HAL
open science

Deep Regression Models and Computer Vision Applications for Multiperson Human-Robot Interaction

Stéphane Lathuiliere

To cite this version:

Stéphane Lathuiliere. Deep Regression Models and Computer Vision Applications for Multiperson Human-Robot Interaction. Human-Computer Interaction [cs.HC]. Université Grenoble Alpes, 2018. English. NNT : 2018GREAM026 . tel-01801807v2

HAL Id: tel-01801807

https://theses.hal.science/tel-01801807v2

Submitted on 29 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR LA COMMUNAUTE UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Mathématiques et Informatique**

Arrêté ministériel : 25 Mai 2016

Présentée par

Stéphane Lathuilière

Thèse dirigée par **Radu Horaud**

préparée au sein **l'INRIA Grenoble Rhône-Alpes**

et de **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Modèles profonds de regression et applications à la vision par ordinateur pour l'interaction homme-robot

Deep Regression Models and Computer Vision Applications for Multi-person Human-Robot Interaction

Thèse soutenue publiquement le **22 Mai 2018**,
devant le jury composé de :

Dr Josef Sivic
INRIA Paris, Rapporteur
Pr Elisa Ricci
University of Perugia, Rapporteur
Dr Xavier Alameda-Pineda
INRIA Grenoble Rhône-Alpes, Examinateur
Pr Christian Wolf
INSA Lyon, Examinateur
Dr. Radu Horaud
INRIA Grenoble Rhône-Alpes, Directeur de thèse
Dr Cordelia Schmid
INRIA Grenoble Rhône-Alpes, Président

Abstract

In order to interact with humans, robots need to perform basic perception tasks such as face detection, human pose estimation or speech recognition. However, in order have a natural interaction with humans, the robot needs to model high level concepts such as speech turns, focus of attention or interactions between participants in a conversation. In this manuscript, we follow a top-down approach. On the one hand, we present two high-level methods that model collective human behaviors. We propose a model able to recognize activities that are performed by different groups of people jointly, such as *queueing*, *talking*. Our approach handles the general case where several group activities can occur simultaneously and in sequence. On the other hand, we introduce a novel neural network-based reinforcement learning approach for robot gaze control. Our approach enables a robot to learn and adapt its gaze control strategy in the context of human-robot interaction. The robot is able to learn to focus its attention on groups of people from its own audio-visual experiences.

Second, we study in detail deep learning approaches for regression problems. Regression problems are crucial in the context of human-robot interaction in order to obtain reliable information about head and body poses or the age of the persons facing the robot. Consequently, these contributions are really general and can be applied in many different contexts. First, we propose to couple a Gaussian mixture of linear inverse regressions with a convolutional neural network. Second, we introduce a Gaussian-uniform mixture model in order to make the training algorithm more robust to noisy annotations. Finally, we perform a large-scale study to measure the impact of several architecture choices and extract practical recommendations when using deep learning approaches in regression tasks. For each of these contributions, a strong experimental validation has been performed with real-time experiments on the NAO robot or on large and diverse data-sets.

Résumé

Dans le but d'interagir avec des êtres humains, les robots doivent effectuer des tâches de perception basique telles que la détection de visage, l'estimation de la pose des personnes ou la reconnaissance de la parole. Cependant, pour interagir naturellement, avec les hommes, le robot doit modéliser des concepts de haut niveau tels que les tours de paroles dans un dialogue, le centre d'intérêt d'une conversion, ou les interactions entre les participants. Dans ce manuscrit, nous suivons une approche ascendante (dite "top-down"). D'une part, nous présentons deux méthodes de haut niveau qui modélisent les comportements collectifs. Ainsi, nous proposons un modèle capable de reconnatre les activités qui sont effectuées par différents des groupes de personnes conjointement, tels que *faire la queue*, *discuter*. Notre approche gère le cas général où plusieurs activités peuvent se dérouler simultanément et en séquence. D'autre part, nous introduisons une nouvelle approche d'apprentissage par renforcement de réseau de neurones pour le contrôle de la direction du regard du robot. Notre approche permet à un robot d'apprendre et d'adapter sa stratégie de contrôle du regard dans le contexte de l'interaction homme-robot. Le robot est ainsi capable d'apprendre à concentrer son attention sur des groupes de personnes en utilisant seulement ses propres expériences (sans supervision extérieur).

Dans un deuxième temps, nous étudions en détail les approches d'apprentissage profond pour les problèmes de régression. Les problèmes de régression sont cruciaux dans le contexte de l'interaction homme-robot afin d'obtenir des informations fiables sur les poses de la tête et du corps des personnes faisant face au robot. Par conséquent, ces contributions sont vraiment générales et peuvent être appliquées dans de nombreux contextes différents. Dans un premier temps, nous proposons de coupler un mélange gaussien de régressions inverses linéaires avec un réseau de neurones convolutionnels. Deuxièmement, nous introduisons un modèle de mélange gaussien-uniforme afin de rendre l'algorithme d'apprentissage plus robuste aux annotations bruitées. Enfin, nous effectuons une étude à grande échelle pour mesurer l'impact de plusieurs choix d'architecture et extraire des recommandations pratiques lors de l'utilisation d'approches d'apprentissage profond dans des tâches de régression. Pour chacune de ces contributions, une intense validation expérimentale a été effectuée avec des expériences en temps réel sur le robot NAO ou sur de larges et divers ensembles de données.

ACKNOWLEDGMENT

Tout d'abord, je tiens à remercier Radu pour m'avoir encadré pendant ces années de thèse. J'ai passé des très bonnes années dans ton équipe et j'ai pu apprendre énormément à tes cotés. Je te remercie aussi pour la confiance que tu m'as accordé, ta proximité et de ta disponibilité au quotidien. Ta passion et ta personalité y sont pour beaucoup dans l'ambiance de l'équipe qui allie convivialité, bonhumeur et la rigueur de l'excellence scientifique.

Un grand merci aussi à Georgios, Pablo et Xavi pour avoir eu la chance de travailler avec vous. Au delà de votre apport scientifique, je vous remercie également pour très bons et riches moments passés ensemble.

Merci beaucoup Nathalie, pour ton soutient et ta présence à nos cotés.

Je remercie également l'ensemble des membres de l'equipe Perception: Benoit, Fabien, Laurent, Yutong, Dyo, Xiaofei, Silèye, Simon, Rémi, Richard , Bastien, Quentin, Guillaume, Guillaume (le petit), Sylvain, Soraya, Israel, Fata.

Merci à Jean-Francois qui travaille dans l'ombre pour qu'on puisse travailler.

Thanks to Nicu Sebe for giving me the chance to visit and work with his team. Thanks to Gloria, Enver, Moin and Radu that welcomed me so well in there office.

A great thanks to Elisa Ricci and Josef Sivic that reviewed this manuscript. Thank you to my examining committee for thoroughly evaluating my thesis.

Merci à mes amis avec qui j'ai passé tant de bon moments: Clément, Florent, Pauline, Tom, Benoit, Christophe, Lina, Nachwa, Manu, Ines, Boris, les amis du GUCEM et ceux que j'oublie.

Merci à Grenoble et ses montagnes qui m'ont offert un magnifique terrain de jeu. Merci à mon Peugeot carbolite 103 qui a supporté mon poids quotidiennement sur plus de 15000km.

Enfin pour finir, merci à ma famille pour avoir été à mes cotés pendant toutes ces années et à Marie qui me supporte et me soutinent au quotidien.

CONTENTS

1 Introduction **11**

1.1 General Context 11

1.2 Project, Scientific Context and Motivation 12

1.3 Datasets 13

1.4 Contributions 15

1.5 Material Resources 16

1.6 Manuscript Structure 17

2 Recognition of Group Activities Based on Single and Two-Person Descriptors **19**

2.1 Introduction 19

2.2 Related Work 20

2.3 Modeling Group Activity 21

2.4 Two-Person Mapping 23

2.5 Model Learning 24

2.6 Feature Extraction 26

2.7 Experiments 27

2.7.1 Evaluation of the Proposed Model 28

2.7.2 Comparison with State of the Art Methods 30

2.8 Conclusions 31

3 Deep Reinforcement Learning for Audio-Visual Gaze Control in Human-Robot Interaction **33**

3.1 Introduction 34

3.2 Related Work 36
3.3 Reinforcement Learning for Gaze Control 37
3.4 Proposed DQN Architectures 39
3.5 Simulated Environment for Training 41
3.6 Experiments 42
3.6.1 Evaluation with Recorded Data 42
3.6.2 Live Experiments with Nao 42
3.6.3 Implementation Details 43
3.6.4 Results and Discussion 45
3.6.5 Results and Discussion 45
3.7 Conclusions 50

4 Deep Mixture of Linear Inverse Regressions Applied to Head-Pose Estimation 51
4.1 Introduction 51
4.2 Related Work 52
4.3 Mixture of Linear Inverse Regressions 53
4.4 Training the Proposed Model 56
4.5 Experiments 58
4.6 Conclusions 64

5 DeepGUM: Deep Robust Regression with Gaussian-Uniform Mixtures 65
5.1 Introduction 65
5.2 Related Work 66
5.3 Robust Regression with DeepGUM 68
5.3.1 Unsupervised Outlier Detection through EM 68
5.3.2 Supervised Inlier Deep Regression 69
5.3.3 DeepGUM training 70
5.4 Experiments 72
5.4.1 Fashion Landmark Detection 72
5.4.2 Age Estimation 75
5.4.3 Head Pose Estimation 76
5.4.4 Facial Landmark Detection 78
5.5 Conclusions 80

6 A Comprehensive Analysis of Deep Regression 83
6.1 Introduction 83
6.2 Experimental Protocol 85
6.2.1 Base Architectures 85
6.2.2 Data Sets 86
6.2.3 Computational Environment 87
6.3 Network Optimization 87
6.3.1 Impact of the Network Optimizer 89
6.3.2 Impact of the Batch Size 89
6.4 Statistical Analysis of the Results 90
6.4.1 Wilcoxon Signed-rank Test 91
6.4.2 Confidence Intervals for the Median 91
6.5 Network Variants 92
6.5.1 Batch Normalization 94
6.5.2 Dropout ratio 95
6.5.3 Fine tuning depth 95
6.5.4 Regressed Layer 96
6.5.5 Pooling layer 96
6.5.6 Discussion on network variants 97
6.6 Data Pre-Processing 98
6.7 Positioning of the Standard Baselines 102
6.8 Overall Discussion 103
6.9 Conclusion 105

7 Conclusion 107
7.1 Summary 107
7.2 Future research directions 108

A Appendix 111
A.1 Chapter 3: Simulated environment 112
A.2 Chapter 5: Fashion Landmark Detection 115
A.3 Chapter 5: Age Estimation 115

A.4 Chapter 5: Head pose estimation 115
A.5 Chapter 5: Facial Landmark Detection 123
A.6 Teaching, Internship and Collaborations 125
A.7 Publications and Submissions 125

CHAPTER 1

INTRODUCTION

1.1 GENERAL CONTEXT

In recent years, robots have achieved perception skills such that they can start moving from research and development labs to public spaces, such as stores, hospitals, or homes. More precisely, in the case of stores, they can be used to bring an original way to welcome the clients and to help them choosing the products that fit their needs. In hospitals or homes, robots can be designed, for instance, to develop elderly care tasks, accompany people with reduced mobility, or to therapeutically assist people with social interaction difficulties (e.g. autistic children). In all these examples, the benefit experienced by the human user will depend, to a large extent, on whether the interaction developed is sufficiently natural, and socially and culturally appropriate. As a consequence, the interaction capabilities of robots must be improved, in order to simulate human behavior more accurately.

In order to interact with its environment, the robot needs first to model it. For this, the robot needs to perform elementary perception tasks such as recognizing faces, human gestures or speech, and to understand high-level concepts such as speech turns, focus of attention or interactions that can occur between the participants of a social meeting. In this PhD Thesis, we follow a top-down approach. First, we present two high-level models (chapters 2 and 3) that extract information about the behavior of groups of people from visual representations. Second, we focus on methods that extract low-level cues (chapter 4, 5 and 6). More specifically, we focus on regression problems, where the goal is to predict continuous values, such as the head or the human pose from a given image. Regression problems are defined in opposition to classification problems, where the goal is to estimate categorical values (corresponding to labels/classes).

In a social environment, visual data contain useful information to perform a rich description of the environment surrounding a robot. In contrast to the audio modality, which is available only when people are speaking, vision continuously provides information. Moreover, visual data provide accurate localization information that helps to model the

geometry of the scene. As humans communicate via their facial expressions and body gestures, robots must be able to perceive these cues to interact naturally with humans. For all these reasons, our work mainly focuses on visual data. Nevertheless, robots are regularly equipped with other type of sensors, such as microphones, self-motion sensors, or tactile sensors. Combining different cues is a challenging problem in it-self. Therefore, we also propose, in chapter 3, a model that combines visual, audio and motor information.

Interestingly, even if the initial goal is to work on robotic applications, most of the contributions presented in this manuscript are more general and could be applied in many other contexts. Except chapter 3, that is directly related to robot gaze control, the methodologies developed in the four other chapters do not contain any design choice specifically related to robotics. For instance, the framework presented in 2, could be used to model group behavior in the context of video surveillance or web video indexing. The chapters 4, 5 and 6 can be applied to any situation in which the goal is to estimate continuous values from images. Therefore, other exemplar applications could be web image indexing, human behavior modeling, computer-human interaction or medical image analysis to name a few.

1.2 Project, Scientific Context and Motivation

This Thesis was part of the Vision and Hearing in Action (VHIA) project supported by an ERC Advanced Grant obtained by Prof. Radu Horaud. The objective of VHIA is to elaborate perception models that integrate perception-action loops. In the context of this project, the Perception team proposes methods that learn mappings between auditory/visual inputs, structured outputs, and sensorimotor information. These learning models are used in order to execute perception-action cycles with a humanoid robot in a real world environment. These methods address highly difficult issues, such as how to build joint audiovisual representations from heterogeneous, noisy, ambiguous and physically different data, how to deal with high-dimensional input data, and how to robustly and efficiently perform human-humanoid communication tasks in the context of realtime applications.

With respect to the scientific context, considerable research has been devoted to the recognition from visual data of activities performed by a single person [90, 134, 175, 179] or two persons [149, 150]. However, in realistic scenarios that occur in the context of human-robot interaction, the number of persons facing the robot cannot be constrained to one or two. Less attention was paid by the computer vision community to group modeling. Consequently, this Thesis tries to fill this gap in this particular research domain.

This PhD Thesis was carried out within the Perception Team at Inria Grenoble Rhône-Alpes under the supervision of Radu Horaud who is Director of Research and the head of the team. During the first year of the PhD, I have been co-supervised by Georgios Evangelidis that left for DAQRI. Then Pablo Mesejo and Xavier Alameda-Pineda joined the team during my third year of PhD as senior Post-doc and Research Scientist , respectively. Their experience in statistics, machine learning and deep learning have been of

great importance in the progress of my research.

My PhD was carried out in a particular moment of computer vision history. Indeed, the raise of deep learning methods has undeniably impacted the community. Deep learning largely outperformed the state-of-the-art in many traditional computer vision tasks such as image classification [86, 169] or object detection [59, 155]. Roughly speaking, deep learning is a class of machine learning methods that can be characterized by the following properties:

- A stack of nonlinear functions (called layers) are employed to learn representations of data with multiple levels of abstraction. A layer is generally composed of a linear operator, as convolutions for instance, followed by a nonlinearity.
- The first layers extract low-level information whereas the last layers can capture high-level concepts.
- The training optimization problem is generally solved via some form of stochastic gradient descent combined with the back-propagation algorithm [147].

Deep learning approaches have also been used to solve regression problems. Classical applications in computer vision include human pose estimation [174], age estimation [145], head-pose estimation [38] and facial landmark detection [166].

In this context, we propose methods that are based on a sound optimization or probabilistic framework. However, beyond strong mathematical foundations, we try to perform extensive experimental evaluations of the proposed methods. This desire of strong experimental validation can be seen, in particular, in the realtime experiments performed on the NAO robot (see chapter 3) and in the large scale study of chapter 6.

1.3 DATASETS

Following the experimental approach mentioned above, we performed experiments on several publicly available datasets. Here is the list of the datasets that are used across this manuscript:

- *Collective activity dataset* [25]. This dataset is composed of 44 videos with 5 activities: *crossing, waiting, queueing, walking, talking*, and 8 poses: *right, front-right, front, front-left, left, back-left, back, back-right*.
- *New collective activity dataset* [22]. This dataset is composed of 33 videos with 6 activities: *gathering, talking, dismissal, walking together, chasing, queueing*. Even if different activities can occur sequentially in a video all persons perform the same activity, which means that there is a single activity at each frame. Consequently, this dataset is easier and less general than the *Collective activity dataset*.
- Audio-visual *AVDIAR dataset* for audio-visual diarization [56]. This dataset have been recorded by the Perception team and I personally participated to the recording

and the annotation process design. This dataset has been recorded with 4 microphones and high-resolution binocular cameras (1920×1080). Interestingly, as the employed cameras have spheric lenses with low focal lengths (6mm), the images have a particular wide field of view.

- *LFW* and *NET facial landmark detection datasets* for facial landmark detection [166]. These datasets consists of 5590 and 7876 face images, respectively. Combining these two datasets have been proposed in [166]. Each face is labeled with the positions of five key-points in Cartesian coordinates, namely left and right eye, nose, and left and right corners of the mouth.

- *Fashion landmark dataset* for fashion landmark detection [106]. It includes more than $120K$ images, where each image is labeled with landmarks. The dataset is equally divided in three subsets: upper-body clothes (6 landmarks), full-body clothes (8 landmarks) and lower-body clothes (4 landmarks).

- *Cross-age celebrity dataset (CACD)* for age estimation [21] contains 163446 images from 2000 celebrities. The images are collected from search engines using the celebrity's name and desired year (from 2004 to 2013). 1800 celebrities are used for training, 80 for validation and 120 for testing. The validation and test sets are manually cleaned whereas the training set is noisy.

- *McGill real-world face video dataset* for head pose estimation[38]. It consists of 60 videos (a single participant per video, 31 women and 29 men) recorded with the goal of studying face classification in unconstrained environments. The videos were recorded in both outdoor and indoor environments obtaining diverse illumination conditions. The participants move freely leading to important occlusions for some of the frames. Annotations for head pose are provided for the yaw angle only. The yaw angle ranges from $-90°$ to $90°$.

- *Biwi Kinect head-pose dataset* for head pose estimation [49]. It consists of over $15,000$ RGB-D images corresponding to video recordings of 20 people (16 men and 4 women, some of them recorded twice) using a Kinect camera. It is one of the most widely used dataset for head pose estimation [45, 105, 119, 178]. During the recordings, the participants freely move their head and the corresponding head orientations lie in the intervals $[-60°, 60°]$ (pitch), $[-75°, 75°]$ (yaw), and $[-20°, 20°]$ (roll). The annotations are more precised than the *McGill real-world face video dataset* but extreme poses are scarcer.

- *PARSE dataset* for full body pose estimation [138]. It is a standard dataset used for human pose estimation. It is a small dataset as it contains only 305 images. Therefore, this dataset is very challenging when using very deep architectures.

In addition to the high number of datasets, we notice the diversity in term of application, range and nature of the target values, and output space dimension.

1.4 CONTRIBUTIONS

The contributions of this Thesis are the following:

- We propose a model to recognize activities that are performed jointly by groups of people, such as *dancing* or *talking* and so forth. The model can handle videos containing one or several group activities that occur either simultaneously or in sequence. We cast the group-activity recognition problem into a structured-output labelling problem in order to model the dependecies between the activities of the people. In order to capture interactions between people, we introduce a mapping that defines unary and pairwise potentials. Therefore, we model the two-person geometry, motion and relative pose, in order to describe high-level cues such as '*persons A and B face each other*', or '*stand side-by-side*', or '*move in the same direction*'. We propose an unsupervised formulation in order to encode these interaction descriptors into pose-activity words. The model parameters are learned using the structured SVM framework. In this work, we use spatio-temporal motion features in conjunction with bounding boxes to obtain state of the art results on widely used datasets.

- In the context of human-robot interactions, the robot must first look towards people in order to interact with them. Consequently, we introduce a novel neural network-based approach for robot gaze control. The proposed model combines visual, auditory and motor information to chose the head motor commands. We cast this problem into a reinforcement learning (RL) problem and combine the Q-learning framework with recurrent neural network architectures to find an optimal action-selection policy. Hence, the robot learns to focus its attention on groups of people from its own audio-visual experiences without the use of external sensors or human supervision. Recurrent networks training is known to be particularly slow, especially in the case of RL. Consequently, we propose a synthetic environment that simulates moving and discussing participants to pretrain our model. Therefore, we avoid the need of interacting with people for hours at learning time. The model is experimentally validated via realtime experiments on the NAO robot. The experimental evaluation suggests that the proposal is robust in terms of parameters configuration. The best results are obtained when the different modalities are jointly used in order to predict the action to perform, and when a late fusion strategy is employed (i.e. when visual and audio information are separately processed and then fused). It is important to notice that this work is a common work with Benoit Massé, PhD candidate within Perception Team.

- The next contributions are related to deep regression models. The standard approach when applying deep learning to regression problems consists in using a convolutional neural network previously trained for a classification on a very large dataset. In that case, the last classification layer is replaced by a fully connected regression layer with linear or sigmoid activations. The network is then trained via Euclidean loss minimization. This type of configuration ignores the existence of other regression techniques, like inverse regression models, that are suitable in high-dimensional

to low-dimensional settings [37, 82, 99, 132] Consequently, we propose to couple a Gaussian mixture of linear inverse regressions with a neural network. We provide the methodological foundations and the associated algorithm to train the deep network and the regression function jointly. We empirically show that inverse regression outperforms L_2-based regression models currently used in head-pose estimation. More precisely, it outperforms state-of-the-art methods in head-pose estimation using a widely used head-pose dataset.

- Creating large and clean data-sets is tedious and highly time-consuming. In order to use cheap (fully automatic) or on-line (i.e. interactive) large-scale annotations, we present a model that is able to deal with noisy annotations at training time. We combine the representation power of deep neural networks with a probabilistic formulation for outlier rejection. We use a Gaussian-Uniform Mixture as the last layer of a neural network front-end. The errors for the inliers are modeled by a Gaussian distribution. Conversely, a uniform distribution is employed to model the outliers. We combine the Expectation-Maximization (EM) algorithm to unsupervisedly detect the outliers, with stochastic gradient descent to estimate the network parameters. Hence, the deep network weights are updated mainly with clean data, ignoring the contaminated training data points.

- We perform a systematic evaluation and a statistical analysis of the performance of ConvNets for regression tasks. To the best of our knowledge, this is the first large analysis of deep regression techniques. We perform experiments on three classic vision problems. Therefore, we extract practical recommendations that should be followed when tackling a new regression problem. We report confidence intervals for the median performance as well as the statistical significance of the results, if any. We observe that the variability of the performance of different network variants (e.g. the fine-tuning depth) strongly depends on the base architecture. Indeed, small differences in the architecture of the network or in the data pre-processing procedure can lead to notably different results. Surprisingly, the impact of the data-preprocessing procedures may exceed the improvement/deterioration results from changing the network architecture. Hence, we conclude that the data pre-processing should take a more relevant role in studies based on deep regression techniques.

1.5 Material Resources

Concerning the material resources, Inria and the Perception Team in particular, are well equipped. Indeed, I performed experiments on data gathered with the POPEYE robot and run interactive experiments on the NAO robot. On one side, the POPEYE robot allowed us to evaluate our gaze control model on realistic but offline videos and consequently to perform a quantitative evaluation. The processing of the recorded data has been highly facilitated by the synchronization package developed by Quentin Pelorson, engineer in Perception. On the other side, thanks to the NAO robot, we validate qualitatively the resulting behavior of the robot. These realtime experiments have been possible thanks

to the NAOLab API developed by the engineers of the Perception team, namely Soraya Arias, Fabien Badeig, Bastien Mourgue, Quentin Pelorson and Guillaume Sarrazin.

Most of the contributions presented in this manuscript were computationally demanding and consequently could not have been performed without the resources that have been at my disposal. More precisely, the experiments related to group activity recognition have been performed on the Inria CPU cluster managed by the SIC (Service Informatique du Centre) and more precisely Jean-Francois Scariot. The experiments related to Audio-Visual Gaze Control were performed on a Perception's computer with a Nvidia GTX 1070. The experiments related to deep regression models have been performed on the newly purchased GPU cluster composed of two computers with two Nvidia Titan X each. The tools provided by Inria and in particular Jean-Francois Scariot, considerably helped me to manage these GPU resources.

1.6 Manuscript Structure

This manuscript is organised as follows. In chapter 2, our framework for group activity recognition is described. In chapter 3, we present our neural-network-based model for audio-visual gaze control in human-robot interaction. In the next three chapters, we focus on regression problems. In chapter 4, we present a deep mixture model of linear inverse regressions and present its application to the head-pose estimation problem. In chapter 5, we focus on improving regression model in the case of noisy annotations. Finally, in chapter 6, we perform a comprehensive and experimental analysis for applying pretrained deep models to regression tasks.

In appendix, some technical details related to chapter 3 and additional experiments accompanying chapter 5 are reported. Then, some details are given about the courses I gave, the internships is supervised, and my collaborations during my PhD. Finally, the list of publications and submissions during my PhD is reported.

CHAPTER 2

RECOGNITION OF GROUP ACTIVITIES BASED ON SINGLE AND TWO-PERSON DESCRIPTORS

2.1 INTRODUCTION

Human activity recognition has been an active topic in computer vision over the past years. Although considerable research has been devoted to the recognition of single-person [90, 134, 175, 179] and two-person [149, 150] activities, less attention has been paid to group activities. The latter is however important for a number of applications, e.g. video surveillance, video monitoring, human-robot interaction and video indexing, etc.

In this chapter we aim to recognize activities that are performed jointly by groups of people, such as *dancing*, *talking* and so forth. More precisely, we propose a model that can handle videos that contain one or even several **distinct** group activities that occur either simultaneously or in sequence. Such a problem cannot be solved by simply considering multiple instances of single-person activities as long as the collective context carries out important information [25]. As recently shown, the inter-person distance, the motion, and the relative pose of group members seem to be important cues for good recognition performance [23, 89, 165]. However, the way these cues should be combined together is crucial for robust recognition.

In this chapter we cast the group-activity recognition problem into a structured-output labelling problem that is solved within an optimization framework. A two-stage mapping is used to obtain low-dimensional features and, in turn, to define the unary/pairwise potentials of the energy terms, e.g. Fig. 3.1. First, spatiotermporal motion features, e.g. [179] are used in conjunction with bounding boxes which are provided in advance (person detection is not addressed in this thesis. Second, we carefully investigate the two-person geometry, motion and relative pose, in order to describe high-level cues such as *persons A and B face each other, or stand side-by-side, or move in the same direction'*. These descriptions are further encoded into pose-activity words in an unsupervised manner. Third,

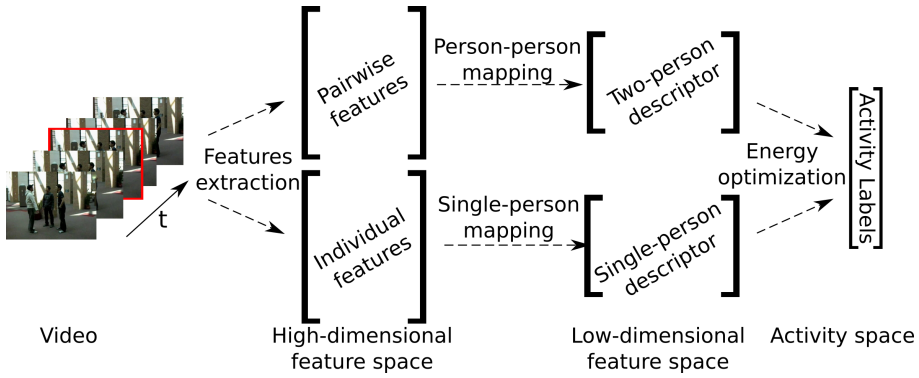


Figure 2.1: High-dimensional feature vectors are mapped onto a low-dimensional space that encodes both single-person and two-person descriptors. Based on these descriptors, a group-activity label is assigned to each person by solving an energy optimization problem.

such a multi-stage mapping provides low-dimensional yet meaningful interaction descriptors that are eventually used (along with single-person descriptors) to define unary and pairwise potentials of an energy function. The parameters of the latter are learned using structured SVM. It is worth noticing that the estimation of person poses needed to compute two-person descriptors is only performed for model training and not for testing (recognition).

The remainder of the chapter is organized as follows. Sec. 5.2 discusses the related work. In Sec. 5.3, the energy function is presented. Sec. 2.4 details the pairwise interaction description, while model learning is explained in Sec. 2.5. Sec. 2.6 discusses the features that encode the raw data and Sec. 2.7 presents the experimental results. Finally, Sec. 6.9 concludes this work.

2.2 Related Work

Early human action recognition methods could only handle simple activities performed by a single person in in controlled environments, e.g. [154]. More complex scenarios have been subsequently addressed, such as presence of occlusions, changing illumination conditions, moving cameras, dynamic background, e.g. [110]. We note that challenging single-person activities can be dealt with using sophisticated feature representations, e.g. [179]. At the same time, significant effort has been put toward using contextual information to improve activity and object recognition [120, 137]. Such strategies benefit from using the global image content, thus not suffering from low-quality appearance, small objects, or occlusions. The object-action context is addressed in [64, 79, 84, 110, 189] while spatial coherence constraints may be enforced as well [64].

When it comes to group activity, the recognition performance benefits from a cross-person context [2, 88, 129, 151, 165]. As a first attempt, [25] showed that the collective behavior, via modeling relative poses of persons, improves the classification of group

activities. While this framework can deal with multiple group activities that are performed simultaneously, the classification is still done individually, i.e. for each detected person. Subsequent work focused on the simpler scenario of a single group activity per frame. Any person interaction description can then be simply integrated in the model, e.g. [65, 88, 165] and not only in feature encoding. This led to the introduction of pairwise potentials within energy-based formulations, so that persons involved in the same activity are jointly considered. Group structure analysis and selection of meaningful pairs of persons [89, 165] seem to further help at the expense of higher computational complexity. However, these methods relies on tracking each person, that can be inaccurate in crowded scenarios. Note that [23] models groups of humans in still images. More recently, deep structured models have been used for single group activity [39, 75]. While these methods reach satisfying results, their generalization to the multiple activity case is not straightforward. Actually, when several activities can occur simultaneously, the size of the label space grows exponentially with the number of persons. Thus, complex or restrictive formulations cannot be used in practice. Temporal information has been also used in [22, 31, 81] to deal with issues that arise from the social context, e.g. occlusions, gathering etc. In a similar manner, [100] considers temporal interactions between persons to analyse team sports. In some sense, however, these two strategies contradict each other since the finer the group modelization, the more difficult its integration into a temporal framework. Conversely, adding the time dimension into graph formulations leads to computationally demanding solutions, in particular when several groups of different activities appear at the same time. As a consequence, one may need to properly combine these two approaches.

Instead, we propose to investigate the two-person (or person-person) geometry, the motion and the relative poses in order to describe precisely the interactions. The proposed model exploits temporal information only when building descriptors, thus avoiding inference on temporal models. Moreover, the proposed descriptors can be used efficiently and without computationally demanding optimization algorithms in difficult scenarios in which several group activities occurs simultaneously. In addition, and unlike [23, 24, 25], body poses are used for training purposes only, so that the body pose is not required at test time to estimate our pairwise descriptor.

2.3 Modeling Group Activity

Provided that $I < I_{max}$ persons are detected in an image, we extract high-dimensional single-person feature vectors, $\boldsymbol{x}_i \in \mathbb{R}^D$, from each bounding box and high-dimensional pairwise feature vectors, $\boldsymbol{y}_{ij} \in \mathbb{R}^F$, from each pair of boxes, so that the sets $\boldsymbol{X} = \{\boldsymbol{x}_i \mid 1 \leqslant i \leqslant I\}$ and $\boldsymbol{Y} = \{\boldsymbol{y}_{ij} \mid 1 \leqslant i, j \leqslant I, i \neq j\}$ are available. The goal is to find a set of activity labels $\boldsymbol{A} = \{a_i \mid 1 \leqslant i \leqslant I\}$ (one label per detected person) where $a_i \in \mathcal{L} = \{l_m \mid 1 \leqslant m \leqslant M\}$, that is an M-activity label set. To this end, we define an energy function $E(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{A})$ and we seek the optimizer $\boldsymbol{A}^*$ that best fits with the features:

$$\boldsymbol{A}^* = \underset{\boldsymbol{A} \in \mathcal{A}_I}{\operatorname{argmax}}\, E(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{A}), \tag{2.1}$$

where $\mathcal{A}_I$ is the activity label set. The energy function is defined as:

$$E(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{A}) = \sum_{i=1}^{I} \underbrace{\Psi_1(\boldsymbol{x}_i, a_i)}_{\textit{individual potential}} + \sum_{i \neq j}^{I \times I} \underbrace{\Psi_2(\boldsymbol{y}_{ij}, a_i, a_j)}_{\textit{pairwise potential}} + \underbrace{\Psi_3(\boldsymbol{A})}_{\textit{regularizer}}. \tag{2.2}$$

The individual potential $\Psi_1(\boldsymbol{x}_i, a_i)$ models the compatibility of label a_i with $\boldsymbol{x}_i$; the pairwise potential $\Psi_2(\boldsymbol{y}_{ij}, a_i, a_j)$ models the compatibility of a pair of labels (a_i, a_j) with $\boldsymbol{y}_{ij}$; the term $\Psi_3(\boldsymbol{A})$ enforces grouping, i.e., by assigning the same activity label to multiple persons. A log-linear model leads to the following energy function:

$$E(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{A}) = \sum_{i=1}^{I} \boldsymbol{w}_{a_i}^{1\top} \boldsymbol{\phi}_1(\boldsymbol{x}_i) + \sum_{i \neq j}^{I \times I} \boldsymbol{w}_{a_i, a_j}^{2\top} \boldsymbol{\phi}_2(\boldsymbol{y}_{ij}) \mathbb{1}(a_i = a_j) + \boldsymbol{w}^{3\top} \boldsymbol{\phi}_3(\boldsymbol{A}). \tag{2.3}$$

The indicator function in the pairwise term makes the sum valid for the pairs of persons that perform the same activity. Grouping is then implicitly considered without adding group variables. Note that the parameters $\boldsymbol{w}_{a_i}^1$,$\boldsymbol{w}_{a_i,a_j}^2$ and $\boldsymbol{w}^3$ are learned on a training set (Sec. 2.5). As for the feature functions, they are defined as follows.

