-Marc	Maître d'Enseignement et de Recherche, IDIAP	Rapporteur
BASKURT, Atilla	Professeur, INSA-Lyon	Examinateur
BISCHOF, Horst	Professeur, TU Graz	Examinateur
SCHMID, Cordelia	Directeur de Recherche, INRIA Rhône-Alpes	Examinateur
THOME, Nicolas	Maître de Conférences, HDR, UPMC	Examinateur
WOLF, Christian	Maître de Conférences, HDR, INSA-Lyon	Directeur de thèse
TAYLOR, Graham	Assistant Professor, University of Guelph	Co-directeur de thèse

ABSTRACT

The research goal of this work is to develop learning methods advancing automatic analysis and interpreting of human motion from different perspectives and based on various sources of information, such as images, video, depth, mocap data, audio and inertial sensors. For this purpose, we propose a number of deep neural models for supervised classification and semi-supervised feature learning, as well as modelling of temporal dependencies, and show their efficiency on a set of fundamental tasks, including detection, classification, parameter estimation and user verification.

First, we present a method for human action and gesture spotting and classification based on multi-scale and multi-modal deep learning from visual signals (such as video, depth and mocap data). Key to our technique is a training strategy which exploits, first, careful initialization of individual modalities and, second, gradual fusion involving random dropping of separate channels (dubbed ModDrop) for learning cross-modality correlations while preserving uniqueness of each modality-specific representation. Start by exploring different fusion strategies, including per-frame voting and modeling temporal dependencies with a recurrent model (RNN), we show that fusing multiple modalities at several spatial and temporal scales leads to a significant increase in recognition rates, allowing the model to compensate for errors of the individual classifiers as well as noise in the separate channels. Futhermore, the proposed ModDrop training technique ensures robustness of the classifier to missing signals in one or several channels to produce meaningful predictions from any number of available modalities. In addition, we demonstrate the applicability of the proposed fusion scheme to modalities of arbitrary nature by introducing an additional audio channel.

Moving forward, from 1 to N mapping to continuous evaluation of gesture parameters, we address the problem of hand pose estimation and present a new method for regression on depth images, based on semi-supervised learning using convolutional deep neural networks. An intermediate representation is first constructed based on a segmentation into parts and is learned from two training sets: one containing labelled synthetic training images produced from 3D models by a rendering pipeline, and a second set containing unlabelled real images acquired with a consumer depth sensor. Our method does not rely on labels for the real data, and no explicit transfer function is defined or learned between synthetic and real data. Instead, a loss is defined on these data by extracting geometrical, structural and topological information related to a strong prior on the intermediate representation, i.e. on the segmentation of a hand into parts. We show that regression of joint positions is easier and more robust from a rich semantic representation like a segmentation into parts than from raw depth data, provided that this semantic segmentation is of high fidelity.

In separate but related work, we explore convolutional temporal models for human authentication based on their motion patterns. In this project, the data is captured by inertial sensors (such as accelerometers and gyroscopes) built in mobile

devices. Having explored existing temporal models (RNN, LSTM, clockwork RNN), we show how the convolutional Clockwork RNN can be extended in a way that makes the learned features shift-invariant, and propose a more efficient training strategy for this architecture. Finally, we incorporate the learned deep features in a probabilistic biometric framework for real time user authentication.

RÉSUMÉ

L'objectif de ce travail est de développer des méthodes avancées d'apprentissage pour l'analyse et l'interprétation automatique du mouvement humain à partir de sources d'information diverses, telles que les images, les vidéos, les cartes de profondeur, les données de type «MoCap» (capture de mouvement), les signaux audio et les données issues de capteurs inertiels. A cet effet, nous proposons plusieurs modèles neuronaux et des algorithmes d'entrainement associés pour l'apprentissage supervisé et semi-supervisé de caractéristiques. Nous proposons des approches de modélisation des dépendances temporelles, et nous montrons leur efficacité sur un ensemble de tâches fondamentales, comprenant la détection, la classification, l'estimation de paramètres et la vérification des utilisateurs (la biométrie).

Premièrement, nous présentons une méthode pour la détection, la classification et la localisation de gestes humains basée sur le Deep Learning multi-échelle et multi-modal à partir de plusieurs signaux. Un aspect clé de notre technique est une nouvelle stratégie d'entrainement, exploitant, d'une part, une initialisation méticuleuse des modalités individuelles et, d'autre part, une fusion progressive impliquant l'annulation aléatoire («ModDrop») de modalités. Cela permet l'apprentissage efficace des corrélations inter-modalités tout en préservant le caractère unique de la représentation spécifique à chaque modalité.

En explorant différentes stratégies de fusion, nous montrons que la fusion des modalités à plusieurs échelles spatiales et temporelles conduit à une augmentation significative des taux de reconnaissance, ce qui permet au modèle de compenser les erreurs des classifieurs individuels et le bruit dans les différents canaux. En outre, la technique proposée assure la robustesse du classifieur face à la perte éventuelle d'un ou de plusieurs canaux. Nous démontrons l'extension de la méthode de fusion proposée aux modalités de nature arbitraire en introduisant un canal audio supplémentaire.

Dans un deuxième temps nous abordons le problème de l'estimation de la posture de la main en présentant une nouvelle méthode de régression à partir d'images de profondeur. Basée sur l'apprentissage semi-supervisé à l'aide de réseaux de neurones profonds, la méthode procède par une fusion des données de profondeur brutes et d'une représentation intermédiaire sous forme d'une carte de segmentation de la main en parties. Nous argumentons que cette représentation intermédiaire contient des informations topologiques pertinentes, fournissant des indices utiles pour l'estimation des positions des articulations de la main.

Le mapping fonctionnel entre cartes de profondeur et cartes de segmentation des cartes est appris de manière semi-supervisée et de manière faiblement supervisée à partir de deux ensembles de données : un jeu de données synthéthiques créé par un pipeline de rendu, comprenant une annotation dense des pixels; et un ensemble de données réelles comprenant une annotation des positions des articulations, mais sans cartes de segmentation.

Dernièrement, dans le cadre d'un projet séparé (mais lié thématiquement), nous explorons des modèles temporels pour l'authentification automatique des utilisateurs de smartphones à partir de leurs habitudes de tenir, de bouger et de déplacer leurs téléphones. Dans ce contexte, les données sont acquises par des capteurs inertiels embaraqués dans les appareils mobiles. Après avoir exploré plusieurs architectures neuronales (RNN, LSTM, Clockwork RNN) pour l'apprentissage efficace des extracteurs de caractréristiques, nous proposons une variante efficace et invariante des Clockwork RNN, nommée Dense Clockwork RNNs (DCWRNN).

Nos résultats démontrent que le mouvement humain véhicule des informations pertinentes sur l'identité des utilisateurs; ces informations peuvent servir comme une composante précieuse pour les systèmes automatiques d'authentification multimodale.

ACKNOWLEDGEMENTS

This manuscript is the fruit of more than three intense years, full of exciting deadlines, sleepless nights and transatlantic flights. More than 150 times I have found myself in airports, and throughout these years, at different moments of time, there were four different countries which I called home.

It was an interesting and enjoyable time, which taught me a lot, and there are a number of people who should take the credit for that.

Most importantly, I would like to thank Christian Wolf, my main academic advisor and my best friend on this adventure. None of this would ever be possible without his optimism and genuine enthusiasm, his extensive help, encouragements and precious moral support. If I had the Archimedes' place-to-stand these years, it was on the second floor of Jules Verne INSA building.

My deep gratitude goes to my second advisor, Graham Taylor, who introduced me to the broader deep learning community and whose wisdom and gentle guidance often helped me to find the way. Also, I would like to thank him for all long hours he spent on proof reading our conference and journal submissions, as well as this manuscript.

Furthermore, I would like to express my sincere appreciation to Frédéric Jurie and Jean-Marc Odobez who agreed to review this manuscript, as well as to the rest of the PhD committee members, including Atilla Baskurt, Horst Bischof, Cordelia Schmid and Nicolas Thome.

I would like to thank the whole Awabot engineering team, especially Florian Nebout and Guillaume Bono, as well as Véronique Aubergé and her research group from Laboratoire d'Informatique de Grenoble (LIG), and also Giulio Paci and Giacomo Sommavilla, our collaborators from ICST, CNR (Padova, Italy).

My sincere gratitude goes to Jean-Marie Becker and other colleagues from CPE Lyon, under whose guidance I had an opportunity to gain a teaching experience in the context of a French *grande école*.

The research sprint at Google ATAP was, with no doubt, one of the brightest highlights of these years, and I would like to express my great appreciation to Deepak Chandra and Brandon Barbello, who introduced terms *blocker* and *action unit* in my academic life. In this context, I would like to thank Griffin Lacey for his gentle humour and taking care about all practical issues, as well as to Lex Fridman for sharing experiences and night scooter rides. Long working days at Google Tech Corners would not be the same without Elie Khoury, Laurent El Shafey, Sujeeth Bharadvaj, Jagadish Agrawal, Ethan Rudd, Lalit Jain, and all others.

Furthermore, I would like to thank:

— Joel Best, Eric Lombardi, Dominique Barriere, Rebecca Reimer and Ecole Doctorale InfoMath for their technical and administrative support during these years;

— GDR-Isis for providing a mobility grant for my stay in Guelph in 2014; also, I acknowledge financial support from IPAM, UCLA and University of Montréal

who provided me with full scholarships for attending deep learning and computer vision summer schools;

— LIRIS colleagues Julien Mille, Stefan Duffner and Guillaume Lavoué for their help and discussions;
— friends and colleagues from University of Guelph, namely Matt Veres, Dan (Jiwoong) Im, Weiguang Gavin Ding, Dhanesh Ramachandram, Fan Li, Jan Rudy, Carolyn Augusta, Michal Lisicki, Nikhil Sapru and others, especially for making our NIPS experiences fun and inspiring;
— Mireille and Jean-Marie Nicolas, Michael and Denise Sharp, Marty Guericke and Elaine McMillan, my wonderful hosts, who made every effort to make me feel like home wherever I stayed;
— Rizwan Khan, Peng Wang, Thomas Konidaris, Rahul Mourya, Vamsidhar Reddy Gaddam and Rahat Khan for our discussions, trips and for being a rare but healthy distraction from long working days in Lyon;
— Johann and André for their infinite kindness, support and eager to help;
— Vladimir, Elena, Victor, Alexey and Sergey, my friends-of-all-times – for everything they are doing for me.

Finally, I would like to express my infinite gratitude to my parents, for their constant care and encouragement, and to my brother, who always has my back.

CONTENTS

LIST OF TABLES

ACRONYMS

CNN Convolutional Neural Network
ConvNet Convolutional Network
CRF Conditional Random Field
CWRNN Convolutional Recurrent Neural Network
DBN Dynamic Bayesian Network
DCWRNN Dense Convolutional Recurrent Neural Network
DoF Degree of Freedom
EER Equal Error Rate
ERT Extremely Randomized Trees
FAR False Acceptance Rate
FRR False Rejection Rate
FSM Finite State Machine
GA Genetic Algorithms
GMM Gaussian Mixture Model
GRU Gated Recurrent Unit
HL Hidden Layer
HMM Hidden Markov Model
HTER Half Total Error Rate
HoG Histogram of Gradients
HoF Histogram of Optical Flow
ICP Iterative Closest Point
kNN k Nearest Neighbor
LOP Local Occupancy Pattern
LRM Latent Regression Model
LRT Latent Regression Tree
LSTM Long Short Term Memory
MAP Maximum a posteriori
MFCC Mel-frequency Cepstral Coefficients
MLP Multi-layer Perceptron
Mocap Motion Capture
NLL Negative Log Likelihood
NLP Natural Language Processing
PCA Principal Component Analysis
PSO Particle Swarm Optimization
RBM Restricted Boltzmann Machine
RDF Random Decision Forest
RNN Recurrent Neural Network
SGD Stochastic Gradient Descent
SIFT Scale-Invariant Feature Transform
SIP Segmentation Index Point
STIP Spatio-Temporal Interest Point
SURF Speeded Up Robust Features
UBM Universal Background Model

Part I

MOTIVATION AND BACKGROUND

«А чем вы занимаетесь?» — спросил я.
«Как и вся наука, — сказал горбоносый. —
Счастьем человеческим».

— Аркадий и Борис Стругацкие,
«Понедельник начинается в субботу» [1]

1

INTRODUCTION

Automatic analysis and interpretation of human motion from visual input, one of the essential computer vision challenges, has been a decades-long effort resulting in a vigorous growth of this body of research from the 1970s until the present. As one could guess, the problem occupies the minds not only as a captivating scientific puzzle, but also due to its practical importance. Hundreds of existing potential applications in urgent need of this technology include, to name a few, control, navigation and manipulation in real and virtual environments, human-robot interaction and smart homes, telepresence systems, clinical diagnosis and monitoring, elderly support and systems which aid the hearly impaired, learning applications and games, engineering and computer-aided design systems, automatic video annotation and indexing, forensic identification and lie detection.

Application-driven by definition, human motion analysis has been explored in parallel by different research communities and from different perspectives. Today, it has evolved into collection of isolated standard tasks, ranging from interpreting subtle facial expressions to crowd motion analysis. Numerous taxonomies have been proposed for this domain, but the majority of existing works consider group or individual activities, primitive actions or gestures. Apart from differences in scale and level of abstraction, each subproblem can be viewed as a variation of one of two classical computer vision challenges, namely *Recognition* or *Reconstruction*.

1. “What are you working on?”, I asked. “As with all science – the happiness of man”. (Arkady and Boris Strugatsky, "Monday Begins on Saturday”)

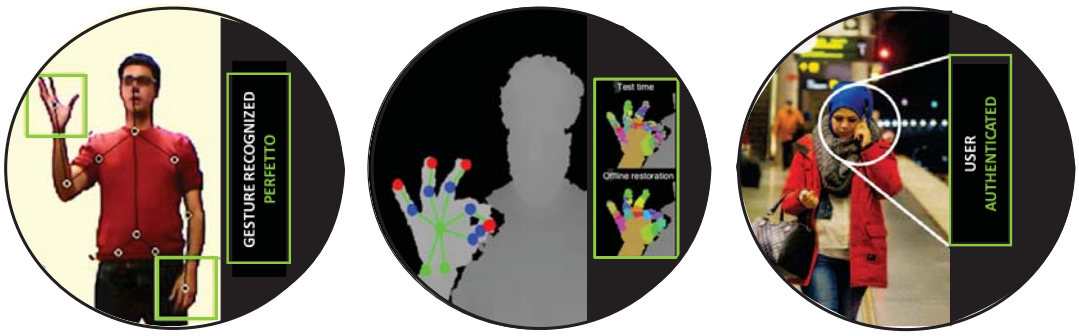


Figure 1 – Directions of human motion analysis explored in this thesis: gesture recognition, hand pose estimation and user validation from motion.

The first formulation, purely discriminative in spirit, aims on triggering events associated with one or another type of visual inputs. *Action* (such as walking or running), hand *gesture* or *emotion* recognition are typical examples of this area of research.

The reconstruction task, which is typically more challenging computationally and algorithmically, requires, in turn, higher quality of visual input and better understanding of the processes behind it. It also requires more sophisticated models, e.g. structured output. In the context of human motion analysis, reconstruction challenge is formulated as *pose estimation* and *tracking* (of a full body or a single hand).more sophisticated models (e.g., structured output) In this thesis, we mainly focus on interpreting near-range human activities, where body parts are clearly visible and articulated, i.e. *gestures*.

Numerous gesture taxonomies have been proposed, but there is little agreement among researchers on what gesture characteristics are the most discriminative and useful. In this work we focus on intentional gestures, bearing communicative function and targeting enhancement of the communicative value of speech by doubling or conveying additional semantic information. Depending on the application, their functional load may be more significant than simply augmenting the audio channel: in robot-computer interaction, for example, gestures often play the primary role.

Conversational gestures, i.e. hand movements accompanying speech, may convey semantic meaning, as well as information about a speaker's personality and cultural specifics, momentary emotional state, intentions and attitude towards the audience and subject at hand. Interestingly, a number of psychological studies (e.g. [249]) suggest that motor movements do not only aim at illustrating pronounced utterances, amplifying their emotional impact or compensating for the lack of linguistic fluency, but, in turn, influence the process of speech production allowing the speaker to produce more complex and vivid verbal descriptions. Therefore, an ability to recognize and interpret non-verbal cues coupled with verbal information would be a natural step towards understanding and plausible simulation of human behavior and, consequently, making human-machine communication truly effortless and intuitive.

In this work, we approach the problem of gesture analysis from both the recognition and reconstruction perspectives, exploring classification and estimation of parameters of movements with an ultimate goal to create a generalized motion under-

standing framework for human-robot and human-device interaction. In particular, we are interested in multi scale and multi level models, capable of jointly interpreting both subtle finger movements and full body motion (see Figure 1).

For completeness, we additionally explore an inverse problem of motion analysis for user *validation* (Figure 1). While the gesture recognition task naturally requires invariance with respect to inter-person differences between subjects performing gestures, here the target of discrimination and the invariant are swapped such that the person is recognized regardless of tasks or types of gestures he or she is performing.

In spite of the existing extensive body of research, the problem of gesture analysis remains an unsolved challenge when it comes to many real life applications. State-of-the-art touchless interfaces based on precise finger tracking have a limited working range of distances (typically 0.5-1.2m) or require that the subject wear special markers or other attire, such as colored gloves or t-shirts [310]. They are therefore often impractical, while full body skeleton-based models typically require unnatural exaggerated articulation, have a very limited vocabulary and therefore lack expressive power.

1.1 APPLICATIONS

The research which we present in this manuscript was initially motivated and funded by two joint industrial-academic projects, namely *Project Interabot* and *Project Abacus*, which we briefly describe in this section.

1.1.1 *Project Interabot*

A majority of the work described in this thesis was conducted collaboratively between of several industrial and academic partners brought together under the umbrella of project *Interabot* lead by the robotic company *Awabot* (Lyon, France), which aims to create a low-cost mobile domestic robot-companion, bearing entertaining and social functions for children and adults and serving as personal assistant for elderly people. As a part of a *smart home* environment, such an assistant could, for example, process spontaneously expressed requests for regulating temperature and lighting, watering plants, turning on and off various devices. At the beginning of this project, our colleagues from LIG, Grenoble [11] have demonstrated that elderly people naturally tend to perceive a moving controller, performing such actions by request, as a companion, rather than a service robot.

At the beginning of this chapter, the reader can see an early prototype of a domestic companion, developed under this project and named *Emox*. In its current implementation, each Emox robot is equipped with a number of sensors, among which are RGB and depth cameras, accelerometer and a microphone. Its basic functionality includes navigation while avoiding obstacles, object and face recognition and scene segmentation allowing for a "following" mode. Furthermore, each robot has a built-in picoprojector and speakers enabling such applications as interactive games, karaoke and videoconferencing.

Thus, the initial motivation of this work was to develop a visual human-robot interaction system that could be used in unconstrained indoor environments for

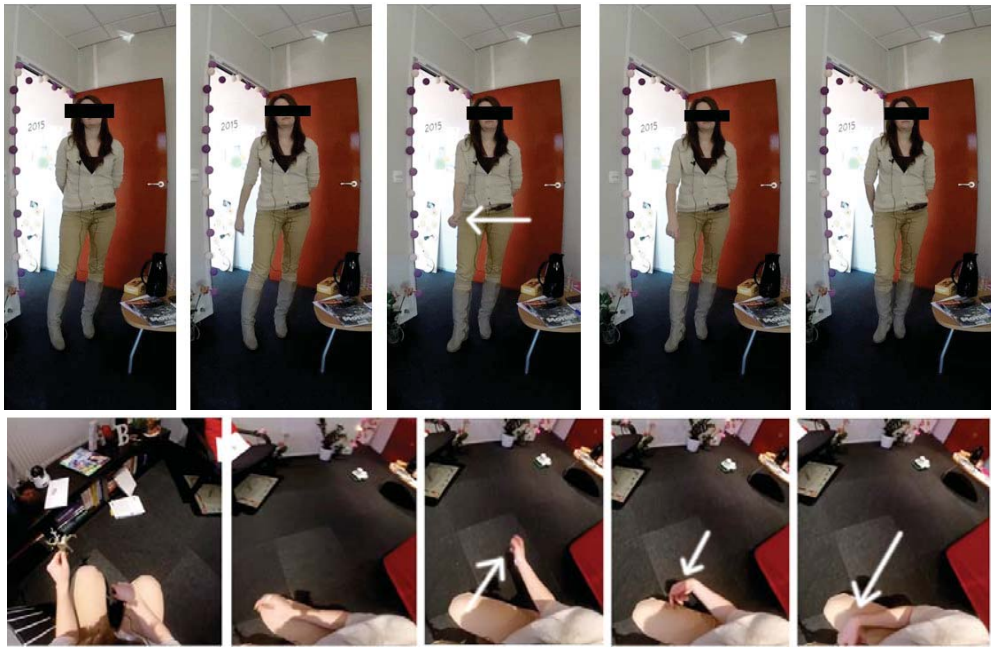


Figure 2 – An example of a gesture (*come close to me*), as viewed from the front and from above (figures by our partner from GIPSALab, [115]).

human action and gesture recognition, in real time and on a multi-sensor mobile robotic platform.

Using the *wizard of oz* technique, our partners have collected a large amount of samples of real-life gestures naturally performed by elderly people in different contexts while interacting with the robot and expressing various requests. By sorting and clustering several hours of video recordings, we have defined a hierarchically organized vocabulary of navigational gestures with all typical variations corresponding to each gesture type. As example of such a gesture is shown in Figure 2.

This vocabulary has been used for collecting of a large amount of training data for the gesture recognition system. In each case, RGB, depth and skeleton data, provided by the robot sensors, have been recorded.

All of these preparatory research and data collection steps allowed us to build a platform for creating a robot-based multimodal gesture-recognition system which will be one of the main subjects of this thesis.

1.1.2 *Project Abacus*

The research described in Chapter 6 has been conducted as a part of project *Abacus* in collaboration and under the guidance of *Google, Inc.* (Mountain View, USA).

Generally, project Abacus aims at eliminating the necessity of entering passwords on smartphones and shifting the burden of authentication from the user to the device itself. Accordingly, the research goal of this project was defined as studying the possibility of active authentication of smartphone users from multimodal data provided by built-in sensors.

Specifically for this project, Google ATAP has organized and performed a large scale multi-modal data collection. To facilitate the research, we worked with a third party vendor's panel to recruit and obtain consent from volunteers and provide them with LG Nexus 5 research phones which had a specialized read only memory (ROM) for data collection. Volunteers had complete control of their data throughout its collection, as well as the ability to review and delete it before sharing for research. Further, volunteers could opt out after the fact and request that all of their data be deleted. The third party vendor acted as a privacy buffer between Google ATAP and the volunteers.

This study included approximately 1,500 volunteers using the research phones as their primary devices on a daily basis. The data collection was completely passive and did not require any action from the volunteers in order to ensure that the data collected was representative of their regular usage.

The data corpus consisted of 27.62 TB of smartphone sensor signals, including images from a front-facing camera, touchscreen, accelerometer, gyroscope, magnetometer, GPS, bluetooth, wifi, cell antennae, etc.

Our role in this project was to explore the perspective of user authentication from built-in inertial motion sensors, such as accelerometer, gyroscope or magnetometer. This task, while being relatively new and approaching the idea of exploiting human movements from a completely differently perspective, falls naturally into the general domain of human motion analysis we are exploring in this work.

1.2 KEY METHODOGICAL ASPECTS AND CONTRIBUTIONS

As mentioned earlier, this manuscript introduces a number of contributions to different aspects of human motion analysis in its different sub domains. In this section, we would like to emphasize a number of key ideas which we believe are important in the context of this work and which will guide and motivate a number of crucial strategical decisions in the following chapters. Furthermore, here we summarize our main contributions proposed for several different applications and attempt to draw parallels between corresponding computer vision and machine learning problems to show what unites them.

— First of all, as explicitly stated in the title of this manuscript, each problem approached here is in one way or another associated with the **automatic analysis of human motion**. We begin our exploration journey from the top level by looking at the problem of gesture analysis in its general formulation and then zoom in on hands and focus specifically on their articulation. Even though these two applications are conventionally considered separately by the computer vision community, the tight connection between them is apparent: an exact hand pose, for example, can serve as an intermediate representation for recognition of gestures and estimation of their parameters, while the predefined gesture class per se can be seen as a prior in the task of reconstruction of hand joint positions in 3D space.

  Finally, to abstract away from semantic meaning of hand gesticulation, we look at the third problem, formulated as user recognition from their hand motion patterns, where the input data is non-visual and thus hardly interpretable.

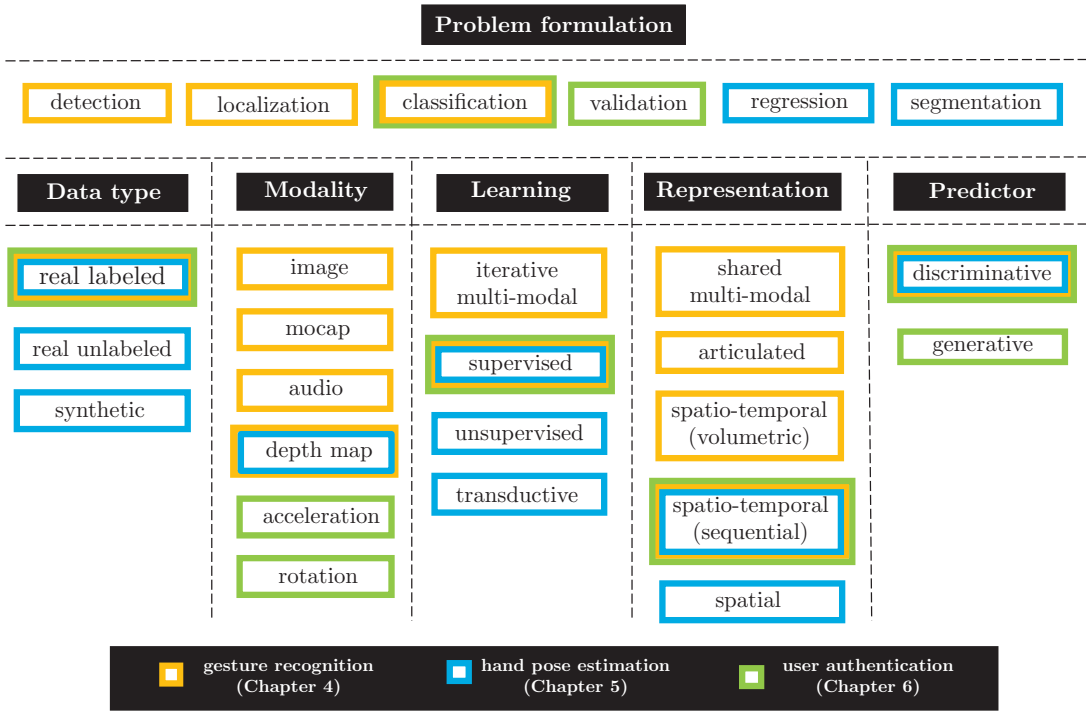


Figure 3 – Summary of our contributions represented as a set of key words, which characterize applications considered in each part of this thesis.

— Regardless of a practical objective, each application in this work is considered as a instance of a **machine learning problem**. Therefore, in the following discussion to better identify and highlight important aspects and make contributions at different stages of the learning pipeline, we will stick to the generalized problem structure represented as a table in Figure 3. At each step, the chosen color encoding indicates key words specific for each of the considered applications (gesture recognition, hand pose estimation and user validation from motion). As the reader may see from the top part of this table, the learning problem formulation itself differs depending on the application. In the gesture recognition setting, our objective is to perform simultaneous gesture *detection*, *recognition* and accurate *localization*, while the hand pose estimation is formulated as a *regression* task with intermediate *segmentation* of a hand into parts. Finally, user *validation* also involves an intermediate step of discriminative feature learning with a *classification* training objective.

— An important conceptual priority of this work is data-driven **learning of static and dynamic data representations**, as opposed to designing **hand crafted descriptors** specific to a given type of the input. Therefore, a particular class of models which will interest us in this work is a group of *connectionist* approaches commonly referred to in modern literature (somewhat abusively) as **deep learning**. In our context of human motion analysis, we explore both spatial and temporal strategies for feature learning and propose a number of fully connected, convolutional and recurrent deep learning architectures.

— Furthermore, in an attempt to gain a larger view on the problem, we aim to develop generalized **modular models** built on **composable blocks**, with composability meaning both pluggable data channels as well as spatial, temporal and semantic scales of data processing. We would like to stress though that the idea of creating high level composable *black boxes* which can further be used for engineering purposes is what represents the *practical goal* of the present research rather than our research methodological *strategy* per se: on the contrary, developing truly universal algorithms, as a rule, requires deep understanding and thorough analysis of their underlying properties and dynamics.

— In particular, throughout this thesis, we have explored **various data modalities**, both visual and non-visual, which include color video and depth maps, full body motion caption data, audio signals as well as measurements performed with inertial motion sensors, such as accelerometer and gyroscope sensors. Naturally, we also pay special attention to **fusion aspects** in multimodal deep learning architectures and increasing of *robustness* of developed multimodal models to missing or corrupted signals during test time in real world conditions. In the gesture analysis part of this manuscript, we thoroughly analyze the role of each data channel and, even more importantly, the dynamics of fusion process and its effectiveness and efficiency as a function of the training strategy. In this context, we propose a multimodal training technique (called *ModDrop*) which ensures robustness of the classifier to missing signals in one or several channels to produce meaningful predictions from any number of available modalities. As mentioned earlier, whenever possible, we endevour to stick to the modular approach, assuming that one or another data channel can also be *intentionally* removed or added without loss of generality.

— The same logic applies to the other plane of decomposition, namely **spatial and temporal scales** of learned data representations. In our proposed framework for gesture recognition, for example, the visual input is organized according to the scale of the movement: the motion capture information is used to characterize the motion of the whole upper body, while intensity and depth images cropped around positions of hands more specifically capture fine finger articulation. At the same time, the data stream is sampled at several temporal scales with following combining the obtained spatio-temporal representations in a single framework. The idea of multi-scale temporal processing was further explored by us in the context of recurrent temporal deep learning models for the task of user validation from motion, and in particular learning abstract temporal representations from acceleration and rotation measurements. In this setting, we propose an optimized dense convolutional mechanism (DCWRNN) performing such temporal decomposition while being shift-invariant in the temporal domain.

— Furthermore, in order to ease the burden of annotating huge amounts of training data, which typically requires human participation and is therefore tedious and costly, we ask ourselves the question of how **synthetic data** can be leveraged in real computer vision applications through **unsupervised** or **semi-supervised** learning and **domain adaptation**. More precisely, in this thesis this idea is explored as a part of the hand pose estimation framework, for which we have artificially generated a huge amount of synthetic hand depth maps with corre-

sponding annotations of hand parts. The real and synthetic sets of data are then combined in a single semi-supervised trunsductive deep learning framework, which performs estimation of hand joint positions in 3D space. This is done by first aligning the real and synthetic domains in the intermediate feature space, which is produced by pixelwise segmentation of a hand into parts. In this context, we propose and analyze a number of different strategies for the domain adaptation step, formulated in the spirit of both unsupervised and weakly-supervised learning.

— The final issue, which we see as a key requirement in mobile contexts and which will appear as a leitmotif in every chapter, is a **real time inference** and **lightweight deployment** of the proposed models. This aspect is particularly important for real-time hand pose estimation in the human-computer interaction setting, as well as in the context of user validation on mobile devices. In the latter case, for example, we explore how deeply learned feature extractors can be efficiently incorporated in a generative biometric framework which can be easily adapted to a new user given only the computational resources of a commodity smartphone.

1.3 THESIS ORGANIZATION

The remainder of this thesis is organized as follows:

— Chapter 2 gives an overview of existing state-of-the-art computer vision approaches to action and gesture recognition, as well as human pose estimation, to provide a background for the present work.

— In Chapter 3 we temporarily put the application aside and dive deeper into general machine learning (and, in particular, deep learning) aspects to review relevant existing models in more detail.

— Chapter 4 describes our research findings in the context of human gesture recognition from multiple modalities and discusses corresponding modeling and fusion aspects.

— Chapter 5 is dedicated to the problem of hand pose estimation and introduces a number of supervised, unsupervised and semi-supervised approaches allowing for integration of synthetic data in a real application.

— Chapter 6 approaches the problem of user validation from a combination of non-visual data signals and discusses learning dynamic data representations as a part of a biometric framework.

— Finally, Chapter 7 concludes the thesis by discussing remaining challenges and future directions.

History doesn't repeat itself, but it does rhyme.

— Mark Twain

2

BACKGROUND: VISUAL RECOGNITION

This chapter provides a review of existing *traditional* approaches to human motion analysis from visual data, i.e. various methods which mainly exploit *specifically designed* static and dynamic descriptors, along with anatomical constraints. Even though the majority of manuscripts discussed here are not directly related to the research presented in this thesis (with respect to the chosen methodology, *deep learning*), they nevertheless provide a necessary background and insights for ongoing research and are essential for understanding challenges of the domain. For more general methods in a spirit of deep learning, giving inspiration to our work, the reader is invited to proceed to the next chapter.

The field of human motion analysis from vision is fairly broad, as it covers a large variety of tasks differing in scale, both spatially and temporally (such as, for example, gesture, action and activity recognition). Although similar approaches can often be applied to different problems, one or another aspect may play more or less significant role, depending on a level of abstraction. In this chapter, we will focus on a near-range subset of action-related problems, zooming in at hand gesticulation but also mentioning more general methods, when applicable.

The research area, which covers hand motion modeling and analysis, can be further quantized in a set of typical subproblems, listed below (see Figure 4):

— **gesture spotting and recognition** (including sign language interpretation), i.e. assigning a visual input instance with a single-valued category number from a predefined gesture or sign vocabulary. This can be done both from egocentric videos or from a third person view;

— **authentication from gestures**, i.e. an inverse formulation of the previous task, where the goal is to identify the person performing a gesture, rather than the

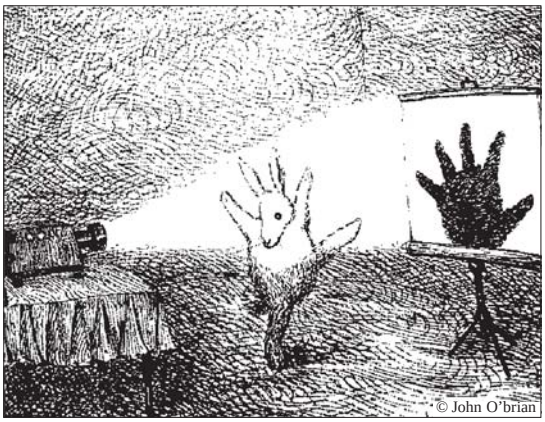


Figure 4 – Different applications of human motion analysis: action and gesture recognition, authentication from motion, pose estimation and motion capture.

gesture category itself. Another research problem, which can be assigned to the same group, but involves whole body motion rather than just hands, is *gait recognition*;

— **hand pose estimation**, i.e. mapping an observed input to a point in a high-dimensional space of possible hand configurations, usually taking into account anatomical constraints. This problem can be directly extended to the whole body pose estimation;

— **hand motion capture**, the goal of which is developing algorithms, defining optimal physical setup and conditions for capturing movements of a single hand with the highest accuracy (i.e unlike the hand pose formulation, whose goal is optimizing for accuracy given the data, here the objective is to optimize the data acquisition per se). The same applies to the problem of motion capture of the whole body, which is typically easier;

— **modeling hand-object interaction**, typically involving recognition and modeling grasp-like gestures with particular focus on handling occlusions and physical collisions with different objects.

Although each gesture-related problem from the given list has a direct analog in full body motion analysis, modeling deformation of a hand is generally considered to be more challenging due to its smaller size, greater variety of poses and higher degree of self occlusions.

In our work, we will focus on three first directions, namely gesture recognition, hand pose estimation and authentication from motion. Technically speaking, *hand pose estimation* can itself serve as an initial step for the task of *gesture recognition*, but this problem deserves special attention as it leaves potential not only for identifying gesture category, but also for estimating additional gesture parameters, such as amplitude, speed, energy, etc. To the best of our knowledge, there are no existing regression based methods allowing to precisely estimate gesture parameters

without explicitly relying on hand geometry reconstruction. Therefore, developing gesture-based interaction interfaces in this context remains an open problem.

To introduce some structure in the following discussion, we will adhere to the following taxonomy, even though boundaries between some groups of methods in recent works start to vanish.

— **Target application**: recognition (i.e. gesture detection and classification) or reconstruction (i.e. pose estimation);

— **Nature of gestures**: static or dynamic. In the context of human-machine interaction, gestures are typically dynamic, however static gestures are still actively researched by the community due to their particular importance for automatic sign speech translation.

— **Nature of input**: color and intensity images or video, stereo and multiple views, depth maps. Exploiting depth and combined RGB-D sensors (and, consequently, a multi-modal input) in the context of human-machine interaction is becoming a pronounced trend.

— **Modalities used for motion analysis**: raw visual signal or articulated pose, aka skeleton (output of a pose estimation step, which in this case is considered as a separated task). As soon as hand skeleton is provided, motion estimation is not a vision problem anymore, therefore here our attention will be drawn mostly to the pose estimation per se.

— **Spatial strategy**: gesture recognition methods can typically be split between two main groups, namely appearance based (or data-driven) methods and methods explicitly based on 3D modeling. In the former strategy, the main computational burden lies in a preliminary offline step of training a discriminative classifier on a large amount of pre-collected data. In the latter case, the classification is performed on-the-fly by inverse rendering, where the whole space of hand configurations is searched for the one which best explains the observed image frame. Both mentioned strategies rely on some sort of feature extraction. In the first case, these features are then fed to a classifier and serve to the purpose of smart dimensionality reduction. In the second case, efficient data representation is required for metric construction. However, if in the hand model-based approaches extracted features are usually straightforward, semantic and essentially geometrical (and can be, typically, silhouettes or masks), in appearance-based methods these data representations can be much more complex and abstract, as long as their performance for the given discriminative task is justified. 3D model-based formulation can be seen as an optimization problem and typically narrows down to developing an efficient search strategy, construction and minimization of a target cost function in a high dimensional space.

— **Temporal strategy** is an another important aspect, except for those cases when gesture recognition or pose estimation are performed from a single image, and the temporal aspect is therefore not involved. Due to the fact that hand gestures typically can be decomposed in a set of characteristic poses with a certain order of transitions between them, sequential models seem especially promising and are broadly used in existing literature. Alternatively, gesture recognition from

video can rely on direct classification of temporal (or spatio-temporal) features, such as, for example, optical flow, trajectories [178] or descriptors, which are directly extracted or learned from spatio-temporal volumes of concatenated video frames.

State-of-the-art work on gesture and action recognition and pose reconstruction has been surveyed multiple times over the last two decades. An interested reader may refer to [2, 240, 312] for a more detailed review of action recognition methods, while other surveys [193, 320, 265, 200, 155, 154, 133, 120, 85] are focused mostly on gesture recognition and hand pose estimation. Furthermore, introducing range sensors has given an additional impetus for vision scientists and resulted in forming a new research direction such as human motion analysis specifically from depth data (surveyed in [48] and [333]).

A conventional method for action recognition and pose estimation typically involves the following sequence of steps:

— **preprocessing operations** (i.e. localization of object of interest, such as a human body or a hand, tracking, background subtraction, segmentation, denoising);

— **constructing** or **learning data representations** (i.e. transforming the visual input in a format more suitable for further processing): in general case, this operation can be realized in several steps as a cascade of successive data reformulations and compressions;

— **matching, regression or classification**, involving one or another machine learning algorithm or optimization based on a predefined metrics.

In next sections, we will briefly discuss each of the steps listed above, starting with **preliminary operations** (Section 2.1), then moving to existing approaches to feature extraction and modeling temporal dependencies for hand **gesture recognition** (Section 2.2) and finally focusing on a challenging problem of **hand pose estimation** (Section 2.3). As we have mentioned at the beginning of this chapter, **deep learning** methods are not included in this review and will be a subject of the next chapter.

2.1 PREPROCESSING AND HAND DETECTION

Preliminary steps include a set of well established techniques and routines that are used in different combinations depending on the type of input data. Generally, the main goal of this step is to localize and segment the object of interest (i.e. hands, in our case) from background, while eventually compensating for noise and various artifacts. Some methods and datasets assume this step to be preformed in advance, however in practice, quality of data preprocessing is an issue which may affect an algorithm performance in a crucial way. Supančič et al. [134], for example, have shown that in real life conditions, hand segmentation may be challenging, and poor performance on this step completely sabotages gesture recognition and hand pose estimation (see Figure 5).

Different approaches to hand localization, which are based on color images, often involve *skin color detection*. In simple contexts, it works fairly well, due to the fact that in appropriately chosen color spaces, skin tones are typically distinct from object

Figure 5 – An example of pixel-wise hand detection [296] (on the left) and of a failure case of hand pose estimation in a case of poorely segmented hand [134] (on the right). Figures are reproduced from the corresponding papers.

colors. There exist several surveys on color spaces and methods for skin detection (for example, [260, 241]). In particular, Schmugge et al. [260] have demonstrated that HSI is a one of color spaces the most suitable for this task.

However, hand detection, which is based solely on skin color, remains highly sensitive to illumination, race and individual differences and backgrounds. As a practical ad-hoc solution, a more robust *face detector* is often used first to compensate for differences in illumination and to estimate the expected skin tone, plus to define an approximate expected position of hands. Alternatively, a hand detector can be learned from a large amount of pre-collected data by, for example, training an SVM on extracted HoG descriptors [178]. Van den Bergh and van Gool [301] proposed to use a pre-trained Gaussian Mixture Model (GMM) to detect a possible range of skin colors under different illumination conditions. GMM in this approach is combined with skin tone estimation from a face of each given user, and the resulting probability of each pixel to belong to a skin region is then thresholded to produce a hand mask. Furthermore, they improve their method by calculating and enforcing a prior on the distance between the face and the hand in a 3-dimensional space using depth information from a time-of-flight camera.

Development of range sensors has extended the space for extracting spatial features to 6 (or, strictly speaking, 5.5) dimensions, including 3 spatial dimensions and 3 color channels. Most existing systems, however, do not treat all of them equally. Color images, for example, are often used exclusively for hand detection, while discriminative features for following gesture classification are calculated based on depth maps. At the same time, there is a number of works exploiting the opposite strategy, where hand positions are calculated based on point cloud segmentation, and spatial features are color or even intensity-based descriptors.

Speaking about solely depth-based methods, hand detection in this case is typically done by simple thresholding of the depth map, region growing [159] with, eventually, following refinement (by, for example, wrist detection). Otsu's method is typically used to separate a hand from a background [287, 164, 284]. In these cases the hand is typically assumed to be the closest object to the camera. Oberweger et al. [215], for example, extract a cube around the center of mass of the closest ob-

ject and rescale it along three dimensions to a predefined fixed size. The following preprocessing steps for depth maps may consist in hole filling.

Alternatively, Tompson et al. [296] trained the Randomized Decision Forest (RDF) to perform *binary classification of each pixel* of a depth map as either belonging to a hand or to background (see Figure 5). Even though for the gesture recognition problem hand detection on the pixel level may not be strictly required, having a clean and accurate hand mask appears to be crucial for the task of hand pose estimation. In their implementation, given a depth image I, each node of a decision tree evaluates the following expression (inspired by an original idea of Shotton et al. [266], which will be discussed in Section 2.3):

$$I\left(u + \frac{\Delta u}{I(u,v)}, v + \frac{\Delta v}{I(u,v)}\right) - I(u,v) \geqslant \tau, \tag{1}$$

where (u,v) are pixel coordinates of the point in the image where the feature is evaluated, $I(u,v)$ is a corresponding depth value, Δu and Δv are learned pixel offsets along each coordinate axis, and τ is a learned threshold. The offsets and the threshold, used by different weak learners in the tree, are log-spaced to cover a large range of possible values. Due to the fact that RDFs are easily parallelizable and their complexity is logarithmic in the number of parameters, this step comes at almost no cost in terms of time and can be efficiently incorporated into real-time gesture recognition or pose estimation frameworks.

Since the majority of depth-based approaches rely on using structured light sensors, which provide not only range information, but *full body skeleton* (extracted algorithmically from depth frames), this additional information can be used directly as initialization of hands positions. However, jitter compensating filtering on joints position typically causes unwanted delays in skeleton tracking, and for rapid gestures more complex hand stabilization may be required. Furthermore, in the case when palms are parallel to the optical axis of the sensor, hands appear to be poorly visible, and their positions are likely to be estimated incorrectly.

Finally, in some works, hand detection is based on segmentation from motion, in assumption that other objects in the scene remain still.

2.2 GESTURE RECOGNITION

Discussing the problem of *gesture recognition* in *traditional setting*, we will start by reviewing existing approaches to designing general and hand-specific descriptors, and then will discuss common strategies of modeling temporal dependencies between video frames.

Before we proceed, we need to mention a number of standard gesture recognition benchmarks that are used by the community for solving different tasks.

2.2.1 *Gesture recognition benchmarks*

While there exist a great amount of benchmarks, which target general problems of *full body action* and *activity* recognition, a number of datasets, which are suitable

for exploring specifically *hand movements* in detail, is still limited. Below we briefly describe the most important ones from this category.

— **MSRC-12 Kinect Gesture Dataset** [93] includes 6244 gesture instances collected from 30 people and corresponding to 12 categories (such as crouch, shoot, throw an object, karate kick) and basic controlling actions (such as navigating a menu and switching the music on and off). The data contains tracks of 20 full body joints captured with the Kinect sensor.

— **Sheffield KInect Gesture (SKIG) Dataset** [181] contains RGB and depth recordings or 1080 gesture instances corresponding to 10 simple gesture types (circle (clockwise), triangle (anti-clockwise), up-down, right-left, wave, "Z", cross, come-here, turn-around, and pat), performed with three different hand postures (fist, index and flat).

— **The ChAirGest Multimodal Dataset** [253] consists of 1200 annotated gestures from 10 classes, collected from 10 subjects under different lighting conditions and starting from 3 different resting postures. In addition to RGB and depth data, this dataset includes data captured with 4 Inertial Motion Units attached to the right arm and to the neck of a subject.

— Starting from 2011, ChaLearn has been proposing a number of gesture recognition challenges and corresponding gesture datasets. The first version of it, **CGD2011** [116], targeted the problem of *one shot learning*. For that competition, the data corpus included 50000 gesture instances from over 30 categories (such as Activity, Body Language, Dance, Emblems, Gesticulations, Illustrators, Pantomime, Sign language, Signals), with each category consisting of from 8 to 12 subclasses.

— Unfortunately, today there exist a single data corpus, which contains sufficient amount of data necessary for development of *data driven* gesture recognition methods. This dataset was first released under the 2013 ChaLearn Multi-modal Challenge on Gesture Recognition [86] and then, in its second reduction, as a part of 2014 ChaLearn Looking At People Challenge. At the present time, this data corpus is known as **Montalbano Gesture Dataset** [87]. It consists of more than 14000 gesture instances drawn from a vocabulary of 20 Italian conversational gestures and performed by several different users. In addition to color and depth video, this corpus contains skeleton information provided by the Kinect sensor (an earlier version of it also included an accompanying audio track). This dataset, both in its first and second reductions, was actively used in our experiments and will be mentioned multiple times throughout this manuscript.

2.2.2 *Generalized descriptors*

In traditional gesture recognition frameworks, the preprocessing step is typically followed by extraction of image descriptors, or features. Under *generalized descriptors* we will understand all content-independent data representation methods, which are typically used in *Bag-of-X* like frameworks, where X may stand for *pixels*, *features*, *visual* and *depth words* (see Figure 6).

Raw pixel values from color images and depth maps can be directly used as features. Naturally, this approach results into extremely high dimensional feature vectors and requires following linear (PCA, for example) or non-linear dimensionality reduction.

In this spirit, Van den Bergh and van Gool [301] combine all pixel values of an intensity image, a thresholded depth map and a segmentation mask of a hand into a single feature vector. In their case, the Average Neighborhood Margin Maximization (ANMM) is then applied for dimensionality reduction and its output is fed to a Haarlet-based classifier [29].

Gradient and edge-based descriptors and their variations are probably the most common in the category of general feature extractors. In the simplest case [242], both intensity and depth images of a hand can be convolved with a bank of Gabor filters at different scales and different orientations. The obtained feature maps can then be averaged across overlapping Gaussian basis functions positioned on a regular grid.

Well known visual descriptors, such as HoG [67], SIFT [183] or SURF [21], commonly used for object recognition and typically extracted around preliminary detected interest points, fall in the same category of gradient-based feature extractors. We will assume though, that the reader is already familiar with their working principles, and skip the detailed description for compactness of discussion.

In the context of gesture recognition, Takimoto et al. [287] proposed a gradient-based descriptor inspired by SURF [21]. In this work, the authors make use of additional information in a form of full-body skeleton provided by modern depth sensors. The hand image, obtained by thresholding a raw depth map, is first rotated in accordance with the direction of the corresponding wrist-elbow *bone* extracted from the skeleton. Next, the image is divided into 8×8 blocks of pixels, and histograms of horizontal and vertical gradients and their absolute values are then computed for each pixel block. Finally, Principal Component Analysis (PCA) is applied for feature dimensionality reduction.

Furthermore, SURF features, extracted from interest points, have been directly exploited by Bao [20] in conjunction with motion trajectories. Yao et Li [330] use the same descriptor for hand posture recognition from color images with an AdaBoost classifier. An earlier work [306] relies on SIFT descriptors [183] in a similar context. A combination of HoG [67] and Haar features is used in [89]. Finally, Yang et al. [327] used Depth Motion Maps (DMM) obtained from 3 projections by integrating depth

Figure 6 – On variety of existing generalized handcrafted descriptors.

images over time and then extracted HoG descriptors from those. Their method is thus related to Motion History Images (MHI), which are used mostly in a number of earlier works [300].

Local binary patterns have been used by [94]. A related feature, called Modified Census Transform (MCT) was considered previously in [144]. For comparative studies on performance of various descriptors, we recommend an interested reader to consult [61, 16, 294].

While all previous descriptors were designed for 2D images, there is also a number of approaches adapted to **depth maps** and **point clouds**: Point Feature Histogram (PFH) and Fast Point Feature Histogram (FPFH) [256], Viewpoint Feature Histogram (VFH) [255], Clustered Viewpoint Feature Histogram (CVFH) [3], a concatenation of VFH with a Camera's Roll Histogram (VFHCRH) [125].

The last descriptor, VFHCRH, has been used, in a combination with various image descriptors and spatio-temporal pyramids, in the context of gesture recognition as a part of Bag-of-Visual-and-Depth-Words classification framework. Zhang et el. [336] proposed another depth-based descriptor called Histogram of 3D facets (H3DF) and applied it to the problem of gesture recognition.

Spatio-temporal descriptors are widely used for recognition of actions and gestures from video. In most cases, these descriptors have evolved from their 2D predecessors, by augmenting existing descriptors with additional features extracted along the temporal dimension.

The spatio-temporal descriptors are typically extracted densely (on a regular grid or globally), sparsely (around salient interest points (STIPS [166])) or along sparse or dense trajectories [305]. Most of spatio-temporal detectors of interest points, just as the descriptors per se, are obtained by extending corresponding algorithms to the third dimension in order to operate on spatio-temporal volumes.

However, it has been shown by [308], that dense sampling generally leads to higher classification rates when it comes to complex realistic data. Among the most widely accepted engineered local descriptors, one could name Cuboid [77], HoG/HoF [167], HoG3D [158], ESURF [317] (extension of SURF [21]), 3D-SIFT [261], MoSIFT [50] (extensions of SIFT [183]) and several others.

Some of the spatio-temporal descriptors have been further extended for depth and RGBD video: 3D MoSIFT [192], CCD [52], LDP [337], DCSF [321]. Once extracted, features are typically treated in the bag-of-words fashion and are fed to a non-linear kernel SVM. Efficiency of proposed spatio-temporal descriptors have been demonstrated by several top performers of 2011/2012 ChaLearn One-shot-learning Challenge on gesture recognition, who have used HoG/HoF [167] and 3D MoSIFT [192] features extracted around interest points [117].

2.2.3 *Shape-based descriptors*

In parallel with general approaches, which are suitable for basically any visual problem involving human motion, a great amount of *ad-hoc* methods have been proposed specifically for *hand gesture* recognition in narrow contexts. Most of them include hand detection, tracking, and gesture recognition, which are based on **global**

hand shape descriptors such as contours, silhouettes, fingertip positions, palm center, number of visible fingers, etc. [279, 185]. Similar descriptors have been proposed for depth and a combination of RGB and depth videos [187, 299].

Relative advantages of general and hand-specific approaches are still a subject of discussion. A recent comparative study [294] shows that in comparison with a specific silhouette-based descriptor (shape context [23]), general HoG appears to be more robust, insensitive to low resolution and segmentation errors. It may therefore be more appropriate for real-world conditions.

Additional geometric information, which is specific to the hand, can be used in a *combination* with locally extracted general descriptors. For example, such an approach has been employed by Li et al. [176] in order to first sample interest points exclusively from a 3D contour of a hand or of a body, and then to treat them in bag-of-points fashion.

Binary segmentation masks alone are often used for recognition and matching in pose reconstruction related problems. In the simplest case, they are obtained from color images or depth maps as a result of background subtraction. Furthermore, operating on point clouds directly and projecting them on 3 orthogonal planes produces two more views in addition to the frontal one [9].

Different shape descriptors can be further extracted from binary masks such as, for example, region-based **Hu moments**, where the first six descriptors encode a shape with invariance to translation, scale and rotation [63]. The seventh component ensures skew invariance, which enables to distinguish between mirrored images. Priyal et Bora [226] use **Krawtchouk moments** instead.

Convex hull and convexity defects on the contour are used by [228], where classification is based on the minimum distance between Fourier shape descriptors [63]. Okkonen et al. [222] extract an affine-invariant **Fourier descriptor** from a hand contour and fed it to an SVM. A Finite State Machine (FSM) if then used for modeling temporal dependencies. Ren et al. [247] represented hand silhouettes as time-series contour curve. In their frameworks, such curves are matched using Finger Earth Mover's Distance (FEMD), originally proposed for measuring distances between signatures of histograms.

Another group of depth-based methods exploits **occupancy grids**. To preserve spatial context information, Vieira et al. [303] proposed to use space-time occupancy patterns on a 4D space-time grid for classifying human action from depth sequences. In the context of hand gesture recognition, occupancy grids have been used by Kurakin et al. [164], who proposed to use cell and silhouette features extracted from thresholded depth maps of hands. In their case, an image is split into square cells of different sizes. For each cell they then compute the occupied area and the average depth, which is called a cell occupancy feature. For the silhouette feature, they divide the image into multiple fan-like sectors and compute the average distance of each contour segment from the center. The dimensionality of the obtained feature vector is then reduced with PCA and hand poses are quantized using K-means clustering. Action graph is finally used for gesture classification (originally proposed for action classification in [175]). A similar in spirit combination of 2D and 3D shape descriptors was proposed in [284].

All methods, which have been described in this section so far, aimed on geometry representation without associating it with any anatomical meaning. However, there is another group of approaches where hands are further segmented and positions and orientations of their parts (typically palms and fingers) are estimated and further used for feature extraction.

For example, **finger-based descriptors** coupled with SVM classifiers are used for hand posture classification in [182]. Dominio et al. [79] designed several types of anatomically meaningful descriptors, such as distances of fingertips an the estimated palm center ("distance features") and from a plane fitted on the palm samples ("elevation features"), as well as curvature of a contour of the palm and fingers regions ("curvature features") and the shape of palm region indicating whether each finger is raised or bent on the palm ("palm features"). In order to find positions of hand parts, they first employ skin-based segmentation and a circle fitting algorithm and 3D plane fitting for palm detection. Mestetskiy et al. [190] base their method on morphological skeletons extracted from hand binary masks and use them for calculation of positions of hand joints.

All these ideas bring us closer to the problem of explicit hand pose estimation, which will be discussed later in this chapter.

2.2.4 *Skeleton-based descriptors*

At the present time (as of the beginning of 2016), there is no existing visual technology that could provide a reliable full *hand skeleton* (assuming that the hand is seen from significant distance). However, many classes of gestures can be recognized solely from upper body skeletons without taking into account movements of individual fingers. This has been confirmed, for example, by the 2013 Multimodal Challenge on gesture recognition [86] results, where top participants built their frameworks only on skeleton information, without relying on raw video input.

For simple actions, skeleton-based descriptors, being informative and fairly low dimensional, are particularly efficient when combined with sequence modeling. Lv et Nevatia [184], for example, used 3 coordinates of 23 skeleton joints directly to obtain a 67-dimensional feature vector (for the sake of normalization, however, only one coordinate is used for the hip joint). Xia et al. [322] proposed a skeleton-based descriptor called *Histogram of 3D joint locations (HOJ3D)*, where angular positions of joints are calculated in a spherical coordinate system centered at the hip center joint. The obtained descriptor is then quantized and classified using a HMM model. Similar descriptors have also been proposed in [246] and [43]. Relational pose features [198] have been employed by [328], in a combination with random forests in a multi camera setup. Yang et Tian [326] calculated inter and intra frame differences between positions of joints, applied PCA and obtained a descriptor called EigenJoints.

Since we are mostly interested in *hand gesture* recognition, we will not go deeper into discussion of methods based solely on skeleton descriptors – instead, we address the reader to existing surveys. However, there exist a number of frameworks utilizing a combination of articulated poses with visual inputs. For example, Sung et al. [282, 283] formulate their descriptor as a combination of 4 types of features:

normalized coordinates of 10 body joints and angles, hands positions with respect to the torso and to the head, joint rotations calculated along 9 subsequent frames and RGB and depth HOG features extracted from head, torso and arms regions. Further classification is based on a hierarchical maximum entropy Markov model (MEMM). As a result, they have demonstrated that RGB-based features were sensitive to changes in people appearance but performed especially well in the *person seen before* setting. Skeletal features in combination with depth HOGs appear to be more robust but less informative in user-specific systems.

Wang et al. [309] proposed to explore *depth appearance* of a 3D joint and designed a feature called Local Occupancy Pattern (LOP): a 3D occupancy grid is calculated from a point cloud around a particular joint. The obtained information is combined with pairwise relative positions of the joints and the Fourier temporal pyramid is used for modeling temporal dynamics.

2.2.5 *Sequence modeling*

Treating internal temporal correlations in the same way as spatial dependencies may seem a stretch, since in this formulation, specific properties of the time space can easily get ignored. Alternatively, the temporal dynamics of motion can be modeled by a time-series algorithm.

The state-of-the-art *traditional* (i.e. non *deep learning*) temporal frameworks tend to treat video in a frame-wise fashion, extracting spatial features first and then feeding them into what can be called a *temporal classifier*, i.e. a sequence modeling algorithm performing final action classification. The most recent trend, in this context, is to replace single frames with short sequences (aka short spatio-temporal blocks) in order to capture some instant characteristics of movements. The rest of the framework in this case remains the same.

We must note here, that traditional temporal models may suffer from serious limitations and fail to capture all complexity of more challenging data. Depending on the task, such methods may have difficulties accounting for data nonlinearity, high dimensionality, long-term dependencies, multi modalities, etc. Nevertheless, sequence models have been shown to be suitable for many practical applications, they are well understood and known to be easy to train.

In the category of non-connectionist methods, generative *Hidden Markov Models (HMMs)*, discriminative *Conditional Random Fields (CRFs)* and extensions of both classes have gained mainstream acceptance and have proven to be effective and efficient in many relatively simple recognition tasks. While HMM models joint probability of states and observed features, in CRFs model parameters are conditioned on observations. Most of highly-ranked participants of the recent ChaLearn Gesture recognition challenge claimed to use HMMs, CRFs or similar models [117].

Relative strengths and downsides of these classes of temporal models are still being discussed in the community: a comparative study by Mendoza and de la Blanca [189], for example, has shown that in the context of action recognition, CRFs generally outperform HMMs when using spatial features, but fall behind when optical flow is used.

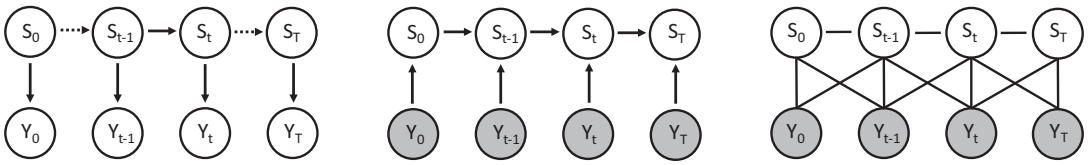


Figure 7 – (a) Hidden Markov Model, which represents $P(S_t|S_{t-1})$ and $P(Y_t|S_t)$. (b) A directed conditional model, representing $P(S_t|S_{t-1}, Y_t)$, i.e. a locally normalized conditional distribution based on the previous state (like in Maximum Entropy Markov Model, MEMM). (c) Conditional Random Field modeling arbitrarily long term dependencies. Gray units show conditioning on observations rather than modeling them. The figure is inspired by [102].

The most widely accepted method for gesture recognition, that came from speech and natural language processing and since then keeps the status of state-of-the-art (or, at least, kept this status until recently), is exploiting one or another modification of **Hidden Markov Models** (HMM), that can be seen as a special case of *Dynamic Bayesian Networks* (see [199] for an overview).

HMMs are a simple and extremely useful class of probabilistic temporal models, having their own type of structure that makes them particularly interesting for many applications. Generally, they are built under two key assumptions (see Figure 7a for a graphical illustration). First, they assume that each observation at a given moment of time is generated by a process being in some *discrete* state S, which is hidden from the observer. Second, this *hidden state* at time t is assumed to satisfy the *Markovian property*, i.e. to be independent of all preceding history of states except for its predecessor at time $t-1$. Consequently, each *observation* also satisfies the Markovian property with respect to the state from which it was generated.

To summarize, the joint distribution over observations Y and states S in HMM is factorized as follows [102]:

$$P(S_0 \dots S_T, Y_0 \dots Y_T) = P(S_0)P(Y_0|S_0) \sum_{t=1}^{T} P(S_t|S_{t-1})P(Y_t|S_t), \tag{2}$$

which shows that an HMM is specified by the probability distribution over initial state $P(S_0)$, a *transition matrix* $P(S_t|S_{t-1})$ of size $K \times K$ (where K is the number of possible hidden states), and *output model* $P(Y_t|S_t)$ (if the observations are also discrete, the output model is specified by its *emission matrix* of size $K \times L$, where L is the number of possible outputs).

Numerous extensions to the classic HMM model have been proposed in order to expand its modeling capacity and to overcome one or another of its limitations. For example, *hierarchical HMMs* were designed to model movements of each body part separately. Such models are more efficient and are easier to train, since breaking down direct connections between movements of different limbs reduces the combinatorial complexity. Moreover, in this setting, *out-of-vocabulary* gestures, which are composed of primitive limb motions, observed previously, can be learned and recognized, which leaves more freedom for adaptive online learning. This class of models has also been applied to a problem of *pose recovery*.

Ikizler and Forsyth [135] individually construct HMMs for legs and arms, whose 3D trajectories are treated as observations. For each limb, states of different ac-

tion models with similar emission probabilities are linked, which allows for automatic segmentation of actions. Chakraborty et al. [44] apply a similar strategy of component-wise HMMs, where arms, legs and a head are first found with a set of view-dependent detectors. Peursum et al. [231] proposed *Factored-State Hierarchical HMM (FS-HHMM)* for joint modeling observations and body dynamics per class.

Another extension, introduced by Duong et al. [82], is called *Switching Hidden Semi-Markov Model (S-HSMM)*, which is designed as a two layer network, where the bottom layer models fine grained activities and the top layer represents a sequence of those. In the same spirit, Caillette et al. [41] proposed to use *Variable Length Markov Models (VLMM)* which are capable of capturing local dynamics along with longer temporal dependencies. Unlike classical HMMs, in these models the memory length (*order*) can change instead of being fixed to a particular value. Natarajan and Nevatia [208] introduced a *Hierarchical Variable Transition HMM (HVT-HMM*) which consists of three layers that model composite actions, primitive actions and poses, which are recognized with low latency.

A number of authors proposed action and gesture recognition frameworks based on **Conditional Random Fields (CRF)** and their modifications [274, 311], driven by an idea to eliminate simplistic and unrealistic assumptions built in the HMMs. A graphical structure of CRF is shown if Figure 7c. Unlike HMMs, these models are potentially capable of modeling longer term temporal dependencies, while offering exact inference and possibility to be trained using convex optimization methods.

Sminchisescu et al. [274] have compared their CRF-based action recognition framework with traditional HMM and a directed model (*Maximum Entropy Markov Model, MEMM*, shown in Figure 7b) and demonstrated that CRF typically outperforms MEMM, which, in turn, typically outperforms HMM. Generally, CRFs have also been used in much broader contexts related to human motion analysis: for example, for multi-person tracking [123].

2.3 HAND POSE ESTIMATION

The problem of *hand pose estimation* was always on the radar of computer vision researchers, in particular due to its practical significance in the context of human-computer interaction. This interest has only grown with emergence of affordable range sensors, which brought the technology on the new level of performance.

Today, there exists a huge body of research dedicated to hand pose estimation, that considers this problem both separately and as a part of gesture recognition pipeline. Among those are works describing first non-visual glove based approaches, and then systems based on video cameras [10, 310, 251, 114], depth sensors [130, 147, 235, 269], and multi camera setups [72].

Providing an exhaustive overview of all existing strategies and their nuances would certainly be worth writing a separate manuscript. For this reason, we trust the reader will excuse us for omitting the description of early efforts in this thesis. Instead, we will aim on giving a more detailed summary of the most strategically important and influential works from the recent past, which are likely to guide research directions in the field in the upcoming years. Meanwhile, the reader interested in the earlier history of the question may refer to two existing surveys on vision based

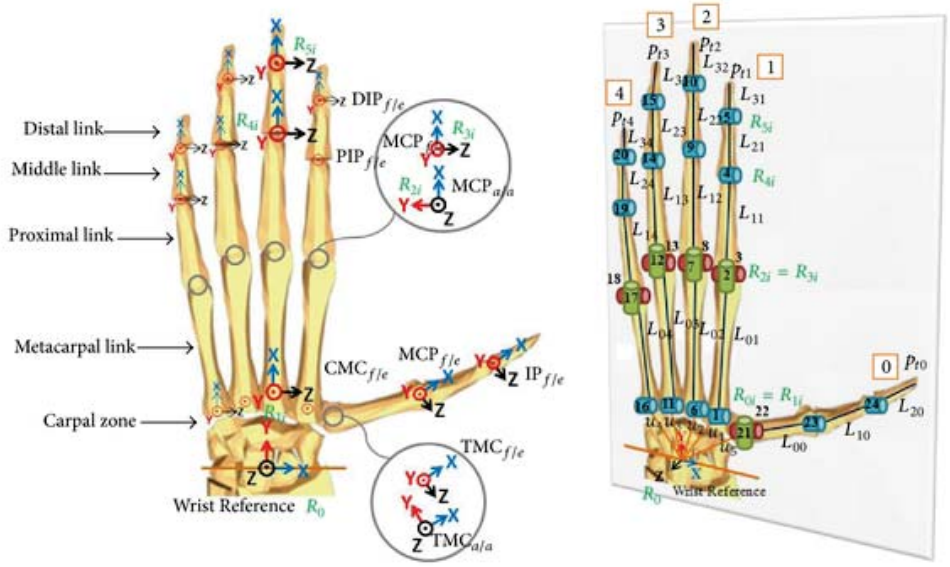


Figure 8 – Kinematic configuration and 24 degrees of freedom of a human hand, as considered in robotics and analyzed by Chen et al. [48] (the figure is reproduced from the paper). The thumb has 3 links and 4 degrees of freedom, while other fingers have 4 links and 5 degrees of freedom.

hand pose estimation [85, 270], which provide a sufficiently detailed snapshot of the field as it was several years ago.

From the computer vision perspective, a **human hand** can be seen as a highly articulated object consisting of 27 bones. A typical kinematic model used for hand pose reconstruction has about 24-26 degrees of freedom (according to different authors), where a palm is usually represented as a single rigid unit (see Figure 8). Naturally, a hand has a number of anatomical constraints associated with it (such as bending ranges of fingers, for example) which are measured, analyzed in the literature [59] and widely used both in mechanical robotics systems and vision-based hand pose estimation and modeling.

The number of degrees of freedom (DoFs) associated with the hand can be further increased by taking into account its position and orientation with respect to the camera, as well as its texture and ambient illumination properties. Hamer et al. [118], for example, proposed a 96 DoF model. Similarly, De la Gorce et al. [73] went beyond purely geometrical descriptors and created realistically looking rendered images taking into account surface properties and lights, adopting Gouraud shading model and Lambertian reflectance. Accordingly, in their work the problem of finding optimal parameters, explaining the observed test sample, also included estimation of texture and illumination.

At the same time, *full* pose reconstruction in some contexts may not be required (as in the case of finger tracking) and the number of degrees of freedom to be estimated can thus be significantly reduced.

Generally, the problem of hand pose estimation is considered to be more challenging than full body pose estimation, not only due to larger number of associated parameters and more pronounced articulation, but also because of massive self-occlusions and mutual occlusions while interacting with different objects.

In our discussion of existing approaches we will stick to the following order. After mentioning the most important hand pose estimation **benchmarks** and providing a brief summary of earlier works on **global matching**, we will proceed by reviewing state-of-the-art **generative models**, which, in traditional setting, almost always include a rendering pipeline and a 3D model of a hand. Most of existing *tracking* approaches are generative, as they allow for a cheap initialization of a hand pose from the previous frame, which limits the search space and allows for precise matching of the observation with the model. An immediate advantage of this group of methods is that physical collisions can be explicitly modeled and therefore anatomically implausible hand configurations can be effectively rejected. However, these methods are known to accumulate tracking errors over time and tend not to recover from significant mistakes.

The next part of this overview will be dedicated to alternative strategies, namely data driven **discriminative methods** and a discussion of how these two opposite logics, generative and discriminative, can be coordinated in a single **hybrid framework**. The data driven discriminative models are formulated as a learning problem and have an offline learning stage where one or another classifier is trained on a large amount of pre-collected data. As opposed to the generative models, this group of methods do not require careful initialization, but, on the downside, they are likely to produce predictions which are not physically meaningful. The hybrid approaches aim on bringing together the best from both worlds and combine a robust discriminative initialization step with highly accurate generative tuning.