Single-person mapping $\boldsymbol{\phi}_1$: Let $\boldsymbol{B} = \{b_i \mid 1 \leqslant i \leqslant I\}$ be the discrete body poses of the detected persons where $b_i \in \Pi = \{\pi_q \mid 1 \leqslant q \leqslant Q\} \subset [-\pi, \pi]$, i.e., Π denotes a set of possible discrete body poses. The poses are modeled by an angle between the body orientation and a reference orientation. As in [88], $\boldsymbol{\phi}_1$ maps the high-dimensional feature space onto a low-dimensional space of posterior probabilities: $\boldsymbol{\phi}_1(\boldsymbol{x}_i) = \Big(P(b_i = \pi_q, a_i = l_m | \boldsymbol{x}_i)\Big)_{(\pi_q, l_m) \in \Pi \times \mathcal{L}}$ The mapping $\boldsymbol{\phi}_1$ is estimated via a linear SVM on training data in the second stage of the pipeline.

Two-person mapping $\boldsymbol{\phi}_2$: This mapping projects a pairwise high-dimensional feature vector onto a low-dimensional space that describes the interaction between two detected persons, e.g., facing each other. The explicit design and estimation of ϕ_2 is one of the main contributions and it is explained in detail in Sec. 2.4.

Regularization term $\boldsymbol{\phi}_3$: The regularizer is simply defined as $\boldsymbol{\phi}_3(\boldsymbol{A}) = \Big(\mathbb{1}(\nu(\boldsymbol{A}) = i)\Big)_{1 \leqslant i \leqslant I_{max}}$, where $\nu(\boldsymbol{A})$ is the number of different activities in $\boldsymbol{A}$. For instance, if $\boldsymbol{A} = \{l_1, l_3, l_3, l_5\}$, then $\nu(\boldsymbol{A}) = 3$ and $\boldsymbol{\phi}_3(\boldsymbol{A}) = (0, 0, 1, 0 \cdots, 0)$.

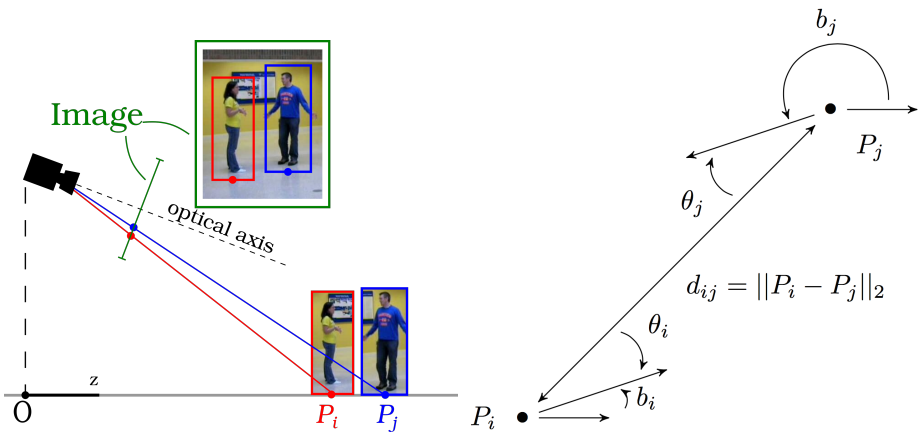


Figure 2.2: We estimate the projection-free 3D positions $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$ considering that the persons are standing on the ground (left image). We then extract the angles θ_i and θ_j as well as the distance d_{ij}. The right image shows a top-view of the geometry between two persons and their associated variables.

2.4 Two-Person Mapping

As with the single-person case, we want to map the high-dimensional pairwise feature space into a low-dimensional space of posterior probabilities, that is,

$$\phi_2(\boldsymbol{y}_{ij}) = \Big(P(\xi_{ij} = k|\boldsymbol{y}_{ij})\Big)_{k\in\mathcal{K}} \tag{2.4}$$

where $\mathcal{K}$ is a *pose-activity* dictionary and $\xi_{ij} \in \mathcal{K}$ defines the *pose-activity* word associated to feature $\boldsymbol{y}_{ij}$. However, the construction of such a dictionary is not straightforward as long as we need to learn *relevant* pose-activity words. Therefore, we learn this dictionary in an unsupervised manner using a spectral clustering method.

Suppose two persons are observed in a frame. Using the method of [70], we extract their 3D positions $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$ as illustrated in Fig. 2.2. We define an interaction descriptor $\boldsymbol{p}_{ij}$ that encapsulates their relative distance and pose, as well as the associated activities: $\boldsymbol{p}_{ij} = (d_{ij}, \theta_i, \theta_j, a_i, a_j)$, where d_{ij} is the relative distance $d_{ij} = \|\boldsymbol{P}_i - \boldsymbol{P}_j\|_2$ and θ_i, θ_j are the angles spanned between the line $\boldsymbol{P}_i\boldsymbol{P}_j$ and the respective pose vectors. Fig. 2.2 (right) illustrates the geometry of the model. To learn the pose-activity words, we use a spectral clustering method [192] with a similarity function $\delta(\boldsymbol{p}_{ij}, \boldsymbol{p}_{kl}) = e^{-f(\boldsymbol{p}_{ij},\boldsymbol{p}_{kl})^2}$, where f is defined by:

$$\begin{aligned} f\left(\boldsymbol{p}_{ij}, \boldsymbol{p}_{kl}\right) = \min\{&2 - (\mathbb{1}(a_i = a_k) + \mathbb{1}(a_j = a_l)) \\ &+ \lambda_1(\mu(\theta_i, \theta_k) + \mu(\theta_j, \theta_l)), \\ &2 - (\mathbb{1}(a_i = a_l) + \mathbb{1}(a_j = a_k)) \\ &+ \lambda_1(\mu(\theta_i, \theta_l) + \mu(\theta_j, \theta_k))\} \\ &+ \lambda_2|d_{ij} - d_{kl}| \end{aligned} \tag{2.5}$$

$$\text{with } \mu(\theta, \theta') = \min_{k\in\{1,0,-1\}} |\theta - (\theta' + 2k\pi)|. \tag{2.6}$$

The parameters in (2.6), λ_1 and λ_2 are weights and the $\min$ operator is used to make the function f invariant to 2π shifting and symmetric, that is, $f\left(\boldsymbol{p}_{ij}, \boldsymbol{p}_{kl}\right)$ is equal to zero if $\boldsymbol{p}_{ij} = \boldsymbol{p}_{kl}$ or if $\boldsymbol{p}_{ij} = \boldsymbol{p}_{lk}$. The value of $f\left(\boldsymbol{p}_{ij}, \boldsymbol{p}_{kl}\right)$ increases with the distance and the pose/action difference. As a consequence, the spectral clustering method tends to group together pairs of persons with the same activity, same distances and same relative poses. Considering a training set of interaction descriptors $\tilde{\boldsymbol{p}}_{ij}$, the spectral clustering algorithm provides a cluster assignment label $\tilde{\xi}_{ij} \in \mathcal{K}$ for each descriptor. This label is used as a pairwise pose-activity word of the respective dictionary. As with single-person mapping, ϕ_2 is learned via a linear SVM. At test time, the assignment labels ξ_{ij} are not computed. Instead, ϕ_2 maps directly the high-dimensional representation onto the low-dimensional space where energy optimization is performed.

We illustrate our clustering approach on an example in order to clarify its contribution. For visualization simplicity we consider a dataset in which there is only one activity per image. As a consequence, $a_i = a_j$ holds for all interaction descriptors $\boldsymbol{p}_{ij}$. After extracting each interaction descriptor, we apply the above described clustering method. Fig. 2.3 shows which cluster each interaction descriptor is assigned to. The interaction descriptors are displayed on one of the two plots according to the a_i value. In the case of *talking* activity, most of points are gathered in a cyan cluster where $(\theta_i, \theta_j) \approx (0, 0)$. This confirms the intuitive idea that the persons are likely to face each other when they are talking. On the contrary, the area $(\theta_i, \theta_j) \approx (0, 0)$ is almost empty in the case of the *queueing* activity since the persons are supposed to be in single-file when they are in a queue. Moreover, the only parameter that distinguishes the two clusters of the *queueing* activity is the distance d_{ij}.

2.5 Model Learning

We use structured support vector machine (SSVM) [121] to estimate the vectors $\boldsymbol{w}^1_{a_i}$,$\boldsymbol{w}^2_{a_i,a_j}$ and $\boldsymbol{w}^3$. To this end, Equation (2.3) is rewritten as

$$\begin{aligned} E(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{A}) &= \sum_{m}^{M} \boldsymbol{w}^{1\top}_{l_m} \Big[\sum_{i=1}^{I} \boldsymbol{\phi}_1(\boldsymbol{x}_i) \mathbb{1}(l_m = a_i) \Big] \\ &+ \sum_{m}^{M} \boldsymbol{w}^{2\top}_{l_m} \Big[\sum_{\substack{i,j \\ i \neq j}}^{I \times I} \boldsymbol{\phi}_2(\boldsymbol{y}_{ij}) \mathbb{1}(a_i = a_j) \mathbb{1}(l_m = a_i) \Big] \\ &+ \boldsymbol{w}^{3\top} \boldsymbol{\phi}_3(\boldsymbol{A}) \\ &= \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{A}), \end{aligned} \tag{2.7}$$

Considering a set of N training images, From each of the $\tilde{I}^{(n)}$ bounding boxes, we extract the individual feature vectors $\tilde{\boldsymbol{X}}^{(n)}$ and the pairwise feature vectors $\tilde{\boldsymbol{Y}}^{(n)}$ to build our

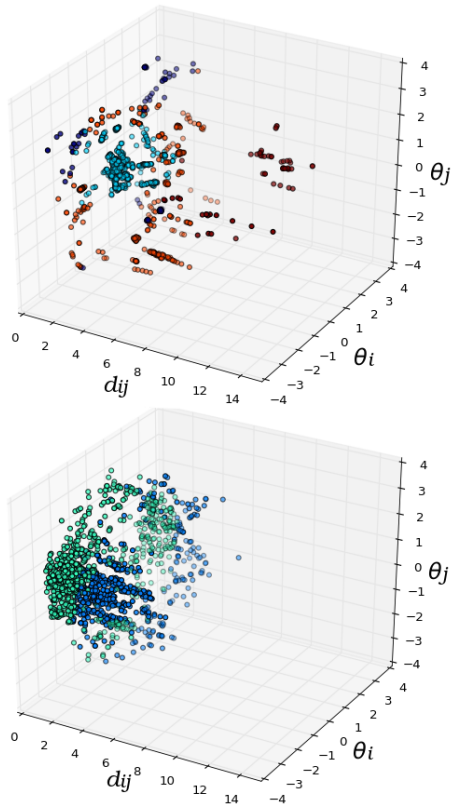


Figure 2.3: Clustering results obtained on dataset *A'* (see Sec.4.5 for more details about the datasets) for two activities: *talking, queueing*. Each color represents one cluster and each point is an interaction descriptor. θ_i and θ_j are measured in radians and d_{ij} in meters.

training set $\{\tilde{\boldsymbol{X}}^{(n)}, \tilde{\boldsymbol{Y}}^{(n)}, \tilde{\boldsymbol{A}}^{(n)}\}_{1 \leqslant n \leqslant N}$. We estimate $\boldsymbol{w}^*$ such that:

$$\begin{aligned} \boldsymbol{w}^* = & \underset{\boldsymbol{w}}{\operatorname{argmin}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 \\ & + C \sum_{n=1}^{N} \max_{A \in \mathcal{A}_{I(n)}} \big(\boldsymbol{w}^\top \boldsymbol{\phi}(\tilde{\boldsymbol{X}}^{(n)}, \tilde{\boldsymbol{Y}}^{(n)}, \boldsymbol{A}) \\ & - \boldsymbol{w}^\top \boldsymbol{\phi}(\tilde{\boldsymbol{X}}^{(n)}, \tilde{\boldsymbol{Y}}^{(n)}, \tilde{\boldsymbol{A}}^{(n)}) + \Delta(\tilde{\boldsymbol{A}}^{(n)}, \boldsymbol{A}) \big) \end{aligned} \tag{2.8}$$

Commonly, C is a hyper-parameter to adjust the SVM fitting, while $\Delta(\tilde{\boldsymbol{A}}^{(n)}, \boldsymbol{A})$ is the loss function that should penalize different labelings. Here, the following loss function is used: $\Delta(\tilde{\boldsymbol{A}}^{(n)}, \boldsymbol{A}) = \sum_{i=1}^{\tilde{I}^{(n)}} \mathbb{1}(\tilde{a}_i^{(n)} \neq a_i)$. This function counts the number of incorrect labels. It is deliberately not normalized to account the fact that the model can learn more from images with a large number of persons.

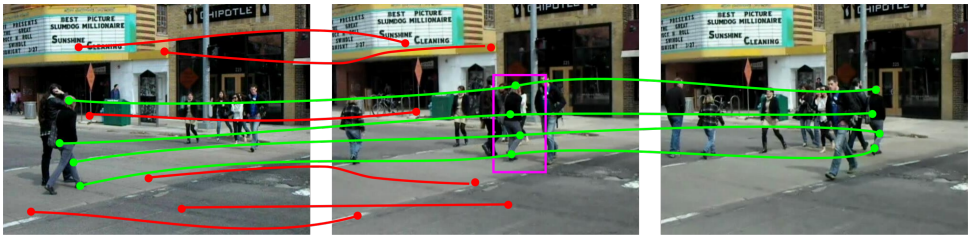


Figure 2.4: Trajectory-based description of (pink) bounding-box at frame t: Points of frame $t - \Delta t$ are densely sampled and tracked up to frame t. At time t, (red) trajectories out of the bounding box are rejected. The remaining end points of trajectories (green) are further tracked until $t + \Delta t$. The local space-time area around the green trajectories is then encoded into the individual features x_i.

2.6 Feature Extraction

In this section we describe how we extract the individual (single-person) features $\boldsymbol{x}_i$ and the pairwise features $\boldsymbol{y}_{ij}$, that is the first-level features of our pipeline (see Fig. 3.1). The individual features $\boldsymbol{x}_i$ are built on two state-of-the-art feature extraction methods. The first one is based on point trajectories [179] and catches the local body motion whereas the second one is based on histogram of oriented gradients (HOG) [32] and catches the posture of a person. While local spatial gradients are described by the trajectories, HOG descriptor carries complementary information that is missing from trajectory features and is mainly useful for static activities. For each bonding box, we extract a trajectory-based feature vector $\boldsymbol{x}_i^{traj}$ and a HOG feature vector $\boldsymbol{x}_i^{hog}$. Two single-person mappings ϕ_1^{traj} and ϕ_1^{HOG} are learned. We then concatenate the two outputs (the low-dimensional vectors) to obtain $\phi_1(\boldsymbol{x}_i)$. Next, we explain how $\boldsymbol{x}_i^{traj}$ is built.

In order to use temporal information, we use the information from neighboring frames to build our box-wise descriptor. Thus, we propose to adapt the improved trajectory features of [179]. Improved trajectories consist of tracking over time densely sampled image points at different scales, while the local volume around the trajectory is encoded via several descriptors: HOG, histogram of optical flow, motion boundary histograms and trajectory shape, thus leading in a vector of dimension D. However, to build our single-person feature vector $\boldsymbol{x}_i^{traj}$, we proceed as follows. To describe a bounding-box of a person at frame t, we consider the interval $[t - \Delta t, t + \Delta t]$. First, sampled points at frame $t - \Delta t$ are tracked until frame t, and only trajectories that cross the bounding box of a person at frame t are kept. The remaining points are tracked until the frame $t + \Delta t$. Fig. 2.4 illustrates trajectories that are kept at frame t while their end points are further tracked. Recall that this allows us to use temporal information without building a complete temporal model. In [179], trajectories with low variance are rejected based on the assumption that they most likely belong to the background. Here, we keep all the trajectories since our region of interest (bounding box) does not contain lots of background information. Moreover, lack of motion seems to be informative for some activities like *waiting* or *queuing*. The dimension of the feature vectors is halved ($D/2$) via PCA. We

then encode the individual features of each person into Fisher vectors of size κD using κ Gaussian components [152].

We also use trajectory-based features to obtain the pairwise (two persons) feature vectors $\boldsymbol{y}_{ij}$. Considering two persons (and their bounding boxes) detected in frame $t-\Delta t$, we densely sample points and track them until frame t as with the individual features. Here, only trajectories that cross one of the bounding boxes at frame t are kept. As before, the remaining points are tracked until frame $t + \Delta t$. We encode the descriptors into Fisher vectors using κ Gaussian components to obtain the pairwise vector $\boldsymbol{y}_{ij}$.

2.7 EXPERIMENTS

We first describe the datasets used to evaluate our model and give the implementation details. Choi *et al.* first published a collective activity dataset in [25]. This dataset is composed of 44 videos with 5 activities: *crossing, waiting, queueing, walking, talking*, and 8 poses: *right, front-right, front, front-left, left, back-left, back, back-right*. We refer here to this dataset as *dataset A*. This dataset is used to evaluate our model for the problem of recognizing multiple-group (distinct) activities. In order to compare with methods that can only deal with the single-group activity problem, two other datasets are used. Choi and Savarese published a new collective activity dataset [22], which we refer to as *dataset B*. This dataset consists in 33 videos with 6 activities: *gathering, talking, dismissal, walking together, chasing, queueing*. It is important to note that even if different activities can occur sequentially in a video of dataset *B*, all persons perform the same activity, which means that there is a single activity at each frame. Instead, dataset *A* may contain one or several distinct activities that are performed in parallel by different groups. At the same time, Choi and Savarese [22] published new annotations for dataset *A* in which they only consider one dominant activity per frame.In other words, if two groups of people perform two different activities, all the persons are considered as belonging to the activity of the largest group. We refer to this dataset as dataset-*A'*. As with dataset -emphA, eight pose labels are considered. Note that in both datasets *A'* and *B*, annotations are provided for one out of three frames, whereas one out of ten frames are annotated in dataset *A*.

As in [22], we split each dataset in training and test sets. One of the main difficulties with these datasets is that the classes are unbalanced. To figure out this issue, we use synthetic data augmentation as proposed in [20] and we produce new features that belong to the convex hull of the original features for minority classes, before training with SVM. While data augmentation does not directly apply to structured data, we augment the training examples of SSVM by perturbing the data of minor classes with Gaussian noise.

For the trajectory features, we use the values $D = 426$ and $\Delta t = 7$ in order to track points over 15 frames, as recommended in [179]. Then, Fisher vectors are built based on mixture of $\kappa = 128$ Gaussian components, while its dimensionality is reduced to 1500 using PCA, trained on a balanced set of features. As for pairwise features, we use weights $\lambda_1 = 0.33$ and $\lambda_2 = 0.25$. The weights have been chosen such that each

term of the function f in (2.5) is equally important. The camera parameters used for the projection-free map estimation have been chosen to give results which look realistic but we cannot evaluate the predicted positions quantitatively because ground-truth values are not provided. However the satisfying clustering results, e.g. Fig. 2.3, confirm the quality of the 3D position estimations. We chose $K = 30$ for dataset *A* and $K = 12$ clusters for datasets *A'* and *B*.

2.7.1 Evaluation of the Proposed Model

In this section, we show results from a series of experiments in order to make clear the contribution of each component of the proposed methodological pipeline. We compare six variants of the proposed framework:

- *Single person*: Here, only the individual terms are used, namely the first double sum of (2.2). In the case of dataset *A* this is equivalent to a single SVM estimation per person. For dataset *B*, since we use the prior that all the persons are perform the same activity, our method yields better performance than single SVM estimations.
- *Static Pairwise*: Here we use simpler pairwise features on the following grounds. Instead of using the trajectories, we use HOG features and a linear SVM to estimate person poses, then we deduce an interaction descriptor $\boldsymbol{p}_{ij}$. We then estimate $(P(\xi_{ij} = k|\boldsymbol{p}_{ij}))_{k\in\mathcal{K}}$ with a second SVM. The posterior probabilities of each pose-activity word are then used as a two-person mapping ϕ_2. This way, only one frame is used to extract the pairwise features.
- *Without regularization*: We use the model as described above but the regularization term in (2.2) is omitted. Note that only the performance with dataset *A* is affected by this modification. As long as only one group activity occurs at a time in datasets *A'* and *B*, the regularizer has no impact.
- *Full model*: We make use of the full model, as proposed above.
- *4 meter threshold*: Since [88] observe some benefits from pruning, we test the full model along with the constraint that only person pairs whose distance is less than 4 meters are summed up in (2.3). We consider this threshold as a reasonable assumption for the used activities, i.e., people that are at least4 meters away from each other are not taken into account.
- *Temporal smoothing*: A median filter is applied on the *full model* output to smooth predictions. Only the past predictions are used within a 12-frame window. As we choose $\Delta t = 7$ to extract the trajectories, a 26-frame window is used in total for each prediction. We test however the contribution of temporal smoothing only for datasets *A'* and *B*, since different activities occur simultaneously in dataset *A* and this would require to track all the persons.

The performance of each one of these variants is evaluated on the three datasets. We summarize the results in Table 2.1. We can see that pairwise features drastically help the

	dataset *A*	dataset *A'*	dataset *B*
Single person	56.3%	59.6%	74.6%
Static pairwise	66.6%	73.8%	75.4%
Without regularization	73.3%	-	-
Full model	74.5%	79.8%	78.8%
4 meter threshold	**74.8%**	78.9%	78.4%
Temporal smoothing	-	**81.5%**	**80.2%**

Table 2.1: Average per-class activity recognition accuracy on three datasets for the single and multiple group activity problems

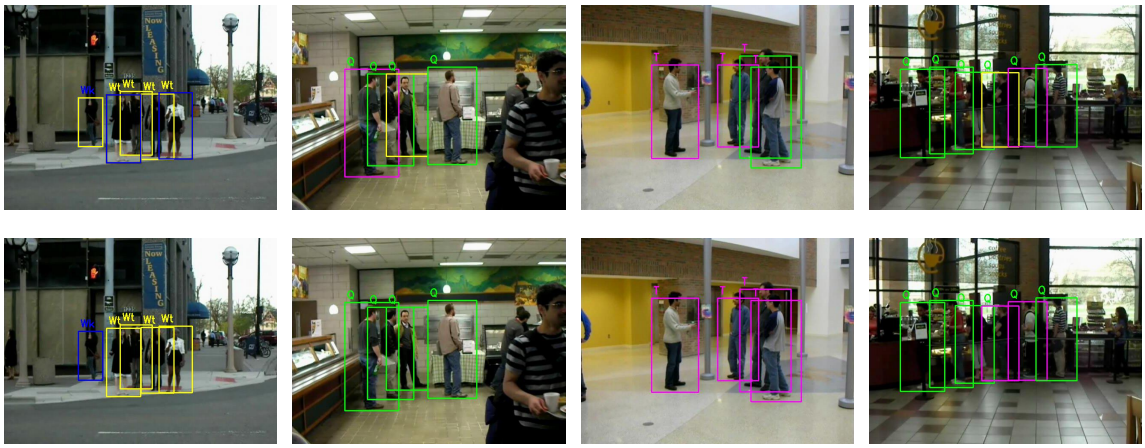


Figure 2.5: Example frames from sequences of dataset *A* with single person model (first row) and the proposed model (second row) . The color of the box represents the predicted activity (red for *Crossing*,yellow for *Waiting*, green for *Queueing*, blue for *Walking*, purple for *Talking*). On top of each bounding box, the letters show the ground truth activity (*better viewed on screen*)

recognition in datasets *A* and *A'*. Fig. 2.5 illustrates this gain with a few examples. The gain is smaller for dataset *B*. This can be explained by the fact that dataset *B* constitutes an easier case compared to datasets *A* and *A'*: stable camera, simpler action scenarios, fewer persons, and less overlap. It validates the principle that taking pairwise interactions into consideration helps in complex and natural environments. Unlike [88, 165], the pruning strategy is not very helpful in our case, i.e., we do not consistently notice improvement when using a distance threshold. This may mean that other methods rely more on the used features compared to the proposed one. Our pairwise features succeed in capturing the information of the distance between persons, and SSVM has learned that pairs with high distances have to be ignored. A temporal yet non-sophisticated smoother helps on dataset *A'* and *B* and even increases the performance of the proposed model.

2.7.2 Comparison with State of the Art Methods

Designing a benchmark for group activity recognition is difficult for the following reasons. Some authors use additional information/annotation for training or for testing, or they use different evaluation measures. For example, group labels are required to evaluate the method proposed in [165]. Table 2.2 and Table 2.3 show the performance of several methods for the single and multiple group activity problems. We use the per-class average to compute the performance considering the unbalanced number of available classes for each example. In the case of multiple activity recognition (Table 2.2) our method is the second-best performing method. Note however that additional group annotations are used by [165]. In the case of the recognition of single group activities (Table 2.3), our method also yields the second-best results. It should be noted that the best performing method, i.e. [3], uses its own person detections which biases the comparison with the other methods.

Choi *et al.* [25]	65.0%
Lan *et al.* [87]	68.2%
Sun *et al.* [165]	**77.1%**
Proposed	**74.8%**

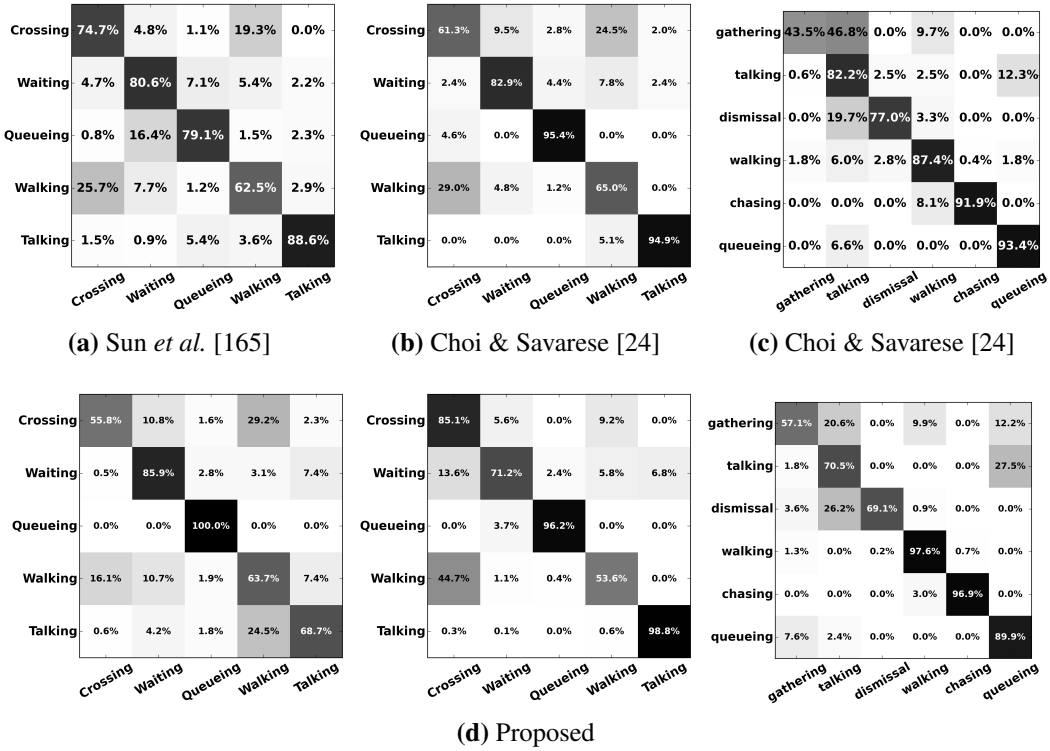
Table 2.2: Classification accuracies based on *per-class averages* for multiple group-activity recognition on dataset *A*. Note that [165] uses additional annotations.

	dataset *A'*	dataset *B*	
Khamis *et al.* [81]	72.0%	-	
Lan *et al.* [89]	78.4%	-	#
Deng *et al.* [39]	80.6%	-	*
Ibrahim *et al.* [75]	80.9%	-	
Sun *et al.* [165]	81.2%	-	*
Hajimirsadeghi *et al.* [65]	81.9%	-	
Amer *et al.* [3]	**92.2%**	**87.2%**	*#
Choi & Savarese [24]	79.9%	79.2%	†
Nabi *et al.* [123]	-	72.4%	
Proposed	**81.5%**	**80.2%**	

*Only the average accuracies are provided.
#Uses the authors' person detector.
†Uses extra annotations.

Table 2.3: Classification accuracies based on *per-class averages* for single group-activity recognition.

Fig. 5.5 plots the confusion matrices obtained with three methods and with our method. Note that it is not possible to show the confusion matrix obtained with [3] because only average results are provided in this chapter. On dataset *A*, we compare our method with [165] which uses extra annotations. The very good recognition scores obtained with the proposed method for the *queueing* activity shows the added value brought in by the use of pairwise features. As expected, both methods have difficulties in discriminating between *crossing* and *walking*. It seems difficult to reduce the confusion between these



(a) Sun *et al.* [165] **(b)** Choi & Savarese [24] **(c)** Choi & Savarese [24]

(d) Proposed

Figure 2.6: Confusion matrices for dataset *A* (left column), dataset *A*' (middle column) and dataset *B* (right column).

two categories in the absence of context information. On dataset *A'*, the proposed method discriminates between *crossing* and *walking* slightly better then [24] which, again, uses extra annotations, namely interaction labels: *facing-each-other*, *standing-side-by-side*, and *standing-still*. In contrast, our method relies on two-person descriptors with no extra annotations.

On dataset *B*, the proposed method recognizes well activities for which motion plays an important role, such as *chasing* and *walking*. It performs less well whenever there are large motion variations during the activity, e.g. *gathering*. Nevertheless, our method yields a sensible result for this activity which is split into two parts: when the participants are spread out, the activity is labeled as *walking*, and once they get close to each other, it is labeled as *talking*. We believe that these are valid recognition results, although the activity is annotated as *gathering*. Nevertheless, activities with large dynamic variations might be dealt with by using a more sophisticated temporal model.

2.8 Conclusions

In this chapter, we proposed to model the group activity recognition problem as a structured labeling problem which is solved via an energy optimization framework. The pro-

posed model can handle videos with either single or multiple group activities that occur simultaneously. Once high-dimensional feature vectors are calculated for each bounding-box and each for each pair of bounding boxes, we learn mapping functions in order to get meaningful representations in low-dimensional spaces. The geometry of the relationship between pairs of persons is included in the mapping function by learning a dictionary of relevant interactions. The geometry of the activity is thus taken into account without relying on pose estimations. We propose to use trajectory features to encode individual person description as well as the interactions between pairs of persons. The proposed framework outperforms several state-of-the-art methods for the recognition of multiple group-activities and compares favorably with other methods for the recognition of a single group activity. Future work includes the further use of group features as well as adding a temporal model to the energy formulation.

CHAPTER 3

DEEP REINFORCEMENT LEARNING FOR AUDIO-VISUAL GAZE CONTROL IN HUMAN-ROBOT INTERACTION

In the previous chapter, we presented a model able to recognize activities performed by groups of people, and we outperformed the state of the art employing widely used datasets. However, such datasets were designed for video surveillance purposes and, consequently, the models learned are not directly applicable to human-robot interaction problems. Indeed, in video surveillance, the camera is generally static and distant from the people, whereas a social robot is generally close to the people with whom it interacts and can change its head orientation accordingly. In addition, those datasets are limited to videos were people are in the field of view and perform a reduced and predetermined set of activities. Along with this limitation, in the context of human-robot interaction, a task of vital importance, and prior to recognizing activities, is to observe the people performing those activities. For this reason, this chapter tackles the problem of robot gaze control.

As previously introduced, robots are equipped with different types of sensors, like microphones, video-cameras or self-motion sensors. The combination of all this heterogeneous information, which improves the performance of methods based on a single modality, remains a challenging operation. If we think about the robot gaze control problem combining auditory and visual information, we must bear in mind that those are complementary signals. On the one hand, vision provides an accurate localization estimation but is available only when people are within the field of view. On the other hand, audio provides a less accurate localization estimation but is also available when people are out of the field of view. In addition, the learning of a perceivable and socially acceptable gaze behavior strategy can hardly be handcrafted a priori. As a consequence, we propose, in this chapter, to combine visual, auditory and motor information to perform robot gaze control by means of a novel neural network-based reinforcement learning approach. We enable a robot to autonomously learn and adapt its gaze control strategy for human-robot interaction without using external sensors or human supervision. The robot learns to focus its

attention on groups of people from its own audio-visual experiences, and independently of the number of people in the environment, their position and appearance.

3.1 INTRODUCTION

In recent years there has been a growing interest in the development of robotic systems able to communicate with people, i.e. human-robot interaction (HRI). Unlike traditional robot perception systems that have primarily been used for robot localization and navigation, HRI implies that there are people in the loop, therefore the robot must take decisions in order to optimally interact with users. For example, a robot can recognize an user's gestures, intentions, or speech only if the robot faces that user, i.e. dyadic interaction. Moreover, robots are likely to be present in populated spaces, such as hospitals, museums, hotel lobbies, etc. Consequently, a robot should be able to interact with a group of people or be part of a team. In situations such as the ones cited, a robot teammate must constantly maintain the participants in its visual and acoustic fields of view such that it can easily receive instructions while respecting social etiquette.

In this chapter, we address the problem of audio-visual gaze control, or more precisely, how a robot should combine controlled motions with acoustic and visual observations in order to direct its head towards groups of people. *Active perception* is necessary for making inferences from observations; it is equally needed for deciding to look at something or to speak with someone. The objective is to design a methodology that enables robots to learn gazing strategies from data; for example, to maximize the number of persons that are present in its visual field of view and, possibly, to favor people engaged in spoken communication.

Also, it is interesting to note that gaze control has been mainly addressed for dyadic interaction. In multi-party scenarios, focusing on only one person may lead to miss important information such as who looks at whom and who is the speaker and who are the listeners [111]. Hence there is a danger that the controller makes suboptimal decisions with respect to the task at hand. We address gaze control in the specific case of multi-party interaction.

Gaze control was already addressed within the framework of sensor-based robot servoing. For example, visual servoing consists of designing a control loop that aligns the observed position of an object with a targeted position [29]. This implies that the direct and inverse robot Jacobians are known. Alternatively, these Jacobians may be estimated via reinforcement learning [55]. Recently, the concept of sensor-based servoing was applied to the audio modality by directly linking observed acoustic features to robot control. However this approach makes the strong assumption that there is a single sound source that emits continuously. e.g. [17, 108]. Unfortunately this cannot be applied to speech uttered by several participants. Currently, sensor-based servoing methods that are able to combine visual and audio features, possibly associated with several persons, are not available. When several modalities and hence several types of sensors are available, it is

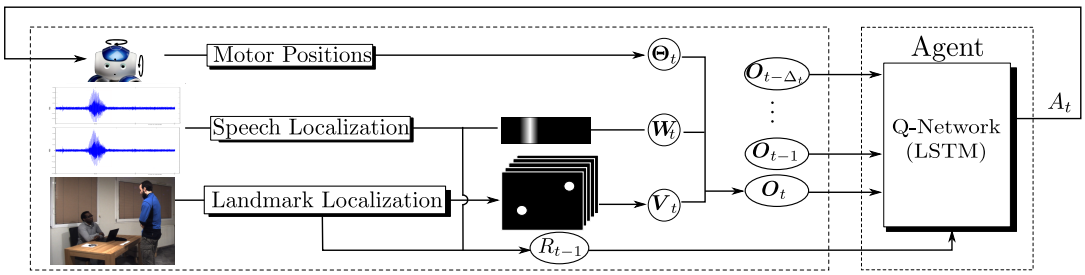


Figure 3.1: Overview of the proposed deep RL method for controlling the gaze of a robot. At each time index t, audio and visual data are represented as features maps which, together with motor positions, form the set of observations $\boldsymbol{O}_t$. A motor action A_t (rotate left, right, up, down, or stay still) is selected based on past and present observations via maximization of current and future rewards. The rewards R are based on the number of visible persons as well as on the presence of speech sources in the camera field of view. We use a deep Q-network (DQN) model that can be learned both offline and online. Please refer to Section 5.3 and Section 3.4 for the mathematical notations and detailed problem formulation.

difficult to optimally fuse the available sensory data and to implement an optimal controller, based on handcrafted rules that must consider all the situations that may occur.

In this chapter, we propose a reinforcement learning approach [168] to the gaze control problem, e.g. Fig. 3.1. Reinforcement learning (RL) has several advantages over sensor-based servoing as it replaces a handcrafted control strategy with a trial-and-error learning model. Over time, the agent, e.g. the robot, refines its behavior via optimization of a reward-based function that may well be viewed as a feedback signal that indicates whether the robot actions are beneficial or not. The model can be trained both offline and online, which yields interesting adaptation capabilities. As it will be described in detail below, there is no need of an annotated training dataset as is often the case with machine learning techniques.

The chapter has the following contributions. We built a novel audio-visual fusing framework that is well suited for controlling the gaze of a robotic head in a multi-party interaction scenario. We map the gaze control problem in the framework of RL and we propose a reward function based on the available temporal sequence of camera and microphone observations. We use deep RL to model the action-value function, and suggest several deep architectures based on LSTM (a recurrent neural network model) that allow us to experiment with early fusion and late fusion of audio and visual data. We introduce a simulated environment that enables us to learn the proposed deep RL model without the need of spending hours of tedious interaction. By experimenting on a publicly available dataset and on a real robot, we provide empirical evidence that our method achieves state-of-the-art performance.

This chapter is organized as follows. Section 3.2 describes related work. Section 5.3 presents the proposed mathematical formulation and Section 3.4 describes the deep reinforcement learning architectures. Section 3.5 briefly describes the simulated environment needed for offline training. Section 3.6 reports experiments and results obtained with a publicly available dataset and with a Nao robot.

3.2 Related Work

RL has been successfully employed in different domains, including robotics [85]. The RL goal is to find a function, called a policy, which specifies which action to take in each state, so as to maximize some function ($e.g.$, the mean or expected discounted sum) of the sequence of rewards. Therefore, learning the suitable policy is the main challenge, and there are two main categories of methods to address it. First, policy-based methods define a space from the set of policies, and sample policies from this space. The reward is then used, together with optimization techniques, *e.g.* gradient-based methods, to increase the quality of subsequent sampled policies [182]. Second, value-based methods consist in estimating the expected reward for the set of possible actions, and the actual policy uses this value function to decide the suitable action, *e.g.* choose the action that maximizes the value-function. In particular, popular value-based methods include Q-learning [180] and its deep learning extension, Deep Q-Networks (or DQNs) [117].

There are several RL-based HRI methods relevant to our work. In [58] an RL algorithm is used for a robot to learn to play a game with a human partner. The algorithm uses vision and force/torque feedback to choose the motor commands. The uncertainty associated with human actions is modeled via a Gaussian process model, and Bayesian optimization selects an optimal action at each time step. In [116] RL is employed to adjust motion speed, timing, interaction distances, and gaze in the context of HRI. The reward is based on the amount of movement of the subject and the time spent gazing at the robot in one interaction. As external cameras are required, this cannot be easily applied in scenarios where the robot has to keep learning in a real environment. Moreover, the method is limited to the case of a single human participant. Another example of RL applied to HRI can be found in [172], where a human-provided reward is used to teach a robot. This idea of interactive RL is also exploited in [30] in the context of a table-cleaning robot. Visual and speech recognition are used to get advice from a parent-like trainer to enable the robot to learn a good policy efficiently. An extrinsic reward is used in [144] to learn how to point a camera towards the active speaker in a conversation. Audio information is used to determine where to point the camera, while the reward is provided using visual information: the active speaker raises a blue card that can be easily identified by the robot. The use of a multimodal deep Q-network (DQN) to learn human-like interactions is proposed in both [135] and [136]. The robot must choose an action to shake hands with a person. The reward is either negative, if the robot tries unsuccessfully to shake hands, positive, if the hand-shake is successful, or null otherwise. In practice, the reward is obtained from a sensor located in the hand of the robot and it takes fourteen training days to learn this skill successfully. To the best of our knowledge, the closest work to ours is [176] where an RL approach learns good policies to control the orientation of a mobile robot during social group conversations. The robot learns to turn its head towards the speaking person. However, their model is learned on simulated data that are restricted to a few predefined scenarios with static people and a predefined spatial organization of the group.

As already mentioned, gaze control has been addressed in the framework of sensor-

based servoing. In [13] a method is proposed that uses audio-visual input to detect, track, and involve multiple persons into an interaction. In a multi-person scenario, [6] investigated the complementary nature of tracking and visual servoing that enables the system to track several persons and to visually control the gaze such as to keep a selected person in the camera field of view. Also, in [193], a system for gaze control of socially interactive robots in multiple-person scenarios is presented. This method requires external sensors to locate human participants. However, in opposition to all these works, we aim at learning the optimal behavior for gaze control, using the minimal supervision represented by a reward function, instead of adopting an arbitrary and handcrafted gaze control strategy.

3.3 Reinforcement Learning for Gaze Control

We consider a robot which should gaze towards a group of people. Hence, the robot must learn by itself a gaze control strategy via a trial-and-error procedure. The desired robot action is to rotated its head, on which are mounted a camera and two microphones, such as to maximize the number of persons visible in the camera field-of-view. Moreover, the robot should prefer to look at speaking people. The overall architecture of the proposed methodology is shown in Fig. 3.1. The terms *agent* and *robot* will be used indistinctly.

Random variables and their realizations are denoted with uppercase and lowercase letters, respectively. Vectors and matrices are in bold italic. At each time index t, the agent gathers motor $\boldsymbol{\Theta}_t$, visual $\boldsymbol{V}_t$, and audio $\boldsymbol{W}_t$ observations and performs an action $A_t \in \mathcal{A}$ from an action set according to a policy π, i.e. controlling the head motors such that the robot gazes in a selected direction. Once an action is performed, the agent receives a reward R_t, as explained in detail below.

Without loss of generality we consider the companion robot Nao whose head has two rotational degrees of freedom, pan and tilt. Motor observations correspond to pan and tilt angles, $\boldsymbol{\Theta}_t = (\Theta_t^1, \Theta_t^2)$. The values of these angles are relative to a reference head orientation, e.g. aligned with the robot body. This reference orientation together with the motor limits define the robot-centered *motor field-of-view*, or M-FOV.

We use the multiple person detector of [18] to estimate visual landmarks for each detected person, namely the nose, eyes, ears, neck, shoulders, elbows, wrists, hip, knees and ankles, or a total of $J = 18$ possible landmarks for each person. Based on the detection of these landmarks, one can determine the number of (totally or partially) observed persons, N_t, as well as the number of observed faces, F_t. Notice that in general the number of faces that are present in the image (i.e. detection of nose, eyes or ears) may be smaller than the number of detected persons. The landmark coordinates are described in image coordinates. Since the camera is mounted onto the robot head, the landmarks are described in a head-centered reference system. The visual landmarks are represented by J binary grids of size $K_v \times L_v$, namely $\boldsymbol{V}_t \in \{0,1\}^{K_v \times L_v \times J}$, where 1 (or zero) corresponds to the presence (or absence) of a landmark. Notice that this representation gathers all the detected landmarks associated with the N_t detected persons.

Audio observations are provided by the multiple speech-source localization method described in [103]. Audio observations are also represented with a binary grid of size $K_a \times L_a$, namely $\boldsymbol{W}_t \in \{0,1\}^{K_a \times L_a}$. A grid cell is set to 1 if a speech source is detected at that grid location and 0 otherwise. The audio grid is robot-centered and hence it remains fixed whenever the robot turns its head. Moreover, the audio grid spans an *acoustic field-of-view*, or A-FOV, which is much wider than the *visual field-of-view*, or V-FOV, associated with the camera mounted onto the head. The motor observations allow to estimate the relative alignment between the audio and visual grids and to determine whether a speech source lies within the visual field-of-view or not. This is represented by the binary variable $\Sigma_t \in \{0,1\}$, such that $\Sigma_t = 1$ if a speech source lies in the visual field-of-view and $\Sigma_t = 0$ if none of the speech sources lies inside the visual field-of-view.

Let $\boldsymbol{O}_t = \{\boldsymbol{\Theta}_t, \boldsymbol{V}_t, \boldsymbol{W}_t\}$ and let $\boldsymbol{S}_t = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_t\}$ denote the state variable. Let the set of actions be defined by $\mathcal{A} = \{\varnothing, \leftarrow, \uparrow, \rightarrow, \downarrow\}$, namely either remain in the same position or turn the head by a fixed angle in one of the four cardinal directions. We propose to define the reward R_t as follows:

$$R_t = F_{t+1} + \alpha \Sigma_{t+1}, \tag{3.1}$$

where $\alpha \geq 0$ is an adjustment parameter. High α values return high rewards when speech sources lie within the camera field-of-view. We consider two types of rewards which are referred to in Section 3.6 as *Face_reward* ($\alpha = 0$) and *Speaker_reward* ($\alpha > 0$). Notice that the number of observed faces F_t is independent of each person's speaking status. Upon the application at hand, the value of α allows one to weight the importance given to speaking persons.

In RL, the model parameters are learned on sequences of states, actions and rewards, called episodes. At each time index t, an optimal action A_t should be chosen by maximizing the immediate and future rewards, $R_t, R_{t+1}, \ldots, R_T$. We make the standard assumption that future rewards are discounted by a factor γ that defines the importance of short-term rewards as opposed to longer term ones. We define the discounted future return $\bar{R}_t$ as the discounted sum of future rewards, $\bar{R}_t = \sum_{\tau=0}^{T-t} \gamma^\tau R_{\tau+t}$. If $\gamma = 0$, $\bar{R}_t = R_t$ and, consequently, we aim at maximizing only the immediate reward whereas when $\gamma \approx 1$, we favor policies that leads to better rewards in the long term. Considering a fixed value of γ, we now aim at maximizing $\bar{R}_t$ at each time index t. In other words, the goal is to learn a policy, $\pi(a_t, \boldsymbol{s}_t) = P(A_t = a_t | \boldsymbol{S}_t = \boldsymbol{s}_t)$ with $(a_t, \boldsymbol{s}_t) \in \mathcal{A} \times \mathcal{S}$, such that if the agent chooses its actions according to the policy π, the expected $\bar{R}_t$ should be maximized. The Q-function (or the action-value function) is defined as the expected future return from state $\boldsymbol{S}_t$, taking action A_t and then following any given policy π:

$$Q_\pi(\boldsymbol{s}_t, a_t) = \mathbb{E}_\pi[\bar{R}_t | \boldsymbol{S}_t = \boldsymbol{s}_t, A_t = a_t]. \tag{3.2}$$

Learning the best policy corresponds to the following optimization problem $Q^*(\boldsymbol{s}_t, a_t) = \max_\pi [Q_\pi(\boldsymbol{S}_t = \boldsymbol{s}_t, A_t = a_t)]$. The optimal Q-function obeys the identity known as the Bellman equation:

$$Q^*(\boldsymbol{s}_t, a_t) = \mathbb{E}_{\boldsymbol{S}_{t+1}, R_t}\Big[R_t + \gamma \max_a (Q^*(\boldsymbol{S}_{t+1}, a)) \Big| \boldsymbol{S}_t = \boldsymbol{s}_t, A_t = a_t\Big] \tag{3.3}$$

This equation corresponds to the following intuition: if we have an estimator $Q^*(\boldsymbol{s}_t, a_t)$ for $\bar{R}_t$, the optimal action a_t is the one that leads to the largest expected $\bar{R}_t$. The recursive application of this policy leads to equation (3.3). A straightforward approach would consist in updating Q at each training step i with:

$$Q^i(\boldsymbol{s}_t, a_t) = \mathbb{E}_{\boldsymbol{S}_{t+1}, R_t}\left[R_t + \gamma \max_a (Q^{i-1}(\boldsymbol{S}_{t+1}, a)) \middle| \boldsymbol{S}_t = \boldsymbol{s}_t, A_t = a_t\right] \tag{3.4}$$

Following equation 3.4, we estimate each action-value $Q^i(\boldsymbol{s}_t, a_t)$ given that we follow, for the next time steps, the policy implied by Q^{i-1}. In practice, we approximate the true Q function by a parametric function. In our case, we employ a network $Q(\boldsymbol{s}, a, \boldsymbol{\omega})$ parametrized by weights $\boldsymbol{\omega}$ to estimate the Q-function $Q(\boldsymbol{s}, a, \boldsymbol{\omega}) \approx Q^*(\boldsymbol{s}, a)$. We minimize the following loss:

$$\mathcal{L}(\boldsymbol{\omega}_i) = \mathbb{E}_{\boldsymbol{S}_t, \boldsymbol{A}_t, \boldsymbol{R}_t, \boldsymbol{S}_{t+1}}\left[(Y_{i-1} - Q(\boldsymbol{S}_t, \boldsymbol{A}_t, \boldsymbol{\omega}_i))^2\right] \tag{3.5}$$

with $Y_{i-1} = R_t + \gamma \max_a (Q(\boldsymbol{S}_{t+1}, a, \boldsymbol{\omega}_{i-1}))$. It can been seen as minimizing the mean squared distance between the approximations of the right and left hand sides of (3.4). In order to compute (3.5), we sample quadruplets $(\boldsymbol{S}_t, \boldsymbol{A}_t, \boldsymbol{R}_t, \boldsymbol{S}_{t+1})$ following the policy implied by Q^{i-1}:

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q(\boldsymbol{s}_t, a, \omega_{i-1}) \tag{3.6}$$

However, instead of sampling only according to 3.6, random actions a_t are taken in ϵ percents of the time steps in order to explore new strategies. This approach is known as epsilon-greedy policy. $\mathcal{L}$ is minimized over $\boldsymbol{\omega}_i$ by stochastic gradient descent. Refer to [118] for more technical details about the training algorithm.

3.4 Proposed DQN Architectures

We propose to model the Q-function with a long short-term memory (LSTM) [69] recurrent neural network that takes as input $\boldsymbol{S}_t^{\Delta t} = \{\boldsymbol{O}_{t-\Delta t}, \ldots, \boldsymbol{O}_t\}$ and that outputs a vector of size $\#\mathcal{A}$ that corresponds to each $Q(\boldsymbol{s}_t^{\Delta t}, a_t, \boldsymbol{\omega})$ with $a_t \in \mathcal{A}$, i.e. Section 5.3. We argue that LSTM is well-suited for our task as it is capable of learning temporal dependencies better than other recurrent neural networks and than hidden Markov models. In practice, when a person is not detected anymore, the network should be able to use previous detections (back in time) in order to predict the direction towards which the robot should be gazing. The J grids of $\boldsymbol{V}_t$ are flattened before the LSTM layers. Batch normalization is applied to the output of the LSTM in order to accelerate training [76]. Following [118], the output layer is a fully-connected layer (FCL) with linear activations.

Four different network architectures were tested. They are described below and evaluated in Section 3.6. In order to evaluate how the visual and audio streams of information

should be fused, we propose to compare two strategies: *early fusion* and *late fusion*. In early fusion, *EFNet*, the unimodal features are combined into a single representation before modeling time dependencies, i.e. Fig. 3.2a. In late fusion, *LFNet*, visual and audio features are processed separately before they are fused, i.e. Fig. 3.2b. In order to measure the impact of each modality, we propose two more network architectures that use either visual-only, *VisNet*, or audio-only, *AudNet*, input, e.g. Fig. 3.2c, where we used the compact graphical representation proposed in [60].

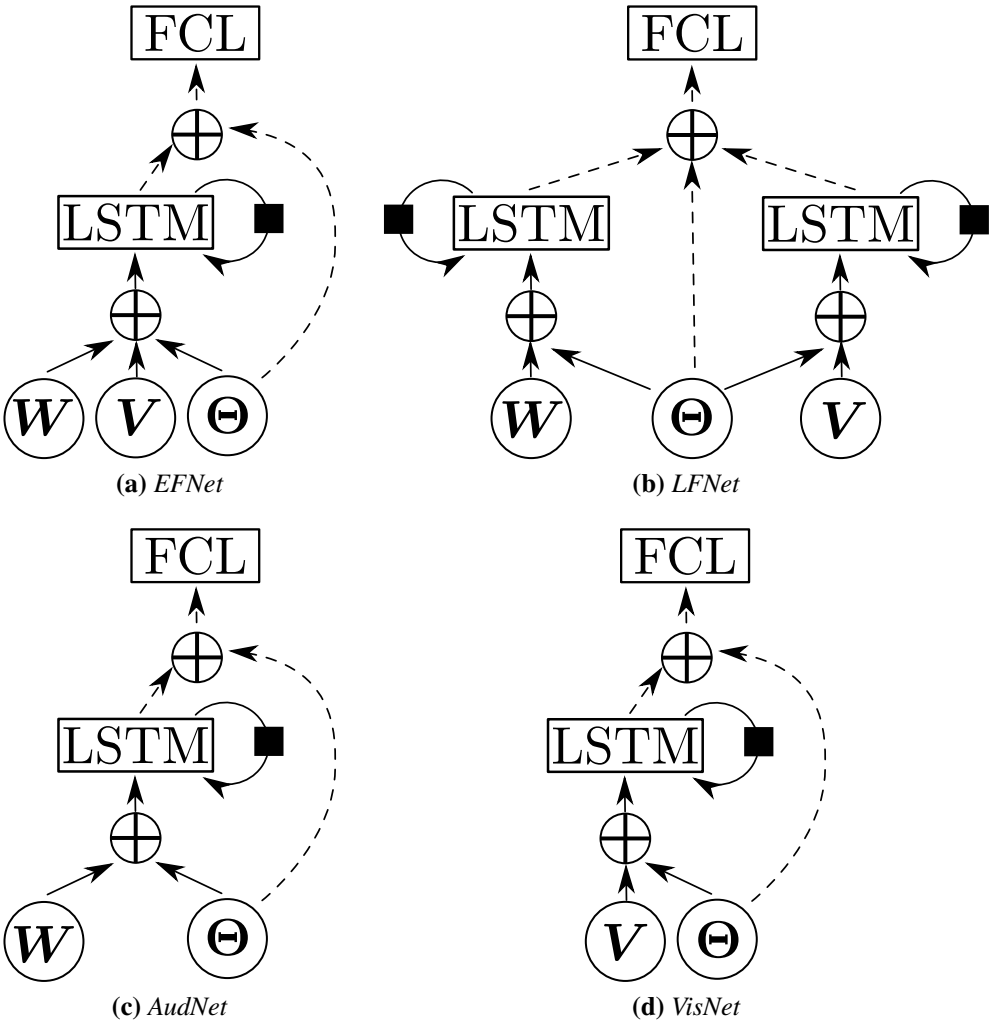


Figure 3.2: Proposed architectures to model the Q-function. Dashed lines indicate connections only used in the last time step. Black squares represent a delay of a single time step. Circled crosses represent the concatenation of inputs. FCL outputs a Q-value for each action.

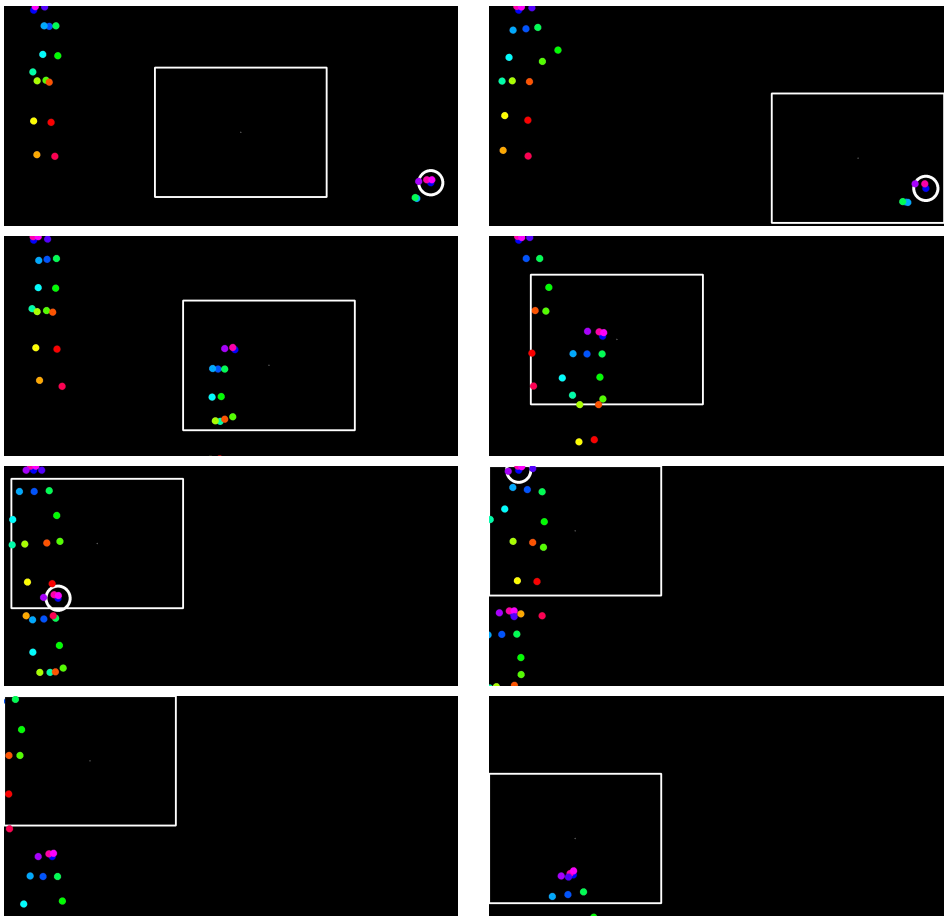


Figure 3.3: Example of a simulated sequence used for offline training. The field of view is shown with a white rectangle. Visual landmarks associated with two persons are shown as colored dots. The white circles correspond to simulated speech sources that may correspond to a person.

3.5 Simulated Environment for Training

Training a DQN model from scratch may require long periods until convergence, e.g. of the order of 150000 time steps in our case. Moreover, using a robot for training may not be convenient for two reasons. First, each robotic action takes an irreducible time. Second, in the case of HRI, participants would need to be actually present in front of a robot for tens of hours and to mimic realistic behaviors. Therefore, we propose to perform training using a simulated environment. DQN is learned using a simulated robot and people that move and speak. Then the Q-function thus learned is used to initialize the DQN associated with real gaze control in the presence of people. Importantly, the network learned from this simulated environment can be successfully used by the robot without the need of fine-tuning with real data. In this simulated environment, we do not need to generate realistic images and sounds, instead we directly generate observations and rewards as needed by DQN learning.

Using the formulation introduced in Section 5.3 we defined motor, acoustic, and visual fields of view that correspond to the robot characteristics, e.g. Fig. 3.3. Without loss of generality we assumed that the motor field of view is the same as the acoustic field of view.

We simulated people that can freely move in a space that is larger than the motor/acoustic field of view. This allows us to consider people that randomly enter and quit this field of view. To simulate realistic human movements, we applied the person detector of [18] to the AVDIAR dataset [56] in order to collect a large number person poses and their associated landmarks. Realistic human trajectories were obtained using a smoother. Then the landmarks and their trajectories were mapped onto our simulated environment such that people move at different speeds, suddenly change their trajectories, come in and out the motor field of view, etc. Speech sources were simulated as follows. Three situations were randomly selected: one speaking person, two speaking persons, and no speaking person. A Markovian model was used to enforce temporal continuity of the speaking status. In addition, we also simulated speech sources that do not correspond to a person location. For more technical details, please refer to section A.1 in Appendix.

3.6 EXPERIMENTS

3.6.1 EVALUATION WITH RECORDED DATA

The evaluation of HRI systems is not an easy task. In order to fairly compare different models, we need to train and test the different models on the exact same data. In the context of RL and HRI, this is problematic because the data, i.e. what the robot actually sees and hears, depends on the action taken by the robot. Thus, we propose to first evaluate our model with the AVDIAR dataset [56]. This dataset was recorded with four microphones and one high-resolution camera (1920×1080 pixels). These images, due to their wide field of view, are suitable to simulate the motor field of view of the robot. In practical terms, only a small box of the full image simulates the robot's camera field of view.

However, it is important to highlight that transferring the model learned using AVDIAR to Nao is problematic. First, faces are almost always located at the same position (around the image center). Second, all videos are recorded indoors using only two different rooms, and participants are not moving too much. Finally, the audio setting is unrealistic for a robotics scenario, e.g. absence of motor noise. Therefore, the main reason for using the AVDIAR dataset is to compare our method with other methods.

3.6.2 LIVE EXPERIMENTS WITH NAO

In order to carry out an online evaluation of our method, we performed experiments with a Nao robot. Nao has a 640×480 pixels cameras and four microphones. This robot is particularly well suited for HRI applications because of its design, hardware specifications and affordable cost. Nao's commercially available software can detect people, locate

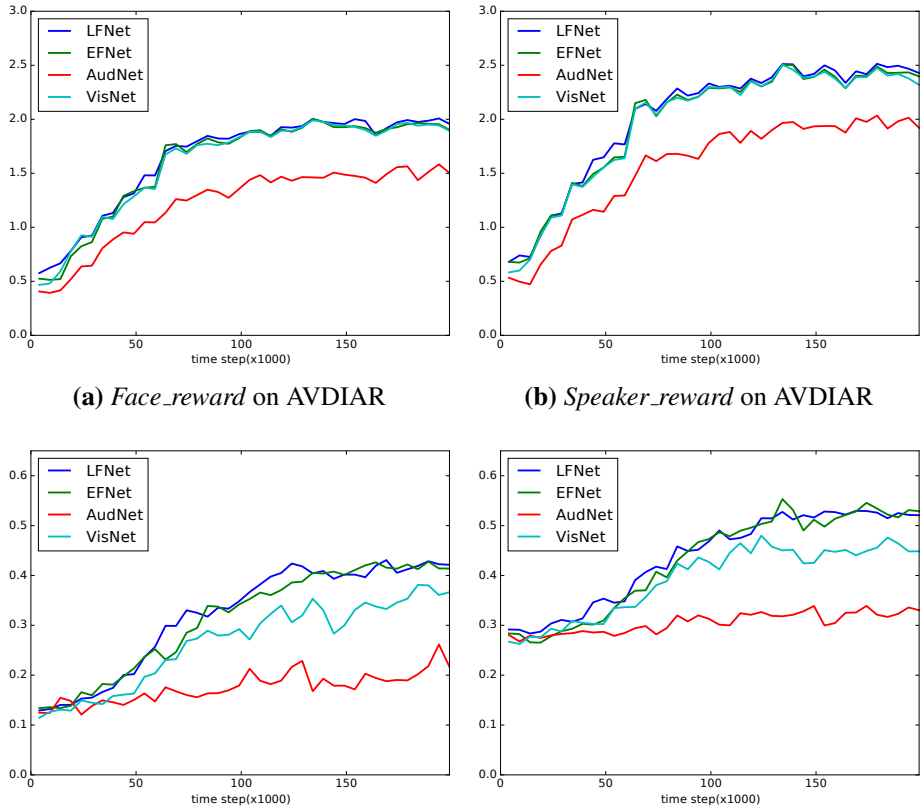
sounds, understand some spoken words, synthesize speech and engage itself in simple and goal-directed dialogs. Our gaze control system is implemented on top of the NAOLab middleware [5] that synchronizes proprioceptive data (motor readings) and sensor information (image sequences and acoustic signals). The reason why we use a middleware is threefold. First, the implementation is platform-independent and, thus, easily portable. Platform-independence is crucial since we employ a transfer learning approach to transfer the model parameters, obtained with the proposed simulated environment, to the Nao software/hardware platform. Second, the use of external computational resources is transparent. This is also a crucial matter in our case, since visual processing is implemented on a GPU which is not available onboard of the robot. Third, the use of middleware makes prototyping much faster. For all these reasons, we employ the remote and modular layer-based middleware architecture named NAOLab. NAOLab consists of four layers: drivers, shared memory, synchronization engine and application programming interface (API). Each layer is divided into three modules devoted to vision, audio and proprioception, respectively. The last layer of NAOLab provides a general programming interface in C++ to handle the sensory data and to manage its actuators. NAOLab provides, at each time step, an image and the direction of the detected sound sources using [102, 103].

It is important to highlight that we pre-train the proposed model using the simulated environment before running live experiments on Nao. This environment is flexible and allows us to be closer to the actual conditions that Nao would face in practice (field of view range, uniform location of the people, etc.). For instance, in AVDIAR, heads are almost always at the same height. As a consequence, the learned model would not be sufficiently general and flexible to perform well in real scenarios.

3.6.3 Implementation Details

We managed to obtain the full-body pose using [18] in less than 100 ms by carefully selecting the resolution used to perform the detection. Considering that NAOLab gathers images at 10 FPS, this pose estimator can be considered as fast enough for our scenario. Moreover, [18] follows a bottom-up approach, which allows us to speed-up landmark detection by skipping the costly association step.

The parameters of our model are based on a preliminary experimentation. We set $\Delta_T = 4$ in all scenarios, such that each decision is based on the last 5 observations. The output size of LSTM is set to 30 (since a larger size does not provide an improvement in performance), and the output size of the FCL is set to 5 (one per action). We use a discount factor (γ) of 0.90. Concerning the training phases, we employed the Adam optimizer [83] and a batch size of 128. In order to help the model to explore the policy space, we use an ϵ-greedy algorithm: while training, a random action is chosen in $\epsilon\%$ of the cases; we decrease linearly the ϵ value from $\epsilon = 90\%$ to $\epsilon = 10\%$ after 120000 iterations. Concerning the observations, we employ visual and audio grids of sizes 7×5 for the three environments used in our experiments. The models were trained in approximately 45 minutes on both AVDIAR and the simulated environment. It is interesting to notice that we obtain this training time without using GPUs. A GPU is only needed for person



(a) *Face_reward* on AVDIAR **(b)** *Speaker_reward* on AVDIAR

(c) *Face_reward* on Simulated **(d)** *Speaker_reward* on Simulated

Figure 3.4: Evolution of the reward obtained while training with the two proposed rewards on the AVDIAR dataset and on the simulated environment. We average over a 5000 time-step window for a cleaner visualization.

detection and estimation of visual landmarks (in our case, a Nvidia GTX 1070 GPU).

We also provide details specifically related to the Nao implementation. The delay between two successive observations is $\sim$0.3 seconds. The head has a motor field of view 180 degrees. The head motion parameters are chosen such that a single action corresponds to 0.15 radians ($\sim 9^\circ$) and 0.10 radians ($\sim 6^\circ$) for horizontal and vertical motions, respectively. Concerning the AVDIAR dataset, we employ 16 videos for training. The amount of training data is doubled by flipping the video and the Audio maps. In order to save computation time, the original videos are down-sampled to 1024×640 pixels. The size of the camera field of view where faces can be detected is set to 300×200 pixels using motion steps of 36 pixels each. These dimensions approximately correspond the coverage angle and motion of Nao. At the beginning of each episode, the position of the camera field of view is selected such that it contains no face. We noticed that this initialization procedure favors the exploration abilities of the agent. To avoid a bias due to the initialization procedure, we used the same seed for all our experiments and iterated three times over the 10 test videos (20 when counting the flipped sequences). An action

is taken every 5 frames (0.2 seconds). In the simulated environment, the size of field in which the people can move is set to $\xi = 1.4$. In the case of Nao, the audio observations are provided by the multiple speech-source localization method described in [103].

3.6.4 Results and Discussion

In all our experiments, we run five times each model and display the mean of five runs to lower the impact of the stochastic training procedure. On AVDIAR, the results on both training and test sets are reported in the tables. As described previously, the simulated environment is randomly generated in real time, so there is no need for a separated test set. Consequently, the mean reward over the last 10000 time steps is reported as test score.