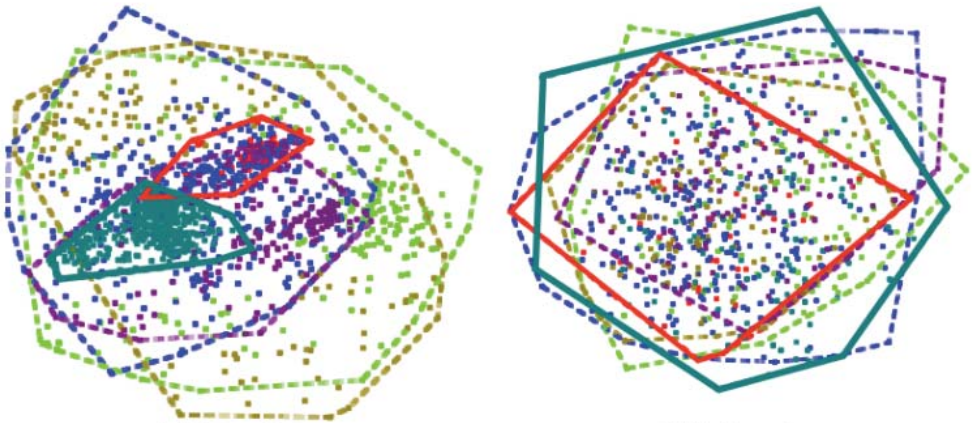
Finally, a detailed overview of state-of-the-art **deep learning** methods for hand pose estimation, both discriminative and hybrid, will be a subject of a separate discussion in the next chapter.

2.3.1 *Hand pose estimation benchmarks*

In this section, we briefly describe the most interesting datasets created specifically for evaluation of hand pose estimation methods. Unfortunately, most of them were created and publicly released when the work, presented in this thesis, was already close to its final stage.

A summary of numerical statistics of each corpus is given in Table 1, while a short description of the purpose and the setup of each data collection is provided below.

— **The NYU Hand Pose Dataset** [296] has been captured with a depth sensor in human-computer interaction setup, from a frontal and 2 side views, where the training set contains samples from a single user and the test set contains samples from two users. Annotations are provided in the form of 36 key points, among which 14 are typically used for pose estimation and for comparison across different methods. In addition, a smaller set of frames is annotated with hand segmentation masks and can be used to train classifiers for pixel-wise hand segmentation from the background.



(a) Position (b) Angle

Figure 9 – Analysis of pose variation in different hand pose test datasets in terms of hand joint positions and joint angles, as performed by Supančič et al. [134]. ICVL dataset [288] is shown in red, NYU dataset [296] is shown in light green, A*STAR [323] is shown in darker green. For each dataset, multi-dimensional scaling is used to estimate the occupied space. As noted by Supančič et al., most datasets cover a similar set of poses (joint angles), but differ significantly in their variety of represented viewpoints (joint positions).

— **The ICVL Hand Posture Dataset** [288] contains depth images with corresponding annotations of 16 hand joints. As shown in Figure 9, the space of hand joint locations, covered by this corpus, is relatively small, compared to other datasets, and corresponds to a single view point. Furthermore, a number of authors reported presence of noisy ground truth annotations [134].

— **The Microsoft Research hand tracking dataset** [243] is a collection of recordings of 6 subjects' right hands, captured using Intel Creative Interactive Gesture Camera and annotated with 21 hand joints positions in 3D.

— **The General-HANDS dataset** [134] (egocentric) was created for *evaluation* of hand detection and pose estimation systems in different contexts. The annotations contain 1) center positions of a hand in each frame, 2) locations of 17 key points and 3) segmentation maps consisting of 19 regions.

— **The CVRR VIVA Challenge: detection, tracking and gesture recognition** [1] – this data corpus includes RGB recordings collected in driving settings from 7 possible viewpoints, including first person view. For detection and tracking, annotations are provided as positions and sizes of tight bounding boxes surrounding each hand in each frame. The subset used for gesture recognition contains 19 classes of driving-related gestures collected from 8 subjects.

— **The HandNet dataset** [316] contains depth images, collected from 10 subjects and annotated with 6 key points (a palm and five fingertips), using an automated magnetic annotation technique.

— **The FingerPaint dataset** [264] is a collection of hand gestures, performed by several individuals in front of the Kinect V2 depth sensor. The ground truth is

1. http://cvrr.ucsd.edu/vivachallenge/

Dataset	GT	Subjects	Training fr.	Test fr.	Used by
NYU [296]	J-36	2	72 757	8 252	[296, 134, 290, 215, 216]
ICVL [288]	J-16	10	331 000	1 599	[288, 289, 134, 290, 215, 281, 225]
MSRA [243]	J-21	6	2 400	2 400	[243]
FingerPaint [264]	S-7	5	56 500		[264]
HandNet [316]	J-6	10	204 971	10 000	[316]
A*STAR [323]	J-20	30	435	435	[323, 134]
Dexter 1 [275]	J-5	1	–	3 157	[275, 134, 264]
MSRA2 [281]	J-21	9	76 500		[281]

Table 1 – Comparison of existing hand pose estimation benchmarks. The letter in a ground truth annotations type (GT) stands for joints (J) or segmentations (S), where the following digits indicate the number of joints or regions, respectively.

represented as segmentation maps with 8 class labels: 1-5 for the five fingers, 6 for the palm and back of the hand, 7 for the forearm, and 0 for background and unlabeled pixels (obtained by simultaneously capturing a video of painted hands with an RGB camera and manually corrected).

— **The A*STAR dataset** [323] is acquired with a time-of-flight camera, while the ground truth annotations are taken from a dataglove. This dataset is created for single-frame hand pose estimation and contains gestures from the American Sign Language and Chinese number counting. This study includes 30 volunteers of different ages and races, both genders.

— **The Dexter 1 dataset** [275] includes video clips of fast and slow hand movements recorded with 5 RGB cameras, the Kinect sensor and a time-of-flight camera (among which RGB and ToF cameras are synchronized). Annotating was performed manually based on 3D fingertip positions from the ToF camera view.

— **The Microsoft dataset MSRA2** described in [281] consists of 76,500 depth images of 17 gestures (mostly from the American Sign Language) collected from 9 subjects. The ground truth annotation was done in semi-automatic way with manual correction.

2.3.2 *Early works and global matching*

Now, it is time to talk about different strategies for building hand pose estimation frameworks in traditional setting.

First, there exist a significant number of works on hand pose estimation by *global matching* of observed visual inputs with pose instances from a training set, which are typically sampled in advance using a synthetic 3D hand model. This strategy, more popular in earlier works, was often used when only a subset of all possible hand configurations could be observed in a given application and when the discrete set of poses was sufficient for interaction. Unlike in rendering based methods (Section 2.3.3), here the hand pose estimation is formulated as a database index-

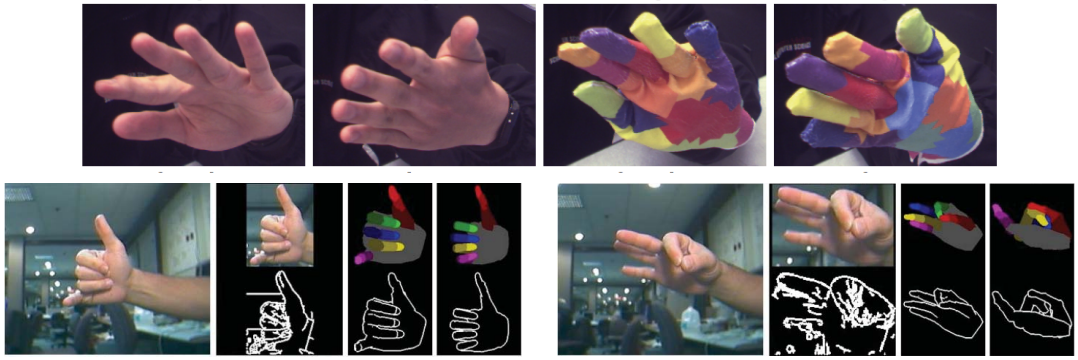


Figure 10 – Hand pose estimation as a database indexing problem. Top row: *palm down* and *palm up* poses as seen from RGB images of bare hands and gloved hands [310]. Bottom row: two examples reproduced from [10], where the first column is an original image, the second one shows the extracted edges, the third is the closest correct entry from the database and the fourth column is the closest incorrect entry.

ing problem, where the search is performed in a space, which is *different* from the original space of kinematic parameters of the hand model.

For example, Athitsos and Sclaroff [10] estimated 3D hand poses from color images, using a synthetic dataset featuring 26 hand shape prototypes, 86 viewpoints and 48 images from each viewpoint representing rotations. In a naive implementation, the search for the closest analog in the dataset could be performed by calculating approximate directed chamfer distance between corresponding edges in the observed image X (with edge pixels $x \in X$) and a reference image Y (with edges $y \in Y$):

$$c(X,Y) = \frac{1}{|X|} \sum_{x \in X} \min \|x - y\| \tag{3}$$

In such an implementation, calculating and storing all distances between the test input and all images in the database would require a lot of memory and result into $O(dn \log n)$ time complexity, where d is the number of images in the database and n is the number of edge pixels. However, using intermediate representation of all images in the database as a set of clamfer distances between them and a small number k of reference images leads to a much faster approximate solution with time complexity $O(kn \log n + dk)$ (where $k << d$). The remaining issue of establishing correspondences between edge pixels, prior to calculating the distances, is solved by probabilistic line matching and discussed in [10] in detail (see Figure 10).

In this context, Wang et al. [310] achieved reasonable quality of hand pose reconstruction by requiring the user to wear colored gloves (also shown in Figure 10).

2.3.3 *Generative models, or optimization through rendering*

The subject of the discussion in this section will be *3D model-based* hand pose estimation, which is essentially a search optimization problem and is therefore built on such blocks as a *hand model*, *metric*, *cost function* and a *search strategy*.

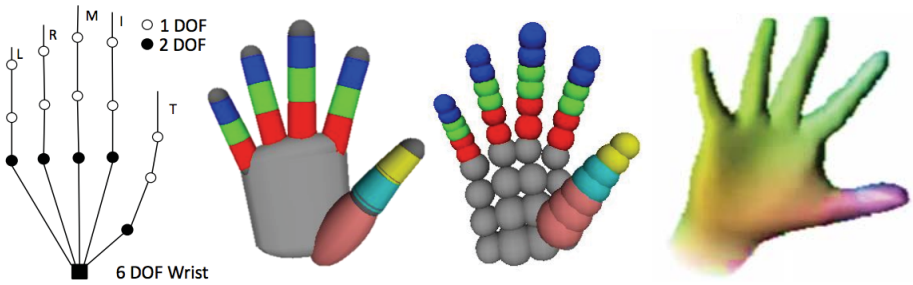


Figure 11 – Hand model: 26 DoF model and its realizations based on cylinders (similar to what is used in [218]), on spheres [243] and on close to realistic 3D meshes [264].

From this group of models, one of the best known recent series of works have been done by Oikonomidis et al. [217, 218, 219, 220, 221] who employ pixel-wise comparison of rendered and observed depth maps with following search over hand configuration hypotheses for the best match (see Figure 12 for examples of hand pose estimation output in different scenarios). More recent generative models also fall into the same pipeline, with numerous optimizations proposed for each step. Let us briefly talk about each of the building blocks which are typically present in each 3D model-based approach, starting from the very hand representation.

In different works, **3D models of a hand** were formulated in several ways varying in their geometric complexity and realism. The general trend is, however, moving from symbolic models, which consist of a number of geometric primitives, to detailed meshes, representing a hand with high degree of fidelity. Most of existing 3D models adopt a 26 DoF hand kinematic structure shown in Figure 11 (or similar).

An immediate advantage of simpler models is their computational efficiency and capacity for parallelism, when it comes to calculating distances and intersections between the primitives. A more complex representation, however, allows for better model fitting through more accurate estimation of correspondences between real observations and rendered depth maps.

The mentioned model by Oikonomidis et al. [218], for example, represents a palm as an elliptic cylinder merged with two ellipsoids, the thumb as a combination of an ellipsoid, two cones and three spheres, and the rest of the fingers as three cones and four spheres each.

Alternatively, in [243] a hand is composed of 48 spheres: 16 for a palm, 8 for a thumb and 6 for the rest of the fingers. In this case, the radii of the spheres are fixed, while positions of their centers are dependent on vector θ of 26 hand parameters corresponding to 26 degrees of freedom. A process of fitting the geometric primitives to a model starts with creation of a polygonal mesh of a hand and empiric adjustment of each sphere size. The same could be done in, probably, more optimal way, by using variational sphere set approximation [308].

At the same time, a more recent work by Sharp et al. [264] directly exploits a hand mesh consisting of triangles and vertices. For a given pose, a 3D position of each vertex is a deterministic function of hand configuration θ including, in their particular case, 58 parameters: global scale (3), global rotation (3), global translation (3), relative scale (1) and rotations (3) of each of 16 joints (3 per finger and a wrist).

The mapping of hand configurations to mesh parameters is done using a computer graphics technique called *linear blend skinning (LBS)*. In this case, a kinematic tree structure of a hand consists of bones $b \in B$, where each bone b has an associated with it rotational transformation $G_b(\theta)$, establishing the correspondence between the local (for the bone) and the global coordinate systems.

For each bone b, transformation $G_b(\theta)$ first applies rotations associated with b, and then recursively calls the corresponding transformation on the parent node, if this parent exists. Each vertex $\mathbf{v}_m$ is skinned to each of the bones b by a sparse set of skinning coefficients α_{mb}. Thus, the position of each vertex $\mathbf{v}_m$, given a set of parameters θ, is defined as follows:

$$\mathbf{v}_m = G_{glob}(\theta) * \sum_{b \in B} \alpha_{mb} G_b(\theta) * G_b^{-1}(\theta_0) * \mathbf{v}_{m,0}, \tag{4}$$

where θ_0 is a base pose corresponding to initial positions of vertices $\mathbf{v}_0$, and the symbol $*$ denotes *applying transformation* (see [292] for more detail). This mapping can be additionally parameterized by a set of hand shape parameters κ, which was used, for example, by Taylor et al. [292] in the process of creating a user-specific hand model from observations.

Different ways of creating 3D models and mesh rigging is alone an interesting topic of research, but it is merely a first step in the general hand pose estimation pipeline. In our context, probably a more important question is a definition of the **cost function** E, which is used to search for optimal hand configurations.

The cost function is calculated based on a given observation I and a rendered image I_r, obtained from a given hypothesis parameterized by θ. Summarizing the recent works, the cost function can be represented as a sum of several general terms:

$$E(I, \theta) = D(I, \theta) + B(I, \theta) + L(\theta), \tag{5}$$

where $D(I, \theta)$ (**metric term**) expresses an integrated distance between the observed depth image I and a rendered depth image $I_r(\theta)$ (given a hypothesis θ), the **silhouette matching term** $B(I, \theta)$ ensures global consistency between these two images and, finally, the **collision term** $L(\theta)$ penalizes self occlusions and, consequently, anatomically incorrect hypotheses. In a Bayesian sense, the first two components are data-attached terms, and the last one can be seen as a prior over poses.

The **metric term** $D(I, \theta)$ is often based on the truncated pixel-wise L1 distance between two images (referred, somewhat fancifully, as *golden energy* in [264, 290]):

$$D(I, \theta) = \sum_{\mathbf{u} \in I} \min\left(|I(\mathbf{u}) - I_r(\mathbf{u}, \theta)|, \tau\right), \tag{6}$$

where $\mathbf{u}=(u, v)$ is a set of 2D pixel coordinates and τ is a depth threshold. This term roughly aligns a point cloud with a hand model, checking for global consistency between corresponding depth maps and efficiently penalizing *model over background* and *background over model* misalignments.

In another hand model, represented as a collection of spheres $\mathbf{C}(\theta)$, which are arranged in accordance with a given hypothesis θ (and specified by their centers and

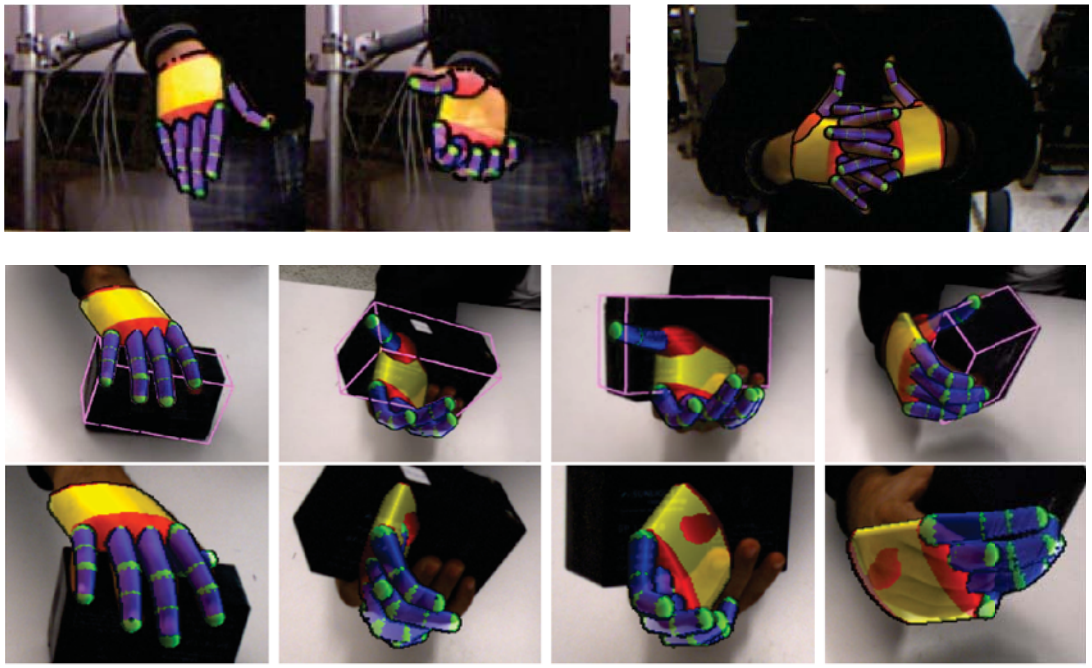


Figure 12 – Output of 3D model-based hand pose estimation: for a single hand [218], for two strongly interacting hands [220] and for hand-object interaction [219].

radii $(\mathbf{c}, r(\mathbf{c}))$, Qian et al. [243] apply L2 distance to corresponding point coordinates in 3D space (instead of L1 applied exclusively to depth values in the previous case) and omit the truncation:

$$D(I, \theta) = \sum_{\mathbf{x} \in \mathbf{P}(I)} \Big| ||\mathbf{x} - \mathbf{c}(\mathbf{x})|| - r(\mathbf{c}(\mathbf{x})) \Big|, \tag{7}$$

where $\mathbf{P}(I)$ is the point cloud corresponding to the observation I (subsampled in this implementation), $\mathbf{x}$ is a point in the point cloud, $\mathbf{c}(\mathbf{x}) \in \mathbf{C}(\theta)$ is the closest sphere (represented by 3D coordinates of its center) to the point $\mathbf{x}$ given the hypothesis θ, and $r(\mathbf{c}(\mathbf{x}))$ is the radius of this sphere.

Please note, that the formulation of the golden energy given by equation (6) does not involve any normalization terms, as in [264] the hand region is assumed to be pre-segmented from the image and normalized to a given size. Otherwise, some normalization would be required.

In the original work by Oikonomidis et al. [218], for example, an RGB input is used along with the depth stream to produce initial pixel-wise hand detection based on skin color, which results in creating a hand binary mask I_m. This segmentation mask, along with the rendered image I_r, are used to compute a residual map I_{res}, where each pixel value is set to 1 if the difference between the rendered map and the real depth observation is less than a certain threshold τ (and to 0 otherwise). These two additional images are used to normalize the integrated distance by the area occupied by the hand:

$$D(I, \theta) = \frac{\sum_{\mathbf{u} \in I} \min \left[|I(\mathbf{u}) - I_r(\mathbf{u}, \theta)|, \tau \right]}{\sum_{\mathbf{u} \in I} \left[I_m(\mathbf{u}) \vee I_{res}(\mathbf{u}, \theta) \right] + \epsilon}, \tag{8}$$

where ϵ is a small offset preventing from dividing by 0.

Finally, one may note, that the global formulations of the distance term, discussed so far, are expensive to compute, as the integration is performed over all pixels in a depth map (assuming there is no subsampling). Instead, Tang et al. [290] perform local computations of each term exclusively at points, corresponding to estimated locations of hand joints $j \in \mathbf{J}(\theta)$. In Section 2.3.6, we will discuss how such a preliminary joint localization is performed.

In the case of Tang et al. [290], the distance term is formulated as follows (dubbed *silver energy*, to rhyme with equation (6)):

$$D(I, \theta) = \frac{1}{\alpha} \sum_{j \in \mathbf{J}(\theta)} D'(j) \tag{9}$$

$$D'(j) = \begin{cases} 0 & \text{if } \nexists I(j_u, j_v), \\ \max\left[I(j_u, j_v) - j_z - \tau_1, \alpha\right] & \text{if } I(j_u, j_v) - j_z > \tau_1, \\ \max\left[j_z - I(j_u, j_v) - \tau_2, \alpha\right] & \text{if } j_z - I(j_u, j_v) > \tau_2, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where (j_u, j_v) are pixel coordinates of the joint projected on the image plane, j_z is its depth, τ_1, τ_2 are thresholds, α is a coefficient for smoothness. In this case, $\nexists I(j_u, j_v)$ stands for the case when the pixel (j_u, j_v) lies *outside* of the silhouette of hand image I. Of course, this term is only meaningful in a combination with a silhouette matching term, explicitly penalizing such misalignments.

The general purpose of the second term of equation (5) (which we will call the **silhouette matching** term) is to make sure that the proposed model lies inside the observed point cloud. In the example with the sphere-based model [243], this term is defined as follows:

$$B(I, \theta) = \sum_{x \in \mathbf{P}(I)} B'(\mathbf{x}, \mathbf{c}(\mathbf{x})) \tag{11}$$

$$B'(\mathbf{x}, \mathbf{c}(\mathbf{x})) = \begin{cases} \max\left[I(c_u(\mathbf{x}), c_v(\mathbf{x})) - c_z(\mathbf{x}), 0\right] & \text{if } \exists I(c_u(\mathbf{x}), c_v(\mathbf{x})) \\ \text{dist}\left[(c_u(\mathbf{x}), c_v(\mathbf{x})), \pi(I)\right] & \text{otherwise} \end{cases} \tag{12}$$

where $(c_u(\mathbf{x}), c_v(\mathbf{x}))$ is a set of 2D coordinates and $c_z(\mathbf{x})$ is a depth value of a sphere center $\mathbf{c} \in \mathbf{C}(\theta)$ on the image plane ($\mathbf{c}(\mathbf{x})$ is the closest sphere to the point $\mathbf{x}$), and $\pi(I)$ is a silhouette of the hand in image I. As before $\exists I(c_u(\mathbf{x}), c_v(\mathbf{x}))$ denotes the case when the pixel $(c_u(\mathbf{x}), c_v(\mathbf{x})$ lies *inside* the hand image silhouette. This way, the term penalizes those configurations, where the z-coordinate of some sphere center is less than the corresponding depth value in the image, and also if the projection of the sphere center falls on the background in the depth image.

In the case when only several point-wise checks are performed at the locations of pre-estimated hand joint positions $j \in \mathbf{J}(\theta)$ [290], the silhouette matching term is defined as follows:

$$B(I, \theta) = \sum_{j \in \mathbf{J}(\theta)} \max\Big[\min\left[\text{dist}((j_u, j_v), \pi(I)), \tau_b\right], 0\Big], \tag{13}$$

where each component in the sum is clamped to the range $[0, \tau_b]$, $\pi(I)$ is a silhouette of the hand in image I and (j_u, j_v) are projected coordinates of the hand joint $j \in \mathbf{J}(\theta)$.

Alternatively, Oikonomidis et al. [218] formulate this term via discrepancies between the skin-based hand mask and the rendered image:

$$B(I, \theta) = 1 - \frac{2 \sum_{\mathbf{u} \in I} (I_m(\mathbf{u}) \wedge I_{res}(\mathbf{u}, \theta))}{\sum_{\mathbf{u} \in I} (I_m(\mathbf{u}) \wedge I_{res}(\mathbf{u}, \theta)) + \sum_{\mathbf{u} \in I} (I_m(\mathbf{u}) \vee I_{res}(\mathbf{u}, \theta))} \tag{14}$$

Finally, the third term of equation (5) is needed to penalize anatomically implausible configurations and is called the **collision term**. In the case of sphere-based model [243], the brute force check for self-collisions is straightforward and is calculated for each pair of spheres from $\mathbf{c} \in \mathbf{C}(\theta)$:

$$L(\theta) = \sum_{\mathbf{c}_1, \mathbf{c}_2 \in \mathbf{C}(\theta)} \max(r(\mathbf{c}_1) + r(\mathbf{c}_2) - \|c_1 - c_2\|, 0) \tag{15}$$

In this work, only collisions between neighboring fingers are computed for efficiency.

Similarly, Oikonomidis et al. [218] perform this check only for adjacent pairs of fingers, by calculating angular difference φ between abduction-adduction angles for those fingers in the hypothesis θ:

$$L(I_r(\theta)) = \sum_{f \in F} -\min(\varphi(f, \theta), 0), \tag{16}$$

where F denotes the three pairs of adjacent fingers (without the thumb).

To summarize, we would like to note that while equation (5) provides the most general formulation of the cost function, which is typically used in the literature, in some works, one or another term may be omitted for simplicity and faster inference. In particular, in the most recent works [264, 290] explicit check for self-collisions is not performed.

The **search strategy** is a crucial step of the generative models based of rendering and its optimization is particularly important due to high complexity of straightforward solutions. In this context, the evolutionary **Particle Swarm Optimization (PSO)** algorithm, originally developed by Eberhart and Kennedy in 1995 [150], deserves special attention as it remains a popular optimization strategy for similar tasks also in more recent works on hand pose estimation. Generally, the popularity of stochastic evolutionary algorithms in this context is explained by the fact that in the presence of noise, discontinuities and uncertainties, which are typical for this problem, many other optimization mechanisms, such as gradient descent, fail to converge to a robust solution.

PSO was initially inspired by modeling in biological systems (more precisely, bird flocking) and is strategically close to the family of Genetic Algorithms (GA), while being technically simpler and missing some of basic GA concepts (e.g., mutations).

Starting with random initialization, the algorithm updates a number of solutions (i.e. *particles*, or, originally, *birds*) on each iteration. Each *bird* has position $\mathbf{x}$ and velocity $\mathbf{v}$ associated with it in the parameter space and acts in accordance with the following strategy: at each moment of time t it follows another bird from the

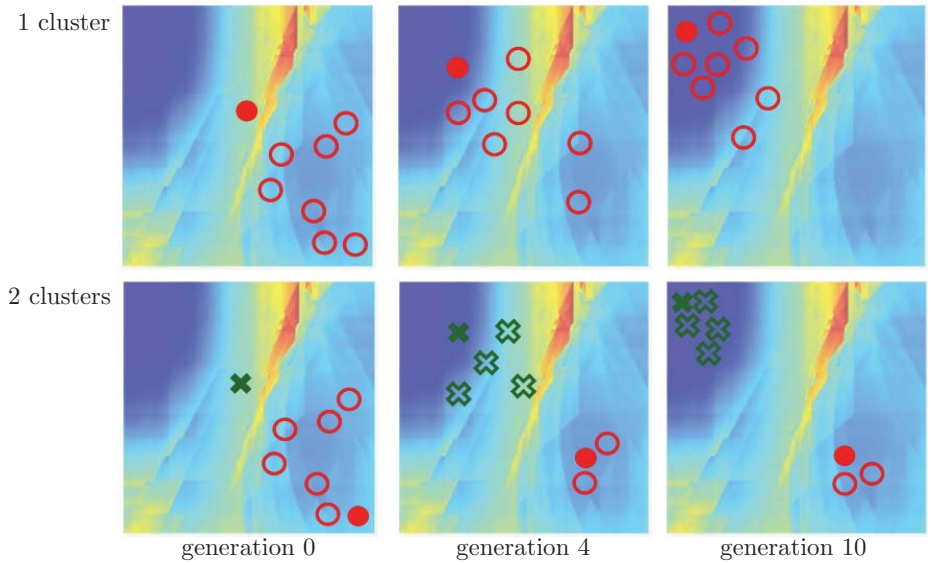


Figure 13 – ICP-PSO optimization process proposed in [243] (the figure is reproduced from the paper). The top row corresponds to the case of a single cluster and the bottom one corresponds to two clusters.

population that got the closest to the optimal solution so far (global optimum), while remembering its own best position (local optimum). The update equation for the velocity and position are formulated as follows:

$$\mathbf{v}^{t+1} = w(\mathbf{v}^t + c_1\mathbf{r}_1(\mathbf{x}_{best} - \mathbf{x}^t) + c_2\mathbf{r}_2(\hat{\mathbf{x}}^t - \mathbf{x}^t)), \tag{17}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{v}^{t+1}. \tag{18}$$

where w [58], c_1 and c_2 are constant parameters (c_1+c_2>4 [58]), $\mathbf{x}_{best}$ is the best value achieved by a given particle and $\hat{\mathbf{x}}^t$ is the best solution achieved so far by the whole population. Vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ introduce stochasticity and are sampled from a uniform distribution in the range $[0 \dots 1]$.

After the original work, a number of extensions and improvement to the algorithm have been proposed, with particular focus on minimization of convergence time. In a case of poor initialization and non-convexity of the error function in a high-dimensional parameter space, this algorithm is likely to result in local suboptimal solutions. To increase the quality of parameter estimation, some authors introduce additional random periodic perturbation into particle parameters along different subsets of dimensions [331, 218]. Different variations of evolutionary optimization algorithms have been proposed, including Sobol Search [221] which resulted in 4x speed up of the optimization step for a single hand.

Furthermore, Qian et al. [243] improved their PSO framework by incorporating an additional step based on the **Iterative Closest Point (ICP)** algorithm [33] which uses gradient-based optimization. We must note here, that this is not the first attempt to exploit ICP (or similar methods) in the context of hand tracking and hand pose estimation: for earlier works see, for example, [36, 73, 177]. But generally, as we mentioned previously, gradient-based methods and, in particular, ICP methods alone do not allow for robust real time tracking of such highly articulated objects

Algorithm 1 ICP-PSO optimization method

```
initialize the hand pose (from the last frame, for example)
generate random particles around the initial pose
for each generation g = 1 ... G do
    for each particle x ∈ X do
        find the closest point on the closest sphere
        for m = 1 ... M do
            gradient descent on a random parameter (eq. (19))
    k-means clustering of all particles
    particle swarm update within each cluster
return the best particle
```

as human hands. However, introducing an additional ICP descent step before each PSO update, as done in [243], results in more efficient cost minimization and faster convergence.

In case if the reader is not familiar with the idea of the ICP method [33], let us briefly review the basics. Generally, this class of algorithms perform alignment of an observed point cloud with a geometric 3D model of the object in 3D space, by minimizing distances between the model surface and the corresponding points in the cloud. A number of improvements and modifications to the algorithm have been proposed in last two decades, including extensions for modeling not only completely rigid, but also articulated objects using iterative Levenberg-Marquardt (LM) optimization [76, 96]. Since the correspondences between the model and the observed points are usually unknown, for each point sampled from the model surface such algorithms iteratively pick the closest point in the observed point cloud and optimize the following cost function [229]:

$$E = \sum_{\theta \in \boldsymbol{\Theta}} \sum_{\mathbf{m} \in M} \min \|G_{\mathbf{m}} \mathbf{m}(\theta) - \mathbf{x}\|^2, \tag{19}$$

where $\mathbf{m} \in M$ are points on the model surface, $G_{\mathbf{m}}$ is a transformation of the corresponding hand part calculated based on the tree-structured kinematic model for a given pose $\theta \in \boldsymbol{\Theta}$, and points $\mathbf{x}$ belong to the observed point cloud.

The exact ICP-PSO optimization procedure, proposed by Qian et al. [243], is illustrated in Figure 13 and formalized in Algorithm 1. All particles are first sampled randomly in a vicinity of the initialization pose, which is typically taken from the previous frame (or, as in hybrid methods, approximately estimated from the given image, see Section 2.3.6). The first ICP step locally improves positions of all particles and the best candidate quickly moves towards the most promising solution. The following PSO iteration shifts the whole population towards the promising region of the best candidate. In addition to the ICP component, Qian et al. [243] introduced dynamic k-means clustering of the particles which results in faster convergence of the algorithm to a number of local optima (hopefully, including the globally optimal solution, see Figure 13).

Up to this point, talking about different cost functions and optimization strategies, we have always assumed that the **initialization hypothesis** is provided, which is

typically the case in tracking-based approaches where each next frame is initialized from the previous one. However, estimating the hand pose at the starting point and, eventually, correction of this estimate to recover from tracking failures has been so far left out of the scope of our discussion. The most recent works in this context are based on a combination of generative and discriminative models, which will be discussed in Section 2.3.6. In a traditional purely tracking-based pipeline, however, this has been done typically with ad-hoc methods of fingertip localization (for example, as extreme points of hand silhouettes or point clouds [177, 243]) or simply by asking the user to take the canonical pose for initialization [218].

2.3.4 *Discriminative models, or hand pose estimation as a learning problem*

Instead of *direct matching* between test and database images, as in Section 2.3.2, or *iterative search* over hand configurations, as in Section 2.3.3, global hand pose estimation can be formulated as a regression **learning problem**.

As in Section 2.2 (discussing the problem of gesture recognition), the discriminative pose estimation pipeline also includes *feature extraction* (or *learning* hierarchical representations directly from the data) and a following machine learning algorithm, which performs *mapping* from the extracted or learned features to hand pose parameters. Generally, these visual features can represent either global characteristics of a hand shape, or can be expressed as local descriptors extracted over a dense grid or around each pixel.

Learning-based pose estimation can be performed in a single step for the whole object (which we will call **holistic regression**), or iteratively, taking into account the underlying anatomical structure of the body and conditioning predictions of "child" joints on detected positions of their "parents". In this section, we will refer to the latter strategy as **hierarchical regression**.

2.3.4.1 *Holistic regression*

Back in 2001, Rosales et al. [251] proposed a holistic hand pose estimation framework, for which they artificially created *synthetic training data* by capturing multiple views of human hand motion using a CyberGlove. Their final training dataset contains 2400 poses, each rendered from 86 uniformly distributed viewpoints. In this work, Hu moments are computed on hand silhouettes and are used as *visual features*. However, instead of direct matching of test images with training samples, as in [10], the output of the system is expressed in *hand pose parameters*, which are estimated by additional *learned mapping* from visual features.

Furthermore, de Campos and Murrey [72] developed a regression-based method for hand pose estimation from *multiple views*. Their approach is based on converting each image into a silhouette contour with following calculation of compact hand descriptors using shape contexts (which are aligned with the hand axis to insure rotational invariance). Descriptors from all cameras are concatenated in a single feature vector. Following an earlier work [1] for body pose recovery, they relate image descriptors and pose settings through a linear combination of basis functions.

Figure 14 – Illustrations of depth-based body and hand segmentation methods, proposed in [266, 151] and based on classification random forests (RDF) (figures reproduced from the original papers). This segmentation step is typically followed by calculating pixel-wise votes for joint positions with a following aggregation stage.

One of the most important recent advances in computer vision in recent years, a seminal paper on **pixel-based body segmentation** with random decision forests by Shotton et al. [266] (see also a journal version [267]), gave birth to a whole group of follow up works, including several where this method was adapted for *hand segmentation* (see Figure 14).

This group of methods is based on pixel-wise classification of a foreground depth image into parts, using an ensemble classifier (random decision forest, or RDF) with following inferring positions of joints from these auxiliary segmentation maps. In a follow up proposal [267], the same RDF, instead of pixel-wise classification, performs direct **pixel-wise voting for joint positions**. As these works build the ground for many hand and body pose estimation algorithms which were proposed shortly after, we will review their main steps in detail.

We have already mentioned one of the strategies, inspired by Shotton el al., while talking about a pixel-wise hand/background segmentation algorithm by Tompson et al. [296] (see Section 2.1), where we also provided an equation for a *split function* (equation (1)). In the original paper, however, this split function, evaluated by each node n in a tree m for each pixel $\mathbf{u}=(u,v)$ from a depth image $I(\mathbf{u})$, is formulated slightly differently:

$$h(\mathbf{u},\theta_{m,n}) = \left[I\left(\mathbf{u} + \frac{\Delta\mathbf{u}_{m,n,1}}{I(\mathbf{u})}\right) - I\left(\mathbf{u} + \frac{\Delta\mathbf{u}_{m,n,2}}{I(\mathbf{u})}\right) \geqslant \tau_{m,n} \right], \tag{20}$$

where $\theta_{m,n}$ is a vector of parameters associated with the given node n from the given tree m, $\theta_{m,n}=\{\Delta\mathbf{u}_{m,n,1}, \Delta\mathbf{u}_{m,n,2}, \tau_{m,n}\}$, $\Delta\mathbf{u}$ are learned depth offsets (sampled randomly from a given range) and $\tau_{m,n}$ is a learned threshold.

This split function given by equation (20) has been reused by a great number of RDF-based frameworks with no change, as it has proven to be effective and computationally efficient.

As a slight modification, a number of works [323, 281], which also include a *preliminary normalization step* compensating for in-plane image rotations by an angle α, formulate this split function as follows:

$$h(\mathbf{u}, \theta_{n,m}) = \left[I\left(\mathbf{u} + \frac{\text{Rot}(\Delta\mathbf{u}_{m,n,1}, \mathbf{u}, \alpha)}{I(\mathbf{u})}\right) - I\left(\mathbf{u} + \frac{\text{Rot}(\Delta\mathbf{u}_{m,n,2}, \mathbf{u}, \alpha)}{I(\mathbf{u})}\right) \geqslant \tau_{m,n} \right], \tag{21}$$

where $\text{Rot}(\Delta\mathbf{u}_{m,n}, \mathbf{u}, \alpha)$ defines a mapping from $\Delta\mathbf{u}$ to its new position as an in-plane rotation of pixel $\Delta\mathbf{u}$ by a preliminary estimated angle α around point $\mathbf{u}$.

Given the split function, each node n in a tree m performs partitioning of the whole set of training pixels S in two subsets, *left* S^L and *right* S^R. During training, a vector of parameters $\theta_{m,n}$ for each node in each tree is learned to minimize an *objective function* H (defined on given depth features) while trying to keep the split between two nodes $\{L, R\}$ balanced:

$$\theta^* = \underset{\theta}{\text{argmin}} \sum_{d \in \{L,R\}} \frac{|S^d(\theta)|}{|S|} H(S^d(\theta)). \tag{22}$$

What remains to define, is a precise form of this objective function H, and this depends on a particular task, performed by an RDF – which, in our case, can be either *pixel-wise classification into parts*, or, skipping this intermediate step, a direct *pixel-wise voting for locations of joints* by regression.

Both algorithms have the same general structure and include three following steps: (1) obtaining predictions for each image pixel, (2) creating a global aggregated distribution of votes over a whole image, and (3) generation of final predictions of joint locations.

In the first step, a **pixel-wise classification forest** learns to output a pixel-wise *probability distribution* $p(c|(u,v))$ over all classes c (i.e. hand or body parts) at each leaf node l of each tree m. The training objective function in this case is defined as Shannon entropy (in the following equations we will sometimes omit the dependency on θ for convenience):

$$H(S) = -\sum_c p(c|S) \log p(c|S), \tag{23}$$

where $p(c|S)$ is the normalized histogram of labels c for all pixels in the set S. To produce a final distribution for a given pixel, output probabilities at corresponding leaf nodes from all trees $l \in \mathcal{L}(u,v)$ are averaged:

$$p(c|(u,v)) = \frac{1}{|\mathcal{L}|} \sum_{l \in \mathcal{L}(u,v)} p_l(c), \tag{24}$$

where $|\mathcal{L}|$ is a number of trees in the forest and $p_l(c)$ is a posterior probability of class c at leaf node l.

The obtained segmentation maps are then used, first, to create an aggregated distribution for a whole image (which we will discuss in a minute) and, second, to infer joint positions on an additional postprocessing step (which we will discuss later in this chapter).

Algorithm 2 Global vote distribution from *classification forests* [267]

1: initialize a set of votes $Y_j = \emptyset$ for all joints j
2: **for** each foreground pixel (u, v) in test image I **do**
3: compute 3D pixel position $(u, v, I(u, v)) \rightarrow (x, y, z)$
4: **for** each tree m in the forest **do**
5: evaluate the tree to reach the leaf node $l(u, v)$
6: evaluate distribution $p(c|(u, v,))$ using eq. 24
7: **for** each joint j **do**
8: compute corrected depth $z'_j = z_j + \zeta_j$
9: lookup relevant body part $c(j)$
10: compute weight $\omega = p(c = c(j)) \cdot z^2$
11: add vote $((x, y, z'), \omega)$ to set X_j
12: **return** set of votes Y_j for each joint j

Alternatively, a **pixel-wise regression forest** is trained to infer *joint positions* directly, in which case for each image pixel $(u, v) \in S$ a corresponding leaf node l contains a set of relative votes $\mathbf{x}_j \in \mathbf{X}_l$ for locations of all joints $j \in \mathbf{J}$ in 3D space.

Training is performed by minimizing a sum H of squared differences between all ground truth positions $\mathbf{p}_j$ and all votes $\mathbf{x}_j$ corresponding to all pixels in the training set $(u, v) \in S$ and integrated over all joints j:

$$H(S) = \sum_{j \in \mathbf{J}} \sum_{(\mathbf{u}, \mathbf{v}) \in S} \left|\left|\mathbf{p}_j - \mathbf{x}_j(u, v) - \mu_j\right|\right|^2, \tag{25}$$

$$\mu_j = \frac{1}{|S_j|} \sum_{(u,v) \in S_j} (\mathbf{p}_j - \mathbf{x}_j(u, v)), \tag{26}$$

$$S_j = \{(u, v) \in S \,|\, ||\mathbf{p}_j - \mathbf{x}_j(u, v)|| < \rho\}, \tag{27}$$

where ρ is a fixed threshold and S_j is therefore a subset of pixels (u, v) in a proximity of the ground truth joint position $\mathbf{p}_j$. As in the case of pixel-wise segmentation, regression-based pixel-wise votes are yet to be aggregated to produce a global vote distribution, which is then used to infer final predictions of joint locations.

Now, it is time to discuss how the **aggregated distribution** (step (2)) over all pixels in an image is obtained. For clarity, we provide pseudo code for two algorithms: the first one corresponds to integration of *pixel labels* in classification forests (Algorithm 2), and the second one to integration of *pixel-wise votes* in regression forests (Algorithm 3) (presentation adapted from [267]).

In Algorithm 2, for a given pixel (u, v), each tree in the forest is first evaluated and the resulting distributions over discrete classes are averaged (lines 2-6). Using camera calibration parameters and a depth value $I(u, v)$, each pixel is mapped back to the original 3D space (line 3).

One should keep in mind, however, that all pixels in the image lie on the body surface, while the final objective is to estimate positions of joints which are located *inside* the body. Therefore, for each joint j, a corresponding depth shift ζ_j is introduced to estimate how far from the surface this joint is expected to be (in terms of differences in depth values), and the depth value is corrected (line 8).

Algorithm 3 Global vote distribution from *regression forests* [267]

1: initialize a set of votes $Y_j = \emptyset$ for all joints j
2: **for** all each foreground pixel (u, v) in test image I **do**
3: compute 3D pixel position $(u, v, I(u, v)) \rightarrow (x, y, z)$
4: **for** each tree m in the forest **do**
5: evaluate the tree to reach the leaf node $l(u, v)$
6: **for** all joints j **do**
7: perform clustering to obtain $\mathbf{V}_{lj}$ (or lookup if previously computed)
8: **for** all $(\mathbf{x}_{ljk}, \omega_{ljk}) \in \mathbf{V}_{lj}$ **do** (where ω_{ljk} are confidence weights)
9: compute absolute position $(x', y', z') = (x, y, z) + \mathbf{x}_{ljk}$
10: compute weight $\omega = \omega_{ljk} \cdot z^2$
11: add vote $((x', y', z'), \omega)$ to set X_j
12: sub-sample X_j to contain at most N votes [103]
13: **return** set of votes Y_j for each joint j

Furthermore, we assume that in reality, a position of a joint always corresponds to a position of a center of mass of some body part c. Therefore, to obtain a global voting distribution for a given joint, pixel-wise probabilities of the corresponding class are used, with squared corrected depths as weighting coefficients, to make the distribution invariant to the distance between the person and the camera (lines 9-11).

The Algorithm 3, for the regression case, is overally similar. However, at each node l the raw distribution $\mathbf{X}_l$ over pixel votes for each joint j is typically multimodal and, before the aggregation process starts (lines 8-11), needs to be clustered to obtain k subsets of pixel votes $\mathbf{x}_{ljk}$ belonging to the same region in 3D space. This clustering step corresponds to line(7) in Algorithm 3, where $\mathbf{V}_{lj}$ is a list of clusters obtained at node l and corresponding to the vote distribution of joint j.

Since the sizes of all obtained clusters can be different, normalization coefficients ω_{ljk} (called confidence weights) are set in accordance with the size of the cluster (see [267] for more detail). Line (8) in the algorithm is formalized for a general case, when several clusters from the same distribution are used for vote aggregation, taken with corresponding confidence weights. In the original implementation by Shotton et al. [267], however, a single most prominent cluster was used.

The output of both algorithms is a set of per-joint global vote distributions which are then used to generate final proposals. This can be done by simply locating centers of mass of the obtained regions, or applying the mean shift algorithms, eventually combined with additional refinement procedures. We will come back to this step a bit later, but before we need to review another important building block of state-of-the-art hand pose estimation systems.

In the case of pixelwise classification, the training objective given in equation (23) has also been extended to include a prior on the label space [141]. This work exploits the fact, that the the classification problem at hand features a structured label space, as pairs of labels can be neighbors in label space (not to be confused with the image lattice) or not. In [141], this additional prior was integrated into the random forest objective (23), whereas in [140], this prior was integrated in a deep neural work used for body part segmentation.

Almost immediately after the work by Shotton et al. [266] came out, the pixel-wise classification RDFs have been adapted to the problem of hand segmentation in [151], see Figure 14. However, while the originally proposed framework demonstrates excellent results on the full body pose problem, its adaptation for hands turned out to be more challenging. This is not at all surprising, given that a human hand is an object essentially more flexible and articulated than torso and limbs. For this reason, algorithms addressing the problem of hand pose estimation generally require higher modeling capacity which, in straightforward implementation, results in increased depth of the RDF trees and therefore higher time and space complexity. To simplify the task, a popular trend in recent works has been to perform preliminary **view point clustering** of data samples in a hand pose space and to train individual *expert* classifiers or regressors for each subset of the training data.

For this purpose, Keskin et al. [152, 153] have proposed what they call *multi-layered RDFs*, where classification of pixels into hand parts is preceded by additional **shape classification forest**, which first assigns each image of a hand with a *pose class label*. In this case, the classification RDF, as in [266], is then trained for each pose class, and the system performs online switching between classifiers depending on the prediction at the first step.

Furthermore, in the same work, two different strategies were explored, namely **global shape classification forest** (Global Expert Network) and **local shape classification forest** (Local Expert Network) with the difference in aggregation of predictions at the first step. In the global approach, all pixel-wise predictions of the shape classification forest are first integrated and averaged over the whole image producing a single pose label per hand. Alternatively, this step was skipped in the local approach, where the following switching between pose-specific classifiers into hand parts is performed separately for each pixel (see Figure 15).

Overall, in this work, the global method has led to better performance scores, however it was noted that the locally organized shape classification, while being less robust to noise, might better generalize to previously unseen hand poses.

Going farther in this direction, Tang et al. [288] implemented a more general framework optimizing for a quality function formulated as a weighted linear combination of several terms reflecting viewpoint classification, hand part (or patch) classification and joint regression. Adaptive switching between those terms is performed in soft fashion, such that at the top of the tree *viewpoint labeling* has the highest importance, gradually giving the priority first to *patch classification* and then to *regression* while moving down along the tree.

In this work, the authors exploit three types of training sets: annotated synthetic hand maps, unlabeled real data and a small amount of labeled real images. First, each synthetic image is assigned with a viewpoint label α (or, strictly speaking, each synthetic image is *rendered* given the viewpoint drawn from a set of 135 quantized viewpoints). Furthermore, all subsampled *patches* from each image are annotated with label p of the closest hand joint (which is basically the same as annotating *pixels* with hand parts in previous works) and vector $\mathbf{v}$ of relative ground truth positions of all hand joints with respect to the center pixel. As a sample goes down the tree, the task gradually shifts from viewpoint clustering first to segmentation and at the

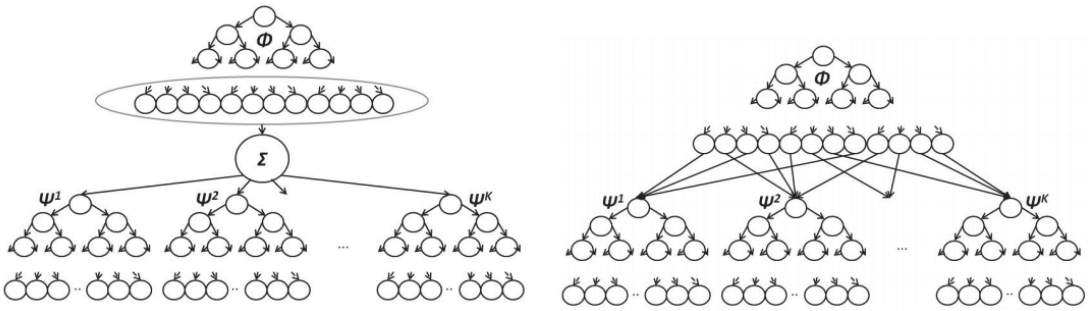


Figure 15 – View point clustering by Keskin at al. [152] (figure reproduced from the original paper). The left image illustrates the idea of *global clustering*, and the right image corresponds to the case of *local clustering* perform in pixel-wise fashion.

end to regression, which is expressed in the following formalization of a quality function:

$$Q_{apv} = \alpha Q_a + (1-\alpha)\beta Q_p + (1-\alpha)(1-\beta)Q_v, \tag{28}$$

where α and β are switching coefficients, Q_a and Q_p are defined as information gains for viewpoint and patch classification respectively, as described in [37] (H_a and H_p is Shannon entropy for a given task):

$$Q_a(\theta) = H_a(S) - \sum_{d\in\{L,R\}} \frac{|S^d(\theta)|}{|S|} H_a(S^d(\theta)), \tag{29}$$

where S is the training set of all image patches. For the patch classification term, the information gain Q_p is calculated in the same way.

The third term Q_v in equation (28) is the regression term minimizing compactness of vote vectors $J(S)$:

$$Q_v(\theta) = \left[1 + \sum_{d\in\{L,R\}} \frac{|S^d(\theta)|}{|S|} \operatorname{tr}\left(\operatorname{var}\left[J(S^d(\theta))\right]\right)\right]^{-1}, \tag{30}$$

where tr is a trace operator. The maximum value of this last term corresponds to the case when all votes in one node are identical.

Switching coefficients α and β in equation (28) are defined at each node, based on empirical purity of classification achieved so far, for the hand pose and the closest hand joint, respectively. Denoting $\Delta(S)$ as a difference in probabilities between the first and the second most probable classes and setting empirical thresholds $\tau_{\alpha,\beta}$, the coefficients are expressed as follows:

$$\alpha = \begin{cases} 1 & \text{if } \Delta_a(S) < \tau_\alpha, \\ 0 & \text{otherwise;} \end{cases} \qquad \beta = \begin{cases} 1 & \text{if } \Delta_p(S) < \tau_\beta, \\ 0 & \text{otherwise,} \end{cases} \tag{31}$$

where $\Delta_a(S)$ corresponds to view point classification and Δ_p corresponds to patch classification. Thus, when one of $\Delta(S)$ reaches a certain threshold and, consequently, the classifier starts to produce confident predictions on a given step, the corresponding coefficient (α or β) gets switched from 1 to 0, and the training objective gets switched to the next task. This formulation results in automatic adaptation of the

RDF to the complexity of the problem by determining a number of splits which are necessary to achieve good performance on different subtasks (as opposed to a single-step pose clustering in the previous work by Keskin et al. [152]).

Another important aspect of hand pose estimation, which has been actively researched by the community and also in the same work by Tang et al. [288], is how **synthetic data** can be efficiently integrated in the training process, along with a large amount of **unlabeled real samples**. This direction is motivated by the idea that modern computer graphics renders, given a parameterized 3D hand model, are capable of producing infinite amount of training data along with perfectly accurate ground truth annotations. At the same time, manual groundtruthing of real images remains a tedious and time consuming procedure, while even the most advanced state-of-the-art motion capture systems, given the scale of the task and general complexity of the hand motion, do not guarantee sufficiently robust solutions. This is a problem which we will also tackle in Section 5 of this thesis.

Along with the labeled dataset of real images, the work by Tang et al. [288] exploits additional subsets of data, namely unlabeled samples of the same nature, plus synthetic depth images of a hand. Accordingly, they introduce additional, *unsupervised* quality function Q_{tss}, expressed as follows:

$$Q_{tss} = Q_t^{\omega} Q_u, \tag{32}$$

where ω is a predefined parameter, Q_u is the unsupervised term applied to both labeled and unlabeled real data (denoted as $\mathcal{R}$, from *real*), enforcing similar looking patches to be assigned to the same node:

$$Q_u(\theta) = \left[1 + \sum_{d \in \{L,R\}} \frac{|\mathcal{R}^d(\theta)|}{|\mathcal{R}|} \operatorname{tr}\left(\operatorname{var}\left[\mathcal{R}^d(\theta)\right]\right) \right]^{-1}, \tag{33}$$

and Q_t is a *transductive* term applied to all labeled samples, both real and synthetic (denoted as $\mathcal{L}$, from *labeled*), and performing alignment of real and synthetic domains through a certain number of provided ground truth correspondences:

$$Q_t(\theta) = \frac{|\{r, s\} \subset \mathcal{L}^l| + |\{r, s\} \subset \mathcal{L}^r|}{|\{r, s\} \subset \mathcal{L}|}, \tag{34}$$

where $\{r, s\}$ are pairs of real and synthetic patches, which are marked as *corresponding*. These correspondences are established through matching patches in synthetic and real labeled subsets, according to their 3D joint locations (see [288] for more detail). This transductive term is thus the ratio of preserved associations $\{r, s\}$ after a split.

Such defined quality function for semi-supervised and transductive learning is used in combination with the purely supervised terms Q_{apv}, while the switching between them is performed randomly.

As we have mentioned earlier, regardless of particular form of quality functions using for training, this group of methods outputs a distribution of hand joint locations which is yet to be post-processed to produce a single vector of coordinates per joint.

For this purpose, Shotton et al. [267] employ a local mode finding approach based on a Gaussian Parzen density estimator, defined as follows:

$$p_j(\mathbf{x}') = \sum_{(\mathbf{x}', w) \in X_j} w \cdot \exp\left(-\left|\left|\frac{\mathbf{x}' - \mathbf{x}}{b_j}\right|\right|\right), \tag{35}$$

where $\mathbf{x}'$ is a vector of coordinates in the 3D space and b is a learned parameter. The mean shift algorithm [62] is then used to find modes in this density, while grouping the votes converging to the same point in 3D space, producing final proposals for joint positions.

A follow-up work [124] introduced a modification added graph cuts segmentation, while Polrola et Wojciechowski [239] proposed another adaptation inspired by [266]. At the same time, the mean shift algorithm is actively used in a number of follow up works, in some cases followed by kinematic refinement of joint positions [288].

2.3.4.2 *Cascaded regression*

Under *cascaded regression* we will understand a group of holistic, by spirit, methods, where positions of all hand joints are estimated simultaneously, but by the means of progressive refinement in several iterative steps. This idea is based on **residual learning**, where each following regressor aims to minimize the residual error between previously obtained predictions and the ground truth:

$$\theta^{(t)} = \theta^{(t-1)} + \mathcal{R}^{(t)}(I, \theta^{(t-1)}), \tag{36}$$

where, $\theta^{(t)}$ is a hand pose hypothesis at iteration t, $\theta^{(t-1)}$ is a hypothesis obtained on the previous step, I is an input image and $\mathcal{R}^{(t)}$ is a regressor trained to minimize the residual error at the given stage t.

In the context of pose estimation, this idea was first explored by Dollar et al. [78] and then served as a starting point for a work by Sun et al. [281] (for an illustration, see the top row in Figure 16). As a further improvement, Sun et al. [281] introduced an additional hand pose normalization transform, which is applied at each stage of refinement to compensate for in-plane image rotations (as given by equation (21)).

It is important to understand, that a crucial difference between these methods and what has been discussed in the previous section, is that here a *whole image* serves as a single input and is processed *at once*, instead of producing votes on a pixel-wise level. Therefore, an output *distribution* of votes for each joint is replaced by a *single vector* indicating a joint position. Naturally, such an approach makes the processing significantly more computationally efficient.

In general, the progressive refinement can be done by a *single* regressor for a whole pose, or by a set of *joint-specific* predictors taking as an input a smaller but higher-resolution regions containing each joint, as estimated by their predecessors. A number of *deep learning* frameworks also make use of this strategy, and we will talk about this in the next chapter.

Holistic cascaded regression methods, discussed so far, do not explicitly take into account a kinematic structure of the hand and do not model anatomical structural dependencies between possible locations of different joints. However, one may argue that this information is provided to the predictor somewhat implicitly by augmenting its input with preliminary estimates of all joint positions, which makes this

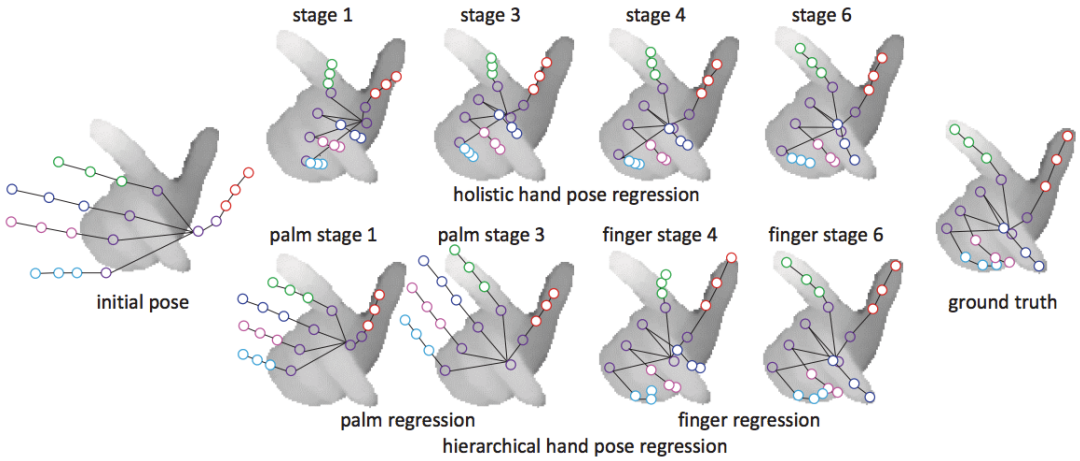


Figure 16 – A comparison of cascaded holistic (top row) and hierarchical (bottom row) regression as presented by Sun et al. [281]. In the first case, positions of all joints are estimated and refined at each step. In the second approach, joints are localized in groups and in a particular order corresponding to their position in a tree-structured hand model.

group of algorithms quite effective as compared to a single-stage pipeline. In a sense, this formulation can be seen as a generalization of hierarchical methods (described in the next section), which are defined specifically on tree structured kinematic graphs.

2.3.4.3 *Structured hierarchical regression*

Now it is time to look closely at some recent regression-based works on hand pose estimation with random forests, which are based on *hierarchical* search, or hierarchical refinement of hand joint positions.

Unlike purely bottom-up strategies (discussed in Section 2.3.4.1), where *pixel-wise* predictions were estimated first and then integrated to produce a final output, the hierarchical methods are based on *iterative optimization*, where a *whole image* is taken as in input at each iteration, as cascaded methods from Section 2.3.4.2.

As opposed to holistic cascaded approaches, hierarchical models make explicit use of underlying hand *structure*, represented as a graph.

A number of recent works [289, 281] explicitly incorporated a topological model of a hand into the training process. Sun et al. [281], for example, use a *manually defined articulated chain*. Going down along the tree, the authors start by estimating a palm position, then finger roots, then fingertips. In each step, a set of predictions for a new group of joints is conditioned on the output of the previous step (illustrated in Figure 16, bottom row).

An alternative approach was taken by Tang et al. [289], who instead of *manually* constructing the graph representing a hand (based on its anatomical skeleton), *learn* a tree-structured graphical model of a hand directly from the data. To estimate a pose, instead of targeting positions of hand joints immediately, the algorithm iteratively learns to proceed by *splitting* the image into *semantically meaningful groups* of hand parts, which get smaller and smaller with each iteration. Training objectives

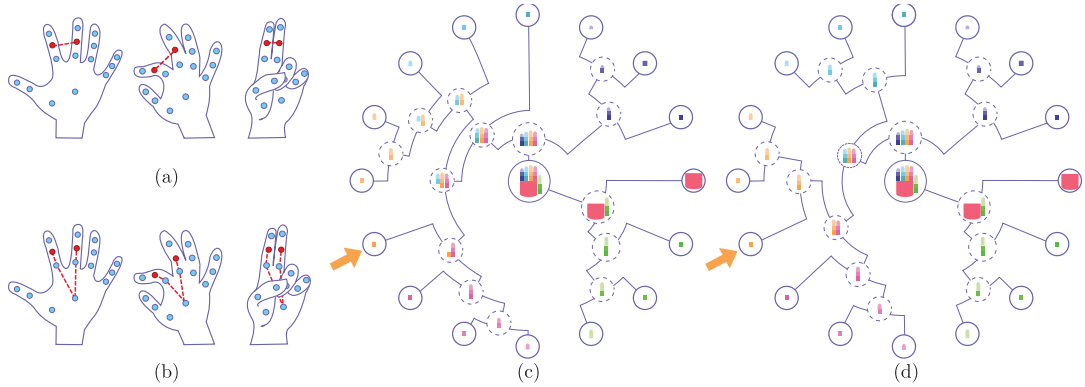


Figure 17 – Structure of the learned Latent Hand Model [289]. (a) Euclidean distance, (b) geodesic distance, (c) the latent model learned by using the Euclidean distance, (d) the same model learned using the geodesic distance.

for each step, which describe how the image should be split, are first automatically defined based on the training data, given its ground truth statistics.

In addition to *split* and *leaf* nodes, the proposed modification of a RDF, called **latent regression forest (LRF)** and consisting of **latent regression trees (LRT)**, has a set of *division* nodes. Split nodes, as before, evaluate each data sample to route it left or right. Division nodes perform splitting of the *training objective* into two and propagate the data along both paths in parallel.

Originally, the idea by Tang et al. [289] to infer structure from data takes its inspiration from a subset of probabilistic graphical models, called **latent tree models (LTMs)** [91] and their recent application to the problem of body pose estimation by Wang et al. [307].

As the reader may infer from its name, *latent tree models* is a subclass of *latent variable models*, relating a set of *observed variables* to a set of *latent variables*, which has a *tree structure*. In such latent tree models, all leaf nodes are considered to be *observed*, while all internal nodes are *latent* variables. Accordingly, with respect to the problem of hand pose estimation, 3D coordinates of hand joints are *observed vertices*, and the hierarchical topology of the hand, represented by *latent vertices*, are learned (see Figure 17c for an illustration).

In the discussed work by Tang et al. [289], the latent model of a hand is represented as a *binary tree*, which makes it straightforward to integrate this model into a random forest, in order to obtain the *latent regression forest (LRF)* mentioned earlier.

The theoretical basis of latent tree models and fundamental principles for learning LTMs are provided in a recent study by Mourad et al. [244]. Generally, there exist three groups of methods for inferring LTM structure from the data, namely search-based methods, variable clustering and distance-based algorithms. The Chow-Liu Neighbor-Joining (CLNJ, [55]) method, used by Tang et al. [289], belongs to the last group and performs grouping of joints consistently appearing close to each other in the data.

The employed CLN algorithm takes as an input a set of pairwise distances D between all observed variables (i.e. 3D joint coordinates), which are estimated from the training data:

$$D(v_1, v_2) = \frac{\sum_{I \in S} \delta^{(I)}(v_1, v_2)}{|S|}, \tag{37}$$

where δ is a metric (discussed below), v_1 and v_2 is a pair of joints and I is an image from the training set S.

The metric δ can be defined in an arbitrary way: in particular, Tang et el. [289] compare Euclidean and geodesic formulations and show that the geodesic distance, which takes into account depth discontinuities in the image, is more robust to variations in poses and results in a more semantically meaningful learned topological structure (see Figure 17 for an illustration).

The core process of neighbor joining [258, 98] starts with a star-shaped tree, where all observed vertices are connected, and iteratively selects a pair of joints $v_i - v_j$ (or, at later stages, groups of joints), which minimizes the following criteria:

$$f(i,j) = (|N| - 2)\delta(v_i, v_j) - \sum_{k \in N} \delta(v_i, v_k) - \sum_{k \in N} \delta(v_j, v_k), \tag{38}$$

where N is the set of vertices. Once the pair $v_i - v_j$ is combined in a single unit v_{ij}, the distances are reestimated as follows:

$$\delta(v_i, v_{ij}) = \frac{1}{2}\delta(v_i, v_j) + \frac{1}{2(|N| - 2)} \left(\sum_{k \in N} \delta(v_i, v_k) - \sum_{k \in N} \delta(v_j, v_k) \right), \tag{39}$$

$$\delta(v_{ij}, v_k) = \frac{1}{2}(\delta(v_i, v_k) - \delta(v_i, v_{ij})) + \frac{1}{2}(\delta(v_j, v_k) - \delta(v_j, v_{ij})). \tag{40}$$

Once the LTM structure is learned based on the chosen metric (see [289, 258] for the exact procedure), training of the LRF begins. The *split function* is defined in a similar manner to what we have seen before for pixel-wise processing:

$$h(\rho_i, \theta_n) = \left[I\left(\rho_i + \frac{\Delta\rho_{n,1}}{I(\rho_0)}\right) - I\left(\rho_i + \frac{\Delta\rho_{n,2}}{I(\rho_0)}\right) \geqslant \tau_n \right], \tag{41}$$

but this time this expression is formulated for a given vertex i with 3D coordinates ρ_i, rather than for each pixel in the image. As before, $\rho_{n,1}$ and $\rho_{n,2}$ are random offsets, and normalization is performed with respect to a image depth value at the root node position $I(\rho_0)$. Quality function Q for a given split is formulated as information gain:

$$Q(S) = H(S) - \sum_{d \in \{L,R\}} \frac{S^d}{|S|} H(S^d). \tag{42}$$

In this expression, H(S) minimizes the trace of the covariance matrix of the set of offset vectors between the current center and positions of left and right subregions, calculated over all images from set S:

$$H(S) = \sum_{m \in \{L(i), R(i)\}} \mathrm{tr}\left(\mathrm{Cov}\left\{ \rho_m^{(I)} - \rho_i^{(I)} | I \in S \right\} \right), \tag{43}$$

where tr is a trace operator.

As the reader may remember, in the previous work by Tang et al. [288], a number of splits which are necessary to achieve sufficient performance at a given training stage (in that case, shape classification, patch classification and regression on hand joint positions) was defined automatically from the current performance and the switching between them was therefore performed in adaptive manner. A similar strategy is applied to the latent regression trees, where the number of levels in the latent regression tree assigned to each level in the latent tree model is defined by comparing the information gain achieved so far with a certain threshold. When a desired level of performance on a given stage is achieved, a division node splits the training objective in next two.

In spite of their great performance, latent regression trees proposed by Tang et al. [289] do not take into account differences in *viewpoints*. However, as we have seen in the case of *shape classification forests* [152], introducing this additional degree of freedom in the model drastically increases its modeling capacity.

In these lines, inspired by the work by Tang et al. [289], Li et al. [225] proposed a similar framework with the difference being that instead of latent variables from the fixed LTM, their hierarchy is built on a number of flexible landmarks called **segmentation index points (SIPs)**. As in the previous case, positions of hand joints are estimated only at the leaf nodes of an RDF.

In short, as in [289], SIPs represent centroids of groups of several hand joints and are used for tree growing. However, while in [290] those intermediate targets were predefined and aligned with a *fixed* LTM, SIPs are *flexible* and are also dependent on a hand pose.

Proposed SIPs are obtained by clustering ground truth joint positions *in parallel* with growing a tree (based on the same geodesic distance, as in the previous case). This way, only subsets of training data reaching the particular node contribute into SIPs definition at this node, which makes these landmarks dependent on the view point and on the type of hand poses specific for the given subset. This idea is illustrated in Figure 19 (also see Figure 18 for comparison with [289]).

The distortion for hand joints clustering is formulated as follows:

$$J_{\text{clust.}} = \min_{Q} \sum_{v_1 \in N} \sum_{v_2 \in N} \sum_{q \in Q} r_{v_1 v_2 q} D(v_1, v_2), \tag{44}$$

where Q is a set of possible splits over which $J_{\text{clust.}}$ is minimized, $v \in N$ are skeletal hand joints, $r_{v_1 v_2 q} \in \{0, 1\}$ performs hard clustering as defined by q and $D(v_1, v_2)$ is defined by equation (37). Unlike in LTR, however, information about chosen splits q should be stored at each node to be used during test time.

2.3.5 *Refinement of joint positions: inverse kinematics*

Purely *discriminative* data-driven approaches, which were discussed in this section, do not explicitly take into account a *physical* model of the hand (except for some information on its skeletal structure in hierarchical methods) and are therefore likely to produce anatomically implausible configurations with deformed structures and self-collisions, especially when some of hand joints are occluded. In order to address this issue, at least partially, a number of works [288, 296] introduce in their frameworks an additional *kinematic refinement step* to tune and clean up their predictions.

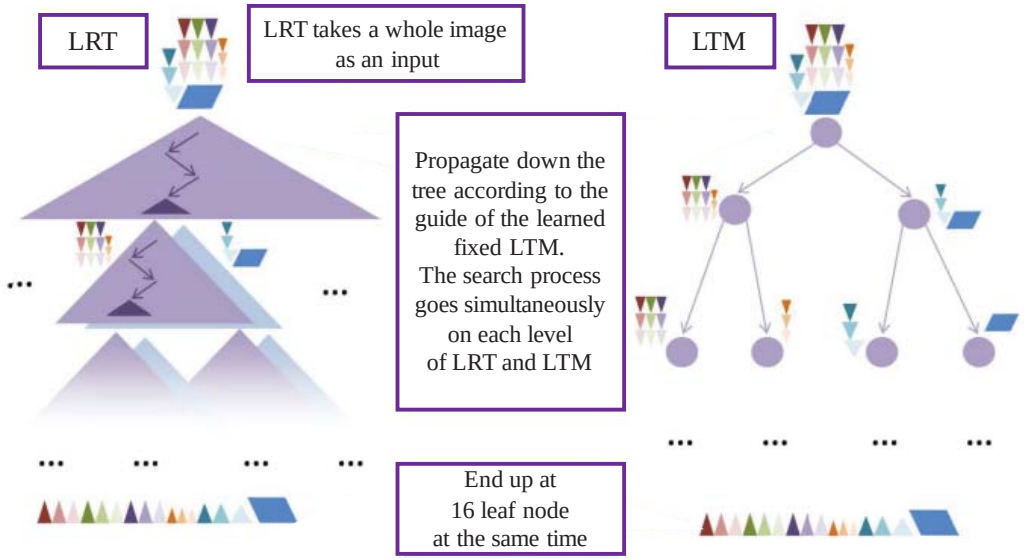


Figure 18 – Growing a Latent Regression Tree (LRT) given a Latent Tree Model (LTM) of a hand, proposed by Tang et al. [289]. The LTM is first learned from the training set and then used for training of the LRT (while being kept fixed).