3.6.5 Results and Discussion

Table 3.1: Comparison of the final reward obtained using different window lengths (Δ_T). The mean and standard deviation over 5 runs are reported. The best average results obtained are displayed in bold. The training time is reported for each configuration

	AVDIAR			*Simulated*	
$\Delta_T + 1$	Training	Test	Time(s$\times 10^3$)	Test	Time(s$\times 10^3$)
1	1.92 ± 0.03	1.82 ± 0.03	3.05 ± 0.22	0.26 ± 0.04	3.07 ± 0.15
2	1.94 ± 0.02	$\mathbf{1.85 \pm 0.02}$	2.25 ± 0.99	0.36 ± 0.04	3.09 ± 0.17
3	1.93 ± 0.01	1.84 ± 0.01	2.95 ± 0.38	0.42 ± 0.02	2.98 ± 0.27
5	1.94 ± 0.02	1.84 ± 0.02	3.30 ± 0.46	$\mathbf{0.43 \pm 0.01}$	3.40 ± 0.14
10	1.94 ± 0.02	1.84 ± 0.02	2.05 ± 0.22	0.40 ± 0.02	3.85 ± 0.36
20	$\mathbf{1.96 \pm 0.01}$	1.82 ± 0.02	3.00 ± 0.00	0.42 ± 0.02	5.35 ± 0.36
128	1.94 ± 0.02	1.82 ± 0.03	18.90 ± 0.77	0.41 ± 0.03	52.98 ± 5.23

First, we describe the experiments devoted to evaluate the impact of some of the principal parameters involved. Different window sizes (i.e. the number of past observations necessary to make a decision) are compared in Table 3.1. We can conclude that the worst results are obtained when only the current observation is used (window size of 1). We also observe that, on AVDIAR, the model performs well even with short window lengths (2 and 3). In turn, with a more complex environment, as the proposed simulated environment, a longer window length tends to perform better. We interpret that using a larger window size helps the network to ignore the noisy observations and to remember the position of people that left the field of view. We report the training time for each window length. We observe that, using a smaller time window speeds up training since it avoids back-propagating the gradient deeply in the LSTM network.

In Table 3.2, different discount factors are compared. We notice that, on AVDIAR, high discount factors are prone to overfit as the difference in performance between training and test is higher. On the simulated environment, low discount values perform worse because we think that, as the environment is more complex, the model may need several actions to reach a face. Consequently, a model that is able to take into account the future

benefit of each action performs better. Finally, in Table 3.3, we compare different LSTM sizes. We observe that increasing the size does not lead to better results; an interesting conclusion since, from a practical point of view, smaller LSTMs are faster to train.

Table 3.2: Comparison of the final reward obtained using different discounted factors (γ). The mean and standard deviation over 5 runs are reported. The best average results obtained are displayed in bold.

	AVDIAR		*Simulated*
γ	Training	Test	
25	$\mathbf{1.96 \pm 0.02}$	1.85 ± 0.02	0.33 ± 0.09
50	$\mathbf{1.96 \pm 0.02}$	$\mathbf{1.86 \pm 0.03}$	0.35 ± 0.08
75	$\mathbf{1.96 \pm 0.02}$	1.85 ± 0.02	$\mathbf{0.43 \pm 0.11}$
90	1.94 ± 0.02	1.83 ± 0.02	0.42 ± 0.12
99	1.95 ± 0.01	1.84 ± 0.02	0.42 ± 0.12

Table 3.3: Comparison of the final reward obtained using different LSTM sizes. The mean and standard deviation over 5 runs are reported. The best average results obtained are displayed in bold.

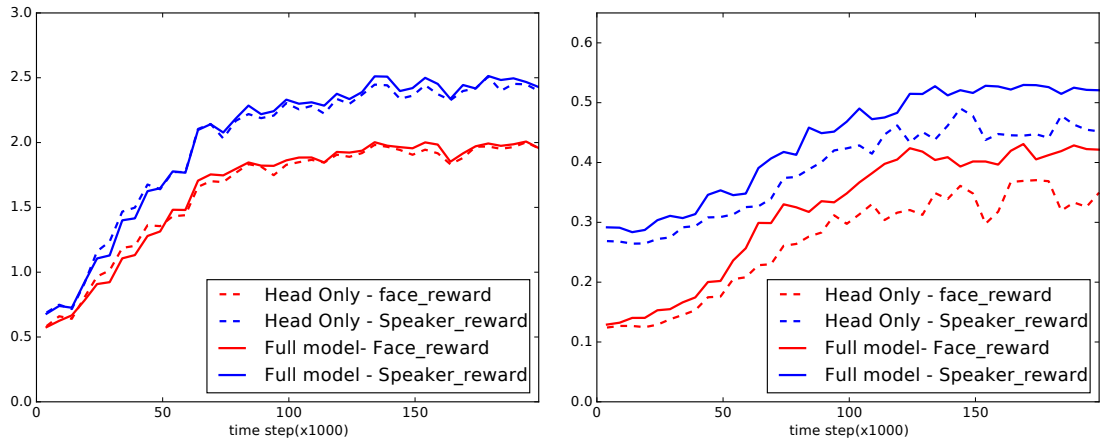
LSTM size	AVDIAR		*Simulated*
	Training	Test	
30	$\mathbf{1.96 \pm 0.01}$	1.85 ± 0.03	0.42 ± 0.11
60	1.95 ± 0.02	1.86 ± 0.02	$\mathbf{0.43 \pm 0.12}$
120	1.92 ± 0.04	$\mathbf{1.87 \pm 0.02}$	0.41 ± 0.10

Table 3.4: Comparison of the reward obtained with different architectures. The best results obtained are displayed in bold.

	AVDIAR				Simulated	
	Face		Speaker		*Face*	*Speaker*
Network	*Training*	*Test*	*Training*	*Test*		
AudNet	1.50 ± 0.03	1.47 ± 0.04	1.92 ± 0.02	1.82 ± 0.03	0.21 ± 0.01	0.33 ± 0.01
VisNet	1.89 ± 0.03	$\mathbf{1.85 \pm 0.02}$	2.32 ± 0.04	2.23 ± 0.03	0.37 ± 0.04	0.45 ± 0.06
EFNet	1.90 ± 0.03	1.81 ± 0.04	2.40 ± 0.02	2.22 ± 0.03	0.41 ± 0.03	$\mathbf{0.53 \pm 0.03}$
LFNet	$\mathbf{1.96 \pm 0.02}$	1.83 ± 0.02	$\mathbf{2.43 \pm 0.02}$	$\mathbf{2.29 \pm 0.02}$	$\mathbf{0.42 \pm 0.01}$	0.52 ± 0.03

In Table 3.4, we compare the final reward obtained while training on the AVDIAR dataset and on our simulated environment with the two proposed rewards (*Face_reward* and *Speaker_reward*). Four different networks are tested: *EFNet*, *LFNet*, *VisNet*, and *AudNet*. The y-axis of Figure 3.4 shows the average reward per episode, with a clear growing trend as the training time passes (specially in the experiments with the AVDIAR dataset), meaning that the agent is learning (improving performance) from experience. The best results are indistinctly provided by the late and early fusion strategies (*LFNet* and *EFNet*), showing that our model is able to effectively exploit the complementarity of both modalities. We observe that the rewards we obtain on AVDIAR are higher than those obtained on the simulated environment. We suggest two possible reasons. First, the simulated environment has been specifically designed to enforce exploration and tracking abilities. Consequently, it poses a more difficult problem to solve. Second, the number

of people in AVDIAR is higher (about 4 in average), thus finding a first person to track would be easier. We notice that, on the AVDIAR dataset using the *Face_reward*, we obtain a mean reward greater than 1, meaning that, on average, our model can see more than one face per frame. We also observe that *AudNet* is the worst performing approach. However, it performs quite well on *AVDIAR* compared to the simulated environment. This behavior can be explained by the fact that, on *AVDIAR*, the speech source detector returns a 2D heatmap whereas only the yaw angle is used in the simulated environment. As conclusion, we select *LFNet* to perform experiments on Nao.



(a) AVDIAR **(b)** Simulated

Figure 3.5: Evolution of the training reward obtained when using as visual observation the result of either the full-body pose estimation or the face location information.

Figure 3.5 displays the reward obtained when using only faces as visual observation (dashed lines) in contrast to using the full-body pose estimation (continuous lines). We observe that for on the simulated data, the rewards are significantly higher when using the full-body pose estimator. This figure intends to respond empirically to the legitimate question of why a full-body pose estimator is used instead of a simple face detector. From a qualitative point of view, the answer can be found in the type of situations that can solve one and the other. Let's imagine that the robot looks at the legs of a user; in case of using only a face detector, there is no clue that could help the robot to move up its head in order to see a face; however, if a human full-body pose detector is used, the detection of legs implies that there is a torso over them, and a head over the torso. Figure 3.6 shows a short sequence of the AVDIAR environment, displaying the whole field covered by the AVDIAR videos as well as the smaller field of view captured by the robot (the red rectangle in the figure).

Concerning the experiments performed on Nao, Figure 3.7 shows an example of a two-person scenario using the *LFNet* architecture. We managed to transfer the exploration and tracking abilities learned using the simulated environment. In our experiments, we see that our model behaves well independently of the number of participants, and the main failure cases are related to quick movements of the participants.

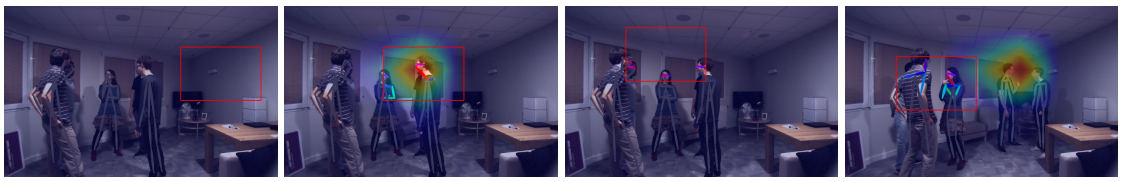


Figure 3.6: Example of a sequence from the AVDIAR dataset. The speech direction binary grid is superimposed on the image, and the visible landmarks are displayed using a colored skeleton. The camera field of view (in red) is randomly initialized (far left), speech emitted by one of the persons is detected and hence the gaze is controlled (left). The agent manages to get all the persons in the field of view (right), and it gazes at a group of three persons when two other persons move apart (far right).

Figure 3.7: Example of a live sequence with two persons. First row shows an overview of the scene, including the participants and the robot. Second row shows the images gathered with the camera mounted onto the robot head. The robot head is first initialized in a position where no face is visible (first column), and the model uses the available landmarks (elbow and wrist) to find the person onto the right (second column). The robot detects the second person by looking around while keeping the first person in its field of view (third column), and gazes the two people walking together (fourth column).

We now perform a comparative evaluation with respect to the state of the art. To the best of our knowledge, there is no existing work that tackles the problem of finding an optimal head motion policy in the HRI context. Only Bennewitz et al. [13] propose a heuristic that uses an audio-visual input to detect, track and involve multiple persons into interaction. We compare our learned policy with their proposed algorithm. On the simulated environment, as the speech source detector does not provide vertical information (see section 3.6.3), in the case where no person has been observed so far but a sound is detected, we randomly move along the vertical axis corresponding to the horizontal speech source position. In their experiments, Ban et al. [6] propose two strategies to evaluate their visual head control method. A first strategy consists in following a person and orienting the robot head in order to align the person's face with the image center. A second strategy consists in randomly jumping every 3 seconds between persons. Obviously, the second strategy was designed as a toy experiment and does not correspond to a natural behavior. Therefore, we compare our RL approach with their first strategy. Unfortunately, the case where nobody is in the field of view is not considered in [6]. To be able to compare their method in the more general scenario we tackle, we propose the

Table 3.5: Comparison of the rewards obtained with different handcrafted policies. The performances of competitor methods are reported considering the two speed assumptions (*equal/infinite*) described in the text.

	AVDIAR		Simulated	
	Face_reward	*Speaker_reward*	*Face_reward*	*Speaker_reward*
Ban et al.[6]+*Rand*	1.19/1.21	1.45/1.59	0.25/0.26	0.40/0.37
Ban et al.[6]+*Center*	1.62/1.68	1.95/2.01	0.14/0.11	0.28/0.29
Ban et al.[6]+*Body*	1.23/1.20	1.40/1.52	0.27/0.26	0.39/0.37
Ban et al.[6]+*Audio*	1.54/1.63	1.84/2.06	0.32/0.39	0.43/0.48
Bennewitz et al.[13]	1.56/1.55	2.07/2.05	0.30/ **0.42**	0.35/0.50
LFNet	$\mathbf{1.83 \pm 0.02}$	$\mathbf{2.29 \pm 0.02}$	$\mathbf{0.42 \pm 0.01}$	$\mathbf{0.52 \pm 0.03}$

following handcrafted policy in the case no face is detected in the field of view:

- *Rand*: A random action is chosen.
- *Center*: Go towards the center of the *acoustic field-of-view*.
- *Body*: If a limb is detected, the action $\uparrow$ is chosen in order to find the corresponding head. Otherwise, *Rand* is followed.
- *Audio*: Go towards the position of the last detected speaker.

Importantly, in our model the head motion speed is limited, since the robot can only select unitary actions. When implementing other methods, one could argue that this speed limitation is inherent to our approach and that other methods may not suffer from it. However, it is not realistic to consider that the head can move between two opposite locations of the auditory field in two consecutive frames with an infinite speed. Therefore, we report two scores in our comparison. The first one is obtained using the same speed value than the one used in our model (referred to as *equal*). The second score is obtained by making the unrealistic assumption that the head motion speed is infinite (referred to as *infinite*). This second evaluation protocol is, therefore, biased in favor of handcrafted methods. The results obtained are reported in Table 3.5.

First, we observe that no handcrafted policy can compete with our RL approach when considering models with equal head motion speeds. On both environments, *LFNet* largely outperforms all handcrafted policies. This clearly justifies the need of policy learning and the use of RL for the audio-visual gaze control. Concerning [6], *Center* obtains the best result among the [6]'s variances on AVDIAR and the worst on Simulated according to the *Face_reward* metric. It can be explained by the fact that, as mentioned in section 3.6.1, most people are located around the image center and, therefore, this dummy strategy works better than more sophisticated ones. A similar behavior can be observed with the *Speaker_reward* metric. We observe that, in both environments, using audio information when no face is detected improves the performance with respect to *Rand*. Concerning [13], it obtains the second best performance on AVDIAR with *Speaker_reward*. On the simulated environment, they equal the score obtained by our proposal when making the unrealistic assumption of infinite head motion speed. In that case, their performance is

marginally inferior to our proposal according to the *Speaker_reward*. When considering equal speed limit, our RL approach significantly outperforms their handcrafted approach (26% and 48% higher according to *Face_reward* and *Speaker_reward* respectively).

All these results highlight the major importance of audio-visual fusion in the context of gaze control for HRI, and that RL is an effective tool to tackle this task. The high variances on AVDIAR are coming from the impact of the random initial head orientation. On the contrary, our method has a low variance as it is able to adapt to any initialization. This illustrates the importance of combining tracking ability with an exploration strategy when no or only a single face is detected.

3.7 CONCLUSIONS

In this chapter, we presented a neural network-based reinforcement learning approach to solve the gaze robot control problem. In particular, our agent is able to autonomously learn how to find people in the environment by maximizing the number of people present in its field of view (and favoring people who speak). A synthetic environment is used for pretraining in order to perform transfer learning to the real environment. Neither external sensors nor human intervention are necessary to compute the reward. Several architectures and rewards are compared on three different environments: two offline (a real and a synthetic datasets) and one online (real time experiments using the Nao robot). Our results suggest that the late fusion of audio and visual information represents the best performing alternative, as well as that pretraining on synthetic data can even make unnecessary to train on real data. By thoroughly experimenting on a publicly available dataset and on a real robot, we provide empirical evidence that our method achieves state-of-the-art performance.

CHAPTER 4

DEEP MIXTURE OF LINEAR INVERSE REGRESSIONS APPLIED TO HEAD-POSE ESTIMATION

4.1 INTRODUCTION

In the previous two chapters, we presented models that are able to perform high level recognition tasks without considering how the input information, eg. full body pose, has been obtained. In the next three chapters, we focus on regression problems using deep learning approaches in order to perform these required elementary recognition tasks. Deep learning has been playing a very important role in the computer vision field during the last years. Many methods have been proposed for challenging tasks, such as image classification [86, 169] or object detection [59, 155]. State-of-the-art results in these classification tasks have been achieved with the use of Convolutional Neural Networks (ConvNets) trained to minimize a loss function on the output of a softmax layer. Besides classification, ConvNets have also been employed to solve regression problems, $e.g.$ image registration [114], organ volume estimation [195], or salient object detection [101], just to name a few. In most cases, when dealing with regression problems, the last softmax layer used in classification tasks is replaced with a fully connected regression layer with linear or sigmoid activations that minimizes an Euclidean loss. We refer to such architectures as *vanilla deep regression* in the next three chapters of this manuscript. This type of configuration ignores the existence of other regression techniques, like inverse regression models, that are suitable in high-dimensional to low-dimensional settings [37, 82, 99, 132] which are of particular interest in computer vision. To identify the benefit of using inverse regression instead of forward (or standard) regression, let's consider the simple case in which we want to estimate a linear regression from $\boldsymbol{x} \in \mathbb{R}^D$ to $y \in \mathbb{R}$, with N training samples such that $D \gg N$. The problem is ill-posed in the case of forward regression ($y = \boldsymbol{a}^\top \boldsymbol{x}, \boldsymbol{a} \in \mathbb{R}^D$) and regularization is required because one needs to estimate D parameters from only N equations. Interestingly, for linear models in the

inverse regression setting ($\boldsymbol{x} = \boldsymbol{a}^* y, \boldsymbol{a}^* \in \mathbb{R}^D$), the problem is well defined since one still needs to estimate a set of D parameters but from $D \times N$ equations.

The most common strategy when using deep learning consists in taking an architecture that has already proven to be competitive (in our case VGG-16 [157], the model with the smallest localization error on the ImageNet Large-Scale Visual Recognition Challenge 2014 [148]), download a pre-trained model, slightly modify it ($e.g.$ replacing the last softmax layer with a regression layer), and fine-tune it on the particular application under study. In this scenario, we propose a new output layer designed specifically to perform regression. This output layer is a Gaussian mixture of inverse linear regressions. Mixtures of inverse linear regressions [37] have already been successfully applied to hyper-spectral image analysis [36], sound-source localization [35] and head-pose estimation [41]. Moreover, it has been extended to mixtures of t-distributions [132] which provides an inverse regression formulation that is robust to outliers.

We believe that inverse regression models are well-suited in the deep learning framework because deep neural networks represent images in high-dimensional feature spaces that must subsequently be mapped onto low-dimensional manifolds. Interesting enough, recognizing the caveats of high-dimensional regression and exploring inverse regression models have received little attention in the literature of both computer vision and deep learning fields. In this work, we propose to couple a Gaussian mixture of linear inverse regressions with a ConvNet, we describe the methodological foundations and the associated algorithm to jointly train the network and the regression function, and we evaluate our model on the problem of head-pose estimation. The training algorithm we propose is a fine-tuning procedure designed to transfer the representations learned in a classification task to our specific problem. However, using pre-trained ConvNets to estimate the head-pose is not an easy task because very deep ConvNets have been trained to classify objects independently of their pose. As a consequence, the deep features have been designed to be as pose-invariant as possible. Conversely, in our case we want the model to be highly dependent on the pose but independent of the person.

In this chapter, we show that inverse regression outperforms *vanilla regression* models (L_2-based) currently used in head-pose estimation. Our proposal works well without the use of additional data, in opposition to other approaches yielding state-of-the-art results. Finally, it outperforms state-of-the-art methods in head-pose estimation using a widely used head-pose dataset. The implementation of the proposed method used in our experiments is publicly available[1].

4.2 Related Work

In this section, we discuss deep learning approaches in conjunction with regression methods. In addition, we review the related work on head-pose estimation, since it is used in our experiments for evaluation and comparison with other methods.

[1] `https://team.inria.fr/perception/research/dmlir/`

Deep regression algorithms, where the goal is to predict a set of interdependent continuous values, have been well studied in recent years. For instance, in human pose estimation, the target represents the positions of the human-body joints [174]; in head-pose estimation, the target represent the yaw, pitch and roll angles [119]; and in facial landmark detection, the predicted target denotes the image locations of the facial points [166]. In all aforementioned references, a ConvNet has been trained using a loss function that measures the L_2 distance of the prediction from the target, without considering its vulnerability to outliers (as evidenced by [9], where the authors argue that training a ConvNet using a loss function that is robust to outliers, *e.g.* Tukey's biweight function, results in faster convergence and better generalization). Also, when substituting the final regression layer by a more sophisticated regression, most of previous networks employed discriminative approaches like random forests [195] or support vector regression [54]. Finally, the vast majority of existing works tries to solve a specific computer vision task which is expressed as a regression problem without focusing onto the regression framework itself. To the best of our knowledge, there are no other methods that attempt to incorporate inverse regression into deep learning for computer vision applications.

Head-pose estimation is an important cue for tasks such as human-robot interaction, computer-human interaction, analysis of human behavior, or driver-assistance systems [122]. The pose is typically expressed by three angles that describe the orientation of the head (looking up or down: pitch, left or right: yaw, and tilting left or right: roll). The estimation of the pose parameters is challenging due to changing illumination conditions, to the background scene, to partial occlusions, and to inter-person and intra-person variabilities.

There are few recent chapters using deep learning to regress the angles that determine the human head pose. The pioneering work of Osadchy et al. [130] synergistically performs face detection and pose estimation by employing a ConvNet to map face images to points on a manifold parameterized by pose, and non-face images to points away from that manifold. In [119], the authors use GoogLeNet [169] and replace the last softmax layer with an Euclidean loss layer that measures the L_2 distance of the prediction from the target. Liu et al. [105] train on synthetic head images and employ a quite simple ConvNet (3 convolutional and 2 fully connected layers; with a linear activation function to predict the head poses in the output layer) to perform head-pose estimation. A quite similar approach can be found in [1], [187] and [141], where slightly different ConvNet architectures are used. Finally, HyperFace [139] is a single ConvNet model (5 convolutional layers along with 3 fully connected layers using the Euclidean loss to train the head-pose estimates) for simultaneous face detection, landmark localization, pose estimation and gender classification. The proposed method is one of the first attempts to use a very deep pre-trained network to effectively tackle the head-pose estimation problem.

4.3 Mixture of Linear Inverse Regressions

In this section we describe in detail the regression layer of the proposed model. We consider a deep neural network $\boldsymbol{\phi}$ with weights $\boldsymbol{w}$ that maps an image $\boldsymbol{i} \in \mathbb{R}^M$ onto a

high-dimensional feature vector $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{i}; \boldsymbol{w}) \in \mathbb{R}^D$. The regression $\boldsymbol{r}^*$, with parameters $\boldsymbol{\theta}^*$, maps $\boldsymbol{x}$ onto a *low-dimensional* target $\boldsymbol{y} = \boldsymbol{r}^*(\boldsymbol{x}; \boldsymbol{\theta}^*) \in \mathbb{R}^L$ with $L \ll D$. The regression layer can be expressed probabilistically in the following way. Let $\boldsymbol{i}$, $\boldsymbol{x}$ and $\boldsymbol{y}$ be realizations of the random variables $\boldsymbol{I}$, $\boldsymbol{X}$ and $\boldsymbol{Y}$. The goal is to estimate the target $\boldsymbol{Y}$ given an input image $\boldsymbol{I}$ and the model parameters $(\boldsymbol{w}, \boldsymbol{\theta}^*)$, i.e. the conditional density $p(\boldsymbol{Y}|\boldsymbol{\phi}(\boldsymbol{I}; \boldsymbol{w}); \boldsymbol{\theta}^*)$. Once this posterior distribution is estimated, one can predict the target corresponding to an input based on the conditional expectation of the target, namely $\hat{\boldsymbol{y}} = \boldsymbol{r}^*(\boldsymbol{\phi}(\boldsymbol{i}; \bar{\boldsymbol{w}}); \boldsymbol{\theta}^*) = \mathbb{E}[\boldsymbol{y}|\boldsymbol{\phi}(\boldsymbol{i}; \bar{\boldsymbol{w}}); \boldsymbol{\theta}^*]$, where $\bar{\boldsymbol{w}}$ denotes the optimized values of the weights.

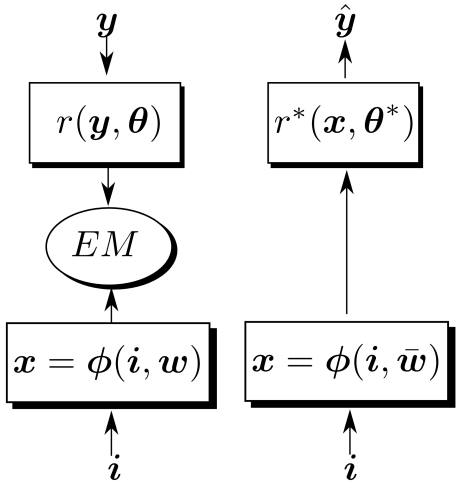
Estimating a regression function defined over a high-dimensional space, as above, is generally difficult because one has to estimate a large number of parameters, typically of the order of D^2. We bypass this difficulty by training an *inverse regression*, *e.g.* Fig. 4.1. More precisely, at training the low-dimensional target $\boldsymbol{Y}$ is the input to the regression model, while the high-dimensional feature vector $\boldsymbol{X}$ is the output. Hence, $\boldsymbol{Y}$ is assumed to lie on a low-dimensional (linear or non-linear) manifold embedded in $\mathbb{R}^D$ and parameterized by $\boldsymbol{X}$. Choosing the low-dimensional variable to be the input implies a smaller number of parameters, typically $L(D + L)$), to be estimated. Hence, the parameters of the *inverse* conditional density are estimated at training, i.e. $p(\boldsymbol{\phi}(\boldsymbol{I}; \boldsymbol{w})|\boldsymbol{Y}; \boldsymbol{\theta})$, from which the *forward* conditional density is then derived and used for prediction, i.e. $p(\boldsymbol{Y}|\boldsymbol{\phi}(\boldsymbol{I}; \bar{\boldsymbol{w}}); \boldsymbol{\theta}^*)$. Such an inverse regression can be implemented with either non-parametric [99] or parametric [37, 132] methods. The advantage of the latter over the former is twofold: (i) the inverse parameters $\boldsymbol{\theta}$ can be estimated in closed-form either with Gaussian mixtures [37] or with mixtures of t-distributions [132], and (ii) the forward parameters $\boldsymbol{\theta}^*$ can be analytically derived from the inverse parameters with both these two mixture models. Moreover, a parametric model allows us to alternate between the optimization of the network weights and of the regression parameters.

We consider the following mixture of K affine regressions:

$$\boldsymbol{X} = \sum_{k=1}^{K} \mathbb{I}(Z = k)(\mathbf{A}_k \boldsymbol{Y} + \boldsymbol{b}_k + \boldsymbol{E}_k), \tag{4.1}$$

where $\mathbb{I}$ is the indicator function, Z is a hidden variable such that $Z = k$ if and only if $\boldsymbol{X}$ is the result of mapping $\boldsymbol{Y}$ using the affine transformation $\mathbf{A}_k \boldsymbol{Y} + \boldsymbol{b}_k$, with $\mathbf{A}_k \in \mathbb{R}^{D \times L}$ and $\boldsymbol{b}_k \in \mathbb{R}^D$, and $\boldsymbol{E}_k \in \mathbb{R}^D$ is an error vector. By marginalization over Z, the joint probability of $\boldsymbol{y}$ and $\boldsymbol{x}$ can be written as $p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}, \boldsymbol{w}) = \sum_{k=1}^{K} p(\boldsymbol{x}|\boldsymbol{y}, Z = k; \boldsymbol{\theta}, \boldsymbol{w}) p(\boldsymbol{y}|Z = k; \boldsymbol{\theta}) p(Z = k; \boldsymbol{\theta})$. Under the assumption that $\boldsymbol{E}_k$ is a zero-mean Gaussian variable with diagonal covariance $\boldsymbol{\Sigma}_k \in \mathbb{R}^{D \times D}$, we obtain that $p(\boldsymbol{x}|\boldsymbol{y}, Z = k; \boldsymbol{\theta}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{x}; \mathbf{A}_k \boldsymbol{y} + \boldsymbol{b}_k, \boldsymbol{\Sigma}_k)$. We further assume that $\boldsymbol{Y}$ follows a mixture of Gaussians. We can now write $p(\boldsymbol{y}|Z = k; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{c}_k, \boldsymbol{\Gamma}_k)$ and $p(Z = k; \boldsymbol{\theta}) = \pi_k$, where $\boldsymbol{c}_k \in \mathbb{R}^L$, $\boldsymbol{\Gamma}_k \in \mathbb{R}^{L \times L}$ and $\sum_{k=1}^{K} \pi_k = 1$.

Altogether, the regression layer is described by the parameter set $\boldsymbol{\theta} = \{\boldsymbol{c}_k, \boldsymbol{\Gamma}_k, \pi_k, \mathbf{A}_k, \boldsymbol{b}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$. Both $\boldsymbol{\theta}$ and $\boldsymbol{w}$ can be estimated via the EM algorithm described in detail in Sec. 4.4. Once optimal values for $\boldsymbol{\theta}$ and $\boldsymbol{w}$ are estimated, namely $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$, the inverse conditional density



(a) Inverse training **(b)** Forward prediction

Figure 4.1: The method proposed in this chapter performs training by gluing inverse regression ($\boldsymbol{r}$ parameterized by $\boldsymbol{\theta}$) and network fine-tuning ($\boldsymbol{\phi}$ parameterized by $\boldsymbol{w}$) in an EM procedure. The parameters $\boldsymbol{\theta}^*$ of the forward regression $\boldsymbol{r}^*$ can be derived analytically from $\boldsymbol{\theta}$, which allows to predict a target $\boldsymbol{y}$ associated with an input $\boldsymbol{i}$.

can be written as:

$$p(\boldsymbol{\phi}(\boldsymbol{i}; \bar{\boldsymbol{w}})|\boldsymbol{y}; \bar{\boldsymbol{\theta}}) = \sum_{k=1}^{K} \bar{\nu}_k \mathcal{N}(\boldsymbol{\phi}(\boldsymbol{i}; \bar{\boldsymbol{w}}); \bar{\mathbf{A}}_k \boldsymbol{y} + \bar{\boldsymbol{b}}_k, \bar{\boldsymbol{\Sigma}}_k), \tag{4.2}$$

with $\bar{\nu}_k = \bar{\pi}_k \mathcal{N}(\boldsymbol{y}; \bar{\boldsymbol{c}}_k, \bar{\boldsymbol{\Gamma}}_k) / \sum_{j=1}^{K} \bar{\pi}_j \mathcal{N}(\boldsymbol{y}; \bar{\boldsymbol{c}}_j \bar{\boldsymbol{\Gamma}}_j)$. The forward predictive distribution can then be expressed as:

$$p(\boldsymbol{y}|\boldsymbol{\phi}(\boldsymbol{i}; \bar{\boldsymbol{w}}); \boldsymbol{\theta}^*) = \sum_{k=1}^{K} \nu_k^* \, \mathcal{N}(\boldsymbol{y}; \mathbf{A}_k^* \boldsymbol{\phi}(\boldsymbol{i}; \bar{\boldsymbol{w}}) + \boldsymbol{b}_k^*, \boldsymbol{\Sigma}_k^*), \tag{4.3}$$

with $\nu_k^* = \pi_k^* \mathcal{N}(\boldsymbol{x}; \boldsymbol{c}_k^*, \boldsymbol{\Gamma}_k^*) / \sum_{j=1}^{K} \pi_j^* \mathcal{N}(\boldsymbol{x}; \boldsymbol{c}_j^* . \boldsymbol{\Gamma}_j^*)$ and with parameters $\boldsymbol{\theta}^* = \{\boldsymbol{c}_k^*, \boldsymbol{\Gamma}_k^*, \pi_k^*, \mathbf{A}_k^*, \boldsymbol{b}_k^*, \boldsymbol{\Sigma}_k^*\}_{k=1}^{K}$. An interesting feature of this model is that the forward parameters $\boldsymbol{\theta}^*$ can be expressed

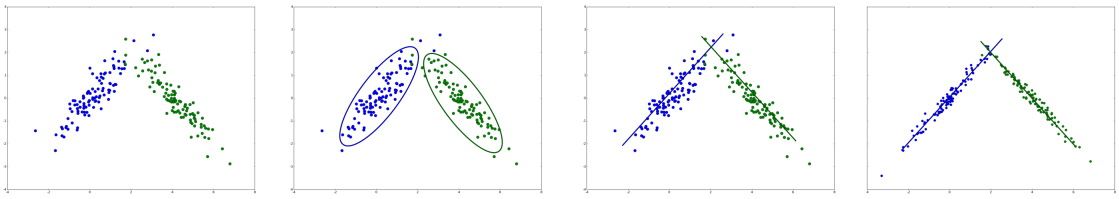


Figure 4.2: Training the deep inverse regression EM with a toy example, *e.g.* $L = 1$, $D = 2$ and $K = 2$. The E-step computes the posteriors $\mu_{nk}^{(i+1)}$. The M-GMM-step fits a mixture to the data given the posteriors. The M-mapping-step estimates the parameters of the affine regressions (the two lines illustrate the projection of the target space onto the feature space). Finally, the M-network-step fine-tunes the network weights via minimization of a mean-square error loss function.

analytically from the inverse parameters $\boldsymbol{\theta}$:

$$\begin{aligned}
\boldsymbol{c}_k^* &= \mathbf{A}_k \boldsymbol{c}_k + \boldsymbol{b}_k, \\
\boldsymbol{\Gamma}_k^* &= \boldsymbol{\Sigma}_k + \mathbf{A}_k \boldsymbol{\Gamma}_k \mathbf{A}_k^\top, \\
\pi_k^* &= \pi_k, \\
\mathbf{A}_k^* &= \boldsymbol{\Sigma}_k^* \mathbf{A}_k^\top \boldsymbol{\Sigma}_k^{-1}, \\
\boldsymbol{b}_k^* &= \boldsymbol{\Sigma}_k^* (\boldsymbol{\Gamma}_k^{-1} \boldsymbol{c}_k - \mathbf{A}_k^\top \boldsymbol{\Sigma}_k^{-1} \boldsymbol{b}_k), \\
\boldsymbol{\Sigma}_k^* &= (\boldsymbol{\Gamma}_k^{-1} + \mathbf{A}_k^\top \boldsymbol{\Sigma}_k^{-1} \mathbf{A}_k)^{-1}.
\end{aligned}$$

As a consequence, one can use the conditional expectation associated with (4.3) to predict a target:

$$\hat{\boldsymbol{y}} = \boldsymbol{r}^*(\boldsymbol{\phi}(\boldsymbol{i}; \bar{\boldsymbol{w}}); \boldsymbol{\theta}^*) = \sum_{k=1}^{K} \nu_k^* (\mathbf{A}_k^* \boldsymbol{\phi}(\boldsymbol{i}; \bar{\boldsymbol{w}}) + \boldsymbol{b}_k^*). \tag{4.4}$$

4.4 Training the Proposed Model

In this section we describe the estimation of the model parameters $\boldsymbol{\theta}$ and $\boldsymbol{w}$ based on an expectation-maximization algorithm and using a training dataset $\{\boldsymbol{i}_n, \boldsymbol{y}_n\}_{n=1}^N$, i.e. Alg. 1. Fig. 4.2 shows the proposed training applied to a toy example.