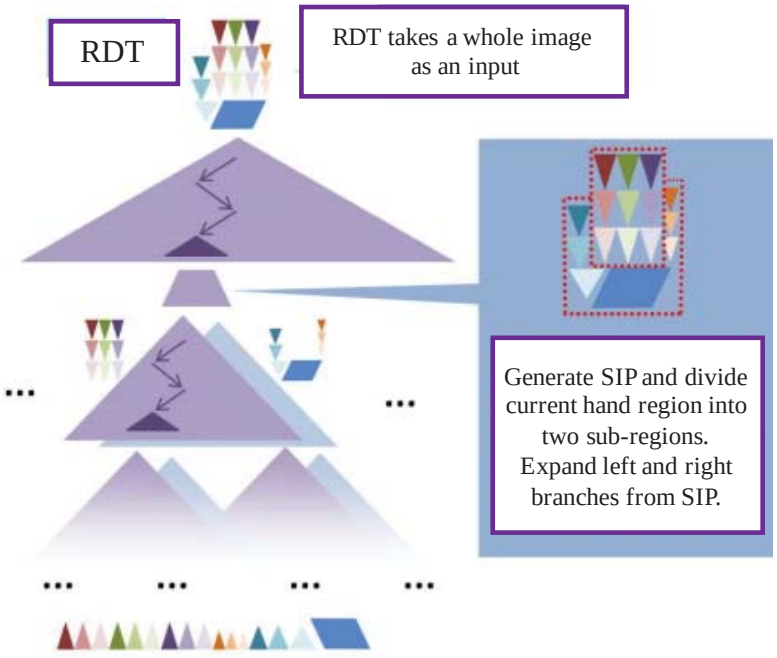


Figure 19 – Growing a Random Decision Tree (RDT) with Segmentation Index Points (SIPs), proposed by Li et al. [225]. Unlike in the previous case of LTM [289], where the model was learned in advance and kept *fixed*, here the training objectives (SIPs) are *adaptively* estimated for each split.

Algorithm 4 Pose refinement by [288]

Input: vote distribution for each joint
for each joint $j \in \mathbf{J}$ **do**
    Learn a 2-component GMM of the voting vectors
    if $\|\mu_j^1 - \mu_j^2\| < \tau_g$ **then**
        The j-th joint is a high-confidence joint
        Compute the j-th location as a mean of the stronger component
    else
        The j-th joint is a low-confidence joint
Find the Gaussian in $\mathcal{G}$ by finding the nearest neighbor of the high-confidence joints in $\mathcal{G}$
Update the remaining low-confidence joint locations from this Guassian
return Output pose

For this purpose, Tang et al. [288] created an additional large dataset of hand configurations (sampled using a 3D hand model and specified anatomical constraints) and then computed a data-driven kinematic model $\mathcal{G}$ containing view-specific distributions of joint locations in a form of Gaussian Mixture Models (GMMs) represented by a set of means and variances.

An advantage of their local regression method [288] (discussed previously), in which *each pixel* votes for positions of all neighboring joints, is that from the structure of obtained *vote distribution* (output of Algorithm 3) one may reason about the *confidence* of the resulting prediction.

This is done by fitting into each j-th vote distribution a 2-component GMM with parameters $\{\mu_j^1, \Sigma_j^1, \rho_j^1, \mu_j^2, \Sigma_j^2, \rho_j^2 | j \in \mathbf{J}\}$ where μ are corresponding means, Σ are variances and ρ are weights of each component. Furthermore, if the Euclidean distance between two Gaussians $\|\mu_j^1 - \mu_j^2\|$ for a given joint j does not exceed a certain threshold τ_g, the prediction is considered as confident (since the output distribution has a single mode). In this case, the mean of the stronger component with the weight $\rho_j = \max_j(\rho_j^1, \rho_j^2)$ is taken as a pre-refinement estimate.

Using these results, the refinement procedure is based on first alignment of the most confident predictions (reference points) with the learned model $\mathcal{G}$, in order to find the closest Gaussian corresponding to the most similar pose seen from a matching viewpoint. Once the closest Gaussian in $\mathcal{G}$ in found, the rest of joint positions are set to means of its corresponding components (see Algorithm 4).

In spite of the fact, that such data driven refinement apparently results in statistically more accurate hand pose estimates, it still does not fully exploit all information contained in a hand model. And this brings us closer to the idea of hybrid models, which include both discriminative predictions and model based refinements – these models will be the subject of the next section.

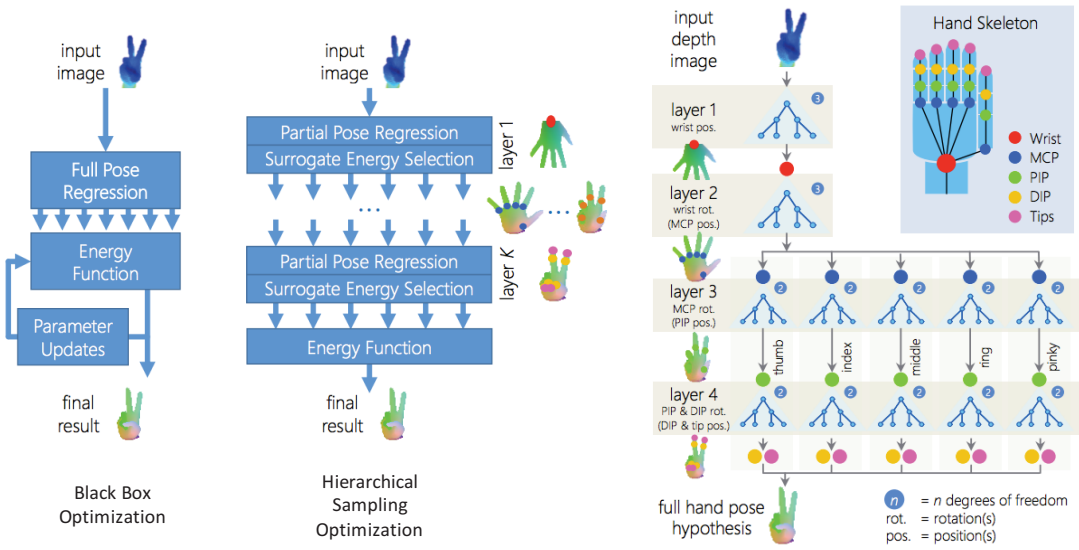


Figure 20 – Comparison of hybrid methods: black box optimization by Sharp et al. [264] (on the right) and hierarchical sampling optimization by Tang et al. [290] (on the left). Figures are reproduced from [290].

2.3.6 *Hybrid discriminative-generative models, or putting it all together*

All pose estimation methods, which we have discussed so far, are built on either data-driven learning-based hand pose estimation from a single frame (discriminative approaches) with eventual data-driven refinement, or model-based optimization by search given a provided initial guess (generative approaches) typical for tracking.

Naturally, both strategies have their advantages and downsides. Discriminative methods, even those that exploit hierarchies of joints, do not explicitly incorporate all available information about the hand (such as self-collisions, for example), neither do they model temporal dependencies between frames. This is addressed in model-based tracking approaches, but their efficiency is highly dependent on the quality of the initial guess, which also means that they tend to accumulate errors over time.

A logical conclusion based on these observations would be to create a *hybrid* approach that would take the best of both worlds. Indeed, there exist a number of recent works [264, 290, 237] which are based on a combination of discriminative and generative approaches, where the model-based search is initialized (or corrected) by a distribution of joint locations predicted by a discriminative model.

Sharp et al. [264], for example, significantly increased robustness of hand pose estimation by augmenting their tracking-based framework with a discriminative 2-step **tracking reinitializer** which learns to predict, first, global rotation, and, second, global offsets and pose type, conditioned on the first-stage prediction. After initialization, the real depth image is compared with the rendered depth map using the truncated L1 pixel-wise distance from equation (6) dubbed the *golden energy*.

Their approach, however, is based on two purely holistic and independent stages, where the rendering pipeline is treated as a *black box* (as in most classical generative frameworks considered so far). One of the main criticisms here would be that com-

puting the golden energy over the whole image for multiple candidates θ to find a good match is obviously computationally expensive.

In this sense, Tang et al. [290] went farther in this attempt to introduce kinematic hierarchy in the rendering-based search process and proposed a hybrid hand pose estimation method called **hierarchical sampling optimization (HSO)**. By focusing on generating a small number of highly accurate pose hypotheses, they were able to replace the iterative PSO-based search (as in [264] and earlier works) with simple ranking and choosing the best candidate as the final output. As a result, they achieve similar to [264] performance while requiring orders of magnitude fewer full image evaluations. A graphical comparison of the two strategies, holistic and hierarchical, is shown in the left part of Figure 20.

The hybrid method by Tang et al. [290] is based on a simplified (in comparison to their previous work [289]) 4-step hierarchical generation of proposals for positions of groups of joints called *partial poses*. This procedure starts with estimation of a wrist position, and then, sequentially, estimates locations of finger roots, intermediate finger joints and fingertips (see Figure 20 for a graphical illustration).

At each layer l, a set of hypotheses $\Theta^{(l)}$ is generated for the given subset of joints, and each hypothesis is combined with a set of its ancestors $\Theta^{(l-1)}$. Then the best candidate is selected by a cost function (dubbed *silver energy*), which is computed for the given subset of joints as a combination of two terms given by equations (9) and (13). The best hypothesis θ^* is then used as a prior for sampling proposals for a new group of joints at the next level $l+1$. Finally, the output pose is generated by concatenating the best configurations for the partial poses obtained at each level.

Each time, the initial sampling of hypotheses for a given group of joints is performed by a discriminatively trained decision forest producing a GMM over the pose parameters at each level. Their implementation uses standard offset-based depth features given by equation (20). We address an interested reader to consult the original work by [290] for more detail.

There is also a number of *deep learning* approaches (discussed in the next chapter), which fall in the same category of hybrid models [296, 216]. And if implementation of *discriminative* frameworks with deep learning methods can be fairly straightforward, their *generative* counterparts, which do not involve a traditional 3D rendering-based pipeline, seem particularly interesting.

2.3.7 *Hand-object interaction*

Analyzing hands in their interaction with objects, as well as reasoning about objects by how they are being manipulated, is a separate and quite exciting branch or research. It finds its direct applications in robotics (mainly for robot learning from visual demonstration [8]) and, furthermore, this idea became particularly interesting with development of wearable cameras and raising demand for automatic analysis of egocentric video streams *in the wild*.

This task is also particularly interesting from the modeling perspective. On one side, massive occlusions of hand parts during their interactions with objects make hand pose reconstruction even more nontrivial. Moreover, anatomical restrictions, which are a common place of isolated hand motion analysis systems, should be

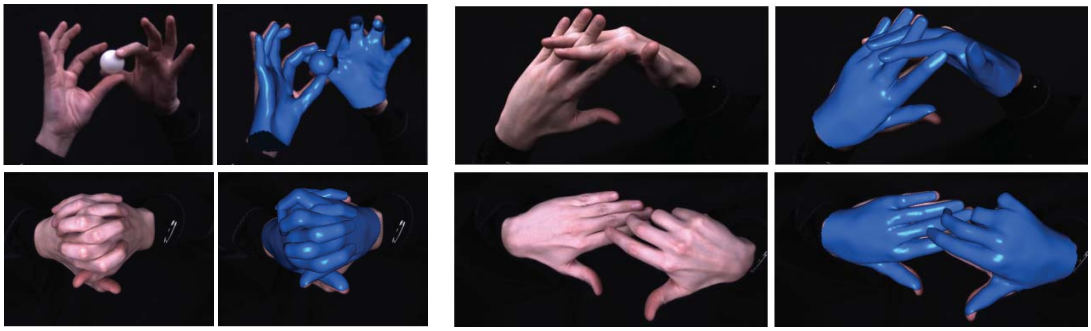


Figure 21 – Output of a multi-camera motion capture system by Ballan et al. [19].

reconsidered here, since some additional *unnatural* stretches often appear. On the other side, interactions with objects introduce additional physical constraints, which can serve as a source of additional information for pose estimation.

The problem of **recovering the pose** of a hand occluded by an object was addressed, for example, in a work by Hamer et al. [118]. In their framework, an individual tracker is associated with each joint of a hand, and a Markov Random Field (MRF) expresses its anatomical constraints in terms of proximity and angles.

Ballan et al [19] worked on a slightly different problem, namely **motion capture** of interacting hands (with each other and with objects) in a multi camera setup (see Figure 21). Their generative method is based on iterative rendering of a synthetic 3D hand model and comparing the obtained results with the observed input using a metric based on 3 kinds of features: edges, optical flow and *salient points*. For salient point extraction, they train an additional *nail detector* based on a Hough forest classifier [95] on a dataset of 400 images with labeled nails seen from different distances and view points. Their method significantly outperforms an alternative solution proposed by Oikonomidis et al. [219] (which is based on depth maps and is similar to other works from the same research group [217, 218] which we discussed in Section 2.3.3) and relies exclusively on edge and skin color maps.

In the context of **human-computer interaction** (and, more precisely, augmented and virtual reality environments), a recent work by Jang et al. [138] is dedicated specifically to ego-centric videos and the problem of clicking action and finger position estimation under self-occlusions.

Finally, a significant number of recent works, especially in the robotics community, focus on the problem of **human grasp analysis** [40, 250, 132], where the objective is formulated completely differently: instead of accurate *hand pose estimation*, the task is to perform *fine grained classification* of hand grasps from egocentric videos, to reconstruct positions of hand-object *contact points* and to estimate *direction of forces*. We must note here, that this problem is not necessarily easier than direct pose estimation by regression on hand joint locations and, more than that, accurate pose reconstruction per se provides no guaranties for quality of grasp classification (which is illustrated in Figure 22).

For this task, Rogez et al. [250] proposed a large dataset of egocentric videos containing 18 kinds of hand grasps with corresponding force and contact points annotations. Their own method is then based on *feature learning* of data representations from a whole video frame, a pre-segmented hand mask and a close-up view

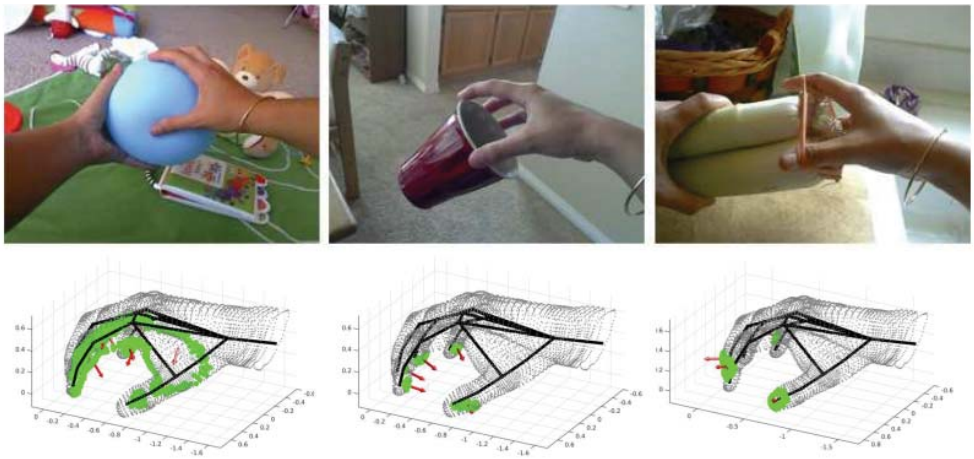


Figure 22 – Challenges of human grasp analysis: three images with almost identical hand poses correspond to three kinds of grasps () wide-object grasp, precision grasp and finger extension) with completely different distributions of forces (red) and positions of contact points (green) [250].

of a hand. Finally, the extracted features are fed to an SVM for grasp classification, the nearest neighbor from the training cluster of the corresponding class is returned, and its ground truth annotations of forces and contact points are projected on the test sample and serve as a final output of the system.

This concludes our overview of the prior art on human (and especially *hand*) motion analysis with *non-deep learning methods*.

As a final remark, we would like to note that although the hand-object interaction problem, that we have just discussed in this last section, is not directly related to the contributions of this thesis (neither application-wise nor, strictly speaking, methodologically), we still find it essential to keep in mind a broader picture of how similar problems are approached by different research communities (such as robotics and multimedia), which methods and practices are employed and how priorities are defined depending on the context of a particular application.

"O Deep Thought computer," he said, "the task we have designed you to perform is this. We want you to tell us..." he paused, "the Answer!"

— Douglas Adams,
"The Ultimate Hitchhiker's Guide to the Galaxy"

3

BACKGROUND: DEEP LEARNING

Deep learning has been known under a variety of names and has seen a long history or research, experiencing waves of excitement alternated with periods of oblivion. However, it was not until recently that technological progress in hardware engineering along with a number of algorithmic breakthroughs made it possible for computers to *learn concepts with minimal feature engineering*, end-to-end and in a purely data driven fashion, which significantly raised hopes and expectations of those on the quest of solving artificial intelligence.

Early works on deep learning, or rather on *cybernetics*, as it used to be called back then, are dated 1940s-1960s and already describe simple neural networks trained in supervised and unsupervised fashion [252, 122]. The second wave came under the name of *connectionism* in the 1980s-1990s [254] with the invention of backpropagation. Finally, the modern era of *deep learning* per se, as seen by certain authors [109], has started in 2006 [126, 26, 245], and since around 2011-2012 its ideas have been actively conquering speech processing, NLP and computer vision communities. Due to this highly non-linear history of development, these days, a fair amount of connectionist works and methods from previous periods are being naturally rediscovered, reformulated and reconsidered in new contexts. Those readers, who are interested to know more about the history of the matter, are encouraged to consult an exhaustive survey of the field by J. Schmidhuber [259].

During the last three years, the ideas of deep learning have made a tremendous impact in computer vision, demonstrating previously unattainable performance on the tasks of object detection, localization [104, 262], recognition [162] and image segmentation [90, 64]. Convolutional neural networks (ConvNets) [171] have excelled on several scientific competitions such as ILSVRC [162, 285, 121], Emotion Recognition in the Wild (EmotiW 2013) [145], Kaggle Dogs vs. Cats [262] and Galaxy Zoo. Taigman et al. [286] recently claimed to have reached human-level performance using ConvNets for face recognition. On a more sober note, deep learning in its modern formulation remains a relatively young field with tendencies to grow uncontrollably fast and in ad-hoc manner. A vast amount of models, concepts and algorithms which are already used today are yet to be completely understood.

In this chapter, we aim to review the most relevant classes of deep learning models to build a ground for our work. Starting with discussing state-of-the-art **feed-forward architectures** and **temporal neural models** (the ones which are commonly used in a supervised setting), we then briefly mention the milestones of **unsuper-**

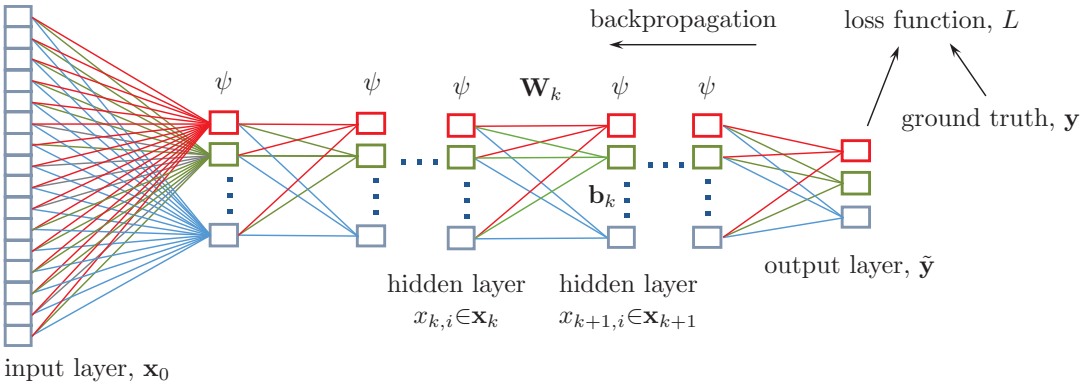


Figure 23 – The architecture of a fully connected deep feedforward network, where ψ is an activation function, $\mathbf{W}_k$ and $\mathbf{b}_k$ are a weight matrix and a vector of biases of a given hidden layer, respectively.

vised learning [1] and deep learning of **multimodal representations**. Finally, we will look at how deep learning methodology is being applied to the problem of human motion analysis. Many of those methods, that will be discussed, were proposed and extended by the community during the last three years, when some parts of this thesis were already completed. Nevertheless, we will attempt to provide a whole picture, as of the beginning of 2016, for completeness.

3.1 DEEP FEEDFORWARD MODELS

Fully connected **deep feedforward neural networks**, also called **multi-layer perceptrons (MLPs)**, are *the quintessential deep learning models* [109]. Their general purpose is to approximate an arbitrary surjective mapping $\mathcal{R}: \mathbf{x}_0 \mapsto \mathbf{y}$ (where $\mathbf{x}_0$ is an input and $\mathbf{y}$ is an output), which is typically done in several steps of computations given a set of learned model parameters θ.

We assume that the reader is familiar with fundamental principles of artificial neural systems. The basic equations will still be provided for the sake of introducing notations and terminology, which will be used throughout the whole manuscript, but we will make every effort to keep this discussion brief.

Deep learning models belonging to this class are conventionally called **networks** as they are assembled from a number of connected **layers** built from **hidden units**, where each layer performs one step of computation.

A **fully connected** network is wired in such a way that each unit in one layer receives information from all units from the previous layer. The term **neural** comes from an analogy with biological systems. A basic architecture of a fully connected feedforward network is shown in Figure 23.

In theory, even the simplest network with a single hidden layer and proper activation has the capacity to approximate an arbitrary smooth function, up to an arbitrary

1. We will not go into details of existing models for unsupervised learning, since they are not directly related to the work presented in this thesis.

precision. This was formulated and proved in a famous *universal approximation theorem* (quoted from [109] with adapted numbers of references):

> *...the universal approximation theorem [131, 65] states that a feedforward network with a linear output layer and at least one hidden layer with any "squashing" activation function (such as the logistic sigmoid activation function) can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units. [109]*

For practical purposes, it is enough to say that any continuous function on a closed and bounded subset of $\mathbb{R}^n$ satisfies the requirement of Borel measurability.

The ability of a feedforward network to approximate any smooth function does not provide guarantees that this approximation *can be learned*. Among the main reasons for that are specific properties of currently used optimization algorithms, as well as *overfitting*.

Furthermore, the theorem does not specify what amount of hidden units is *sufficient for such an approximation*, and in the worst case scenario, this number may be exponential, in order to memorize every possible input. For this reason, it is often preferable to choose *deeper architectures*, as many families of functions can be approximated in a much more compact way when the depth of the network is greater than some value d.

In addition, it has been noted in [109], that a deep model may be preferred for statistical reasons, as it only encodes *a very general prior* that the target function can be represented as a composition of simpler transforms. This intuition was *empirically* confirmed by a great number of works, showing that deeper models are able to reach a higher *degree of generalization* on a large variety of tasks [162, 285, 90].

Finally, let us introduce notations and formalize the basic mechanics of fully connected feedforward networks.

Starting from the input, at each next layer $k+1$ and for each input stimulus, the values taken by hidden units $\mathbf{x}_{k+1}$, called **activations**, are recursively calculated by taking a learnable linear affine transformation from activations of the previous layer $\mathbf{x}_k$, followed by a nonlinear function, which is called **activation function** and is applied elementwise:

$$x_{k+1,j} = \psi\left(\sum_{i=1}^{N} w_{kij} x_{k,i} + b_{kj}\right), \tag{45}$$

$$\text{or, in the vector form, } \mathbf{x}_{k+1} = \psi\left(\mathbf{W}_k \mathbf{x}_k + \mathbf{b}_k\right), \tag{46}$$

where $x_{k,i} \in \mathbf{x}_k$, $i=1...N$ are elements of the vector of previous activations (or inputs, for the first hidden layer in the network), $x_{k+1,j} \in \mathbf{x}_{k+1}$, $j=1...M$ are activations of the given layer, $\mathbf{W}_k$ is a **weight matrix** defining weight coefficients of each input-output connection w_{kij}, and $\mathbf{b}_k$ is a vector of **biases** consisting of elements b_{kj} (weight matrices and bias vectors are defined for each layer independently).

Finally, ψ is a non-linear activation function (where among the most popular options are sigmoid, hyperbolic tangent and, in more recent works, rectified linear unit, dubbed ReLU).

In the classification task, the number of *output units* $\tilde{\mathbf{y}}$ corresponds to the number of classes in the vocabulary. Normalized **output activations**, calculated by a softmax transformation from activations of the previous layer $\mathbf{x}$ and matched with an appropriate loss (see below), have a meaning of *posterior probabilities* of a given class n given observation $\mathbf{x}_0$:

$$\tilde{y}_n = P(c_n|\mathbf{x}_0) = \frac{\exp(\mathbf{W}_n^T\mathbf{x})}{\sum_j \exp(\mathbf{W}_j^T\mathbf{x})} \tag{47}$$

where $\mathbf{x}$ are activations of the last hidden layer, $\mathbf{W}$ is a weight matrix (this time, of the output layer). In this particular case, $\mathbf{W}_n$ denotes a set of weights connecting the previous layer with an output element n. Biases $\mathbf{b}$ are omitted for compactness.

Training in deep networks is typically done by *gradient descent* using the *chain rule* and **back-propagation** of the error from the output to the input in order to minimize a predefined **loss function** L, which is typically non-convex in the parameter space. In the context of supervised learning, classification cost function is conventionally formulated as the cross-entropy between the ground truth labels and the model predictions and is calculated by taking negative log-likelihood of the softmax activations:

$$L_C(\theta, \mathcal{D}) = -\sum_{i=1}^{|\mathcal{D}|} \log P(Y = y^{(i)}|\mathbf{x}_o^{(i)}, \theta), \tag{48}$$

where $\mathcal{D}$ is a set of training samples, $y^{(i)}$ is a ground truth label of sample i, $P()$ are posterior probabilities calculated from equation (47) and θ is a set of all network parameters, including, in the simplest case, weights and biases of all layers.

For regression, as in other machine learning algorithms, the mean squared error remains a classical solution:

$$L_R(\mathcal{D}) = \sum_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{N} \|y_j^{(i)} - \tilde{y}_j^{(i)}\|_2^2, \tag{49}$$

where N is the number of outputs, $y_j \in \mathbf{y}^{(i)}$ are elements of a ground truth vector and $\tilde{y}_j \in \tilde{\mathbf{y}}^{(i)}$ are model predictions (not normalized) for sample i.

3.1.1 *Convolutional neural networks*

According to the definition given by [109], **convolutional neural networks** (also dubbed CNNs, ConvNets) are *simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers*. This class of models is particularly suitable for processing signals of a spatial or temporal nature (i.e. images, video or audio, rather than ordered feature sets), where patterns in the data are treated under the assumption of spatial or temporal invariance.

An illustration of such a convolutional architecture for processing a 2D input is shown in Figure 24. The immediate advantage of this class of models is that the number of their parameters is significantly reduced in comparison with fully connected networks due to parameter sharing, since the input is processed locally by sliding a set of convolutional filters over it. This form of parameter sharing makes the convolutional network *equivariant* to translation, meaning that the output activations of a

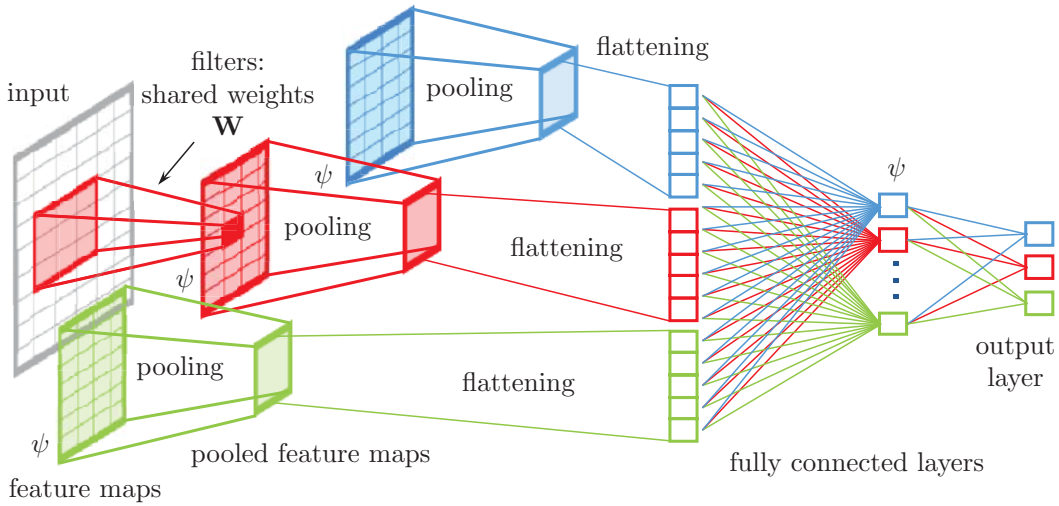


Figure 24 – An example of a convolutional neural network taking as input a single channel 2D image. In general case, an arbitrary number of convolutional layers (or convolutional-pooling pairs) can be stacked on top of each other, combined with an arbitrary number of fully connected layers closer to the output of the network.

convolutional layer change in the same way as its input [109]. The dimensionality of convolutions can be extended to an arbitrary number of input dimensions and applied, for example, to 3D ultra-sound images or 3D spatio-temporal volumes, using the same spatial metaphor for time.

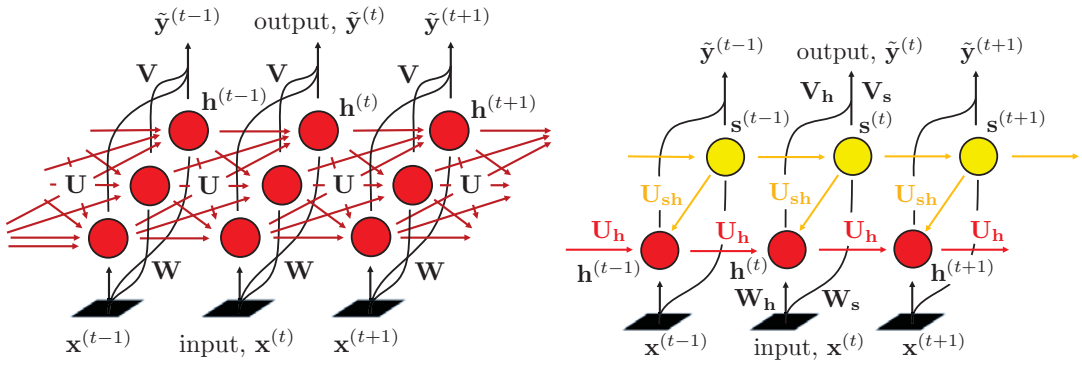
The result of the convolutional operation followed by an application of an element-wise non-linearity ψ is called a **feature map**. Each convolutional layer is typically followed by a **pooling** layer – an operation which outputs a maximum, for example (for max pooling), value within a local neighborhood, reducing the size of a feature map and introducing additional invariance to small local translations. In some recent architectures, the pooling step may be omitted.

The *AlexNet* [162] realization of a deep convolutional architecture for images, proposed in 2012, signified the first major success of deep learning in computer vision and was eight layers deep, where the three last layers were fully connected. A number of more advanced and complex convolutional networks, especially in the context of visual recognition and detection from images, have been proposed over last years, including *Network in network* [179], *Inception* modules [285], *VGG* networks [272], and *region-based CNNs* [105].

The most recent convolutional architectures, as of the beginning of 2016, which are based on *Highway networks* [278] and *residual learning* [121], are dozens to hundreds layers deep and are able to dynamically adapt to the complexity of the task.

3.2 DEEP TEMPORAL MODELS

The problem of *sequence modeling* with deep neural networks has been a subject of decades of scientific exploration, and it is probably safe to say that today this is one of the most actively researched topics in the community. Moreover, since the temporal aspect is often crucial for human motion analysis, neural temporal models



(a) (b)

Figure 25 – Deep temporal models, shown unrolled in time: (a) a basic recurrent network (RNN); (b) structurally constrained recurrent network (SCRNN) [191]. Red hidden units have activation function ψ and yellow hidden units have no activation.

will be one of the most important subjects of the main part of this thesis. For these reasons, in this section we will aim to review the most important points concerning existing architectures and their training in more detail.

3.2.1 *Recurrent neural networks*

A vanilla **recurrent neural network (RNN)** is a simple but general temporal architecture, where at each moment of time t the output is governed by *feedforward connections* from input $\mathbf{x}^{(t)}$ (as in the previous case), as well as *recurrent connections* conveying additional information about the history of the network's activations at preceding steps. In Elman's simple recurrent network (SRN) [84], for example, it is implemented by recursive conditioning of each state on the previous one.

Figure 25a shows a view of the simple one layer recurrent network unrolled over time. Such a vanilla RNN is governed by the following update equation:

$$\mathbf{h}^{(t)} = \psi(\mathbf{W}\mathbf{x}^{(t)} + \mathbf{U}\mathbf{h}^{(t-1)}), \tag{50}$$

where $\mathbf{x}^{(t)}$ is the input at time t, $\mathbf{h}^{(t)}$ denotes the network's hidden state at time t, $\mathbf{W}$ and $\mathbf{U}$ are feed-forward and recurrent weight matrices, respectively, and ψ is a non-linear activation function, typically tanh.

The output is produced as a non-linear transform of the hidden state:

$$\tilde{\mathbf{y}}^{(t)} = \phi(\mathbf{V}\mathbf{h}^{(t)}), \tag{51}$$

where $\mathbf{V}$ is a weight matrix and ϕ is an activation function. Here and in all equations in this section we omit bias vectors for the sake of compactness of notation, but in practice they are typically included as learnable parameters.

In deeper temporal architectures, several recurrent layers can be stacked on top of each other. In this case, the output activations serve as input to the next layer and, strictly speaking, should be denoted as $\mathbf{x}_{l+1}$ for a given input $\mathbf{x}_l$ (where l is a number of layer).

Theoretically, a RNN is a general and powerful model, which is potentially capable of extracting complex patterns, storing them and propagating them through time. However, in practice, training of such systems by backpropagation remains difficult due to well known problems of *exploding* and *vanishing gradients* [128, 25]. And though explosion can be addressed by simple clipping the gradient magnitudes [227], the latter issue is more difficult to solve.

The reason for gradients to vanish becomes apparent if we mentally unfold a recurrent network in time, in which case it becomes no different from very deep feed-forward networks. In the recurrent case, the gradient magnitude, propagated back through time, gets affected by the recursive multiplication by the matrix of recurrent weights. If the eigen values of this matrix are less than one, the gradients exponentially decay to almost zero in several steps. This process only gets accelerated if easily saturatable sigmoid-like activations are used. As a result, such networks can capture short-term history, up to 5-10 steps, but dependencies spanning over longer intervals are much harder to learn.

There were a number of works exploring possible solutions to this problem, including Hessian-Free optimization methods [186], replacing non-linearities with rectified linear units (ReLU) and initialization of recurrent weights with an identity matrix or its scaled version [170]. Strictly speaking, however, the problem for the given architecture remains unsolved.

More successful were approaches proposing modifications to the *architecture* and to the *structure of recurrent weights*, rather than to the training process.

In this spirit, Mikilov et al [191] augmented the vanilla RNN with an integrating module, consisting of *context units* $\mathbf{s}$, whose function was to propagate integrated statistics of the history of observations forward through time. Such architecture received the name of **Structurally Constrained RNN**, or **SCRNN**, and has been shown to be significantly more effective than the vanilla RNN on a number of challenging sequence modeling tasks.

The SCRNN connectivity is shown in Figure 25b (again, unrolled in time for the sake of explanation). In this diagram, the red elements correspond to the quasi-conventional RNN units $\mathbf{h}^{(t)}$ fully connected to their previous state $\mathbf{h}^{(t-1)}$ (with weight matrix $\mathbf{U_h}$) and to input $\mathbf{x}$ (with weight matrix $\mathbf{W_h}$). In addition, these elements are connected to a group of *context units*, and these connections are defined by the third weight matrix, $\mathbf{W_{sh}}$. As before, an element-wise non-linear activation function ψ is applied to the sum of these terms once all three linear transforms are calculated.

Similarly, the values taken by *context units* $\mathbf{s}^{(t)}$ (i.e. yellow elements in the figure) at each time t are computed as a linear combination of transformed input $\mathbf{x}$ (by weight matrix $\mathbf{W_s}$) and their previous state $\mathbf{s}^{(t-1)}$ multiplied by an identity matrix, with *no non-linear activation applied*. As a result, these additional recurrent hidden units explicitly store information about the previous state and, through recursion, provide conventional units with summarized context of the whole history of inputs, while not suffering from vanishing gradients as much.

It remains to note that receiving this context information, conventional units do not, in turn, influence the state of the context units.

The following modified set of update rules summarizes everything what has been said so far about the SCRNN:

$$\begin{aligned} \mathbf{s}^{(t)} &= (1-\alpha)\mathbf{W_s}\mathbf{x}^{(t)} + \alpha\mathbf{s}^{(t-1)}, \\ \mathbf{h}^{(t)} &= \psi\left(\mathbf{W_h}\mathbf{x}^{(t)} + \mathbf{U_h}\mathbf{h}^{(t-1)} + \mathbf{U_{sh}}\mathbf{s}^{(t)}\right), \\ \tilde{\mathbf{y}}^{(t)} &= \phi(\mathbf{V_h}\mathbf{h}^{(t)} + \mathbf{V_s}\mathbf{s}^{(t)}), \end{aligned} \tag{52}$$

where $\alpha \in (0,1)$ is a weight coefficient defining the contribution of the given state to the history, $\mathbf{U_{sh}}$ is a weight matrix of connections from the context units to the conventional units at time t, $\tilde{\mathbf{y}}^{(t)}$ is the ouput of the network at time t, $\mathbf{V_h}$ and $\mathbf{V_s}$ are weight matrices of the units connecting two blocks of the hidden layer (conventional and context, respectively) to the output and ϕ and ψ are activation functions (notations are shown in Figure 25b).

As it becomes fairly obvious from Figure 25, instead of positioning the context-augmented SCRNN module as a new architecture, it is equivalent to say that the SCRNN can be considered as a vanilla RNN with a block structured matrix of recurrent weights $\mathbf{U}$, feedforward input weights $\mathbf{W}$ and output weights $\mathbf{V}$ defined as follows:

$$\mathbf{U} = \begin{pmatrix} \mathbf{U_h} & \mathbf{U_{sh}} \\ \mathbf{0} & \mathbf{I_m} \end{pmatrix}, \quad \mathbf{W} = \begin{pmatrix} \mathbf{W_h} \\ \mathbf{W_s} \end{pmatrix}, \quad \mathbf{V} = \begin{pmatrix} \mathbf{V_h} \\ \mathbf{V_s} \end{pmatrix} \tag{53}$$

where I_m is an identity matrix, and m is the number of context units. Assuming that the layer has n conventional units, the size of $\mathbf{U_h}$ is $n \times n$, $\mathbf{U_{sh}}$ is of size $n \times m$ and $\mathbf{U}$ is a square matrix of size $n+m$.

More generally, the first line in the set of SCRNN equations 52 agrees well with the principle of **leaky integration** described by Bengio et al. [27], whose essential purpose is to expand the time scale of gradient vanishing. This can be done by applying the same principle, as in SCRNN, to a standard RNN, using the update rule from equation 50, but with significantly smaller values of $\boldsymbol{\alpha}$, sampled randomly and specifically for each unit:

$$\mathbf{h}^{(t)} = (1-\boldsymbol{\alpha})\mathbf{W_s}\mathbf{x}^{(t)} + \boldsymbol{\alpha}\mathbf{h}^{(t-1)}, \tag{54}$$

where, once again, $\boldsymbol{\alpha}$ is a vector, rather than a single value, as before.

An alternative strategy to providing the network with the historical information could be to *explicitly connect each state* t *to its predecessors from more distant past*.

El Hihi and Bengio [83], in their early work, have shown that a structural decomposition of a recurrent hidden layer in several groups, which correspond to different time scales, is beneficial for modeling *hierarchies of temporal dependencies*. In this way, a group of *slow* units, which operate at long time scale, preserve the context longer, while *fast* units (short time scale) are more responsive to local variations in fast changing inputs.

One may note that this idea can be seen as closely related to the principle of wavelet transforms [70] representing the signal at multiple resolutions. A similar concept based on discrete time delays and shifts was previously implemented in feedforward Time-Delay Neural Networks (TDNNs) [165].

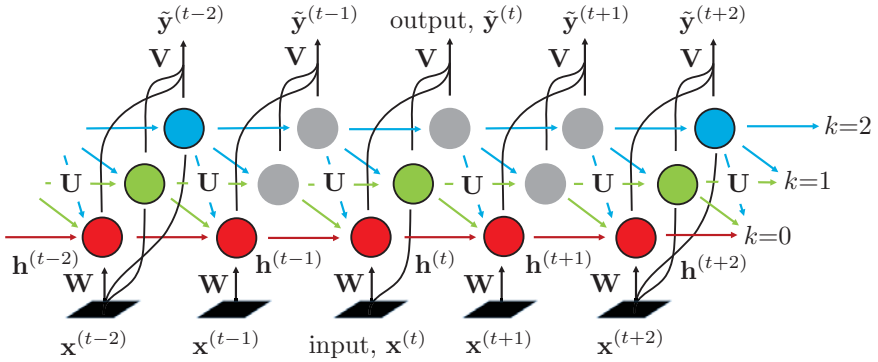


Figure 26 – An example of a Clockwork Recurrent Neural Network (CWRNN) [161] with 3 temporal bands, exponential low and a base of 2. Increasing k indicates lower operating frequency. Grey color indicates inactivity of a slow unit (during both training and test time) at a given iteration t, meaning that the value from the last time when it was active is propagated.

More recently, Koutnik et al. [161] proposed a particular implementation of this principle called the **Clockwork RNN (CWRNN)**, where the hidden layer is partitioned into several groups of scale-specific bands, including *fast* and *slow* units (an example of such an architecture is shown in Figure 26). As a result, the CWRNN operates at several temporal scales which are incorporated in a single network and trained jointly, while each band is updated at its own pace.

The size of the step from band to band typically increases exponentially (which we call *exponential update rule*) and is defined as n^k, where n is a base and k is the number of the band. In general, the set of step sizes can be defined arbitrarily, but due to advantages in implementation (see [161]), in the rest of this manuscript we will assume that all clockwork architectures have exponential bands, if not stated otherwise.

In the CWRNN, fast units (shown in red in Figure 26) are connected to all bands, benefiting from the context provided by the slow bands (green and blue), while the low frequency units ignore noisy high frequency oscillations.

Equation (50) from classical RNNs is modified, leading to a new update rule for the k-th band of output $\mathbf{h}$ at iteration t as follows:

$$\mathbf{h}_k^{(t)} = \begin{cases} \psi\left(\mathbf{W}(k)\mathbf{x}^{(t)} + \mathbf{U}(k)\mathbf{h}_k^{(t-1)}\right) & \text{if } (t \bmod n^k)=0, \\ \mathbf{h}_k^{(t-1)} & \text{otherwise.} \end{cases} \tag{55}$$

where $\mathbf{U}(k)$ and $\mathbf{W}(k)$ denote rows k from matrices $\mathbf{U}$ and $\mathbf{W}$. Matrix $\mathbf{U}$ has an upper triangular structure, which corresponds to the connectivity between frequency bands.

A part of our work is based on the CWRNN architecture, so we will revisit it in Chapter 6 and analyze its mechanics in more detail.

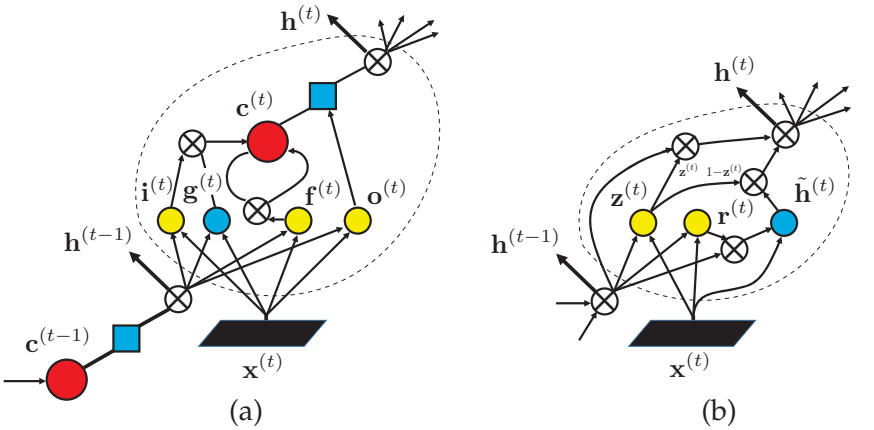


Figure 27 – Gated temporal models: a) LSTM unit [129], b) Gated Recurrent Unit (GRU) [54]. Yellow units have sigmoid activations, while the blue ones are activated by tanh.

3.2.2 *Gated recurrent models*

One of the most efficient ways to increase the efficiency of training in neural temporal models discovered so far is using a *gated activation function*, instead of a simple linear transform followed by an elementwise non-linearity.

In this context, **Long-Term Short Memory (LSTM)** networks [129, 110], and their recent convolutional extensions [80, 257] have proven to be, so far, one of the best performing models for learning long-term temporal dependencies. They handle information from the past through additional *gates*, which regulate how a *memory cell* is affected by the input signal.

In order to explain the idea of a LSTM *memory cell*, C. Olah [223] drew a vivid analogy with a conveyor belt, since a state of the memory cell is propagated through time with only minor interactions with the input.

The basic LSTM cell is composed of input $\mathbf{i}$, output $\mathbf{o}$, forget $\mathbf{f}$, and input modulation $\mathbf{g}$ gates. Finally, $\mathbf{c}$ denotes the state of memory cell per se (shown in Figure 27a). In this setting, the *input gate* allows the network to add new memory to the cell's state, the *forget gate* resets the memory and the *output gate* regulates how gates at the next step will be affected by the current cell's state.

As in the vanilla RNN and its extensions, each element of an LSTM is parameterized by corresponding feed-forward ($\mathbf{W}$) and recurrent ($\mathbf{U}$) weights (and bias vectors):

$$\begin{aligned}
\mathbf{i}^{(t)} &= \sigma(\mathbf{W}_i \mathbf{x}^{(t)} + \mathbf{U}_i \mathbf{h}^{(t-1)}), \\
\mathbf{f}^{(t)} &= \sigma(\mathbf{W}_f \mathbf{x}^{(t)} + \mathbf{U}_f \mathbf{h}^{(t-1)}), \\
\mathbf{o}^{(t)} &= \sigma(\mathbf{W}_o \mathbf{x}^{(t)} + \mathbf{U}_o \mathbf{h}^{(t-1)}), \\
\mathbf{g}^{(t)} &= \tanh(\mathbf{W}_g \mathbf{x}^{(t)} + \mathbf{U}_g \mathbf{h}^{(t-1)}), \\
\mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \mathbf{g}^{(t)}, \\
\mathbf{h}^{(t)} &= \mathbf{o}^{(t)} \odot \psi(\mathbf{c}^{(t)}),
\end{aligned} \tag{56}$$

where $\odot$ denotes the Hadamard product, $\mathbf{x}^{(t)}$ in an input at time t, $\mathbf{h}^{(t)}$ is the corresponding output activation, σ is a sigmoid and ψ is a tanh activation function. In some implementations, the activation on the input modulation gate is omitted.

LSTMs have demonstrated state-of-the-art performance on most standard sequence modeling tasks. In addition, it was noted by a number of authors [100, 143] that to further improve performance, it is often important to initialize the bias of the forget gate to 1 (or another positive value).

Despite its effectiveness, the LSTM architecture has been repeatedly criticized for being too ad-hoc, since the purpose and significance of each of its components is not immediately apparent. For this reason, the high complexity of LSTMs may turn out computationally wasteful, especially in real time applications when computational resources are limited.

Accordingly, there exist a body of work which aim to answer the question of whether or not this architecture is optimal, by performing an empirical analysis of contribution of each gate [112, 143, 56]. The conclusion of those experiments was summarized by Jozefowicz et al. [143] as follows.

> *We discovered that the input gate is important, that the output gate is unimportant, and that the forget gate is extremely significant on all problems except language modelling. This is consistent with Mikolov et al. [191], who showed that a standard RNN with a hard-coded integrator unit (similar to an LSTM without a forget gate) can match the LSTM on language modelling.*

In this spirit, there were attempts to simplify the LSTM architecture while optimizing its performance.

Recently, Cho et al. [54] proposed the **Gated Recurrent Unit (GRU)** which is claimed, by the authors, to be easier to compute and implement. GRU has only two gates, namely *reset gate* $\mathbf{r}$ (similar to the forget gate of LSTM) and *update gate* $\mathbf{z}$ (acting similar to the memory cell). The GRU update rule is formulated as follows:

$$\begin{aligned} \mathbf{r}^{(t)} &= \sigma(\mathbf{W}_r \mathbf{x}^{(t)} + \mathbf{U}_r \mathbf{h}^{(t-1)}), \\ \mathbf{z}^{(t)} &= \sigma(\mathbf{W}_z \mathbf{x}^{(t)} + \mathbf{U}_z \mathbf{h}^{(t-1)}), \\ \tilde{\mathbf{h}}^{(t)} &= \tanh\left(\mathbf{W}_h \mathbf{x}^{(t)} + \mathbf{U}_h (\mathbf{r}^{(t)} \odot \mathbf{h}^{(t-1)})\right), \\ \mathbf{h}^{(t)} &= \mathbf{z}^{(t)} \odot \mathbf{h}^{(t-1)} + (1 - \mathbf{z}^{(t)}) \odot \tilde{\mathbf{h}}^{(t)}, \end{aligned} \tag{57}$$

where $\tilde{\mathbf{h}}$ is a candidate for the update and, as before, $\mathbf{W}$ and $\mathbf{U}$ are the parameterization of input and recurrent connections, respectively. Biases are omitted for compactness of notation, but are typically present.

When being close to zero, the reset gate forces the network to forget its history. The update gate regulates the contribution of a previous state into its current state. A graphical representation of the architecture is shown in Figure 27b.

Finally, a recent work by Chung et al. [57], is dedicated to the problem of *multi-scaling* and *multi-resolution* in the context of *gated architectures*. Their proposed model received the name of **Gated Feedback RNN (GF-RNN)**.

Unlike the Clockwork networks [161], instead of introducing multiple scales *inside* one hidden layer by modifying its own recurrent wiring, the GF-RNN architecture exploits the idea of learning fine-to-coarse representations in deeper recurrent models obtained by *stacking several recurrent layers*. Their idea lies in drawing a direct analogy with feedforward convolutional networks, where first layers learn to become *atomic* feature extractors, while with increasing depth the level of abstraction and, in a sense, the *scale* of the learned representation changes.

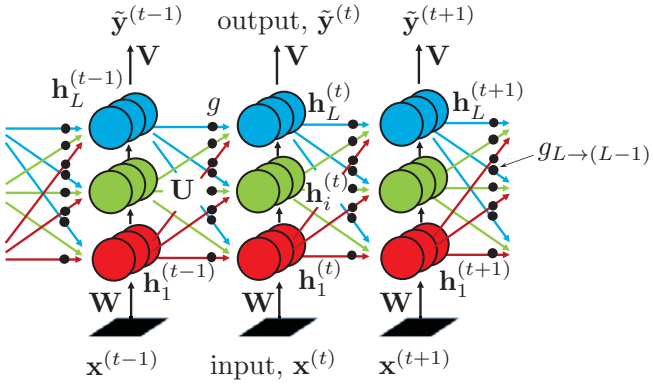


Figure 28 – An example of a Gated Feedback RNN [57] with three stacked recurrent layers.

Accordingly, to let different layers corresponding to different *temporal hierarchical scales* influence each other, Chung et al. introduce inter-layer top-down and bottom-up diagonal connections allowing high-level activations affect layers which are closer to the input and vice versa (see Figure 28).

Although the wiring between all layers in the hierarchy is initially fully connected, introducing scalar *global reset gates* g into each path allows the network to automatically adapt to the task in a purely data-driven fashion, and to exploit the scales optimal for the given application, while dropping the others. This global reset gate is updated as follows:

$$g_{i\to j} = \sigma\left(\mathbf{w}_{g,i\to j}\mathbf{h}_{j-1}^{(t)} + \mathbf{u}_{g,i\to j}\mathbf{h}^{*(t-1)}\right), \tag{58}$$

where i and j are layer numbers (and i→j is a directed connection between them), $\mathbf{w}_{g,i\to j}$ and $\mathbf{u}_{g,i\to j}$ are weight vectors corresponding to the input and the recurrent connections, respectively, and $\mathbf{h}^{*(t-1)}$ is a concatenation of all activations at the previous step.

This logic can be incorporated into existing gated architectures, such as LSTM and GRU, by associating a gate to each connection instead of each cell. To this end, we modify the update rule of the input modulation gate (LSTM) or the update gate (GRU). Equation 4 from the set 56 becomes:

$$\mathbf{g}_j^{(t)} = \tanh\left(\mathbf{W}_{g,j-1\to i}\mathbf{h}_{j-1}^{(t)} + \sum_{i=1}^{L} g_{i\to j}\mathbf{U}_{g,i\to j}\mathbf{h}_i^{(t-1)}\right), \tag{59}$$

where L is the number of recurrent layers in the hierarchy. Similarly for the GRU (line 3 from the set of equations 57):

$$\tilde{\mathbf{h}}_j^{(t)} = \tanh\left(\mathbf{W}_{h,j-1\to j}\mathbf{h}_{j-1}^{(t)} + \mathbf{r}_j^{(t)} \odot \sum_{i=1}^{L} g_{i\to j}\mathbf{U}_{h,i\to j}\mathbf{h}_i^{(t-1)}\right) \tag{60}$$

All existing deep learning temporal architectures can be naturally reformulated in a *bi-directional* way, which is especially practical, when the modeled sequences have a spatial nature.

3.2.3 *Temporal models with external memory*

In addition to conventional simple recurrent and gated recurrent models, a significant amount of work has been done in the direction of temporal models *exploiting large external memory* for explicit memorization of its inputs [69, 196, 142], which is particularly important in such contexts as question answering or task completion. Among the most interesting and promising recent reincarnations of this idea are **Memory networks** [315] (including their end-to-end implementation [280]), **Neural Turing Machines** [111] and **Neural GPUs** [146]. Since these concepts are still going through the initial stage of their development and are not directly related to our work, we will skip reviewing them in this manuscript.

3.3 REMARKS ON UNSUPERVISED LEARNING

Instead of *hand-crafting* data descriptors, or learning them *discriminatively* for specific tasks (which we have been discussing so far), efficient representations can be inferred directly from data, with no additional supervision, by, for example, minimizing *reconstruction error*, e.g. as in autoencoders, or some predefined *energy function*, e.g. as in Restricted Boltzmann Machines (RBMs).

Although there exist a quite large number of classes of such *unsupervised learning* models, they do not fall in the scope of this thesis, therefore we will not dive deeply in this field. However, we must note that today, the unsupervised learning appears to be one of the most intriguing problems, which occupies minds of many researchers working on deep learning. Accordingly, this is more than possible that these models will play more in more significant role in the nearest future.

In the context of human motion analysis, unsupervised models have been employed for learning spatial, temporal and spatio-temporal data representations.

Le et al. [169], for example, used *Independent Subspace Analysis (ISA)* for computationally efficient learning of hierarchies of invariant spatio-temporal features. In the same countext, Baccouche et al. [14] adapted the original work of Ranzato et al. [245] by adding a temporal dimension to 2D sparse convolutional *autoencoders*.

On the other side, Taylor et al. [291] extended and scaled up the *Gated RBM (GRBM)* model proposed by Memisevic and Hinton [188] for learning representations of image patch transformations. Chen et al. [46] used convolutional RBMs as basic building blocks to construct the *Space-Time Deep Belief Networks (ST-DBN)* producing high-level representations of video sequences.

Finally, very recently Srivastava et al. [277] found a way to exploit *LSTM networks as autoencoders* for unsupervised feature learning from video sequences and demonstrated their positive contribution on supervised tasks.

3.4 REMARKS ON MULTIMODAL DEEP LEARNING

Before we start our review of *applications* of deep learning in human motion analysis, there is the last important theoretical aspect which we need to discuss and which is particularly related to the work presented in this manuscript. Here, we are talking about *learning in multimodal systems*.

Although most of existing models can be directly adapted to operate on a *set of input signals*, rather than on a single one, there are a number of modeling aspects, specific to this context, which are worth additional exploration.

Fusion dynamics, for example, has seen a long history of research. While in most practical applications *late* fusion of scores output by several models offers a cheap and surprisingly effective solution [145], both *late* and *early fusion* of either *final* or *intermediate* data representations remain under active investigation.

A significant amount of work on early combination of diverse feature types has been done in the field of object and action recognition. Multiple Kernel Learning (MKL) [15] has been actively discussed in this context. At the same time, as shown by [99], simple additive or multiplicative averaging of kernels may reach the same performance while being orders of magnitude faster.

In [332] the authors propose a late fusion strategy compensating for errors of individual classifiers by minimizing the rank of a score matrix, and in a follow up work [180] identify sample-specific optimal fusion weights by enforcing similarity in fusion scores for visually similar labeled and unlabeled samples. Xu et al. introduced the Feature Weighting via Optimal Thresholding (FWOT) algorithm [324] jointly optimizing feature weights and thresholds. In [209] MKL-based combinations of features act together with Bayesian model combination and weighted average fusion of scores from multiple systems.

A number of deep architectures have recently been proposed specifically for multimodal data. Ngiam et al. [212] employ sparse RBMs and bimodal deep antoencoders for learning cross-modality correlations in the context of audio-visual speech classification of isolated letters and digits. Srivastava et al. [276] use a multi-modal deep Boltzmann machine in a generative fashion to tackle the problem of integrating image data and text annotations. Kahou et al. [145] won the 2013 Emotion Recognition in the Wild Challenge by building two convolutional architectures on several modalities, such as facial expressions from video frames, audio signal, scene context and features extracted around mouth regions.

3.5 ACTION AND GESTURE RECOGNITION WITH DEEP LEARNING

Four years ago, when this project was just about to launch, there was relatively little work on applying ConvNets to the problem of classification from video, both in the general setting and in the particular context of action and gesture recognition. It was not until very recently, that **convolutional learning** of data representations started to emerge in the field, replacing spatial, temporal and spatio-temporal handcrafted descriptors, which were the subject of discussion in the previous chapter.

However, even within this short period of time, the computer vision community has made significant progress in applying deep learning ideas to this field, and today there exists a fair amount of literature on action recognition from *single images* and feature learning with 2D convolutional neural networks, on *spatio-temporal volumes* and 3D convolutions, and, finally, on *modeling temporal transitions* with neural recurrent architectures.

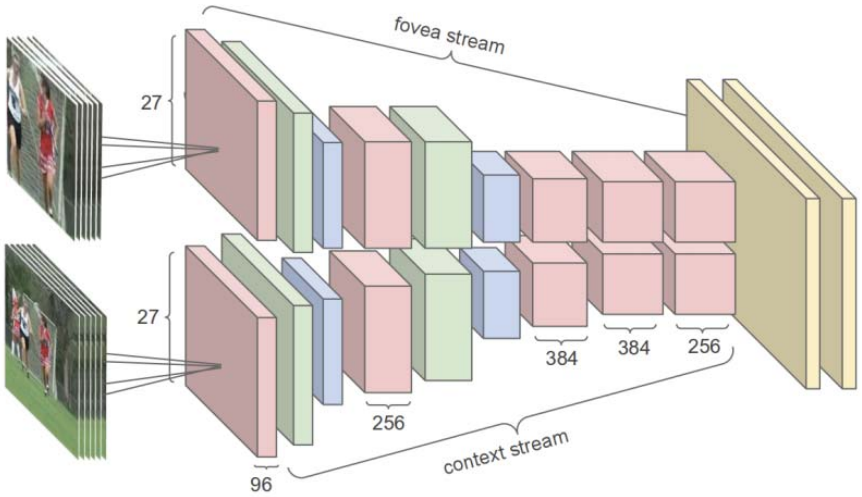


Figure 29 – Multi resolution video classification framework by Karpathy et al. [148].

Action and gesture recognition from *single images* is often based on the same setup, as object recognition models, which typically include joint feature learning and classification done in a single network trained end-to-end [205].

In order to incorporate the temporal aspect, a number of authors extended the dimensionality of the input by concatenating subsequent video frames in *short spatio-temporal blocks*. Accordingly, the dimensionality of convolutional filters in such frameworks is also changed from 2D to 3D [14, 148, 139].

In this category, Karpathy et al. [148], working on the problem of large scale video classification, proposed a multi resolution framework (shown in Figure 29) which includes low-resolution *context stream*, taking as an input a downsampled version of the whole image, and high-resolution *fovea stream*, focusing specifically on the central part of the frame, with no downsampling. Features are extracted from both inputs separately, but parameters of corresponding layers are shared between the two paths. Somewhat surprisingly, in this framework, according to the authors, extending the input with the temporal dimensional resulted in a barely noticeable advantage over multi resolution single-frame processing.

In a recent work, Varol et al. [302] have shown that extending a spatio-temporal framework to *long-term convolutions* (processing up to 60 frames simultaneously), even at cost of reduced spatial resolution, results in better performance. Of course, this result is probably specific to video classification, where a single class is associated to whole videos, and may not be applicable to the case when short gestures should be recognized.

Similar ideas were applied specifically to the problem of *gesture recognition* by Molchanov et al. [194]. Their double resolution framework is based on 3D convolutions applied to concatenated frames, and the final prediction is obtained by multiplying output per-class probabilities from both paths.

In earlier works, a popular approach was to first apply a set of manually defined transforms (such as computing gradients and optical flow) as a preprocessing step. Ji et al. [139], for example, exploited hardwired filters to obtain low level feature maps and then fed them to a 3D convolutional network (ConvNet) for joint learning

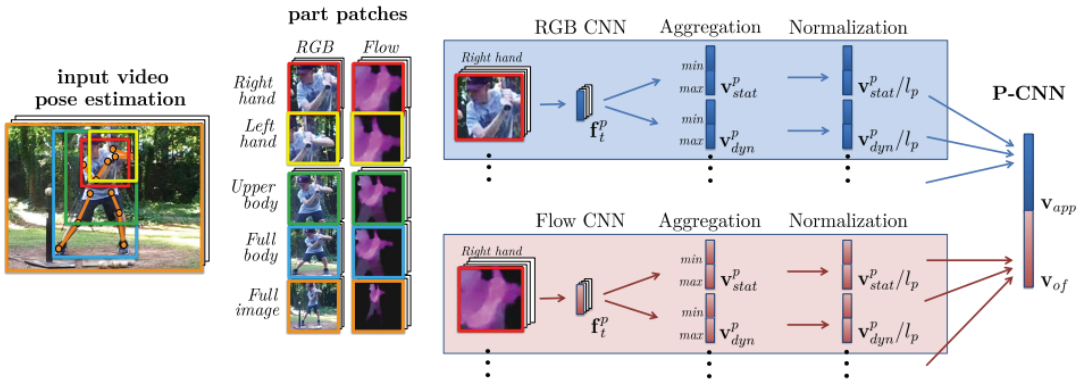


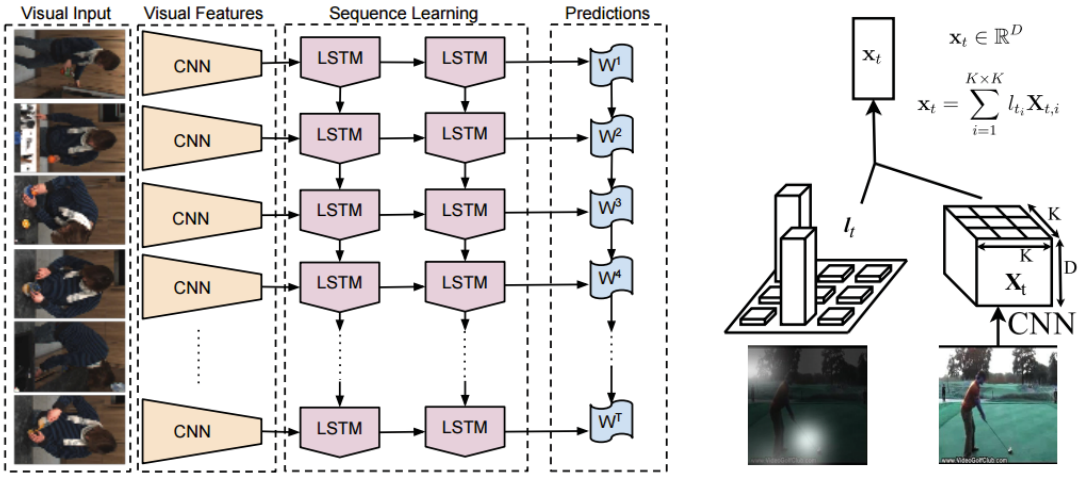
Figure 30 – P-CNN features by Chéron et al. [53]. The figure shows the whole pipeline of feature learning, starting with the input video, patch extraction and learning representations from the RGB channels and optical flow.

of mid-level spatio-temporal representations and classification. Even the most recent works [302, 211, 107, 271] show that careful estimation of optical flow is, at this stage of development, beneficial for learning accurate action models. In this context, Weinzaepfel et al. [313] have been working on solving the problem of patch matching (which they call *DeepFlow*), also in the spirit of deep convolutional networks, but with no learning involved.

Guided by the above observations, Simonyan and Zisserman [271], instead of mixing space and time together as in [148], took another approach to video classification and processed *spatial* and *temporal* streams separately with a pair of ConvNets, whose predictions were then combined by late fusion. In their logic, the two recognition streams are complimentary: while the spatial information is necessary to characterize scenes, the temporal signal conveys solely motion dynamics. In this implementation, the temporal network took as input multi frame dense optical flow, which resulted in significantly better performance compared to the prior art [148]. Working with *egocentric videos*, Poleg et al. [238] used *solely* sparse optical flow volumes (with no spatial color or depth information whatsoever) to recognize human activities from the first person view.

In a more specific context of *action recognition*, instead of extracting features from the whole frame, Chéron et al. [53] proposed to learn motion and appearance representations *along trajectories* of human body parts with what they call *Pose-based Convolutional Neural Networks* (P-CNNs) (see Figure 30). Similarly, Pigou et al. [232] address the problem of *gesture recognition* with two-scale 3D ConvNets, where the first path takes as an input a whole depth image, and the second one processes specifically a region extracted around the human hand (whose position is provided by the Kinect skeleton tracker).

As we have already mentioned, Karpathy et al. [148], as well as their colleagues working on convolutional learning from 3D spatio-temporal volumes, have demonstrated that treating the temporal dimension in the same fashion as the spatial information adds a certain number of points to the recognition performance, but the improvement is not as dramatic as one could expect. For this reason, non surprisingly, an increasing number of deep learning and vision research groups started



(a) (b)

Figure 31 – (a) Long-term Recurrent Convolutional Networks by Donahue et al. [80], where convolutional feature extraction is followed by modeling long term temporal dependencies with a set of LSTM layers. (b) Feature weighting based on soft attention mechanism introduced to (a) by Sharma et al. [263].

focusing more on explicit analysis of extended temporal dependencies with **recurrent temporal models** [14, 80, 329], which we discussed in Section 3.2.

From this category, an influential contribution was the work by Donahue et al. [80], who combined low-level convolutional learning with processing temporal transitions in LSTM layers located closer to the output of the network (see Figure 31a).

This idea is similar in spirit to Simonyan et al. [272] in the sense that the spatial and temporal aspects are kept separated, making the network *double deep* [80] in these two spaces. This composite model, dubbed LRCN (from **Long-term Recurrent Convolutional Network**), has been proven to outperform the prior art on the number of tasks with sequential inputs (such as action recognition, image and video description).

Below, we quote the authors casting their predictions on the near future of computer vision.

> *As the field of computer vision matures beyond tasks with static input and predictions, we envision that "doubly deep" sequence modeling tools like LRCN will soon become central pieces of most vision systems, as convolutional architectures recently have. [80]*

The LRCN model has been used in a number of follow up works, especially in the context of image and video captioning.

In a gesture recognition model by Pigou et al. [233], for example, this architecture was further complexified. In their implementation, the input layers of the network contain intermittent *spatial* and *temporal* convolutions, while modeling longer temporal relations is performed with bi-directional LSTMs.

As a further improvement, Sharma et al. [263] combined the LRCN with the idea of focus selectivity [148]. However, instead of focusing the fovea stream always on the center of the frame, as in [148], it seems to be a promising idea to incorporate **soft attention mechanisms** [17] in the activity recognition pipeline. In this setting,

Figure 32 – Multi stage refinement of body pose estimation in DeepPose framework [297]. Convolutional layers are shown in blue and the fully connected layers are shown in green. The total number of stages in this implementation $s \leqslant 3$.

the model by Sharma et al. [263] adaptively *learns* which parts of the frame are important for the task and the features extracted from the corresponding regions are given higher weights in the classification (see Figure 31b).

3.6 POSE ESTIMATION WITH DEEP LEARNING

Looking back at the taxonomy of pose estimation methods in Chapter 2, we should say that their existing deep learning alternatives are typically discriminative by nature but may include an additional, potentially generative refinement step. However, a recent work by Oberweger et al. [216] also provides interesting insights on fully discriminative-generative learning in neural systems for this application.

The basic pipeline of all state-of-the-art frameworks performing hand pose estimation with deep learning includes convolutional feature extraction from a depth map of a hand with following direct estimation of hand joint positions. A training objective is formulated such that the network outputs either *positions of joints* directly [297, 215, 216], or estimates auxiliary *probability distributions* (called *heat maps*) of each joint being present in a given location [296, 137, 295]. The former provides output for a single input image, whereas the latter provides output for each location, eventually (but not always) in a sliding window fashion.

In the first group of methods, based on **regression on joints**, the network is trained to minimize the mean-square error between normalized ground truth joint positions and produced outputs. Among early works, exploiting this idea, was a facial pose estimation method by Osadchy et al. [224].

In the context of full body pose estimation from color images, this was first implemented in the *DeepPose* framework by Toshev and Szegedy [297] (shown in Figure 32). Their network output has a form of $\tilde{\mathbf{y}}(I;\theta)\in\mathbb{R}^{2|J|}$ where I is an input image, θ is a set of learnable network parameters and $\tilde{\mathbf{y}}$ is a vector of 2D image coordinates of body joints $j\in\mathbf{J}$. Accordingly, the L2 loss for learning parameters θ given a set of ground truth joint positions $\mathbf{y}$ reads:

$$\underset{\theta}{\operatorname{argmin}} \sum_{i\in\mathcal{D}} \sum_{j\in\mathbf{J}} \|\mathbf{y}_j^{(i)} - \tilde{\mathbf{y}}_j^{(i)}(\theta)\|_2^2 \tag{61}$$

where $\mathcal{D}$ is a training set containing pairs of normalized images and annotations, $\mathbf{y}_j^{(i)}$ is a normalized ground truth position of joint j in image i and $\tilde{\mathbf{y}}_j^{(i)}$ is network

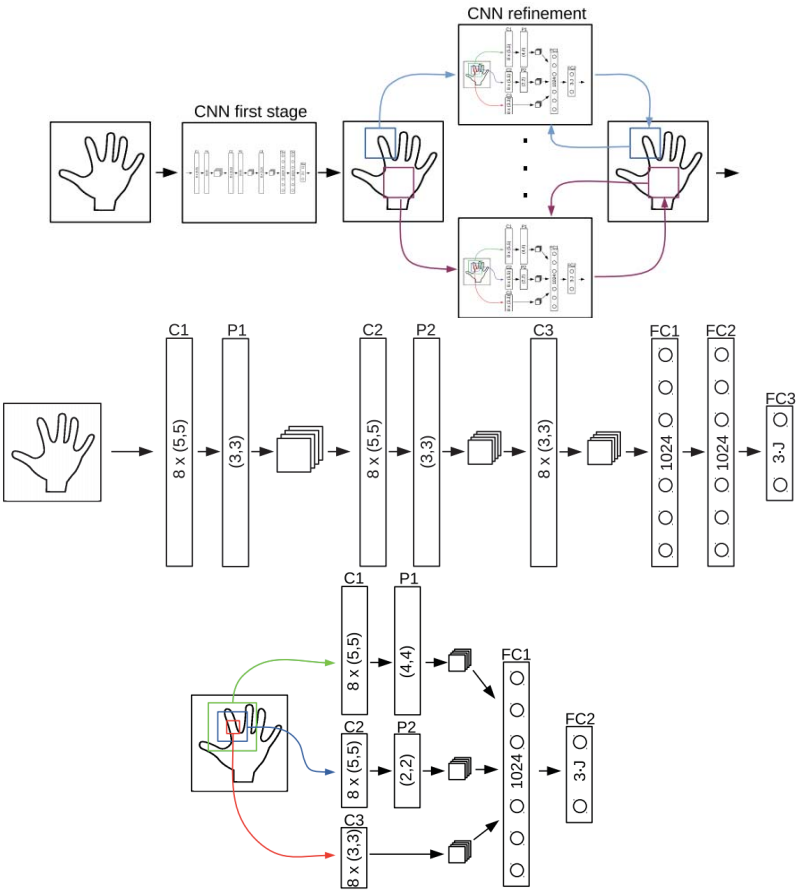


Figure 33 – An overview of a hand pose estimation architecture by Oberweger et al. [215]. First row shows the whole cascaded framework, where the coarse estimation of joint locations with a first-stage deep regressor (second row) is followed by a number of refinement steps performed with multi-resolution regressors with incorporated bottleneck hidden layers to enforce pose constraints (third row).

prediction for a joint j in an image i given a set of network parameters θ. The normalization is performed by detecting a bounding box around the person in the image I with following subtracting its center coordinates and normalizing by its width and height.

This holistic regression approach in the DeepPose framework is combined with a subsequent iterative refinement of predictions based on the output of a previous step (we have previously discussed similar methods in traditional setting in section 2.3.4.2). After initial coarse estimation of joint locations from a low resolution image, a higher resolution fragment, cropped around this point, is used as an input for the next step. At subsequent stages (up to 3 in this implementation), the convolutional regressor predicts displacement of the previously produced prediction from the normalized ground truth location, but relying on more and more detailed input.

The network architecture in this work is the same at each step and consists of 7 layers, among which 5 are convolutional (see Figure 32). However, on each iteration the network learns different sets of parameters specific to the scale of the input.

Similar ideas were applied specifically to the problem of *hand* pose estimation by Oberweger et al. [215], who benchmarked a number of different network architectures on the task of direct regression of hand joint locations (shown in Figure 33).

For this study, the authors tested performance of shallow and deep networks, as well as their multi-scale and multi-resolution extensions [90]. The multi-resolution implementation, unsurprisingly, performed better (at cost of higher network complexity), and was finally chosen for the step of iterative refinement. The coarse regressor was implemented as a single path deep convolutional network.

Their main finding, however, was to introduce a *bottleneck* fully connected layer, which is located closer to the output and performs regularization of network predictions by enforcing a so called *pose prior*. The dimensionality of the bottleneck layer is set to be less than the number of degrees of freedom, i.e. $3|\mathbf{J}|$ (where $\mathbf{J}$ is a set of hand joints and 3 comes from the dimensionality of 3D space). For NYU dataset [296], for example, where the tripled number of joints equals 42, the width of the bottleneck was set to 30, which, together with the multi-resolution refinement, resulted in about 30% improvement of output predictions.

Although the regression based approaches perform sufficiently well in many contexts, some authors raise a number of reasonable criticisms [137] with respect to this strategy. The coordinates of joints, produced by such a regressor, are unbounded, while the very mapping from input space to pose coefficients is highly non-linear and not one-to-one.

Alternatively, there were a number of works performing body and hand pose estimation by forcing the network to output unnormalized **probability distributions** over spatial locations for each joint (also called **heat maps**), which are then post-processed and integrated in order to obtain the final estimate of the joint coordinates.

The intuition behind this approach is that these intermediate heat map representations also describe the *shape* of the predicted distribution, which can be used for several purposes.

First of all, they help to reason about how *confident* the network is in the given prediction – similar, for example, to the refinement step in the work by Tang et al. [288], which we discussed in Section 2.3.5. In that case, the *confidence* of the final prediction was estimated based on the distance between centers of two components in a 2-component GMM fit to the vote distribution. This idea was confirmed by an observation made by Tompson et al. [296] that erroneous predictions typically come from multimodal distributions.

Furthermore, this information can be used to guide a following refinement step, if it is present, in order to choose the correct mode from the distribution by minimizing global energy over the whole pose given some additional priors (we will talk about it later in this section).

An overview of a hand pose estimation framework from the work by Tompson et al. [296], promoting the heat map approach, is shown in Figure 34. They employ a multi-resolution convolutional architecture [90], which takes as an input a 92×92 depth map of a hand and produces $|\mathbf{J}|\times18\times18$ heat maps (i.e. of lower resolution), where $|\mathbf{J}|$ is the number of joints. The ground truth for training is produced by placing truncated 2D Gaussians at ground truth image coordinates of each joint, after which the training is performed using stochastic gradient descent with a standard

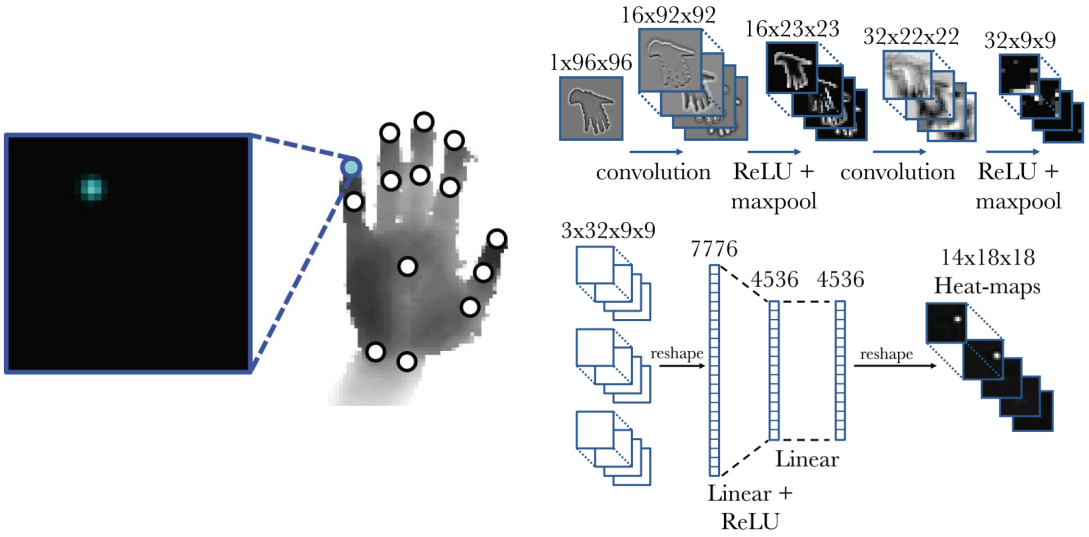


Figure 34 – Hand pose estimation with heat maps. On the left: a depth image with a corresponding ground truth for the pinky tip joint. On the right: a deep learning architecture including a multi-scale convolutional feature extractor (above) and a number of fully connected layers (below) [296].

element-wise L2-norm error function between the ground truth heat maps and the distributions produced by the network.

Finally, during test time, each heat map is used to extract the corresponding predicted joint location by fitting a 2D Gaussian into a maximal lobe of the distribution, to achieve subpixel accuracy. For that, the output heat map is first thresholded to remove low energy noise and then Levenberg-Marquardt is used to find a mean of the Gaussian.

The obtained predictions then serve as an input for an additional **refinement step**, which is based on solving an unconstrained linear optimization problem (similar to *inverse kinematics*), where the objective function is defined in the following way:

$$L(\theta) = \sum_{j \in J} \left(\Delta_j(\theta) + \Phi_j \right), \tag{62}$$

$$\Delta_j(\theta) = \begin{cases} \|(u, v, I(u, v))_j - (\tilde{u}, \tilde{v}, I(\tilde{u}, \tilde{v}))_j\|_2 & \text{if } \exists\, I(u, v), I(\tilde{u}, \tilde{v}) \\ \|(u, v)_j - (\tilde{u}, \tilde{v})_j\|_2 & \text{otherwise} \end{cases} \tag{63}$$

where (u, v) are image coordinates of a joint, in pixels, over which optimization is performed, $(\tilde{u}, \tilde{v})$ are image coordinates of the predicted joint location, $I(u, v)$ and $I(\tilde{u}, \tilde{v})$ are corresponding depth values from the depth map, if they are known. In other words, the first term $\Delta_j(\theta)$ expresses the Euclidean distance between the ground truth and predicted locations in 3D, if there is no hole in the depth map in either of those locations, and in 2D otherwise.

Finally, a linear penalty Φ enforces a constraint on the variables to stay in a predefined range $[C_{min}, C_{max}]$:

$$\Phi_j = \alpha_j [\max(C_j - C_{j,max}) + \max(C_{j,min} - C_j)], \tag{64}$$

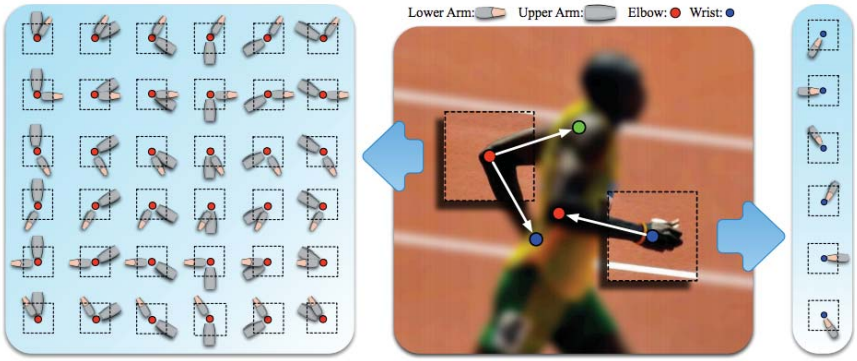


Figure 35 – Image dependent pairwise relations (IDPRs) by Chen and Yuille [51].

where α_j are weighting coefficients, which are set for each joint separately. Similar to what we have discussed in Section 2.3.3, the particle swarm optimization method with partial randomization (PrPSO) [331] was used to find the optimal solution.

In a *full body pose estimation* framework by Tompson et al. [137], refinement (by removing outliers) is performed by introducing an additional *spatial model* consisting of a number of *priors* on pairwise relations between anatomically connected body joints (which generally are much more static than hand joints). This type of model is frequently called a *kinematic tree* and it has been classically used in pose estimation (even before the deep learning era).

In this case, the initial energy distribution in a form of a heat map p_i produced by a ConvNet is considered to be a *unary term*, and the final distribution is defined as follows:

$$\hat{p} \propto p_i^{\lambda} \prod_{j \in \mathbf{J}'} p_{i|j=\vec{0}} * p_j, \tag{65}$$

where $i \in \mathbf{J}$ is a given joint, $j \in \mathbf{J}'$ are its neighboring joints, λ is an empirical parameter reflecting the confidence of each joint's unary distribution and $p_{i|j=\vec{0}}$ is a prior calculated as a histogram of locations of joint i given that the joint j is located in the center of the image (the priors are estimated statistically from the training data by simple histogramming).

Introduced terminology of *unary terms* and *priors* brings us closer to the one of essential ambitions of the community, namely marrying deep learning or data representations with **graphical models** (which becomes particularly meaningful in the context of structured pose estimation).

Going in this direction, Chen and Yuille [51] have introduced a deep learning body pose estimation framework including learning of data driven *unary* and *binary terms* (or, as they call them, **image dependent pairwise relations, IDPRs**), which depend on appearance of image patches containing body joints (see Figure 35). Since they are image-dependent, however, they cannot be seen as prior in a Bayesian sense.

Since in this work, the relational graph representing the human body is formulated as a tree, the inference is efficiently performed using dynamic programming.

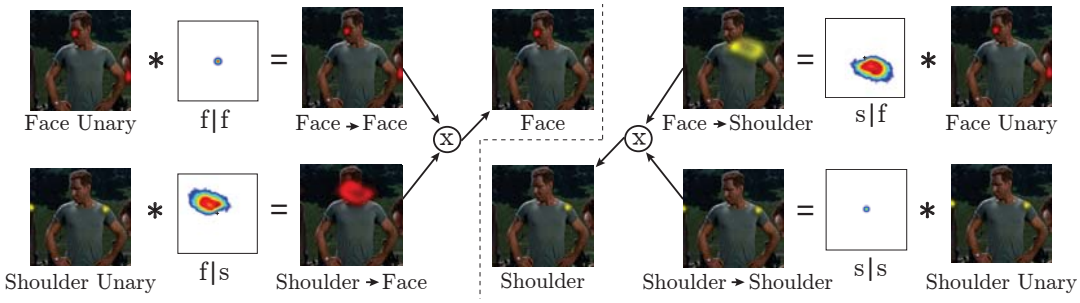


Figure 36 – An illustration of message passing between the face and shoulder joints in [295].

Finally, another work by Tompson et al. [295] shed some light on how **joint training** of deep learning and graphical models could be approximated in a single connectionist framework and applied to full body pose estimation.

In their method, a *spatial model* is formulated similar to an MRF over distributions of spatial locations of body joints. By analogy with equation (65) [137], one step of sum-product belief propagation is formalized as follows (see Figure 36 for a graphical illustration of the idea):

$$\hat{p}_i = \frac{1}{Z} \prod_{j \in J'} \left(p_{i|j} * p_j + b_{i,j} \right), \tag{66}$$

where, as before, p_j is a unary term (heat map), $p_{i|j}$ is a binary term (prior), $b_{i,j}$ is a bias term (describing, in the terminology of graphical models, the background probability of the message from joint i to joint j) and Z is a partition function and $*$ denotes convolution. The priors $p_{i|j}$ intuitively encode distributions of relative joint positions.

In the neural network approximation, this equation is transformed to log space and is reformulated as follows (where SoftPlus and ReLU are added for numerical stability, and e_i, $e_{i|j}$ are energies corresponding to p_i, $p_{i|j}$):

$$\hat{e}_i = \exp\left(\sum_{j \in J'} \left[\log\left(\text{SoftPlus}\left(e_{i|j}\right) * \text{ReLU}\left(e_j\right) + \text{SoftPlus}\left(b_{i,j}\right) \right) \right] \right), \tag{67}$$

Simply speaking, the idea can be summarized as follows. The ConvNet producing unary heat maps is first pretrained and its outputs (while being fixed) are used to train the spatial model with the objective (to maximize) is given by eq. (67). The convolutional weights in eq. 67 are first initialized by priors statistically estimated from the training data and then learned, along with the bias terms. Finally, all constraints are relaxed and both networks are fine tuned together by back-propagation, which further improves pose estimation performance.

We must note that the idea of *structured prediction* with deep learning, where unary terms learned by a ConvNet are combined with MRF/CRF like models, has recently become very popular and is extensively used, for example, in the field of scene parsing [49].

While jointly exploiting graphical models and deep learning predictors is no doubt an exciting direction of research, its applicability to the problem of *hand pose*

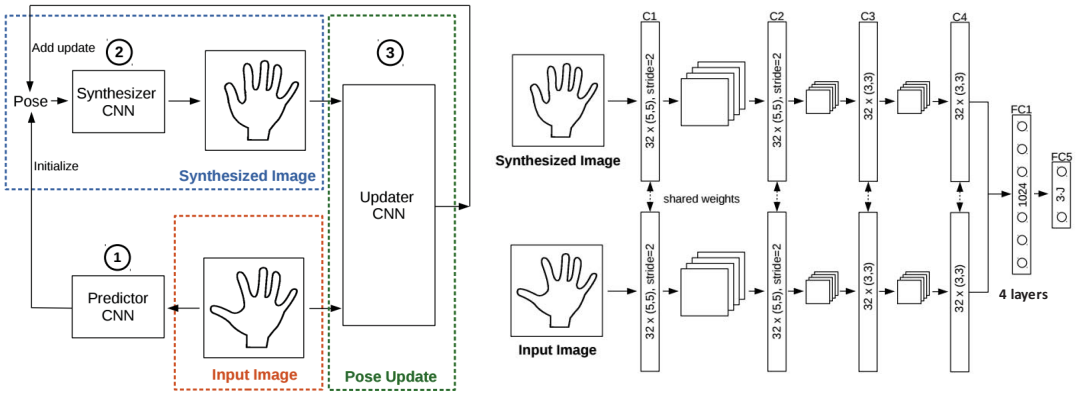


Figure 37 – Hybrid model by Oberweger et al. [216]. On the left: functional overview (see text for description), on the right: architecture of the synthesizer block.

has not yet been explored. Therefore, to conclude this review of prior art on hand pose estimation, we would like to discuss another particularly interesting strategy.

Following the same recent trend of combining *discriminative* and *generative* counterparts in a single framework, as in RDF-based strategies (Sec. 2.3.6), there was an attempt to create a **hybrid model** for hand pose estimation with deep learning, where discriminative and generative steps follow each other interchangeably until convergence [216]. While the most straightforward approach would be to combine discriminative convolutional learning with inverse 3D rendering, in this work the authors go even farther and create a deep learning analog to the *generative* pipeline.

The general framework [216], shown in Figure 37, consists of three functional blocks: predictor $\mathcal{P}$, synthesizer $\mathcal{S}$ and updater $\mathcal{U}$, where the first discriminative block (predictor) is used to initialize the process, after which the generative synthesizer and the discriminative updater acting together lead the training process to a locally optimal solution.

The *predictor* $\mathcal{P}$ is implemented as a single scale deep ConvNet, which is trained, as before, in a purely supervised way to produce the first round of predictions of joint locations (by direct regression on their 3D coordinates). This set of predictions is then used as an input of the *synthesizer* $\mathcal{S}$ which is trained to produce a *depth image* of a hand given the set of joint locations describing its pose (see Figure 38). Finally, an *updater* $\mathcal{U}$ takes both the original and the synthesized images as in input and learns to output a *correction* of the initial prediction, which is then added to the previous estimate. Then the process repeats, until, in theory, convergence, when the joint positions do not longer change (in practice the number of iterations is set to 2).

The described process can be formalized as follows:

$$\tilde{\mathbf{y}}_0 = \mathcal{P}(\mathrm{I}) \tag{68}$$

$$\mathrm{I}_{\mathrm{synth}} = \mathcal{S}(\tilde{\mathbf{y}}) \tag{69}$$

$$\tilde{\mathbf{y}}^{(i+1)} = \tilde{\mathbf{y}}^{(i)} + \mathcal{U}(\mathrm{I}, \mathrm{I}_{\mathrm{synth}}), \tag{70}$$

where I is an input image, $\mathrm{I}_{\mathrm{synth}}$ is an image generated by the synthesizer $\mathcal{S}$, $\tilde{\mathbf{y}}_0$ is the first round of outputs produced by predictor $\mathcal{P}$, and $\tilde{\mathbf{y}}^{(i)}$ is a set of predictions after iteration i.

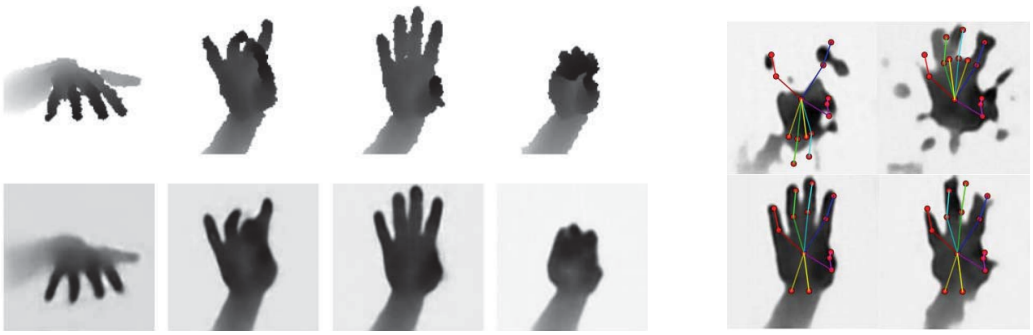


Figure 38 – On the left: original depth maps (above) and images rendered from ground truth joint locations (below). On the right: synthetic images rendered from anatomically unplausible hand poses [216].

The fact that the updater block takes as an input a *pair* of images, rather than a single one, can be considered as a special form of data augmentation. Since for each training image, an arbitrary number of pose hypotheses and, consequently, synthetic counterparts can be generated, this strategy allows the model to explore pairwise relationships between much larger sets of poses than what was initially covered by the training data.

In this implementation, each functional block is trained separately by minimizing a corresponding loss function (which is custom for the updater and is simply defined as L2 norm for two other blocks):

$$\mathcal{P}: \ \underset{\theta_{\mathcal{P}}}{\operatorname{argmin}} \sum_{j \in \mathbf{J}} \|\tilde{\mathbf{y}}_j - \mathbf{y}_j\|_2^2 + \gamma \|\mathbf{W}\|_2^2 \tag{71}$$

$$\mathcal{S}: \ \underset{\theta_{\mathcal{S}}}{\operatorname{argmin}} \frac{1}{|I|} \left|\left| \mathcal{S}(\tilde{\mathbf{y}}) - I \right|\right|_2^2 \tag{72}$$

$$\mathcal{U}: \ \underset{\theta_{\mathcal{U}}}{\operatorname{argmin}} \sum_{j \in \mathbf{J}} \max\left(0, \|\tilde{\mathbf{y}}_j^{(i+1)} - \tilde{\mathbf{y}}_j\|_2 - \lambda \|\tilde{\mathbf{y}}_j^{(i)} - \tilde{\mathbf{y}}_j\|_2\right), \tag{73}$$

where $\mathbf{y}$ is a ground truth vector, γ is a coefficient and $\mathbf{W}$ is a set of weights of the predictor network, $|I|$ is the number of pixels in the image I, and $\theta_{\mathcal{P}}$, $\theta_{\mathcal{S}}$, $\theta_{\mathcal{U}}$ are sets of internal parameters of predictor, synthesizer and updater blocks, respectively.

In discussing the loss functions, the authors particularly stress that direct search for a set of outputs, which would directly minimize the mean square error between the original and rendered images (skipping the updater block), leads to inferior performance, due to the high probability of getting stuck in local minima, divergence and anatomically implausible solutions. The same situation was observed regardless of which optimization algorithm was used.

The discussed manuscript by Oberweger et al. [216] is concluded by drawing an analogy between their hybrid strategy and biological systems, which we provide here for the reader with no change (except for insertion of appropriate citations), as a food for thought.

> *It has been shown that feedback paths in the brain and especially in the visual cortex actually consist of more neurons than the forward path [71]. Their functional role remains mostly unexplained [38]; our approach could be a possible*

explanation in the case of feedback in the visual cortex, but of course, this would need to be proved. [216]

3.7 CONCLUSION

In this chapter, we have reviewed several classes of classic deep learning architectures, which are the most relevant to our work, and also talked about their most recent applications in the field of human motion analysis.

As the reader might have noticed, *deep learning methods* are currently being applied to nearly every related vision problem, with a tendency to replace traditional methods (discussed in Chapter 2), which are based on *feature engineering*. This migration process, however, is far from its completion. We have shown, that in video analysis, for example, careful preliminary estimation of optical flow remains crucial in the state-of-the-art frameworks. Accordingly, there are still a great number of open questions remain to be answered, both in the sense of developing general learning paradigms, as well as their adaptation to the whole concept of motion.

In the following part of this manuscript, we will provide more detailed analysis and introduce our contributions to some deep *temporal architectures*, applied in different contexts, as well as to *multi-modal* neural networks. Naturally, *convolutional learning* will be a leitmotif of the rest of the thesis.

Part II

MAIN PART

"Through seven figures come sensations for a man; there is hearing for sounds, sight for the visible, nostril for smell, tongue for pleasant or unpleasant tastes, mouth for speech, body for touch, passages outwards and inwards for hot or cold breath. Through these come knowledge or lack of it."

— Hippocrates

4 MULTIMODAL GESTURE RECOGNITION

In this chapter, we propose a strategy for gesture detection, classification and localization, which is based on multi-scale and multi-modal deep learning. In our method, each visual modality captures spatial information at a particular spatial scale (such as motion of the upper body or a hand), and the whole vision based system operates at three temporal scales. Key to our technique is a network training strategy which exploits i) careful initialization of individual modalities; and ii) gradual fusion involving random dropping of separate channels (which we call ModDrop) for learning cross-modality correlations while preserving uniqueness of each modality-specific representation.

We present experiments on the ChaLearn 2014 Looking at People Challenge gesture recognition track, in which we placed first out of 17 teams, as well as our earlier results on the ChaLearn 2013 Multi-modal Gesture Recognition dataset. We demonstrate that fusing multiple modalities at several spatial and temporal scales leads to a significant increase in recognition rates, allowing the model to compensate for errors of the individual classifiers as well as noise in the separate channels. Furthermore, the proposed ModDrop training technique ensures robustness of the classifier to missing signals in one or several channels to produce meaningful predictions from any number of available modalities. In addition, we demonstrate the applicability of the proposed fusion scheme to modalities of arbitrary nature by introducing the audio channel.

4.1 INTRODUCTION

As the reader can see from the previous chapters, automatic analysis of human gestures is a quite old and well-researched problem. When we first started working on this project, applying *deep learning* methods in this context was not a common practice, although an amount of literature on *traditional methods*, based on either general or ad-hoc hand specific descriptors, was established.

Even though an increasingly large number of different methods and techniques have been proposed over the last few decades, making gesture recognition work *robustly in a real world setting* still remains an open challenge. This applies particularly to the *human-robot interaction* scenario, which will be the primary context of this chapter.

There are many potential reasons for a gesture recognition system to fail. First of all, it concerns essential cultural and individual differences in tempos and styles of

Figure 39 – Examples of two-scale gesture decomposition: upper body movement and hand articulation. The gesture in the top row can be fully characterized by large-scale body motion, whereas in the one below, subtle finger movements play the primary role. A *full body skeleton*, extracted from range maps, can be seen as an efficient way to summarize large-scale body motion, while *raw* images contain additional information about hand poses, which is, however, more difficult to extract.

human articulation, which, in addition, tend to change over time. Second, one needs to keep in mind, that *recognition* in practice first implies *detection*, and besides *relevant* gesture classes, there also exist infinitely many kinds of *out-of-vocabulary* movements, which need to be modeled, in order to be correctly ignored. Furthermore, a vision-based framework may suffer from typical acquisition-related problems, such as variable observation conditions and noise in camera channels. An additional limitation on a maximal model complexity, that can be afforded, is imposed by a requirement of real-time processing, which is necessary in such applications to maintain a low response time and to make the interaction process natural and intuitive.

An important advantage of modern mobile robots, however, is that they are typically equipped with *a number* of built-in sensors, including, among others, a color and a range camera, and a microphone. All these sensors can capture gesture-related information on different levels of abstraction (such as, for example, full body motion vs fine hand articulation – see Figure 39 for an illustration) and from completely different perspectives (such as video vs audio signals).

Naturally, a *visual signal* is expected to play the most important role in gesture recognition. In the human-robot interaction setting, however, a distance between a person and a camera can, in a general case, be significant. This implies, that the size of hands, and especially fingers, in images taken by a robot sensor in such

conditions, will not exceed several pixels. Given that Kinect-like RGB-D cameras typically have a quite low image resolution, fingers are therefore likely to be not discernible.

At the same time, a *body silhouette* typically remains visible and may provide useful cues for gestural interaction. However, a depth-based automatic tracking of a *body skeleton*, which is currently a part of many standard camera SDKs, often fails – for example, when an optical axis of the depth sensor coincides with extended limbs. In addition, it may also fail on rare poses, or when hands are simply in immediate proximity of other body parts. Furthermore, low-pass filtering, which is typically employed to compensate for jitter in detected joint positions, may cause additional delays in the skeleton localization.

Finally, an *audio channel* can convey additional relevant signal, if a gestural instruction to a robot is doubled verbally. But, apart from the fact that it is not always the case, audio processing has its own associated challenges, including handling differences in voice tones, accents, styles and language vocabularies, as well as handling typically present noise.

To summarize, in unconstrained real-world conditions, each of the mentioned sensors is *likely to fail*, when acting alone. However, combining a number of different by nature channels in a *single multi-modal framework* may open up new perspectives on human-robot interaction, and dramatically increase effectiveness of such interfaces.

In this chapter, we define two main challenges, which we will aim to address. Our first objective will be to find a way to effectively and efficiently *learn human gesture representations* from a number of input signals, regardless of their nature, for which we will adapt the *deep learning* methodology. Second, we will particularly focus on aspects of *fusion* of multi-modal representations, learning cross-modality correlations and ensuring *robustness* of the whole framework to signal corruption.

Before we proceed with description of our contributions, it is important to note that all experiments, which will be presented in this chapter, have been conducted on a standard multi-modal gesture recognition benchmark, rather than on our own data. This was done, first of all, to ensure reproducibility of the results and for the sake of comparison with existing state-of-the-art gesture recognition methods. However, in the context of the industrial project, which funded our research, a new dataset of human-robot interactions has been shot. Evaluation of the proposed methods on this data is an ongoing work, which will not be presented in this thesis.

As we have already mentioned in Section 2.2.1, the only publicly available gesture-oriented dataset, which contains a sufficient amount of training samples for deep learning methods, was proposed for the *ChaLearn 2013 Challenge on Multi-modal gesture recognition* [86] and, a year later, the same data corpus, but with significantly cleaner annotations, was released under the umbrella of the *ChaLearn 2014 Looking at People Challenge* (gesture recognition track) [87]. This dataset (which is lately received the name of *Montalbano dataset*) contains multi-modal recordings of Italian conversational gestures, along with out-of-vocabulary movements, performed by 36 different people, and the challenge if formulated as a gesture *detection*, *recognition* and *localization* task. The deep learning method, which is described in this chapter, placed first in the 2014 version of this competition.

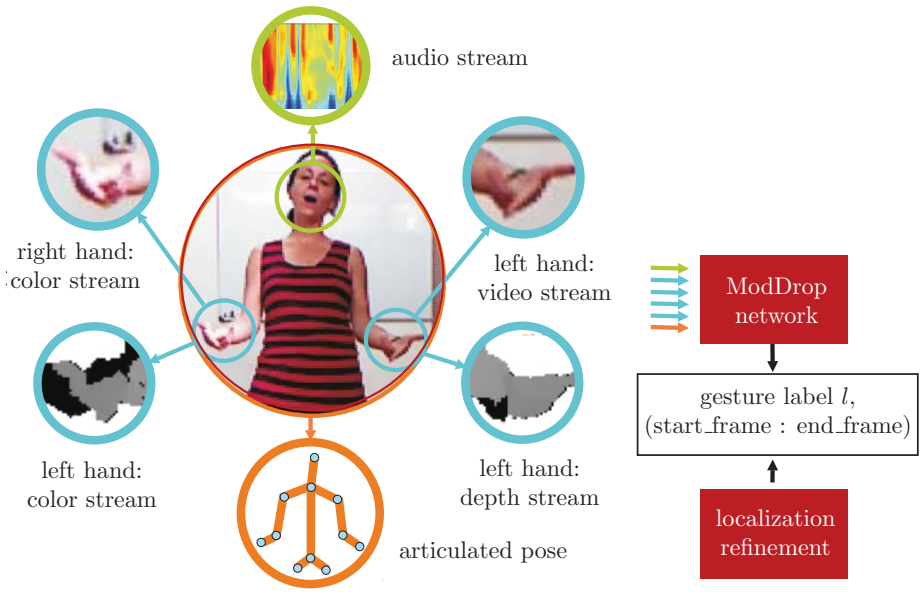


Figure 40 – Overview of our method on an example from the 2014 ChaLearn Looking at People dataset (gesture recognition track) [87]. Multi-modal and multi-scale gesture decomposition is followed by classification by the deep ModDrop network and is combined with parallel gesture localization.

A core aspect of our approach is employing a multi-modal convolutional neural network for classification of so-called *dynamic poses* of varying durations (i.e. temporal scales).[1] Visual data modalities, which are integrated by our algorithm, include intensity and depth video, as well as articulated pose information (or, a full body skeleton) extracted from depth maps (see Figure 40). In our framework, we make use of different data channels to decompose each gesture at multiple scales, not only temporally, but also spatially, to provide context for upper-body body motion and more fine-grained hand/finger articulation.

Furthermore, in this chapter, we pay special attention to developing an *effective learning algorithm*, since training of large-scale multi-modal networks, like the one described here, is a formidable challenge, especially when the amount of training data is strictly limited. In addition, we introduce an advanced multi-modal training strategy, dubbed *ModDrop*, that makes the network predictions robust to missing or corrupted signals in one or more data channels.

Our gesture classification model outputs prediction updates in real-time in a frame-wise manner. However, since temporal integration is involved, precision of gesture localization suffers from a certain degree of inertia. Furthermore, due to high similarity between gesture classes on pre-stroke and post-stroke phases, frame-wise classification at that time is often uncertain and therefore erroneous. To compensate for these negative effects, we introduce an additional module, which is necessary for filtering, denoising and more accurate gesture localization.

1. In our earlier submission, created for the 2013 version of the challenge, we exploited a *temporal deep learning* model (instead of dynamic poses), which we will also discuss in this chapter.

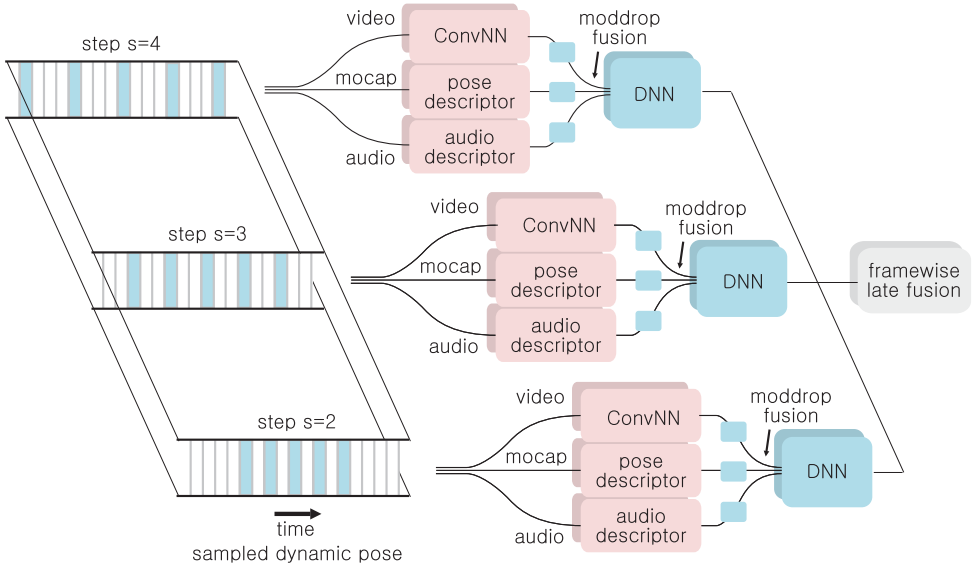


Figure 41 – The deep convolutional multi-modal architecture operating at three temporal scales corresponding to dynamic poses of three different durations. Although the audio modality is not present in the 2014 ChaLearn Looking at People Challenge dataset, we have conducted additional experiments by augmenting the visual signal with audio recordings from the 2013 version of the data.

Finally, we demonstrate, that the proposed scheme can be naturally augmented with more data channels of arbitrary nature, by introducing an additional audio signal to the classification framework.

To summarize, the major contributions of the work, which will be presented in this chapter, are the following:

— we develop a deep learning-based multi-modal and multi-scale framework for gesture detection, localization and recognition, which can be augmented with channels of an arbitrary nature (demonstrated by experiments with an audio signal);

— we propose and theoretically analyze the *ModDrop* procedure for effective fusion of multiple modality channels, which targets learning cross-modality correlations, while prohibiting false co-adaptations between data representations and ensuring robustness of the classifier to missing signals.

4.2 GESTURE CLASSIFICATION

On a large-scale multi-modal dataset, such as *ChaLearn 2014 Looking at People* [87], we face several key challenges, among which are *learning representations* at multiple spatial and temporal scales, *integrating* the various input modalities, and *training* a complex model, in conditions when the number of labeled training examples is not at *web-scale*, like in widely used static image datasets (ImageNet [162], for example).

In this section, we start our discussion by describing how the first two challenges, namely learning and integrating data representations, can be overcome at an archi-

tectural level. Our training strategy, addressing the third issue, will be the subject of the next section (4.3).

The proposed *multi-scale* deep neural model consists of a combination of single-scale paths connected in a parallel way (shown in Fig. 41). In this setting, each path independently learns an input representation and performs gesture classification at its own temporal scale, given input from RGB-D video and articulated pose descriptors (audio channel can be also added, if available).

Predictions from all multi-modal paths are then aggregated through additive late fusion. This strategy allows us to, first, extract the most salient (in a discriminative sense) motions at a fine temporal resolution and, at the same time, consider them in the context of global gesture structure, smoothing and compensating for per-block errors typical for a given gesture class.

To differentiate among temporal scales, we introduce a notion of a *dynamic pose*. Here and in the rest of this chapter, by *dynamic pose* we mean a sequence of video frames, synchronized across modalities, sampled at a given temporal step s and concatenated to form a spatio-temporal 3D volume.

Varying the value of s allows the model to leverage multiple temporal scales for prediction, thereby accommodating differences in tempos and styles of articulation of different users. Our model is therefore different from, for example, the one proposed by Farabet et al. [90] for image segmentation, where by *multi-scale* the authors imply a multi-resolution spatial pyramid, rather than a fusion of temporal sampling strategies. Regardless of the value of step s, we analyze the same number of frames (5) at each scale. Figure 41 illustrates the three single scale multi-modal paths, which are used in this work (with s=2...4). At each temporal scale and for each dynamic pose, the classifier outputs a per-class score.

All available modalities, such as depth, gray scale video, and articulated pose (as well as the audio signal, if provided), contribute to the network's prediction. In particular, global appearance of each gesture instance is captured by the skeleton descriptor, while video streams convey additional information about hand shapes and their dynamics, which are crucial for discriminating between gesture classes performed with similar full body poses.

Although *inter-scale fusion* in this framework is performed at the final stage by late aggregation of per-classifier predictions, developing an optimal strategy for *inter-modality fusion* is more challenging. Due to the high dimensionality of the data input and the non-linear nature of cross-modality structure, an immediate concatenation of raw skeleton and video signals appears to be sub-optimal. However, initial discriminative learning of *individual* data representations from each isolated channel followed by fusion has proven to be efficient in similar tasks [212].

In our approach, discriminative data representations are first learned within each separate channel, followed by joint fine tuning and fusion by a meta-classifier. This procedure is performed independently at each scale, which we will describe in Section 4.3 in more detail. The architecture of a single scale multi-modal predictor is shown in Figure 42.

A shared set of hidden layers is employed at different levels for, first, fusing of *similar by nature* gray scale and depth video streams and, second, combining the

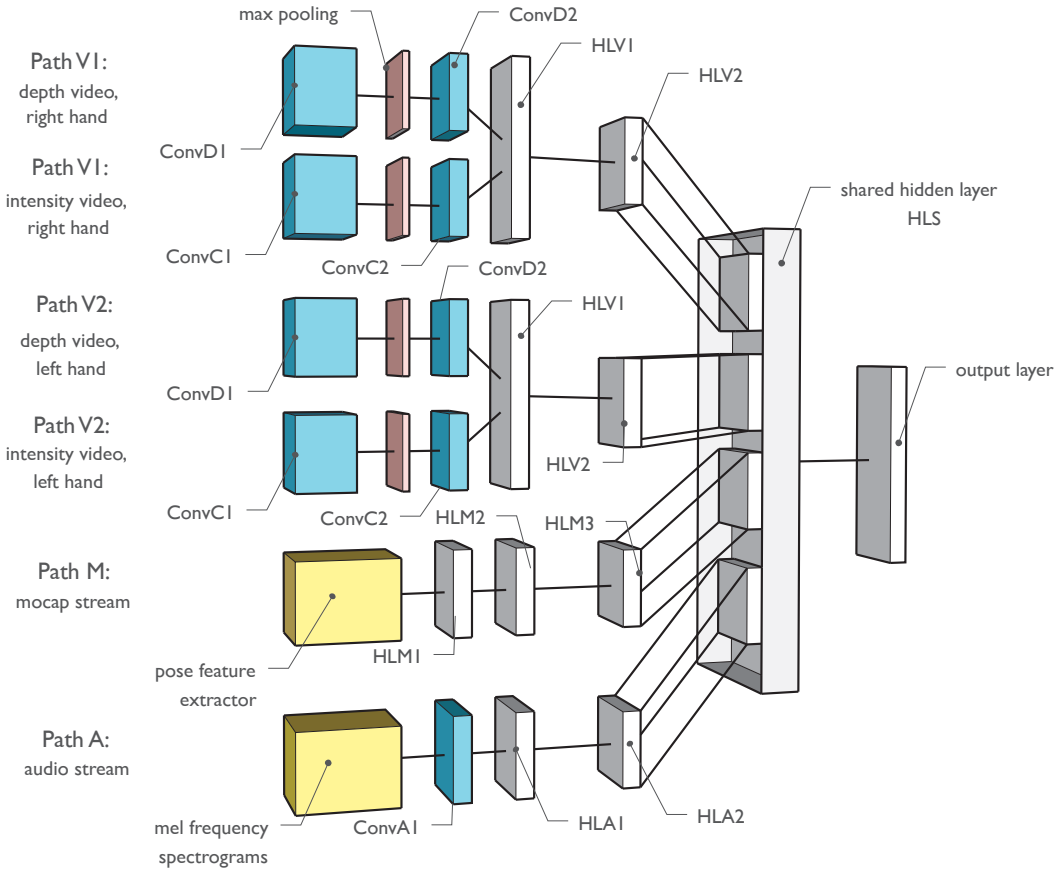


Figure 42 – Single-scale deep architecture. Individual classifiers are pre-trained for each data modality (paths V1, V2, M, A) and then fused using a 2-layer shared fully connected network initialized in a specific way (see Sec. 3.4). The first layers of paths V1 and V2 perform 3D convolutions followed by 3D max pooling eliminating the temporal dimension (not to be confused with V1 and V2 in the human visual system). The second layers on these paths are exclusively spatial. Weights are shared across the V1 and V2 paths.

obtained joint video representation with the transformed articulated pose descriptor (and audio signal, if available). To ensure *detection* functionality, rather than just *classification*, all video fragments from the training data, which are not annotated with any of target gesture classes, are used as *zero-class* samples.

Let us now describe, how learning gesture representations is performed, first separately from each input modality.

4.2.1 *Articulated pose*

A typical full body skeleton, tracked by modern consumer depth cameras and associated middleware, usually consists of 20 (or fewer) body joints, which are identified by their coordinates in a 3D coordinate system, aligned with the depth sensor. For our purposes, in the context of gesture recognition, we exploit only 11 of these

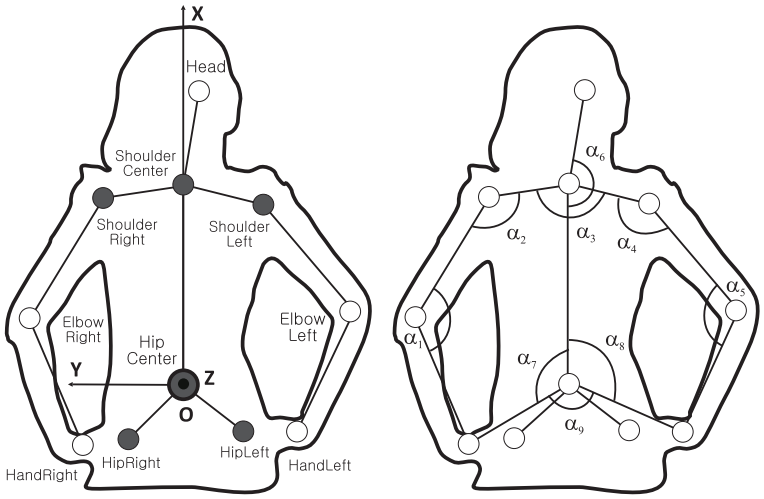


Figure 43 – The proposed pose descriptor is calculated from normalized coordinates of 11 upper body joints, also including their velocities and accelerations, three sets of angles (triples of joints forming inclination angles are shown on the right) and pairwise distances. The body coordinate system (shown on the left) is calculated from 6 torso joints (shown in dark gray on the left).

joints, which correspond to the upper body (shown in Figure 43). In our framework, we also do not use wrist joints as their detected positions are often unstable.

As an input for the network, we formulate a pose descriptor, consisting of 7 logical subsets, and allow the classifier to perform online feature selection.

We denote raw, i.e. pre-normalization, positions of 11 upper body joints in a 3D coordinate system, which is associated with the depth sensor, as $\mathbf{p}_{\text{raw}}^{(i)} = \{x^{(i)}, y^{(i)}, z^{(i)}\}$, $i = 0...10$, where i=0 corresponds to the *HipCenter* joint.

Following the procedure proposed by Zanfir et al. [334], we first calculate normalized joint positions, as well as their velocities and accelerations, and then augment the descriptor with a set of characteristic angles and pairwise distances (similar to what is done in [43]).

— **Normalized joint positions**. The body skeleton is represented as a tree structured graph, with the *HipCenter* joint playing the role of a root node. For the sake of normalization, the coordinates of the root node are subtracted from the rest of the vectors $\mathbf{p}_{\text{raw}}$, in order to eliminate the influence of the absolute position of the body in space. To compensate for differences in body sizes, proportions and shapes, we start from the top of the tree and iteratively normalize each skeleton segment to a corresponding average *bone* length, which is statistically estimated from all available training data.

This normalization step is performed in such a way, that absolute 3D positions of body joints are corrected, while orientations of corresponding *virtual bones*, connecting the joints, remain unchanged:

$$\mathbf{p}^{(i)} = \mathbf{p}_{\text{raw}}^{(i-1)} + \frac{\mathbf{p}_{\text{raw}}^{(i)} - \mathbf{p}_{\text{raw}}^{(i-1)}}{\|\mathbf{p}_{\text{raw}}^{(i)} - \mathbf{p}_{\text{raw}}^{(i-1)}\|} b^{(i-1,i)} - \mathbf{p}_{\text{raw}}^{(0)}, \tag{74}$$

where $\mathbf{p}^{(i)}_{raw}$ is a vector of raw coordinates of the current joint, $\mathbf{p}^{(i-1)}_{raw}$ is a vector of raw coordinates of its parent in the tree structured graph, $b^{(i-1,i)}$ ($i=1\ldots10$) is a set of estimated average lengths of virtual bones, and $\mathbf{p}$ is a vector of corresponding normalized coordinates of the current joint. Once the normalized joint positions are obtained, we perform Gaussian smoothing along the temporal dimension ($\sigma=1$, filter size 5×1) to decrease the influence of skeleton jitter.

— **Joint velocities** are calculated as first derivatives of the normalized joint coordinates (approximated as differences between joint positions in subsequent frames at times t and $t-1$):

$$\delta\mathbf{p}^{(i)}(t) \approx \mathbf{p}^{(i)}(t+1) - \mathbf{p}^{(i)}(t-1). \tag{75}$$

— **Joint accelerations** correspond to the second derivatives of the same positions:

$$\delta^2\mathbf{p}^{(i)}(t) \approx \mathbf{p}^{(i)}(t+2) + \mathbf{p}^{(i)}(t-2) - 2\mathbf{p}^{(i)}(t). \tag{76}$$

— **Inclination angles** are formed by all triples of *anatomically connected* joints (i,j,k), plus two *virtual* angles *RightElbow–RightHand–HipCenter* and *LeftElbow-LeftHand-HipCenter* (shown in Figure 43, on the right),

$$\alpha^{(i,j,k)} = \arccos \frac{(\mathbf{p}^{(k)} - \mathbf{p}^{(j)})(\mathbf{p}^{(i)} - \mathbf{p}^{(j)})}{\|\mathbf{p}^{(k)} - \mathbf{p}^{(j)}\| \cdot \|\mathbf{p}^{(i)} - \mathbf{p}^{(j)}\|}. \tag{77}$$

— **Azimuth angles** β provide additional information about the pose in the coordinate space associated with the body. We apply PCA on the positions of 6 torso joints (*HipCenter*, *HipLeft*, *HipRight*, *ShoulderCenter*, *ShoulderLeft*, *ShoulderRight*) (shown in dark gray in Figure 43) to obtain 3 vectors, which form the basis: $\{\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z\}$, where $\mathbf{u}_x$ is approximately parallel to the shoulder line, $\mathbf{u}_y$ is aligned with the spine and $\mathbf{u}_z$ is perpendicular to the torso.

Then for each pair of connected bones, β are angles between projections of the second bone ($\mathbf{v}_2$) and the vector $\mathbf{u}_x$ ($\mathbf{v}_1$) on the plane, which is perpendicular to the orientation of the first bone. As in the previous case of inclination angles, we also include two *virtual bones RightHand-HipCenter* and *LeftHand-HipCenter*. The azimuth angles are calculated as follows:

$$\begin{aligned} \mathbf{v}_1 &= \mathbf{u}_x - (\mathbf{p}^{(j)} - \mathbf{p}^{(i)})\frac{\mathbf{u}_x \cdot (\mathbf{p}^{(j)} - \mathbf{p}^{(i)})}{\|\mathbf{p}^{(j)} - \mathbf{p}^{(i)}\|^2}, \\ \mathbf{v}_2 &= (\mathbf{p}^{(k)} - \mathbf{p}^{(j)}) - (\mathbf{p}^{(j)} - \mathbf{p}^{(i)})\frac{(\mathbf{p}^{(k)} - \mathbf{p}^{(j)})(\mathbf{p}^{(j)} - \mathbf{p}^{(i)})}{\|\mathbf{p}^{(j)} - \mathbf{p}^{(i)}\|^2}, \\ \beta^{(i,j,k)} &= \arccos \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|\|\mathbf{v}_1\|}. \end{aligned} \tag{78}$$

Bending angles γ are a set of angles between a basis vector $\mathbf{u}_z$, perpendicular to the torso, and normalized joint positions:

$$\gamma^{(i)} = \arccos \frac{\mathbf{u}_z \cdot \mathbf{p}^{(i)}}{\|\mathbf{p}^{(i)}\|}. \tag{79}$$

Pairwise distances. Finally, we calculate pairwise distances between all normalized joint positions:

$$\rho^{(i,j)} = \|\mathbf{p}^{(i)} - \mathbf{p}^{(j)}\|. \tag{80}$$

Combined together, these seven set of features produce a 172-dimensional pose descriptor for each video frame:

$$\mathbf{D} = [\mathbf{p}, \delta\mathbf{p}, \delta^2\mathbf{p}, \alpha, \beta, \gamma, \rho]^T. \tag{81}$$

Finally, each feature is normalized to zero mean and unit variance, and a set of consequent 5 frame descriptors, sampled at a given temporal stride s, are concatenated to form a 860-dimensional dynamic pose descriptor, which is further used for gesture classification. The two subsets of features, which involve the first and the second derivatives, contain dynamic information and for dense sampling may be therefore partially redundant, as several occurrences of same frames are stacked when a dynamic pose descriptor is formulated. Although theoretically unnecessary, in this case, this formulation remains beneficial in the context of a limited amount of training data.

The skeleton descriptor, calculated for each dynamic pose, is then fed to the corresponding branch of the multimodal classifier (Path M in Figure 42), and this part of the network, containing 3 fully connected layers, is pre-trained discriminatively for classification of each dynamic block, based solely on the skeleton data.

4.2.2 *Depth and intensity video: convolutional learning*

In our approach, two *video streams* (RGB and depth) serve as a source of information about *hand pose* and *finger articulation*. Accordingly, to focus specifically on these objects of interest, each frame is used to extract a pair of bounding boxes, which contain images of a right and left hand. These bounding boxes are cropped around positions of the *RightHand* and *LeftHand* joints, provided by the skeleton tracker.

In order to eliminate the influence of the person's position with respect to the camera and to keep the hand size approximately constant, the size of each bounding box is normalized by the distance between the hand and the sensor, as follows:

$$H_x = \frac{h_x X}{z \cdot \tan(\alpha_{FoV,x})}, \quad H_y = \frac{h_y Y}{z \cdot \tan(\alpha_{FoV,y})}, \tag{82}$$

where H_x and H_y are the hand sizes (in pixels) along the x and y axes of the camera sensor, h_x and h_y are the physical sizes of an average hand, in mm, $X \times Y$ are the frame dimensions, in pixels (i.e, for the typical case of VGA resolution, 640×480), z is the distance between the hand and the camera in mm (estimated from the depth map), and $\alpha_{FoV,x}$ and $\alpha_{FoV,y}$ are the camera field of view along the x and y axes respectively (which are intristic parameters of the sensors).

Within each set of frames forming a dynamic pose, hand position is stabilized by minimizing inter-frame square-root distances calculated as a sum over all pixels. As in the case of skeleton descriptors, images of each hand, extracted from subsequent frames, are concatenated to form a single spatio-temporal volume.

As a final preprocessing step, the color stream is converted to gray scale, and each depth and intensity frame is normalized to zero mean and unit variance.

Since the main objective of the modality-wise pre-training step is to learn data representations, which are discriminative for the task of gesture recognition, it is

important to ensure that, in the case of video data, such a representation captures *elements of hand poses*, which are specific and characteristic for a given gesture class. Accordingly, during this pre-training step, we adapt the video pathways to produce predictions *for each hand*, rather than for the whole gesture. For this purpose, left hand videos are flipped about the vertical axis and combined with right hand instances in a single training set.

At the same time, in reality, each gesture can be performed with either the *right* or *left* hand, while some gesture classes require symmetric use of *both* hands. This implies that a ground truth *gesture label* alone, without additionally specifying *which hand is active* in this gesture, does not fully characterize hand movements and cannot be directly used as a target for training for both hand images. In order to eliminate this possible noise associated with switching from one active hand to another, we introduce an additional step of unsupervised hand labeling. For one-handed gesture classes, we detect the active hand and adjust the class label for the inactive one. In particular, we estimate the motion trajectory length of each hand using the respective joints provided by the skeleton stream (summing lengths of hand trajectories projected to the x and y axes):

$$\Delta = \sum_{t=2}^{5} \Big(\big|x(t) - x(t-1)\big| + \big|y(t) - y(t-1)\big| \Big), \tag{83}$$

where $x(t)$ is the x-coordinate of a hand joint (either left or right) and $y(t)$ is its y-coordinate. As a result, the hand with a longer trajectory Δ gets assigned to the label class, while the second one is assigned the zero-class "no action" label.

Finally, for each channel and each hand, we perform 2-stage convolutional learning of data representations independently (first in 3D, then in 2D space, see Paths V1 and V2 in Figure 42) and then fuse the two streams with a set of fully connected hidden layers. Parameters of the convolutional and fully-connected layers at this step are shared between the right hand and left hand pathways.

Our experiments have demonstrated that relatively early fusion of depth and intensity features leads to a significant increase in performance, even though the quality of predictions obtained from each channel alone is unsatisfactory.

4.2.3 *Audio stream*

First of all, we would like to note, that in the context of gesture recognition, the audio signal interests us mostly from the perspective of modeling *fusion dynamics*, as this modality is fundamentally different from the video inputs. For this reason, advancing the field of audio processing and speech recognition per se is something which is out of the scope of this work.

Generally, recent works in the field of speech processing have demonstrated that exploiting *raw* or slightly preprocessed audio data in combination with deep learning architectures leads to higher performance in comparison with *traditional* systems, which are based on hand crafted features (typically from the family of Mel-frequency cepstral coefficients, or MFCC). Deng et al. [74] from Microsoft have demonstrated the advantage of using primitive spectral features, such as 2D spectrograms, in combination with deep autoencoders. Ngiam et al. [212] applied the same strategy to

the task of multi-modal speech recognition while augmenting the audio signal with visual features.

Further experiments from Microsoft [74] have shown, that convolutional neural networks appear to be especially efficient in this context, since they allow the capture and modeling of structure and invariances that are typical for speech. The most advanced state-of-the-art speech recognition systems DeepSpeech [119] and DeepSpeech2 [5] exploit a LRCN-like architecture [80], which we discussed in Section 3.5 and that includes a number of convolutional layers, applied along the temporal dimension, followed by a number of recurrent layers.

The *ChaLearn 2013* dataset originally contained audio recordings, where study participants, demonstrating Italian conversational gestures, were asked to verbally express the meaning of the gesture they were performing, without being forced to pronounce exactly the same words each time. While working on this project, we have explored both *traditional* and *deep learning* approaches to the problem of learning gesture representations from this signal, which we will briefly describe below.

4.2.3.1 *Traditional framework*

Our *traditional* audio processing module uses a simple word-spotting strategy, which assumes that each gesture has a limited verbal vocabulary associated with it. However, numerous practical issues such as illiterate speech, variations in dialects, differences in speech levels (e.g. idiomatic, casual, even ungrammatical colloquial speech vs grand style) make high demands on the level of system generalization.

In order to avoid using complex language models, we associate each class with a single virtual gesture *word*, which is defined as a set of verbal words (in the linguistic sense), word-combinations or short phrases, which have the same semantic meaning and typically accompany each gesture from a given class.

For this purpose, we construct a dictionary by including all possible utterances, which are associated with each gesture word in the form of sequences of phonemes. Here we do not differentiate between slight variations in phrase constructions and different pronunciations of the same phrase.

For example, an Italian gesture *sei_pazzo?* ("are you crazy?") can be associated with phonetic transcriptions *s-E-i-p-a-tts-o*, *m-a-s-E-i-p-a-tts-o* and *s-E-i-m-a-tt-o* (corresponding to the *sei pazzo*, *ma sei pazzo* and *sei matto* orthographic forms). Depending on the training data and the task at hand, the dictionary can be populated by hand, aiming on including the greatest possible number of ways to express every single gesture.

The proposed traditional framework consists of two modules (implemented with the Julius LVCSR engine [172]). First, Voice Activity Detection (VAD) is applied to isolate single speech gesture events (with start and end timestamps), then an automatic speech recognition (ASR) system takes over considering each isolated event as a word instance.

Typically, ASR systems provide a lattice (also called a *wordgraph*), that for each recognized word gives timing, scores and possible connections with other recognized words. For this task, we simplified the algorithm to produce an n-best list for every

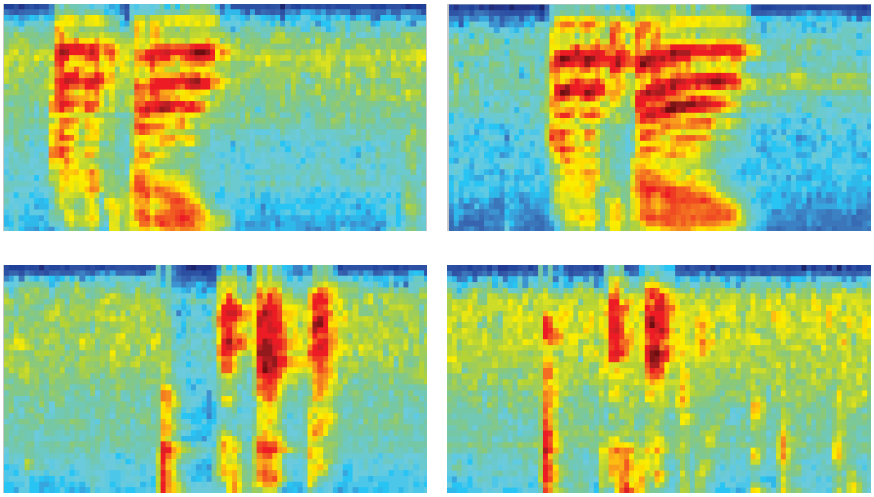


Figure 44 – Mel-scaled spectrograms of two pairs of audio samples corresponding to different types of Italian conversational gestures: (a) *cosa ti farei* and (b) *perfetto* (shown in false colors).

gesture event. As a result, each list contains an infinite-dimension sparse vector of *hypotheses*, i.e. predictions of gesture classes with associated confidence scores.

As the last step, we select a fixed number W of hypotheses with the highest scores and treat the ASR system output in the bag-of-words fashion calculating frequencies of appearances of each of N gesture classes and normalizing the counts by W. As a result, we obtain a distribution of class probabilities that has the same structure as the outputs produced from the video and skeleton modalities.

4.2.3.2 *Deep learning framework*

Comparative analysis of our first phoneme recognition based approach, which we have just described, and an alternative deep learning framework, which we describe in this section, has demonstrated that the latter strategy resulted in significantly better performance on the ChaLearn dataset 2013 (see Section 4.7 for more details). Therefore, in the final version of this work, the audio signal is processed in the same spirit as video data, i.e. by feature learning with a convolutional architecture.

In our deep learning framework, the preprocessing step consists of basic noise filtering and speech detection by thresholding the raw signal along the absolute value of the amplitude (τ_1). Short, isolated peaks of duration less than τ_2 are also ignored during training.

Next, we apply a short-time Fourier transform on the raw audio signal to obtain a 2D local spectrogram, which is further transformed to the Mel-scale to produce 40 log filter-banks on the frequency range from 133.3 to 6855.5 Hz (i.e. the zero-frequency component is eliminated). In order to synchronize the audio and visual signals, the size of the Hamming window at each temporal scale is chosen to correspond to the duration of a dynamic pose (for the corresponding sampling step s) with half-frame overlap. A typical appearance of such Mel-scaled spectrograms for two gesture classes is illustrated in Figure 44. As it was experimentally demon-

strated by Deng et al. [74], the step of the scale transform is important to facilitate the training process. Even state-of-the-art deep architectures, until recently, had difficulty in learning these kind of non-linear transformations.

A one-layer convolutional network in combination with two fully-connected layers form the corresponding path which we, as before, pre-train for preliminary gesture classification from short utterances. The output of the penultimate layer provides audio features for data fusion and modeling temporal dependencies (see Sec. 4.3).

4.3 TRAINING PROCEDURE

In this section, we describe the most important architectural solutions that were critical for our multi-modal setting, including *per-modality pre-training* and aspects of *fusion* such as the initialization of shared layers. Also, we introduce the concept of multi-modal dropout (ModDrop), whose primary goal is to minimize the network sensitivity to the loss of one or more channels.

4.3.1 *Pretraining of per-modality channels*

Depending on the source and physical nature of a signal, the input representation of any modality is characterized by its dimensionality, information density, associated correlated and uncorrelated noise, and systematic errors in measurements. For this reason, a single network, which takes as an input a combined collection of features from all channels is suboptimal, since *uniformly distributing* the parameters per input is likely to *overfit* one subset of features and *underfit* the others.

In this case, performance-based optimization of network hyper-parameters may resolve in cumbersome deep architectures, which would require sufficiently larger amounts of training data and computational resources, both at training and test time. Furthermore, blind fusion of *fundamentally different* signals at early stages has a high risk of learning false cross-modality correlations and dependencies among them (see Section 4.7). In this project, we have demonstrated that in order to capture the complexity within each channel, isolated pretraining of modality-specific input layers and optimization of hyper parameters for each given subtask is required.

Let us once again recall Figure 42, which illustrates our developed architecture of a single-scale deep multi-modal convolutional network. Initially, it starts with six separate pathways: depth and intensity video channels for right (V1) and left (V2) hands, a mocap stream (M) and an audio stream (A).

From our observations, inter-modality fusion is effective at *early stages* if both channels are of the *same nature* and convey *complementary* information (rather than double it). On the other hand, mixing modalities which are *weakly correlated*, is rarely beneficial until the *final stage*. Accordingly, in our architecture, two video channels corresponding to the same hand (hidden layers HLV1 and HLV2) are fused immediately after the first steps of feature learning. We postpone any attempt to capture cross-modality correlations of complementary skeleton motion, hand articulation and audio (if present) until the shared layer HLS.

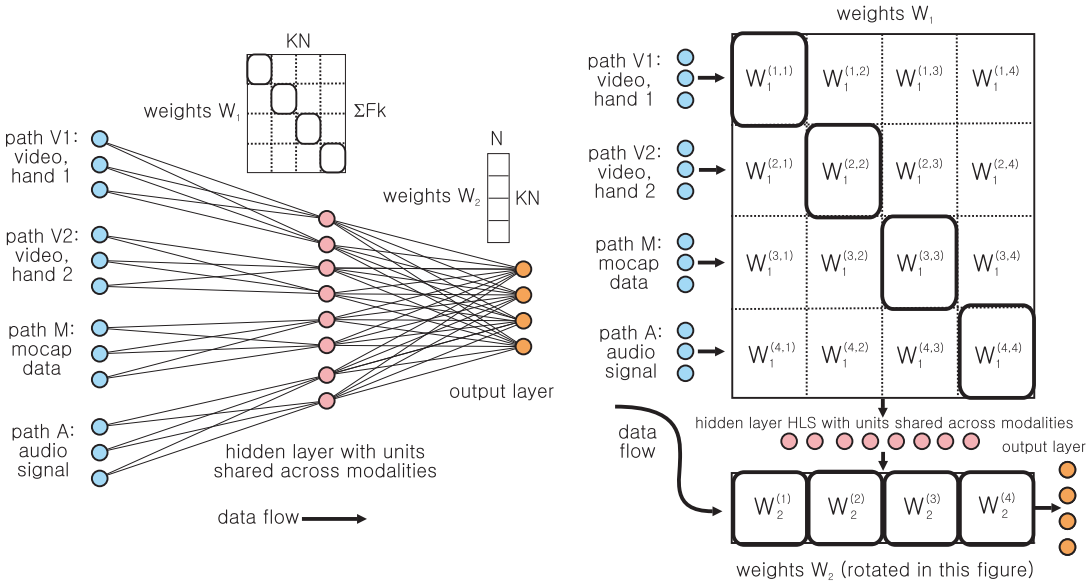


Figure 45 – Illustration of the proposed fusion strategy. On the left: architecture of shared hidden and output layers. Output hidden units on each path are first connected to a subset of neurons of the shared layer. During fusion, additional connections between paths and the shared hidden layer are added (not shown). On the right: structure of parameters of shared hidden and output layers (corresponds to the architecture above). Note that the image scale is chosen for clarity of description and the real aspect ratio between dimensions of the matrix W_1 is not preserved (it has 1600 rows and 84 columns), the ratio between vertical sizes of matrix blocks corresponding to different modalities is 9:9:7:7.

4.3.2 *Initialization of the fusion process*

Assuming that the weights of the modality-specific paths are pre-trained, the next important issue is determining a *fusion strategy*.

Pre-training solves some of the problems related to learning in deep networks with many parameters. However, direct fully-connected wiring of pre-trained paths to the shared layer in large-scale networks is not effective, as the high degrees of freedom afforded by the fusion process may lead to a quick degradation of pre-trained connections. We therefore proceed by initializing the shared layer such that a given hard-wired fusion strategy is performed, and then gradually open up to more powerful fusion strategies.

In a number of works [4], it has been shown that among fusion strategies, the *weighted arithmetic mean of per-model outputs* is a technique, which is the least sensitive to errors of individual classifiers and which is therefore often used in practice as an optimal solution outperforming more complex fusion algorithms.

Inspired by this idea, in our strategy, we consider the weighted mean as a *simple baseline* and aim to *initialize* the fusion process with this *starting point*, with the following intention to proceed with gradient descent optimization towards the minimum, in case if it is not reached.

Unfortunately, implementing the arithmetic mean in the case of early fusion and non-linear shared layers is not straightforward [108]. However, as it has been re-

cently demonstrated by Baldi and Sadowski [18], in dropout-like systems [127], activation units of complete models produce *weighted normalized geometric mean* of per-model outputs. This kind of average approximates the arithmetic mean closer than the simple geometric mean, and the quality of this approximation depends on consistency in the neuron activations.

Based on these observations, we initialize the fusion process to a normalized geometric mean of per-model outputs. For this purpose, data fusion is implemented at two different layers: the shared hidden layer (named HLS in Figure 42) and the output layer. The weight matrices of these two layers, denoted respectively as W_1 and W_2, are block-wise structured and initialized in a specific way. Figure 45 illustrates this situation. The left part of this figure shows the architecture in a conventional form as a diagram of connected neurons. The weights of the connections are indicated by matrices. On the right, we introduce a less conventional notation, which allows one to better visualize and interpret the block structure.

Let us denote the number of hidden units in the modality-specific hidden layers on each path as F_k, where k=1...K, and K is the number of modality-specific paths. We set the number of units of the shared hidden layer to be equal to K·N, where N is the number of target gesture classes in a given dataset (N=21).