The E-step updates the posterior probabilities with the following expression:

$$\begin{aligned}
\mu_{nk}^{(i+1)} =& p(Z_n = k | \boldsymbol{y}_n, \boldsymbol{x}_n; \boldsymbol{\theta}^{(i)}, \boldsymbol{w}^{(i)}) \\
=& \frac{\pi_k^{(i)} p(\boldsymbol{y}_n, \boldsymbol{x}_n | Z_n = k; \boldsymbol{\theta}^{(i)}, \boldsymbol{w}^{(i)})}{\sum_{j=1}^K \pi_j^{(i)} p(\boldsymbol{y}_n, \boldsymbol{x}_n | Z_n = j; \boldsymbol{\theta}^{(i)}, \boldsymbol{w}^{(i)})}
\end{aligned} \tag{4.5}$$

with:

$$\begin{aligned}
p(\boldsymbol{y}_n, \boldsymbol{x}_n | Z_n = k; \boldsymbol{\theta}, \boldsymbol{w}) =& p(\boldsymbol{x}_n | \boldsymbol{y}_n, Z_n = k; \boldsymbol{\theta}, \boldsymbol{w}) \\
& \times p(\boldsymbol{y}_n | Z_n = k; \boldsymbol{\theta})
\end{aligned} \tag{4.6}$$

Data: Training dataset $(\boldsymbol{i}, \boldsymbol{y})_{n=1:N}^{N}$, number of components K, and convergence threshold $\epsilon \in \mathbb{R}$;
Result: $\boldsymbol{\theta}$ and $\boldsymbol{w}$;
Initialize $\boldsymbol{\theta}^{(0)}$ and $\boldsymbol{w}^{(0)}$;
while $||\boldsymbol{\theta}^{(i+1)}$-$\boldsymbol{\theta}^{(i)}|| > \epsilon$ **do**
  E-step: Update the posteriors $\boldsymbol{\mu}^{(i+1)} = \{\mu_{nk}^{(i+1)}\}_{n=1,k=1}^{N,K}$ given the current parameters $\boldsymbol{\theta}^{(i)}$ and weights $\boldsymbol{w}^{(i)}$.
  M-GMM-step: Update the mixture parameters $\{\boldsymbol{c}_k^{(i+1)}, \boldsymbol{\Gamma}_k^{(i+1)}, \pi_k^{(i+1)}, \}_{k=1}^{K}$ given the posteriors $\boldsymbol{\mu}^{(i+1)}$ and the current mapping parameters and the current network weights;
  M-Mapping-step: Update the affine parameters $\{\mathbf{A}_k^{(i+1)}, \boldsymbol{b}_k^{(i+1)}, \boldsymbol{\Sigma}_k^{(i+1)}\}_{k=1}^{K}$ given the posteriors $\boldsymbol{\mu}^{(i+1)}$, the mixture parameters and the current network weights, and
  M-Network-step: Update the weights $\boldsymbol{w}^{(i+1)}$ given the posteriors $\boldsymbol{\mu}^{(i+1)}$ and the current parameter values $\boldsymbol{\theta}^{(i+1)}$.
end

Algorithm 1: EM algorithm for deep inverse regression.

The M-step performs the following maximization:

$$(\boldsymbol{\theta}^{(i+1)}, \boldsymbol{w}^{(i+1)}) = \underset{(\boldsymbol{\theta},\boldsymbol{w})}{\operatorname{argmax}}\ \mathbb{E}\big[\log\ p((\boldsymbol{x}, \boldsymbol{y}, Z)_{1:N}; \boldsymbol{\theta}, \boldsymbol{w})|(\boldsymbol{y}, \boldsymbol{i})_{1:N}; \boldsymbol{\theta}^{(i)}, \boldsymbol{w}^{(i)}\big] \tag{4.7}$$

This is further decomposed in three sub-steps: M-GMM-step, M-Mapping-step, and the M-Network-step, i.e. Alg. 1. The update formulae for the parameters $\boldsymbol{\theta}$ can be found in [37]. The network's weights are estimated as follows. By developing the expected complete-data log-likelihood (4.7) and after keeping the terms that depend on $\boldsymbol{w}$, we obtain the following loss function:

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}) &= \sum_{n=1}^{N}\sum_{k=1}^{K} p(Z_n = k|(\boldsymbol{y}_n, \boldsymbol{i}_n)) \\ &\quad \times \log\ p(\boldsymbol{\phi}(\boldsymbol{i}_n, \boldsymbol{w})|\boldsymbol{y}_n, Z_n = k; \boldsymbol{\theta}) \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} \mu_{nk} \log\ p(\boldsymbol{\phi}(\boldsymbol{i}_n, \boldsymbol{w})|\boldsymbol{y}_n, Z_n = k; \boldsymbol{\theta}) \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} \mu_{nk} \log\ \mathcal{N}(\boldsymbol{\phi}(\boldsymbol{i}_n, \boldsymbol{w}); \mathbf{A}_k\boldsymbol{y}_n + \boldsymbol{b}_k, \boldsymbol{\Sigma}_k)\end{aligned} \tag{4.8}$$

If we further assume that the error covariances are isotropic, i.e. $\boldsymbol{\Sigma}_k = \lambda_k^{-1}\mathbb{I}$ where $\lambda_k \in \mathbb{R} > 0$ is the precision associated with each affine transformation, we obtain the

following loss function:

$$\mathcal{L}(\boldsymbol{w}) = \sum_{n=1}^{N}\sum_{k=1}^{K} \mu_{nk}\lambda_k ||\mathbf{A}_k \boldsymbol{y}_n + \boldsymbol{b}_k - \boldsymbol{\phi}(\boldsymbol{i}_n, \boldsymbol{w})||_2^2 \tag{4.9}$$

This loss function has the form of a weighted mean-squared error and hence gradient descent techniques for deep neural network optimization are well suited and can be easily used [61]. Notice however that gradient stability issues are common. In particular deep regression can be difficult to train when the target space is unbounded because it is likely to lead to exploding gradient problems [12]. As a consequence, the targets $\boldsymbol{x}_n$ may reach really high values after a few EM iterations. To avoid this problem we use a normalization layer [76]. Moreover this layer avoids converging to the undesirable solution where $\mathbf{A}_k = 0$ and $\boldsymbol{b}_k = 0$ that would maximize the likelihood.

The proposed EM algorithm is initialized as follows. We first perform clustering in the target space using a standard procedure, i.e. K-means with random initializations followed by fitting a GMM. This yields initial values for the posteriors, namely $\mu_{nk}^{(0)}$. Notice however from (4.6) that the posteriors depend on feature clustering as well and this clustering is not reliable at the start of the algorithm. For this reason, we freeze the E-step (the posteriors are set to their initial values) and perform a few iterations of the M steps, which amounts to alternate between updating the regression parameters and tuning the network weights.

Finally, it is worth mentioning that the proposed inverse regression can be used with a single Gaussian distribution, i.e. $K = 1$. In this case (4.1) reduces to $\boldsymbol{X} = \mathbf{A}\boldsymbol{Y} + \boldsymbol{b} + \boldsymbol{E}$ where, again, $\boldsymbol{E}$ is a zero-mean Gaussian variable with diagonal covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{D\times D}$, hence $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}|\mathbf{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{\Sigma})$. Notice that it is still interesting to train a low-dimensional to high-dimensional mapping (inverse regression), on the following grounds. The low-to-high regression that we propose to train provides D linear constraints with $D \times (L + 2)$ free parameters. Hence, one needs a minimum $L + 2$ image-target training pairs to estimate the model parameters. In contrast, high-to-low regression would have provided L linear constraints for training, with $L \times (D + 2)$ free parameters; $D + 2$ image-target pairs would have been at least necessary for training.

Because there is no assignment variable in the case of a single Gaussian, the training procedure alternates between estimating the regression parameters $\mathbf{A}, \boldsymbol{b}$ and $\boldsymbol{\Sigma}$, and updating the network weights $\boldsymbol{w}$, i.e. M-step iterations of Alg. 1.

4.5 EXPERIMENTS

In this section we first describe the datasets used to evaluate the performance of our model. After that, we present the ConvNet architecture employed and the results obtained.

Datasets. The *Biwi Kinect head-pose dataset* [49] consists of over $15K$ images including video recordings of 20 people (16 men, 4 women, some of them recorded twice) using

a Kinect camera. During the recordings, the participants freely move their head and the corresponding head angles lie in the intervals $[-60°, 60°]$ (pitch), $[-75°, 75°]$ (yaw), and $[-20°, 20°]$ (roll). Fig. 4.4 shows examples of the synthetic images generated.

Figure 4.3: Example frames of the *Biwi head-pose dataset*.

We employed the following protocol to create a fair data partition: we run Support Vector Regression (SVR) [160] on HOG features [33] using an 8-fold cross-validation (21 randomly selected videos for training and the remaining 3 videos for test). After that, we ordered the performance on each fold in terms of their MSE and, finally, we kept the best performing fold for the HOG-based methods and the median performing fold for the VGG-based methods. We acted in this manner to give some advantage to the most simple approaches and to avoid a bias towards our deep learning proposal. In other words, HOG-based methods are trained and tested on the most advantageous fold for them. It is important to notice that we used 20% of the training set as validation set, and that no person appears both in training and test sets.

The main drawback of the Biwi dataset is that most of faces are looking squarely or present small angles. So, the distribution of the targets is almost Gaussian. Since we suspected this property would favor models with a low number of Gaussians, we evaluated our proposal with different target distributions. To do so, we created a synthetic dataset utilizing the MakeHuman 3D software[2] to generate 50 different body models. Parameters like age, gender or color skin were randomly selected by the software. Then, we randomly generated $100K$ images of the models' head with uniformly distributed angles. To ensure robustness during training, each generated image has a randomly selected lighting position and color for the OpenGL engine lighting system. Fig. 4.4 shows examples of the synthetic images generated.

Figure 4.4: Example frames of our synthetic head-pose dataset.

We generated two smaller datasets (approximately $20K$ images per dataset) from this

[2]www.makehuman.org

uniformly sampled dataset. First, we selected the images in order to obtain a mixture of two Gaussians centered around $(35°, 60°, 0°)$ and $(-35°, -60°, 0°)$. This dataset is referred to as *S2G dataset (Synthetic with 2 Gaussians)*, and is used to study the impact of the target distribution on the number of Gaussians employed (K). Secondly, we removed some images from the uniformly distributed dataset with a Gaussian of mean $(0°, 0°, 0°)$. This dataset is referred to as *SSV dataset (Synthetic with only Side Views)*. Since the distribution of poses in the SSV dataset is not directly obtained by combining Gaussian distributions, it can be considered as a difficult case for our model in which we could think that K=1 would not necessarily perform well.

ConvNet architectures. In practice, it is relatively difficult to train an entire ConvNet from scratch because it requires a sufficiently large dataset. Furthermore, if the network is very deep, an important amount of computational power would be necessary. A common alternative approach consists in taking a network already trained on ImageNet and use its weights as initialization to train your own ConvNet. In this chapter, we use VGG-16 [157]. However, since these networks have been trained to solve a classification task, they have learned to be invariant to the pose of objects. On the contrary, we want our model to be independent of the object but highly dependent on the pose. To face this problem, we use the initialization procedure explained in section 4.4.

The size of the last fully-connected layer of VGG-16 pre-trained on ImageNet is relatively large (4096) as it is designed to recognize approximately 1000 objects. However, in our case we predict only three angles. Thus, we can reduce this dimension in order to reduce the number of parameters in the network and hence the computation burden. To do so, we add a fully connected layer of size 512 with a linear activation function and initialize it with the eigenvectors of a PCA trained on the output of the network. This layer is added before the batch normalizer. This solution presents the advantage that back propagation can be easily performed through this layer.

In practice, we do not exactly iterate between E and M steps as described in Alg.1. We first alternate between the E step and the two M-GMM and M-Mapping steps. When convergence has been reached, we apply the M-Network step. This procedure has two advantages. First, the network weights are not modified before having good mapping functions. Secondly, the $\boldsymbol{\theta}$ updates are performed with the CPU whereas the M-Network utilizes the GPU. The computation is faster if we do not alternate too often between CPU and GPU operations.

Comparison Between Regression Models. We show the results from a series of experiments in order to illustrate the effectiveness of the proposed model. In Table 4.1, we compare 7 different regression models:

- *$(0°, 0°, 0°)$:* In order to have a reference about the learning ability of our model we introduce this mean pose estimator, i.e. a fictitious method that always return $(0°, 0°, 0°)$ as predicted angles.

- *HOG-SVR:* An SVR is trained using the HOG representation of the input images. The HOG features used in this chapter were extracted following the same strategy as in [42], i.e. a HOG pyramid (p-HOG) by stacking HOG descriptors at multiple resolutions and providing a feature vector of dimension 1888.
- *HOG-IR:* An inverse regression (IR) approach equivalent to the model described in [37] with a mixture of 50 affine mappings is trained using HOG features. In other words, *HOG-IR* is equivalent to our proposal in the case K=50, without the M-network step, and using HOG features instead of a deep network.
- *VGG-SVR:* We remove the softmax layer of VGG-16 trained on ImageNet, and we train an SVR to predict the head pose angles from the network features.
- *VGG-IR:* In this case, after removing the softmax layer of the pre-trained VGG-16, our inverse regression model is trained on the output of this network without performing the M-network step and K=50.
- *VGG-FCL-FT:* We replace the softmax layer of a pre-trained VGG-16 by a fully connected layer (FCL) of 3 units and a linear activation function. This layer is trained with a loss function that measures the L_2 distance of the prediction from the target. This model is a *vanilla deep regression model* commonly used in the literature. To draw a fair comparison, we trained this network with different optimizers and kept the best result. We obtained this result with SGD optimizer, a learning rate of 10^{-3} and a learning decay of 0.5 every 3 epochs. We fine-tune (FT) for 3 epochs only the last layer and then the last four layers. The loss is similar to the commonly employed in the literature [78, 119, 166, 174]. It is important to mention that the results reported here are obtained with the use of a batch normalizer before the regression layer as we obtained poor results without it.
- *VGG-IR-FT:* This is our proposal. The results displayed in Table 4.1 correspond to the best performing number of Gaussians (K=2), as shown in Table 4.2.

	Pitch	Yaw	Roll	Mean
$(0°, 0°, 0°)$	23.92	28.50	8.48	20.30
HOG-SVR	8.50	6.45	5.04	6.66
HOG-IR	6.39	5.69	4.77	5.62
VGG-SVR	10.60	19.34	7.79	12.57
VGG-IR	15.26	26.79	10.76	17.60
VGG-FCL-FT	5.65	4.44	**2.93**	4.34
VGG-IR-FT (proposed)	**4.68**	**3.12**	3.07	**3.62**

Table 4.1: Comparison of different methods on the Biwi head-pose database. Mean absolute errors are given in degrees. The best results are highlighted in bold.

First, we notice that, with HOG features, the inverse regression model outperforms SVR (a forward regression). However, the results obtained using the deep features given by the pre-trained VGG (*VGG-SVR* and *VGG-IR*) are somewhat disappointing, since the

error obtained is much worse than the one obtained by HOG-based methods. This could illustrate the previously mentioned pose-invariance property of the pre-trained VGG-16. These results also show that deep features can hardly be used for head-pose estimation without fine-tuning. In fact, if we fine-tune the network we obtain comparable results to state-of-the-art. The proposed method improves the result by 0.71 degrees with respect to *VGG-FCL-FT*.

	K=1	K=2
Biwi	5.12/3.45/3.39	**4.68**/**3.12**/**3.07**
S2G	2.87/3.54/**1.65**	**2.53**/3.36/1.97
SSV	3.12/**3.75**/2.01	2.86/3.83/2.04
	K=5	K=10
Biwi	5.55/3.74/4.01	5.93/3.45/3.99
S2G	2.63/**3.33**/1.94	2.89/3.35/2.06
SSV	2.77/4.12/**1.95**	**2.74**/3.91/2.18

Table 4.2: Mean absolute error in degrees for the Pitch/Yaw/Roll angles on three datasets using *VGG-IR-FT* and different K values. The best results per angle and dataset are highlighted in bold.

In Table 4.2, we study how the performance of our proposal (*VGG-IR-FT*) evolves as we increase K. We also study how this performance is affected by the type of dataset employed. From the results obtained one can hardly draw a definitive conclusion. On real data, K=2 provides the best performance. However, on cases specifically designed to make fail the model employing K=1 (*S2G* and *SSV*), we obtain comparable results independently of the value of K. This behavior can be explained by the following reasons. First, the difficulty of the synthetic datasets is not enough to make fail the model with K=1. Second, as we can see in (4.5), the updates of the EM algorithm do not depend only on the target distribution but also rely on the existing clusters in feature space. So, the optimal K cannot be established only by looking at the target distribution. In particular, in *S2G* the model seems to favor the pre-existing clustering in feature space over the mixture of two Gaussians in the target space (as would happen if the best performing model was K=2). In practical terms, one reasonable solution would be to use K=1 by default, since it reduces the complexity of the approach and at the same time provides sufficiently good results. It confirms that deep neural networks are effective linearizers and therefore adding a nonlinearity in the regression layer does not help very much. The benefit of the proposed model comes mainly from the inverse formulation.

Head-Pose Estimation State-of-the-Art Comparison. We compare the performance of our proposal with state-of-the-art methods on head-pose estimation.

As can be seen in Table 4.3, our proposal *VGG-IR-FT* outperforms state-of-the-art approaches. Drouard et al.[42], in an extension of their previous work [41], employ 4 partially-latent variables in their mixture of linear regression approach to establish the state-of-the-art. Moreover, we can even compete with methods using additional information (see the last four methods in Table 4.3). Deep learning is not used neither in [49] nor

	Pitch	Yaw	Roll	Mean
Methods using only RGB				
Liu et al.[105]	6.1	6.0	5.7	5.94
Mukherjee et al.[119]	5.18	5.67	/	5.43
Drouard et al.[42]	5.43	4.24	4.13	4.60
VGG-IR-FT (proposed)	**4.68**	**3.12**	**3.07**	**3.62**
Methods using additional information				
*Wang et al.[178]***	8.5	8.8	7.4	8.23
*Mukherjee et al.[119]***	4.76	5.32	/	5.04
*Fanelli et al.[49]***	**3.8**	**3.5**	5.4	4.23
*Liu et al.[105]**	4.5	4.3	**2.4**	**3.73**

Table 4.3: Comparison of different methods on the Biwi head-pose database. Mean absolute errors are given in degrees. The last four methods use additional or slightly different data (*extra annotation used for training, **3D depth data used). The best results are highlighted in bold.

in [178], and both use depth information. Importantly, our approach provides a competitive performance even in absence of this additional information. In [105], Liu et al. train using synthetic data because otherwise they get results comparable to the ones provided by HOG-based methods. Their experience confirms the intuition that a forward regression technique does not perform well without using a large dataset. Among all competitor methods the only one using a very deep network is Mukherjee et al. [119]. They use a GoogLeNet architecture on both RGB and depth images. The superiority of *VGG-IR-FT* can indicate again the benefits of using inverse regression. Finally, an important remark is that all these results could even be further improved by including temporal information, $e.g.$ the temporally stable head-pose estimation proposed by [1].

In order to compare the sensibility to the training set size, we randomly down-sample the Biwi database training set. We show the results in Table 4.4. We can notice that even employing only 40% of data we are competitive with most of the methods in Table 4.3. The performance seems to scale linearly with respect to the available training set size. This trend seems to suggest that additional data would further improve the performance.

	Pitch	Yaw	Roll	Mean
20%	9.36	7.00	6.19	7.55
40%	6.2	5.00	5.14	5.45
60%	6.33	4.33	4.18	4.95
80%	5.49	3.77	4.12	4.46
100%	4.68	3.12	3.07	3.62

Table 4.4: Influence on the mean absolute error of amount of training data employed as percentage of total number of training examples in the Biwi dataset.

4.6 CONCLUSIONS

In this chapter, we proposed the coupling of a Gaussian mixture of linear inverse regressions with a ConvNet. We describe the methodological foundations and the associated algorithm to jointly train the deep network and the regression function, and we evaluated our model on the problem of head-pose estimation. From an experimental point of view, our contribution can be summarized as follows. First, we show that the proposed inverse regression model outperforms L_2-based regression models used by most of the state-of-the-art computer vision methods, at least in the case of head-pose estimation. Second, our method works effectively on relatively small training datasets, without the need of incorporating additional data, as it is often proposed in the literature. Lastly, our proposal outperforms state-of-the-art methods in head-pose estimation testing on the most widely used head-pose dataset. To the best of our knowledge, we are the first to propose an inverse regression approach to train a deep network. As future work, we plan to test our method on other computer vision problems, like facial keypoint detection or full body pose estimation, and extend the type of distributions used in our mixtures, as for example t-distributions to make the model more robust to outliers.

CHAPTER 5

DEEPGUM: DEEP ROBUST REGRESSION WITH GAUSSIAN-UNIFORM MIXTURES

5.1 INTRODUCTION

In the previous chapter, we presented an deep inverse regression model that outperforms *vanilla regression* models currently used in head-pose estimation. A *vanilla deep regression* model is defined as a ConvNet where the softmax layer is replaced by a fully connected regression layer with linear or sigmoid activations, and L_2 is used to measure the distance between the prediction and the annotation. However, one of the known prominent limitations of the L_2 loss is its strong sensitivity to outliers, potentially leading to poor generalization performance [9, 74].

This chapter delves into how to soundly mitigate the influence of outliers when employing deep neural architectures, ConvNets in particular, for regression. More precisely, we investigate the use of a general-purpose and principled methodology specifically designed to cope with the two most common acceptations of outlier: (i) a sample that lies at an abnormal distance from the rest of training samples, and (ii) a wrongly annotated training sample. On the one hand, abnormal samples are proper to any regression problem and they may heavily bias the learned regression model. In order to avoid this undesirable situation, a recent study [9] proposed an alternative regression loss, based on the Tukey's biweight function, able to reduce the influence of this kind of outliers during training. On the other hand, data-hungry deep learning architectures require large collections of data. Since the resources (*i.e.* computational power for automatic measurements or human effort for manual assessments) required to obtain clean annotations for all these data are not always available, wrongly annotated data become more and more common, thus further justifying the development of robust regression strategies.

In this chapter, we present a generic framework that combines the representation power of ConvNets with the principled methodology of probabilistic methods for outlier rejec-

tion (see Figure 5.1). We propose to use a Gaussian-Uniform Mixture (GUM) as the last layer of a ConvNet front-end, and refer to this combination as DeepGUM.[1] Intuitively, the probabilistic model hypothesizes a Gaussian distribution of the errors for the inliers and a uniform distribution for the outliers. We combine the Expectation-Maximization (EM) algorithm to unsupervisedly detect the outliers, with stochastic gradient descent to robustly estimate the parameters of the ConvNet. As a consequence, the deep network weights are updated mainly with clean, rather than contaminated information. Overall, the proposed model is able to isolate the outliers, independently of their nature, thus learning a deep regression network robust to extreme training samples and wrong annotations. We demonstrate the effectiveness of the proposed approach in four different computer vision regression tasks: facial and fashion landmarks detection, age estimation and head pose estimation.

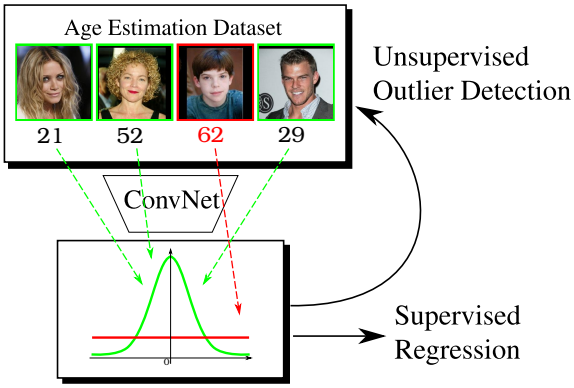


Figure 5.1: DeepGUM in a nutshell: the Gaussian-uniform mixture performs unsupervised outlier detection to down-weight the impact of outliers (in red) in the ConvNet loss for supervised deep robust regression.

5.2 Related Work

Robust regression has been a long studied topic in statistics [74, 109, 146] and has been applied to many computer vision problems [14, 112, 164]. Roughly speaking, robust regression methods have a high *breakdown point*, which is the smallest amount of outlier contamination that an estimator can handle before yielding poor results. Prominent examples of such robust regression models are the least trimmed squares, the Theil-Sen estimator or heavy-tailed distributions [57]. Several robust training strategies for artificial neural networks are also available [10, 126].

The four most popular methodologies for robust statistics are: M-estimators, sampling methods, trimming methods and robust clustering. M-estimators [74] minimize the sum of a positive-definite function of the residuals: such function must reduce the influence of large residual values. The minimization is carried over weighted least squares techniques, whose convergence is not proved for most of the M-estimators. Sampling methods [112], such as least-median-of-squares or random sample consensus (RANSAC), estimate the

[1]The code of DeepGUM will be made publicly available upon the acceptance of the corresponding submitted paper

model parameters by solving a system of equations defined for a randomly chosen data subset. The main drawback of sampling methods is that they require a complex data sampling procedure and it is tedious to use them for estimating a large number of parameters. Trimming methods [146] rank the residuals and down-weight the data points associated to large residuals. They are typically cast into a (non-linear) weighted least squares optimization problem, where the weights are modified at each iteration, leading to iteratively reweighted least squares problems. Robust statistics have also been addressed in the framework of mixture models and a number of robust Gaussian clustering methods were proposed, such as Gaussian mixtures with a uniform noise component [7, 28], heavy-tailed distributions [52], and trimmed likelihood estimators [53, 127]. Importantly, it has been reported that the use of a uniform component exhibits superior performance to other models [28].

Recently, deep learning exhibited impressive performance in (non-robust) regression, overcoming traditional approaches in different applications [119, 139, 166, 174]. In parallel, deep learning-based approaches for robust classification appeared: these approaches usually combine probabilistic models with ConvNets. For instance, [8] assumes that the observed labels were created from the true labels by passing through a noisy channel whose parameters are unknown. A probabilistic model for the transformation from true labels to noisy labels is defined, and the associated EM algorithm is derived. Xiao *et. al.* [183] propose a probabilistic model to exploit the relationships between classes, images and noisy labels for large-scale image classification. Their framework requires the existence of a dataset annotation with clean and noisy labels, as well as an extra dataset annotated with the type of noise of each sample, thus making the method difficult to use in real-world scenarios. Classification algorithms based on a distillation process to learn from noisy data have also been recently presented [104].

Despite classical studies on robust regression, the importance of regression in computer vision, and the successful attempts of combining deep learning architectures with probabilistic models for robust classification, the exploitation of probabilistic models for robust deep regression remains largely unexplored. More precisely, to the best of our knowledge, only one work relates robust statistics and deep regression. Indeed, Belagiannis *et. al.* [9] propose a regression model with ConvNets that achieves robustness to outliers (defined as extreme values in the target space or samples rarely encountered in the training data) by minimizing the Tukey's biweight function.

We pursue this line of research by proposing the use of a Gaussian-uniform mixture on top of a ConvNet. The main hypothesis is that the error of the inliers follows a Gaussian distribution while the error of the outliers follows a uniform distribution. In this way, our model is able to robustly estimate the variance of the error of the inliers as in [9]. Differently from [9], our model also provides an inliner/outlier detection mechanism thanks to the *a posteriori* probability of each training sample. More precisely, in [9], the amount of outliers is dynamic as the residuals are scaled with the median absolute deviation, but a fixed tuning constant determines the residuals that are not going to be backpropagated (because they correspond to a zero gradient). Conversely, DeepGUM allows to automatically estimate the amount and spread of outliers without defining *a priori* any hyperparameter

whatsoever that controls the inlier/outlier threshold.

5.3 Robust Regression with DeepGUM

We assume that the error of the inliers follows a Gaussian distribution, while the error of the outliers follows a uniform distribution. In order to properly define the probabilistic model, we first set the basic notation. Let $\boldsymbol{i} \in \mathbb{R}^M$ and $\boldsymbol{y} \in \mathbb{R}^D$ be respectively the input image and the output vector of corresponding dimensions M (#pixels times #channels) and D ($D \ll M$). Let $\boldsymbol{\phi}$ denote a deep neural network with weights $\boldsymbol{w}$ that maps an input vector $\boldsymbol{i}$ of dimension M onto an output vector $\boldsymbol{\phi}(\boldsymbol{i}, \boldsymbol{w})$ of lower dimension D.

We aim to train a model using the inliers and ignoring the outliers. However, we cannot assume any *a priori* information on how the training samples are split between inliers and outliers. Therefore, we define a hidden binary random variable z valued 0 when the associated training sample is an outlier and 1 otherwise. More formally, given $\boldsymbol{i}$, the probability of $\boldsymbol{y}$ follows a GUM:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{i}; \boldsymbol{\theta}, \boldsymbol{w}) = p(z=0)\, \mathcal{U}(\boldsymbol{y}; \gamma) \\ + p(z=1)\, \mathcal{N}(\boldsymbol{y}; \boldsymbol{\phi}(\boldsymbol{i}; \boldsymbol{w}), \boldsymbol{\Sigma}). \end{aligned} \tag{5.1}$$

The prior distribution on z is a binomial with (unknown) parameter $\pi = p(z=1)$, γ is the normalization constant of the uniform distribution, and $\boldsymbol{\Sigma}$ is the covariance matrix of the multivariate Gaussian distribution. $\boldsymbol{\theta} = \{\pi, \gamma, \boldsymbol{\Sigma}\}$ is the set of parameters of GUM. At training time, we must estimate the GUM parameters, $\boldsymbol{\theta}$, as well as $\boldsymbol{w}$.

The learning stage employs a combination of the EM algorithm and the back-propagation stochastic gradient descent (SGD). Because the EM and SGD algorithms are typically used in unsupervised and supervised scenarios, this combination is a perfect match to the intrinsic nature of robust regression problems. Indeed, while the EM algorithm focuses on addressing the unsupervised problem of outlier detection, the SGD algorithm is devoted to learn the inlier regression in a fully supervised fashion. In other words, given a dataset consisting of N image-vector pairs $\{\boldsymbol{i}_n, \boldsymbol{y}_n\}_{n=1}^N$, the EM algorithm identifies the outliers given the current regression, and the regression model is updated to improve the quality of the regression of the inliers.

5.3.1 Unsupervised Outlier Detection through EM

In a general manner, at iteration i, we need to update all the parameters as well as detect the outliers. While the former corresponds to the M-step, the latter corresponds to the E-step. Indeed, given an estimate of the network and mixture parameters $\boldsymbol{w}^{(i)}$ and $\boldsymbol{\theta}^{(i)}$, the *a posteriori* probability that the n-th pair $(\boldsymbol{i}_n, \boldsymbol{y}_n)$ is an inlier writes:

$$r_n^{(i+1)} = \frac{\pi^{(i)} \mathcal{N}(\boldsymbol{y}_n; \boldsymbol{\phi}(\boldsymbol{i}_n, \boldsymbol{w}^{(i)}), \boldsymbol{\Sigma}^{(i)})}{\pi^{(i)} \mathcal{N}(\boldsymbol{y}_n; \boldsymbol{\phi}(\boldsymbol{i}_n, \boldsymbol{w}^{(i)}), \boldsymbol{\Sigma}^{(i)}) + (1-\pi^{(i)})\gamma^{(i)}}. \tag{5.2}$$

This can be easily alternated with the estimation of the parameters of the mixture model, θ. The covariance matrix is estimated with the usual weighted external product:

$$\boldsymbol{\Sigma}^{(i+1)} = \sum_{n=1}^{N} r_n^{(i+1)} \boldsymbol{\delta}_n^{(i)} \boldsymbol{\delta}_n^{(i)\top}, \tag{5.3}$$

where $\boldsymbol{\delta}_n^{(i)} = \boldsymbol{y}_n - \boldsymbol{\phi}(\boldsymbol{i}_n; \boldsymbol{w}^{(i)})$. As for γ, it is related to the first and second order centered moments C_1 and C_2 by:

$$\frac{1}{\gamma^{(i+1)}} = \prod_{d=1}^{D} 2\sqrt{3\left(C_{d2}^{(i+1)} - \left(C_{d1}^{(i+1)}\right)^2\right)}, \tag{5.4}$$

where the two centered moments write:

$$C_{d1}^{(i+1)} = \sum_{n=1}^{N} \frac{(1 - r_n^{(i+1)})}{N - N_{i+1}} \delta_{nd}^{(i)}, \tag{5.5}$$

$$C_{d2}^{(i+1)} = \sum_{n=1}^{N} \frac{(1 - r_n^{(i+1)})}{N - N_{i+1}} \left(\delta_{nd}^{(i)}\right)^2, \tag{5.6}$$

with $N_{i+1} = \sum_{n=1}^{N} r_n^{(i+1)}$ and $\delta_{nd}^{(i)}$ denoting the d-th dimension of the vector $\boldsymbol{\delta}_n^{(i)}$. The update of the parameter of the binomial distribution writes:

$$\pi^{(i+1)} = N_{i+1}/N. \tag{5.7}$$

One prominent advantage of DeepGUM when predicting multidimensional outputs is its flexibility to easily handle the granularity of the outliers. For instance, when it comes to estimate the 2D locations of a set of landmarks in an image, we may want the model to remove from the training set only a few landmarks per image and not the whole image. To do so, we could employ one mixture per landmark and thus, one r_n per landmark. This *landmark-wise* r_n induces $D/2$ covariances of size 2, where D is the dimensionality of the target space. On the other hand, a *component-wise* r_n could be used if we intend to consider each dimension of every particular training sample as a potential outlier. In this case, we would have D single value covariances. Finally, one could use an *image-wise* r_n, where an outlier would be an image in its entirety. This flexibility represents an attractive property of DeepGUM, since similar previous approaches, as [9], use only a component-wise strategy.