As a consequence, the weight matrix W_1 of the shared hidden layer HLS is of size F×(N·K), where $F = \sum_k F_k$, and the weight matrix W_2 of the output layer is of size (N·K)×N. Weight matrix W_1 can be thought of as a matrix of K×K blocks, where each block k is of size F_k×N.

This structure imposes a certain meaning on the units and weights of the network. Each column in a block (and each unit in the shared layer) is therefore related to a specific gesture class. Let us stress here, that this block structure (and meaning) is *forced* on the weight matrix during the *initialization step* and in the *early phases* of training. If only the diagonal blocks are non-zero, which is forced at the beginning of the training procedure, then individual modalities are trained independently, and no cross correlations between modalities are modelled.

During the *final phases* of training, no structure is imposed and the weights can evolve freely. Considering the shared layer, this can be formalized by expressing activation of each hidden unit $h_l^{(k)}$ as follows:

$$h_l^{(k)} = \sigma\left[\sum_{i=1}^{F_k} w_{i,l}^{(k,k)} x_i^{(k)} + \gamma \sum_{\substack{m=1 \\ m\neq k}}^{K} \sum_{i=1}^{F_n} w_{i,l}^{(m,k)} x_i^{(m)} + b_l^{(k)}\right], \tag{84}$$

where $h_l^{(k)}$ is unit l initially related to modality k, and all weights w are from weight matrix W_1. Notation $w_{i,l}^{(m,k)}$ stands for a weight between non-shared hidden unit i from the output layer of modality channel m and the given shared hidden unit l related to modality k. Accordingly, $x_i^{(m)}$ is input number i from channel k. Finally, $b_l^{(k)}$ is a bias of the shared hidden unit $h_l^{(k)}$.

The first term of equation (84) contains the diagonal blocks, and the second term contains the off-diagonal weights. Setting parameter γ=0 freezes learning of the off-diagonal weights responsible for inter-modality correlations.

This initial meaning forced onto both weight matrices W_1 and W_2 produces a setting where the hidden layer is organized into K subsets of units $h_l^{(k)}$, one for each modality k, and where each subset comprises N units, one for each gesture class. The weight matrix W_2 is initialized in a way such that these units are interpreted as posterior probabilities for gesture classes, which are averaged over modalities by the output layer controlled by weight matrix W_2. In particular, each of the $N \times N$ blocks of the matrix W_2 (denoted as $v^{(k)}$) is initialized as an identity matrix, which results in the following expression for the softmax activated output units:

$$o_j = \frac{e^{\sum_{k=1}^{K} \sum_{c=1}^{N} v_{j,c}^{(k)} h_c^{(k)}}}{\sum_{i=1}^{N} e^{\sum_{k=1}^{K} \sum_{c=1}^{N} v_{i,c}^{(k)} h_c^{(k)}}} = \frac{e^{\sum_{k=1}^{K} h_j^{(k)}}}{\sum_{i=1}^{N} e^{\sum_{k=1}^{K} h_i^{(k)}}}, \tag{85}$$

where we used that $v_{j,c}^{(k)} = 1/K$ if $j=c$ and 0 otherwise.

From equation (85), the reader may see, that the diagonal initialization of W_2 forces the output layer to perform modality fusion as a normalized geometric mean over modalities, as motivated in the initial part of this section. Again, this setting is forced in the early stages of training and relaxed later, freeing the output layer to more complex fusion strategies.

4.3.3 *ModDrop: multimodal dropout*

Inspired by the concept of dropout as the normalized geometric mean of an exponential number of weakly trained models, in our multi-modal setting we furthermore aim on exploiting *a priori* information about groupings in the feature set. For that reason, we initiate a process similar to dropout, but with a *fixed number* of models corresponding to *separate modalities*, which are first pre-trained to convergence.

We have two main motivations for this scheme. First, we aim to learn a *shared model* while preserving *uniqueness* of per-channel features and avoiding false co-adaptations between modalities. Our second objective is to be able to *handle missing data* in one or more of the channels at test time. Therefore, the key idea is to train the shared model in a way that it would be capable of producing meaningful predictions from an arbitrary number of available modalities (with minimal loss in precision when one or more signals are missing).

To formalize this idea, let us consider a set of $\mathcal{M}_k$, $k=1 \ldots K$ modality-specific models. During pretraining, the joint learning objective can, in a general case, be formulated as follows:

$$\mathcal{L}_{\text{pretraining}} = \sum_{k=1}^{K} \mathcal{L}\left[\mathcal{M}^{(k)}\right] + \alpha \sum_{h=1}^{H} \|W_h\|^2, \tag{86}$$

where each term in the first sum represents a loss of the corresponding modality-specific model (in our case, negative log likelihood, summarized over all samples x_d for the given modality k from the training set $|\mathcal{D}|$):

$$\mathcal{L}\left[\mathcal{M}^{(k)}\right] = -\sum_{d=1}^{|\mathcal{D}|} \log o_Y^{(k)}(Y = y_d | x_d^{(k)}), \tag{87}$$

where $o_Y^{(k)}$ is output posterior probability distribution over classes of the network, corresponding to modality k, and y_d is a ground truth label for a given sample d.

The second term in equation (86) is L_2 regularization on all weights W_h from all hidden layers $h{=}1 \ldots H$ in the neural network (with corresponding coefficient α). At this pretraining stage, all loss terms in the first sum are minimized independently for each data channel.

Once the weight matrices W_1 and W_2 are initialized with pre-trained diagonal elements and initially zeroed out off-diagonal blocks of weights are relaxed (which corresponds to $\gamma{=}1$ in equation (84)), fusion is learned from training data. The desired training objective during the fusion process can be formulated as a combination of losses of all possible combinations of modality-specific models:

$$\mathcal{L}_\Sigma = \sum_{k=1}^{K} \mathcal{L}\left[\mathcal{M}^{(k)}\right] + \sum_{k\neq m} \mathcal{L}\left[\{\mathcal{M}^{(k)}, \mathcal{M}^{(m)}\right] + \sum_{k\neq m\neq n} \mathcal{L}\left[\{\mathcal{M}^{(k)}, \mathcal{M}^{(m)}, \mathcal{M}^{(n)}\}\right] + \ldots$$
$$+ \alpha \sum_{h=1}^{H} \|W_h\|^2 = \sum_{m=1}^{2^K} \mathcal{L}\left[S_m\right] + \alpha \sum_{h=1}^{H} \|W_h\|^2, \tag{88}$$

where notation $\{\}$ stands for fusion and S_m is an element of the power set of all models, which correspond to all possible combinations of modalities.

The loss function, which is formulated in equation (88), reflects the theoretical objective of the training procedure. In practice, however, we approximate this objective by the ModDrop process, i.e. as iterative interchangeable training of one term at a time.

In particular, the fusion process starts by joint training of the whole network, through back propagation over the shared layers and fine tuning all modality specific paths. As this step, the network takes as an input multi-modal training samples

$$\{\delta^{(k)} x_d^{(k)}\}, \quad k = 1 \ldots K, \tag{89}$$

from the training set $|\mathcal{D}|$, where for each sample each modality component $x_d^{(k)}$ is dropped (set to 0) with a certain probability:

$$q^{(k)} = 1 - p^{(k)}. \tag{90}$$

In the last expression, $p^{(k)}$ is indicated by Bernoulli selector:

$$\delta^{(k)} : P(\delta^{(k)}{=}1) = p^{(k)}. \tag{91}$$

Accordingly, one step of gradient descent, given an input with a certain number of non-zero modality components, minimizes the loss of a corresponding multi-modal subnetwork denoted as $\{\delta^{(k)}\mathcal{M}^{(k)}\}$. This idea aligns well with the initialization process, described above, which ensures that modality-specific subnetworks that are being removed or added by ModDrop are well pre-trained in advance.

4.3.4 *Analysis of regularization properties*

In the following discussion, we will study the *regularization properties* of modality-wise dropout on inputs (ModDrop) on a simpler network architecture, namely a

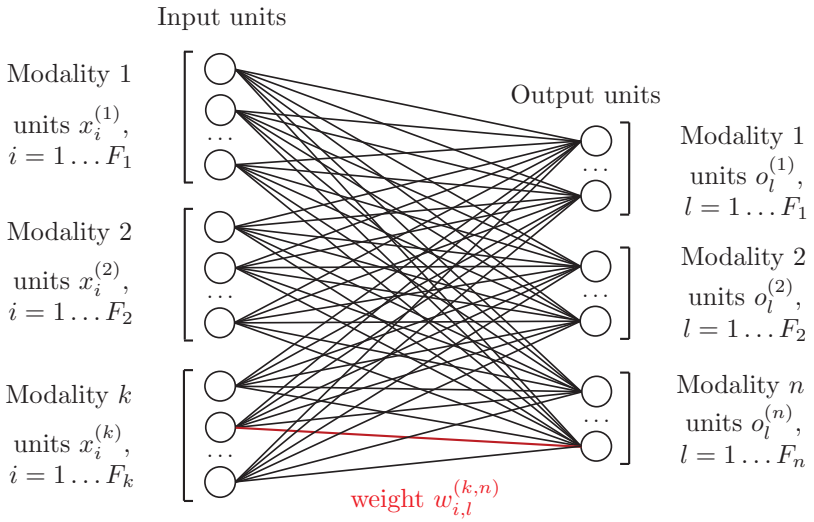


Figure 46 – Toy network architecture and corresponding notations, which are used for derivation of ModDrop regularization properties.

one-layer shared network with K modality specific paths and sigmoid activation units (illustrated in Figure 46). The obtained results can be then extended to the case of more general deep learning architectures.

In this case, input i for modality k is denoted as $x_i^{(k)}$, and we assume that there are F_k inputs, which are coming from each modality k (see Figure 46). Output unit l, which is related to modality n, is denoted as $o_l^{(n)}$. Finally, a weight coefficient connecting input unit $x_i^{(k)}$ with output unit $o_l^{(n)}$ is denoted as $w_{i,l}^{(k,n)}$.

In our example, all output units are sigmoidal, i.e. each output unit o_l, which is related to modality n, is calculated as follows:

$$o_l^{(n)} = \sigma(s_l^{(n)}) = \frac{1}{1 + e^{-\lambda s_l^{(n)}}}, \tag{92}$$

where λ is a coefficient and $s_l^{(n)}$ is the input to the activation function, which is coming to the given output unit from the previous layer:

$$s_l^{(n)} = \sum_{k=1}^{K} \sum_{i}^{F_k} w_{i,l}^{(k,n)} x_i^{(k)}. \tag{93}$$

During training, we minimize cross-entropy error, which is calculated from the targets y (indices are dropped for simplicity):

$$E = -(y \log o + (1 - y) \log (1 - o)), \tag{94}$$

whose partial derivatives can be given as follows:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial o} \frac{\partial o}{\partial s} \frac{\partial s}{\partial w}, \tag{95}$$

where each of the three multipliers is calculated as:

$$\frac{\partial E}{\partial o} = -y \frac{1}{o} + (1 - o) \frac{1}{1 - o}, \quad \frac{\partial o}{\partial s} = \lambda o (1 - o).$$

$$\frac{\partial E}{\partial w} = -\lambda(y - o)\frac{\partial s}{\partial w}. \tag{96}$$

Along the lines of [18], we consider two situations corresponding to two different loss functions: E_Σ, corresponding to the *complete network*, where all modalities are present, and E_{drop} where *ModDrop* is performed. In our case, we assume that whole modalities (sets of units corresponding to a given modality k) are either dropped (with probability $q^{(k)} = 1 - p^{(k)}$) or preserved (with probability $p^{(k)}$).

In a ModDrop network, this can be formulated such that the input to the activation function of a given output unit l related to modality n (denoted as $\tilde{s}_l^{(n)}$) involves a Bernoulli selector variable $\delta^{(k)}$ for each modality k which can take on values in $\{0, 1\}$ and is activated with probablity $p^{(k)}$:

$$\tilde{s}_l^{(n)} = \sum_{k=1}^{K} \delta^{(k)} \sum_{i=1}^{F_k} w_{i,l}^{(k,n)} x_i^{(k)}. \tag{97}$$

As a reminder, in the case of the complete network (where none of the channels is dropped) the output activation is the following:

$$s_l^{(n)} = \sum_{k=1}^{K} \sum_{i=1}^{F_k} w_{i,l}^{(k,n)} x_i^{(k)}. \tag{98}$$

As the following reasoning always concerns a single output unit l related to modality n, from now on these indices will be dropped for simplicity of notation. Therefore, we denote $s = s_l^{(n)}$, $\tilde{s} = \tilde{s}_l^{(n)}$ and $w_i^{(k)} = w_{i,l}^{(k,n)}$.

Gradients of the corresponding complete and ModDrop sums with respect to the weights can be expressed as follows:

$$\frac{\partial \tilde{s}}{\partial w_i^{(k)}} = \delta^{(k)} x_i^{(k)}, \quad \frac{\partial s}{\partial w_i^{(k)}} = x_i^{(k)}. \tag{99}$$

Using the gradient of the error E

$$\frac{\partial E}{\partial w_i^{(k)}} = -\lambda \left[y - \sigma(s) \right] \frac{\partial s}{\partial w_i^{(k)}}, \tag{100}$$

we can derive the gradient of the error for the complete network:

$$\frac{\partial E_\Sigma}{\partial w_i^{(k)}} = -\lambda \left[y - \sigma\left(\sum_{m=1}^{K} \sum_{j=1}^{F_m} w_j^{(m)} x_j^{(m)} \right) \right] x_i^{(k)}. \tag{101}$$

In the case of ModDrop, for one realization of the network, where a modality is dropped with corresponding probability $q^{(k)} = 1 - p^{(k)}$, indicated by the means of Bernoulli selectors $\delta^{(k)}$, i.e. $P(\delta^{(k)} = 1) = p^{(k)}$, we get:

$$\frac{\partial E_{drop}}{\partial w_i^{(k)}} = -\lambda \left[y - \sigma\left(\sum_{m=1}^{K} \delta^{(m)} \sum_{j=1}^{F_m} w_j^{(m)} x_j^{(m)} \right) \right] \delta^{(k)} x_i^{(k)}. \tag{102}$$

Taking the expectation of this expression requires an expression introduced in [18], which approximates $E[\sigma(x)]$ by $\sigma(E[x])$. We take the expectation over the $\delta^{(m)}$ with

the exception of $\delta^{(k)}=1$, which is the Bernoulli selector of the modality k for which the derivative is calculated, giving:

$$
\begin{aligned}
E\left[\frac{\partial E_{drop}}{\partial w_i^{(k)}}\right] &\approx -\lambda\left[y-\sigma\left(\sum_{m\neq k}^{K} p^{(m)}\sum_{j=1}^{F_m} w_j^{(m)}x_j^{(m)}+\sum_{j=1}^{F_k} w_j^{(k)}x_j^{(k)}\right)\right]p^{(k)}x_i^{(k)},\\
E\left[\frac{\partial E_{drop}}{\partial w_i^{(k)}}\right] &= -\lambda\left[y-\sigma\left(\sum_{m\neq k}^{K}\sum_{j=1}^{F_m} w_j^{(m)}x_j^{(m)}-\right.\right.\\
&\qquad \left.\left.-\sum_{m\neq k}^{K}(1-p^{(m)})\sum_{j=1}^{F_m} w_j^{(m)}x_j^{(m)}+\sum_{j=1}^{F_k} w_j^{(k)}x_j^{(k)}\right)\right]p^{(k)}x_i^{(k)},\\
E\left[\frac{\partial E_{drop}}{\partial w_i^{(k)}}\right] &= -\lambda\left[y-\sigma\left(\sum_{m=1}^{K}\sum_{j=1}^{F_m} w_j^{(m)}x_j^{(m)}-\right.\right.\\
&\qquad \left.\left.-\sum_{m\neq k}^{K}(1-p^{(m)})\sum_{j=1}^{F_m} w_j^{(m)}x_j^{(m)}\right)\right]p^{(k)}x_i^{(k)}.
\end{aligned}
\tag{103}
$$

Taking the first-order Taylor expansion of the activation function σ around the point $s=\sum_m\sum_j w_j^{(m)}x_j^{(m)}$ gives:

$$
E\left[\frac{\partial E_{drop}}{\partial w_i^{(k)}}\right]\approx -\lambda\left[,y-\sigma(s)+\sigma'(s)\left(\sum_{m\neq k}^{K}(1-p^{(m)})\sum_{j=1}^{F_m} w_j^{(m)}x_j^{(m)}\right)\right]p^{(k)}x_i^{(k)},
\tag{104}
$$

where $\sigma'(s)=\sigma(s)/(1-\sigma(s))$. Plugging in equation (101), we get:

$$
E\left[\frac{\partial E_{drop}}{\partial w_i^{(k)}}\right]\approx p^{(k)}\frac{\partial E_\Sigma}{\partial w_i^{(k)}}-\lambda\sigma'(s)x_i^{(k)}p^{(k)}\sum_{m\neq k}^{K}(1-p^{(m)})\sum_{j=1}^{F_k} w_j^{(m)}x_j^{(m)}.
\tag{105}
$$

If $p^{(k)}=p^{(m)}=p$, then $p(1-p)=\mathrm{Var}(\delta)$. From the gradient, we can calculate the error E integrating out the partial derivatives and summing over the weights i:

$$
E\approx pE_\Sigma-\lambda\sigma'(s)\mathrm{Var}(\delta)\sum_{k=1}^{K}\sum_{m\neq k}^{K}\sum_{i=1}^{F_k}\sum_{j=1}^{F_m} w_i^{(k)}w_j^{(m)}x_i^{(k)}x_j^{(m)}.
\tag{106}
$$

As it can be seen from the final expression (106), the error of the network with ModDrop is approximately equal to the error of the complete model (up to a coefficient) minus an additional term including a sum of products of inputs and weights corresponding to different modalities in all possible combinations. We need to stress here that this second term reflects exclusively cross-modality correlations and does not involve multiplications of inputs coming from the same channel.

To understand the influence of the cross-product term on the training process, we analyze two extreme cases depending on whether or not the signals in different channels are correlated.

Let us consider two input units $x_i^{(k)}$ and $x_j^{(m)}$, which come from different modalities, and first assume that they are independent. As a rule, each network input is preliminary normalized to zero mean and unit variance, therefore the expectation of a product of two independent inputs is also equal to zero:

$$E\left[x_i^{(k)} x_j^{(m)}\right] = E\left[x_i^{(k)}\right] E\left[x_j^{(m)}\right] = 0. \tag{107}$$

Furthermore, weights in a single layer of a neural network typically obey a unimodal distribution with zero expectation [268]. It can be shown [174], that under these assumptions, Lyapunov's condition is satisfied, and therefore Lyapunov's central mean theorem allows us to make a conclusion, that in this case, the sum of products of inputs and weights will tend to a normal distribution, given that the number of training samples is sufficiently large. As both input and weight distribution have zero mean, the resulting law is also centralized, and its variance is defined by magnitudes of weights (assuming inputs to be fixed).

The important conclusion from the reasoning above is that assuming independence of inputs in different channels, the second term in equation (106) tends to vanish if the number of training samples in a batch is sufficiently large. In practice, it shows that additional regularization on weights needs to be added to the error function to prevent weights from exploding.

Now let us consider a more interesting scenario, when two inputs $x_i^{(k)}$ and $x_j^{(m)}$, belonging to different modalities, are positively correlated. In this case, given zero mean of distribution of each input, their product is expected to be positive:

$$E\left[x_i^{(k)} x_j^{(m)}\right] = E\left[x_i^{(k)}\right] E\left[x_j^{(m)}\right] + \mathrm{Cov}\left[x_i^{(k)}, x_j^{(k)}\right]. \tag{108}$$

Therefore, on each step of gradient descent, this term enforces the product of weights $w_i^{(k)} w_j^{(m)}$ to be positive and therefore introduces correlations between them (given, again, the additional regularization term preventing one of multipliers from growing significantly faster than the other). The same logic applies if inputs are negatively correlated, which would enforce negative correlations on corresponding weights. Accordingly, for correlated modalities, this additional term in the error function, introduced by ModDrop, acts as a cross-modality regularizer, which forces the network to generalize by discovering similarities between different signals and *aligning* them with each other by introducing constraints on corresponding weights.

Finally, as it has been shown by [18] for the case of dropout, the multiplier, which is proportional to the derivative of sigmoid activation, makes the regularization effect adaptive to the magnitude of the weights. As a result, it is strong in the mid-range of weights, plays less significant of a role when weights are small and gradually weakens with saturation of the neuron.

Our experiments have shown that ModDrop achieves the best results if combined with dropout, which introduces an adaptive L2 regularization term $\hat{E}$ in the error function [18]:

$$\hat{E} \approx \lambda \sigma'(s) \mathrm{Var}(\hat{\delta}) \sum_{k=1}^{K} \sum_{i=1}^{F_k} (w_i^{(k)} x_i^{(k)})^2, \tag{109}$$

where $\hat{\delta}$ is a Bernoulli selector, such that $P(\hat{\delta}{=}1){=}\hat{p}$, and $\hat{p}$ is a probability of a given input unit to be present in the given network realization.

4.4 ITERATIVE FUSION: PRACTICAL RECOMMENDATIONS

In this section, we will give a number of *practical recommendations* on how the efficiency of our proposed iterative fusion strategy can be further improved, depending on the nature of modalities which are being fused. The points described here summarize our experience gained from long hours, which were spent on optimization of the training process in this multi-modal framework (using a validation set). A reader who is not interested in such nuances, is invited to skip this discussion and proceed to the experimental Section 4.7.

As we have seen in the previous sections, proper initialization of the shared layer before fusion is important. Otherwise, direct fully connected wiring of pre-trained paths to the shared layer with randomly initialized weights leads to quick degredation of pre-trained connections and as a result, our experience suggests, that the joint representation performs worse than a linear combination of the predictions of individual classifiers.

However, even in the case when all diagonal blocks in the shared weight matrix are pre-initialized and out of diagonal blocks are zeroed out and relaxed, the network still may struggle learning cross-modality correlations immediately. From our observations, this is especially the case, when architectures of each modality-specific paths are not perfectly optimized, and therefore perfomance of each individual per-modality classifier alone is relatively low.

To address this issue and to stimulate learning on this fusion stage, one possible strategy would be to feed a classifier with training data arranged in a specific meaningful order, starting from clean samples, that are easy to classify, and proceeding to the most complex ones, allowing the network to learn more and more sophisticated concepts. In the machine learning community, this idea is widely known under the name of *curriculum learning* [24]. Generally, this approach has shown to yield better generalization in less time. Alternatively, the network itself can be changed in an iterative way, evolving from a weak prediction model to more and more complex prediction models. In this section, we employ the latter strategy, which we will first explain intuitively, and then formalize.

In this sense, our per-modality pre-training step, when the network itself is divided into meaningful parts, that are adjusted separately and then combined, can be considered as the first step of this model growing process. Here, we begin training by presenting modality-specific parts of the network with samples, where only one modality is present. As we described earlier, in this way, we pre-train initial sets of modality-specific layers that extract features from each data channel and create more meaningful and compact data representations.

Once pre-training is completed, it is practically beneficial to proceed with integrating all channels, one by one, in an *iterative manner*. At this step, we choose the order of modalities in a specific way, in order to first combine the data where the *strongest* cross-modality structure is expected. This permits the model to gradually and effectively learn a joint distribution, focusing its representational power on where it is most effective, while keeping the input compact and the number of parameters relatively small. In the task of multi-modal gesture recognition, the audio stream and

articulated pose alone convey sufficient information about the gesture, i.e. recognition can be performed reasonably well from each channel independently. However, data in the two video channels, representing the articulation of each of the two hands, is complementary for the skeleton classifier and can improve its accuracy.

To ensure that the joint model is meaningful, both the shared representation layer and output layer are first configured to produce an an averaged combination of individual modalities (as described in Section 4.3.2). The network parameters are further optimized starting from this initialization. We start the fusion procedure by integrating two highly dependent video channels (V1 and V2) with shared parameters, then add the third visual modality (articulated pose, Path M) and finally connect the audio Path A (see Figure 42).

The whole training-fusion process, which increases the power of the prediction model iteratively by adding connections, will now be presented in a more formal way. For clarity and ease of notation, the connectivity of the network will stay constant over all iterations of the training procedure. In practice, a subset of weights will be clamped to zero initially; these weights will then be freed iteratively and allowed to be trained.

For this following discussion, we will once again exploit the network representation and notations introduced in Figure 45.

The training procedure starts by pre-training each modality separately. This is equivalent to initializing the block diagonal elements of the matrix W_1 with blocks $W_1^{(1,1)}$, $W_1^{(2,2)}$, $W_1^{(3,3)}$ and $W_1^{(4,4)}$, containing sets of weights corresponding to different modalities:

$$W_1 = \begin{pmatrix} W_1^{(1,1)} & 0 & 0 & 0 \\ 0 & W_1^{(2,2)} & 0 & 0 \\ 0 & 0 & W_1^{(3,3)} & 0 \\ 0 & 0 & 0 & W_1^{(4,4)} \end{pmatrix}. \tag{110}$$

Accordingly, the set of biases b_1 can be represented as a concatenation of k modality-specific vectors:

$$b_1 = \left[b_1^{(1)}, b_1^{(2)}, b_1^{(3)}, b_1^{(4)} \right]^T. \tag{111}$$

The weights of the output layer, W_2, are initialized as a concatenation of K identity matrices of size $N \times N$, $W_2 = [I_N, I_N, \ldots, I_N]^T$. Bias elements b_2 are set to 0. This setting can be interpreted as follows: (i) The structure of W_1 forces each of the K modalities to be represented by N units in the shared hidden layer, one for each gesture class. Columns in the weight matrix are *assigned* to modality specific blocks, and to gesture classes within each block. (ii) The diagonal/identity structure of matrix W_2 forces the output layer to create a sum over modalities. This restriction of W_2 corresponds to late fusion with constant identical weights. The prediction P_n for class n can be written as follows:

$$P_n = \text{softmax}\left[\sum_{k=0}^{K-1} I_N \cdot \text{ReLU}\left(\sum_{i=1}^{F} U_i W_{1,i,(kN+n)} + b_{1,(kN+n)} \right) \right], \tag{112}$$

where $\{u_i\}$, $i = 1 \dots F$ are neuron activations from the last hidden layer of each path and the notation ReLU stands for rectified linear units.

During pre-training, only one path is connected (i.e. all elements of W_1 except for the corresponding block $W_1^{(k)}$ are set to zero) and weights and biases of the output layer are fixed. The shared layer consists of rectified linear units so the combination of the shared hidden and output layer works as a simple logistic regression unit for the given channel. On this step, parameters of all modality-specific convolutional and hidden layers belonging to the given path are trained by gradient descent, as well as the weights and biases of the corresponding subset of neurons in the shared layer.

The detailed training procedure, step by step, is shown on Fig. 47. Green blocks in W_1 show weights which are *not* clamped to zero, while gray blocks show weights clamped to zero. The first three steps (top row) illustrate pre-training of 3 channels corresponding to the intensity-depth, articulated pose, and audio streams. Since the weights of paths V1 and V2 are tied, in this case the network is trained once on a combined training set containing blocks corresponding to both hands. Then the pre-trained connections are copied from one path to another and coefficients of matrix W_2 are normalized. This is denoted as:

$$W_1 = \begin{pmatrix} W_1^{(1,1)} & 0 & 0 & 0 \\ 0 & W_1^{(2,2)} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{113}$$

$$W_2 = [\alpha_d \cdot I_N, (1 - \alpha_d) \cdot I_N, 0, 0]^T, \tag{114}$$

where $W_1^{(1,1)}$=$W_1^{(2,2)}$ and the zero entries in W_2 are zero-valued matrices. Here, the hyper-parameter $\alpha_d = 0.6$ reflects the fact that there is a greater variety of training data for the right hand in the dataset and predictions obtained from the first path are more reliable.

After training all paths separately, additional connections between output hidden layers on each path are added, and at the end the shared hidden layer becomes fully connected. Returning to the matrix W_1, it signifies initialization out of diagonal blocks, representing the structure among individual modalities. The block at position (k_1, k_2), for example, contains weights representing interactions between modalities k_1 and k_2.

From this moment on, all parameters of the preceding modality-specific layers are kept fixed. We start the integration procedure with two video channels (step 4 on Figure 47). In this case, the matrix W_1 is masked for updating 4 blocks: $W_1^{(1,1)}$, $W_1^{(2,2)}$, $W_1^{(1,2)}$ and $W_1^{(2,1)}$. The audio and articulated pose channels are disconnected and the corresponding parameters of the weight matrix W_1 and biases b_1 are set to 0. Whereas previously, a hand-specific classifier was trained for distinguishing individual hand movements, on this step *gesture labels* are used as a ground truth and a *combination* of two hand movements is classified. On this and the following steps, corresponding blocks of the weight matrix W_2 (and biases b_2) are initialized as identity matrices (zeros) and trained.

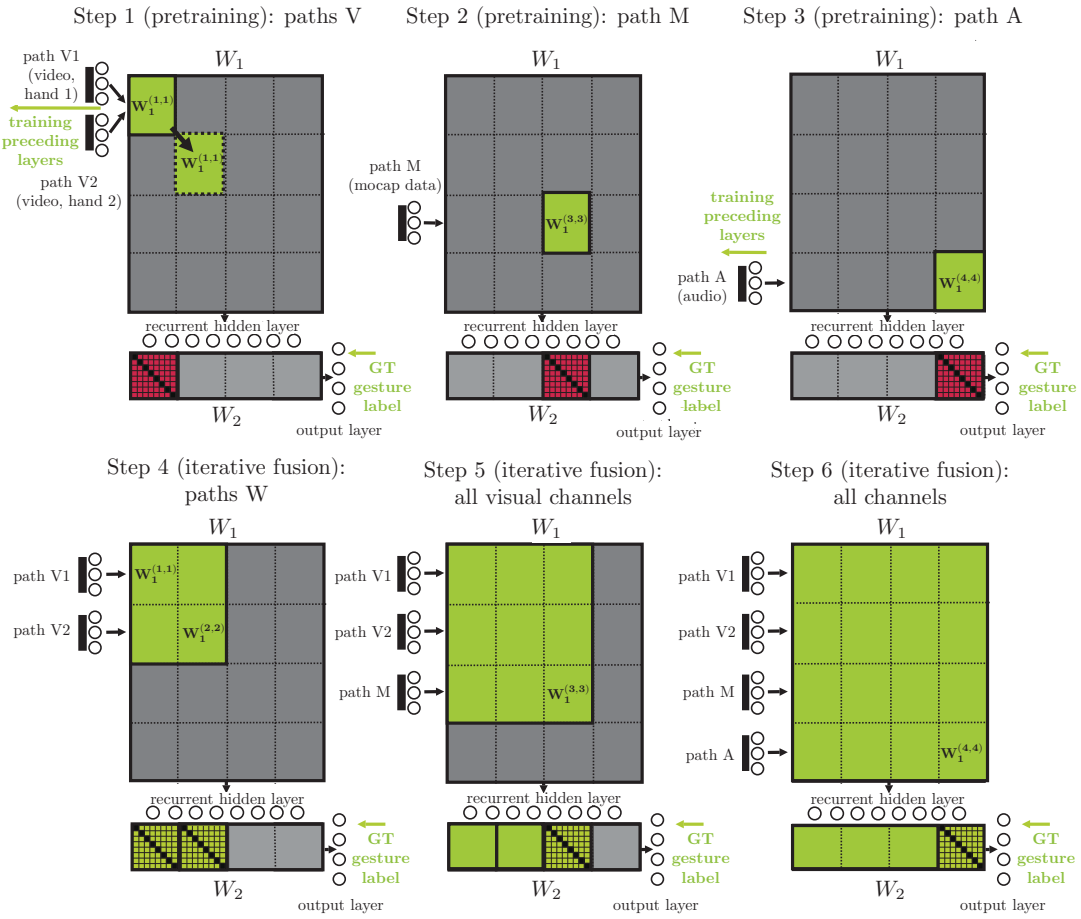


Figure 47 – Iterative training procedure. The areas shown in red correspond to parameters which are set but kept fixed during the current iteration. The grean color signifies that the corresponding parameters are set and updated. Gray areas are masked, i.e. the corresponding blocks are set to 0. The first raw: one-by-one pretraining of the individual channels. The second raw: learning inter-modality dependencies and final tuning.

After combining two video channels, articulated pose is added, and the mask on the weight matrix W_1 is extended to the size of 3×3 blocks. The third block of the matrix W_2 is initialized with an identity matrix and updated on each training iteration. This is denoted as:

$$W_1 = \begin{pmatrix} W_1^{(1,1)} & W_1^{(1,2)} & 0 & 0 \\ W_1^{(1,2)} & W_1^{(2,2)} & 0 & 0 \\ 0 & 0 & W_1^{(3,3)} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{115}$$

$$W_2 = [(1-\alpha_m)W_2^{(1)}, (1-\alpha_m)W_2^{(2)}, \alpha_m I_N, 0], \tag{116}$$

where, as before, α_m is an empirical coefficient balancing initial contributions of modalities which are being fused.

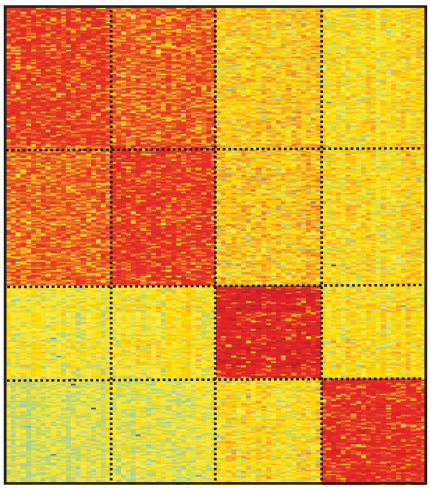


Figure 48 – Energy structure of weights W_1 after iterative training. Diagonal blocks are dominated by individual modalities (right and left hands, articulated pose and audio) out of diagonal elements reflect cross-modality correlations. As the reader may see, the energy corresponding to video-skeleton crossterms is significantly higher than on the step of visual-audio fusion.

The same procedure is finally repeated for the audio channel, where all feedforward weights and biases of the shared layers are once again updated and interactions among the visual and audio data are learned. The weights are now:

$$W_1 = \begin{pmatrix} W_1^{(1,1)} & W_1^{(1,2)} & W_1^{(1,3)} & 0 \\ W_1^{(2,1)} & W_1^{(2,2)} & W_1^{(2,3)} & 0 \\ W_1^{(3,1)} & W_1^{(3,2)} & W_1^{(3,3)} & 0 \\ 0 & 0 & 0 & W_1^{(4,4)} \end{pmatrix}, \tag{117}$$

$$W_2 = [(1-\alpha_a)W_2^{(1)}, (1-\alpha_a)W_2^{(2)}, (1-\alpha_a)W_2^{(3)}, \alpha_a I_N]. \tag{118}$$

Normalization coefficients α_m and α_a are set proportionally to the performance of individual modalities in the sole modality classification task (α_m=0.7, α_s=0.3).

The resulting weight matrix W_1 after all training steps described above is shown in Figure 48. The most discriminative articulated pose (third diagonal block) is highlighted. We can see that interacitons between the two video channels are especially strong, while visual and audio dependencies are not that pronounced.

Finally, we would like to stress once again, that for the case, when each of the modality-specific classifiers can ensure relatively strong performance, the significance of the *iterative* character of the fusion process decreases. Our experiments have shown, that in this case, training converges to approximately the same validation rates (even though it may take longer to converge, if all modalities are fused at once).

In practice, however, multi-modal systems can be built from a great number of *weak classifiers*, in which case introducing additional priors on the expected energy of cross-modality correlations, by adjusting the training routine, can be highly beneficial. We have observed such behavior in the same gesture recognition framework

in our earlier experiments, when architectures of all modality-specific paths were not yet sufficiently optimized, and proper ordering of fused channels turned out to be crucial to boost network performance.

4.4.1 *Additional remarks on introducing recurrent connections*

Discussing practical recommendations for training, we have considered, so far, exclusively inter-modality dependencies, staying at the level of classification of short spatio-temporal blocks (dynamic poses). In our ealier research, however, we considered introducing in this multi-modal framework an explicit *modeling of temporal dependencies*, by adding additional *recurrent connections* to the shared layer, denoted as HLS in Figure 42.

This architecture was implemented in our submission for the *Chalearn 2013* gesture recognition challenge. Even though in the following year, in our *Chalearn 2014* submission, the recurrent elements were replaced by multi-scale late fusion (for practical reasons), we describe our experience with training the multi-modal RNN for completeness.

In these experiments, for integrating the data over time and capturing temporal dependencies, we introduced additional recurrent connections to the shared layer, which were trained at the last step, once modality fusion was completed. During training, feedforward connections were renormalized, to avoid saturation, and kept fixed.

In this case, to facilitate the training, the recurrent weights W_r were first initialized to an identity matrix $I_{N\cdot K}$. This was done following the same logic, as in our initialization of feedforward ways, and served for the sake of initiating the fusion process starting from a meaningful *averaging point*. This idea of initialization of recurrent connections with identity matrices was later proposed, developed and analyzed in a paper by Le et al. [170].

Since we assume that all gestures are independent from each other and their order is randomized, incorporating really long-term dependencies in the model is not beneficial and even harmful. Therefore, for RNN training and testing we did not use a continuous data stream, but rather split the input into sequences of a length which roughly corresponded to the duration of a typical gesture.

4.5 INTER-SCALE FUSION AT TEST TIME

Finally, we need to return to our particular application, in order to describe how gesture recognition is implemented at test time, considering the proposed multi-scale feedforward architecture.

Once individual single-scale predictions are obtained from each multi-modal classifier, we employ a simple voting strategy for fusing the output probabilities with a single weight per model. We would like to note here, that introducing additional per-class per-model weights and training meta-classifiers (such as an MLP) on this step quickly leads to overfitting.

Accordingly, at each given frame t, per-class network outputs o_k are obtained via per-frame aggregation and temporal filtering of predictions at each scale with

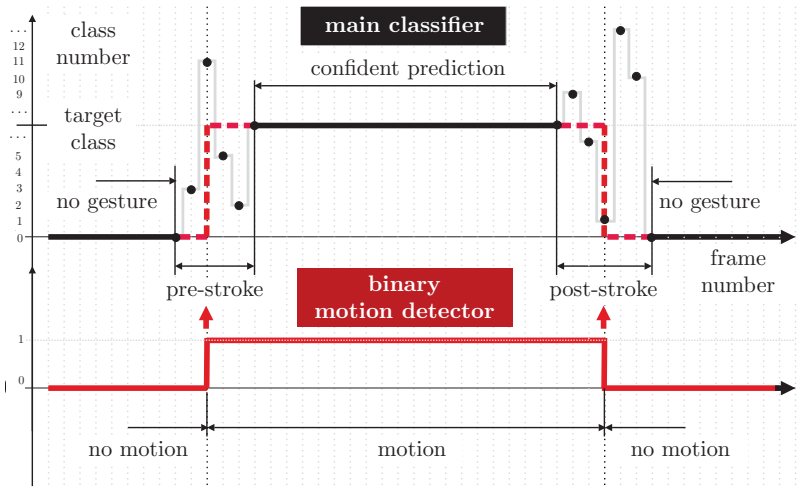


Figure 49 – Gesture localization. Top: output predictions of the main classifier; Bottom: output of the binary motion detector. Noise at pre-stroke and post-stroke phases in the first case is due to high similarity between gesture classes at these time periods and temporal inertia of the classifier.

corresponding weights μ_s, which are defined empirically through cross-validation on a validation set:

$$o_k(t) = \sum_{s=2}^{4} \mu_s \sum_{j=-4s}^{0} o_{s,k}(t+j), \tag{119}$$

where $o_{s,k}(t+j)$ is the score of class k obtained for a spatio-temporal block sampled starting from the frame $t+j$ at step s.

Finally, each frame is assigned the class label $l(t)$, which has the maximum value of the posterior probability, integrated over all relevant dynamic poses and all temporal scales:

$$l(t) = \underset{k}{\operatorname{argmax}}\, o_k(t). \tag{120}$$

The same logic can be applied in the case when temporal models are used, by simplifying equation (119) for a single temporal scale.

4.6 GESTURE LOCALIZATION

With increasing duration of a dynamic pose, *recognition* rates of the main gesture classifier increase at a cost of loss in precision in gesture *localization*. Using wider sliding windows leads to noisy predictions at pre-stroke and post-stroke phases, in some cases overlapping several gesture instances at once. On the other hand, too short dynamic poses are not discriminative either, as most gesture classes at their initial and final stages have a similar appearance (e.g. raising or lowering the hands).

To address this issue, we introduce an additional binary classifier, in order to distinguish *resting moments* from *periods of activity*. Trained on dynamic poses at the finest temporal resolution $s=1$, this classifier is able to precisely localize starting and ending points of each gesture.

The module is implemented based on the same articulated pose descriptor, which is fed to an MLP. All training frames, labeled with some gesture class, are used as positive examples, while a set of frames *right before* and *after* each gesture are considered as negatives. This strategy allows us to assign each frame with a label *motion* or *no motion* with accuracy of 98%.

Mapping from block-wise to frame-wise binary predictions for the motion detector is performed in the same fashion, as for the main classifier, using equations (119) and (120). The resulting stream of predictions is a smooth stepwise function, where switching from 0 to 1 indicates a beginning of a gesture, and switching from 1 to 0 marks its ending point (see the graph in the bottom of Fig. 49).

To combine the classification and localization modules, frame-wise gesture class predictions are first obtained as described in Section 4.5. At this stage, the output predictions at the beginning and at the end of each gesture are typically noisy (illustrated by the top curve at Fig. 49).

During the localization refinement, for each gesture, spotted by the main classifier, its boundaries are extended or shrunk towards the closest switching point produced by the binary classifier (given a certain range $-\tau \dots \tau$, in which the gesture boundaries are allowed to be adjusted).

4.7 EXPERIMENTAL RESULTS

In this section, we will discuss and analyze empirical performance of the proposed gesture recognition strategy, including each of the single channel classifiers as well as the whole multi-modal setup.

As we have mentioned earlier, the results presented in the chapter, correspond to our submission entries for two gesture recognition competitions, which were held in 2013 and 2014. The two different architectures will be presented in *reverse* chronological order, starting with submission from 2014, after which we will mention key observations from the earlier work, which we believe were important. Generally, the more recent entry corresponds to a better optimized architecture, in which some elements of its predecessor, however, were removed (such as recurrent connections of the shared layer).

Both *ChaLearn 2014 Looking at People Challenge (track 3)* and *ChaLearn 2013 Multi-modal Gesture recognition* datasets consist of 13,858 instances of Italian conversational gestures performed by different people and recorded with a consumer RGB-D sensor. In fact, the 2014 version represents an updated release of the same data corpus, as in 2013, in which previously noisy annotations were manually corrected, and the training and test sets were reshuffled. This version includes color, depth video and articulated pose streams, while the earlier release also included audio recordings.

In both datasets, gestures are drawn from a large vocabulary, from which 20 categories are identified to detect and recognize, and the rest are arbitrary gestures and movements [87]. Training data is accompanied by a ground truth label for each gesture, as well as information about its starting and ending points. For these challenges, the corpus was split into development, validation and test set, where the test data has been released to participants after submitting their codes.

To further explore dynamics of learning in multi-modal systems, using a cleaner ChaLearn 2014 version, we also augment this data with audio recordings extracted from its previous release. As a result, in both versions of the dataset, each gesture in a video sequence is accompanied with a corresponding vocal phrase bearing the same meaning. Due to dialectical and personal differences in pronunciation and vocabulary, gesture recognition from the audio channel alone appeared to be surprisingly challenging.

To summarize, we report results on several different versions of the dataset:

— the original version used for the *ChaLearn 2014 Looking at People Challenge* (LaP) (track 3, dedicated to gesture recognition),

— an extended version of the *ChaLearn 2014 LaP* dataset, augmented with audio recordings taken from *ChaLearn 2013 Multi-modal Gesture Recognition* dataset,

— the original version of the data used for the *ChaLearn 2013 Multi-modal Gesture Recognition Challenge*.

4.7.1 *Evaluation metrics*

Performance of all participants in both scientific competitions was evaluated based on a number standardized metrics. While in the 2013 version of the dataset, the objective of the challenge was to solely *detect* and *classify* a gesture, the more recent version also required accurate gesture *localization*.

— **ChaLearn 2014**. In these experiments, we followed the evaluation procedure proposed by the challenge organizers and adopted the *Jaccard Index* to quantify model performance:

$$J_{s,n} = \frac{A_{s,n} \cap B_{s,n}}{A_{s,n} \cup B_{s,n}}, \tag{121}$$

where $A_{s,n}$ is the ground truth label of gesture n in sequence s, and $B_{s,n}$ is the obtained prediction for the given gesture class in the same sequence. Here $A_{s,n}$ and $B_{s,n}$ are binary vectors where the frames in which the given gesture is being performed are marked with 1 and the rest with 0. Overall performance was calculated as the mean Jaccard index among all gesture categories and all sequences, with equal weights for all gesture classes.

— **ChaLearn 2013**. Following the methodology originally proposed by the challenge organizers for this earlier competition, we evaluate the performance as the *edit distance (ED)*, also known as *Levenshtein distance*, between an ordered sequence of gestures recognized by the system and the ground truth. Here, one sequence corresponds to one video from the dataset. This metric is calculated as a number of uniformly penalized edit operations (substitution, insertion, deletion) necessary to transform one sequence into another. The overall score is a sum of the edit distances over the whole test set divided by the real number of gesture instances.

Finally, recall, precision or a per-dynamic pose classification accuracy may be reported in some of the experiments, when appropriate.

4.7.2 *Baseline models*

For the sake of self-assessment and more fair comparison of the proposed deep learning strategy with a close *traditional* analog, we have implemented a baseline model, which is based on an ensemble classifier trained in a similar iterative fashion, but on purely handcrafted descriptors. It was done to explore the relative advantages (and disadvantages) of using learned representations and also the nuances of fusion. In addition, due to differences in feature formulation as well as in the nature of classifiers, we found it beneficial [2] to combine the proposed deep network with the baseline method in a hybrid model as separately two models make different errors (see Table 5).

A baseline has also been created for the audio channel, where we compare the proposed deep learning approach with traditional phoneme recognition framework in the given context.

4.7.2.1 *Baseline visual models*

We use depth and intensity hand images and extract three sets of features. HoG features describe the hand pose in the image plane, and histograms of depths describe pose along the third spatial dimension. The third set reflects temporal dynamics of the hand shape.

— **HoG features from intensity images.** First, we make use zero-mean and unit variance-normalized intensity images to extract HoG features $\mathbf{h}_{\text{int}}$ [67] at 9 orientations from a 2-level spatial pyramid [168], i.e. from the whole image and a magnified version of it containing 3×3 cells.

— **Histograms of depths.** 9-bin depth histograms $\mathbf{h}_{\text{dep}}$ are extracted on two scales from depth maps of both hands: from a whole map and from each quarter of its upsampled version (by a factor of 2).

— **Derivatives of histograms.** First derivatives of HoGs and depth histograms are calculated as follows:

$$\delta\mathbf{h}(t) \approx \mathbf{h}(t+1) - \mathbf{h}(t-1), \tag{122}$$

where $\mathbf{h}$ can stand for both $\mathbf{h}_{\text{int}}$ and $\mathbf{h}_{\text{dep}}$. Combined together, these three sets of features form a 270-dimensional descriptor $[\mathbf{h}_{\text{int}}, \mathbf{h}_{\text{dep}}, \delta\mathbf{h}_{\text{int}}, \delta\mathbf{h}_{\text{dep}}]$ for each frame and, consequently, a descriptor of dimension of 1350 for the dynamic pose of each hand.

— **Extremely randomized trees** (ERT) [101] are adopted for data fusion and gesture classification. Ensemble methods of this sort have generally proven to be especially effective in conjunction with handcrafted features. During training, we followed the same iterative strategy as in the case of the neural architecture.

First, three ERT classifiers are trained independently on:
- skeleton descriptors (the same as described in Section 4.2.1),
- video features for the right hand,
- video features for the left hand.

2. for the sake of the competition

Once training is completed, features from all modalities with importance above the mean value are selected and once again fused for training a new, general ERT classifier. Feature importance is calculated as mean decrease in impurity (i.e. total decrease in node impurity weighted by proportion of samples reaching that node and averaged over all trees [37]).

At each step, ERT classifiers are trained with 300 estimators, an information gain criterion, no restrictions in depth and $\sqrt{N_f}$ features considered at each step (where N_f is the total number of features).

4.7.2.2 *Baseline audio model*

As a baseline method for the audio channel we employ a phoneme recognition method described in Section 4.2.3.1 and implemented with the Julius engine [172]. In this approach, each gesture is associated with a pre-defined vocabulary of possible ordered sequences of phonemes that can correspond to a single word or a phrase. After spotting and segmenting periods of voice activity, each utterance is assigned with a n-best list of gesture classes with corresponding scores. Finally, frequencies of appearances of each gesture class in the list are treated as output class probabilities.

4.7.3 *Experimental setup*

The set of hyper-parameters of two deep architectures, used for two submissions, slighly differ. In 2013, the color video signal was not used, and hand articulation was estimated solely based on the depth channel. At the same time, the earlier prototype had recurrent connections in the hidden layers, which were replaced by multi-scale processing.

— **Submission for ChaLearn 2014.** Hyper-parameters of the main multi-modal neural network for *gesture classification* are provided in Table 2, the architecture is the same for each temporal scale. Gesture *localization* is performed with an another MLP with 300 hidden units (see Section 4.6). All hidden units of both modules (frame-wise classification and following localization) have hyperbolic tangent activations. Hyper-parameters were optimized on the validation data with early stopping to prevent the models from overfitting, using dropout regularization. A single scale predictor operates at frame rates close to real time (24 fps on GPU). For simplicity, fusion weights for the different temporal scales are set to μ_s=1, as well as the weight of the baseline model (see Section 4.5).

— **Submission for ChaLearn 2013.** The earlier version of the architecture is defined by its parameters in Table 3 (including slight further optimization after the competition). In this case, all hidden units had ReLU activations. For the skeleton descriptor, we used only inclination, azimuth and bending angles, as well as pairwise distances.

Both deep learning architectures are implemented with the Theano library [31]. Threshold values for the audio preprocessing are set to τ_1=0.03 and $\tau_2 = 0.2$ s (see Section 4.2.3.2). In both cases, training was performed by SGD using NLL loss.

ChaLearn 2014: Network architecture			
Layer	Filter size / n.o. units	N.o. parameters	Pooling
Paths V1, V2			
Input D1,D2	72×72×5	-	2×2×1
ConvD1	25×5×5×3	1 900	2×2×3
ConvD2	25×5×5	650	1×1
Input C1,C2	72×72×5	-	2×2×1
ConvC1	25×5×5×3	1 900	2×2×3
ConvC2	25×5×5	650	1×1
HLV1	900	6 481 800	-
HLV2	450	405 450	-
Path M			
Input M	860	-	
HLM1	700	602 000	-
HLM2	700	490 700	-
HLM3	350	245 350	-
Path A (optional)			
Input A	40×9	-	1×1
ConvA1	25×5×5	650	1×1
HLA1	700	3 150 000	-
HLA2	350	245 350	-
Shared layers			
HLS1	1600	3 681 600	-
HLS2	84	134 484	-
Output layer	21	1 785	-

Table 2 – Hyper-parameters chosen for the deep learning models (a single temporal scale) for the *ChaLearn 2014* submission. For each temporal scale the architectural parameters are the same, although each network is trained independently.

ChaLearn 2013: Network architecture			
Layer	Filter size / n.o. units	N.o. parameters	Pooling
Paths V1, V2			
Input D1,D2	72×72×5	-	2×2×1
ConvD1	25×5×5×3	1900	2×2×3
ConvD2	25×5×5	650	1×1
HLV1	900	3 240 900	-
HLV2	450	405 450	-
Path M			
Input M	420	-	
HLM2	300	126 300	-
Path A			
Input A	40×9	-	1×1
ConvA1	25×5×5	650	1×1
HLA1	700	3 150 000	-
HLA2	350	245 350	-
Shared layers			
HLS1 (recurrent)	1550	5 535 050	-
HLS2	84	130 284	-
Output layer	21	1 785	-

Table 3 – Hyper-parameters chosen for the deep learning models in our post-competition experiments on *ChaLearn 2013 Multi-modal Gesture Recognition* dataset (where the color video channel was not used).

ChaLearn 2014: Official ranking					
#	Team	Score	#	Team	Score
1	**Ours (submission)**	**0.850**	7	Camgoz et al. [42]	0.747
2	Monnier et al. [195]	0.834	8	Evangelidis et al. [88]	0.745
3	Chang [45]	0.827	9	Undisclosed authors	0.689
4	Peng et al. [230]	0.792	10	Chen et al. [47]	0.649
5	Pigou et al. [232]	0.789		...	
6	Wu [318]	0.787	17	Undisclosed authors	0.271
Ours, improved results after the competition		**0.870**			

Table 4 – Official *ChaLearn 2014 "Looking at people" Challenge (track 3)* results (include only visual modalities).

ChaLearn 2014: Post-challenge results			
Model	Without motion detector	With motion detector	(Virtual) rank
ERT (baseline)	0.729	0.781	(6)
Ours (submission)	0.812	0.849	(1)
Ours (submission) + ERT (hybrid)	0.814	0.850	1
Ours (improved)	**0.821**	**0.868**	(1)
Ours (improved) + ERT (hybrid)	**0.829**	**0.870**	(1)

Table 5 – Performance of different proposed architectures on visual modalities, during and after the ChaLearn 2014 competition (reported as Jaccard Index). The table also illustrates the contribution of the additional localization step in the overall scores.

4.7.4 *Experiments on the original ChaLearn 2014 LaP dataset*

The top scores of the *ChaLearn 2014 "Looking at people" Challenge (track 3)* are reported in Table 4. We remind the reader, that in the official competition that year the audio was not included in the modality set, therefore all results, presented in this table, as well as in Table 5, we obtained using visual modalities only.

Our winning entry, correponding to a hybrid model (i.e. a combination of the proposed deep neural architecture from and the baseline model) surpasses the second best score by a margin of 1.61 points in terms of Jaccard index. We also note, that not only the hybrid model demonstrates the best performance, but also the multi-scale neural architecture alone, as well as the best performing *single-scale* neural model alone still outperform the closest competitor (see Table 5 and 6).

In a post-challenge work, we were able to further improve the score by 2.0 percentage points to 0.870 by introducing additional capacity into the model (in accordance

[Augmented] ChaLearn 2014: Per-scale / per-modality scores					
Step	Pose	Video	Pose & Video	Audio	All
2	0.823	0.818	0.856	0.709	0.870
3	0.824	0.817	0.859	0.731	0.873
4	0.827	0.825	0.859	0.714	0.880
all	0.831	0.836	**0.868**	0.734	**0.881**

Table 6 – Post-competition performance of the proposed deep learning architecture at different temporal scales (where gesture localization is refined by the additional binary motion detector). All numbers reported in the table are the Jaccard Index.

with Table 3), optimizing architectures of video and skeleton paths and employing a more advanced training and fusion procedure (ModDrop), which was not used for the challenge submission.

In the same Table 5, the reader can observe comparative performances of the baseline and hybrid models for visual modalities. In spite of low scores of the isolated ERT baseline model, fusing its outputs with the ones provided by the neural architecture is still slightly beneficial, mostly due to differences in feature formulation in the video channel (adding ERT to mocap alone did not result in a significant gain).

Detailed information on the performance of neural architectures for each modality and at each scale is provided in Table 6, including the multi-modal setting and per-modality tests (not including, however, the baseline model).

Our experiments have proven, that necessary information can be extracted at any scale given sufficient model capacity (which is typically higher for small temporal steps). Trained independently, articulated pose models corresponding to different temporal scales demonstrate similar performance, if predictions are refined by the gesture localization module. Video streams, containing information about hand shape and articulation, are also little sensitive to the sampling step and demonstrate good performance even for short spatio-temporal blocks.

The overall highest score is nevertheless obtained in the case of a dynamic pose with a maximal duration, roughly corresponding to the length of an average gesture (i.e. 17 frames, corresponding to a temporal sampling stride s=4).

Tables 7 and 8 illustrate the performance of proposed *modality-specific architectures* in comparison with results reported by other participants of the challenge. For both visual channels, articulated pose and video data, our methods significantly outperform the proposed alternative solutions.

For each combination, we also provide results obtained with a classification module alone (without additional gesture localization) and coupled with the binary motion detector (see Table 5). The experiments have shown that the localization module contributes significantly to overall classification-localization performance, expressed in term of the Jaccard Index.

ChaLearn 2014: Path M	
Model	Jaccard index
Evangelidis et al. [88], submitted entry	0.745
Camgoz et al. [42]	0.747
Evangelidis et al. [88], after competition	0.768
Wu and Shao [318]	0.787
Monnier et al. [195]	0.791
Chang [45]	0.795
Ours, submitted entry	**0.808**
Ours, after competition	**0.831**

Table 7 – Official *ChaLearn 2014 "Looking at people" Challenge (track 3)* results: performance of methods proposed by different participants solely on mocap (articulated pose) data.

ChaLearn 2014: Paths V1 + V2	
Model	Jaccard index
Wu and Shao [318]	0.637
Pigou et al. [232]	0.789
Peng et al. [230]	0.792
Ours, submitted entry	**0.810**
Ours, after competition	**0.836**

Table 8 – Official *ChaLearn 2014 "Looking at people" Challenge (track 3)* results: performance of methods proposed by different participants on video data (including both depth and RGB videos).

ChaLearn 2014: Path A				
Method used	Recall, %	Precision, %	F-measure, %	Jaccard index
Phoneme recognition	64.70	50.11	56.50	0.256
Learned representation	**87.42**	**73.34**	**79.71**	**0.545**

Table 9 – Comparison of two approaches to gesture recognition from audio: a method based on phoneme recognition (proposed in our earlier work, described in Sec. 4.2.3.1) and the representation learning pipeline presented in Sec. 4.2.3.2. The additional localization refinement step was not used in these experiments.

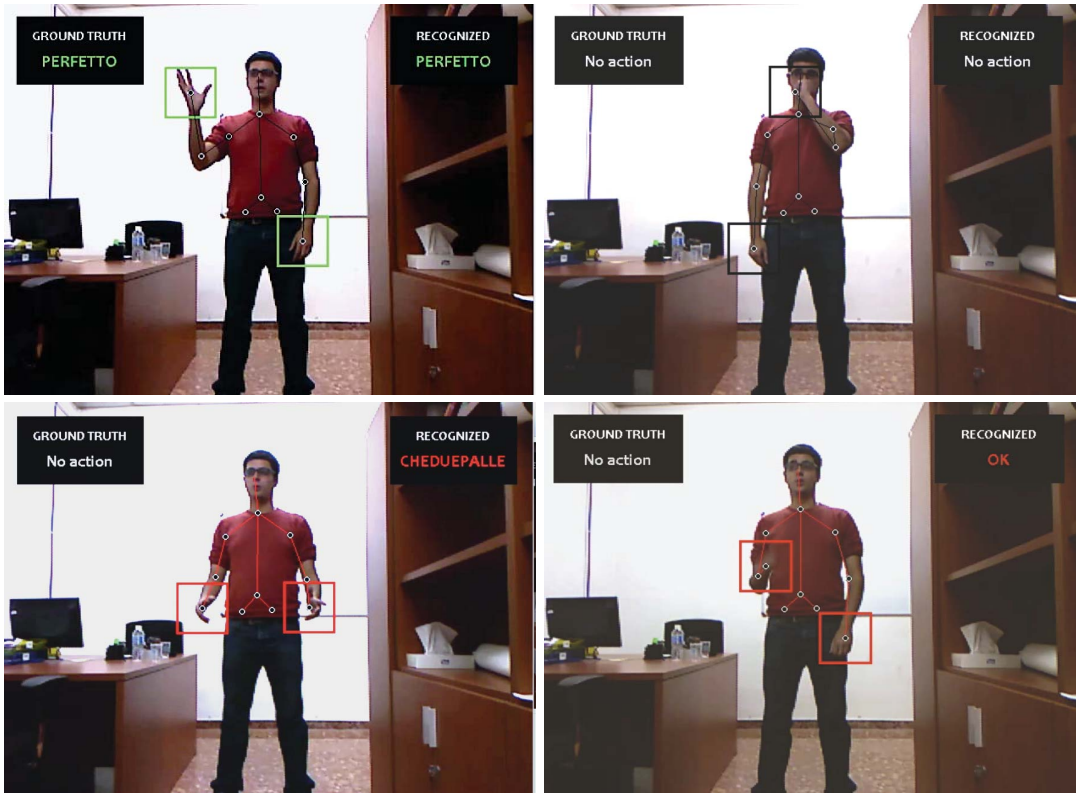


Figure 50 – Examples of gestures: correctly recognized, correctly ignored, false detections. Examples of gestures: correctly recognized and correctly ignored (the first row), false detections due to high similarity between gesture elements (the second row).

An example of the final output of the proposed framework is shown in Figure 50, including correct gesture detection and a failure case.

4.7.5 *Experiments on the ChaLearn 2014 LaP dataset augmented with audio*

In order to demonstrate how the proposed *visual* model can be further extended with *arbitrary* data modalities, we introduce speech to the existing setup. In this setting, each gesture in the dataset is accompanied by a word or a short phrase expressing the same meaning and pronounced by each actor while performing the gesture. As expected, introducing a new data channel resulted in significant gain in classification performance which was increased by 1.3 points of the Jaccard index (shown, along with other modalities, in Table 6).

As before, an audio-specific neural network was first optimized and discriminatively pre-trained on the audio data alone. Next, the same fusion procedure was employed without any change. In this case, the quality of predictions produced by the audio path depends on the temporal sampling frequency: the best performance is achieved for dynamic poses of duration ~0.5 s (see Table 6).

Although the overall score after adding the speech channel is improved significantly, the audio modality alone does not perform so well, especially when the additional refinement of gesture localization is not performed (shown in Table 9).

This can be partly explained by natural gesture-speech desynchronisation, which results in poor audio-based gesture localization. In this dataset, starting and ending points of each gesture are annotated based on *video* recordings, while pronounced words and phrases are typically shorter in time than movements. Moreover, depending on a style of each actor, vocalization can be either slightly delayed to coincide with gesture culmination, or can be slightly ahead in time announcing the gesture. Therefore, audio signal alone does not allow the model to robustly predict positions of starting and ending points of a gesture, which results in poor Jaccard scores.

Table 9 shows comparative performance of the proposed solution based on learning representations from mel-frequency spectrograms with our baseline model, which involves traditional phoneme recognition. Here, we report the values of Jaccard indices for the reference, but, as it was mentioned above, accurate gesture localization based exclusively on the audio channel is not possible for reasons outside of the model's control. To make a more meaningful comparison of the classification performance, we report recall, precision and F-measure for each model. In this case we assume that the gesture was correctly detected and recognized if temporal overlap between predicted and ground truth gestures is at least 20%.

The obtained results show that in the given context employing the deep learning approach drastically improves performance in comparison with the traditional framework based on phoneme recognition.

4.7.6 *Experiments on the ChaLearn 2013 Multi-modal Gesture Recognition dataset*

Finally, we report comparative performance of our earlier prototype, which was used during our participation in *ChaLearn 2013 Multi-modal Gesture Recognition* challenge, as well as in some later work. Initially, we were placed the 6th (still outperforming other competitors on *visual modalities*), and after performing post-competition analysis, we were able to improve our performance and achieve the highest score (with architecture parameters provided in Table 2). The official challenge ranking is provided in Table 10.

As the reader may see from Table 10, most of other participants used *traditional* methods in their framework, which are based on feature engineering, random forest classifiers or modeling temporal transitions with HMMs. More detailed scores, including per-modality performance, are provided in Table 11, which also shows the contribution of recurrent modeling.

As we have said before, the goal of this competition was to output a sequence of gesture classes present in a given video, without indicating gesture positions. To adapt to this protocol, in this framework, a predicted label for each dynamic pose $l(t)$ is calculated via finding the most probable class from the corresponding distribution of probabilities $o(t)$ produced by the RNN and thresholding:

$$l(t) = \begin{cases} \underset{k}{\operatorname{argmax}}\,(o_k(t)), & \text{if } \max(o_k(t)) > \tau, \\ 0, & \text{else.} \end{cases} \tag{123}$$

where τ is a parameter which is set empirically and which we will explain in a minute. Here, the zero value corresponds to the non-gesture class. Note, that the

ChaLearn 2013: Official ranking						
Rank	ED	Audio	Pose	Depth	Color	Method (keywords)
1	**0.128**	+	+	−	−	**HMM, kNN**[a]
2	0.154	+	+	−	−	random forest, kNN
3	0.168	+	+	−	−	random forest, AdaBoost[b]
4	0.172	+	−	+	+	Fisher vectors, SVM, GMM, kNN
5	0.173	−	+	+	+	HMM, GMM
6	**0.177**	+	+	+	−	**neural architecture (ours)**[c]
7	0.245	+	+	−	−	deep Boltzmann machines
8	0.258	+	+	−	+	HMM, SVM[d]
				...		
17	0.177	−	−	+	−	random forest
Ours, improved results after the competition						
-	**0.108**	+	+	+	−	**neural architecture (ours)**

Table 10 – Official results of the *ChaLearn 2013 Multi-modal Gesture Recognition Challenge* (based on [86]). Published work based on the entries includes: *a*. Wu et al. [319], *b*. Bayer and Silbermann [22], *c*. our earlier work, and *d*. Nandakumar et al. [206].

ChaLearn 2013: Per-modality scores			
Modalities used	Recall	Precision	ED
Independent dynamic poses (MLP)			
Depth	0.6569	0.6895	0.4454
Pose	0.7627	0.8010	0.3063
Audio	0.8588	0.8981	0.1665
Depth, Pose	0.7898	0.8285	0.2705
Depth, Pose, Audio	**0.8896**	**0.9130**	**0.1255**
Modeling temporal dependencies (RNN$^{\heartsuit}$)			
(Depth, Pose, Audio)$^{\heartsuit}$	**0.9004**	**0.9236**	**0.1081**
Random predictions			1.4747

Table 11 – Experimental results of per-modality tests on the *ChaLearn 2013 Multi-modal Gesture Recognition Challenge* dataset. Increasing recall and precision and decreasing edit distance indicate an improvement in performance.

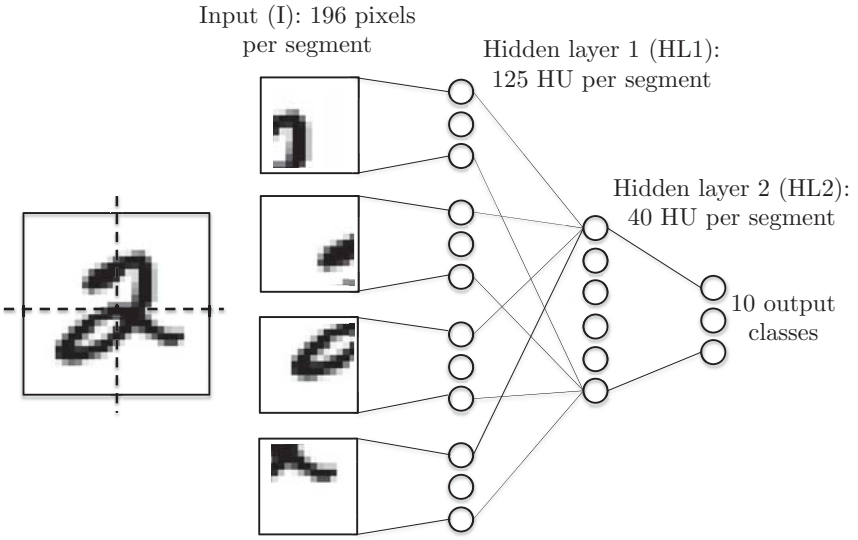


Figure 51 – "Multi-modal" setting for the MNIST dataset. Each digit is divided into 4 non-overlapping segments. A set of modality-specific units HL1 is first discriminatively pretrained on each image segment independently. The fusion is performed with a shared layer HL2 and the output layer.

argmax expression in the equation above can also result in zero. All neighboring dynamic poses assigned to the same labels are then aggregated into a single gesture.

Finally, it was communicated by the challenge organizers, that the number of gesture events per video in the test set should be between 8 and 12, with an average of 10. To adjust sensitivity of our gesture detector accordingly, we exploited this information by adapting the threshold τ through a simple iterative procedure. We proceeded through each test sequence, applying our model, and noting the estimated number of gestures. If this was below 8, we lowered τ. If this was above 12, we increased τ. This was repeated until convergence. Note that the test *labels* were never touched during this procedure.

We also provide values of precision and recall, calculated by comparing output to ground truth gesture sequences and establishing correspondences between detected gestures (Table 11). This correspondence is a byproduct of the edit distance algorithm. In this case we do not differentiate among gesture classes, i.e. each gesture positively contributes to the overall recall and precision values if it was detected with the correct label.

4.8 EMPIRICAL EVALUATION OF DIFFERENT FUSION STRATEGIES

In this section, we explore relative advantages of different training strategies, including such aspects as pre-training, initialization of shared layers and general and modality-specific regularization (dropout, ModDrop).

We start with preliminary experiments on MNIST dataset [171], to illustrate the idea, and then provide analysis and comparative scores on the ChaLearn 2014 LaP dataset augmented with audio channel.

MNIST: Fully connected setting			
Training mode	Errors	# of parameters	
Dropout, 784-1200-1200-10 [127]	107	2 395 210	N
Dropout, 784-500-40-10 (ours)	119	412 950	0.17N

Table 12 – Experiments on the original MNIST dataset: scores of fully connected single-modal architecture and a more compact network, which was used as a starting point for our experiments.

MNIST: "Multi-modal" setting, 196×4-125×4-40-10					
Pretraining (HL1)	Dropout (I)	ModDrop (I)	Errors	# of parameters	
no	no	no	142	118 950	0.05N
no	**yes**	no	123		
yes	no	no	118		
yes	**yes**	no	**102**		
yes	**yes**	**yes**	**102**		

Table 13 – Experiments on the original MNIST dataset: in the multi-modal setting, the first hidden layer is split between 4 channels, which results in a 196×4(I)-125×4(HL1)-40(HL2)-10 tree-structured architecture.

4.8.1 *Preliminary fusion experiments on MNIST dataset*

For our preliminary experiments, as a sanity check of the concept, we transform the MNIST dataset [171] to imitate multi-modal data. A classic deep learning benchmark, MNIST dataset consists of 28×28 grayscale images of hand written digits (from 0 to 9), where 60000 examples are used for training and 10000 images form the test set. In this work, we use the original version of it with no noise and data augmentation. We also avoid any additional data preprocessing and stay with the simplest architecture consisting of a set of fully connected neurons.

To explore the idea of multi-modal fusion, we cut each digit image in 4 quarters and assume that each image part corresponds to one modality (see Fig. 51). In spite of the apparent simplicity of this formulation, in the following section we show that obtained results reflect well the dynamics of a real multi-modal setup.

In this setting, we define two main objectives for the multi-signal training. First, we optimize the architecture and the training procedure to obtain the best overall performance on the full set of modalities. The second goal is to make the model robust to missing signals or high level of noise in separate channels. To explore the latter aspect, during test time we "cover" one or more image quarters (or add pepper noise to one or more image parts).

MNIST: ModDrop tests		
Training mode	Dropout	Dropout + ModDrop
Missing segments, test error, %		
All segments visible	**1.02**	**1.02**
1 segment covered	10.74	**2.30**
2 segments covered	35.91	**7.19**
3 segments covered	68.03	**24.88**
Pepper noise 50% on segments, test error, %		
All clean	**1.02**	**1.02**
1 corrupted segment	1.74	**1.56**
2 corrupted segments	2.93	**2.43**
3 corrupted segments	4.37	**3.56**
All segments corrupted	7.27	**6.42**

Table 14 – Effect of the ModDrop training on MNIST dataset: this additional modality-wise regularization makes the deep network predictions more robust to missing signals and noise in one or several input channels.

Currently, the state of the art for a fully connected 782-1200-1200-10 network with dropout regularization (50% for hidden units and 20% for the input) and tanh activations [127] is at the level of 107 errors on MNIST test set (see Table 13).

In this case, the number of units in the hidden layer is unnecessarily large, which is a basis of efficiency of dropout-like strategies. When real-time performance is a constraint, this redundancy in the number of operations becomes a serious limitation. Instead, switching to a tree-structured network (i.e. a network with separated modality-specific input layers connected with a set of shared layers) is helpful for independent modality-wise tuning of model capacity (which in this case does not have to be uniformly distributed over the input units) and minimizing the redundancy.

For this multi-modal setting we use optimized number of units (125) for each channel and do not apply dropout on the hidden units (which in this case turns out to be harmful due to compactness of the model), limiting ourselves to 20% dropping of input pixels. In addition, we apply ModDrop on the input, where the probability of each segment to be dropped is 0.1.

The experiments shown in Tables 14 and 13 have demonstrated that:

— separate pre-training of "modality-specific" paths generally yields better performance and leads to significant decrease in the number of network parameters due to attributing each channel with proper model capacity (see the 4th row of the bottom part of Table 13: in the case of pre-training the better performance (102 errors) obtained with 20 times less parameters);

— additional modality specific regularization, ModDrop, while does not affect the overall performance on MNIST (Table 13), makes the model significantly less

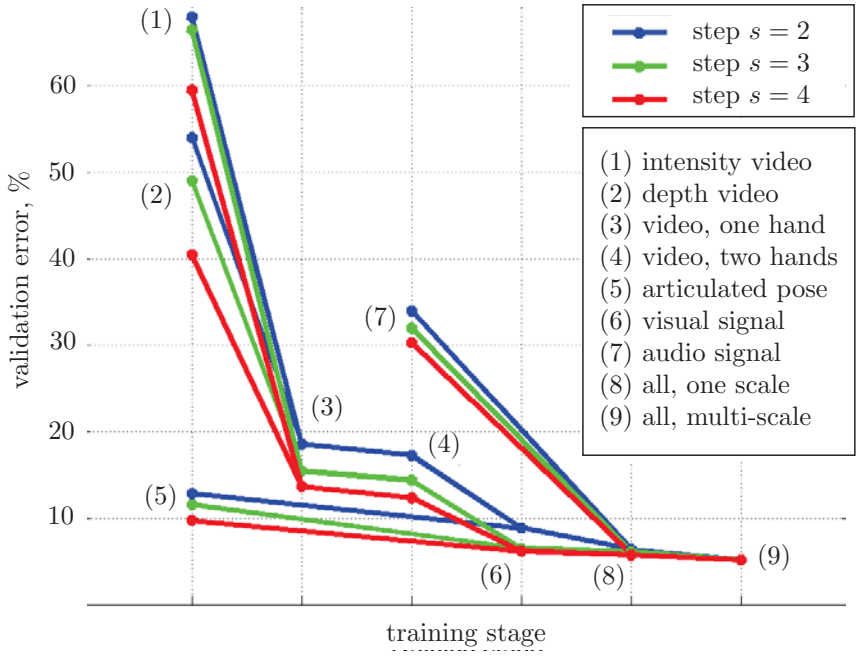


Figure 52 – Evolution of the validation error when the training moves from one stage to another. Starting from single-modality inputs, the networks learns to fuse color and video for each hand (point 3), then combines hands together (4), adds the articulated pose information (6), then audio (7), and finally integrates per-block predictions obtained at each of the temporal scales (9).

sensitive to missing signals and noise (see Table 14 where for each experiment one or more image part is covered or corrupted).

4.8.2 *Fusion experiments on the ChaLearn 2014 LaP dataset augmented with audio*

In a real multi-modal setting, optimizing and balancing a tree-structured neural architecture is an extremely difficult task, as its separated parallel paths vary in complexity and operate on different feature spaces. The problem becomes even harder given the constraint of real time performance and therefore it is necessary to minimize the number of operations in the network.

The whole history of our experiments on the ChaLearn 2014 LaP dataset has shown that insufficient modeling capacity of one of modality-specific subnetworks leads to drastic degradation in performance of the whole system due to multiplicative nature of the fusion process. Those bottlenecks are typically difficult to find without thorough per-channel testing.

As we have previously described in this chapter, we start by optimizing the architecture and hyper parameters for each modality separately through a discriminative pre-training. During the fusion, input paths are initialized with pre-trained values and fine tuned while training the output shared layers.

Furthermore, the shared layers can also be initialized with pre-trained diagonal blocks as described in Section 4.3.2, which results in significant speed up in the

ChaLearn 2014: Training strategies				
Pretraining	Dropout	Initial.	ModDrop	Accuracy, %
no	no	no	no	91.94
no	**yes**	no	no	93.33
yes	no	no	no	94.96
yes	**yes**	no	no	96.31
yes	**yes**	**yes**	no	96.77
yes	**yes**	**yes**	**yes**	**96.81**

Table 15 – Comparison of different training strategies on the ChaLearn 2014 LaP dataset augmented with audio. "Pretraining" column corresponds to modality-specific paths while "Initial." indicates whether or not the shared layers have also been pre-initialized with pre-trained diagonal blocks. In all cases, dropout (20%) and ModDrop (probability 0.1) are applied to the input signal. The accuracy corresponds to per-block (or per-dynamic pose) classification on the validation set.

training process. We have observed, that in this case, setting the biases of the shared hidden layer plays the key role and leads to convergence to a better solution.

Evolution of the validation error at different stages of the training process on the Chalearn 2014 LaP dataset in shown in Figure 52.

As in the case of MNIST experiments, we apply dropout on the input signal (with the found optimal probability of dropping 20%), as well as per-channel ModDrop with probability of 0.1. As before, dropping hidden units during training led to degradation in performance in our architecture due to its compactness.

A comparative analysis of efficiency of training strategies is reported in Table 15. Here, we provide scores in terms of a validation error of per dynamic pose classification as a direct indicator of convergence of the training process. Differences in effectiveness of different strategies agree well with what we have observed previously on MNIST. Modality-wise pre-training and regularization of the input have a strong positive effect on the performance. Interestingly, in this case ModDrop resulted in further improvement in scores even for the complete set of modalities (while simply increasing of dropout rate did not have the same effect).

Analysis of the network behavior in conditions of noisy or missing signals in one or several channels is provided in Table 16. Once again, ModDrop regularization resulted in much better network stability with respect to signal corruption and loss.

4.9 CONCLUSION

In this part of the thesis, we have described and analyzed dynamics of a generalized method for gesture and near-range action recognition from a combination of range video data and articulated pose.

As we have seen, one of the key aspects of our multi-modal approach is that each of the visual modalities captures spatial information at a particular spatial scale

ChaLearn 2014: ModDrop tests				
Modality	Dropout		Dropout + ModDrop	
	Accuracy, %	Jaccard index	Accuracy, %	Jaccard index
All present	96.77	0.876	96.81	0.880
Missing signals in separate channels				
Left hand	89.09	0.826	91.87	0.832
Right hand	81.25	0.740	85.36	0.796
Both hands	53.13	0.466	73.28	0.680
Mocap	38.41	0.306	92.82	0.859
Audio	84.10	0.789	92.59	0.854
Pepper noise 50% in channels				
Left hand	95.36	0.874	95.75	0.874
Right hand	95.48	0.873	95.92	0.874
Both hands	94.55	0.872	95.06	0.875
Mocap	93.31	0.867	94.28	0.878
Audio	94.76	0.867	94.96	0.872

Table 16 – Effect of the ModDrop training on ChaLearn 2014 "Looking at people" dataset augmented with audio channel. As before, ModDrop training results in more stable output predictions even in conditions of missing or corrupted signals. Furthermore, In this case of real multi-modal setting it also improves model performance on a complete set of modalities.

(such as motion of the upper body or a hand), and the whole system operates at three temporal scales.

This model can be further extended and augmented with arbitrary channels (depending on available sensors) by introducing additional parallel pathways, without significant changes in the general structure (the concept is illustrated by augmenting the video with speech). Furthermore, multiple spatial and temporal scales per channel can easily be integrated.

Finally, we have explored various aspects of multi-modal fusion in terms of joint performance on a complete set of modalities as well as robustness of the classifier with respect to noise and dropping of one or several data channels. As a result, we have proposed a modality-wise regularization strategy (ModDrop) allowing to obtain stable predictions even in the case of corrupted input signals.

Quoy des mains ? nous requerons, nous promettons, appellons, congedions, menaçons, prions, suplions, nions, refusons, interrogeons, admirons, nombrons, confessons, repentons, craignons, vergoignons, doubtons, intruisons, commandons, incitons, encorageons, jurons, tesmoignons, accusons, condamnons, absolvons, injurions, mesprisons, deffions, despittons, flatons, applodissons, benissons, humilions, moquons, reconsilions, recommandons, exaltons, festoyons, rejouïssons, complaignons, attristons, desconfortons, desesperons, estonnons, escrions, taisons : et quoy non ? d'une variation et multiplication à l'envy de la langue.

— Michel de Montaigne, Essais

5 HAND POSE ESTIMATION

In this chapter, we propose a strategy for hand pose estimation based on a deep regressor, which is trained on two different kinds of input. Raw depth data is fused with an intermediate representation in the form of a segmentation of the hand into parts. This intermediate representation contains important topological information and provides useful cues for reasoning about joint locations. The mapping from raw depth to segmentation maps is learned in either an unsupervised or weakly supervised way from two different datasets: (i) a synthetic dataset created through a rendering pipeline including densely labeled ground truth (pixelwise segmentations); and (ii) a dataset with real images, for which ground truth joint positions are available (but not dense segmentations). Loss for training on real images is formulated in either an unsupervised or weakly supervised fashion. In the former strategy, the network is trained to penalize local and global inconsistencies in segmentation outputs. In the latter case, the loss is generated from a patch-wise restoration process, which aligns tentative segmentation maps with a large dictionary of synthetic poses. The underlying premise is that the domain shift between synthetic and real data is smaller in the intermediate representation, where labels carry geometric and topological meaning, than in the raw input domain. Our experiments on the NYU Hand Pose Dataset [296] show that the proposed training method, including the weakly supervised adaptation step, decreases the error of joint localization by 15.7%.

5.1 INTRODUCTION

In the previous chapter, we have discussed how information about fine movements of hands and fingers can *complement* and *enrich* visual and audio signals when being part of a multi-modal system. Now it is time to refocus our attention on the hands specifically, and to discuss the problem of *hand articulation analysis*, starting by identifying its challenges, its significance and its potential relation to other building blocks of gestural interfaces.

Our next goal, which motivates this chapter, will be to go beyond gesture classification and to estimate hand motion with greater accuracy, allowing for *richer* human-machine interactions. From an application perspective, this would ensure,

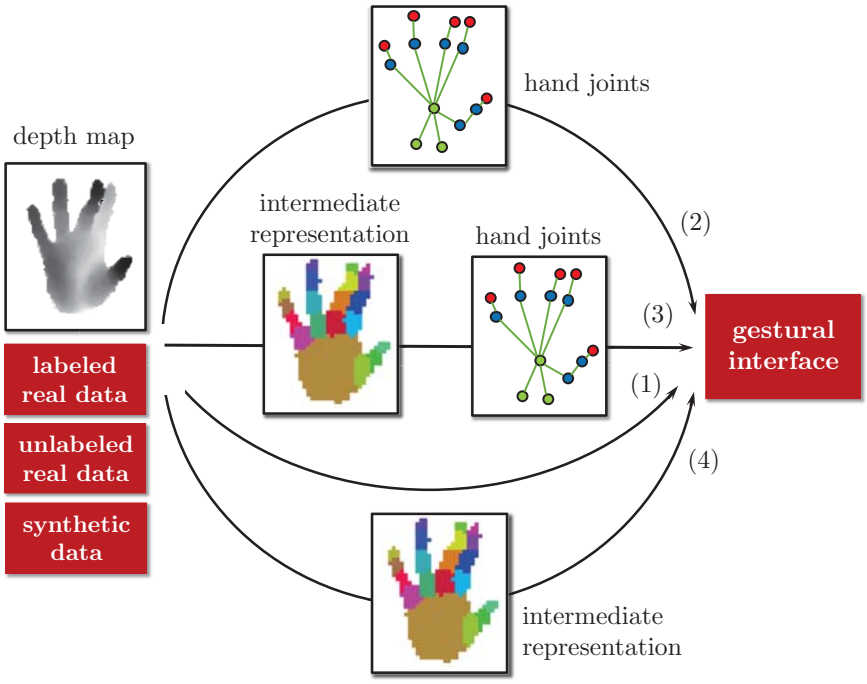


Figure 53 – The role of hand pose estimation in a human-computer interaction framework (see text for description).

for example, tight coupling between users and objects of interest, such as a cursor, or a manipulated virtual object.

In general, the problem of hand pose estimation is conventionally formulated as identifying positions of several key points in a hand, typically corresponding to its anatomical joints – similar to what we have seen in the case of a skeletal representation of a full body, which is provided by the Kinect SDK. Numerous examples of hand pose estimation frameworks, built on this principle, were given earlier in the state-of-the-art review (see Chapters 2.3 and 3.6). Made possible by the introduction of these cheap and reliable consumer depth sensors, such skeletal representations have revolutionized the field, effectively putting visual recognition into the hands of non-specialists in vision. However, while robust skeleton tracking is now possible in real time, at least in controlled settings [266], most systems provide coarse *body joints* and do not give the positions of individual *joints of the hand*.

Generally, estimating hand pose is inherently more difficult than tracking a full body skeleton. Given the relatively low resolution of commodity sensors and the strong noise they produce, fingers in these conventionally obtained depth images are usually composed of only a few pixels. For this reason, existing methods of hand pose estimation are mostly applicable in situations where the hands are close to the sensor. This is suited to interfaces where the user interacts with a computer in close proximity. However, applications in domotics, mobile robotics and games, which interest us the most, do not fall into this category.

Thinking about possible ways of *representing* a hand pose, it is important to keep in mind the *final goal* to which it serves. Assuming that the target application corre-

sponds to one or another form of a gestural interface, Figure 53 summarizes several ways of how hand pose estimation can be integrated in such a framework.

— Strategy (1) corresponds to our previously discussed case of establishing direct correspondence between a depth map and a discrete gesture label, or other gesture parameters. Except for the simplest case of gesture classification, to the best of our knowledge, there is no existing work on direct *continuous real-valued mapping* of raw depth frames into dynamically changing physical parameters of virtual objects (such as, for example, motion, rotation or scaling).

— Strategy (2) reflects the principle of most existing pose estimation frameworks, e.g. [296, 289, 290, 215, 216], where the goal is defined as minimization of the error, in Euclidean sense, between estimated positions of hand joints and the ground truth. The appealing advantage of this hand pose representation, expressed as joint positions, is that it allows for creating physical simulations by parameterizing a virtual 3D model of a hand with obtained estimates. However, the final stage of binding the obtained estimates and an interaction interface is typically left out of the scope of these methods.

— Strategy (3) introduces an additional *intermediate step* into the joint position estimation pipeline (such as, for example, hand segmentation into parts) [266, 152]. We will talk about potential advantages and shortcomings of this strategy a bit later.

— Finally, strategy (4) expresses a hypothesis that in a more general case, instead of joint positions, some *different* internal hand pose representation can be used to build a gestural interface (such as, for example, segmentation of a hand into parts, as shown in the figure).

It the pose itself is *not* required by the application and assuming an infinite amount of annotated data (meaning annotations of target *interactions*), end-to-end training exploiting Strategy (1) would probably be the best solution, except that this assumption is hardly realistic. In practice, introducing intermediate representations and auxiliary training objectives into the pipeline allows for enforcing certain priors and injecting additional knowledge about hand structure in the model. In this setting, however, the question which representation is *optimal* for which application, remains open.

On the other hand, existence of an *optimal* representation per se would not provide guarantees that such a representation could be *learned*, both in principle and, in particular, given available resources for creating training data. In this perspective, search for optimality is often substituted for an idea of creating a model, which could encode *as much as possible information* about a hand in the *most compact way*. Another desirable model property is generality, making it possible to use the representation for a wide range of applications.

A practical issue which arises at this step is data annotation. Most existing hand pose estimation benchmarks (see Chapter 2.3.1) contain a certain amount of depth images annotated with key locations (joints). However, manual labeling of large datasets is expensive and, in addition, inaccurate, especially when the image resolution is low. Automated methods, in turn, exploit additional wearable equipment or cumbersome acquisition setups requiring installation and calibration of several sensors [296], with the output still being imperfect.

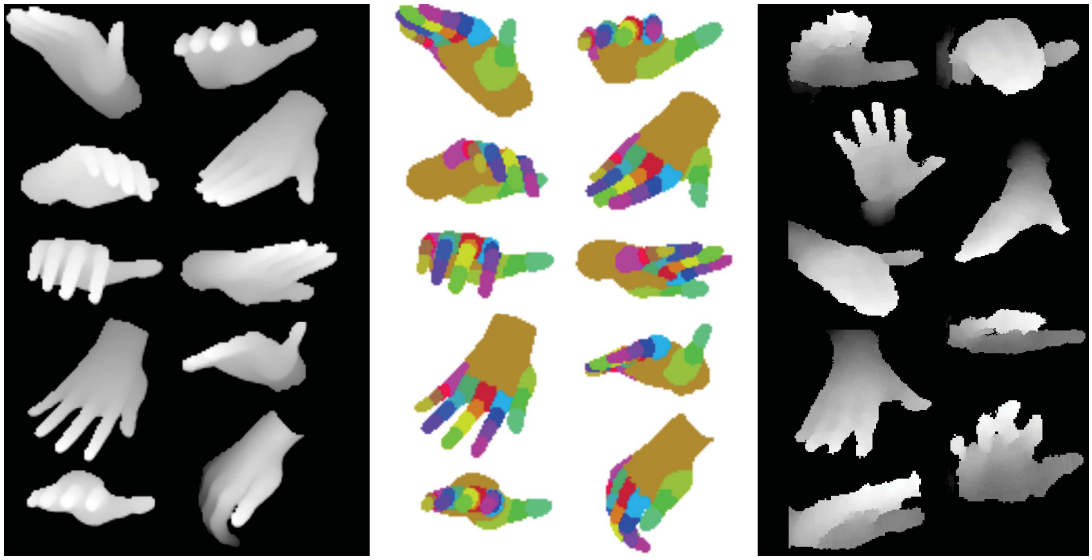


Figure 54 – Our model learns from labeled synthetic data and unlabeled real data. Left: synthetic depth input images, middle: ground truth segmentation maps for synthetic input, right: Real depth images captured with the Kinect sensor.

Alternatively, 3D modeling software offers a tempting opportunity to create potentially infinite amount of training data, which can be labeled, for example, not only with joint positions, but also with hand segmentation into parts. While manual pixel-wise annotation of large real datasets does not seem to be an option, such a representation, being dense and essentially spatial, may provide useful cues for learning. Examples of synthetic images with corresponding ground truth annotations in the form of segmentation maps, as well as similar poses captured with a real depth camera, are shown in Figure 54 (images taken from our dataset, which was created for this work).

Having explored the end-to-end Strategy (1) in the context of gesture recognition (see Chapter 4), this time we accommodate the setting of hand pose estimation to go deeper into Strategies (3) and (4), exploring the idea of learning *rich intermediate representations* both from solely synthetic annotated data and from heterogeneous labels of mixed real-synthetic sets. In this context, we also implement and optimize Strategy (2), which maps the depth maps into joint locations directly, and consider it as a baseline for our work.

Segmentations of the body or hand into parts were one of the of the most widely used representations when consumer depth cameras first became popular [266, 288]. A typical pipeline of such an approach is shown in Figure 53, path (3), where the segmentation is first obtained from raw input data and then used as the *unique* input for the estimation of joint positions.

This idea of using intermediate representations (i.e. segmentations, or others) recently seems to have lost the favor of the machine learning community, which now prefers direct regression of joints from raw data. However, in this work we argue that an innovative use of segmentations *can* boost the performance of direct regression based methods, as long as the new representations are *complementary* rather

than the sole input to regression. In this case, the main arguments we develop are the following:

— an intermediate pose representation, which is defined as a segmentation into parts, contains *rich structural* and *topological information*, since the label space itself is structured. Labels have adjacency, topological and geometric relationships, which can be leveraged and translated into loss for weakly-supervised training;
— a *regression* of joint positions is *easier* and *more robust* if the depth input is combined with a rich semantic representation like a segmentation into parts, provided that this semantic segmentation is of high fidelity;
— incorporating additional rich information provided by a network, which is trained on a vast amount of cheap and adjustable synthetic data, results in *additional external supervision* for the training, which is especially crucial when the size of a labeled real dataset is strictly limited.

Along these lines, the **research priorities** of this work, along with the imposed constraints, are formulated as follows.

(1) We explore frontiers and possibilities of a purely *learning based* and *data-driven* strategy, with no explicit use of 3D rendering during test time.
(2) In order to minimize the burden of real data annotation, we create a large dataset of densely labeled *synthetic data*, aiming on *injecting knowledge* from rendered images in the *training process*.
(3) We perform *unsupervised* or *weakly-supervised domain adaptation* of the pretrained model from *synthetic* images to *real* depth maps (described below).
(4) Our primary goal is to perform accurate pose estimation from a *single frame*, without incorporating dynamic information at this step of development.
(5) *Real time inference* (with no expensive optimization during test time) is necessary, having in mind human-computer interaction as a target application.

To implement principles (2) and (3) in a structured machine learning setting, we tackle the problem of *pixel-wise segmenting hands into parts*, first in a *supervised* manner by training a deep neural network on synthetic data, followed by *transductive adaptation* of the learned model from the synthetic to real domain. In this context, we propose two domain adaptation scenarios, including *unsupervised* and *weakly-supervised* formulations (illustrated in Figure 55), which we describe below.

— **Weakly supervised adaptation**.

In the *weakly supervised setting*, we present a new method for the regression of *hand joint positions*, which is based on joint training of several learners working on *heterogeneous* real-synthetic labels (see Figure 55b). For this setting, we employ an intermediate pose formulation based on a segmentation into parts and show that this representation is a powerful tool in the special context of training with multiple heterogeneous training sets. The segmentor is first trained on the synthetic label data following weakly supervised adaptation of the model to the real set and, finally, shifting to the new joint localization objective.

This model is learned from two training sets: (i) a (possibly small) set of real images acquired with a consumer depth sensor. Ground truth joint positions are assumed to be available and can be obtained, for instance, by multi-camera motion capture systems; (ii) a second (and possibly very large) set of synthetic

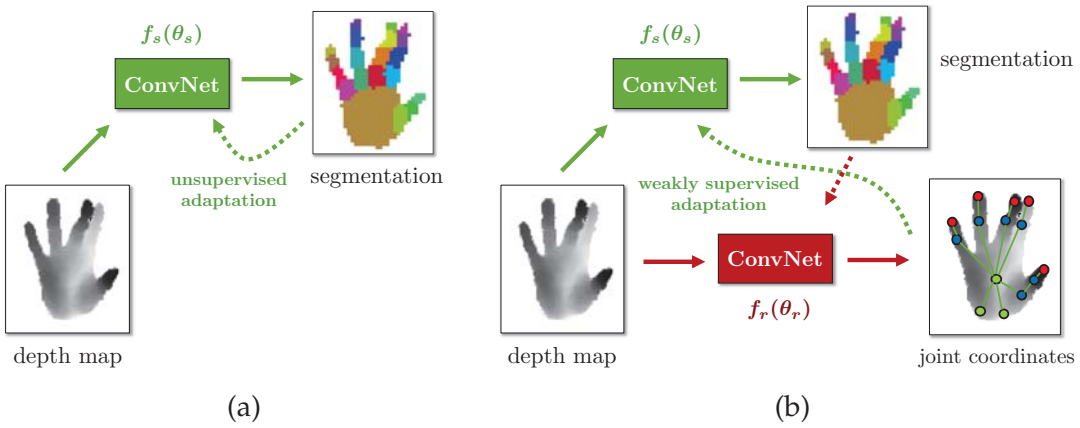


Figure 55 – Our adaptations of Strategies (3) (on the right) and (4) (on the left), introduced in Figure 53, which are based on learning intermediate pose representations in the form of hand segmentation into parts. The input image is taken from the NYU Hand Pose Dataset [296].

training images produced from 3D models by a rendering pipeline, accompanied by dense ground truth in the form of a segmentation into hand parts. This ground truth is easy to come by, as it is usually automatically created by the same rendering pipeline.

The proposed weakly supervised training-adaptation method exploits the rich geometrical and topological information of the intermediate representation. During the training process, predicted segmented patches from real images are aligned with a very large dictionary of labeled patches extracted from rendered synthetic data. The novelty here lies in the fact, that we do not match input patches, but patches taken from the intermediate representation, which include the to-be-inferred pixel label and its local context.

— **Unsupervised adaptation**.

Another possible scenario is when annotations for the real data are *not* available in any amount. In this case, our goal is to train the hand segmentor solely on the synthetic data and then perform unsupervised model adaptation to the real domain. This approach was explored in our earlier work, where we did not perform estimation of hand joint positions, optimizing solely for the quality of produced segmentation maps (in the spirit of Strategy (4), assuming that this output can be useful for tasks other than estimating joint positions).

Our main contribution to the unsupervised adaptation scenario (shown in Figure 55a) is the way in which structural information is treated in the learning process. Instead of combining the pre-trained predictor with a structured model, for instance a graphical model, we integrate structural information directly into the learning algorithm, aiming to improve the prediction model per se. As a consequence, at test time, image pixels are classified independently, keeping the advantages of low computational complexity and retaining the ability to parallelize.

The information integrated into the training procedure is related to prior information, which can be assumed over a segmented image. Our unsupervised

method is based on two contributions. Firstly, *contextual* information is extracted from local context in unlabeled samples through a learned model trained on synthetic labeled examples. Secondly, similar to body part maps, ground truth hand part segmentation maps are assumed to contain a *single connected region* per part, which commonly holds if we ignore rare exceptions occurring due to severe self occlusion. We show that this information can be formalized and leveraged to integrate unlabeled samples into the training set in an unsupervised adaptation setting.

Although we focus on the application of hand pose estimation, the proposed method is also applicable to other problems involving segmentation of entities, for example, objects, people, and scenes into parts.

In the following discussion, we will show that the intermediate representation (the segmentation into parts) is able to improve pose regression performance if it is combined with raw depth input, and a key component to obtaining this improvement is getting reliable segmentations for real data. We will consider the weakly supervised setting first, as this technology is more advanced and was evaluated in terms of accuracy of hand joint positions, and then we will review our earlier findings in the context of unsupervised transductive adaptation.

5.2 WEAKLY SUPERVISED PIPELINE: METHOD OVERVIEW AND DEFINITIONS

In our framework, the set of training images consists of depth maps and is denoted as $\mathcal{D}=\{I^{(i)}\}$, $i=1\ldots|\mathcal{D}|$. Generally, in this chapter, we will talk about two training subsets: *synthetic data* labeled with pixel-wise segmentations into parts and *real data* weakly annotated with key points, a.k.a. joints.

Let us introduce notation that will be used to formalize the training algorithm.

Inputs and corresponding annotations

— From the entire set of depth maps, L samples are *synthetic* and *densely annotated*, which we denote as $\mathcal{D}_S=\{I_S^{(i)}\}$, where $i=1\ldots L$, $L<|\mathcal{D}|$.

— The subset of *weakly labeled real* images is denoted as $\mathcal{D}_R=\{I_R^{(i)}\}$, where $i=L+1..|\mathcal{D}|$.

— The set of ground truth segmentation maps corresponding to the synthetic data $\mathcal{D}_S$ is denoted as $\mathcal{G}=\{Y^{(i)}\}$, where $i=1\ldots L$, $L\leqslant|\mathcal{D}|$.

— The set of ground truth hand joint locations corresponding to the subset of real data $\mathcal{D}_W$ is denoted as $\mathcal{J}=\{J^{(i)}\}$.

— Pixels in the different maps are indexed using a linear index j: $I^{(i,j)}$ denotes the j^{th} pixel of the i^{th} depth map I.

The synthetic frames are rendered using a deformable 3D hand model. A large variety of viewpoints and hand poses (typical for interactive interfaces) is obtained under manually defined physical and physiological constraints. In Section 5.4.2.1, we will provide a more detailed description of the data creation process. Our intermediate representation is a segmentation into 20 parts, illustrated in Figure 63. Among these, 19 parts correspond to finger parts, 1 part to the palm. The background is considered to be subtracted in a preprocessing step and is not part of the segmentation process.

For the sake of generalization, and also keeping in mind that manual dense labeling of real data is tedious and impractical, we do not assume that ground-truth segmentation of real data is available in any amount. Instead, in parallel with supervised learning on annotated synthetic images, we use weakly labeled frames (i.e. frames annotated only with positions of key points, or joints) for global optimization and model adaptation during training time.

Learning pathways

— $f_s(\theta_s)$: $I^{(i,j)} \rightarrow Y_s^{(i,j)}$ denotes a **segmentation learner** with parameters θ_s mapping each pixel $j=1 \dots M$ in a depth map i (having depth value $I^{(i,j)}$) into a corresponding pixel of an output segmentation map Y_s with elements $Y_s^{(i,j)}$, having one of possible K possible values corresponding to hand segments, indexed by $k=1 \dots K$. Recall that Y with no subscript stands for *ground truth* segmentation maps.

— $f_r(\theta_r)$: $\{I, Y_s\} \rightarrow J$ denotes a **regression learner** parameterized by vector θ_r, which learns a mapping from raw depth data combined with segmentation maps (output by the segmentation learner $f_s(\theta_s)$) to the set of joint positions J.

The probabilistic setting of our training algorithm makes it convient to introduce a difference between a random variable and its realization. In the following and as usual, uppercase letters denote random variables or fields of random variables and lower case letters denote realizations of values of random variables or of fields of random values. Realizations of random fields Y_s, Y_c and Y defined above are thus denoted as y_c, y_d and y, respectively. Furthermore, $P(X=x)$ will be abbreviated as $P(x)$ when it is convenient.

Considering Strategy (3) from Figure 53 and setting the target goal as localization of hand joint positions, our learning pipeline can be decomposed into three main strategic steps:

(1) supervised *pre-training* of the *segmentation learner* $f_s(\theta_s)$ on the synthetic training data $\mathcal{D}_S$,

(2) *weakly supervised adaptation* of the segmentation learner $f_s(\theta_s)$ to the real data $\mathcal{D}_R$ (where the labels of the real data, i.e. joint locations, are used to guide the process),

(3) training the *regression learner* $f_r(\theta_r)$ using the weakly labeled real set $\mathcal{D}_R$.

In practice, we do not make a strong differentiation between steps (1) and (2). Instead, during training, once the segmentation learner starts to produce meaningful predictions on the *synthetic* data, we proceed with injecting a gradually increasing amount of *real samples* and perform supervised and weakly-supervised updates interchangeably. The training loss function Q in this case is formulated as follows:

$$Q = Q_s + Q_r, \tag{124}$$

where Q_s is a *synthetic* (supervised) term and Q_r is a *real* (weakly-supervised) term. In the following sections, we will first discuss the pre-training step with the Q_s objective (Section 5.2.1) and then proceed with describing our strategy for formulating the weakly supervised term Q_r (Section 5.3).

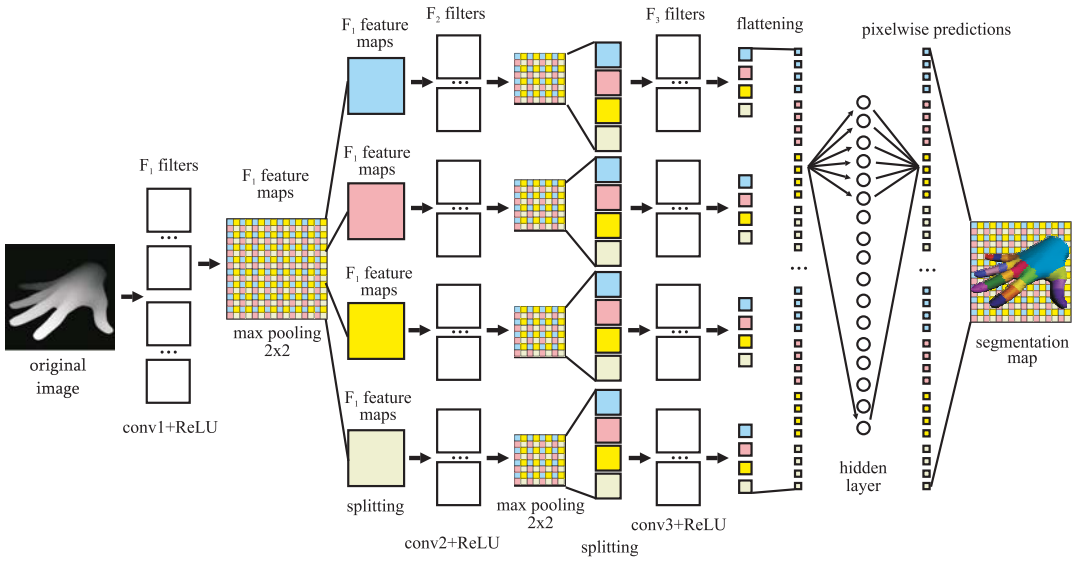


Figure 56 – The proposed deep convolutional architecture of the segmentation learner $f_s(\theta_s)$.

5.2.1 *Segmentation Learner: supervised training*

The supervised term in equation (124) classically links the predicted class of each pixel to the ground truth hand part label. For the segmentation learner $f_s(\theta_s)$ it is formulated as negative log-likelihood (NLL) for pixel-wise classification:

$$Q_s(\theta_s|\mathcal{D}_S) = -\sum_{i=0}^{L}\sum_{j=0}^{M} \log P\left(Y_s^{(i,j)} = y^{(i,j)} \middle| I_S^{(i,j)}; \theta_s\right). \tag{125}$$

Recall here, that $Y_s^{(i,j)}$ denotes the output of the segmentation learner and $y^{(i,j)}$ is a ground truth label for the pixel $I_S^{(i,j)}$.

Structurally, the segmentation learner $f_s(\theta_s)$ is based on a modified convolutional network architecture (see Figure 56). We will describe its parameters in more detail in the experimental part (Section 5.4). At this step, however, it is important to understand that the generated segmentations have the same spatial resolution as the input depth maps.

5.2.2 *Weakly supervised transductive adaptation*

In this section, we present a *weakly supervised* strategy for performing transductive adaptation of the segmentation learner $f_s(\theta_s)$ to the real data.

In our weakly supervised setting, we rely on ground truth annotations for joint positions $\mathcal{J}$, which can be obtained in several ways: in [266], external motion capture using markers is employed, while in [296], training data is acquired in a multi-view setting from three different depth sensors and an articulated model is fitted offline (the NYU Hand Pose Dataset presented in the latter paper is used in our work, along with our synthetic data).

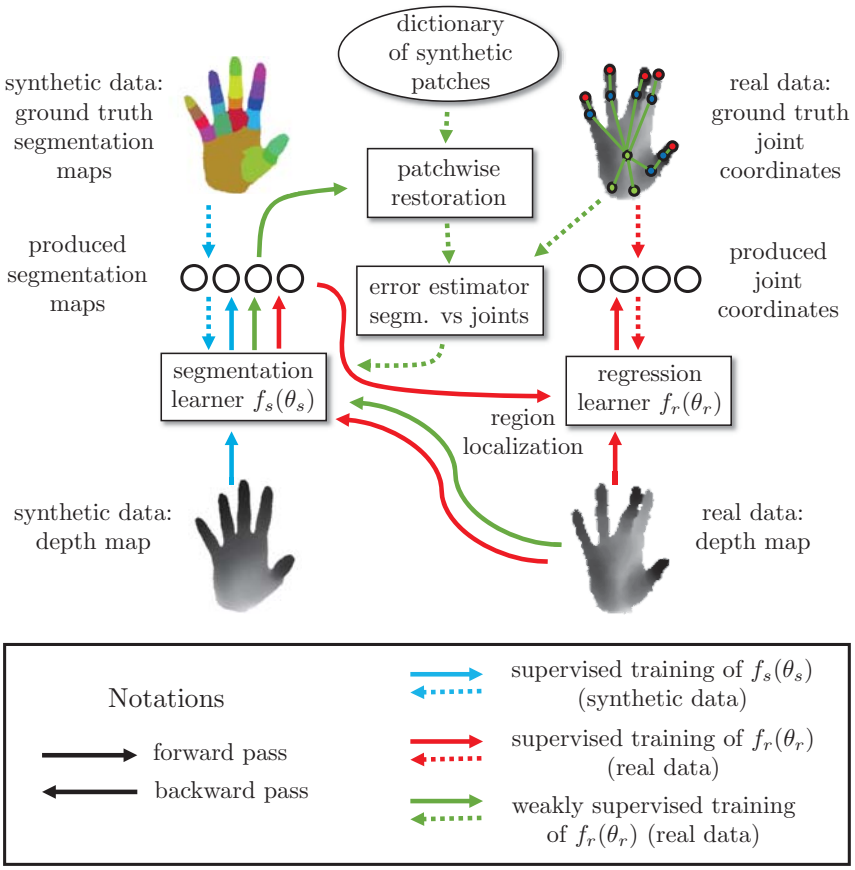


Figure 57 – A functional overview of the *weakly supervised* transductive adaptation pipeline. Blue data flow corresponds to supervised training of the segmentation learner $f_s(\theta_s)$. Green flow corresponds to weakly supervised training of $f_s(\theta_s)$. Red arrows show supervised training of the regressor $f_r(\theta_r)$, which will be discussed in Section 5.2.4.

The proposed method leverages two different mappings learned on two training sets. The functional decomposition as well as the dataflow during training and testing are outlined in Figure 57. In this case, the segmentation learner $f_s(\theta_s)$ is pre-trained on the synthetic data $\mathcal{D}_S$ (shown in blue), and then fine-tuned in a subsequent step by weakly supervised training on real data $\mathcal{D}_R$, resulting in a refined prediction model. This latter step is shown as green data flow in Figure 57.

At the adaptation step, since no ground truth segmentation maps exist for the real data, we generate a loss function for training based on two sources:

— sparse information in the form of ground truth joint positions $\mathcal{J}$, and
— *a priori* information on the local distribution of part labels on human hands through a patch-wise restoration process, which aligns noisy predictions with a large dictionary of synthetic poses.

The weakly supervised training procedure is shown as green data flow in Figure 57. Each real depth image is passed through the pre-trained segmentation learner $f_s(\theta_s)$, resulting in a segmentation map Y_s. This noisy predicted map is restored through a **restoration process** f_{nn}, described further below.

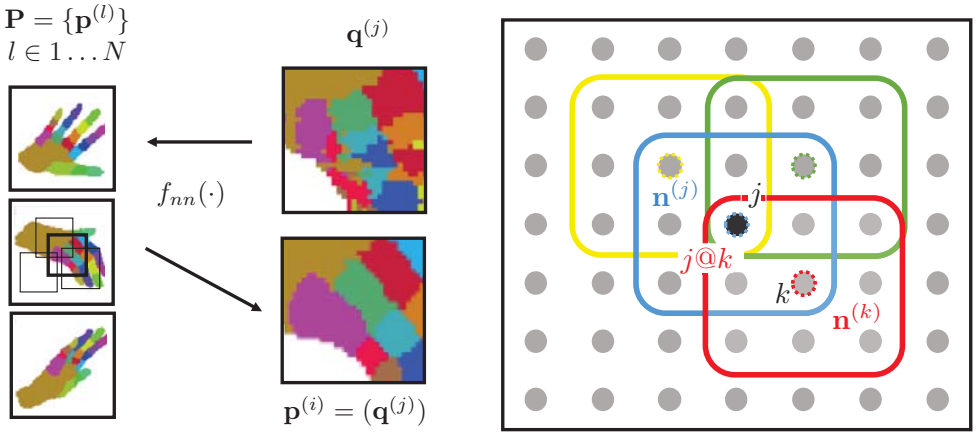


Figure 58 – Illustration of the patchwise restoration process. On the left: search for the nearest neighbor in the patch dictionary, on the right: integration procedure.

The quality of the *restored segmentation map* is estimated by comparing it to the set of ground truth joint positions $\mathcal{J}$ for this image. In particular, for each joint, a corresponding part-label is identified, and the barycenter of the corresponding pixels in the segmentation map is calculated. A rough quality measure for a segmentation map can be given as the sum (over joints) of the L_2 distances between barycenters and ground truth joint positions. The quality measure is used to determine whether the restoration process has lead to an improvement in segmentation quality, i.e. whether the barycenters of the restored map are closer to the ground truth joint positions than the barycenters of the original prediction. For images where this is the case, the segmentation learner is updated for each pixel, minimizing NLL using the labels of the restored map as artificial *ground truth*.

5.2.3 *Patchwise restoration*

For the restoration step, we proceed patchwise and extract patches of size $P \times P$ from a large set of synthetic segmentation images $\mathcal{G}$, resulting in a dictionary of patches $\mathbf{P}=\{\mathbf{p}^{(l)}\}$, $l \in \{1 \ldots N\}$ (shown in Figure 58).

In our experiments, we used patches of size 27×27 and a dictionary of 36 million patches was extracted from the training set (see Section 5.4). As also reported in [275], the range of poses which can occur in natural motion is extremely large, making full global matching with pose datasets difficult. This motivates our patch-based approach, which aims to match at a patch-local level rather than matching whole images.

A given real input depth image I_R is aligned with this dictionary in a patchwise process using the intermediate representation. For each pixel j (here, a single index j identifies a given pixel, for compactness of notation), a patch $\mathbf{q}^j$ is extracted from the segmentation produced by the learner $f_s(\theta_s)$, and the nearest neighbor is found by searching the dictionary $\mathbf{P}$:

$$(\mathbf{q}^{(j)}) = \arg \min_{\mathbf{p}^{(l)} \in \mathbf{P}} d_H(\mathbf{q}^{(j)}, \mathbf{p}^{(l)}), \tag{126}$$

where $d_H(\mathbf{q}, \mathbf{p})$ is the Hamming distance between the two patches $\mathbf{q}$ and $\mathbf{p}$. The search performed in equation (126) can be calculated efficiently using KD-trees.

In the naïve setting, a restored label for pixel j could be obtained by choosing the label of the center of the retrieved patch $\mathbf{n}^{(j)}$. This however leads to noisy restorations, which suggest the need for spatial context. Instead of choosing the center label of each patch only, we propose to use all of the labels in each patch (see Figure 58). For each input pixel, the nearest neighbor results in a local window of size $W \times W$ are integrated. In particular, for a given pixel j, the assigned patches $\mathbf{n}^{(k)}$ of all neighbors k are examined and the position of pixel j in each patch is calculated. A label is estimated by voting, resulting in a mapping $f_{nn}(.)$:

$$f_{nn}(\mathbf{q}^{(j)}) = \arg\max_{l} \sum_{k \in W \times W} \mathbb{I}(l = \mathbf{n}^{(k, j@k)}), \tag{127}$$

where $\mathbf{n}^{(k)} = (\mathbf{q}^{(j)})$ is the nearest neighbor result for pixel k, $\mathbf{n}^{(j,m)}$ denotes pixel m of patch $\mathbf{n}^{(j)}$, $\mathbb{I}(\omega) = 1$ if ω holds and 0 else, and the expression $j@k$ denotes the position of of pixel j in the patch centered on neighbor k.

This integration bears some similarity to work by Ganin and Lempitsky [97], where information of nearest neighbor searches is integrated over a local window, albeit through averaging a continuous mapping. It is also similar to the patchwise integration performed in structured prediction forests [160].

If real-time performance is not required, the patch alignment process, formalized by equation (126), can be regularized with a graphical model including pairwise terms favoring consistent assignments, for instance, a Potts model or a term favoring patch assignments with consistent overlaps. Interestingly, this produces only slight gains in performance, especially given the higher computational complexity. Moreover, the gains vanish if local integration (equation (127)) is added.

5.2.4 *Regression Learner*

Finally, the regression learner $f_r(\theta_r)$, taking a real depth image I_R as an input and producing 3 coordinates for a given joint, also incorporates the information provided by the segmentation learner $f_s(\theta_s)$ into its training.

Structurally, it resembles an inception unit [285, 179] where the output of the first convolutional layer after max pooling as passed through several parallel feature extractors capturing information at different levels of localization. The structural organization of this module is shown in Figure 59.

The output of the first convolutional layer c1 (followed by pooling p1) is aligned with the segmentation maps Y_s produced by the segmentation learner $f_s(\theta_s)$. From each feature map, for a given joint we extract a localized *region of interest* filtered by the mask of a hand part to which it belongs (or a number of hand parts which are naturally closest to this joint). These masks are calculated by performing morphological opening on the regions, which have the corresponding class label in the segmentation map.

Once the local region is selected, the rest of the feature map area is set to 0. The result, along with the original feature maps is then fed to the next layer, i.e. an inception module.

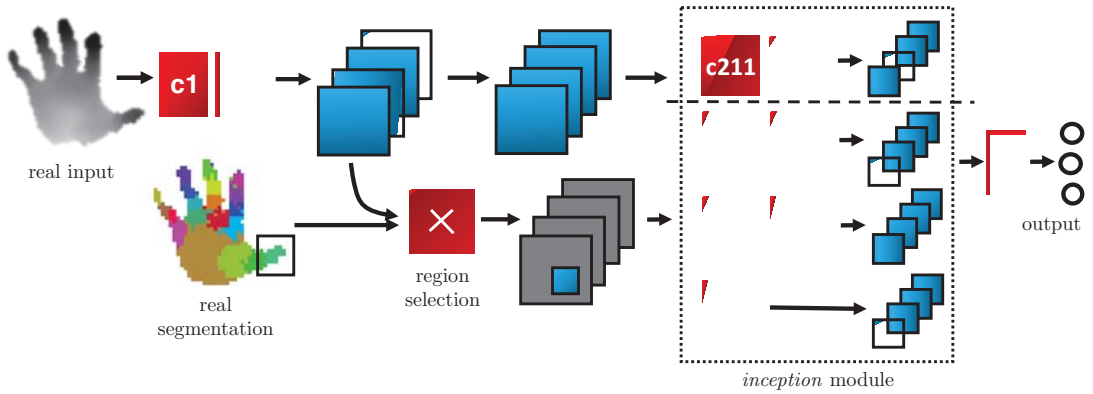


Figure 59 – Organization of the regression learner $f_r(\theta_r)$, mapping input depth maps $\mathcal{D}_R$ into predicted joint locations given additional information produced by the segmentation learner $f_s(\theta_s)$. All functional modules are shown in red and produced feature maps are shown in blue. The gray area corresponds to masked regions on the feature maps.

The rest of the training process is organized in such a way that the network's capacity is split between *global structure* of the whole image and the *local neighborhood*, and a subset of inception 3×3 filters is learned specifically from the local area surrounding the point of interest.

5.3 UNSUPERVISED PIPELINE: METHOD OVERVIEW AND DEFINITIONS

In this section, we introduce an **alternative strategy** to the weakly supervised pipeline described in the previous section. This research corresponds to our earlier work and is dedicated to *unsupervised* transductive adaptation.

First of all, we must note, that lately, there has been renewed interest in semantic segmentation or semantic labeling methods in the community, in which an image of a natural scene (typically RGB) is segmented into semantically meaningful regions (e.q., road, water, sky, building, vegetation). In these tasks, taking into account *structural (contextual) information* in addition to local appearance information is primordial. Contextual information often allows the model to disambiguate decisions where local information is not discriminative enough. In principle, increasing the support region of a learning machine can increase the amount of context taken into account for the decision. In practice, this places all of the burden on the classifier, which needs to learn a highly complex prediction model from a limited amount of training data, most frequently leading to poor performance. An elegant alternative is to apply multi-scale approaches. Farabet et al. [90] propose a multi-scale convolutional net for scene parsing which naturally integrates local and global context.

Our proposed method is similar to auto-context [298, 234] in that the output of an initial classifier is fed into a second classifier, which is learned to integrate the context of the pixel to predict. However, whereas auto-context models aim at repairing the errors of individual classifiers, our model uses contextual information to extract structural information from unlabeled images in a unsupervised setting. The ability

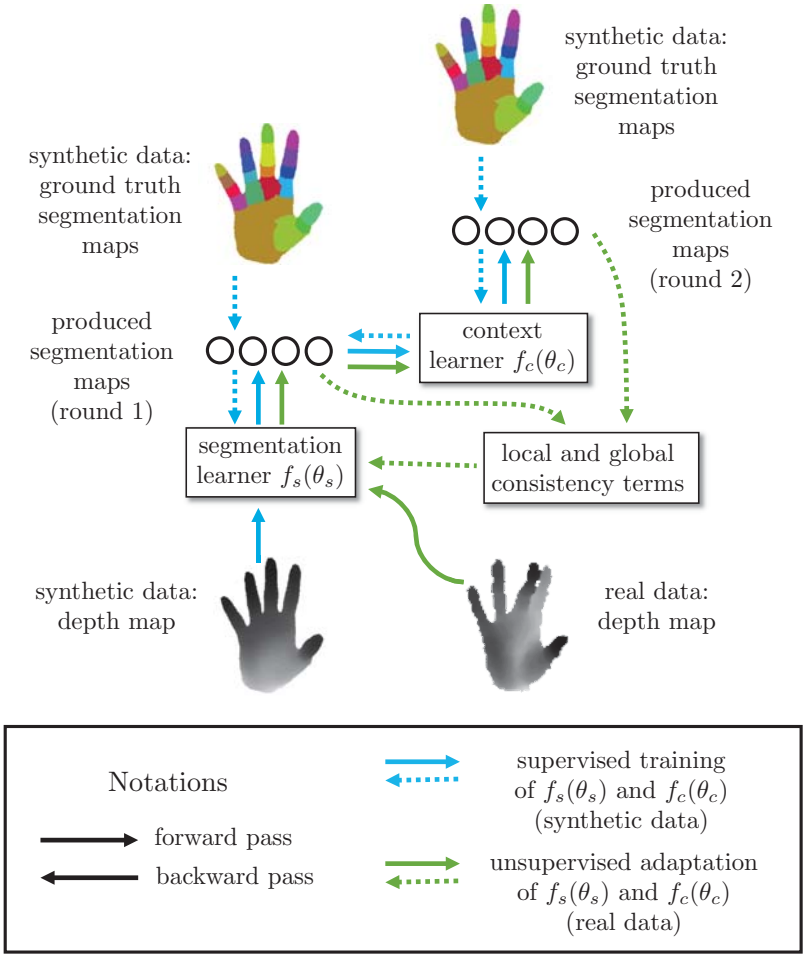


Figure 60 – A functional overview of the *unsupervised* transductive adaption pipeline. Blue data flow corresponds to supervised training of the segmentation learner $f_s(\theta_s)$ and the context learner $f_c(\theta_c)$ using synthetic data $\mathcal{D}_S$. Green flow corresponds to the adaptation step using real data $\mathcal{D}_R$ with no ground truth annotations.

to seamlessly combine unlabeled samples and labeled training data is an important motivation behind the field of deep learning of representations [28, 314].

A functional overview of our *unsupervised* transductive adaptation is presented in Figure 60. Here, we do not consider the regression learner $f_r(\theta_r)$ explicitly, however, it could be plugged in and take as in input, as in the weakly supervised pipeline, the depth images along with the produced segmentations.

In this framework, in addition to the *segmentation learner*, which was described in Section 5.2, we employ an additional *context learner* $f_c(\theta_c)$: $N^{(i,j)} \rightarrow Y_c^{(i,j)}$ [149], predicting the pixel label $Y_c^{(i,j)}$ from its neighborhood $N^{(i,j)}$ on the same segmentation map. The neighborhood is *punctured*, i.e. the center pixel to be predicted, j, is missing. Intuitively, this classifier learns to repair segmentation maps produced by the segmentation learner $f_s(\theta_s)$.

Just as the segmentation learner, the context learner $f_c(\theta_c)$ is first pre-trained in a purely supervised way on the synthetic images (as described in Section 5.2.1, see Figure 61). The pre-training of the context learner is divided into two steps. First,

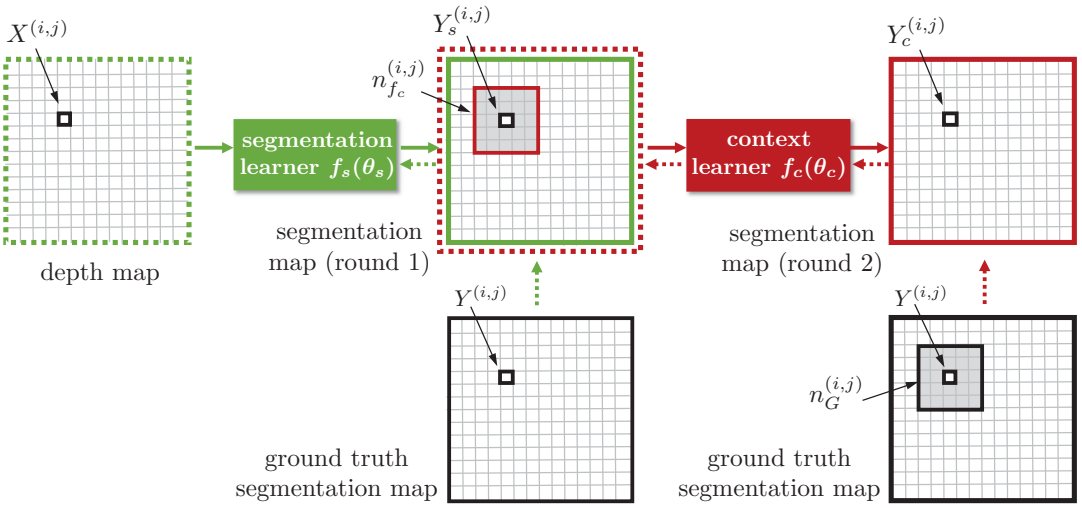


Figure 61 – The two learning pathways involving a direct learner $f_s(\theta_s)$ and a context learner $f_c(\theta_c)$. The context learner operates on punctured neighborhood maps $n^{(i,j)}$, where the (to be predicted) middle pixel is missing.

ground truth label maps are used as the training input. After convergence of the segmentation learner $f_s(\theta_s)$, its output is used instead for input to the context learner, and the context learner is fine-tuned to cope with realistic output segmentation maps.

The objective function Q_c for training the context learner $f_c(\theta_c)$ is also a negative log-likelihood loss for pixelwise classification. Learning of θ_c proceeds in two steps. First, ground truth segmentation maps are fed to the learner, denoted as $n_G^{(i,j)}$, minimizing NLL:

$$Q_c^{(1)}(\theta_c \,|\mathcal{G}) = -\sum_{i=0}^{L}\sum_{j=0}^{M} \log P\left(Y_c^{(i,j)} = y^{(i,j)} \,\middle|\, N^{(i,j)} = n_G^{(i,j)}; \theta_c\right). \tag{128}$$

After convergence, in a second phase, segmentation maps produced by the segmentation learner $f_s(\theta_s)$ are fed into the context learner, denoted as $n_{f_s}^{(i,j)}$.

$$Q_c^{(2)}(\theta_c \,|f_s(\mathcal{D}_S)) = -\sum_{i=0}^{L}\sum_{j=0}^{M} \log P\left(Y_c^{(i,j)} = y^{(i,j)} \,\middle|\, N^{(i,j)} = n_{f_s}^{(i,j)}; \theta_c\right). \tag{129}$$

During this step, parameters of the segmentation learner θ_s are kept fixed.

The context learner $f_c(\theta_c)$ learns a prior distribution over segmentation maps. It could very well be used during the testing phase to improve the classification accuracy of the method, as it is frequently done in auto-context models [234]. However, we preferred to remove this classifier from the testing pipeline to maximize speed and minimize latency.

Instead, the context learner is used during training only to fine-tune the segmentation learner on real data. Since no ground truth is supposed to be available for real data, training at this step is unsupervised. We use the trained context learner $f_c(\theta_c)$ to create loss for training of the segmentation learner $f_s(\theta_s)$ without ground truth

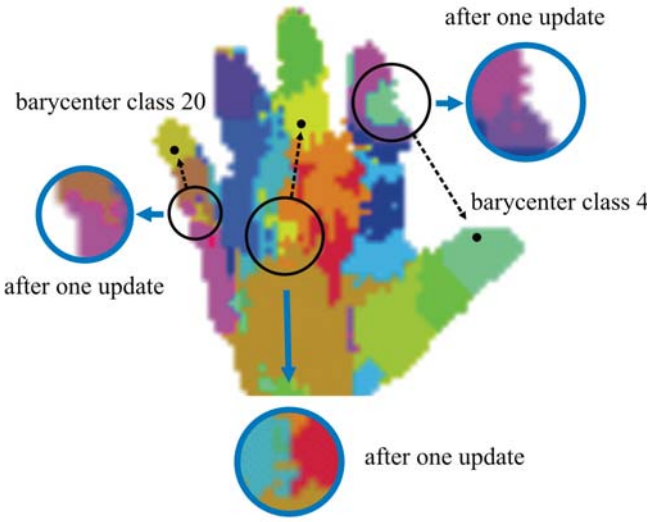


Figure 62 – Global structural information which can be extracted from a segmented image even if ground truth labels are **not** available. Small circles with thin black borders contain segmented pixels, which are far away from the principal pixel mass of the given hand part, indicated by the barycenter of the hand part. The global unsupervised loss term Q_{glb} punishes these results. Large circles with thick blue borders show the same content after a single parameter update of the segmentation learner $f_s(\theta_s)$ using this global term Q_{glb}.

labels. The underlying hypothesis here is that the domain shift between synthetic data and real data is smaller in the target domain (segmentation maps) than it is in the input domain (depth maps).

In the proposed unsupervised adaptation strategy, following optimization criteria are based on, first, consistency of each predicted pixel class with its local neighborhood on the output segmentation map Y_s (produced by the segmentation learner and estimated from the output of the context learner) and, second, global compactness and homogeneity of the predicted hand segments (see Figure 60).

The unsupervised loss function Q_u measures structural properties of the predicted segmentation at two different scales, either on context (at a neighborhood level), or globally on the full image. The estimated properties are then related to individual pixelwise loss:

$$Q_u = f(Q_{loc}, Q_{glb}), \tag{130}$$

where Q_{loc} is a *local term*, Q_{glb} is a *global term*, and f denotes the way how they are integrated. The definitions of each term are discussed in the rest of this section.

5.3.1 *Local structural term*

The term Q_{loc} in equation (130) is introduced for capturing local structure. It favors predictions which are consistent with other outputs in a local neighborhood. In particular, it favors those classification results, where the segmentation learner $f_s(\theta_s)$ *agrees* with the context learner $f_c(\theta_c)$.

This term is formulated as a conditional negative likelihood loss. For each given pixel, if both classifiers $f_s(\theta_s)$ and $f_c(\theta_c)$ (the latter one operates on the output of the former one) agree on the same label, this pixel is used to update parameters θ_s of the segmentation learner and the error is minimized using the classical NLL scenario treating the predicted label as corresponding to the ground truth:

$$Q_{loc}(\theta_s|\mathcal{D}_R) = -\sum_{j=0}^{M} \mathbb{I}_{Y_s^{(i,j)}=y_c^{(i,j)}} \log P\left(Y_s = y_c^{(i,j)} \middle| I^{(i,j)}; \theta_s, \theta_s\right), \tag{131}$$

where $\mathbb{I}_\omega$=1 if ω holds and 0 else. In this case the parameters θ_c of the context learner remain unchanged. The indicator function is non-smooth with respect to the parameters. For backpropagation, we treat it as a constant once both segmentation maps are computed (similar to the way max pooling is dealt with in classical deep networks)

We would like to stress again, that during *test time* the context learner is not used, and outputs of the segmentation learner $f_s(\theta_s)$ are considered as the final predictions. This way, we keep the test architecture as simple as possible and focus mainly on developing an effective *training* procedure to learn meaningful data representations that are robust to noise typical of real-world data.

5.3.2 *Global structural term*

The second term Q_{glb} in equation (130) captures global structure. It favors output predictions, which fit into global image statistics, and penalizes the ones, which do not, by changing parameters of the segmentation learner $f_s(\theta_s)$ in the direction of a more probable class. Technically, this term aims on to minimize variance (in terms of pixel coordinates) of each hand segment. Figure 62 illustrates the type of errors which can be corrected in this strategy, to give the reader an intuitive understanding of the ongoing process.

Although ground truth labels are not available for the real images dealt with in this part, there is still *intrinsic structural information*, which can be extracted from a segmented image and which is related to strong priors we can impose on the segmentation map. In particular, unlike in general segmentation problems, this term exploits an assumption that body and hand part segmentation maps contain a single connected region per hand part label (ignoring cases of strong partial self-occlusion, which are extremely rare).

In Figure 62, small circles with thin black borders contain segmented pixels which are not connected to the principal region of the given hand part, indicated by the barycenter of the pixels of this hand part. The global unsupervised loss term Q_{glb} punishes these results. Large circles with thick blue borders show the same content, produced by the segmentation learner $f_s(\theta_s)$ after a single iteration of the adaptation process using Q_{glb}.

We formalize this concept as follows. For each class k present in the output map Y_s, produced by the segmentation learner, barycentric coordinates of the corresponding segment are calculated:

$$\mathbf{R}_k = \frac{\sum_{j:Y_s^{(i,j)}=k} P(y_s^{(i,j)}|I^{(i,j)})\mathbf{r}^{(i,j)}}{\sum_{j:Y_s^{(i,j)}=k} P(y_s^{(i,j)}|I^{(i,j)})}, \tag{132}$$

where pixel coordinates, in vector form, are denoted as $\mathbf{r}^{(i,j)}$.

If $\left|\mathbf{r}^{(i,j)} - \mathbf{R}_k\right| > \tau$, i.e. the pixel (i,j) is close enough to its barycenter (τ is estimated from the labeled synthetic data), then the pixel is considered as correctly classified and used to update parameters of the segmentation learner θ_s. The loss function term for one pixel (i,j) is given as follows:

$$Q_{glb}^{+}(\theta_s \left| y_s^{(i,j)} \right.) = -F_{y_s}^{(i)} \log P\left(Y_s = y_s^{(i,j)} \left| I^{(i,j)}, \theta_s, \theta_c \right.\right), \tag{133}$$

where $F_k^{(i)}$ is a weight related to the size of class components:

$$F_k^{(i)} = |\{j : Y_s^{(i,j)} = k\}|^{-\alpha}, \tag{134}$$

and $\alpha>0$ is a gain parameter. In the opposite case, when $\left|\mathbf{r}^{(i,j)} - \mathbf{R}_k\right| \leqslant \tau$, the current prediction is penalized and the class γ corresponding to the closest segment in the given distance τ is promoted:

$$Q_{glb}^{-}(\theta_s \left| y_s^{(i,j)} \right.) = -F_{\gamma}^{(i)} \log P\left(Y_s = \gamma \left| I^{(i,j)}, \theta_s, \theta_c \right.\right), \tag{135}$$

where

$$\gamma = \operatorname{argmin}(|\mathbf{r}^{(i,j)} - \mathbf{R_k}|). \tag{136}$$

This formulation is related to the k-means algorithm. However, data points in our setting are embedded in two spaces: the space spanned by the network outputs (or, alternatively, feature space), and the 2D geometric space of the part positions. Assigning cluster centers requires therefore optimizing multiple criteria and distances in heterogeneous spaces. Other clustering costs could also be adopted.

5.3.3 *Integrating local and global structure*

Finally, local structure and global structure are fused, emphasizing agreement between both terms. In particular, activation of the penalizing global term (which favors parameters pushing a pixel away from currently predicted class) is confirmed by similar structural information captured by the local term ($Q_{loc}=0$):

$$Q_u = \beta_{loc} Q_{loc} + \beta_{glb} \begin{cases} Q_{glb}^{+} & \text{if } \left|\mathbf{r}^{(i,j)} - \mathbf{R}_k\right| \leqslant \tau, \\ Q_{glb}^{-} & \text{if } \left|\mathbf{r}^{(i,j)} - \mathbf{R}_k\right| > \tau \text{ and } Q_{loc} = 0, \\ 0 & \text{else,} \end{cases} \tag{137}$$

where β_{loc} and β_{glb} are weight coefficients, defined empirically.

Fusing the two terms, Q_{loc} and Q_{glb}, in a non-trivial way is essential as they are acting in an adversarial way. The local term alone leads to convergence to a trivial solution when all pixels in the image are assigned to the same class by both classifiers. The global term favors multi-segment structure composed of homogeneous regions, while exact shapes of the segments may be distorted as to not satisfy the desirability of of compactness. The two terms acting together, as well as mixing the labeled and unlabeled data, allows the classifier to find a balanced solution.

5.4 EXPERIMENTAL RESULTS

In this section, we will begin by discussing the performance of the proposed *weakly supervised* transductive adaption strategy, first in terms of quality of produced segmentation maps from real images. After, we will show how the obtained intermediate representations of hand poses can improve the quality of hand joint localization. Finally, at the end of the section, we will also provide empirical evaluation of the framework exploiting the *unsupervised adaptation pipeline*.

However, before we proceed with this evaluation, let us discuss aspects of the datasets we considered, and justify the architectural solutions chosen for each of the learners.

5.4.1 *Data collection and rendering*

As we have described previously, the proposed framework requires a large amount of synthetic images, as well as real data for model adaptation and, optionally, localization of hand joints.

— **Synthetic dataset.** For this project, we have created a vast collection of 170 974 synthetic training samples, including both normalized 8 bit depth maps and ground truth segmentations with image resolution 640×640 pixels. An additional set of 6 000 images is used for validation and testing.

The rendering was performed using the *Poser* commercial software and a number of highly detailed 3D human models, including both genders (see Figure 63). Hand shapes, proportions, poses and orientations were generated with a random set of parameters. Each pose was captured from 5 different camera view points chosen arbitrarily for each frame.

During this process, the 3D renderer was automated by a Python script, sampling the parameters randomly from predefined ranges, which were defined in accordance with anatomical constraints (set empirically, see Table 17). For each rigid segment of the arm, including the hand, we modified its *twist*, *side-side* and *bend* angles, and scaled its length and thickness with a random coefficient from the range $[0.95, 1.05]$. Furthermore, we introduced additional scale coefficients for groups of bones, which needed to be modified together in order to look realistic (for example, lengths of different fingers are correlated). Complex hand parameters, built in the software, such as *grasp* and *spread* were also sampled randomly from specified ranges ($[-10, 10]$ and $[0, 70]$, respectively). Yaw and pitch of the

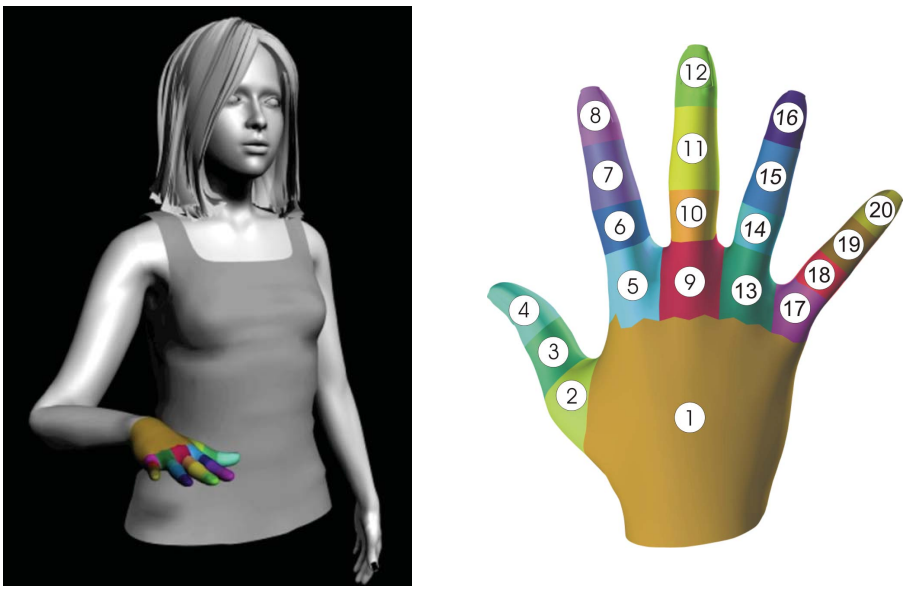


Figure 63 – On the left: an example of a 3D human model used for data generation in *Poser*, on the right: labeling order of the ground truth hand segments.

Object	minBend	maxBend	minSide	maxSide	minTwist	maxTwist
Shoulder	0	75	−15	65	0	50
ForeArm	0	110	−10	10	−85	85
Hand	−70	50	−20	20	−20	20
Index0	−10	10	−15	15	−15	10
Index1	−30	70	−15	10	−5	5
Index2	−10	115	0	0	−4	4
Index3	−10	110	0	0	0	0
Mid0	−10	10	−10	10	−10	10
Mid1	−30	70	−10	10	−5	5
Mid2	−10	115	0	0	−4	4
Mid3	−8	110	0	0	0	0
Ring0	−10	10	−9	13	−10	10
Ring1	−30	70	−13	10	−4	4
Ring2	−10	115	0	0	0	0
Ring3	−12	110	0	0	0	0
Pinky0	−10	15	−10	15	−13	7
Pinky1	−30	70	−18	15	−4	4
Pinky2	−10	115	0	0	0	0
Pinky3	−12	110	0	0	0	0
Thumb	−30	20	−15	30	−35	50

Table 17 – Anatomical constraints for hand elements, used for the data generation in *Poser*.

camera were chosen from a discrete range $[0°, 180°]$ with an angular granularity of $10°$.

An example of a generated ground truth segmentation map is shown in Figure 63, where the hand is divided into 20 segment classes. Background pixels were set to 0 and assigned with a zero class label.