5.3.2 Supervised Inlier Deep Regression

The parameters of the GUM, $\boldsymbol{\theta}$, are updated as described in the previous section, and this allows us to detect outliers accordingly to the new distribution. These updated outlier detections are used not only to estimate $\boldsymbol{\theta}$, but also to update the ConvNet parameters predominantly using the data associated to the inliers. We thus modify the L_2 regression

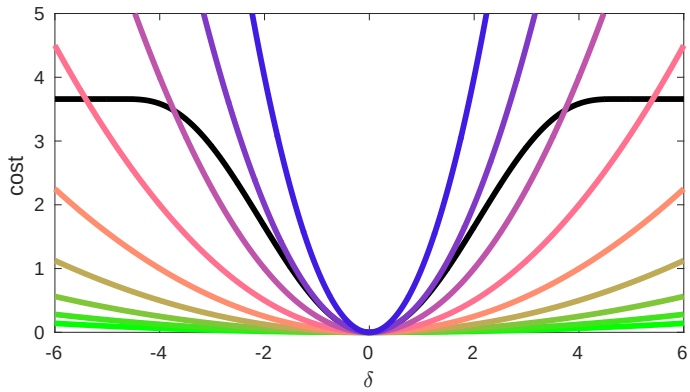


Figure 5.2: Cost of the proposed weighted loss (in color) compared to Tukey's biweight function (in black). Different colors represent different values of r_n, exponentially decreasing from 1 (blue curve, corresponds to classical L_2) to 2^{-8} (green curve). Small values of r_n clearly mitigate the effect of the outlier during the learning of $\boldsymbol{w}$.

loss, and weight it with the *a posteriori* probability of each datum belonging to the inlier class:

$$\mathcal{L}_{\text{GUM}} = \sum_{n=1}^{N} r_n^{(i+1)} ||\boldsymbol{y}_n - \boldsymbol{\phi}(\boldsymbol{i}_n; \boldsymbol{w})||_2^2. \tag{5.8}$$

Intuitively, only the error associated to the inlier data is back-propagated through the network. Consequently, the parameters $\boldsymbol{w}$ are updated only with clean, rather than contaminated information. Figure 5.2 plots the weighted loss as a function of a one-dimensional residual δ for different values of r_n, and compares it to the (fixed) biweight function used in [9]. The effect of r_n clearly mitigates the influence of the outliers during the learning of $\boldsymbol{w}$. One could argue that when the level of data corruption is moderated, the effect of the stochasticity inherent to the usual gradient descent strategies could be larger than the effect of the corrupted data. However, when the amount of wrong labels or outlier data points is large, the use of all this corrupted information could potentially lead to adverse effects during learning.

5.3.3 DEEPGUM TRAINING

In order to train the proposed model, we assume the existence of a training and a validation sets, denoted respectively by $\mathcal{T} = \{\boldsymbol{i}_n^{\text{T}}, \boldsymbol{y}_n^{\text{T}}\}_{n=1}^{N_{\text{T}}}$ and $\mathcal{V} = \{\boldsymbol{i}_n^{\text{V}}, \boldsymbol{y}_n^{\text{V}}\}_{n=1}^{N_{\text{V}}}$. The test set $\mathcal{E} = \{\boldsymbol{i}_n^{\text{E}}, \boldsymbol{y}_n^{\text{E}}\}_{n=1}^{N_{\text{E}}}$ is devoted only to the evaluation of the model once trained. Therefore, neither the input images $\boldsymbol{i}_n^{\text{E}}$ nor the output labels $\boldsymbol{y}_n^{\text{E}}$ are used during training.

The training of DeepGUM alternates between the unsupervised EM introduced in section 5.3.1 and the supervised stochastic gradient descent of section 5.3.2. In details, during the first step and given the network parameters, the EM iterates to update the inlier/outlier detection as well as the mixture parameters $\boldsymbol{\theta}$ until the mixture parameters do not evolve anymore. The second step consists in estimating the network parameters using SGD given the obtained r_n's, so as to minimize (5.8) on $\mathcal{T}$. We iterate the SGD algorithm until the

input: Training set $\mathcal{T} = (\boldsymbol{i}_n^{\mathrm{T}}, \boldsymbol{y}_n^{\mathrm{T}})_{n=1}^{N_{\mathrm{T}}}$, validation set $\mathcal{V} = \{\boldsymbol{i}_n^{\mathrm{V}}, \boldsymbol{y}_n^{\mathrm{V}}\}_{n=1}^{N_{\mathrm{V}}}$ and a convergence threshold $\epsilon > 0$.
initialization: Run the gradient descent algorithm on $\mathcal{T}$ to minimize (5.8) with all r_n's set to 1, until the convergence criterion on $\mathcal{V}$ is reached.
repeat
  EM algorithm: unsupervised outlier detection
  repeat
    Update the r_n's with (5.2).
    Update the mixture parameters with (5.3), (5.4) and (5.7).
  until The parameters $\boldsymbol{\theta}$ are stable.
  SGD: learn the deep inlier regression network
  repeat
    SGD to minimize $\mathcal{L}_{\mathrm{GUM}}$ in (5.8).
  until $\mathcal{L}_{\mathrm{GUM}}$ grows on $\mathcal{V}$.
until $\mathcal{L}_{\mathrm{GUM}}$ grows on $\mathcal{V}$.

Algorithm 2: DeepGUM training procedure

relative loss on the validation set with respect to the previous K epochs is not lower than $\epsilon > 0$. The overall procedure is initialized by running SGD with all samples considered to be inliers $r_n = 1, \forall n$ and stops when the $\mathcal{L}_{\mathrm{GUM}}$ does not decrease after iterating between the EM and SGD algorithms. The summary is shown in Algorithm 2. It is important to notice that we do not need to constrain the model to avoid the trivial solution where all the points are considered as outliers. This is because after the first M-network step, the network can clearly distinguish the two components. In an imaginary pathological case where DeepGUM would consider all points as outliers, the algorithm would stop after the first SGD step and output the initial regression (corresponding to the initialization step of Algorithm 2).

Now that the overall training procedure for DeepGUM has been presented, it is worth discussing three important properties of it. First, the loss minimized during the SGD step, *i.e.* (5.8) is insensitive to the estimated covariance matrix $\boldsymbol{\Sigma}$. The rationale behind this choice is the following: since $\boldsymbol{\Sigma}$ measures "how big is the inlier error for each component of $\boldsymbol{y}$ given the previous weights $\boldsymbol{w}$," it does not make sense to weight the loss by $\boldsymbol{\Sigma}$, since it would bias the network to improve on what it is already good at. Therefore we opt to train the network encouraging equal effort for all the components of $\boldsymbol{y}_n$, but not for all training samples thanks to the use of r_n. Preliminary experiments supported this intuition. Second, the estimation of γ, apart from being a methodological novelty *per se*, it is also crucial for the performance of DeepGUM. Indeed, as opposed to classical robust statistics studies using hand-crafted/static features, the features used in GUM, $\boldsymbol{\phi}(\boldsymbol{i}_n, \boldsymbol{w})$, evolve with the iterations thanks to the SGD training of $\boldsymbol{w}$. Intuitively, the network will improve its regression capabilities reducing the variance of the inliers, and probably of the outliers as well. Neglecting to reestimate the the parameters γ will result in limiting the outlier rejection power of DeepGUM. Third, we argue that DeepGUM allows to solve regression tasks with per-sample learning rates. Indeed, a possible interpretation of $\mathcal{L}_{\mathrm{GUM}}$

is that the r_n's multiply the learning rate associated to observation n, since the gradient is simply multiplied by r_n. In this regard, DeepGUM can be seen as a principled way to automatically choose the right learning rate separately for each of the training samples.

5.4 EXPERIMENTS

This experimental section has two main purposes. On the one hand, we empirically validate DeepGUM in three real-world datasets that are naturally corrupted with outliers. More precisely, experiments are performed on the problems of fashion landmark detection (section 5.4.1), age estimation (section 5.4.2) and head pose estimation (section 5.4.3). On the other hand, we delve into the robustness of DeepGUM under controlled conditions and compare it to other deep regression methods in order to have a better understanding of its behavior. To this aim, we artificially generate outliers for the task of facial landmark detection (see section 5.4.4), progressively corrupting larger and larger percentages of annotation.

Common Baselines We systematically compare DeepGUM to the classical L_2 loss and to the only robust deep regression method in the literature [9]. This is done for all the aforementioned tasks and in all experimental setups. In addition, we compare to the state-of-the-art in each specific task. These extra baselines are different for each problem and are specified below. For the common baselines, we use VGG-16 [157] pre-trained on ImageNet [148] with a mini-batch of size 128 and learning rate set to 10^{-4} (we fine-tuned the last convolutional block and the fully connected layers). Early stopping with a patience of 5 epochs is employed. The data is augmented using mirroring. Importantly, unlike [9], we did not report results on LSP and Parse because we observed that robust regression ([9] and DeepGUM), when initializing the weights with those pretrained on ImageNet, do not help significantly on these datasets. Moreover, the performance we obtained with the standard L_2 loss was 73% and 74% strict PCP on Parse and LSP respectively, much better than the architecture of [9] with L_2 loss (61% and 60%). We thus judged unfair to compare architectures with such different results. Importantly, in all our experiments the covariances are diagonal, since we empirically concluded that it was not worth employing more complex models.

In this chapter, we re-implemented the complete method of [9] where the MAD value is multiplied by 7 during the first 50 iterations. We noticed that this heuristic rule does not generalize to other datasets. For the sake of fairness, in all our experiments we run 3 epochs with L_2 loss before exploiting the robust regression scheme of [9] (or ours). We observed that computing MAD after each epoch on the entire dataset is beneficial, however the output of the normalization did not have much impact most probably due to the use of MAD.

5.4.1 FASHION LANDMARK DETECTION

Set up Visual fashion analysis presents a wide spectrum of applications such as clothe recognition, retrieval, and recommendation. We employ the fashion landmark dataset

Table 5.1: Mean absolute error on the upper-body subset of FLD, per landmark and in average. The landmarks are left (L) and right (R) collar (C), sleeve (S) and hem (H).

Method	Upper-body landmarks						
	LC	RC	LS	RS	LH	RH	Avg.
L_2	12.08	12.08	18.87	18.91	16.47	16.40	15.80
Biweight [9]	17.41	18.45	24.15	26.06	24.66	25.54	22.71
DFA [107] (L_2)	15.90	15.90	30.02	29.12	23.07	22.85	22.85
DFA [107] (5 VGG)	10.75	10.75	20.38	19.93	15.90	16.12	15.23
DeepGUM	11.97	11.99	18.59	18.50	16.44	16.29	15.63

Table 5.2: Mean absolute error on the full body and lower-body subsets of FLD, per landmark and in average. The landmarks are left (L) and right (R) collar (C), sleeve (S), hem (H) and trouser leg (T). DFA [107] does not report on these two subsets.

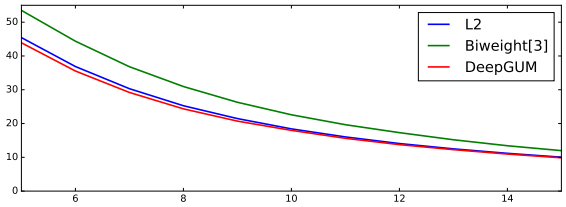
Method	Full body landmarks								
	LC	RC	LS	RS	LH	RH	LT	RT	Avg.
L_2	8.69	8.78	15.65	15.89	10.84	10.88	12.11	12.25	11.89
Biweight [9]	11.56	11.73	20.58	20.29	14.36	14.06	14.24	14.10	15.11
DeepGUM	8.62	8.68	15.42	15.59	10.76	10.84	11.96	11.97	11.73

Method	Lower-body landmarks				
	LH	RH	LT	RT	Avg.
L_2	12.50	12.51	13.28	13.19	12.87
Biweight [9]	15.75	15.77	18.00	17.96	16.87
DeepGUM	12.19	12.23	12.80	12.81	12.51

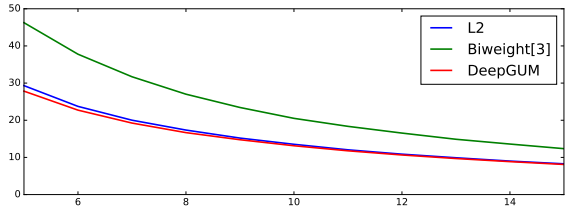
(FLD) [106] that includes more than $120K$ images, where each image is labeled with eight landmarks. The dataset is equally divided in three subsets: upper-body clothes (6 landmarks), full-body clothes (8 landmarks) and lower-body clothes (4 landmarks). We randomly split each subset of the dataset into test ($5K$), validation ($5K$) and training ($\sim 40K$). Two metrics are used: the mean absolute error (MAE) of the landmark localization and the percentage of failures (landmarks detected further from the ground truth than a given threshold). We employ *landmark-wise* r_n.

Figure 5.3 displays the failure rate on fashion landmark detection on the different subsets as a function of the distance threshold. DeepGUM reports a slight improvement over classical L_2 regression and a major improvement over Tukey's Biweight function. Table 5.2 displays results on two more subsets of the fashion landmark detection dataset (full-body and lower-body landmarks).

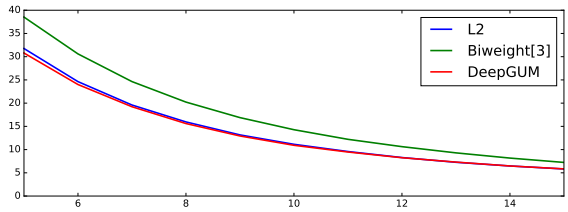
Discussion Table 5.1 reports the results obtained on the upper-body subset of the fashion landmark dataset (additional results on full-body and lower-body subsets are included in the supplementary material). We report the mean average error (in pixels) for each landmark individually, and the overall average (last column). While for the first subset we can compare with the very recent results reported in [107], for the other subsets we compare with the common baselines. Generally speaking, we outperform all other baselines in



(a) Upper-body



(b) Full body



(c) Lower-body

Figure 5.3: Failure rate on fashion landmark detection as a function of the distance threshold.

average, but also in each of the individual landmarks for all the three subsets. The only exception is the comparison against the method utilizing five VGG pipelines to estimate the position of the landmarks. Although this method reports slightly better performance than DeepGUM, we recall that we are using one single VGG as front-end, and therefore the representation power cannot be the same as the one associated to a pipeline employing five VGG's trained for tasks such as pose estimation and clothe classification that clearly aid the fashion landmark estimation task.

Interestingly, DeepGUM yields slightly better results than L_2 regression and a major improvement over Biweight [9]. This behavior is also observed when computing the failure rate shown in supplementary materials and seems counter-intuitive at first glance. Our assumption is that [9] converges towards a solution where too many points are considered as outliers, thus reducing the diversity of the training set. To validate this assumption, we measured the amount of points classified as outliers (those with a gradient equal to zero) and we obtained $\approx 10\%$ for all folds. This situation has not been encountered in our experiments with synthetic noise (see section 5.4.4), and illustrates the higher robustness of DeepGUM when facing a more complex problem and noise distribution. Figure 5.4 displays the landmark locations proposed by DeepGUM for few images in FLD. The upper right and the lower left corners of this figure show wrong predictions.

Figure 5.4: Visual results of fashion landmark detection obtain with the proposed DeepGUM model.

Table 5.3: Mean absolute error (in years) on CACD.

Method	MAE
L_2	5.75
Biweight [9]	5.55
Dex [145]	5.25
DexGUM	5.14
DeepGUM	**5.08**

5.4.2 Age Estimation

Set Up Age estimation from a single face image is an important task in computer vision with applications in access control and human-computer interaction. This task is closely related to the prediction of other biometric and facial attributes, such as gender, ethnicity, and hair color. We use the cross-age celebrity dataset (CACD) [21] that contains 163446 images from 2000 celebrities. The images are collected from search engines using the celebrity's name and desired year (from 2004 to 2013). The dataset splits into 3 parts, 1800 celebrities are used for training, 80 for validation and 120 for testing. The validation and test sets are manually cleaned whereas the training set is noisy. In our experiments, we report results using *image-wise* r_n.

Apart from DeepGUM and the common baselines (L_2 and Biweight [9]), we also compare to the age estimation method based on deep expectation (Dex) [145], which was the winner of the Looking at People 2015 challenge. This method uses the VGG-16 architecture and poses the age estimation problem as a deep classification problem followed by a softmax expected value refinement. We report results with two different approaches using Dex. First, our implementation of the original Dex model. Second, we add the GUM model on top the the Dex architecture; we termed this architecture DexGUM.

Figure 5.5: Images considered as outliers by DeepGUM during training. The annotation is displayed below each image.

Discussion Table 5.3 reports the results obtained on the CACD test set for age estimation. We report the mean absolute error (in years) for five different methods. We can easily observe that DeepGUM exhibits the best results: 5 years of MAE (0.7 years better than L_2). Importantly, the architectures using GUM (DeepGUM followed by DexGUM) are the ones offering a better performance. This is further supported by the histogram of the error included in the supplementary materials. DeepGUM considered that 7% of images were outliers and thus these images were undervalued during training. Figure 5.5, where some of the images classified as outliers are displayed, illustrates the ability of DeepGUM to detect outliers. Since the dataset was automatically annotated, it is prone to corrupted annotations. Indeed, the age of each celebrity is automatically annotated by subtracting the date of birth from the picture time-stamp. Intuitively, this procedure is problematic since it assumes that the automatically collected and annotated images show the right celebrity and that the times-tamp and date of birth are correct. Our experimental evaluation clearly demonstrates the benefit of a robust regression technique to operate on datasets populated with outliers.

In Figure 5.6, we display the histogram of the absolute error obtained with different methods. We can see the importance of using DeepGUM to reduce the number of large errors.

5.4.3 Head Pose Estimation

Set Up The McGill real-world face video dataset [38] consists of 60 videos (a single participant per video, 31 women and 29 men) recorded with the goal of studying unconstrained face classification. The videos were recorded in both indoor and outdoor environments under different illumination conditions and participants move freely. Consequently, some frames suffer from important occlusions. Annotations for head pose are provided for the yaw angle only. The yaw angle ranges from $-90°$ to $90°$. The annotation

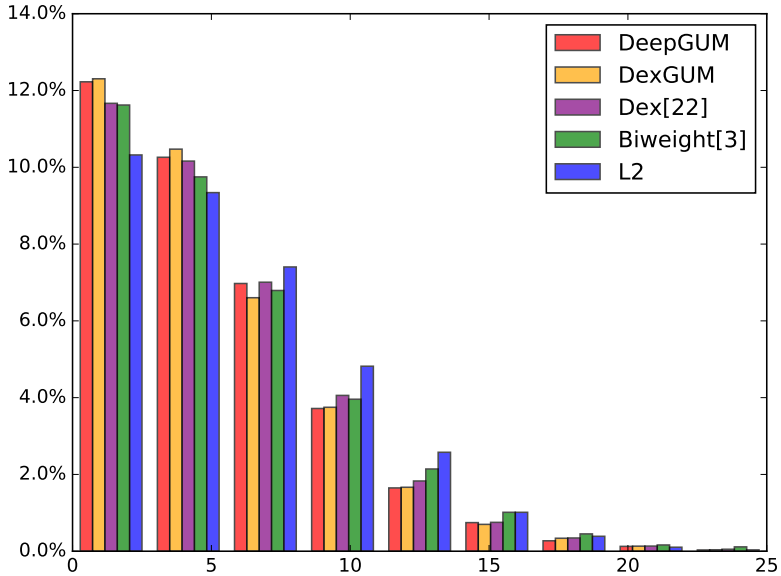


Figure 5.6: Histogram of the absolute error obtained with the different methods tested on the CACD dataset.

Table 5.4: Errors on the McGill dataset. *Uses extra data

Method	MAE	RMSE
Xiong et al. [185]	-	$29.81 \pm 7.73^*$
Zhu and Ramanan [196]	-	$35.70 \pm 7.48^*$
Demirkus et al. [38]	-	$12.41 \pm 1.60^*$
Drouard et al. [45]	12.22 ± 6.42	23.00 ± 9.42
L_2	8.60 ± 1.18	12.03 ± 1.66
Biweight [9]	7.81 ± 1.31	11.56 ± 1.95
DeepGUM	$\mathbf{7.61 \pm 1.00}$	$\mathbf{11.37 \pm 1.34}$

for each frame was estimated using a two-step labeling procedure that, first, automatically provides the most probable angle as well as a degree of confidence, and then the final label is chosen by a human annotator among the plausible angle values. Since the resulting annotations are not perfect it makes this dataset suitable to compare robust regression models. As the training and test sets are not separated in the original dataset, we perform a 7-fold cross-validation. Importantly, only a subset of the dataset is publicly available (35 videos over 60).

Discussion In Table 5.4, we report the results obtained with different methods and employ an asterisk to indicate when a particular method uses the entire dataset (60 videos). We can easily notice that DeepGUM exhibits the best results compared to the other ConvNets methods (respectively $0.99°$ and $0.20°$ lower than L_2 and Biweight in MAE). The last three approaches, all using deep architectures, significantly outperform the current state-of-the-art approach [45].

In Figure 5.7, we display the error obtained on one fold of the training set. It visually justifies the choice of a Gaussian-Uniform model for the error distribution.

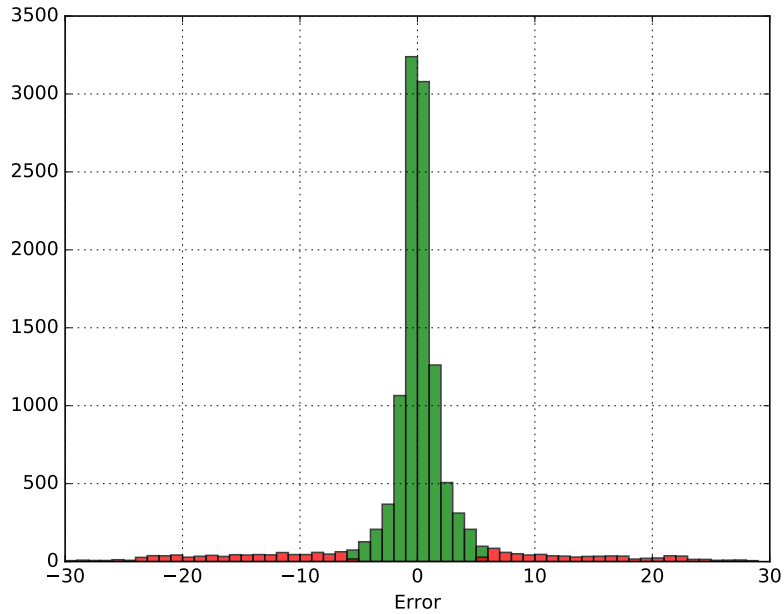


Figure 5.7: Error histogram on the McGill Dataset. Points that are considered as outliers are displayed in red ($r_n < 0.5$) and inliers are displayed in green ($r_n \geq 0.5$)

5.4.4 Facial Landmark Detection

Set up We perform experiments on the LFW and NET facial landmark detection datasets [166] that consist of 5590 and 7876 face images, respectively. We combined both datasets and employed the same data partition as in [166]. Each face is labeled with the positions of five key-points in Cartesian coordinates, namely left and right eye, nose, and left and right corners of the mouth. The detection error is measured with the Euclidean distance between the estimated and the ground truth position of the landmark, divided by the width of the face image, as in [166]. The performance is measured with the failure rate of each landmark, where errors larger than 5% are counted as failures. The two aforementioned datasets can be considered as outlier-free since the average failure rate reported in the literature falls below 1%. Therefore, we artificially modify the annotations of the datasets for facial landmark detection to find the breakdown point of DeepGUM. Our purpose is to study the robustness of the proposed deep mixture model to outliers generated in controlled conditions. To do so, we use three different types of outliers:

- Normally Generated Outliers (*NGO*): A percentage of landmarks is selected, regardless of whether they belong to the same image or not, and shifted a distance of d pixels in a uniformly chosen random direction. The distance d follows a Gaussian distribution, $\mathcal{N}(25, 2)$. *NGO* simulates errors produced by human annotators that made a mistake when clicking, thus annotating the landmark in a slightly wrong location.
- Local - Uniformly Generated Outliers (*l-UGO*): It follows the same philosophy as *NGO*, sampling the distance d from a uniform distribution over the image, instead

of a Gaussian. Such errors simulate human errors that are not related to the human precision, such as not selecting the point or misunderstanding the image.

- Global - Uniformly Generated Outliers (*g-UGO*): As in the previous case, the landmarks are corrupted with uniform noise. However, in *g-UGO* the landmarks to be corrupted are grouped by image. In other words, we do not corrupt a subset of all landmarks regardless of the image they belong to, but rather corrupt all landmarks of a subset of the images. This strategy simulates problems with the annotation files or in the sensors in case of automatic annotation.

The first and the last types of outlier contamination employ *landmark-wise* r_n, while the second uses *image-wise* r_n.

Discussion The three plots in Figure 5.8 report the failure rate of the common baselines together with DeepGUM for the three aforementioned types of synthetic perturbation as a function of the percentage of outliers. The dashed lines report the precision and recall of Biweight (in green) and DeepGUM (in red). Precision represents the percentage of training samples classified as outliers that are true outliers; while recall represents the percentage of outliers that are classified as such. The first conclusions that can be drawn directly from this figure are that, on the one hand, Biweight systematically presents a lower recall than DeepGUM (that is, DeepGUM is a more reliable tool than Biweight to identify and, therefore, ignore outliers during training). And, on the other hand, DeepGUM tends to present a better failure rate than Biweight in most of the scenarios contemplated.

The first two plots, *l-UGO* and *g-UGO*, show the performance under outliers corrupted with uniform noise. We can clearly observe that, while for limited amounts of outliers (i.e. $< 10\%$) all methods report comparable performance, DeepGUM is clearly superior to L_2 and to Biweight [9] for larger amounts of outliers. We can also safely identify a breakdown point of DeepGUM on *l-UGO* at $\sim 40\%$. To further analysis the behavior of the baselines, we computed the precision and recall for the outlier detection task. While for Biweight both decrease when increasing the number of outliers, these measures are constantly around 99% for DeepGUM (before the breakdown point for *l-UGO*).

In the last one, *NGO*, the corrupted annotation is always around the ground truth, leading to a failure rate smaller than 7% for all methods. We can see that the three methods exhibit comparable performance up to 30% of outliers. Beyond that threshold, Biweight outperforms the other two methods in spite of presenting a progressively lower recall and a high precision (i.e. Biweight has a hard time to correctly identify outliers, but those that it identifies as outliers they are true outliers). When training DeepGUM, the second M-Network step does not lead to any improvement on the validation set, therefore the algorithm stops and returns the network we obtained at the initialization. Such a network corresponds to the network we obtain using the standard L_2 loss. We can conclude that the strategy of DeepGUM, consisting in removing all points detected as outliers, is not effective in this particular experiment. In other words, having more noisy data is better than having only few clean data in this particular case of 0-mean highly correlated noise. Nevertheless, we consider an attractive property of DeepGUM the fact that it can automatically identify these particular cases and return an acceptable solution.

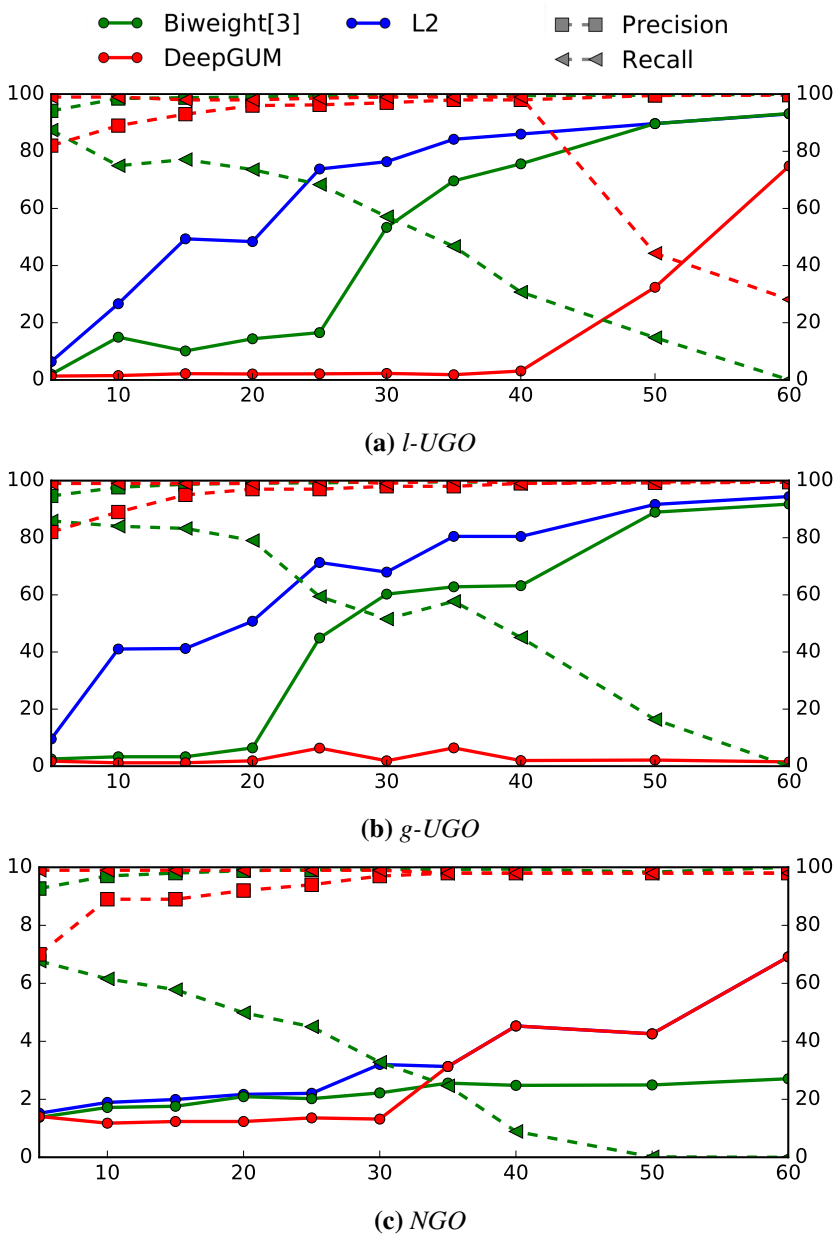


Figure 5.8: Evolution of the failure rate (left y-axis) when augmenting the noise for the 3 types of outliers considered. We also display the corresponding precisions and recalls in percentage (right y-axis) for the outlier class.

5.5 Conclusions

This chapter introduces DeepGUM: a Gaussian-uniform mixture for deep robust regression. The novelty of the proposal resides in combining graphical models for robust regression with deep learning in a jointly trainable fashion. In this context, previous studies only deal with the classical L_2 loss and Tukey's Biweight function, an M-estimator robust to outliers that is the state-of-the-art on robust deep regression [9]. Our proposal yields a better performance than both approaches by proposing a principled methodology, and the

derived optimization procedure, that alternates between solving the unsupervised task of outlier detection with the supervised task of inlier deep regression. The experimental validation addresses four different tasks: facial and fashion landmarks detection, and age and head pose estimation from face images. We have empirically shown that (1) DeepGUM is a robust deep regression approach that does not need to rigidly specify *a priori* the distribution (number and spread) of outliers, (2) DeepGUM exhibits a higher breakdown point than competitor methods when the outliers are sampled from a uniform distribution (being able to deal with more than 50% of outlier contamination without providing incorrect results), and (3) DeepGUM is capable of providing comparable or better results than current state-of-the-art approaches in the four aforementioned tasks. Finally, DeepGUM could be easily used to investigate either problems in the annotations or highly unusual training samples from automatically collected massive datasets, that are cleaned using tedious, error-prone, and time-consuming manual human supervision.

CHAPTER 6

A COMPREHENSIVE ANALYSIS OF DEEP REGRESSION

6.1 INTRODUCTION

As mentionned in previous chapters, regression techniques are widely employed to solve tasks where the goal is to predict continuous values. In computer vision, regression techniques span a large ensemble of applicative scenarios such as: head-pose estimation [50, 196], facial landmark detection [16, 34] and age estimation [63, 186]

Besides classification, ConvNets are also used to solve regression problems. In section 4.1, a *vanilla deep regression* network is defined as a ConvNet architecture where the last softmax layer used in classification tasks is replaced with a fully connected regression layer with linear or sigmoid activations. L_2 norm is generally used to measure the distance between the prediction and the annotation and, consequently, the model is trained via its minimization. Many *vanilla deep regression* network were proposed, obtaining state-of-the-art results in classical vision regression problems such as like human pose estimation [9, 174] or facial landmark detection [166]. In both cases, the goal is to estimate the location of distinctive features within the input image. *Vanilla deep regressors* are often improved using a cascade approach [9, 166, 174] that consists in applying regressors iteratively to refine the prediction. Similarly, a multiple stage regression model is proposed in [107] in order to estimate the positions of functional keypoints of fashion items, such as neckline or hemline corners. Deep regression models are also used for head-pose estimation [93, 105], where the angles that describe the egocentric orientation of the head are predicted.