— **Real dataset [unsupervised pipeline]**. For our *earlier* experiments on the unsupervised adaptation, we have collected an unlabeled our own real set consisting of $3,000$ images captured with the Kinect depth sensor. In order to evaluate the segmentation performance on real-world data, we have manually densely annotated 50 test samples from this data.

— **Standard benchmark [weakly-supervised pipeline]**. In the *more recent* experiments, which included the proposed weakly supervised adaptation, as well as estimation of hand joint positions, we used the NYU Hand Pose Dataset [296] [1]. Unfortunately, this corpus had not yet been released at the time when the first group of experiments on unsupervised adaptation was performed. As we have previously mentioned in Section 2.3.1, it consists of $72,757$ training and $8,252$ test images, annotated with positions of 36 key points, among which 14 are typically used for evaluation of state-of-the-art hand pose estimation frameworks.

To evaluate the performance of various segmentation and restoration methods in the weakly supervised framework, we have manually segmented 100 images from this dataset (referred to as NYU-100 in the following discussion). These 100 images were solely used for evaluation and never entered any training procedure.

5.4.2 *Experimental setup*

The summary of architectural hyper-parameters of all neural models, discussed in this chapter, is provided in Table 18. Both segmentation and regression architectures (as well as the context learner) have ReLU activation functions at each layer and employ batch normalization [136]. During the test time, the regression learner uses the batch normalization parameters estimated on the training data, however, in the segmentation network, batch normalization is performed across all pixels from the same image, for both training and test samples.

All deep learning architectures were implemented using the Torch7 library [60]. Approximated NN-search using KD-trees (for patchwise restoration) was performed using the FLANN library [197].

Below, we discuss certain technical aspects of the explored segmentation and recognition frameworks in more detail.

5.4.2.1 *Data handling*

In the weakly supervised framework, we extract hand images of normalized metric size (taking into account depth information) and normalize them to 48×48 pixels. In the earlier experiments on the unsupervised pipeline, higher resolution bounding boxes of 80×80 pixels were used instead.

1. http://cims.nyu.edu/~tompson/NYU_Hand_Pose_Dataset.htm

All images were first preprocessed by local contrast normalization with a kernel size of 9×9. For supervised training of the segmentation learner (and the context learner, if applicable), the full set of 170,974 synthetic training images was used.

In the weakly supervised framework, a third of this set was also used to extract patches for the patch alignment mapping f_{nn}, giving a dictionary of 36M patches of size 27×27 extracted from 56,991 images.

Local integration, as given in equation (127), was performed using sliding windows of size $W \times W = 17 \times 17$ pixels.

5.4.2.2 *Training parameters*

The learning rate for the regression learner was initially set to 0.05 with linear decay corresponding to 1% of the initial value. For the segmentation learner, we set the learning rate to 0.1 and employed exponential weight decay with a value of 10^{-5}. Batch size of 64 was used for the regression learner, while the segmentation model was trained on each image separately, considering its pixels as a batch. Even though the number of pixels belonging to different classes is typically unbalanced, initially normalizing the gradients accordingly did not affect the final segmentation performance after convergence and was removed in the final implementation. All predictors were trained by gradient descent using the Adam [157] update rule.

At the segmentation step, the training procedure was started with purely supervised learning by backpropagation which proceeded until 50% of the synthetic training data was seen by the network. From this moment on, we replaced 10% of the training set with unlabeled real world samples.

In the unsupervised adaptation framework, the parameters balancing different terms were set to β_{loc}=0.1, β_{glb}=1.2.

5.4.2.3 *Segmentation learner: remarks on the architecture*

As we have seen in Section 5.2.1, the direct and the context learners are based on a convolutional network architecture and have the same general structure (see Figure 56), which includes three consecutive convolutional layers F_1, F_2 and F_3 with rectified linear activation units (ReLU). Layers F_1 and F_2 are followed by 2×2 max pooling and reduction.

The structure of these learners is motivated by the idea of performing efficient pixelwise image segmentation preserving original resolution of the input, and is inspired by the *OverFeat* networks which were proposed for object detection and localization [262]. As opposed to most existing methods for scene labeling, instead of randomly sampling pixels (or patches), training of the segmentation learner $f_s(\theta_s)$ is performed image-wise, i.e. all pixels from the given image are provided to the classifier at once and each pixel gets assigned with an output class label based on information extracted from its neighborhood.

Applying the convolutional classifier with pooling/reduction layers to an image in the traditional way would lead to loss in resolution by a factor of 4 (in the given configuration). On the other hand, simply not reducing the image resolution will prevent higher layers from learning higher level features, as the size of the filter support does not grow with respect to the image content. To avoid this dilemma, we

Network architectures		
Layer	Filter size / units	Pooling
Segmentation learner f_s (also f_c)		
Depth input	48×48	-
Convolutional layer 1	32×5×5	2×2
Convolutional layer 2	64×5×5	2×2
Convolutional layer 3	128×5×5	1×1
Hidden layer	500	-
Hidden layer	500	-
Output	20×48×48	-
Regression learner f_r		
Depth input	24×24	-
Convolutional layer c1	32×3×3	2×2
Convolutional layer c211, c221	16×1×1	
Convolutional layer c212, c222	8×3×3	
Pooling p231	-	2×2
Convolutional layer 232	8×1×1	
Convolutional layer 241	16×1×1	
Hidden layer fc	1200	-
Output	14×3	-

Table 18 – Hyper-parameters chosen for the deep networks.

employ specifically designed splitting functions originally proposed for image scanning in [106] and further exploited in *OverFeat* networks [262]. Intuitively speaking, each feature map at a given resolution is reduced to four different feature maps of lower resolution using max pooling. The amount of elements is preserved, but the resolution of each map is lower compared to the maps of previous layers.

In more detail, let us consider the output of the first convolutional layer F_1 of the network. Once the output feature maps are obtained, four virtual extended copies of them are created by zero padding with 1) one column on the left, 2) one column on the right, 3) one row on top, 4) one row in the bottom. Therefore, each copy will contain the original feature map, but shifted in four different directions. On the next step, we apply max pooling (2×2, with stride 2×2) to each of the extended maps producing four low-resolution maps. By introducing the shifts, pixels from all extended maps combined together can reconstruct the original feature map as if max pooling with stride 1×1 had been applied. This operation allows the network to preserve results of all computations for each pixel on each step and, at the same time, perform the necessary reduction, resulting in a significant speed up during training and testing.

Weakly supervised transductive adaptation: evaluation

Method	Training data	per pixel	per class
Fully supervised training only	synth.	51.03	39.38
Unsupervised adaptation	synth. + real	52.16 (+1.13)	44.17 (+4.79)
Weakly supervised adaptation	synth. + real	**57.18** (+6.15)	**47.20** (+7.82)

Table 19 – The contribution of the weakly-supervised adaptation on the segmentation accuracy. The reported results were obtained on 100 manually labeled images of the NYU Hand Pose Dataset [296] (NYU-100).

After pooling, convolutions of the following layer F_2 are applied to all four low-resolution maps separately (but in parallel). The same procedure is repeated after the second convolutional layer F_2, where each of four branches is split again into four, producing 16 parallel pathways overall. If necessary, the algorithm can be extended to an arbitrary number of layers and employed each time when reduction is required.

All outputs of the third convolutional layer F_3 are flattened and classified with an MLP, producing a label for each pixel. Finally, the labels are rearranged to form the output segmentation map corresponding to the original image.

The direct and the context learners have the same architecture with the only difference, that the middle parts of the first layer filters of the context learner are removed. It helps to prevent the network from converging to a trivial solution, where a pixel label is produced by directly reproducing its input. This is especially important on the initial training stage, when the context learner is trained on ground truth segmentation maps.

5.4.3 *Weakly supervised adaptation of $f_s(\theta_s)$: empirical evaluation*

In this set of experiments on the *weakly supervised adaptation* we used a manually annotated subset of 100 images from the newly released NYU Hand Pose Dataset [296] for testing and the training subset from the same corpus for modal adaptation.

Table 19 shows the contribution of weakly supervised training, where sparse annotation (joint positions) are integrated into the training process of the segmentation learner $f_s(\theta_s)$. This procedure achieves an improvement of +6.15 percentage points per pixel and +7.82 percentage points per class, an essential step in learning an efficient intermediate representation.

Figure 64 shows improvement in the classification accuracy for each pixel class, which is consistent for all hand parts. Examples of produced segmentations are provided in Figure 65.

The performance of the offline step of *restoration* is evaluated in Table 20. We emphasize once more that training was performed on synthetic while we test on real images, thus of an unseen distribution. This domain shift is clearly a problem, as accuracy on the synthetic dataset is very high, 90.16%. Using patchwise restoration of

Figure 64 – Average segmentation accuracy per class obtained with the supervised method (in black) and after the weakly supervised adaptation of the segmentation learner $f_s(\theta_s)$ to the real data (in red). These results were obtained using 100 manually labeled images from the NYU Hand Pose Dataset [296] (NYU-100). The horizontal axis shows numbers of hand segments (see Figure 63).

the predicted real patches with the large dictionary of synthetic patches gives a performance increase of +3.5 percentage points per pixel and +7 percentage points per class. This corroborates our intuition that the intermediate representation carries important structural information. Integration of patch-labels over a local window with equation (127) is essential, pure NN-search without integration performs poorly. The same table highlights that the weakly supervised pipeline generally outperforms the unsupervised solution.

In order to optimize the restoration step, we also compare local integration equation (127) to potentially more powerful regularization methods by implementing a CRF-like discrete energy function.

In this case, instead of choosing the nearest neighbor in patch space for each pixel as described in equation (126), a solution is searched which satisfies certain coherence conditions over spatial neighborhoods. To this end, we create a global energy function $E(x)$ defined on a 2D-lattice corresponding to the input image to restore:

$$E(x) = \sum_i u(x_i) + \alpha \sum_{i \sim j} b(x_i, x_j), \tag{138}$$

where $i \sim j$ indices neighbors i and j. Each pixel i is assigned a discrete variable x_i taking values between 1 and $N=10$, where $x_i = l$ signifies that for pixel i the $l-th$ nearest neighbor in patch space is chosen. For each pixel i, a nearest neighbor search is performed using KD-trees and a ranked list of N neighbors is kept defining the label space for this pixel. The variable N controls the degree of approximation of the model, where $N=\infty$ allows each pixel to be assigned every possible patch of the synthetic dictionary.

The unary data term $u(x)$ guides the solution towards approximations with low error. It is defined as the Hamming distance between the original patch and the synthethic patch.

Weakly supervised transductive adaptation: restoration performance		
Method	per pixel	per class
No restoration	51.03	39.38
NN-search – no integration	48.76 (−2.27)	39.72 (+0.34)
NN-search – integration with equation (127)	**54.55** (+**3.52**)	**46.38** (+**7.00**)
CRF – Potts-like model	53.10 (+2.07)	43.64 (+4.26)
CRF – Hamming distance on overlapping area	52.45 (+1.42)	42.68 (+3.30)

Table 20 – Restoration (=segmentation) accuracy on 100 manually labeled images of the NYU Hand Pose Dataset [296] (NYU-100). This restoration step is used exclusively for training purposes and is not performed during the test time.

We tested two different pairwise terms $b(x_i, x_j)$:

— **Potts-like terms** – a Potts model classically favors equality of labels of neighboring sites. In our setting, we favor equality of the center pixels of the two patches assigned to x_i and x_j.

— **Patch-overlap distance** – the alternative pairwise term is defined as the Hamming distance between the two synthethic patches defined by x_i and x_j, in particular, the distance restricted to the overlapping area.

In this setup, inference was performed through message passing using the opengm library [6], and the hyper-parameter α was optimized through grid-search. Interestingly, local patch integration with equation (127) outperformed the combinatorial models significantly while at the same time being much faster.

To conclude this section, we would like to note that the absolute segmentation performance of 54.55% may seem to be low. However, in the pose estimation framework for the *full body*, described in [266], the body part segmentation accuracy corresponds to 60%, while this was a far simpler problem. However, this level of performance is reported to be sufficient for joint estimation, even solely from the segmentation maps. In our framework, unlike [266], the segmentation map is not the sole input for regression.

5.4.4 *Joint position estimation: empirical evaluation*

Finally, we aim to evaluate how the introduced intermediate representation, in the form of segmentation maps, can help joint position estimation.

Table 21 illustrates the effect of incorporating segmentation information on the regression learner performance. In the top part of the table we provide information on our own baselines, in the middle there are results of other deep learning methods reported in the literature. The error is expressed as mean distance in mm (in 2D or 3D) between the predicted position of each joint and its ground truth location. In comparison to a single network regressor, the 2D mean error was improved by 15.7%. The second best model, cascade regression, was inspired by the work of [297], where the initial rough estimation of hand joints positions is then improved by zooming in on the regions indicated by the first round of predictions.

Figure 65 – Different segmentation results: (a) input image; (b) output of the segmentation learner after supervised training; (c) after restoration; (d) output of the segmentation learner after joint training; (e) ground truth segmentation; (f) estimated joint positions. The image itself was *not* part of the training set.

The distribution of 2D and 3D accuracies reached by the best model is shown graphically in Figure 66. Examples of joint localization performed by the regression learner are given in Figure 67.

We need to mention here, that prior to augmenting the classical regression method with the unsupervised pipeline, we spent additional time on careful optimization of this baseline by tuning its architecture and the training regime. Introducing the batch normalization and Adam optimization appeared to be crucial and allowed us to reach higher performance than in the literature, even in the baseline case. Before that, we were able to reproduce the results originally reported in [215].

Naturally, the quality of the network outputs can be further improved by optimization through inverse kinematics, as it has been done, for example, in [296]. However, the focus of this work is to explore the potential of pure learning approaches with no priors enforcing structure on the output.

The bottom part of the table contains non deep learning methods. In a recent work by Tang et al. [290], where the optimization of hand pose estimation is formulated as an inverse kinematics problem, the authors report performance similar to [296] in terms of 2D UV-error (no error in mm provided).

Localization of hand joints				
Method	Mean 2D	Median 2D	Mean 3D	Median 3D
Direct regression	13.27	10.56	17.26	14.45
Cascade direct regression	12.56	9.94	16.88	14.10
Regression + segmentation	**11.18**	**8.67**	**14.94**	**13.62**
DeepPrior [215]	-	12.0†[134]	19.8†	-
Multi-Scale [215]	-	-	27.5†	-
Deep [215]	-	-	30.5†	-
Shallow [215]	-	-	34.5†	-
Tompson et al. [296]	7.05	6.54	21.0†[215]	-
Xu and Cheng [323]	-	58.0†[134]	-	-
Keskin et al. [152]	-	72.5†[134]	-	-

Table 21 – Joint position estimation error on the NYU dataset. † indicates that values were estimated from plots if authors do not provide numerical values (a reference indicates where performance was reported, if not in the original paper).

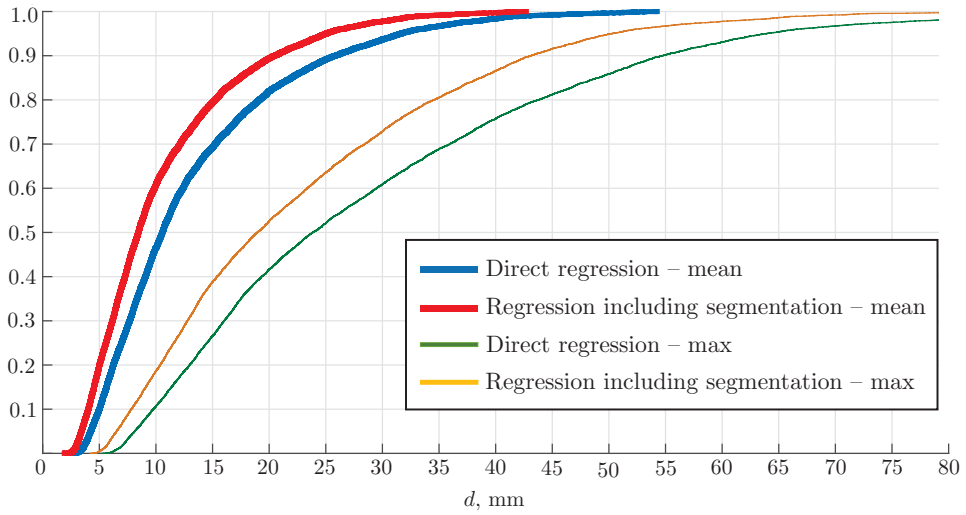


Figure 66 – Joint estimation accuracy: proportion of correctly detected joints as a function of a threshold on the mean error or max error. The regression model includes the segmentation outputs produced by the weakly adapted segmentation learner.

5.4.5 *Computational complexity*

For the weakly supervised framework, all models have been trained and tested using GPUs, except the patchwise restoration process which is pure CPU code and not used at test time. Estimating the pose of a single hand takes 31 ms if the segmentation resolution is set to 24×24 pixels (which includes the forward passes of both learners $f_s(\theta_s)$ (12 ms) and $f_r(\theta_r)$ (18 ms) and 58 ms for 48×48 segmentation

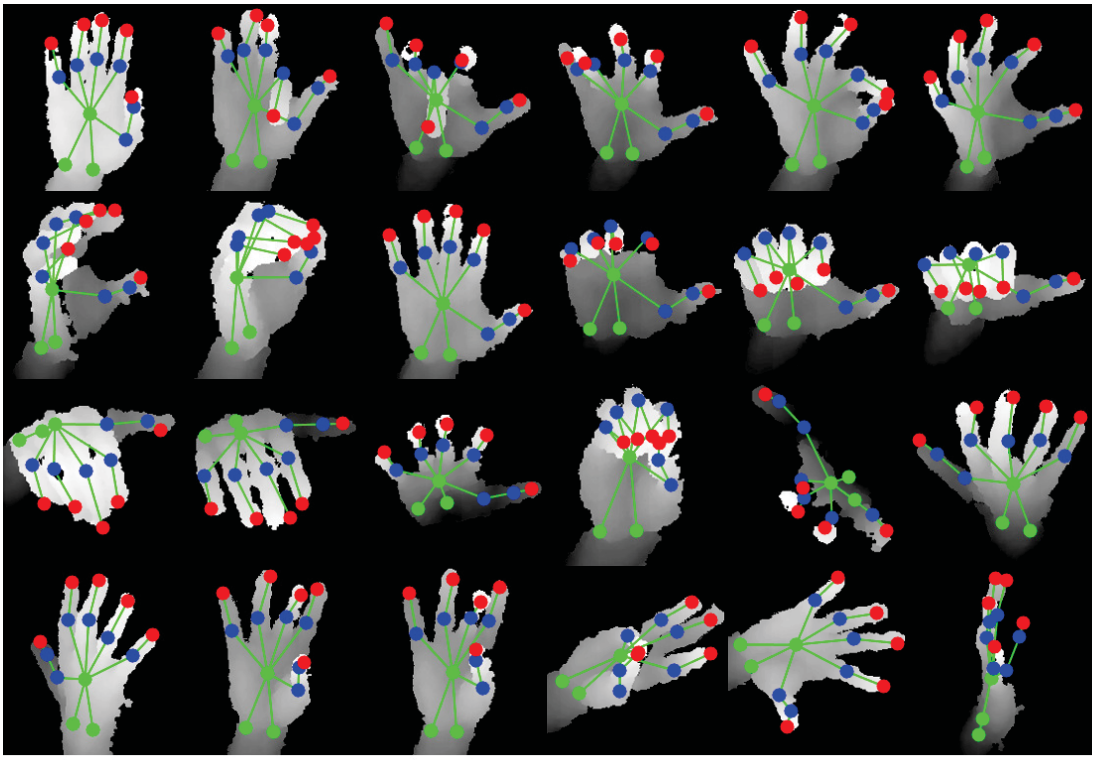


Figure 67 – Visualization of estimated hand skeletons produced by the regression learner $f_r(\theta_r)$. The regressor takes into account the intermediate representation produced by the segmentation network, which was tuned in a weakly supervised fashion.

outputs (40 ms) for $f_s(\theta_s)$, corresponding to the results reported in the experiments section).

Training of the segmentation network requires up to 24 hours to minimize validation error, while the regression network is trained in 20 min on a single GPU.

The reported results were obtained using a cluster configured with two E5-2620 v2 Hex-core processors, 64 GB RAM and three Nvidia GTX Titan Black cards with 6 GB memory per card.

5.4.6 *Unsupervised adaptation of $f_s(\theta_s)$: empirical evaluation*

The effect of the unsupervised transductive adaptation on the quality of segmentation was estimated using our own set of manually annotated real images. These experiments were conducted significantly earlier, when the NYU Hand Pose Dataset, used for evaluation of the weakly supervised pipeline, had not yet been released.

The comparative performance of different modifications of the pixel-wise segmentation learner $f_s(\theta_s)$, which were trained by including and excluding different unsupervised terms of the loss function is summarized in Table 22. In general, exploiting unlabeled real data for unsupervised training and network regularization has proven to be beneficial, especially for reconstruction of small segments (such as finger parts), which leads to a significant increase of average *per-class* accuracy. The

Unsupervised transductive adaptation				
Loss function	Training data	Test data	Accuracy	Average per class
Q_s (supervised baseline)	synth.	synth.	85.90%	78.50%
		real	47.15%	34.98%
$Q_s + Q_{loc} + Q_{glb}$ (semi-supervised, ours)	all	synth.	85.49%	78.31%
		real	**50.50%**	**43.25%**

Table 22 – The contribution of the unsupervised adaptation to the segmentation accuracy. The reported results were obtained on our own dataset of real depth images.

Figure 68 – Average segmentation accuracy per class obtained with the supervised method (in black) and after the unsupervised adaptation of the segmentation learner $f_s(\theta_s)$ to the real data (in red). These results were obtained using our own dataset of real images. The horizontal axis shows numbers of hand segments (see Figure 63).

bar plot on Figure 68 demonstrates significant improvement of recognition rates for almost all classes, except for the first, base *palm* class, which can be seen as a background for a hand image against which finger segments are usually detected. This illustration reflects the fact that more confident detection in the case of the unsupervised network adaptation comes together with a certain increase in the amount of false positives.

Table 23 illustrates the impact of *one update* of the segmentation learner parameters using different loss functions. This is estimated on a given image, which was used for computing the gradients (and averaged over the test set). We note that a combination of two competitive unsupervised terms (local and global) produces a more balanced solution than the same terms separately.

Unsupervised transductive adaptation: term-wise tests

Terms	Q_{loc}	$Q_{glb}{}^{+}$	$Q_{glb}{}^{+} + Q_{glb}{}^{-}$	$Q_{loc} + Q_{glb}{}^{+} + Q_{glb}{}^{-}$	Q_{sd}
Requires labels	no	no	no	no	yes
Gain in % points	+0.60	+0.36	+0.41	**+0.82**	+16.05

Table 23 – Performance improvement on a real image after a single iteration of updating parameters of the segmentation learner $f_s(\theta_s)$, using different supervised and unsupervised terms. Estimated as an average over 50 real images from our own dataset.

We would like once again to stress the importance of pre-training the segmentation and context learners on the synthetic data in order to be capable of producing structurally representative initial predictions for the unlabeled data. Furthermore, the frequency of supervised gradient updates during the final training stage should remain significant to prevent training from diverging.

Output segmentation maps produced by the proposed method, including unsupervised adaptation of the segmentation learner, are shown in Figure 69. Figure 70 visualizes several *problematic* images, where the baseline supervised network performs poorly. It demonstrates, that our algorithm is capable of finding regions, which would not have otherwise been reconstructed, and often leads to more consistent predictions and a reduction in the amount of noise in the segmentation maps.

5.5 CONCLUSION

In this chapter, we have presented a number of ideas on how hand pose estimation can be improved by introducing an intermediate representation in the form of segmentation maps. We showed that the additional structured information of this representation provides important cues for joint regression which leads to lower estimation error. Unlike most deep learning methods which require large amounts of labeled data, we do not assume that ground-truth segmentation of real data is available, and synthetic data plays in this case the primary role.

Our main contributions concern training procedures which may exploit i) context learning; ii) unsupervised learning of local and global structure, balancing a prior for large homogenous regions with pixel-wise accuracy; and iii) weakly supervised learning where the quality of image restoration is point-wise evaluated from weak annotations in the form of hand joint positions. In the last case, the weak supervision is dealt with by patch-wise alignment of real data to synthetic data performed in the space of the intermediate representation, exploiting its strong geometric and topological properties.

By integrating structural information into learning rather than the model architecture, we retain the advantages of very fast test-time processing and the ability to parallelize.

Finally, we would like to note, that in theory, even the earlier approach, which explored segmentation refinement from local and global consistency, could benefit

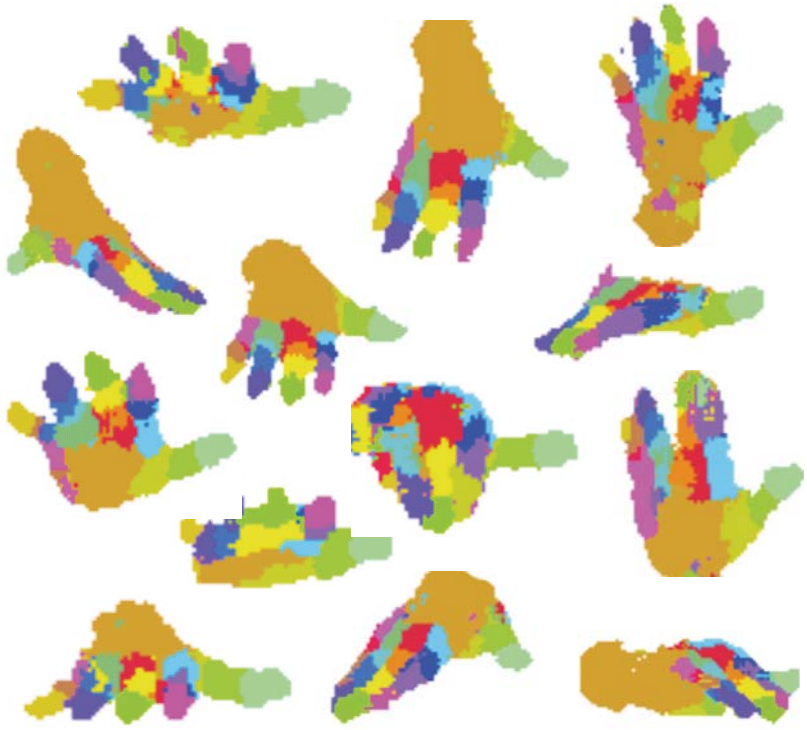


Figure 69 – Output segmentation maps produced by the segmentation learner $f_s(\theta_s)$ adapted to the real data using the unsupervised strategy. The samples are taken from our dataset of real images captured with the Kinect depth sensor.

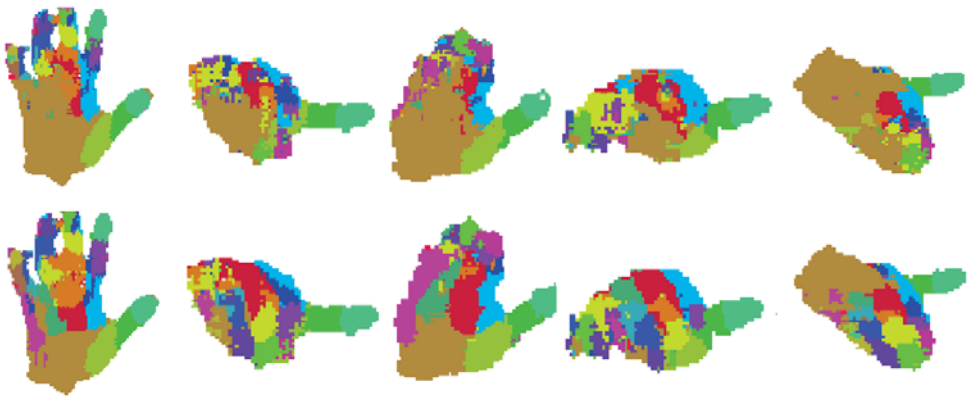


Figure 70 – Challenging examples. Top row: examples where the baseline method has difficulty in segmentation. Bottom row: outputs produced by the segmentation learner after the unsupervised adaptation.

from additional guidance by weak annotations. At the same time, the patch-wise restoration could be used alone, in unsupervised fashion, with no quality additional check. In the latter case, however, the restoration process typically causes more dramatic changes in the appearance of the segmentation maps, which, in the case of failure, may result in degradation of the pretrained model. Finally, other restoration strategies, such as impainting or more complex model based optimization can be incorporated to further improve the quality of produced segmentation maps.

Why are people born? Why do they die? Why do they want to spend so much of the intervening time wearing digital watches?

— Douglas Adams, The Ultimate Hitchhiker's Guide to the Galaxy

And starting today, all passwords must contain letters, numbers, doodles, sign language and squirrel noises.

— Scott Adams, Dilbert

6 AUTHENTICATION FROM MOTION

In this chapter, we present a large-scale study, which investigates the potential capability of temporal deep neural networks in interpreting natural human kinematics, and introduce the first method for active biometric authentication with mobile inertial sensors. Our patners at Google have created a first-of-its-kind dataset of human movements, passively collected by 1500 volunteers using their smartphones daily over several months. We (1) compare several neural architectures for efficient learning of temporal multi-modal data representations, (2) propose an optimized shift-invariant dense convolutional mechanism (DCWRNN) and (3) incorporate the discriminatively-trained dynamic features in a probabilistic generative framework taking into account temporal characteristics. Our results demonstrate that human kinematics convey important information about user identity and can serve as a valuable component of multi-modal authentication systems. Finally, we show that the proposed method can be successfully applied in the visual context, for example, for gesture recognition.

6.1 INTRODUCTION

Problems such as action and gesture recognition, as well as pose estimation, are all examples of *classical* computer vision tasks, which benefit from hundreds of vision and machine learning research teams exploring their territory. As we have seen in the state-of-the-art review (Chapters 2 and 3), *image*- and *speech*-specific feature descriptors have been designed and polished for decades, while the deep learning community has already made tremendous progress in *learning* visual and audio representations.

At the same time, there exist numerous areas of research where *analytical* (traditional) descriptors have not (yet) matured, and therefore learning data representations is the main hope to leap beyond the current level of understanding of underlying processes. In this sense, deep learning opens up new frontiers for identifying patterns in data which is not completely understood.

However, so far applied deep learning research has been mostly driven by the demand in *web-based* applications, being stimulated by an immediate availability

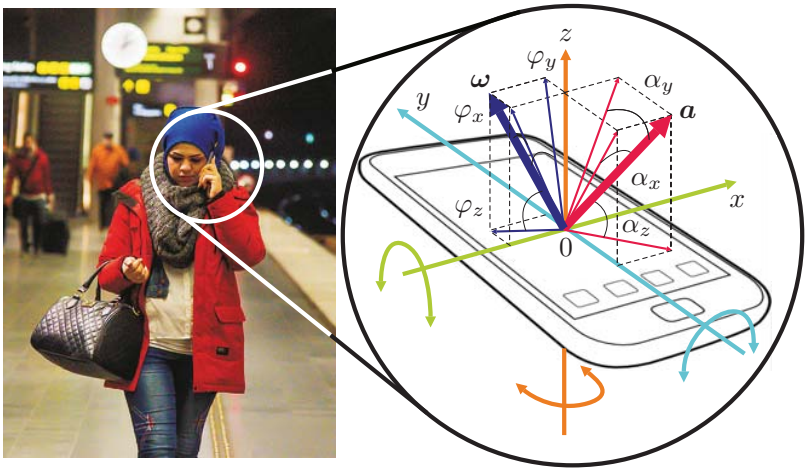


Figure 71 – In our mobile setting, an accelerometer captures linear acceleration and a gyroscope provides angular velocity (photo taken from [214]).

of practically *infinite* amounts of training data. Lack of such vast data collections, however, often becomes the first and sometimes fundamental obstacle for applying deep learning methods in other domains.

In this work, we begin the exploration of terra incognita and raise the question of how to understand and interpret natural human kinematics *at scale*, using learned representations of data extracted from *inertial physical sensors* (see Fig. 71). Can a smartphone, held in your hand, for example, recognize you by your motion patterns, whether or not you are interacting with the device?

This particular formulation is motivated by the idea of facilitating the daily interaction between people and their devices, while still providing security guarantees. For the billions of smartphone users worldwide, remembering dozens of passwords for all services we need to use and spending precious seconds on entering pins or drawing sophisticated swipe patterns on touchscreens becomes a source of frustration. For this reason, in recent years, researchers in different fields have sought fast and secure authentication alternatives that would make it possible to remove this burden from the user [273, 35].

Historically, biometrics research has been hindered by the difficulty of collecting data, both from a practical and legal perspective. For this reason, previous studies have been limited to tightly constrained lab-scale data collection, poorly representing real world scenarios: not only due to the limited amount and variety of data, but also due to essential *self consciousness* of participants which are asked to perform the tasks in laboratory conditions.

In response, our partners at Google ATAP created an unprecedented dataset of *natural* prehensile movements (i.e. those movements in which an object is seized and held, partly or wholly, by the hand [207]) collected by 1,500 volunteers over several months of daily use.

Apart from data collection, the main challenges in developing a *continuous* authentication system for smartphones are (1) efficiently learning task-relevant representations of noisy inertial data, and (2) incorporating them into a biometrics setting,

characterized by limited resources. Limitations include low computational power for model adaptation to a new user and for real-time inference, as well as the absence (or very limited amount) of *negative* samples.

In response to the above challenges, in this chapter we will be talking about developing a non-cooperative and non-intrusive method for on-device authentication based on two key components, namely temporal feature extraction by deep neural networks, and classification via a probabilistic generative model.

Unlike in gesture recognition and hand pose estimation, where we have been focusing mostly on learning representations from a *single* frame or from *spatio-temporal* dynamic blocks, here we will go deeper into exploration of explicit *temporal* deep learning models. For this purpose, we start by assessing several popular deep architectures, which include one-dimensional convolutional nets and recurrent neural networks for feature extraction.

Furthermore, apart from the application itself, our main contribution of this work is in developing a *shift-invariant* temporal architecture capable of modeling multiple temporal scales. This new model has been specifically developed for the biometric application at hand. However, it is applicable to other problems, in particular in computer vision. We will provide additional experimental validations on the gesture recognition problem discussed in Chapter 4 at the end of this chapter.

Finally, with respect to the original biometric application, we model the extracted features in a generative framework based on Gaussian Mixture Models specific to temporal representations.

6.1.1 *Authentication from motion: historical remarks*

To give appropriate credit, we must say that exploiting wearable or mobile inertial sensors for authentication, action recognition or estimating parameters of a particular activity has been previously explored in different contexts.

Gait analysis has attracted significant attention from the biometrics community as a non-contact, non-obtrusive authentication method resistant to spoofing attacks. A detailed overview and benchmarking of existing state-of-the-art is provided by Nickel et al. [213]. Derawi et al. [75], for example, used a smartphone attached to the human body to extract information about walking cycles, achieving 20.1% equal error rate. Tessendorf et al. analysed parameters of periodic motion provided by inertial sensors to evaluate and improve rowing skills [293].

There also exist a number of works which explore the problem of activity and gesture recognition with motion sensors, including methods based on deep learning. In [92] and [39], exhaustive overviews of preprocessing techniques and manual feature extraction from accelerometer data for activity recognition are given. Perhaps most relevant to this study is [236], the first to report the effectiveness of RBM-based feature learning from accelerometer data, and [34], which proposed a data-adaptive sparse coding framework.

Convolutional networks have been explored in the context of gesture and activity recognition from inertial data [81, 335]. Lefebvre et al. [173] applied a bidirectional LSTM network to a problem of 14-class gesture classification, while Berlemont et al. [32] proposed a fully-connected Siamese network for the same task.

Finally, motion data has been used to provide additional characteristics for *touch* events, such as clicking, swiping and typing on touchscreens [273].

We believe that multi-modal frameworks, however, are more likely to provide meaningful security guarantees for the task of biometric authentication. In this context, Khoury et al. [156] have been exploiting a combination of face recognition and speech, while earlier Vildjionauite et al. [304] combined voice with gait.

6.2 METHOD OVERVIEW

The ultimate goal of the application presented in this chapter is to discriminate between *authentic* users and *impostors* based on a time series of inertial measurements. For this purpose, we propose a method which is based on two main components:

— a **feature learning** pipeline which associates each user's motion sequence with a collection of discriminative features,

— a lightweight **biometric model**, which takes those learned features as inputs and performs user verification on a mobile device.

While the feature extraction component is probably the most interesting aspect of our technique, we will delay its discussion to Section 6.4 to first define the general context of the task. Therefore, in the next sections, we begin by discussing the data format and the biometric model.

6.2.1 *Movement data*

As is the case of any rigid body, a smartphone, has 6 physical degrees of freedom, including translation and rotation along three perpendicular axes in its coordinate system. These 6 motion components are captured with an accelerometer and a gyroscope, that measure linear acceleration and angular velocity, respectively (shown in Figure 71). For this study, the force of gravity is removed from the accelerometer recordings by applying a high-pass Butterworth filter.

We note that in this work, we chose not to use the *magnetic field* data (even if such sensor is present in the device) due to its low reliability in indoor environments in proximity to other electronic devices, and its dependence of the orientation and the intensity of the ambient magnetic field on geolocation.

In our setting, each *reading* (i.e. a single measurement) in a synchronized raw input stream of accelerometer and gyroscope data has the form:

$$\{a_x, a_y, a_z, \omega_x, \omega_y, \omega_z\} \in \mathbb{R}^6,$$

where a represents linear acceleration, ω represents angular velocity and x, y, z denote projections of these vectors on corresponding axes, aligned with the smartphone. There are two important steps, which we take prior to feature extraction, namely *obfuscation* and *preprocessing*, which we discuss below.

6.2.1.1 *Obfuscation-based regularization*

First of all, we must note that there are many nuances in this framework and data collection, that mush be addressed in order to pose the biometric problem correctly.

In this context, it is important to differentiate between the notion of *device* and *user*. In the dataset we collected (Section 6.5), each device is assigned to a single user, thus all data is considered to be *authentic*. However, in real-world scenarios, such as theft, authentic and imposter data may originate from the *same* device. For this reason, it is crucial to keep in mind that the problem of *user* authentication should be disentangled from identification of a *device* signature.

In a recent study [68], it was shown that under lab conditions, a particular *device* could be identified by a response of its motion sensors to a given signal. This happens due to imperfection in calibration of a sensor resulting in constant offsets and scaling coefficients (gains) of the output, that can be estimated by calculating integral statistics from the data. Formally, the measured output of both the accelerometer and gyroscope can be expressed as follows [68]:

$$\mathbf{a} = \mathbf{b}_a + \mathrm{diag}(\boldsymbol{\gamma}_a)\tilde{\mathbf{a}}, \tag{139}$$

$$\boldsymbol{\omega} = \mathbf{b}_\omega + \mathrm{diag}(\boldsymbol{\gamma}_\omega)\tilde{\boldsymbol{\omega}}, \tag{140}$$

where $\tilde{\mathbf{a}}$ and $\tilde{\boldsymbol{\omega}}$ are real acceleration and angular velocity vectors, $\mathbf{b}_a$ and $\mathbf{b}_\omega$ are offset vectors and $\boldsymbol{\gamma}_a$ and $\boldsymbol{\gamma}_\omega$ represent gain errors along each coordinate axis.

To partially obfuscate the inter-device variations and ensure decorrelation of user identity from device signature in the learned data representation, we introduce low-level additive (offset) and multiplicative (gain) noise per training example. Following [68], the noise vector is obtained by drawing a 12-dimensional (3 offset and 3 gain coefficients per sensor) obfuscation vector from a uniform distribution:

$$\boldsymbol{\mu} \sim \mathcal{U}_{12}[0.98, 1.02].$$

In the context of a deep learning framework (employed for following feature learning), this procedure, if repeated and randomized for each training sample at each iteration, also serves as a source of data augmentation and can be seen as an additional regularization technique.

6.2.1.2 *Data preprocessing*

In addition to raw measurements, we use acceleration and angular velocity to extract a set of angles $\alpha_{\{x,y,z\}}$ and $\varphi_{\{x,y,z\}}$ describing the orientation of vectors $\mathbf{a}$ and $\boldsymbol{\omega}$ in the phone's coordinate system (shown in Figure 71):

$$\alpha_x = \arctan\left[\frac{a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right], \quad \alpha_y = \arctan\left[\frac{a_y}{\sqrt{{a_x}^2 + {a_z}^2}}\right], \tag{141}$$

$$\alpha_z = \arctan\left[\frac{a_z}{\sqrt{{a_y}^2 + {a_z}^2}}\right]. \tag{142}$$

Also, we compute magnitudes of these vectors, $|\mathbf{a}|$ and $|\boldsymbol{\omega}|$, and normalize each of the projected components x, y, z:

$$\|\mathbf{a}\| = \sqrt{a_x^2 + a_y^2 + a_z^2}, \tag{143}$$

$$a_{x,n} = \frac{a_x}{\|\mathbf{a}\|}, \qquad a_{y,n} = \frac{a_y}{\|\mathbf{a}\|}, \qquad a_{z,n} = \frac{a_z}{\|\mathbf{a}\|} \tag{144}$$

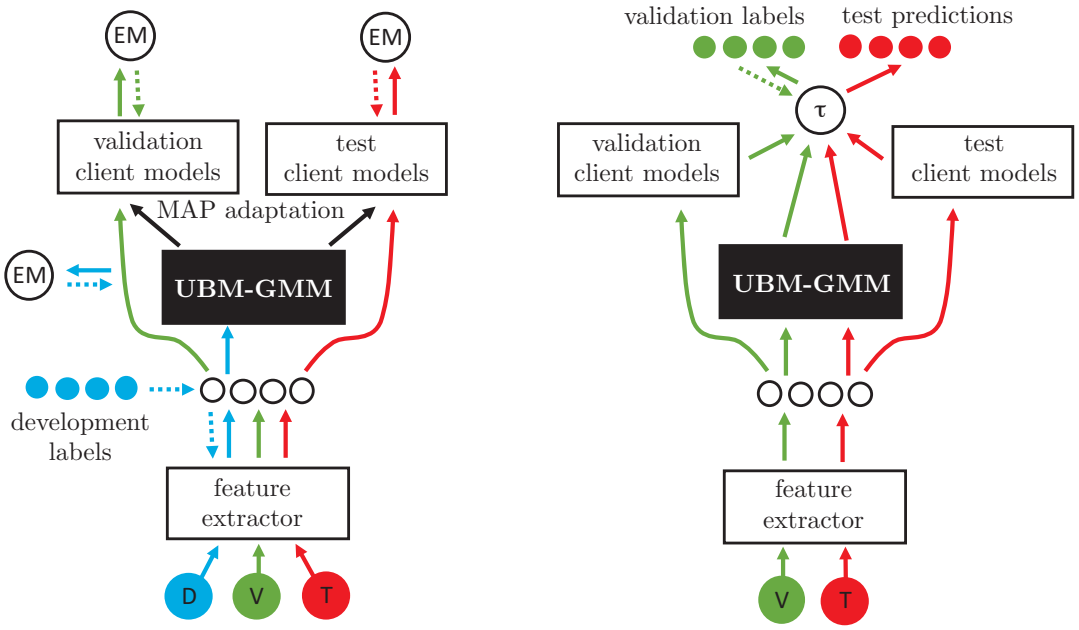


Figure 72 – Overview of the proposed biometric model. On the left: training pipeline, including feature learning, creating a universal background model (UBM) and its MAP adaptation to client devices. On the right: threshold estimation, score normalization and testing pipeline (see Section 6.3 for description).

This set of equations is provided using the notation of the accelerometer stream, but for the gyroscope the calculations are identical.

Finally, the normalized coordinates, angles and magnitudes are combined in a 14-dimensional vector $\mathbf{x}^{(t)}$ with t indexing the frames (readings):

$$\mathbf{x}^{(t)} = \{\|\mathbf{a}\|, a_{x,n}, a_{y,n}, a_{z,n}, \alpha_x, \alpha_y, \alpha_z, \|\boldsymbol{\omega}\|, \omega_{x,n}, \omega_{y,n}, \omega_{z,n}, \varphi_x, \varphi_y, \varphi_z\} \in \mathbb{R}^{14}. \tag{145}$$

To avoid introducing additional systematic errors and signal corruption, we do not perform any additional denoising of the raw signal.

6.3 BIOMETRIC MODEL

Many methodological choices we make in this work are motivated by specific constraints enforced by the application. For example, relying on cloud computing to authenticate a mobile user is unfeasible due to privacy and latency. Although this technology is well established for many mobile services, our application is essentially different from others (such as, for example, voice search), as it involves constant *background* collection of particularly sensitive user data. Streaming this information to the cloud would create an impermissible threat from a privacy perspective for users and from a legal perspective for service providers.

For these reasons, the biometric user authentication must be performed *on the device* and is constrained by available storage, memory and processing power. Furthermore, adapting to a new user should be quick, resulting in a limited amount of training data for the *positive* class. This data may not be completely representative of

typical usage. Consequently, a purely discriminative setting, which would involve learning a separate model per user, or even fine-tuning a model for each new user would hardly be feasible.

Instead, in this work, we adapt a generative model, namely a Gaussian Mixture Model (GMM), in order to estimate a general data distribution in the dynamic motion feature space and create a *universal background model* (UBM).

The UBM is learned offline, i.e. prior to deployment on the phones, using a large amount of pre-collected training data. For each new user, we use a very small amount of enrollment samples to perform online (i.e. on-device) adaptation of the UBM to create a *client model*. The two models are then used for real time inference of trust scores allowing continuous authentication.

In the rest of this section, we will review the process of UBM learning, adaptation and scoring in more detail.

6.3.1 *Universal background model and client models*

As we have mentioned, learning a separate GMM for each user is suboptimal with respect to the objective to shorten the training phase and minimize computation. Furthermore, background data collection typically results in a highly unbalanced dataset in terms of variety of represented gesture patterns. Instead, we perform a background collection of data from hundreds of users, learn a unsupervised universal background model (UBM), which is then used as a prior for online adaptation of the model to a given user.

Let $\mathbf{y}=f(\{\mathbf{x}^{(t)}\}) \in \mathbb{R}^N$ be a vector of features extracted from a raw sequence of prehensile movements (which is done by one of the deep neural networks that will be described later, in Section 6.4).

To create the UBM, probability densities are defined over these feature vectors as a weighted sum of M multi-dimensional Gaussian distributions parameterized by a set $\Theta=\{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \pi_i\}$, where $\boldsymbol{\mu}_i$ is a mean vector, $\boldsymbol{\Sigma}_i$ a covariance matrix and π_i a mixture coefficient:

$$p(\mathbf{y}|\Theta) = \sum_{i=1}^{M} \pi_i \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{146}$$

$$\text{where } \mathcal{N}_i(\mathbf{y}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}_i|}} e^{-\frac{(\mathbf{y}-\boldsymbol{\mu}_i)' \boldsymbol{\Sigma}_i^{-1} (\mathbf{y}-\boldsymbol{\mu}_i)}{2}}. \tag{147}$$

This UBM $p(\mathbf{y}|\Theta_{\text{UBM}})$ is learned by maximising the likelihood of feature vectors extracted from the large training set using the expectation-maximisation (EM) algorithm.

Each client model $p(\mathbf{y}|\Theta_{\text{client}})$ is adapted from the UBM. Both models share the same weights and covariance matrices to avoid overfitting from a limited amount of enrollment data. Along the lines of [248], maximum a posteriori (MAP) adaptation of mean vectors for a given user is performed. This has an immediate advantage over creating an independent GMM for each user, ensuring proper alignment between the well-trained background model and the client model by updating only a subset

of parameters that are specific to the given user. In particular, given a set of Q enrollment samples $\{\mathbf{y}_q\}$ from the new device, we create a client-specific update to the mean of each mixture component i as follows:

$$E_i(\{\mathbf{y}_q\}) = \frac{1}{n_i} \sum_{q=1}^{Q} Pr(i|\mathbf{y}_q)\mathbf{y}_q, \tag{148}$$

$$\text{where } n_i = \sum_{q=1}^{Q} Pr(i|\mathbf{y}_q), \qquad Pr(i|\mathbf{y}_q) = \frac{\pi_i p_i(\mathbf{y}_q)}{\sum_{j=1}^{M} \pi_j p_j(\mathbf{y}_q)}. \tag{149}$$

Finally, the means of all Gaussian components are updated according to the following rule:

$$\hat{\mu}_i = \alpha_i E_i(\{\mathbf{y}_q\}) + (1-\alpha_i)\mu_i, \quad \text{where } \alpha_i = \frac{n_i}{n_i + r}, \tag{150}$$

where r is a relevance factor balancing the background and client models. In our experiments, we set $r = 4$ (estimated empirically).

6.3.2 *Model scoring*

Given a set of samples $Y=\{\mathbf{y}_s\}$ from a given device, user authenticity is estimated by scoring the feature vectors against the UBM and the client model, thresholding the log-likelihood ratio:

$$\Lambda(Y) = \log p(Y|\Theta_{\text{client}}) - \log p(Y|\Theta_{\text{UBM}}). \tag{151}$$

As a final step, *zt-score normalization* [13] is performed to compensate for inter-session and inter-person variations and reduce the overlap between the distribution of scores from authentic users and impostors.

An offline z-step (zero normalization) compensates for inter-model variation by normalizing the scores produced by each client model to have zero mean and unit variance in order to use a single global threshold:

$$\Lambda_z(Y|\Theta_{\text{client}}) = \frac{\Lambda(Y) - \mu(Z|\Theta_{\text{client}})}{\sigma(Z|\Theta_{\text{client}})}, \tag{152}$$

where Y is a test session and Z is a set of impostor sessions. Parameters are defined for a given user once model enrollment is completed. Then, the $\mathcal{T}$-norm (test normalization) compensates for inter-session differences by scoring a session against a set of background $\mathcal{T}$-models.

$$\Lambda_{zt}(Y) = \frac{\Lambda_z(Y|\Theta_{\text{client}}) - \mu_z(Y|\Theta_{\mathcal{T}})}{\sigma_z(Y|\Theta_{\mathcal{T}})}. \tag{153}$$

The $\mathcal{T}$-models are typically obtained through MAP-adaptation from the universal background model in the same way as all client models, but using different subsets of the training corpus. The Z-sequences are taken from a part of the training data which is not used by the $\mathcal{T}$-models.

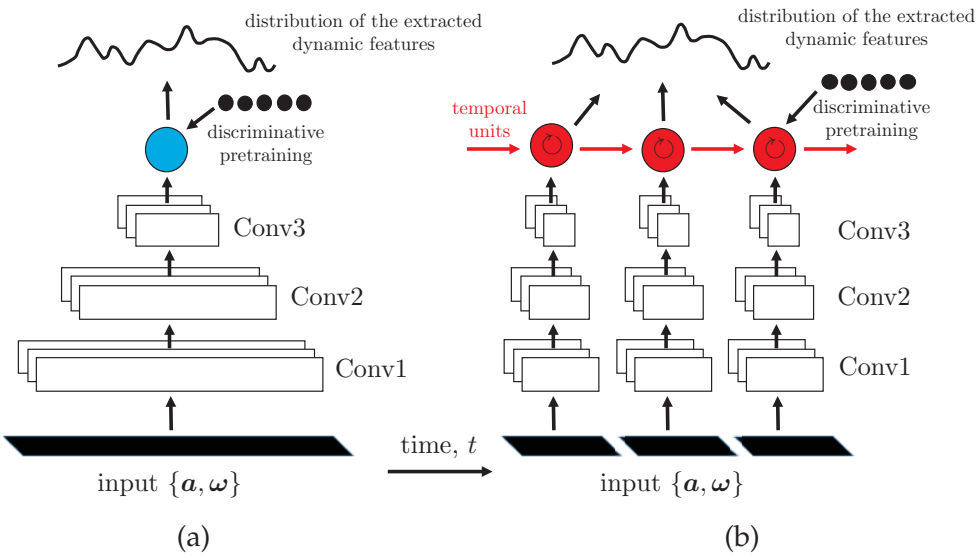


Figure 73 – Learning data representations: (a) static convnet directly operating on sequences, aggregating temporal statistics by temporal pooling; (b) explicitly modeling temporal transitions by with recurrent connections.

6.4 LEARNING EFFECTIVE AND EFFICIENT REPRESENTATIONS

Finally, we proceed with a description of the feature extraction step, the main contribution of this work. Learning effective and efficient data representations is key to our entire framework, since its ability to perform in the real-world is defined by such criteria as latency, representational power of extracted features and inference speed of the feature extractor. The first two conditions are known to contradict each other as performance of a standalone feature typically grows with integration time.

Two paradigms, which strike a balance between *representational power* and *speed*, have dominated the feature learning landscape in recent years. We have presented them in Chapter 3, but we recall them here briefly for convenience. These are multi-scale temporal aggregation via 1-dimensional convolutional networks (shown in Figure 73a), and explicit modeling of temporal dependencies via recurrent neural networks (shown in Figure 73b).

The former model, popular in speech recognition [119], involves convolutional learning of integrated temporal statistics from short and long sequences of data (referred to as *short-term* (ST) and *long-term* (LT) convnets in the experimental Section 6.6). Short-term architectures produce outputs at a relatively high rate (1 Hz in our implementation) but fail to model context. Long-term networks can learn meaningful representations at different scales, but suffer from a high degree of temporal inertia and do not generalize to sequences of arbitrary length.

Recurrent models, which explicitly model temporal evolutions, can generate *low-latency* feature vectors built in the context of previously observed user behavior. The dynamic nature of their representations allow for modeling richer temporal structure and better discrimination among users acting under different conditions.

There have been a sufficiently large number of neural architectures proposed for modeling temporal dependencies in different contexts: the baseline methods, which

are first benchmarked in this work, were earlier described in Section 3.2.1, where we gave a review of existing state-of-the-art deep temporal models. In our experiments, we evaluate the perfomace on the vanilla RNN, the LSTM and the Clockwork architectures (see Section 6.6). Furthermore, in the rest of this section, we introduce a new shift-invariant model based on modified Clockwork RNNs, initially proposed by Koutnik et al.[161].

Given the low correlation of individual frames with user identity, we found it strongly beneficial to make the set of input layers convolutional regardless of model type, thereby forcing earlier fusion of temporal information.

All temporal feature extractors are first pretrained discriminatively for a multi-device classification task. After the following removal of the output layer, the activations of the penultimate layer are provided as input to the generative model, which is described in Section 6.2.

6.4.1 *Dense convolutional clockwork RNNs*

Among the existing temporal models we considered, the Clockwork mechanisms [161] appear to be the most attractive, due to low computational burden associated with them, in combination with their high modeling capacity. This architecture has been previously reviewed in Chapter 3, but let us briefly remind the reader its main working principles.

The Clockwork mechanism is essentially a RNN with the hidden recurrent layer partitioned into several groups of *fast* and *slow* units having different update rates (see the top image in Figure 74). In this work, we assume that the values of the update periods increase exponentially as n^k (as in the original paper), where n is a base and k is the number of the band.

In this context, the fast units are connected to all other bands (shown in red in Figure 74) and integrate both high and low frequency information. The slower units (shown in green and blue in Figure 74) model only low frequency signals.

The update rule of the CWRNN is formulated as follows (reproduced from equation (55) and corresponds to the k-th band of output $\mathbf{h}$ at iteration t):

$$\mathbf{h}_k^{(t)} = \begin{cases} \psi\left(\mathbf{W}(k)\mathbf{x}^{(t)} + \mathbf{U}(k)\mathbf{h}_k^{(t-1)}\right) & \text{if } (t \bmod n^k)=0, \\ \mathbf{h}_k^{(t-1)} & \text{otherwise.} \end{cases}$$

where $\mathbf{U}(k)$ and $\mathbf{W}(k)$ are rows k from matrices $\mathbf{U}$ and $\mathbf{W}$, matrix of recurrent weights $\mathbf{U}$ has an upper triangular structure and in accordance with the connectivity between frequency bands.

One of important shortcomings of the CWRNN model is that, due to inactivity of *slow* units for long periods of time, the efficiency of their training is scaled down exponentially from high to low frequencies. As a result, in practice the low-frequency bands barely contribute to the overall network performance during test time. In addition, in our setting, where the goal is to learn dynamic data representations serving as input to a probabilistic framework, this architecture has one more weakness, which stems from the fact that different bands are active at any given time

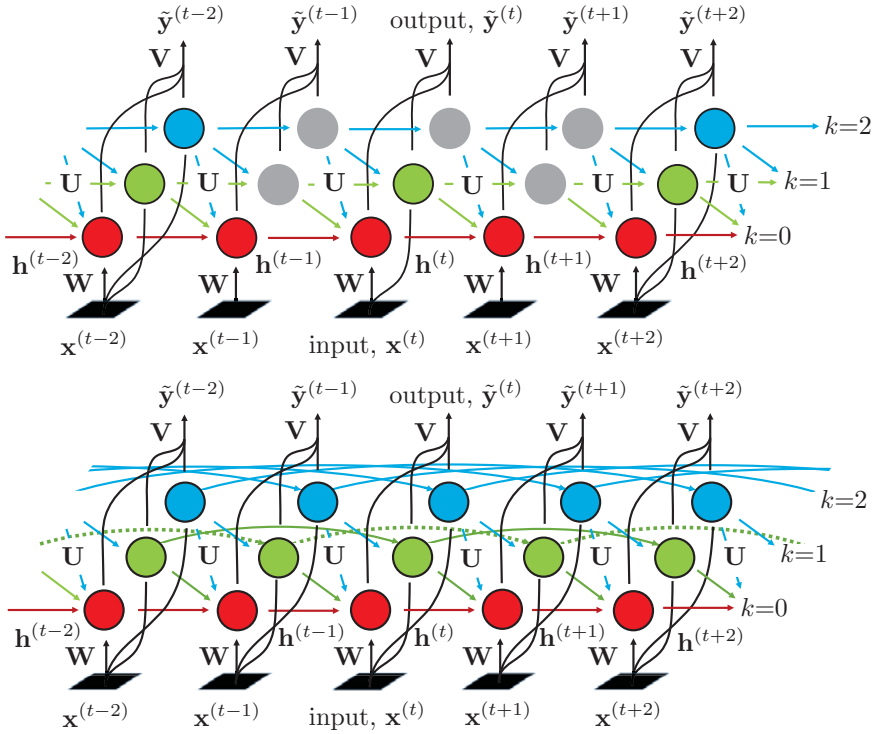


Figure 74 – Comparison of the original Clockwork RNN, proposed by Koutnik et al. [161], and its modification called Dense Clockwork RNN, proposed in this work.

step. As a result, the network will respond *differently* to the same input stimuli applied at different moments in time. This *shift-variance* convolutes the feature space by introducing a shift-associated dimension.

In this work, we propose a solution to both issues, namely *twined* or *dense clockwork mechanisms* (dubbed DCWRNN), which are shown in Figure 74 (in the bottom) in comparison with the original Clockwork RNN (on top).

In such a network, during *inference* at each scale k there exist n^k parallel threads shifted with respect to each other, such that at each time a unit belonging to one of the threads fires, updating its own state and providing input to the higher frequency units. All weights between the threads belonging to the same band are shared, keeping the overall number of parameters in the network the same as in the original clockwork architecture. Without loss of generality, and to keep the notation uncluttered of unnecessary indices, in the following we will describe a network with a single hidden unit h_k per band k. The generalization to multiple units per band is straightforward, and the experiments were of course performed with the more general case.

The feedforward pass for the whole dense clockwork layer (i.e. all bands) can be stated as follows:

$$\mathbf{h}^{(t)} = \psi\left(\mathbf{W}\mathbf{x}^{(t)} + \Delta(\mathbf{U}\mathbf{H})\right), \tag{154}$$

where $H = [\, \mathbf{h}^{(t-1)} \ldots \mathbf{h}^{(t-n^k)} \ \ldots \mathbf{h}^{(t-n^K)} \,]$ is a matrix concatenating the history of hidden units and we define $\Delta(\cdot)$ as an operator on matrices returning its diagonal elements in a column vector. To simplify the presentation, we have not made explicit

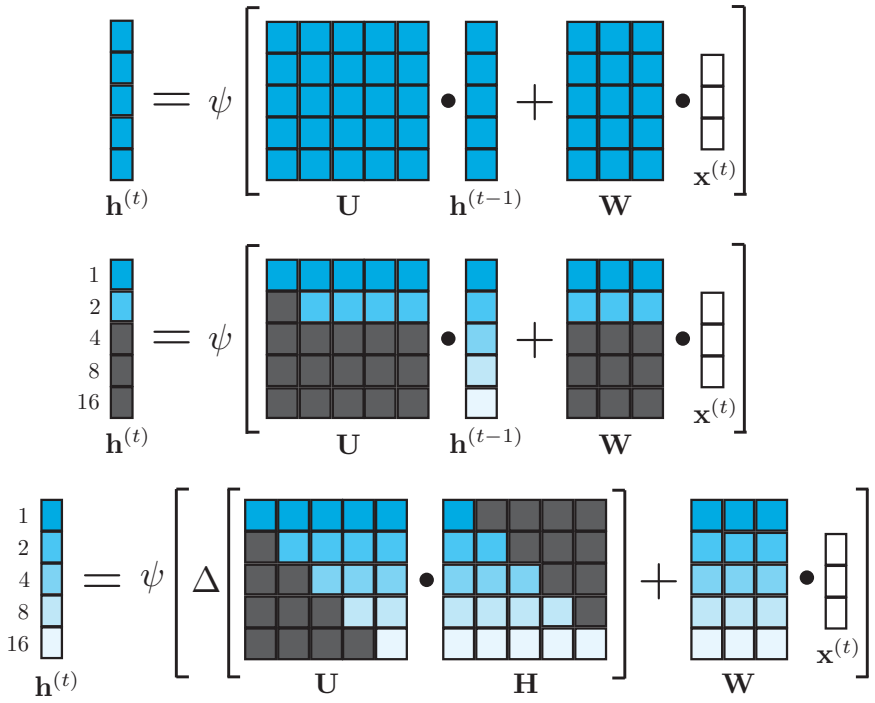


Figure 75 – Updates made by the vanilla RNN (top), the Clockwork RNN [161] (middle) and our proposed Dense CWRNN (bottom). To illustrate the concept, we show in blue those units and weights that are updated or read at the example time step t=6.

the fact that the first layers are convolutional, however, it can be absorbed into the input-to-hidden matrix $\mathbf{W}$.

The intuition for this equation is given in Figure 75, where we visually compare the update rules of the original CWRNN and the proposed DCWRNN using an example of a network with 5 hidden units each associated with one of K=5 base n=2 bands.

To be consistent, in this figure we employ the same matrix form as in the original CWRNN paper [161] and show components, which are inactive at time t, in dark gray. To indicate the general structure of recurrence, we also provide such a representation for the valilla RNN.

As its was mentioned in Section 3.2.1, in the original CWRNN, at time instant t=6, for instance, only units h_1 and h_2 are updated, i.e. the first two lines in the corresponding network from Figure 75. In the dense network (the bottom row), all hidden units h_k are updated at each moment in time.

In addition, what was a vector of previous hidden states $\mathbf{h}^{(t-1)}$ in the original CWRNN, is now replaced with a lower triangular *history matrix* $\mathbf{H}$ of size K×K, which is obtained by concatenating several columns from the history of activations $\mathbf{h}$. Here, K is the number of bands. Time instances are not sampled consecutively, but strided in an exponential range, i.e. $n, n^2, \dots n^K$. Finally, the diagonal elements of the dot product of two triangular matrices form the recurrent contribution to the vector $\mathbf{h}^{(t)}$. The feedforward contribution is calculated in the same way as in a standard RNN.

In practice, implementing the lower-triangular matrix, which containes the history of previous hidden activations in the DCWRNN, requires usage of an additional memory buffer, whose size can be given as

$$m = \sum_{k=1}^{K} |\mathbf{h}_k|(n^{k-1}-1), \tag{155}$$

whereas here we have stated the general case of $|\mathbf{h}_k| \geqslant 1$ hidden units belonging to a recurrent band k.

During training, updating all bands at a constant rate is important for preventing simultaneous overfitting of high-frequency and underfitting of low-frequency bands. In practice it leads to a speedup of the training process and improved performance. Finally, due to the constant update rate of all bands in the dense network, the learned representations are *invariant to local shifts* in the input signal, which is crucial in unconstrained settings when the input is unsegmented (the concept of shift-invariance is demonstrated in Section 6.6).

6.5 DATA COLLECTION

The large scale dataset, which we introduce in this work, is a part of a more general multi-modal data collection effort performed by Google ATAP, known as Project Abacus. We have already provided a brief description of this project in the introduction part of this thesis (Section 1).

The reader may remember, that the ultimate purpose of this study was to explore biometric mobile user authentication in a *multi-modal* framework, therefore the collected data also includes images, voice, data recorded from the touch screen. The motion data, which interests us the most, was acquired from three sensors: accelerometer, gyroscope and magnetometer.

This study included approximately 1,500 volunteers using the research phones as their main devices, and we made every effort to ensure that the data collected representative of their regular and *natural* usage.

Motion data was recorded from the moment after the phone was unlocked until the end of a session (i.e., until it was locked again). For this study, we set the sampling rate for the accelerometer and gyroscope sensors to 200 Hz and for the magnetometer to 5 Hz. However, to prevent drain on the battery, the accelerometer and gyro data were not recorded when the device was at rest. This was achieved by defining two separate thresholds for signal magnitude in each channel. Finally, accelerometer and gyroscope streams were synchronized on hardware timestamps.

Even though the sampling rate of the accelerometer and the gyroscope was set to 200 Hz for the study, we noticed that intervals between readings coming from different devices varied slightly. To eliminate these differences and decrease power consumption, for our research we resampled all data to 50 Hz.

For the following experiments, data from 587 devices were used for discriminative feature extraction and training of the universal background models, 150 devices formed the validation set for tuning hyperparameters, and another 150 devices represented *clients* (users) for testing.

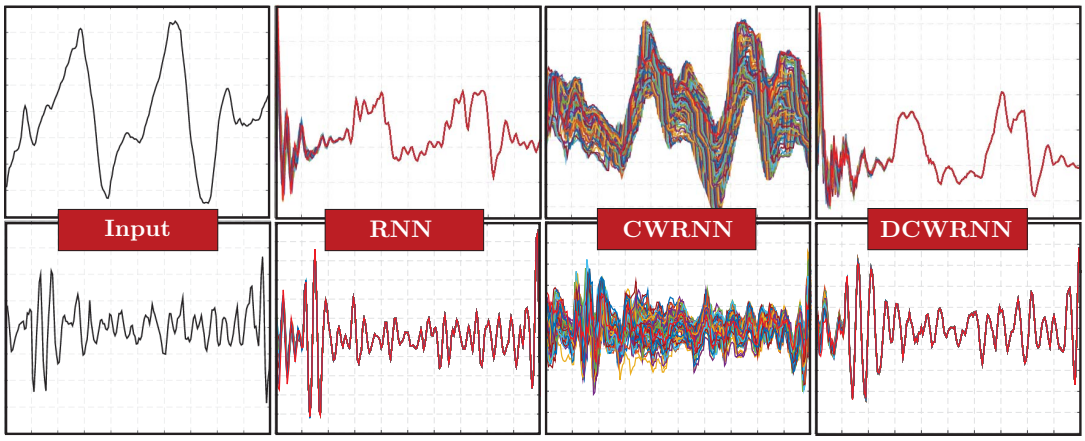


Figure 76 – On spatial invariance. From left to right: original sequence and traces of RNN, CWRNN and DCWRNN units. The first row: reading while walking, the second row: typing while sitting.

6.6 EXPERIMENTAL RESULTS

In this section, before proceeding to the large-scale evaluation of the proposed framework, we use an existing public (but relatively small) inertial dataset to demonstrate the ability of the proposed DCWRNN to learn *shift-invariant* representations. After that, we describe our study involving the large-scale Google Abacus dataset collected *in the wild*.

6.6.1 *Visualization: HMOG dataset*

To explore the nature of inertial sensor signals, we performed a preliminary analysis on the public HMOG dataset [325] containing similar data, but collected in constrained settings as a part of a lab study. This data collection was performed with the help of 100 volunteers, each completing 24 sessions of predefined tasks, such as reading, typing and navigation, while sitting or walking.

Unfortunately, direct application of the whole proposed pipeline to this corpus is not so relevant due to 1) absence of task-to-task transitions in a single session and 2) insufficient data to form separate subsets for feature learning, the background model, client-specific subsets for enrollment, and still reserve a separate subset of *impostors* for testing that haven't been seen during training. In addition, the state-of-the-art authentication method, introduced together with the HMOG dataset, is based mostly on *touch* data captured from the touch screen, and exploits acceleration and rotations as sources of additional features specifically extracted around touch events. To the best of our knowledge, *there are no existing methods performing authentication on a smartphone solely from inertial data*.

A detailed visual analysis of accelerometer and gyroscope streams has proven that the inertial data can be seen as a combination of periodic and quasi-periodic signals (from walking, typing, natural body rhythms and noise), as well non-periodic movements. This observation additionally motivates clockwork-like architectures allowing for explicit modelling of periodic components.

In this subsection, we describe the use of HMOG data to explore the shift-invariance of temporal models that do not have explicit reset gates (i.e. RNN, CWRNN and DCWRNN). For our experiment, we randomly selected 200 sequences of normalized accelerometer magnitudes and applied three different networks each having 8 hidden units and a single output neuron. All weights of all networks were initialized randomly from a normal distribution with a fixed seed. For both clockwork architectures we used a base 2 exponential setting rule and 8 bands.

Finally, for each network we perfomed 128 runs (i.e. 2^{K-1}) on a shifted input: for each run x the beginning of the sequence was padded with $x-1$ zeros. The resulting hidden activations were then shifted back to the initial position and superimposed.

Figure 76 visualizes the hidden unit traces for two sample sequences from the HMOG dataset, corresponding to two different activities: reading while walking and writing while sitting. This figure shows that the RNN and the dense version of the clockwork network can be considered shift-invariant (all curves overlap almost everywhere, except for minor perturbance at the beginning of the sequence and around narrow peaks), while output of the CWRNN is highly shift-dependent.

For this reason, in spite of their atractiveness in the context of multi-scale periodic and non-periodic signals, the usage of the CWRNN for the purpose of feature learning from unsegmented data may be suboptimal due to high shift-associated distortion of learned distributions, which is not the case for DCWRNNs.

6.6.2 *Large-scale study: Google Abacus dataset*

We now evaluate our proposed authentication framework on the real-world Google Abacus dataset, which was described in Section 6.5, and compare it to other temporal models, as well as purely convolutional learning.

For this purpose, we perform *two rounds of evaluation*. The first one compares performance of feature representations, learned by different models, in a discriminative task of *1-to-N* device classification. This is the same task considered in learning features. The second one corresponds to the final goal of this work and evaluates performance of the previously extracted features, but in the authentication setting and as a part of the generative biometric model, described in Section 6.3.