A classification formulation could also be employed to solve regression tasks [142, 145]. In that case, the output space is generally discretized in order to obtain class labels, and a multi-class loss is minimized. However, this approach suffers from important drawbacks. First, there is an inherent trade-off between the complexity of the optimization problem and the accuracy of the method due to the grid discretization. Second, in absence

of a specific formulation, a confusion between two classes has the same cost independently of the corresponding error in the target space. Last, the discretization method that is employed must be designed specifically for each task, and therefore general-purpose methodologies must be used with care. In this study, we focus on generic vanilla deep regressions that escape, not only to the definition of a task-dependent trade-off between complexity and accuracy, but also to a penalization loss that may not take the spatial proximity of the labels into account.

Thanks to the empirical success of deep neural architectures, much of the scientific work currently presented in the computer vision community exploits their representation power. Unfortunately, the immense majority of these works present neither a statistical evaluation of the performance nor a rigorous justification of the methodological choices. The main consequence of this lack of systematic evaluation is that researchers proceed by trial-and-error experimentation because the scientific evidence behind the superiority of newly introduced techniques is neither sufficiently clear nor statistically significant.

There are a few studies devoted to the extensive and/or systematic evaluation of deep architectures, focusing on different aspects. For instance, seminal papers exploring efficient back-propagation strategies [98] and evaluating ConvNets for visual recognition [124] were already published 20 years ago. Some articles provide general guidance and understanding on appropriate architectural choices [77, 159, 170] or on gradient-based training strategies for deep architectures [11], while others try to delve into the differences in performance of the variants of a specific (recurrent) model [62] or the differences in performance of several models applied to a specific problem [19, 115]. Researchers also devised automatic ways to overcome the problem of choosing the optimal network architecture [153, 184]. Overall, there is very little guidance on the plethora of design choices and hyper-parameter settings for deep learning architectures, let alone ConvNets for regression problems.

In summary, the revision of prior art shows, first, the absence of a systematic evaluation of deep learning advances in regression. Second, an overabundance of papers based on deep learning (for instance, ≈ 2000 papers were uploaded to ArXiV only in March 2017, in the categories related to machine learning, computer vision and pattern recognition, computation and language, and neural and evolutionary computing). This highlights again the importance of serious comparative empirical studies to discern which are the key blocks in deep regression. Finally, in our opinion, there is a generalized (and scientifically unjustified) absence of statistical tests and confidence intervals in the experimental sections of the immense majority of works published in the field. The execution of a single run per benchmarked method and the succinct description of the preprocessing strategies used also limit the reliability of the results included in the papers and their reproducibility.

Even if some authors devote time and efforts to make their research reproduceable [15], many published studies lack of implementation, practical and –really often– data preprocessing details [9, 105, 107, 166]. One possible reason to explain this lack of details in the paper may be the unavailability of widely-used systematic comparative procedures to highlight which are the most important details.

We propose to fill this gap with a systematic evaluation and a statistical analysis of the performance of vanilla deep regression networks for computer vision tasks. In order to design such a study we take inspiration from two recent papers in the literature. Mishkin et al. [115] present an extensive evaluation of ConvNets on ImageNet (thus for image categorization), including the type of non-linearity, pooling variants, network width, classifier design, image pre-processing, and learning parameters. In that paper, no statistical analysis accompanies the results and the reader is unable to understand the statistical significance (if any) of the differences in performance.

Secondly, Greff et al. [62] discuss the differences in performance of several variants of the long-short term memory recurrent network for categorical sequence modeling, thus addressing classification of sequential data. Importantly, each variant is run several times, statistical tests are employed to compare each variant with a baseline architecture, and the performance is discussed over box-plots offering some sort of graphical confidence intervals of the different benchmarked methods.

In this chapter, we investigate the impact of various design and technical choices on the performance of deep neural architectures when used to addressed classical regression problems in computer vision, namely: head-pose estimation, facial landmark detection and full body pose estimation. Next section discusses the experimental protocol (data sets, base architecture), including the statistical tests, used to benchmark the different choices, grouped in three categories: those related to *network optimization*, *network architecture* and *data pre-processing*. Sections 6.3, 6.5 and 6.6 present respectively the alternatives studied within each category and the associated comparison. An overall discussion is reported in Section 6.8, before exploring how these simple deep regressions are positioned with respect to the state-of-the-art in Section 6.7. Conclusions are drawn in Section 6.9.

6.2 Experimental Protocol

In this section, we describe the protocol adopted to evaluate the influence of different architectures and their variants, training strategies, and data pre-processing methods. We run the experiments using two common base architectures (see section 6.2.1), on three classic computer vision regression problems (see Section 6.2.2).

6.2.1 Base Architectures

We choose to perform our study using two architectures that are among the most commonly referred in the recent literature: VGG-16 [158] and ResNet-50 [66]. The VGG-16 model was termed *Model D* in [158]. We preferred VGG-16 to AlexNet [86] because it performs significantly better on ImageNet and has inspired several network architectures for various tasks [107, 140]. ResNet-50 performs even better than VGG-16 with shorter training time, which is nice in general, and in particular for benchmarking.

VGG-16 is composed of 5 blocks containing two or three convolution layers and a max pooling layer. Let CB^i denote the ith convolution block (see [158] for the details on the

number of layers and of units per layer, as well as the convolution parameters). Let FC^i denote the ith fully connected layer with a dropout rate of $50\,\%$. Fl denotes a flatten layer that transforms a 2D feature map into a vector and SM denotes a soft-max layer. With these notations, VGG-16 can be written as $CB^1 - CB^2 - CB^3 - CB^4 - CB^5 - Fl - FC^1 - FC^2 - SM$.

The main novelty of ResNet-50 is the use of identity shortcuts, which augment the network depth and reduce the number of parameters. We remark that all the convolutional blocks of ResNet-50, except for the first one, have identity connections, making them *residual* convolutional blocks. However, since we do not modify the original ResNet-50 structure, we denote them CB as well. In addition, GAP denotes a global average pooling layer. According to this notation, ResNet-50 architecture can be described as follows: $CB^1 - CB^2 - CB^3 - CB^4 - CB^5 - GAP - SM$.

Both networks are initialized by training on ImageNet for classification, as it is usually done. We then remove the last soft-max layer SM, employed in the context of classification, and we replace it with a fully connected layer with linear activations equal in number to the dimension of the target space. Therefore this last layer is a regression layer, denoted REG, whose output dimension corresponds to the one of the target space of the problem at hand.

6.2.2 Data Sets

In order to perform the empirical comparison, we choose three highly challenging and classic computer vision problems, namely: head-pose estimation, facial landmark detection and human body pose estimation. For each of this problems we select a widely used data set (see below). The data sets are chosen so as to favor diversity in output dimension, pre-processing and data-augmentation requirements.

The **Biwi** head-pose data set [49] consists of over $15,000$ RGB-D images corresponding to video recordings of 20 people (16 men and 4 women, some of them are recorded twice) using a Kinect camera. It is one of the most widely used data set for head-pose estimation [43, 93, 105, 119, 178]. During the recordings, the participants freely move their head and the corresponding head orientations lie in the intervals $[-60°, 60°]$ (pitch), $[-75°, 75°]$ (yaw), and $[-20°, 20°]$ (roll). Unfortunately, it would be too time consuming to perform cross-validation in the context of our statistical study. Consequently, we employed the split used in [93]. Importantly, none of the participants appears in both the training and test sets.

We also use the LFW and NET facial landmark detection (**FLD**) data sets [166] that consist of 5590 and 7876 face images, respectively. We combined both data sets and employed the same data partition as in [166]. Each face/image is labeled with the pixel coordinates of five key-points, namely left and right eyes, nose, and left and right mouth corners.

Finally, we use the **Parse** data set [138] that is a standard data set used for human pose estimation. It is a relatively small data set as it contains only 305 images. Therefore, this

data set challenges very deep architectures and requires a data augmentation procedure. Each image is annotated with the pixel coordinates of 14 joints. In addition, given the limited size of Parse, we consider all possible image rotations in the interval $[12°, -12°]$ with a step of $0.5°$. This procedure is applied only to the training images.

6.2.3 Computational Environment

Our experiments ran on a Nvidia TITAN X (Pascal generation) GPU with Keras 1.1.1 (on the Theano 0.9.0 backend). The code and the results of each individual run are available online.[1] We remark that our computational environment is *academic*, thus far from being populated with tens of GPUs. This environment is used to compare the many possible choices studied in this paper. In total, we summarize the results of more than 600 experimental runs (≈64 days of GPU time). At a first stage we focuse on selecting the right technique to optimize the network.

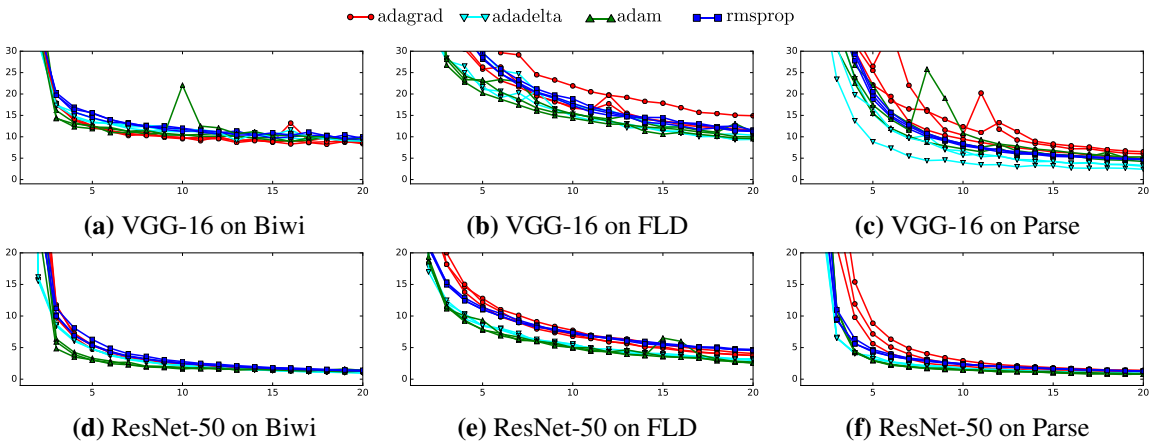
6.3 Network Optimization

Optimizers for training neural networks are responsible for finding the free parameters θ (habitually termed weights) of a cost function $J(\theta)$ that, typically, includes a performance measure evaluated on the training set and additional regularization terms. Such a cost (also called loss) function lets us quantify the quality of any particular set of weights. In this paper, network optimization and network fine-tuning are used as synonym expressions, since we always start from ImageNet pre-trained weights when fine-tuning on a particular new task. The positive impact of pre-training in deep learning has been extensively studied and demonstrated in the literature [48, 67, 171, 191].

Gradient descent, a first-order iterative optimization algorithm for finding the minimum of a function, is the most common and established method for optimizing neural network loss functions [60]. There are other methods to train neural networks, from derivative-free optimization [113, 190] to second-order methods [60], but their use is much less widespread, so they have been excluded from this paper. Attending to the number of training examples used to evaluate the gradient of the loss function, we can distinguish between batch gradient descent (that employs the entire training set in each iteration), mini-batch gradient descent (that uses several examples in each iteration), and stochastic gradient descent (also called sometimes on-line gradient descent, that employs a single example in each iteration). In this chapter, and as is common practice in the deep learning community, we use a mini-batch gradient descent whose batch size is selected through the preliminary experimentation described in section 6.3.2.

Many improvements on the basic gradient descent algorithm have been proposed. In particular, the need to set a learning rate (step size) has been recognized as crucial and problematic: setting this parameter too high can cause the algorithm to diverge; setting it too low makes it slow to converge. In practice, it is generally beneficial to gradually

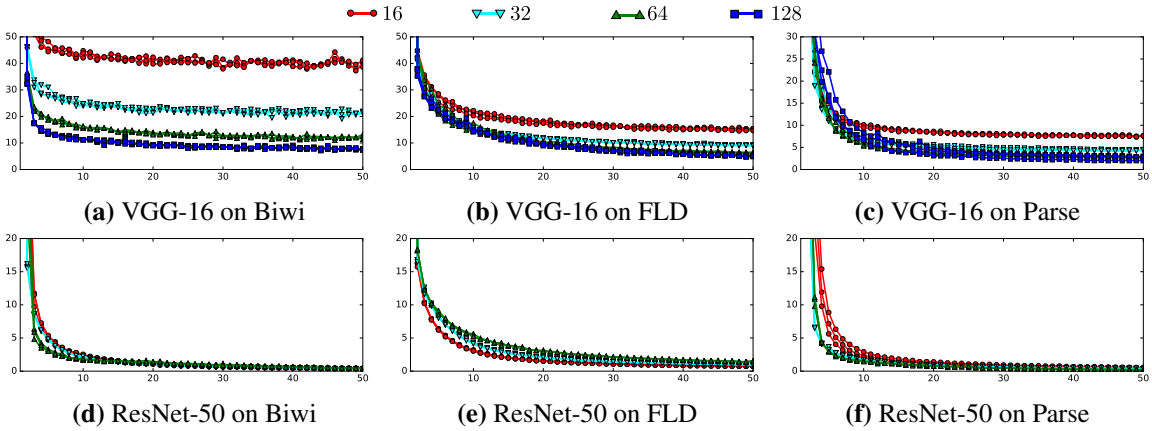
[1] `https://team.inria.fr/perception/deep-regression/`



(a) VGG-16 on Biwi **(b)** VGG-16 on FLD **(c)** VGG-16 on Parse

(d) ResNet-50 on Biwi **(e)** ResNet-50 on FLD **(f)** ResNet-50 on Parse

Figure 6.1: Comparison of the training loss evolution with different optimizers for VGG-16 and ResNet-50.



(a) VGG-16 on Biwi **(b)** VGG-16 on FLD **(c)** VGG-16 on Parse

(d) ResNet-50 on Biwi **(e)** ResNet-50 on FLD **(f)** ResNet-50 on Parse

Figure 6.2: Comparison of the training loss evolution with different batch size on VGG-16 and ResNet-50.

decrease the learning rate over time [60]. Recently, a number of algorithms with adaptive learning rates have been developed, and represent some of the most popular optimization algorithms actively in use.

AdaGrad [46] (for Adaptive Gradient) adapts the learning rate of every weight dividing it by the square root of the sum of their historical squared values. Weights with high gradients will have a rapid decrease of their learning rate, while weights with small or infrequent updates will have a relatively small decrease of their learning rate.

RMSProp [173] (for Root Mean Square Propagation) modifies AdaGrad, to avoid lowering the learning rates very aggressively, by changing the gradient accumulation into an exponentially weighted moving average. AdaGrad shrinks the learning rate according to the entire history of the squared gradient, while RMSProp only considers recent gradients for that weight.

AdaDelta [194] is also an extension of Adagrad, similar to RMSProp, that again dynamically adapts over time using only first order information and requires no manual tuning of a learning rate.

Adam [83] (for Adaptive Moments) is an update to the RMSProp optimizer where momentum [167] is incorporated, i.e. in addition to store an exponentially decaying average of previous squared gradients (like in RMSProp), Adam also employs an exponentially decaying average of previous gradients (similar to momentum, where such moving average of previous gradients helps to dampen oscillations and to accelerate learning with intertia).

These four adaptive optimizers are evaluated in section 6.3.1 and the two exhibiting the highest performance are use in the rest of the chapter. We employ the commonly used mean square error (MSE) as loss function to be optimized. As discussed in detail in Section 6.5, VGG-16 and ResNet-50 are fine-tuned from (and including) the fifth and third convolutional block, respectively.

6.3.1 Impact of the Network Optimizer

As outlined above, we compare four optimizers: AdaGrad, AdaDelta, Adam and RMSProp. For each optimizer, we train the two networks (VGG-16 and ResNet-50) three times during 50 epochs with the default parameter values given in [26]. In this series of experiments, we choose a batch size of 128 and 64 for VGG-16 and ResNet-50, respectively (see section 6.3.2). The evolution of the loss value on the training set is displayed in Figure 6.1 for each one of the three data sets. Even if the differences between the four optimizers are not extremely noticeable, we can state that the best training performances (in terms of loss value and convergence time) correspond to AdaDelta and Adam.

At the light of the results described above, AdaGrad and RMSProp do not seem to be the optimizers to be used when training vanilla deep regression networks. While Adam and AdaDelta perform well, a comparative study prior to the selection of a particular optimizer is strongly encouraged since the choice may depend on the architecture and the problem at hand.

6.3.2 Impact of the Batch Size

In this case, we test the previously selected optimizers (AdaDelta for VGG-16 and Adam for ResNet-50) with batch sizes of 16, 32, and 64. A batch size of 128 is tested on VGG-16 but not in ResNet-50 due to GPU memory limitations. We assess how the batch size impacts the optimization performance: Figure 6.2 shows the loss values obtained when training with different batch sizes.

First, we remark that the impact of the batch size in VGG-16 is more important than in ResNet-50. The latter is more robust to the batch size, yielding comparable results no matter the batch size employed. Second, in general, we could conclude that using a larger batch size is a good heuristic towards good optimization (because in VGG-16 shows to be decisive, and in ResNet-50 does not harm performance). However, the maximal batch size that can be used is constrained by the GPU memory being used. In our case, with an Nvidia TITAN X with 12 GB of memory, we could not train the ResNet-50 with a batch

size of 128. As a consequence, we choose 128 and 64 as batch sizes for VGG-16 and ResNet-50, respectively, and all subsequent experiments are performed using these batch sizes. As it is commonly found in the literature, the larger batch size the better (specially for VGG-16 according to our experiments). Importantly, when used for regression, the performance of ResNet-50 seems to be quite independent of the batch size.

6.4 Statistical Analysis of the Results

Deep learning methods are usually based on stochastic optimization techniques, in which different sources of randomness, i.e. weight initialization, optimization and regularization procedures, have an impact on the results. In the study on optimization techniques and batch size presented in the previous section we already observed some stochastic effects. While these effects did not forbid us to make reasonable optimization choices, other architecture design choices may be in close competition. In other to appropriately referee such competitions, one should draw conclusions based on rigorous statistical tests, rather than based on the average performance of a single training. In this section we describe the statistical procedures implemented to analyze the results obtained after several training trials. We use two statistical tools widely used in many scientific domains.

Generally speaking, statistical tests measure the probability of obtaining experimental results D if hypothesis H is correct, thus computing $P(D|H)$. The null hypothesis (H_0) refers to a general or default statement of a scientific experiment. It is presumed to be true until statistical evidence nullifies it for an alternative hypothesis (H_1). H_0 assumes that any kind of difference or significance observed in the data is due to chance. In this paper, H_0 is that none of the configurations under comparison in a particular experiment is any better, in terms of median performance, than other configurations. The estimated probability of rejecting H_0 when it is true is called p-value. If the p-value is less than the chosen level of significance α then the null hypothesis is rejected. Therefore, α indicates how extreme observed results must be in order to reject H_0. For instance, if the p-value is less than the predetermined significance level (usually 0.05, 0.01, or 0.001, indicated with one, two, or three asterisks, respectively), then the probability of the observed results under H_0 is less than the significance level. In other words, the observed result is highly unlikely to be the result of random chance. Importantly, the p-value only provides an index of the evidence against the null hypothesis, i.e. it is mainly intended to establish whether further research into a phenomenon could be justified. We consider it as one bit of evidence to either support or challenge accepting the null hypothesis, rather than as conclusive evidence of significance [51, 128, 177], and a statistically insignificant outcome should be interpreted as "absence of evidence, not evidence of absence" [163].

Statistical tests can be categorized into two classes: parametric and non-parametric. Parametric tests are based on assumptions (like normality or homoscedasticity) that are commonly violated when analyzing the performance of stochastic algorithms [40]. In our case, the visual inspection of the error measurements as well as the application of normality tests (in particular, the Lilliefors test) indicates a lack of normality in the data, leading to the use of non-parametric statistical tests.

Statistical tests can perform two kinds of analysis: pairwise comparisons and multiple comparisons. Pairwise statistical procedures perform comparisons between two algorithms, obtaining in each application a p-value independent from another one. Therefore, in order to carry out a comparison which involves more than two algorithms, multiple comparisons tests should be used. If we try to draw a conclusion involving more than one pairwise comparison, we will obtain an accumulated error coming from its combination. In statistical terms, we are losing control on the Family-Wise Error Rate (FWER), defined as the probability of making one or more false discoveries (type I errors) among all the hypotheses when performing multiple pairwise tests. Examples of post-hoc procedures, used to control the FWER, are Bonferroni-Dunn [47], Holm [72], Hochberg [68], Hommel [73], Holland [71], Rom [143], or Nemenyi [125]. Following the recommendation of Derrac et al. [40], we use the Holm procedure to control the FWER in this paper.

Summarizing, once determined that non-parametric statistics will be used, we decided to follow standard and well-consolidated statistical approaches: when pairwise comparisons have to be made, the Wilcoxon signed-rank test is applied; when multiple comparisons have to be made (i.e. more than two methods are compared, thus increasing the number of pairwise comparisons), the FWER is controlled by applying the Bonferroni-Holm procedure (also called the Holm method) to multiple Wilcoxon signed-rank tests. Finally, the 95% confidence interval for the median of the MAE is reported.

6.4.1 Wilcoxon Signed-rank Test

The Wilcoxon signed-rank test [181] is a non-parametric statistical hypothesis test used to compare two related samples to assess the null hypothesis that the median difference between pairs of observations is zero. It can be used as an alternative to the paired Student's t-test, t-test for matched pairs,[2] when the population cannot be assumed to be normally distributed. We use Wilcoxon signed-rank test to evaluate which method is the best (i.e. the most recommendable configuration according with our results) and the worst (i.e. the less recommendable configuration according with our results). The statistical significance is displayed on each table using asterisks, as commonly employed in the scientific literature (where * represents a p-value smaller than 0.05 but larger or equal than 0.01; ** represents a p-value smaller than 0.01 but larger or equal than 0.001; and *** represents a p-value smaller than 0.001). When more than one configuration has asterisks that implies that those configurations are significantly better than the others but there are no statistically significant differences between them. The worst performing configurations are displayed using circles and following the same criterion.

6.4.2 Confidence Intervals for the Median

Importantly, with a sufficiently large sample, statistical significance tests may detect a trivial effect, or they may fail to detect a meaningful or obvious effect due to small sample size. In other words, very small differences, even if statistically significant, can be

[2] Two data samples are matched/paired if they come from repeated observations of the same subject.

practically meaningless. Therefore, since we consider that reporting only the significant p-value for an analysis is not enough to fully understand the results, we decided to introduce confidence intervals as a mean to quantify the magnitude of each parameter of interest.

Confidence intervals consist of a range of values (interval) that act as good estimates of the unknown population parameter. Most commonly, the 95% confidence interval is used. A confidence interval of 95% does not mean that for a given realized interval there is a 95% probability that the population parameter lies within it (i.e. a 95% probability that the interval covers the population parameter), but that there is a 95% probability that the calculated confidence interval from some future experiment encompasses the true value of the population parameter. The 95% probability relates to the reliability of the estimation procedure, not to a specific calculated interval. If the true value of the parameter lies outside the 95% confidence interval, then a sampling event that has occurred with a probability of 5% (or less) of happening by chance.

We can estimate confidence intervals for medians and other quantiles using the binomial distribution. The 95% confidence interval for the q-th quantile can be found by applying the binomial distribution [27]. The number of observations less than the q quantile will be an observation from a Binomial distribution with parameters n and q, and hence has mean nq and standard deviation $\sqrt{(nq(1-q)}$. We calculate j and k such that: $j = nq - 1.96\sqrt{nq(1-q)}$ $k = nq + 1.96\sqrt{nq(1-q)}$ We round j and k up to the next integer. Then the 95% confidence interval is between the jth and k^{th} observations in the ordered data.

6.5 Network Variants

The statistical tests described above are used to compare the performance of each choice on the three data sets for the two base architectures. Due to the amount of time necessary to train deep neural architectures, we cannot compare all possible combinations, and therefore we must evaluate one choice at a time (e.g. the use of batch normalization). In order to avoid over-fiting, we use holdout as model validation tecnique, and test the generalization ability with an independent data set. In detail, we use 20% of training data for validation (26% in the case of FLD, because the validation set is explicitly provided in [166]). We use early stopping with a patience equal to four epochs (an epoch being a complete pass through the entire training set). In other words, the network is trained until the loss on the validation set does not decrease during four consecutive epochs. The two baseline networks are trained with the optimization settings chosen in section 6.3. In this section, we evaluate the performance of different network *variants*, on the three problems.

Since ConvNets have a high number of parameters, they are prone to over-fitting. Two common regularization strategies are typically used [161]:

Batch Normalization (BN) was introduced to lead to fast and reliable network convergence [76]. In the case of VGG-16 (resp. ResNet-50), we cannot add (remove) a batch

Table 6.1: Network baseline specification.

Network	BN	FT	DO	LR	PL
VGG-16	$\mathbf{BN}$	$\mathbf{CB^5}$	$\mathbf{10{-}DO}$	$\rho(\mathbf{FC^2})$	$\mathbf{FC^2}$
ResNet-50	$\cancel{\mathbf{BN}}$	$\mathbf{CB^3}$	-	$\rho(\mathbf{GAP})$	$\mathbf{GAP}$

Table 6.2: Impact of the batch normalization (BN) layer on VGG-16 and ResNet-50.

Data Set	BN	VGG-16				ResNet-50			
		MAE test	MSE train	MSE valid	MSE test	MAE test	MSE train	MSE valid	MSE test
Biwi	$\cancel{\mathbf{BN}}$	[5.04 5.23]$^{\circ\circ}$	[2.52 2.56]	[20.68 21.45]	[35.68 37.81]	[3.60 3.71]***	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
	$\mathbf{BN}$	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]	[4.59 4.69]	[2.18 2.22]	[22.93 23.63]	[28.56 30.07]
	$\mathbf{BNB}$	[4.63 4.76]	[8.11 8.29]	[16.69 17.32]	[30.49 32.57]	–	–	–	–
FLD	$\cancel{\mathbf{BN}}$	[3.67 3.90]	[21.19 21.45]	[22.26 22.76]	[19.70 22.25]	[1.96 2.05]	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	$\mathbf{BN}$	[2.61 2.76]***	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]	[1.92 2.01]	[4.81 4.86]	[6.83 6.94]	[5.70 6.34]
	$\mathbf{BNB}$	[15.53 16.65]$^{\circ\circ\circ}$	[300.3 304.5]	[300.4 307.7]	[326.9 369.4]	–	–	–	–
Parse	$\cancel{\mathbf{BN}}$	[6.54 7.17]	[9.50 9.68]	[56.74 57.96]	[69.12 84.83]	[4.86 5.68]***	[0.64 0.64]	[29.40 30.21]	[43.58 55.71]
	$\mathbf{BN}$	[4.90 5.59]***	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]	[5.72 6.25]	[0.89 0.89]	[36.24 37.22]	[55.57 69.71]
	$\mathbf{BNB}$	[11.20 12.68]$^{\circ\circ\circ}$	[152.2 154.0]	[142.9 146.2]	[184.9 238.1]	–	–	–	–

normalization layer deeply in the network since the pre-trained weights of the layers after this batch normalization layer were obtained without it. Consequently, in the case of VGG-16 we can add a batch normalization layer either right before REG (hence after the activation of FC^2, denoted by $\mathbf{BN}$), before the activation of FC^2 (denoted by $\mathbf{BNB}$) or do not use batch normalization $\cancel{\mathbf{BN}}$. In ResNet-50 we consider only $\mathbf{BN}$ and $\cancel{\mathbf{BN}}$, since the batch normalization layer before the activation of the last convolutional layer is there by default. Intuitively, VGG-16 will benefit from the configuration $\mathbf{BN}$, but not ResNet-50. This is due to the fact that the original VGG-16 does not exploit batch normalization, while ResNet-50 does. Using $\mathbf{BN}$ in ResNet-50 would mean finishing by convolutional layer, batch normalization, activation, GAP, batch normalization and REG. A priori we do no expect gains when using ResNet-50 with $\mathbf{BN}$ (and this is why the ResNet-50 baselines do not use BN), but we include this comparison for completeness.

Dropout (DO) is a widely used method to avoid over-fitting [161]. Dropout is not employed in ResNet-50, and thus we perform experiments only on VGG-16. We compare different settings: no dropout (denoted by $\mathbf{00{-}DO}$), dropout in FC^1 but not in FC^2 ($\mathbf{10{-}DO}$), dropout in FC^2 but not in FC^1 ($\mathbf{01{-}DO}$) and dropout in both ($\mathbf{11{-}DO}$).

Other approaches consist in choosing a network architecture that is less prone to overfitting, for instance modifying:

Fine-tuning depth (FT) indicates the deeper block that is trained. Indeed, when using a pre-trained network, only the last layers are modified. Concerning VGG-16, we compare $\mathbf{CB^3}$, $\mathbf{CB^4}$, $\mathbf{CB^5}$ and $\mathbf{FC^1}$. Concerning ResNet-50, we compare $\mathbf{CB^2}$, $\mathbf{CB^3}$, $\mathbf{CB^4}$, $\mathbf{CB^5}$.

Regressed layer (RL) denotes the layer after which the BN and REG layers are added. $\rho(LR)$ denotes the model where the regression is performed on the output activations of the layer LR, meaning that on top of that layer we directly add a batch normalization and a linear regression layers. For VGG-16, we compare $\rho(\mathbf{CB^5})$, $\rho(\mathbf{FC^1})$ and $\rho(\mathbf{FC^2})$. For ResNet-50 we compare $\rho(\mathbf{CB^5})$ and $\rho(\mathbf{GAP})$.

Table 6.3: Impact of the dropout (DO) layer on VGG-16.

Dataset	DO	MAE test	MSE train	MSE valid	MSE test
Biwi	**00−DO**	[4.47 4.60]$^{\circ\circ\circ}$	[6.34 6.45]	[14.42 14.91]	[28.80 30.98]
	01−DO	[3.56 3.67]	[3.35 3.40]	[12.00 12.40]	[17.52 18.54]
	10−DO	[3.66 3.79]	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]
	11−DO	[3.37 3.48]***	[3.46 3.53]	[11.66 12.04]	[15.39 16.38]
FLD	**00−DO**	[2.55 2.70]	[8.60 8.71]	[10.53 10.74]	[10.16 11.23]
	01−DO	[2.26 2.38]***	[7.81 7.90]	[9.19 9.38]	[7.87 8.68]
	10−DO	[2.61 2.76]	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]
	11−DO	[2.27 2.42]***	[7.59 7.68]	[9.10 9.29]	[7.94 8.92]
Parse	**00−DO**	[4.83 5.44]	[1.39 1.41]	[27.28 28.09]	[40.38 51.11]
	01−DO	[4.87 5.52]	[2.87 2.90]	[27.89 28.64]	[42.92 50.80]
	10−DO	[4.90 5.59]	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]
	11−DO	[4.91 5.59]	[3.25 3.28]	[29.50 30.21]	[43.46 54.62]

Pooling layer (PL) defines how the convolutional maps are converted into vectors. In the case of VGG-16, the 2D feature maps of CB^5 are converted via a flattening layer Fl that does not reduce the dimension. Alternatively, we could replace Fl by global (max or average) pooling, denoted by GMP and GAP respectively. ResNet-50 already uses GAP and hence we can only compare to GMP. The models are denoted **GAP**, **GMP**, for both, and $\mathbf{FC^2}$ for VGG-16.

The settings corresponding to our baselines are detailed in Table 6.1. These baselines are chosen at the light of preliminary experiments (not reported) and without important changes with respect to the original design. The background color is gray, as it will be for these two configurations in the following. The rest of the section is devoted to discuss the results obtained for the variants previously discussed.

6.5.1 Batch Normalization

Table 6.2 shows the results obtained with the various choices of batch normalization. In the case of VGG-16, we observe that the impact of adding a batch normalization layer after the activations (i.e. **BN**) is significant and beneficial compared to the other two options. In the case of FLD, we notice that the problem with ~~**BN**~~ and **BNB** occurs at training since the final training MSE score is much higher than the one obtained with **BN**. Interestingly, on the Biwi data set we observe that the training MSE is better with ~~**BN**~~ but **BN** performs better on the validation and the test sets. We conclude that in the case of Biwi, **BN** does not help the optimization, but increases the generalization ability. The high beneficial impact observed in VGG-16 justifies that we use it in the baseline network. The exact opposite trend is observed when running the same experiments with ResNet-50. Indeed, **BN** improves neither for the optimization (with the exception of FLD, which appears to be quite insensitive to BN using ResNet-50), nor for the generalization ability. As discussed, this is expected because ResNet-50 uses already batch normalization (before the activation).

6.5.2 DROPOUT RATIO

Table 6.3 shows the results obtained when comparing different dropout strategies with VGG-16 (ResNet-50 does not have fully connected layers and thus no dropout is used). At the light of these results, it is encouraged to use dropout in both fully connected layers. However, for Parse, this does not seem to have any impact, and on FLD the use of dropout in the first fully connected layer seems to have very mild impact, at least in terms of MAE performance on the test set. Since on the Biwi data set the $\mathbf{00-DO}$ strategy is significantly worse than the other strategies, and not especially competitive in the other two data sets, we would suggest not to use this strategy. Globally, $\mathbf{11-DO}$ is the safest option (best for Biwi/FLD, equivalent for Parse).

Table 6.4: Impact of the finetunig depth (FT) on VGG-16.