Tables 24 and 25 provide the complete set of architectural hyper-parameters, both for the baseline and for the proposed solutions, that were chosen based on a held-out validation set. For convolutional nets, we distinguish between convolutional layers (Conv, which include pooling) and fully-connected layers (FCL). For recurrent models, we report the total number of units (in the case of CWRNN and DCWRNN, over all bands). To make a fair comparison, we set the number of parameters to be approximately the same for all of the temporal models.

The short-term (ST) Convnet is trained on sequences of 50 samples (corresponding to a 1 s data stream), long-term (LT) Convnets take as input 500 samples (i.e. 10 s). All temporal architectures are trained on sequences of 20 blocks of 50 samples with 50% of inter-block overlap to ensure smooth transitions between blocks (therefore, also a 10 s duration). For the dense and sparse clockwork architectures we set the number of bands to 3 with a base of 2. All layers in all architectures use tanh activations.

Network architecture: feedforward learning		
Layer	Filter size / # of units	Pooling
Input	500×14 (50 × 14)	-
Conv1	25×9×1	8×1 (2×1)
Conv2	25×9×1	4×1 (1×1)
Conv3	25×9×1	1×1
FCL1	2000	-
FCL2	1000	-
Output	587	-

Table 24 – Hyper-parameters of baseline feedforward convolutional architectures: values in parentheses are for short-term (ST) Convnets when different from long-term (LT).

Network architecture: sequential learning		
Layer	Filter size / # of units	Pooling
Input	10×50×14	-
Conv1	25×7×1	2×1
Conv2	25×7×1	2×1
Conv3	25×7×1	1×1
Recurrent	RNN 894, LSTM 394,	-
	CWRNN and DCWRNN 1000	-
Output	587	-

Table 25 – Hyper-parameters: values in parentheses are for short-term (ST) Convnets when different from long-term (LT).

The dimensionality of the feature space produced by each of the networks is PCA-reduced to 100. GMMs with 256 mixture components are trained for 100 iterations after initialization with k-means (100 iterations). MAP adaptation for each device is performed in 5 iterations with a relevance factor of 4.

For zt-score normalization, we exploit data from the same training set and create 200 $\mathcal{T}$-models and 200 z-sequences from non-overlapping subsets. Each $\mathcal{T}$-model is trained based on UBM and MAP adaptation. All hyper-parameters were optimized on the validation set.

The networks are trained using stochastic gradient descent, dropout in fully connected layers, and negative log likelihood loss. In the temporal architectures, we add a mean pooling layer before applying the softmax. Each element of the input is normalized to zero mean and unit variance.

All deep nets were implemented with Theano [31] and trained on 8 Nvidia Tesla K80 GPUs. UBM-GMMs were trained with the Bob toolbox [7] and did not employ GPUs.

Feature learning: evaluation		
Model	Accuracy, %	# parameters
ST Convnet	37.13	6 102 137
LT Convnet	56.46	6 102 137
Conv-RNN	64.57	1 960 295
Conv-CWRNN	68.83	1 964 254
Conv-LSTM	68.92	1 965 403
Conv-DCWRNN	**69.41**	**1 964 254**

Table 26 – Performance and model complexity of the feature extractors. These results assume one user per device and accuracy is defined based on whether or not the user is in the top 5% of classes according to the output distribution.

6.6.2.1 *Feature learning: evaluation*

Let us now proceed with the first step of the experiments evaluating the performance of data representations learned with different models.

We first performed a quantitative evaluation of the effectiveness of feature extractors alone as a multi-class classification problem, where one class corresponds to one of 587 devices from the training set. This way, one class is meant to correspond to one *user*, which is equal to a *device* in the training data (assuming devices do not change hands). To justify this assumption, we manually annotated periods of non-authentic usage based on input from the smartphone camera and excluded those sessions from the test and training sets. Experiments showed that the percentage of such sessions is insignificant and their presence in the training data has almost no effect on the classification performance.

Note that for this test, the generative model was *not* considered and the feature extractor was simply evaluated in terms of classification accuracy. To define accuracy, we must consider that human kinematics sensed by a mobile device can be considered as a weak biometric and used to perform a soft clustering of users in behavioral groups. To evaluate the quality of each feature extractor in the classification scenario, for each session we obtained aggregated probabilities over target classes and selected the 5% of classes with highest probability. After that, the user behavior was considered to be interpreted correctly if the ground truth label was among them.

The classification accuracy obtained with each type of deep network with its corresponding number of parameters is reported in Table 26. These results show that the vanilla convolutional architectures generally perform poorly, while among the temporal models the proposed dense clockwork mechanism Conv-DCWRNN appeared to be the most effective, while the original clockwork network (Conv-CWRNN) was slightly outperformed by the LSTM.

We would like to note here, that the not so significant advantage of the DCWRNN over the LSTM should not be immediately seen as a weakness. The DCW mechanism should be considered as a way to efficiently distribute the recurrent weights

User authentication: evaluation		
Model	EER, %	HTER, %
Raw features	36.21	42.17
ST Convnet	32.44	34.89
LT Convnet	28.15	29.01
Conv-RNN	22.32	22.49
Conv-CWRNN	21.52	21.92
Conv-LSTM	21.13	21.41
Conv-DCWRNN	20.01	20.52
Conv-DCWRNN, zt-norm	**18.17**	**19.29**
Conv-DCWRNN (per device)	15.84	16.13
Conv-DCWRNN (per session)	8.82	9.37

Table 27 – Performance of the GMM-based biometric model using different types of deep neural architectures. EER is given on the validation set, while HTER is estimated on the final test set using the same threshold.

over temporal scales, which can be straightforwardly incorporated into any gated architecture, when needed by an application. A Conv-DCWLSTM may achieve superior results, but this is beyond the scope of our work.

6.6.2.2 *User authentication: evaluation*

When moving to the *binary authentication problem*, an optimal balance of false rejection and false acceptance rates, which is not captured by classification accuracy, becomes particularly important.

In this setting, we use a validation subset to optimize the generative model for the minimal equal error rate (EER). The obtained threshold value θ_{EER} is then used to evaluate performance on the test set using the half total error rate (HTER) as a criterion:

$$\text{HTER} = \frac{\text{FAR}(\theta_{EER}) + \text{FRR}(\theta_{EER})}{2}, \tag{156}$$

where FAR and FRR are false acceptance and false rejection rates, respectively.

For the validation set, we also provide an average of per-device and per-session EERs (obtained by optimizing the threshold for each device/session separately) to indicate the upper bound of performance in the case of perfect score normalization (see italicized rows in Table 27).

An EER of 20% means that 80% of the time the correct user is using the device, s/he is authenticated, only by the way s/he moves and holds the phone, not necessarily interacting with it. It also means that 80% of the time the system identifies the user, it was the correct one. These results align well with the estimated quality of feature extraction in each case and show that the context-aware features can be efficiently incorporated in a generative setting.

ChaLearn 2014: sequential learning			
Model	Jaccard Index	Accuracy	N parameters
Single network† (Chapter 4)	0.827	91.62	1 384 621
Ensemble†† (Chapter 4)	0.831	91.96	4 153 863
Conv-RNN	0.826	91.79	3 974 581
Small Conv-LSTM	0.815	91.50	3 976 863
Large Conv-LSTM	0.825	91.89	4 900 621
Conv-CWRNN	0.834	92.38	3 972 496
Conv-DCWRNN	**0.841**	**93.02**	3 972 496

Table 28 – Performance of the proposed DCWRNN architecture on the *Chalearn 2014 Looking at People dataset* (mocap modality). Network parameters: input 183×9, conv. layer 25×3×1, 2 fully connected layers with 700 units, the recurrent layer (RNN-280, CWRNN-300, DCWRNN-300, Small LSTM-88, Large LSTM-300), 21 output class. † taken from Table 6, †† taken from Table 7.

To compare GMM performance with a traditional approach of retraining, or fine-tuning a separate deep model for each device (even if not applicable in a mobile setting), we randomly drew 10 devices from the validation set and replaced the output layer of the pretrained LSTM feature extractor with a binary logistic regression. The average performance on this small subset was 2% inferior with respect to the GMM, due to overfitting of the enrollment data and poor generalization to unobserved activities. This is efficiently handled by mean-only MAP adaptation of a general distribution in the probabilistic setting.

Another natural question is whether the proposed model learns something specific to the user *style* of performing tasks rather than a typical sequence of tasks itself. To explore this, we performed additional tests by extracting parts of each session where all users interacted with the same application (a popular mail client, a messenger and a social network application). We observed that the results were almost identical to the ones previously obtained on the whole dataset, indicating low correlation with a particular activity.

6.7 MODEL ADAPTATION FOR A VISUAL CONTEXT

Finally, we would like to stress that the proposed DCWRNN framework can also be applied to other sequence modeling tasks, including the visual context. The described model is not specific to the data type and there is no particular reason why it cannot be applied to the general human kinematic problem (such as, for example, action or gesture recognition from motion capture).

To support this claim, we have conducted additional tests of the proposed method within a task of *visual gesture recognition*, a task presented in Chapter 4.

Namely, we provide results on the *mocap modality* from the same *ChaLearn 2014 Looking at People* gesture dataset [87], which was previously used in our gesture

recognition experiments. As we have already seen, this dataset contains about 14000 instances of Italian conversational gestures with the aim to detect, recognize and localize in continuous noisy recordings. Generally, this corpus comprises multimodal data captured with the Kinect and therefore includes RGB video, depth stream and mocap data. However, only the last channel is used in this round of experiments.

As before, model evaluation is performed using the Jaccard index, penalizing for errors in classification as well as imprecise localization (see Section 4.7 for a detailed description of the evaluation protocol).

Direct application of the GMM to gesture recognition is suboptimal (as the vocabulary is rather small and defined in advance), therefore, in this task we perform end-to-end discriminative training of each model to evaluate the effectiveness of *feature extraction* with the Dense CWRNN model.

Following our state-of-the art gesture recognition method proposed in Chapter 4 (which was ranked 1st in the ECCV 2014 ChaLearn competition), we use the same skeleton descriptor as input. However, as in the described authentication framework, the input is fed into a convolutional temporal architecture instead of directly concatenating frames in a spatio-temporal volume. The final aggregation and localization step correspond to Chapter 4. Table 28 reports both the Jaccard index and per-sequence classification accuracy and shows that in this application, the proposed DCWRNN also outperforms the alternative solutions.

6.8 CONCLUSION

From a modeling perspective, the work presented in this chapter has demonstrated that temporal architectures are particularly efficient for learning of dynamic features from a large corpus of noisy temporal signals, and that the learned representations can be further incorporated in a generative setting. With respect to the particular application, we have confirmed that natural human kinematics convey necessary information about person identity and therefore can be useful for user authentication on mobile devices. The obtained results look particularly promising, given the fact that the system is completely non-intrusive and non-cooperative, i.e. does not require any effort from the user's side.

Non-standard weak biometrics are particularly interesting for providing the context in, for example, face recognition or speaker verification scenarios. Further augmentation with data extracted from keystroke and touch patterns, user location, connectivity and application statistics (ongoing work) may be a key to creating the first secure non-obtrusive mobile authentication framework.

Furthermore, in the final round of experiments, we have demonstrated that the proposed Dense Clockwork RNN can be successfully applied to other tasks based on analysis of sequential data, such as gesture recognition from visual input.

In its current state, the Google Abacus authentication dataset cannot be released for the research community for security reasons, as it contains sensitive user information obtained from background data collection. However, we believe that our experience can be valuable for communities working on similar problems and, more importantly, as an additional view on the process of spatio-temporal feature learning applied to human kinematics and biometrics.

When you remove the fear of failure, impossible things suddenly become possible.
If you want to know how, ask yourself this question:
"What would you attempt to do if you knew you could not fail?.."

— Regina Dugan

If you want to make a reviewer laugh, tell him about your plans.

— Anonymous

7

SUMMARY AND FUTURE WORK

The presented manuscript summarized our three year long journey in both academic and industrial research, and we were honored to walk the reader step-by-step through our experiences and our findings. Finally, it is time to make some closing remarks, to highlight once again the key contributions of this thesis and to shed some light on the potential directions of future research, some of which are already work in progress, while others are merely intuitions.

For this final review, let us for the last time adopt the same order of presentation, as in the main part of this thesis.

— **Part 1. Multimodal gesture recognition**.

In the initial stage of this work, we created and optimized a *multi-modal system* for action and gesture recognition, with particular emphasis on *fusion aspects* and means of increasing *robustness* of the system to signal degradation in real world conditions. In a general multi-modal setting, we demonstrated how channels of *arbitrary nature* can be incorporated and effectively combined with already functioning modalities, by introducing an audio signal in our vision based framework. In addition to an extensive empirical evaluation, we theoretically derived and analyzed the key properties of the proposed *ModDrop* fusion process.

While working on this project, we have participated in a number of scientific competitions, including the ICMI 2013 ChaLearn Multi-modal gesture recognition challenge, were we placed 6th, and, a year later, the ECCV 2014 ChaLearn Looking at People gesture recognition challenge, where we demonstrated the best performance among all participants. Finally, we were named among the winners of the CVPR 2015 OpenCV State-of-the-art Computer Vision challenge in the category of gesture recognition.

In this multi-modal setting, probably the most puzzling and crucial question, which still remains open, is how to further *automate* the process of optimization of architectural and training hyper-parameters in such complex deep multi-modal networks. Even though we made every effort to keep the formulation of the fusion principles general, the optimization of learning data representations on the *initial stages* of signal-specific processing is still based on manual tuning, which is extremely time consuming and requires a certain level of expertise. The

same applies to determining the particular *order* and localization of *optimal fusing points* for different combinations of signals propagated in a deep architecture.

The first alternative is to treat each aspect of the fusion as a model hyperparameter, and search for the optimal architecture using modern Sequential Model-based Global Optimization approaches (e.g. Bayesian Optimization). This is the simpler approach as it does not require that the objective be differentiable with respect to the control parameters of the architecture.

The second alternative is to propose a type of network where the fusion structure is differentiable such that the model architecture can be optimized by gradient-based methods. This should make the model search much faster, but it requires ingenuity in the design as more restriction is placed on the model structure.

In this context, a number of machine learning research teams have worked on *search* strategies [30] for hyper-parameter optimization in deep learning architectures, as well as exploiting model *compositionality* [113, 163] for determining an appropriate model structure, best explaining statistics of a given dataset. Recently proposed Highway networks [278], allowing for adaptive routing of information through extremely deep networks, may appear to be an important step also in this direction. In any case, we hope and expect that more light will be shed on this aspect in the near future.

Furthermore, training strategies for introducing explicit *modeling of temporal dependencies* in such complex and sometimes cumbersome architectures require further optimization. These temporal aspects were only briefly and preliminary explored in the presented work. From our current observations, augmenting the network with recurrent connections followed by end-to-end training results in slight improvement in its overall performance, but, at the same time, leads to degradation of feedforward connections and, consequently, loss in effectiveness of feature extraction from a single frame in each channel.

Our immediate task, however, is more earthbound and consists in the adaptation of the proposed gesture recognition framework to the *human-robot interaction* setting in the *real world* scenario. As the reader may remember, the primary goal of the robotics project, funding this research, was to develop a gestural interface for a domestic *robot-companion*, manufactured by our industrial partner (Awabot). Furthermore, a similar functionality is planned to be adapted for *telepresence systems*, which are now being actively integrated in museums, conference halls and even schools.

As it was previously mentioned in the introduction, in this project, together with Awabot and our academic partners, we have developed a *natural* gesture vocabulary based on psychological study using a *Wizard of Oz* technique. In this study, the participants were asked to interact with a robot in the most natural and intuitive (for them) way, while robot responses and actions were simulated by a human operator. The whole variety of observed gesticulation was later summarized and quantized into most frequent gesture classes. This defined vocabulary of gesture types was used by us later as a basis for an extensive data collection effort.

The mentioned *wizard-of-oz* social study, performed by our academic partner (Laboratoire d'Informatique de Grenoble), has empirically confirmed an important intuitive assumption, that the intensity of human articulation in their interaction with a robot *decreases with time*, as soon as the robot demonstrates correct understanding of communicated instructions. This phenomenon, sometimes referred to as *socio-affective glue* [12], should be taken into account by one who attempts to make the human-interaction process truly *casual* and *natural* for users. Algorithmically, it implies that the gesture recognition system should have built-in capabilities for *online learning*, in order to be able to perform adaptation of the robot to its owner's style and especially to its evolution with time towards more subtle articulation.

To summarize, in the context of multi-modal gesture recognition, these mentioned engineering challenges in human-robot interaction will be guiding our efforts in the remaining months of this project, while the more abstract theoretical explorations are likely to be among our research priorities in coming years.

— **Part 2. Hand pose estimation**.

Advancing from gesture *classification* to the idea of sensing gesture *parameters*, we have proposed a method for hand pose estimation from a single depth map, formulated as *localization of hand joints* in 3D space.

The key to our approach is exploiting a *rich intermediate representation* of a hand pose in the form of *hand segmentation into parts*. For this work, we have created a *large synthetic dataset* of hand poses accompanied with ground truth segmentations. In Chapter 5 we demonstrate that a model, which is initially trained to perform segmentation on synthetic data, can be effectively adapted to *real input* using a *weakly supervised* or *unsupervised transductive adaptation* pipeline. The exact strategies for unsupervised and weakly supervised scenarios are described and evaluated in the corresponding chapter. Finally, we show that such a structured intermediate representation, if used as a *complementary* input for a regression model (along with the conventionally used depth maps) can significantly improve accuracy of estimation of hand joint positions.

Although the results obtained in this thesis already look promising, there is a significant number of further improvements which are likely to be introduced in our follow up work on hand pose estimation.

1. First of all, we are planning on increasing the effectiveness and efficiency of segmentation and regression learners by introducing a preliminary step of *view point clustering*, as it was previously done in a number of traditional (non deep learning) frameworks [152, 153, 288] (see Section 2.3.4.1 for description and Figure 15 for a graphical illustration). These earlier works have shown that training a separate model for each view point cluster followed by adaptive switching between the models in test time, makes it possible to significantly reduce the required complexity of the model and therefore improve the overall performance and speed up computations.

   In our particular case, as the reader may remember, adding the intermediate representation to the input data is based on the premise that the label space

holds important topological and geometrical information, and these informations are certainly already used during the restoration process. However, their main usage will be as additional input for regression. In this context, whereas the topological information is invariant to viewpoints, the geometrical information is not. Getting information from a single viewpoint, a network can easier translate this information into joint coordinates.

2. In addition, we are investigating potentially more effective strategies for the step of weakly supervised adaptation of the segmentation learner (from the synthetic input to real data). One of the possible approaches would be to introduce an additional rendering pipeline, which would generate synthetic segmented images given ground truth positions of hand joints in the real dataset. In this case, the obtained segmentation maps could be used as an *approximate* ground truth for adaptation of the segmentation model.
3. Furthermore, we believe that we have not yet exhausted the potential of the proposed segmentation-based hand representation, meaning its role in the hand pose estimation framework. Therefore, we are currently exploring alternative and more effective ways to pass the segmentation information (along with the depth input) to the regression network.
4. Finally, we are planning on incorporating an additional *refinement step* on the final stage of the pipeline in order to penalize and filter out anatomically implausible hand pose configurations (which is commonly done in many state-of-the-art frameworks to boost the model performance). Such refinement can be done by enforcing pose priors, which are statistically estimated from the training data, or by introducing an additional generative pathway, based on rendering of a 3D hand model, or on an alternative solution.

So far, we have been mentioning different ways to improve the hand pose estimation pipeline per se, but, in addition, there exist a more high level objective, namely combining the *gesture recognition* interface and the *hand pose estimation* pipeline in a *single framework* and, as before, integrating this system in the context of *human-robot interaction*.

Our domestic companion robot (which we mentioned just before) is equipped with a pico-projector, that serves to display digital content by projecting it on some specified surface. In this sense, an additional gestural functionality, such as navigation in menus, scrolling or zooming, would be extremely beneficial. Naturally, this kind of application requires more accurate estimation of hand motion parameters than in the case of simple gesture recognition. The same idea applies to the context of interactive communication of *navigating instructions* to a robot, in which case hand pose estimation would be required by a robot, for example, to evaluate where the user is pointing.

Looking to the more distant future, we are interested in exploring *less traditional ways* of formulating the hand pose estimation problem. For example, in gestural interfaces, the step of exact localization of hand joints in space could be compromised for an alternative pose representation (such as the same segmentation, for example, or any other), or removed completely.

Going in this direction, we are planning on exploring ways to *directly map hand movements into the response which is expected from the gestural interface*. For training, such ground truth data annotations could be obtained through an *imitation* process, i.e. by asking a user to reproduce a movement, which he or she would perform to get the exact specified response from the system.

We have previously discussed potential advantages of such an approach in introduction to Chapter 5, where we mentioned different strategies for building gestural interfaces, which include the hand pose estimation step (see Figure 53). In those terms, this possible direction of future research corresponds to exploring Strategy (4) (or even Strategy (1), in the simplest case).

We believe that implementing this idea into the setting of end-to-end training, by mapping the visual input into the desired response following the analysis of learned representations would allow us to reach better a understanding of the hand pose estimation problem. As a result, we hope to be able to formulate alternative hand pose representations, which could be potentially more effective and efficient in the specific context of target applications.

— **Path 3. Authentication from motion**.

In the final part of this thesis, we have tackled the problem of *authentication* of smart phone users based on their *motion patterns*, which are captured by a number of *inertial sensors* built in the smart phone (such as an accelerometer and gyroscope). In this context, we have demonstrated that these inertial signals indeed convey information about a user identity and therefore can be considered as useful biometric cues. Accordingly, this channel has a potential for being integrated in multi-modal mobile authentication frameworks.

From the modeling perspective, we have proposed a general improvement for an existing temporal deep learning architecture (namely, the Clockwork RNN [161]) and demonstrated its efficiency not only in the authentication setting, but also for gesture recognition from visual data.

In the authentication framework, the motion data representations, learned by the proposed temporal model, are incorporated in a lightweight generative framework allowing for the creation of a new client model directly on the device without retraining of the feature extraction pipeline (which would be prohibitively computationally expensive from a computational standpoint in the mobile setting).

However, this is only the first step of a big project, and there is a significant amount of work which remains to be done. First of all, it applies to further analysis of representations learned from this temporal data for better understanding of what aspects of human motion are particularly characteristic from a biometric point of view. One potential direction for future investigation could be considering motion patterns specifically in the context of particular applications (such as browsing the Internet, writing emails, playing games or simply picking up the phone).

The ultimate goal of this project, however, is incorporating this module in a larger biometric framework and finding effective and efficient strategies for combining signals of a different nature (including images, speech and touch data).

This concludes our discussion of human motion analysis with deep learning methods. In addition to what has been accomplished, a tremendous amount of work still remains to be done, and it is likely to take years before gesture-controlled telepresence systems are widely integrated in high schools, domestic robot companions become a part of our daily lives and multi-modal mobile authentication is included as a standard component in the Android operating system. However, we believe that future efforts and future research will keep bringing new insights to make *the impossible things possible*.

PUBLICATIONS

The ideas and figures presented in this thesis have previously appeared in the following manuscripts (already published or undergoing review process):

1. N. Neverova, C. Wolf, F. Nebout, G. Taylor. Hand Pose Estimation through Weakly-Supervised Learning of a Rich Intermediate Representation. Preprint, arXiv:1511.06728v1, 2015. **[journal submission, under review]** [210]
2. N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, G. Taylor. Learning Human Identity from Motion Patterns. Preprint, arXiv:1511.03908v3, 2015. **[journal submisssion, under review]** [201]
3. D. Fourure, R. Emonet, E. Fromont, D. Muselet, N. Neverova, A. Trémeau, C. Wolf. Multi-task, Multi-domain Learning: application to semantic segmentation and pose regression, 2016. **[conference submission, under review]** [66]
4. N. Neverova, C. Wolf, G. Taylor, F. Nebout. ModDrop: adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. PP:99, 2015. [204]
5. L. Guillaume, V. Auberge, R. Magnani, F. Aman, C. Cottier, Y. Sasa, C. Wolf, F. Nebout, N. Neverova, et al. Gestural HRI in an ecological dynamic experiment: the GEE corpus based approach for the Emox robot. *International Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 2015. [115]
6. N. Neverova, C. Wolf, G. Taylor, F. Nebout. Hand segmentation with structured convolutional learning. *ACCV*, 2014. [203]
7. N. Neverova, C. Wolf, G. Taylor, F. Nebout. Multi-scale deep learning for gesture detection and localization. *ECCV Workshop on Looking at People*, 2014. [204]
8. N. Neverova, C. Wolf, G. Paci, G. Sommavilla, G. Taylor, F. Nebout. multi-scle approach to gesture detection and recognition. *ICCV Workshop on Understanding Human Activities: Context and Interactions*, 2013. [202]

[1] A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(1):44–58, 2006.

[2] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 2011.

[3] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. R. Bradski. CAD-model recognition and 6DOF pose estimation using 3D cues. *ICCV Workshops*, 2011.

[4] L. A. Alexandre, A. C. Campilho, and M. Kamel. On combining classifiers using sum and product rules. *Pattern Recognition Letters*, 22:1283–1289, 2001.

[5] D. Amodei, R. Anubhai, Case C. Casper J. Battenberg, E., et al. Deep speech 2: End-to-end speech recognition in english and mandarin. Preprint, arXiv:1512.02595, 2015.

[6] B. Andres, T. Beier, and J. H. Kappes. OpenGM: A c++ library for discrete graphical models. Preprint, arXiv:1206.0111v1, 2012.

[7] A. Anjos, L. El Shafey, R. Wallace, M. Günther, C. McCool, and S. Marcel. Bob: a free signal processing and machine learning toolbox for researchers. *ACMMM*, 2012.

[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[9] M. Asad and C. Abhayaratne. Kinect depth stream pre-processing for hand gesture recognition. *ICIP*, 2013.

[10] V. Athitsos and S. Sclaroff. Estimating 3D hand pose from a cluttered image. *CVPR*, 2003.

[11] V. Auberge, Y. Sasa, T. Robert, N. Bonnefond, and B Meillon. Emoz: a wizard of oz for emerging the socio-affective glue with a non humanoid companion robot. *WASSS - Workshop on Affective Social Speech Signals*, 2013.

[12] V. Aubergé, Y. Sasa, N. Bonnefond, B. Meillon, T. Robert, et al. The eee corpus: socio-affective "glue" cues in elderly-robot interactions in a smart home with the emoz platform. *5th International Workshop on Emotion, Social Signals, Sentiment and Linked Open Data*, 2014.

[13] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas. Score normalization for text-independent speaker verification systems. *Digital Signal Processing 10 (1), 42-54*, 2000.

[14] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Spatio-Temporal Convolutional Sparse Auto-Encoder for Sequence Classification. *BMVC*, 2012.

[15] F. Bach, G. Lanckriet, and M. Jordan. Multiple Kernel Learning, Conic Duality, and the SMO Algorithm. *ICML*, 2004.

[16] A. D. Bagdanov. Real-time hand status recognition from RGB-D imagery. *ICPR*, 2012.

[17] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[18] P. Baldi and P. Sadowski. The dropout learning algorithm. *Journal of Artificial Intelligence*, 210:78–122, 2014.

[19] L. Ballan, A. Taneja, J. Gall, L. van Gool, and M. Pollefeys. Motion Capture of Hands in Action using Discriminative Salient Points. *ECCV*, 2012.

[20] J. Bao. Dynamic hand gesture recognition based on SURF tracking. *International Conference on Electric Information and Control Engineering (ICEICE)*, 2011.

[21] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. *ECCV*, 2006.

[22] I. Bayer and T. Silbermann. A Multi Modal Approach to Gesture Recognition from Audio and Video Data. *ICMI*, 2013.

[23] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. *NIPS*, 2000.

[24] Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[25] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[26] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *NIPS*, 2007.

[27] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. Preprint, arXiv:1212.0901v2, 2012.

[28] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):1798–1828, 2013.

[29] M. Van Den Bergh. Haarlet-based hand gesture recognition for 3D interaction. *IEEE Workshop on Applications of Computer Vision (WACV 2009)*, 2009.

[30] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *JMLR*, 2012.

[31] J. Bergstra et al. Theano: a CPU and GPU math expression compiler. *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.

[32] S. Berlemont, G. Lefebvre, S. Duffner, and C. Garcia. Siamese Neural Network based Similarity Metric for Inertial Gesture Classification and Rejection. *FG*, 2015.

[33] P. J. Besl and N. MacKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1992.

[34] S. Bhattacharya, P. Nurmi, N. Hammerla, and T. Plotz. Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive and Mobile Computing*, 2014.

[35] C. Bo, L. Zhang, X.-Y. Li, Q. Huang, and Y Wang. Silentsense: Silent user identification via touch and movement behavioral biometrics. *MobiCom*, 2013.

[36] M. Bray, E. Koller-Meier, and L. Van Gool. Smart particle filtering for high-dimensional tracking. *Computer Vision and Image Understanding*, 106(1):116–129, 2007.

[37] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. Classification and regression trees. 1984.

[38] F. Briggs and W. Usrey. Emerging views of corticothalamic function. *Current Opinion in Neurobiology*, 2008.

[39] A. Bulling, U. Blanke, and B. Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 2014.

[40] M. Cai, K. M. Kitani, and Y. Sato. A scalable approach for understanding the visual structures of hand grasps. *ICRA*, 2015.

[41] F. Caillette, A. Galata, and T. Howard. Real-time 3-D human body tracking using learnt models of behaviour. *Computer Vision and Image Understanding*, 109(2):112–125, February 2008.

[42] N. C. Camgoz, A. A. Kindiroglu, and L. Akarun. Gesture Recognition using Template Based Random Forest Classifiers. *ECCV ChaLearn Workshop on Looking at People*, 2014.

[43] O. Çeliktutan, C. B. Akgül, C. Wolf, and B. Sankur. Graph-Based Analysis of Physical Exercise Actions. *1st ACM MM Workshop on Multimedia Indexing and Information Retrieval for Healthcare (MIIRH)*, 2013.

[44] B. Chakraborty, O. Rudovic, and J. Gonzalez. View-invariant human-body detection with extension to human action recognition using component-wise HMM of body parts. *FGR*, 2008.

[45] J. Y. Chang. Nonparametric Gesture Labeling from Multi-modal Data. *ECCV ChaLearn Workshop on Looking at People*, 2014.

[46] B. Chen, J.-A. Ting, B. Marlin, and N. de Freitas. Deep learning of invariant Spatio-Temporal Features from Video. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[47] G. Chen, D. Clarke, M. Giuliani, D. Weikersdorfer, and A. Knoll. Multi-modality Gesture Detection and Recognition With Un-supervision, Randomization and Discrimination. *ECCV ChaLearn Workshop on Looking at People*, 2014.

[48] L. Chen, H. Wei, and J. Ferryman. A survey of human motion analysis using depth imagery. *Pattern Recognition Letters*, 34(15):1995–2006, November 2013.

[49] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *ICLR*, 2015.

[50] M.-Y. Chen and A. Hauptmann. MoSIFT: Recognizing human actions in surveillance videos. Technical report, 2009.

[51] X. Chen and A. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. *CVPR*, 2014.

[52] Z. Cheng, L. Qin, Y. Ye, Q. Huang, and Q. Tian. Human Daily Action Analysis with Multi-View and Color-Depth Data. *ECCV*, 2012.

[53] G. Cheron, I. Laptev, and C. Schmid. P-cnn: Pose-based cnn features for action recognition. *ICCV*, 2015.

[54] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoderdecoder for statistical machine translation. Preprint, arXiv:1406.1078v3, 2014.

[55] M. J. Choi, V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Learning latent tree graphical models. *JMLR*, 12:1771–1812, 2011.

[56] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. Preprint, arXiv:1412.3555v1, 2014.

[57] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Gated feedback recurrent neural networks. Preprint, arXiv:1502.02367v4, 2015.

[58] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[59] S. Cobos, M. Ferre, M. A. Sanchez Uran, J. Ortego, and C. Pena. Efficient human hand kinematics for manipulation tasks. *International Conference on Intelligent Robots and Systems*, 2013.

[60] R. Collobert, K. K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. *NIPS BigLearn Workshop*, 2011.

[61] J.-F. Collumeau, H. Laurent, B. Emile, and R. Leconge. Hand Posture Recognition with Multiview Descriptors. *ACIVS*, 2012.

[62] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5), 2002.

[63] S. Conseil, S. Bourennane, and L. Martin. Comparison of Fourier Descriptors and Hu Moments for Hand Posture Recognition. *EUSIPCO*, 2007.

[64] C. Couprie, C. Farabet, L. Najman, and Y. LeCun. Indoor Semantic Segmentation using depth information. *ICLR*, 2014.

[65] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematicsof Control, Signals, and Systems*, 2:303–314, 1989.

[66] E. Fromont D. Muselet N. Neverova A. Tremeau C. Wolf D. Fourure, R. Emonet. Multi-task, multi-domain learning: application to semantic segmentation and pose regression. 2016.

[67] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. *CVPR*, 2005.

[68] A. Das, N. Borisov, and M. Caesar. Exploring ways to mitigate sensor-based smartphone fingerprinting. Preprint, arXiv:1503.01874, 2015.

[69] S. Das, C. L. Giles, and G.-Z. Sun. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory. *The 14th Annual Conference of Cognitive Science Society*, 1992.

[70] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Transaction on Information Theory*, 36(5):961–1005, 1990.

[71] P. Dayan and L. Abbott. Theoretical neuroscience: Computational and mathematical modeling of neural systems. 2005.

[72] T. E. de Campos and D. W. Murray. Regression-based Hand Pose Estimation from Multiple Cameras. *CVPR*, 2006.

[73] M. de La Gorce, D. Fleet, and N. Paragios. Model-based 3d hand pose estimation from monocular video. *IEEE-Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(9):1793–1805, 2014.

[74] L. Deng, J. Li, J.-T. Huang, K. Yao, et al. Recent advances in deep learning for speech research at Microsoft. *ICASSP*, 2013.

[75] M. O. Derawi, C. Nickel, P. Bours, and C. Busch. Unobtrusive User-Authentication on Mobile Phones Using Biometric Gait Recognition. *IIH-MSP*, 2010.

[76] G. Dewaele, F. Devernay, R. P. Horaud, and F. Forbes. The alignment between 3-d data and articulated shapes with bending surfaces. *ECCV*, 2006.

[77] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior Recognition via Sparse Spatio-Temporal Features. *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[78] P. Dollar, P. Welinder, and P. Perona. Cascaded pose regression. *CVPR*, 2010.

[79] F. Dominio, M. Donadeo, and P. Zanuttigh. Combining multiple depth-based descriptors for hand gesture recognition. *Pattern Recognition Letters*, October 2013.

[80] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. Preprint, arXiv:1503.01874, 2015.

[81] S. Duffner, S. Berlemont, G. Lefebre, and C. Garcia. 3D gesture classification with convolutional neural networks. *ICASSP*, 2014.

[82] T. Duong, H. Bui, D. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. *CVPR*, 2005.

[83] S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. *NIPS*, 1995.

[84] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[85] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108 (1-2):52–73, October 2007.

[86] S. Escalera, J. Gonzàlez, X. Baró, M. Reyes, O. Lopés, I. Guyon, V. Athitsos, and H. J. Escalante. Multi-modal Gesture Recognition Challenge 2013: Dataset and Results. *Chalearn Multi-Modal Gesture Recognition Workshop, ICMI*, 2013.

[87] S. Escalera, X. Baró, J. Gonzàlez, M. A. Bautista, M. Madadi, M. Reyes, V. Ponce, H. J. Escalante, J. Shotton, and I. Guyon. ChaLearn Looking at People Challenge 2014: Dataset and Results. *ECCV ChaLearn Workshop on Looking at People*, 2014.

[88] G. Evangelidis, G. Singh, and R. Horaud. Continuous gesture recognition from articulated poses. *ECCV ChaLearn Workshop on Looking at People*, 2014.

[89] Y. Fang, J. Cheng, J. Wang, K. Wang, J. Liu, and H. Lu. Hand posture recognition with co-training. *ICPR*, 2008.

[90] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning Hierarchical Features for Scene Labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):1915–1929, 2013.

[91] J. Felsenstein. Inferring phylogenies. 2003.

[92] D. Figo, P. Diniz, D. Ferreira, and J. Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Pervasive and Ubiquituous Computing*, 2010.

[93] S. Fothergill, H. M. Mentis, P. Kohli, and S. Nowozin. Instructing people for training gestural interactive systems. *CHI*, 2012.

[94] H. Francke, J. Ruiz-del solar, and R. Verschae. Real-time Hand Gesture Detection and Recognition using Boosted Classifiers and Active Learning. *Second Pacific Rim Symposium, PSIVT*, 2012.

[95] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(11), November 2011.

[96] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real-time human pose tracking from range data. *ECCV*, 2012.

[97] Y. Ganin and V. Lempitsky. N^4-Fields: Neural Network Nearest Neighbor Fields for Image Transforms. *ACCV*, 2014.

[98] O. Gascuel and M. Steel. Neighbor-joining revealed. *Molecular Biology and Evolution*, 23(11):1997–2000, 2006.

[99] P. Gehler and S. Nowozin. On Feature Combination for Multiclass Object Classification. *ICCV*, 2009.

[100] F. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continuous prediction with lstm. Technical Report IDSIA-01-99, 2000.

[101] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[102] Z. Ghahramani. An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 2001.

[103] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. *ICCV*, 2011.

[104] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[105] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[106] A. Giusti, D. C. Ciresan, J Masci, L. M. Gambardella, and J. Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. *ICIP*, 2013.

[107] G. Gkioxari and J. Malik. Finding action tubes. *CVPR*, 2015.

[108] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout Networks. Preprint, arXiv:1302.4389v4, 2013.

[109] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Book in preparation for MIT Press, 2016. URL http://goodfeli.github.io/dlbook/.

[110] A. Graves. Supervised sequence labelling with recurrent neural networks. *Studies in Computational Intelligence*, 2012.

[111] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. Preprint, arXiv:1410.5401, 2014.

[112] K. Greff, R. K. Srivastava, J. Koutnık, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. Preprint, arXiv:1503.04069, 2015.

[113] R. Grosse, R. Salakhutdinov, W. T. Freeman, and J. B. Tenenbaum. Exploiting compositionality to explore a large space of model structures. *UAI*, 2012.

[114] H. Guan, R. S. Feris, and M. Turk. The Isometric Self-Organizing Map for 3D Hand Pose Estimation. *FGR*, 2006.

[115] L. Guillaume, V. Aubergé, R. Magnani, F. Aman, C. Cottier, et al. Gestural hri in an ecological dynamic experiment: the gee corpus based approach for the emox robot. *International Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 2015.

[116] I. Guyon, V. Athitsos, P. Jangyodsuk, and H. J. Escalante. The chalearn gesture dataset (cgd 2011). *Technical Report*, 2011.

[117] I. Guyon, V. Athitsos, P. Jangyodsuk, and B. Hamner. ChaLearn Gesture Challenge: Design and First Results. *CVPR Workshop on Gesture Recognition and Kinect Demonstration Competition*, 2012.

[118] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a Hand Manipulating an Object. *ICCV*, 2009.

[119] A. Hannun, C. Case, J. Casper, B. Catanzaro, et al. Deepspeech: Scaling up end-to-end speech recognition. Preprint, arXiv:1412.5567, 2014.

[120] M. M. Hasan and P. K. Mishra. Hand Gesture Modeling and Recognition using Geometric Features: A Review. *Canadian Journal on Image Processing and Computer Vision*, 3(1), 2012.

[121] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. Preprint, arXiv:1512.03385, 2015.

[122] D. O. Hebb. The organization of behavior. 1949.

[123] A. Heili, A. Lopez-Menez, and J.-M. Odobez. Exploiting long-term connectivity and visual motion in crf-based multi-person tracking. *IEEE Transactions on Image Processing*, 23(7):3040–3056, 2014.

[124] A. Hernández-Vela and N. Zlateva. Human limb segmentation in depth maps based on spatio-temporal Graph-cuts optimization. *Journal of Ambient Intelligence and Smart Environments*, 4:535–546, 2012.

[125] A. Hernández-Vela, M. Á. Bautista, X. Perez-Sala, V. Ponce-López, S. Escalera, X. Baró, O. Pujol, and C. Angulo. Probability-based Dynamic Time Warping and Bag-of-Visual-and-Depth-Words for Human Gesture Recognition in RGB-D. *Pattern Recognition Letters*, September 2013.

[126] G. E. Hinton, Simon Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[127] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and Salakhutdinov R. R. Improving neural networks by preventing co-adaptation of feature detectors. Preprint, arXiv:1207.0580, 2012.

[128] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[129] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[130] B. Holt, E.-J. Ong, H. Cooper, and R. Bowden. Putting the pieces together: Connected Poselets for human pose estimation. *ICCV Workshops*, 2011.

[131] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks areuniversal approximators. *Neural Networks*, 2:359–366, 1989.

[132] D.-A. Huang, M. Ma, W.-C. Ma, and K. M. Kitani. How do we use our hands? discovering a diverse set of common grasps. *CVPR*, 2015.

[133] N. A. Ibraheem and R. Z. Khan. Vision Based Gesture Recognition Using Neural Networks Approaches: A Review. *International Journal of Human Computer Interaction (IJHCI)*, (3):1–14, 2012.

[134] J. S. Supancic III, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan. Depth-based hand pose estimation: methods, data, and challenges. *ICCV*, 2015.

[135] N. Ikizler and D. Forsyth. Searching for Complex Human Activities with No Visual Examples. *International Journal of Computer Vision*, 80(3):337–357, May 2008.

[136] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. Preprint, arXiv:1502.03167v3, 2015.

[137] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler. Learning human pose estimation features with convolutional networks. *ICLR*, 2014.

[138] Y. Jang, S.-T. Noh, H. J. Chang, and T.-K. Kim. 3d finger cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint. *IEEE Transactions on Visualization and Computer Graphics*, 21(4):501–510, 2015.

[139] S. Ji, W. Xu, M. Yang, and K. Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions of Pattern Analysis and Machine Intelligence (TPAMI)*, 35(1):221–31, January 2013.

[140] M. Jiu, C. Wolf, G. Taylor, and A. Baskurt. Human body part estimation from depth images via spatially-constrained deep learning. *Pattern Recognition Letters*, 50(1):122–129, 2014.

[141] Mingyuan Jiu, Christian Wolf, Graham Taylor, and Atilla Baskurt. Human body part estimation from depth images via spatially-constrained deep learning. *Pattern Recognition Letters*, October 2013.

[142] A. Joulin and T. Mikolov. Inferring algorithmic paterns with stack-augmented recurrent nets. *NIPS*, 2015.

[143] R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. *ICML*, 2015.

[144] A. Just, Y. Rodriguez, and S. Marcel. Hand Posture Classification and Recognition using the Modified Census Transform. *FGR*, 2006.

[145] S. E. Kahou, C. Pal, X. Bouthillier, P. Froumenty, Ç. Gülçehre, R. Memisevic, P. Vincent, A. Courville, and Y. Bengio. Combining modality specific deep neural networks for emotion recognition in video. *ICMI*, 2013.

[146] L. Kaiser and I. Sutskever. Neural gpus learn algorithms. Preprint, arXiv:1511.08228v2, 2015.

[147] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3D mesh segmentation and labeling. *ACM Transactions on Graphics*, 29(3), 2010.

[148] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F.-F. Li. Large-scale Video Classification with Convolutional Neural Networks. *CVPR*, 2014.

[149] T. Kekeç, R. Emonet, E. Fromont, A. Trémeau, and C. Wolf. Contextually Constrained Deep Networks for Scene Labeling. *BMVC*, 2014.

[150] J. Kennedy and R. Eberhart. Particle swarm optimization. *International Conference on Neural Networks*, 1995.

[151] C. Keskin, F. Kiraç, Y. E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. *ICCV Workshops*, 2011.

[152] C. Keskin, F. Kirac, Y. E. Kara, and L. Akarun. Hand Pose Estimation and Hand Shape Classification Using Multi-layered Randomized Decision Forests. *ECCV*, 2012.

[153] C. Keskin, F. Kirac, Y. E. Kara, and L. Akarun. Multi-layered Randomized Classification Forests for Hand Pose Estimation using Depth Sensors. *CVPR Workshop on Gesture Recognition*, 2012.

[154] R. Z. Khan and N. A. Ibraheem. Comparative study of hand gesture recognition system. *International Conference of Advanced Computer Science & Information Technology (CS & IT)*, 2012.

[155] R. Z. Khan and N. A. Ibraheem. Hand gesture recognition: a literature review. *International Journal of Artificial Intelligence & Applications (IJAIA)*, 3(4):161–174, 2012.

[156] E. Khoury, L. El Shafey, C. McCool, M. Gunther, and S. Marcel. Bi-Modal Biometric Authentication on Mobile Phones in Challenging Conditions. *Image and Vision Computing*, 2014.

[157] D. Kingma and J. L. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[158] A Kläser, M Marszalek, and C Schmid. A spatio-temporal descriptor based on 3D-gradients. *BMVC*, 2008.

[159] E. Kollorz, J. Penne, and J. Hornegger. Gesture recognition with a time-of-flight camera. *International Journal of Intelligent Systems Technologies and Applications*, 5(3/4):334–343, 2008.

[160] P. Kontschieder, S.R. Bulo, M. Pelillo, and H. Bischof. Structured labels in random forests for semantic labelling and object detection. *ICCV*, 2011.

[161] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber. A clockwork rnn. *ICML*, 2014.

[162] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, 2012.

[163] P. Kulkarni, J. Zepeda, F. Jurie, P. Perez, and L. Chevallier. Learning the structure of deep architectures using l1 regularization. *BMVC*, 2015.

[164] A. Kurakin, Z. Zhang, and Z. Liu. A real time system for dynamic hand gesture recognition with a depth sensor. *20th European Signal Processing Conference (EUSIPCO 2012)*, pages 1975–1979, 2012.

[165] K. J Lang, A. H. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.

[166] I. Laptev and T. Lindeberg. Space-time interest points. *ICCV*, 2003.

[167] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. *CVPR*, 2008.

[168] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *CVPR*, 2006.

[169] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. *CVPR*, 2011.

[170] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. Preprint, arXiv:1504.00941v2, 2015.

[171] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[172] A. Lee, T. Kawahara, and K. Shikano. Julius - an open source real-time large vocabulary recognition engine. *INTERSPEECH*, pages 1691–1694, 2001.

[173] G. Lefebvre, S. Berlemont, F. Mamalet, and C. Garcia. BLSTM-RNN Based 3D Gesture Classification. *ICANN*, 2013.

[174] E. L. Lehmann. Elements of Large-Sample Theory. *ICML*, 1998.

[175] W. Li, Z. Zhang, and Z. Liu. Expandable Data-Driven Graphical Modeling of Human Actions Based on Salient Postures. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1499–1510, 2008.

[176] W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3d points. *CVPR*, 2010.

[177] H. Liang, , J. Yuan, D. Thalmann, and Z. Zhang. Model-based hand pose estimation via spatial-temporal hand parsing and 3D fingertip localization. *The Visual Computer*, 29:837–848, 2013.

[178] J. Lin and Y. Ding. A temporal hand gesture recognition system based on hog and motion trajectory. *Optik - International Journal for Light and Electron Optics*, 124(24):6795–6798, December 2013.

[179] M. Lin, Q. Chen, and S. Yan. Network in network. *ICLR*, 2014.

[180] D. Liu, K.-T. Lai, G. Ye, M.-S. Chen, and S.-F. Chang. Sample-Specific Late Fusion for Visual Category Recognition. *CVPR*, 2013.

[181] L. Liu and L. Shao. Learning discriminative representations from rgb-d video data. 2013.

[182] L. Liu, J. Xing, H. Ai, and X. Ruan. Hand Posture Recognition Using Finger Geometric Feature. *ICPR*, 2012.

[183] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, November 2004.

[184] F. Lv and R. Nevatia. Recognition and Segmentation of 3-D Human Acition using HMM and Multi-Class AdaBoost. *ECCV*, 2006.

[185] A. Malima and M. Özgür, E .and Çetin. A fast algorithm for vision-based hand gesture recognition for robot control. *IEEE 14th Conference on Signal Processing and Communications Applications*, 2006.

[186] J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. *ICML*, 2011.

[187] C. M. Mateo, P. Gil, J. A. Corrales, S. T. Puente, and F. Torres. RGBD Human-Hand recognition for the Interaction with Robot-Hand. *IROS*, 2012.

[188] R. Memisevic and G. Hinton. Unsupervised learning of image transformations. *CVPR*, 2007.

[189] M. Á. Mendoza and N. P. de la Blanca. Applying Space State Models in Human Action Recognition: A Comparative Study. *5th International Conference on Articulated Motion and Deformable Objects*, pages 53–62, 2008.

[190] L. Mestetskiy, I. Bakina, and A. Kurakin. Hand geometry analysis by continuous skeletons. *Image Analysis and Recognition*, 2011.

[191] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato. Learning longer memory in nrecurrent neural networks. Preprint, arXiv:1412.7753, 2014.

[192] Y. Ming, Q. Ruan, and A. G. Hauptmann. Activity Recognition from RGB-D Camera with 3D Local Spatio-temporal Features. *ICME*, 2012.

[193] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(3):311–324, 2007.

[194] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand gesture recognition with 3d convolutional neural networks. *CVPR Workshops*, 2015.

[195] C. Monnier, S. German, and A. Ost. A Multi-scale Boosted Detector for Efficient and Robust Gesture Recognition. *ECCV ChaLearn Workshop on Looking at People*, 2014.

[196] M. C. Mozer and S. Das. A connectionist symbol manipulator that discovers the structure of context-free languages. *NIPS*, 1993.

[197] M. Muja and D. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP*, 2009.

[198] M Müller, T Röder, and M. Clausen. Efficient content-based retrieval of motion capture data. *ACM Transactions on Graphics (TOG)*, 1(212):677–685, 2005.

[199] K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, 2002.

[200] G. R. S. Murthy and R. S. Jadon. A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management (IJITKM)*, 2(2):405–410, 2009.

[201] G. Lacey L. Fridman D. Chandra B. Barbello G. Taylor N. Neverova, C. Wolf. Learning human identity from motion patterns. *arXiv:1511.03908v3*, 2015.

[202] G. Paci G. Sommavilla G. Taylor F. Nebout N. Neverova, C. Wolf. Multi-scle approach to gesture detection and recognition. *ICCV Workshop on Understanding Human Activities: Context and Interactions*, 2013.

[203] G. Taylor F. Nebout N. Neverova, C. Wolf. Hand segmentation with structured convolutional learning. *ACCV*, 2014.

[204] G. Taylor F. Nebout N. Neverova, C. Wolf. Moddrop: adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, PP, 2015.

[205] J. Nagi, F. Ducatelle, G. Di Caro, D. Ciresan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. *IEEE International Conference on Signal and Image Processing Applications*, 2011.

[206] K. Nandakumar, W. K. Wah, S. M. A. Chan, W. Z. T. Ng, J. G. Wang, and W. Y. Yun. A Multi-modal Gesture Recognition System Using Audio, Video, and Skeletal Joint Data Categories and Subject Descriptors. *ICMI Workshop*, 2013.

[207] J. Napier. The prehensile movements of the human hand. *Journal of Bone and Joint Surgery*, 1956.

[208] P. Natarajan and R. Nevatia. Online, Real-time Tracking and Recognition of Human Actions. *IEEE Workshop on Motion and video Computing*, 2008.

[209] P. Natarajan, S. Wu, S. Vitaladevuni, X. Zhuang, S. Tsakalidis, U. Park, R. Prasad, and P. Natarajan. Multimodal Feature Fusion for Robust Event Detection in Web Videos. *CVPR*, 2012.

[210] N. Neverova, C. Wolf, F. Nebout, and G. Taylor. Hand pose estimation through weakly-supervised learning of a rich intermediate representation. *Preprint, arXiv:1511.06728v1*, 2015.

[211] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *CVPR*, 2015.

[212] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. *ICML*, 2011.

[213] C. Nickel, H. Brandt, and C. Busch. Benchmarking the performance of svms and hmms for accelerometer-based biometric gait recognition. *ISSPIT*, 2011.

[214] S. Nilsson. https://flic.kr/p/qndn43, license cc by-sa 2.0.

[215] M. Oberweger, P. Wohlhart, and V. Lepetit. Hands deep in deep learning for hand pose estimation. *CVWW*, 2015.

[216] M. Oberweger, P. Wohlhart, and V. Lepetit. Training a feedback loop for hand pose estimation. *ICCV*, 2015.

[217] I. Oikonomidis, N. Kyriazis, and A. Argyros. Markerless and efficient 26-dof hand pose recovery. *ACCV*, 2010.

[218] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. *BMVC*, 2011.

[219] I. Oikonomidis, N. Kyriazis, and A. Argyros. Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. *ICCV*, 2011.

[220] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Tracking the Articulated Motion of Two Strongly Interacting Hands. *CVPR*, 2012.

[221] I. Oikonomidis, M. I. A. Lourakis, and A. A. Argyros. Evolutionary quasi-random search for hand articulations tracking. *CVPR*, 2014.

[222] M.-A. Okkonen, V. Kellokumpu, M. Pietikainen, and J. Heikkila. A Visual System for Hand Gesture Recognition in Human-Computer Interaction. *15th Scandinavian Conference on Image Analysis (SCIA)*, 2007.

[223] C. Olah. Understanding lstm networks *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*, 2015.

[224] M. Osadchy, Y. LeCun, and M. L. Miller. Synergistic face detection and pose estimation with energy-based model. *NIPS*, 2005.

[225] X. Li C. Liao P. Li, H. Ling. 3d hand pose estimation using randomized decision forest with segmentation index points. *ICCV*, 2015.

[226] S. Padam Priyal and P. K. Bora. A robust static hand gesture recognition system using geometry based normalizations and Krawtchouk moments. *Pattern Recognition*, 46(8):2202–2219, August 2013.

[227] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. Preprint, arXiv:1211.5063, 2012.

[228] F. Pedersoli and N. Adami. XKin -: eXtendable hand pose and gesture recognition library for kinect. *ACM international conference on Multimedia*, pages 1465–1468, 2012.

[229] S. Pellegrini, K. Schindler, and D. Nardi. A generalisation of the icp algorithm for articulated bodies. *BMVC*, 2008.

[230] X. Peng, L. Wang, and Z. Cai. Action and Gesture Temporal Spotting with Super Vector Representation. *ECCV ChaLearn Workshop on Looking at People*, 2014.

[231] P. Peursum, S. Venkatesh, and G. West. Tracking-as-Recognition for Articulated Full-Body Human Motion Analysis. *CVPR*, 2007.

[232] L. Pigou, S. Dieleman, and P.-J. Kindermans. Sign Language Recognition Using Convolutional Neural Networks. *ECCV ChaLearn Workshop on Looking at People*, 2014.

[233] L. Pigou, A. van der Oord, S. Dieleman, M. van Herreweghe, and J. Dambre. Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. Preprint, arXiv:1506.01911, 2015.

[234] P. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. 2015. CVPR.

[235] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun. Real-time identification and localization of body parts from depth images. *ICRA*, 2010.

[236] T. Plotz, N. Y. Hammerla, and P. Olivier. Feature learning for activity recognition in ubiquitous computing. *22nd International Joint Conference on Artificial Intelligence*, 2011.

[237] G. Poier, K. Roditakis, S. Schulter, D. Michel, H. Bischof, and A. Argyros. Hybrid one-shot 3d hand pose estimation by exploiting uncertainties. *Preprint, arXiv:1510.08039*, 2015.

[238] Y. Poleg, A. Ephrat, S. Peleg, and C. Arora. Compact cnn for indexing egocentric videos. *CoRR*, 2015.

[239] M. Półrola and A. Wojciechowski. Real-time hand pose estimation using classifiers. *Computer Vision and Graphics*, pages 573–580, 2012.

[240] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, June 2010.

[241] C. Prema and D. Manimegalai. Survey on Skin Tone Detection using Color Spaces. *International Journal of Applied Information Systems*, 2(2):18–26, 2012.

[242] N. Pugeault and R. Bowden. Spelling it out: Real-time ASL fingerspelling recognition. *ICCV Workshops*, 2011.

[243] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and Robust Hand Tracking from Depth. *CVPR*, 2014.

[244] N. L. Zhang T. Liu R. Mourad, C. Sinoquet and P. Leray. A survey on latent tree models and applications. *JAIR*, 47:157-203, 2013.

[245] M. A. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. *CVPR*, 2007.

[246] M. Raptis, D. Kirovski, and H. Hoppe. Real-time classification of dance gestures from skeleton animation. *Symposium on Computer Animation*, 2011.

[247] Z. Ren, J. Yuan, and Z. Zhang. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. *19th ACM international conference on Multimedia*, 2011.

[248] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10:19–41, 2000.

[249] B. Rimé, L. Schiaratura, M. Hupet, and A. Ghysselinckx. Effects of relative immobilization on the speaker's nonverbal behavior and on the dialogue imagery level. *Motivation and Emotion*, 8, 1984.

[250] G. Rogez, J. S. Supancic III, and D. Ramanan. Understanding everyday hands in action from rgb-d images. *ICCV*, 2015.

[251] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3D hand pose reconstruction using specialized mappings. *Computer Vision*, 2000, 2001.

[252] F. Rosenblatt. The perceptron: A probabilistic model for information storage andorganization in the brain. *Psychological Review*, 65:386–408, 1958.

[253] S. Ruffieux, D. Lalanne, and E. Mugellini. Chairgest: A challenge for multimodal mid-air gesture recognition for close hci. *ICMI*, 2013.

[254] D. Rumelhart, G. Hinton, and R. Williams. Learning representations byback-propagating error. *Nature*, 323:533–536, 1986.

[255] R. B. Rusu and G. Bradski. Fast 3d recognition and pose using the viewpoint feature histogram. *IROS*, 2010.

[256] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. *Robotics and Automation*, 2009.

[257] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. *ICASSP*, 2015.

[258] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[259] J. Schmidhuber. Deep learning in neural networks: An overview. 2014.

[260] S. J. Schmugge, S. Jayaram, M. C. Shin, and L. V. Tsap. Objective evaluation of approaches of skin detection using ROC analysis. *Computer vision and image understanding*, 108(1-2):41–51, 2007.

[261] P. Scovanner, S. Ali, and M. Shah. A 3-Dimensional SIFT Descriptor and its Application to Action Recognition. *15th International Conference on Multimedia*, pages 357–360.

[262] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *ICLR*, 2014.

[263] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. Preprint, arXiv:1511.04119v2, 2015.

[264] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, et al. Accurate, robust, and flexible real-time hand tracking. *CHI*, 2015.

[265] K. R. Shastry and M. Ravindran. Survey on various gesture recognition techniques for interfacing machines based on a ambient intelligence. *International Journal of Computer Science & Engineering Survey (IJCSES)*, (1(2)), 2010.

[266] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. *CVPR*, 2011.

[267] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, et al. Efficient human pose estimation from single depth images. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2821–40, December 2013.

[268] S. Wang and C. Manning. Fast dropout training. *ICML*, 2013.

[269] M. Siddiqui and G. Medioni. Human pose estimation from a single view point, real-time range sensor. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, 2010.

[270] G. Simion, V. Gui, and M. Otesteanu. Vision based hand gesture recognition: A review. *International Journal of Circuits, Systems and Signal Processing*, 6:275–282, 2012.

[271] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. Preprint, arXiv:1406.2199v1, 2014.

[272] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[273] Z. Sitova, J. Sedenka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. Balagani. HMOG: A New Biometric Modality for Continuous Authentication of Smartphone Users. Preprint, arXiv:1501.01199, 2015.

[274] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Conditional models for contextual human motion recognition. *CVPR*, 2005.

[275] S. Sridhar, A. Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using RGB and depth data. *ICCV*, 2013.

[276] N. Srivastava and R. Salakhutdinov. Multimodal learning with Deep Boltzmann Machines. *NIPS*, 2013.

[277] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. Preprint, arXiv:1502.04681v3, 2016.

[278] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *NIPS*, 2015.

[279] E. Stergiopoulou and N. Papamarkos. Hand gesture recognition using a neural network shape fitting technique. *Engineering Applications of Artificial Intelligence*, 22(8):1141–1158, December 2009.

[280] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks. Preprint, arXiv:1503.08895v5, 2015.

[281] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun. Cascaded hand pose regression. *CVPR*, 2015.

[282] J. Sung and C. Ponce. Unstructured human activity detection from rgbd images. *Robotics and Automation*, 2012.

[283] J. Sung, C. Ponce, B. Selman, and A. Saxena. Human Activity Detection from RGBD Images. *Activity, and Intent Recognition*, 2011.

[284] P. Suryanarayan, A. Subramanian, and D. Mandalapu. Dynamic Hand Pose Recognition Using Depth Data. *ICPR*, 2010.

[285] C. Szegedy, W. Liu, J. Yangqing, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich et al. Going deeper with convolutions. Preprint, arXiv:1502.03167v3, 2015.

[286] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *CVPR*, 2014.

[287] H. Takimoto, J. Lee, and A. Kanagawa. A Robust Gesture Recognition Using Depth Data. *International Journal of Machine Learning and Computing*, 3(2):245–249, 2013.

[288] D. Tang, T.-H. Yu, and T.-K. Kim. Real-time Articulated Hand Pose Estimation using Semi-supervised Transductive Regression Forests. *ICCV*, 2013.

[289] D. Tang, H. J. Chang, A. Tejani, and T.-K. Kim. Latent Regression Forest: Structured Estimation of 3D Articulated Hand Posture. *CVPR*, 2014.

[290] D. Tang, J. Taylor, K. Pushmeet, C. Keskin, T.-K. Kim, and J. Shotton. Opening the black box: Hierarchical sampling optimization for estimating human hand pose. *ICCV*, 2015.

[291] G. Taylor, R. Fergus, Y LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. *ECCV*, 2010.

[292] J. Taylor, R. Stebbing, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Fitzgibbon, and A. Hertzmann. User-specific hand modeling from monocular depth sequences. *CVPR*, 2014.

[293] B. Tessendorf, F. Gravenhorst, B. Arnrich, and G. Troster. An imu-based sensor network to convinuously monitor rowing technique on the water. *ISSNIP*, 2011.

[294] A Thippur, C. H. Ek, and H. Kjellström. Inferring Hand Pose: A Comparative Study of Visual Shape Features. *FG*, 2013.

[295] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *NIPS*, 2014.

[296] J. Tompson, M. Stein, Y. LeCun, and K. Perlin. Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks. *ACM Transaction on Graphics*, 2014.

[297] A. Toshev and C. Szegedy. DeepPose: Human pose estimation via deep neural networks. *CVPR*, 2014.

[298] Z. Tu, X. Chen, A. L. Yuille, and S. C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *IJCV*, 63(2):113–140, 2005.

[299] D. Uebersax, J. Gall, M. Van den Bergh, and L. Van Gool. Real-time sign language letter and word recognition from depth data. *CVPR*, 2011.

[300] M. Vafadar and A. Behrad. Human Hand Gesture Recognition Using Motion Orientation Histogram for Interaction of Handicapped Persons with Computer. *ICISP*, 2008.

[301] M. Van den Bergh and L. Van Gool. Combining RGB and ToF cameras for real-time 3D hand gesture interaction. *IEEE Workshop on Applications of Computer Vision (WACV)*, 2011.

[302] G. Varol, I. Laptev, and C. Schmid. Long-term Temporal Convolutions for Action Recognition. working paper or preprint, December 2015. URL https://hal.inria.fr/hal-01241518.

[303] A. W. Vieira, E. R. Nascimento, and G. L. Oliveira. Stop: Space-time occupancy patterns for 3d action recognition from depth map sequences. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 2012.

[304] E. Vildjiounaite, S.-M. Makela, Mikko Lindholm, R. Riihimaki, et al. Unobtrusive multimodal biometrics for ensuring privacy and information security with personal devices. *Pervasive Computing*, 2006.

[305] A. Wang, H.and Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, 2013.

[306] C.-C. Wang and K.-C. Wang. Hand Posture Recognition Using Adaboost with SIFT for Human Robot Interaction. 2008.

[307] Fang Wang and Yi Li. Beyond Physical Connections: Tree Models in Human Pose Estimation. *CVPR*, 2013.

[308] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. *BMVC*, 2009.

[309] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Mining actionlet ensemble for action recognition with depth cameras. *CVPR*, 2012.

[310] R. Y. Wang and J. Popović. Real-Time Hand-Tracking with a Color Glove. *SIGGRAPH*, 2009.

[311] S. B. Wang, A. Quattoni, L.-P. Morency, D. Demirdjian, and T. Darrell. Hidden Conditional Random Fields for Gesture Recognition. *CVPR*, 2006.

[312] D. Weinland, R. Ronfard, and E. Boyer. A Survey of Vision-Based Methods for Action Representation, Segmentation and Recognition. Technical Report 2, 2010.

[313] P. Weinzaepfel, J. Revaud, and C. Harchaoui, Z.and Schmid. Deepflow: Large displacement optical flow with deep matching. *ICCV*, 2013.

[314] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. pages 639–655. 2012.

[315] J. Weston, S. Chopra, and A. Bordes. Memory networks. *ICLR*, 2015.

[316] A. Wetzler, R. Slossberg, and R. Kimmel. Rule of thumb: Deep derotation for improved fingertip detection. *BMVC*, 2015.

[317] G. Willems, T. Tuytelaars, and L. Van Gool. An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector. *ECCV*, 2008.

[318] D. Wu. Deep Dynamic Neural Networks for Gesture Segmentation and Recognition. *ECCV ChaLearn Workshop on Looking at People*, 2014.

[319] J. Wu and H. Lu. Fusing Multi-modal Features for Gesture Recognition Categories and Subject Descriptors. *ICMI*, 2013.

[320] Y. Wu and T. S. Huang. Vision-based gesture recognition: A review. *Urbana*, 1999.

[321] L. Xia and J. K. Aggarwal. Spatio-Temporal Depth Cuboid Similarity Feature for Activity Recognition Using Depth Camera. *CVPR*, 2013.

[322] L. Xia, C. C. Chen, and J. K. Aggarwal. View invariant human action recognition using histograms of 3d joints. *CVPR workshops*, 2012.

[323] C. Xu and L. Cheng. Efficient hand pose estimation from a single depth image. *ICCV*, 2013.

[324] Z. Xu, Y. Yang, I. Tsang, N. Sebe, and A. G. Hauptmann. Feature Weighting via Optimal Thresholding for Video Analysis. *ICCV*, 2013.

[325] Q. Yang, Ge Peng, D. T. Nguyen, X. Qi, G. Zhou, Z. Sitova, P. Gasti, and K. S. Balagani. A multimodal data set for evaluating continuous authentication performance in smartphones. 2014.

[326] X. Yang and Y. L. Tian. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. *CVPR*, 2012.

[327] X. Yang, C. Zhang, and Y. Tian. Recognizing actions using depth motion maps-based histograms of oriented gradients. *ACM international conference on Multimedia*, 2012.

[328] A. Yao, J. Gall, and L. Van Gool. Coupled Action Recognition and Pose Estimation from Multiple Views. *International journal of computer vision*, 2012.

[329] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. *ICCV*, 2015.

[330] Y. Yao and C. T. Li. Hand Posture Recognition Using SURF with Adaptive Boosting. *BMVC*, 2012.

[331] T. Yasuda, K. Ohkura, and Y. Matsumura. Extended pso with partial randomization for large scale multimodal problems. *World Automation Congress*, 2010.

[332] G. Ye, D. Liu, I-H. Jhuo, and S.-F. Chang. Robust Late Fusion With Rank Minimization. *CVPR*, 2012.

[333] M. Ye, Q. Zhang, L. Wang, J. Zhu, R. Yang, and J. Gall. A Survey on Human Motion Analysis from Depth Data. *Dagstuhl 2012 Seminar on Time-of-Flight Imaging and GCPR 2013 Workshop on Imaging New Modalities*, 2013.

[334] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The Moving Pose: An Efficient 3D Kinematics Descriptor for Low-Latency Action Recognition and Detection. *ICCV*, 2013.

[335] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang. Convolutional neural networks for human activity recognition using mobile sensors. *MobiCASE*, 2014.

[336] C. Zhang, X. Yang, and Y. Tian. Histogram of 3D Facets: A characteristic descriptor for hand gesture recognition. *FG*, 2013.

[337] Y. Zhao, Z. Liu, L. Yang, and H. Cheng. Combining RGB and Depth Map Features for Human Activity Recognition. *Asia-Pasific Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2012.

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : Neverova
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 08/04/2016

Prénoms : Natalia

TITRE : Deep Learning for Human Motion Analysis

NATURE : Doctorat

Numéro d'ordre : 2016LYSEI029

Ecole doctorale : Informatique et Mathématique de Lyon (INFOMATHS), ED 512

Spécialité : Informatique

RESUME : The research goal of this work is to develop learning methods advancing automatic analysis and interpreting of human motion from different perspectives and based on various sources of information, such as images, video, depth, mocap data, audio and inertial sensors. For this purpose, we propose a several deep neural models and associated training algorithms for supervised classification and semi-supervised feature learning, as well as modelling of temporal dependencies, and show their efficiency on a set of fundamental tasks, including detection, classification, parameter estimation and user verification.
First, we present a method for human action and gesture spotting and classification based on multi-scale and multi-modal deep learning from visual signals (such as video, depth and mocap data). Key to our technique is a training strategy which exploits, first, careful initialization of individual modalities and, second, gradual fusion involving random dropping of separate channels (dubbed ModDrop) for learning cross-modality correlations while preserving uniqueness of each modality-specific representation.
Moving forward, from 1 to N mapping to continuous evaluation of gesture parameters, we address the problem of hand pose estimation and present a new method for regression on depth images, based on semi-supervised learning using convolutional deep neural networks, where raw depth data is fused with an intermediate representation in the form of a segmentation of the hand into parts.
In separate but related work, we explore convolutional temporal models for human authentication based on their motion patterns. In this project, the data is captured by inertial sensors (such as accelerometers and gyroscopes) built in mobile devices. We propose an optimized shift-invariant dense convolutional mechanism and incorporate the discriminatively-trained dynamic features in a probabilistic generative framework taking into account temporal characteristics. Our results demonstrate, that human kinematics convey important information about user identity and can serve as a valuable component of multi-modal authentication systems.

MOTS-CLÉS : Action and gesture recognition, human motion analysis, deep learning

Laboratoire (s) de recherche : LIRIS

Directeur de thèse : Christian Wolf, Graham Taylor

Président de jury : Atilla Baskurt

Composition du jury : Frédéric Jurie, Jean-Marc Odobez, Atilla Baskurt, Horst Bischof, Cordelia Schmid, Nicolas Thome, Christian Wolf, Graham Taylor

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