Data	FT.	MAE test	MSE train	MSE valid	MSE test
Biwi	$\mathbf{FC^1}$	[5.13 5.27]	[4.12 4.20]	[29.18 30.44]	[37.10 39.03]
	$\mathbf{CB^5}$	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]
	$\mathbf{CB^4}$	[4.88 5.03]	[4.32 4.40]	[15.96 16.54]	[30.95 32.71]
	$\mathbf{CB^3}$	[5.33 5.46]$^{\circ\circ\circ}$	[9.30 9.51]	[33.48 34.52]	[38.66 40.34]
FLD	$\mathbf{FC^1}$	[3.32 3.47]	[4.00 4.04]	[16.02 16.29]	[16.56 18.25]
	$\mathbf{CB^5}$	[2.61 2.76]***	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]
	$\mathbf{CB^4}$	[2.85 3.10]	[6.31 6.40]	[7.79 7.97]	[12.52 14.45]
	$\mathbf{CB^3}$	[3.48 3.74]$^{\circ\circ\circ}$	[10.83 10.98]	[11.80 12.05]	[18.52 21.96]
Parse	$\mathbf{FC^1}$	[5.50 6.31]$^{\circ}$	[1.31 1.33]	[37.29 38.33]	[49.61 63.18]
	$\mathbf{CB^5}$	[4.90 5.59]***	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]
	$\mathbf{CB^4}$	[4.92 5.87]	[2.42 2.46]	[29.91 30.61]	[43.04 57.09]
	$\mathbf{CB^3}$	[5.38 6.13]	[2.90 2.95]	[37.49 38.26]	[49.21 65.75]

Table 6.5: Impact of the fine tuning depth (FT) on ResNet-50.

Data	FT	MAE test	MSE train	MSE valid	MSE test
Biwi	$\mathbf{CB^5}$	[8.69 8.90]$^{\circ\circ\circ}$	[32.57 33.17]	[140 145.1]	[102.5 107.8]
	$\mathbf{CB^4}$	[3.40 3.51]***	[0.87 0.89]	[17.85 18.43]	[15.98 16.86]
	$\mathbf{CB^3}$	[3.60 3.71]	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
	$\mathbf{CB^2}$	[4.17 4.30]	[1.41 1.43]	[26.31 27.18]	[24.19 25.34]
FLD	$\mathbf{CB^5}$	[8.79 9.30]$^{\circ\circ\circ}$	[114.6 116.2]	[120.2 123.1]	[113.1 127.1]
	$\mathbf{CB^4}$	[2.21 2.31]	[4.48 4.52]	[8.10 8.24]	[7.36 8.01]
	$\mathbf{CB^3}$	[1.96 2.05]***	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	$\mathbf{CB^2}$	[2.04 2.14]	[5.10 5.16]	[7.00 7.13]	[6.28 6.89]
Parse	$\mathbf{CB^5}$	[8.27 9.28]$^{\circ\circ\circ}$	[54.22 54.99]	[77.32 78.88]	[102.5 132.5]
	$\mathbf{CB^4}$	[5.07 5.86]	[0.90 0.91]	[31.74 32.54]	[44.85 56.78]
	$\mathbf{CB^3}$	[4.86 5.68]***	[0.64 0.64]	[29.40 30.21]	[43.58 55.71]
	$\mathbf{CB^2}$	[5.02 5.84]	[0.84 0.85]	[30.48 31.29]	[45.65 61.04]

6.5.3 FINE TUNING DEPTH

Tables 6.4 and 6.5 show results obtained with various fine-tuning depth values, as described in section 6.5, both for VGG-16 and for ResNet-50. In the case of VGG-16, we

Table 6.6: Impact of the regressed layer (RL) when using VGG-16.

Data Set	RL	MAE test	MSE train	MSE valid	MSE test
Biwi	$\rho(\mathbf{FC^2})$	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]
	$\rho(\mathbf{FC^1})$	[5.17 5.31]$^{\circ\circ\circ}$	[9.49 9.66]	[17.84 18.40]	[36.32 38.53]
	$\rho(\mathbf{CB^5})$	[4.64 4.75]	[5.38 5.47]	[16.96 17.49]	[28.84 29.85]
FLD	$\rho(\mathbf{FC^2})$	[2.61 2.76]***	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]
	$\rho(\mathbf{FC^1})$	[3.51 3.68]	[9.87 10.00]	[12.98 13.23]	[18.22 19.97]
	$\rho(\mathbf{CB^5})$	[3.61 3.82]$^{\circ\circ}$	[16.55 16.74]	[18.49 18.84]	[19.00 21.38]
Parse	$\rho(\mathbf{FC^2})$	[4.90 5.59]**	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]
	$\rho(\mathbf{FC^1})$	[4.99 5.66]	[2.61 2.64]	[30.13 30.74]	[43.25 55.67]
	$\rho(\mathbf{CB^5})$	[5.58 6.14]$^{\circ\circ\circ}$	[2.83 2.85]	[34.46 35.12]	[52.62 61.34]

observe a similar behavior of $\mathbf{CB^5}$ as the one encountered for the batch normalizer. Indeed, $\mathbf{CB^5}$ may not be the best choice in terms of optimization ($\mathbf{FC^1}$ exhibits smaller training MSEs) but it is the best in terms of generalization ability (for MSE validation and test as well as MAE test). For VGG-16, this result is statistically significant for all three data sets. In addition, the results shown in Table 6.4 also discourage to use $\mathbf{CB^3}$. It is more difficult to conclude in the case of ResNet-50. While for two of the data sets the recommendation is to choose the baseline (i.e. fine tune from CB^3), ResNet-50 on Biwi selects the model $\mathbf{CB^4}$ by a solid margin in a statistically significant manner. Therefore, we suggest than, when using ResNet-50 for regression, one still runs an ablation study varying the number of layers that are fine tuned. In this ablation study, the option $\mathbf{CB^5}$ should not necessarily be included, since for all data sets this option is significantly worse than the other ones.

6.5.4 Regressed Layer

Tables 6.6 and 6.7 show the results obtained when varying the regressed layer, for VGG-16 and ResNet-50 respectively. In the case of VGG-16 we observe a strongly consistent behavior, meaning that the best method in terms of optimization performance is also the method that best generalizes (in terms of MSE validation and test as well as MAE test). Regressing from the second fully connected layer is a good choice when using VGG-16, whereas the other choices may be strongly discouraged depending on the data set. The results on ResNet-50 are a bit less conclusive. The results obtained are all statistically significant but different depending on the data set. Indeed, experiments on Biwi and Parse point to use **GAP**, while results on FLD point to use $\mathbf{CB^5}$. However, we can observe that the confidence intervals of $\rho(\mathbf{GAP})$ and $\rho(\mathbf{CB^5})$ with ResNet-50 on Biwi are not that different. This means that the difference between the two models is small while consistent over the test set images.

6.5.5 Pooling layer

Table 6.8 shows the results obtained for different pooling strategies on VGG-16 and ResNet-50. Regarding VGG-16, we observe that except for FLD, the best option is to

Table 6.7: Impact of the regressed layer (RL) when using ResNet-50.

Data Set	RL	MAE test	MSE train	MSE valid	MSE test
Biwi	$\rho(\mathbf{GAP})$	[3.60 3.71]***	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
	$\rho(\mathbf{CB^5})$	[3.61 3.73]	[0.71 0.72]	[15.42 15.92]	[17.60 18.55]
FLD	$\rho(\mathbf{GAP})$	[1.96 2.05]	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	$\rho(\mathbf{CB^5})$	[1.61 1.70]***	[1.77 1.79]	[4.81 4.90]	[3.98 4.36]
Parse	$\rho(\mathbf{GAP})$	[4.86 5.68]***	[0.64 0.64]	[29.30 30.09]	[43.58 55.71]
	$\rho(\mathbf{CB^5})$	[5.56 6.24]	[0.48 0.48]	[36.76 37.61]	[54.69 67.21]

Table 6.8: Impact of the pooling layer (PL) on VGG-16 and ResNet-50.

Data Set	PL	VGG-16				ResNet-50			
		MAE test	MSE train	MSE valid	MSE test	MAE test	MSE train	MSE valid	MSE test
Biwi	**GMP**	[3.97 4.08]	[4.03 4.11]	[18.25 18.99]	[21.19 22.38]	[3.64 3.75]	[1.48 1.50]	[20.62 21.23]	[17.80 18.88]
	GAP	[3.99 4.09]	[2.75 2.80]	[20.71 21.40]	[21.03 22.07]	[3.60 3.71]***	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
	FC2	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]	–	–	–	–
FLD	**GMP**	[2.50 2.64]***	[3.02 3.06]	[7.65 7.81]	[9.56 11.07]	[1.75 1.82]***	[2.17 2.19]	[5.21 5.30]	[4.67 5.22]
	GAP	[2.53 2.67]**	[5.18 5.23]	[9.06 9.22]	[9.69 10.64]	[1.96 2.05]	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	FC2	[2.61 2.76]	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]	–	–	–	–
Parse	**GMP**	[5.55 6.06]	[1.17 1.18]	[30.00 30.67]	[52.17 61.86]	[5.74 6.51]	[0.89 0.90]	[36.39 37.17]	[54.63 69.10]
	GAP	[5.61 6.07]	[1.48 1.50]	[31.46 32.15]	[51.33 60.57]	[4.86 5.68]***	[0.64 0.64]	[29.30 30.09]	[43.58 55.71]
	FC2	[4.90 5.59]***	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]	–	–	–	–

keep the flatten layer, and therefore discard the use of pooling layers (**GMP** or **GAP**). The case of FLD is different, since the results would suggest that it is significantly better to use either **GAP** or **GMP**. Similar conclusions can be drawn from the results obtained with ResNet-50 in the sense that the standard configuration (**GAP**) is the optimal for Biwi and Parse, but not for FLD. In the case of FLD, the results point to choose **GMP**.

6.5.6 Discussion on network variants

In this section we summarize the results obtained with different network variants. Firstly, evaluating the use of batch normalization is mandatory, but it would appear that conclusions are constant through different data sets, hence there is no need to run extensive experimentation to select the best option. Second, regarding the number of layers to fine-tune, we recommend to exploit the model $\mathbf{CB^5}$ for VGG-16 unless strong reasons would support a different choice. For ResNet-50 the choice would be between $\mathbf{CB^3}$ and $\mathbf{CB^4}$ (but definitely not $\mathbf{CB^5}$). Third, regression should be done from FC^2 in VGG-16 (we do not recommend any of the tested alternatives) and either from GAP or from CB^5 in ResNet-50. Fourth, in terms of dropout, and as it is classically observed in the literature, one should use dropout in VGG-16, specially in the second layer (this experiment does not apply to ResNet-50). Finally, the pooling strategies should all be tested, since their optimality depends upon the network and data set (in a statistically significant manner).

Very importantly, in case of limited resources (e.g. computing time), taking the suboptimal choice may not lead to a crucial difference with respect to the perfect combination of parameters. However, one must avoid the cases in which the method is proven to be significantly worse than the other, because in those cases performances (in train, validation

and test) have proven to be evidently different.

6.6 DATA PRE-PROCESSING

In this section we discuss the different data pre-processing techniques that we consider in our benchmark. Because VGG-16 has two fully connected layers, we are constrained to use the pre-defined input size of 224×224 pixels. Hence, we systematically resize the images to this size. For the sake of a fair comparison, the very same input images are given to both networks. Firstly, we evaluate the impact of mirroring the training images (not used for test). Table 6.9 reports the results when evaluating the impact of mirroring. The conclusion is unanimous: mirroring is statistically better for all configurations. In addition, in most of the cases the confidence intervals are disjoint meaning that with high probability the output obtained when training with mirroring will have lower error than training without mirroring.

Since the three data sets used in our study have different characteristics, the pre-processing steps differ from data set to data set. Importantly, in all cases the pre-processing baseline technique is devised from the common usage of the data sets in the recent literature. Below we specify the baseline pre-processing as well as other tested pre-processing alternatives for each data set.

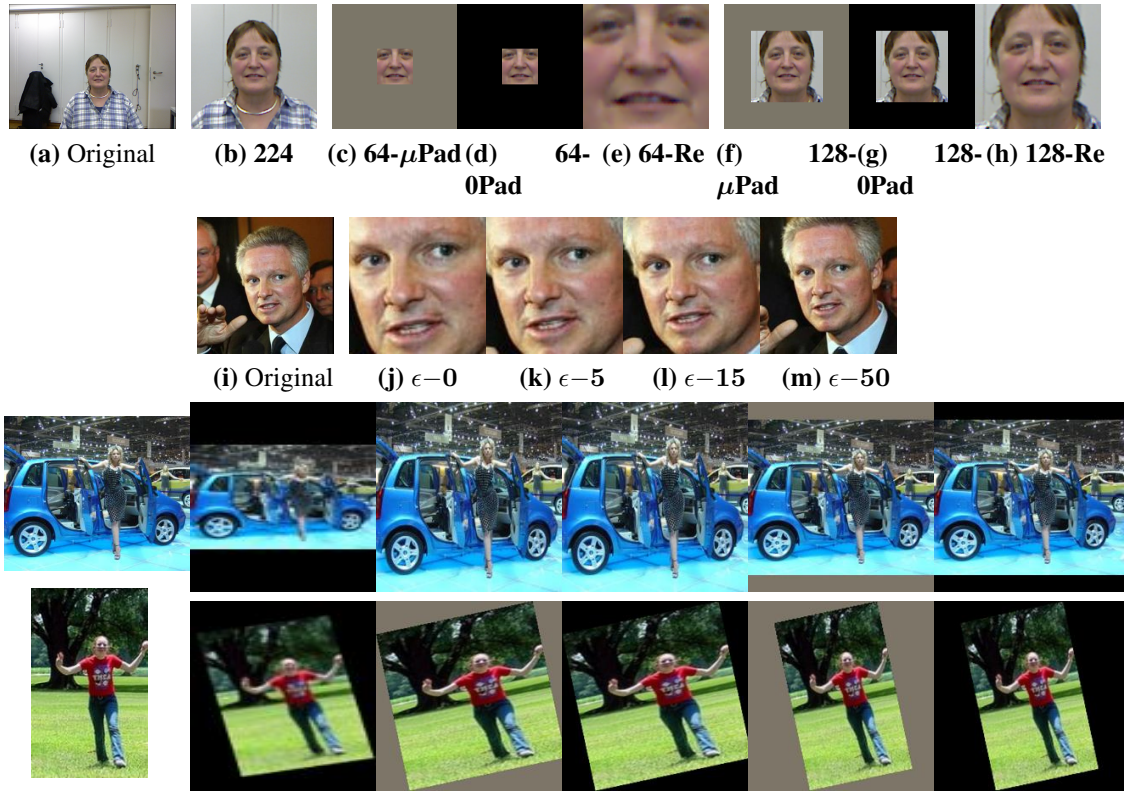
Table 6.9: Impact of the mirroring (Mirr.) on VGG16 and ResNet50.

Data Set	Mirr.	VGG-16				ResNet-50			
		MAE test	MSE train	MSE valid	MSE test	MAE test	MSE train	MSE valid	MSE test
Biwi	Yes	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]	[3.60 3.71]***	[2.38 2.41]	[27.74 28.48]	[17.15 18.14]
	No	[5.49 5.67]	[9.09 9.36]	[24.22 25.61]	[42.55 45.73]	[4.46 4.57]	[1.39 1.42]	[19.70 20.83]	[27.34 28.60]
FLD	Yes	[2.61 2.76]***	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]	[1.96 2.05]***	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	No	[3.06 3.24]	[14.13 14.38]	[15.29 15.73]	[14.28 15.68]	[2.05 2.13]	[4.41 4.46]	[7.04 7.20]	[6.43 7.15]
Parse	Yes	[4.90 5.59]***	[2.38 2.41]	[27.74 28.48]	[41.35 52.13]	[4.86 5.68]***	[0.64 0.64]	[29.40 30.21]	[43.58 55.71]
	No	[5.08 5.76]	[2.31 2.36]	[28.98 30.14]	[45.15 57.08]	[5.88 6.62]	[1.14 1.16]	[42.99 44.38]	[59.30 77.70]

Table 6.10: Impact of the data pre-processing on VGG-16 and ResNet-50.

Data Set & Pre-processing		VGG-16				ResNet-50			
		MAE test	MSE train	MSE valid	MSE test	MAE test	MSE train	MSE valid	MSE test
Biwi	**128-μPad**	[3.93 4.02]	[5.89 6.00]	[15.34 15.85]	[21.13 21.94]	[3.47 3.55]	[1.05 1.06]	[18.71 19.35]	[16.41 17.13]
	128-Re	[3.87 3.97]	[4.98 5.06]	[13.23 13.75]	[19.80 20.79]	[3.18 3.25]	[0.97 0.98]	[16.37 16.91]	[13.63 14.25]
	128-0Pad	[3.97 4.05]	[7.83 7.98]	[15.55 16.09]	[21.51 22.31]	[3.20 3.28]	[1.52 1.55]	[17.94 18.52]	[14.45 15.03]
	64-μPad	[4.48 4.63]	[6.09 6.22]	[21.52 22.31]	[27.91 29.77]	[3.34 3.47]	[1.45 1.47]	[21.18 21.99]	[15.55 16.73]
	64-Re	[4.06 4.21]	[6.79 6.92]	[15.05 15.59]	[22.72 24.32]	[2.98 3.07]**	[0.99 1.00]	[13.18 13.66]	[12.20 12.90]
	64-0Pad	[4.80 5.02]$^{\circ\circ\circ}$	[12.15 12.40]	[19.35 20.05]	[32.72 35.33]	[3.29 3.42]	[3.34 3.40]	[19.21 20.01]	[14.91 16.02]
	224	[3.66 3.79]***	[4.33 4.41]	[12.18 12.56]	[18.77 20.20]	[3.60 3.71]$^{\circ\circ\circ}$	[1.25 1.27]	[21.49 22.25]	[17.15 18.14]
FLD	**ϵ-0**	[2.25 2.36]***	[7.61 7.69]	[8.94 9.11]	[7.69 8.57]	[1.63 1.73]***	[2.41 2.44]	[5.07 5.16]	[4.23 4.63]
	ϵ-5	[2.61 2.76]	[9.43 9.54]	[10.77 11.03]	[10.55 11.62]	[1.96 2.05]	[5.21 5.25]	[6.85 6.95]	[5.79 6.39]
	ϵ-15	[2.35 2.48]	[8.42 8.50]	[9.88 10.07]	[8.37 9.36]	[2.00 2.08]	[4.48 4.53]	[6.59 6.70]	[5.95 6.43]
	ϵ-50	[2.96 3.17]$^{\circ\circ\circ}$	[11.68 11.82]	[13.33 13.60]	[13.32 15.31]	[2.59 2.72]$^{\circ\circ\circ}$	[7.12 7.18]	[9.92 10.07]	[10.11 11.20]
Parse	**α-μPad**	[9.99 11.52]$^{\circ\circ\circ}$	[8.47 8.62]	[114.5 117.5]	[161.4 226.1]	[11.61 13.63]$^{\circ\circ\circ}$	[4.29 4.35]	[180.7 185.8]	[235.3 311.5]
	α-0Pad	[9.72 11.13]	[9.77 9.95]	[116.1 119.2]	[165.3 212.9]	[10.90 12.54]	[3.34 3.39]	[158.5 163.7]	[207.9 279.6]
	$\not\alpha$-μPad	[8.54 9.56]***	[6.22 6.35]	[106.9 109.4]	[125.5 158.7]	[9.30 11.28]***	[2.83 2.87]	[140.3 144.5]	[160.8 218.3]
	$\not\alpha$-0Pad	[8.39 9.33]***	[8.23 8.39]	[111.9 115.3]	[120.9 152.9]	[9.15 10.82]***	[3.35 3.39]	[136.6 140.7]	[151.7 194.1]
	120x80	[9.55 11.34]	[12.05 12.23]	[130.6 134.3]	[161.0 215.8]	[9.14 10.57]***	[3.40 3.45]	[136.5 140.4]	[155.4 204.7]

Biwi. The baseline for the Biwi data set is inspired from [44, 93], where the authors



(a) Original (b) 224 (c) 64-μPad (d) 64-0Pad (e) 64-Re (f) 128-μPad (g) 128-0Pad (h) 128-Re

(i) Original (j) $\epsilon-0$ (k) $\epsilon-5$ (l) $\epsilon-15$ (m) $\epsilon-50$

(n) Original (o) 120x80 (p) α-μPad (q) α-0Pad (r) α-μPad (s) α-0Pad

Figure 6.3: Pre-processed examples for the Biwi (top), FLD (middle) and Parse (bottom) data sets.

crop a 64×64 and a 224×224 window respectively, centered on the face. We investigate the use of three window sizes: 64×64, 128×128 and 224×224. This latter is referred to as **224**. When cropping windows smaller than 224×224 pixels, we investigate three possibilities: resize (denoted by **64-Re** and **128-Re**), padding with the mean value of ImageNet (denoted by **64-μPad** and **128-μPad**) and padding with zeros (denoted by **64-0Pad** and **128-0Pad**). Examples of this pre-processing steps are shown in Figures 6.3a-6.3h.

FLD. In the case of the facial landmark data set, original images and face bounding boxes are provided. Similarly to the Biwi data set, the issue of the amount of context information is investigated. [166] proposes to expand the bounding boxes and then to adopt a cascade strategy to refine the input regions of the networks. As we want to keep our processing as general as possible, we adopt the following procedure: the face bounding box is expanded by $\epsilon\%$ in each direction. We compare four different expanding ratios: 0%, 5%, 15% and 50%. These are denoted with **ϵ-0**, **ϵ-5**, **ϵ-15**, and **ϵ-50** respectively, see Figure 6.3i to 6.3m.

Parse. When using this data set in [9], the images were resized to 120×80 pixels. In our case, we resize the images to fit into a rectangle of this size and pad to a squared image with zeros, followed by resizing to 224×224 pixels. This strategy is referred to

as **120x80**. We also consider directly resizing into 224×224 images, hence without keeping the aspect ratio, and pad with zeros or the mean value of ImageNet (denoted by **$\not\alpha$-0Pad** and **$\not\alpha$-μPad**). Padding plays a role only in rotated images. Finally, we consider applying the same padding to images resized keeping the aspect ratio (**α0-Pad** and **α-μPad**). Examples of these strategies are shown in Figures 6.3n-6.3s.

Table 6.10 reports the results obtained by the different pre-processing techniques for each data set for both VGG-16 and ResNet-50. The point locations are represented by their pixel Cartesian coordinates in the case of FLD and Parse. When we evaluate a data pre-processing strategy, the images are geometrically modified and the same transformation is applied to the annotations. Consequently, the errors cannot be directly compared between two different pre-processing strategies. For instance, in the last row of Figure 6.3 a 5-pixel error for the right elbow location may be acceptable in the case of **$\not\alpha$-0Pad** but may correspond to a confusion with a shoulder location in the case of **α-0Pad**. In order to compare the pre-processing strategies, we transform all the errors into a common space. We choose common spaces such that the aspect ratio of the original images is kept. For Parse, the errors are compared in the original image space (i.e. before any resize or crop operation). In all the experiments of section 6.5, the errors are reported in the space of **120x80**. This choice is justified by the fact that the MSE reported are the exact loss values used to optimize and proceed to early stopping. The drawback of this choice is that the errors on Parse obtained in Tables 6.2, to 6.9 are not directly comparable with those of Table 6.10. In the case of FLD, **ϵ-0** space corresponds to original detections. However, as the transformations between spaces are only linear scalings, the comparison can be performed in any space without biasing the results. Therefore, we chose to report the errors in the space corresponding to **ϵ-5** to allow comparison with Tables 6.2 to 6.9. In the case of Biwi, as the head angle is independent of the pre-processing strategy, no transformation is required to compare the strategies.

Regarding the experiments on Biwi, we notice that the best strategy for VGG-16 is **224**, whereas for ResNet-50 is **64-Re**. Having said that, the differences with the second best (in terms of the confidence interval) are below 1 degree in MAE. Interestingly, we observe that in both cases two strategies are significantly worse: $\mathbf{64-0Pad}$ for VGG-16 and $\mathbf{224}$ for ResNet-50. The fact that the same strategy $\mathbf{224}$ is significantly the best for VGG-16 and significantly the worst for ResNet-50 demonstrates that a serious ablation study of pre-processing strategies is required on Biwi.

Experiments on the FLD data set are clearly more conclusive than the ones on Biwi. Indeed, for both architectures the $\epsilon - \mathbf{0}$ strategy is significantly better and the $\epsilon - \mathbf{50}$ is significantly worse. The differences range from approxiamtively 0.1 pixel (with respect to the second best strategy) to almost one pixel (with respect to the worst strategy).

Regarding the experiments on Parse, we obtain for the first time in this chapter a statistical tie. Indeed, **$\not\alpha$-μPad** and **$\not\alpha$-0Pad** with VGG-16 are better than the rest, but without statistical differences between them. A similar behavior is found in ResNet-50 including also **120x80**.

The results on data pre-processing and mirroring behave quite differently. While mir-

Table 6.11: Comparison of different methods on the FLD data set. Failure rate are given in percentage. We report the median and best run behavior of the worse and best data pre-processing strategies for each baseline network. The best results are highlighted in bold.

	Left Eye	Right Eye	Nose	Left Mouth	Right Mouth	Average
Sun et al.[166]	**0.67**	0.33	**0.00**	1.16	**0.67**	**0.57**
VGG-16 **ϵ-50**	11.65/7.23	9.64/6.83	16.47/13.65	10.04/10.04	12.45/8.43	12.05/9.24
ResNet-50 **ϵ-50**	10.44/4.02	6.83/4.02	5.22/7.63	6.43/6.02	6.83/5.22	7.15/5.38
VGG-16 **ϵ-0**	1.20/0.80	0.40/**0.00**	3.21/2.41	3.61/2.81	3.61/2.41	2.41/1.69
ResNet-50 **ϵ-0**	0.80/1.20	**0.00**/**0.00**	**0.00**/0.40	2.01/**0.80**	1.20/1.20	0.80/0.72

Table 6.12: Comparison of different methods on the Parse database. Strict PCP scores are reported. We report the median and best run behavior of the worse and best data pre-processing strategies for each baseline network. The best results are highlighted in bold.

	Head	Torso	Upper Legs	Lower Legs	Upper Arms	Lower Arms	Full Body
Andriluka et al [4]	72.7	86.3	66.3	60.0	54.0	35.6	59.2
Yang & Ramanan [188]	82.4	82.9	68.8	60.5	63.4	42.4	63.6
Pishchulin et al. [133]	77.6	90.7	80.0	70.0	59.3	37.1	66.1
Johnson et al. [80]	76.8	87.6	74.7	67.1	67.3	45.8	67.4
OuYang et al. [131]	89.3	89.3	78.0	72.0	**67.8**	47.8	71.0
Belagiannis et al. [9]	**91.7**	**98.1**	**84.2**	**79.3**	66.1	41.5	**73.2**
VGG-16 **α-μPad**	68.85/70.49	85.25/83.61	77.05/79.51	63.11/63.93	45.90/45.08	40.16/42.62	60.66/61.64
ResNet-50 **α-μPad**	57.4/65.6	68.9/73.8	71.3/74.6	55.7/62.3	39.3/45.1	36.1/41.0	53.1/58.5
VGG-16 **$\not\alpha$-0Pad**	77.05/68.85	90.16/88.52	80.33/81.15	63.11/67.21	50.82/52.46	44.26/**50.00**	64.43/65.90
ResNet-50 $\mathbf{120 \times 80}$	67.2/68.9	78.7/80.3	77.9/77.9	65.6/63.9	45.1/50.0	46.7/45.9	61.6/62.1

roring results are consistent over data sets and networks, and therefore we recommend to systematically use mirroring for training, the results on data pre-processing require more detailed discussion. Indeed, one must be extremely careful when choosing the pre-processing on a data set. In the three cases we can see how the performance can present strong variations depending on the pre-processing used. Even more dangerous, when comparing the two architectures, the conclusions on which is best can change depending on the pre-processing. More generally, the superiority of a method (potentially under review) with respect to the state-of-the-art may strongly depend on the data pre-processing. Ideally, the community should establish standard ways to pre-process data so as to avoid unsupported conclusions. In the case of Biwi, fixing one pre-processing strategy would either bias the decision towards ResNet-50 ($\mathbf{64-Re}$ or $\mathbf{64-0Pad}$) or towards VGG-16 ($\mathbf{224}$). But the fairest comparison would be ResNet-50 with $\mathbf{64-Re}$ against VGG-16 with $\mathbf{224}$. This indicates a clear interest on discussing different pre-processing strategies when presenting a new data set/architecture.

Table 6.13: Comparison of different methods on the Biwi head-pose database. Mean absolute errors are given in degrees. The best results are highlighted in bold. The superscript † denotes the use of extra training data (see [105] for details). We report the median and best run behavior of the worst and best data-pre-processing strategies for each baseline network. The best results are highlighted in bold.

	Pitch	Yaw	Roll	Mean
Liu et al.[105]	6.1	6.0	5.7	5.94
Mukherjee et al.[119]	5.18	5.67	–	5.43
Drouard et al.[44]	5.43	4.24	4.13	4.60
Lathuiliere et al.[93]	4.68	3.12	3.07	**3.62**
Liu et al.[105]†	4.5	4.3	**2.4**	3.73
VGG-16 **64-0Pad**	10.11/4.75	4.70/3.82	4.16/4.18	6.33/4.25
ResNet-50 **224**	4.73/4.53	2.96/3.49	4.91/4.10	4.20/4.04
VGG-16 **224**	4.51/**4.02**	4.68/3.74	3.22/3.28	4.14/3.68
ResNet-50 **64-Re**	5.98/5.22	2.39/**2.37**	3.93/4.04	4.10/3.88

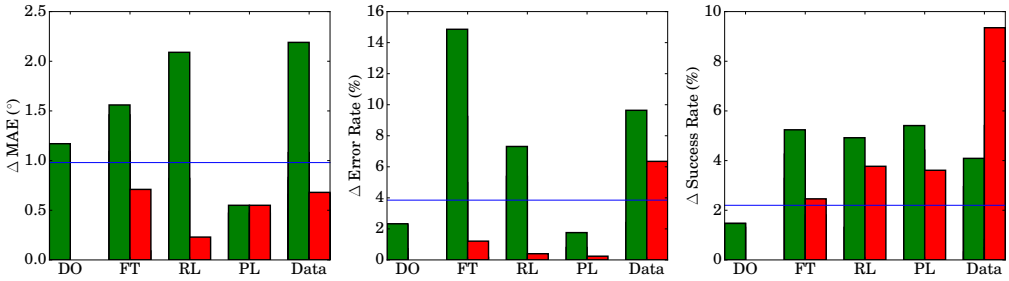
6.7 POSITIONING OF THE STANDARD BASELINES

The aim of this section is to position the standard baselines used in this chapter (i.e. VGG-16 and ResNet-50) with respect to the state-of-the-art. Importantly, the point here is not to outperform all previous methods that are specifically designed for each of the studied tasks, but rather to understand how far or close a vanilla deep regression network is from the state-of-the-art. Another question is whether or not a "correctly fine-tuned base network" (meaning having chosen the optimal network variant and pre-processing strategy) is able to outperform some of the methods in the literature.

Also, in this experimental protocol, we will select the metric proper to each problem instead of the MAE, so as to be able to compare with the state-of-the-art. The experimental results are directly taken from the respective papers (see the tables below). Since, in most of the cases the experimental protocol (in terms of runs and statistical measure) lacks of details. Since in the previous section we observed that the pre-processing strategy is crucial for the performance, we report the results of the two baseline architectures (Table 6.1) combined with the best and worst data pre-processing strategy for each architecture and each problem. For each of these four combinations of base architecture and pre-processing strategy, we run five network trainings and report the performance of the "best" and "average" runs (selected from the overall performance of the problem at hand, i.e., last column of the tables below). Consequently, it is possible that for a particular sub-task the "average" run has better performance than the "best" run.

Table 6.11 reports the failure percentage of different methods on the FLD dataset where errors larger than 5% of the bounding box width are counted as failures. First of all, a correctly fine-tuned simple regression network is in competition with the state-of-the-art. Even if in average the best vanilla regressors do not outperform the literature, they obtain quite decent results (e.g. the average run with the optimal strategy for ResNet-50 is only 0.23% worse than the state of the art).

Regarding the Parse dataset, Table 6.12 reports the results of six methods in the liter-



(a) Biwi **(b)** FLD **(c)** Parse

Figure 6.4: Improvement in performance between the best and worst median runs for each configuration. The three plots show the improvement on Biwi (a), FLD (b) and Parse (c) considering four network variants used and the data pre-processing (*Data*). Horizontal blue lines indicate the improvement obtained by recent approaches when they overcame preceding methods (details in the text).

ature. Performance is measured using the strict PCP (Percentage of Correctly estimated Parts) score. According to strict PCP, a limb is considered as correctely estimated if the distances between the estimated and true joint locations are smaller than 50% of the limb length. The vanilla deep regression do not outperform the best method in the state-of-the-art. However, it is important to notice that regarding the Head, Torso, Upper Legs and Lower Legs, the best vanilla regressor outperforms half of the previous methods and in the case of Lower Arms, it outperforms the state of the art. In average (Full Body), vanilla deep regressors compete with half of the methods in the literature. We believe this is a very interesting result from two perspectives. On the one side, vanilla deep regressors may compete with the state-of-the-art in different tasks as long as they are correctly fine-tuned. On the other side, methods specifically designed for a problem may often outperform standard baselines, and therefore it is worth pursuing research in most applications. In other words, standard baselines may offer a good starting point, but developing problem-specific methods is worth to push the performances.

Finally, Table 6.13 reports the results on the Biwi dataset using MAE. Correctly fine-tuned regression network are clearly competing with the state-of-the-art (even with the method using extra training data). Overall, the best simple regressor is extremely close to the state-of-the-art on the Biwi dataset.

6.8 Overall Discussion

Figure 6.4 displays the difference, in terms of problem-specific metrics (see section 6.7), between the best and worst median runs per configuration. More precisely, we compute the metric for each one of the 5 runs and, from there, compute the median for each configuration. Finally, we display the difference between the best and worst medians. The main goal of this figure is to visualize the impact in performance of each network variant and the data-preprocessing per network and problem. We exclude those unequivocal cases where one particular configuration offers a clear and systematic improvement or

deterioration with respect to other configurations. These are: batch normalization (see Table 6.2), $\mathbf{CB^5}$ when fine-tunning ResNet-50 (see Table 6.5) and mirroring (see Table 6.9). Another element to consider is the improvement that can be achieved when the network variants are properly selected (up to 2° MAE in Biwi, up to 14% error rate in FLD, and up to 6% success rate in Parse). Although, in Figure 6.4, the margins of improvement may seem small in some cases, we are dealing with performance increases that, in many cases, would be sufficient to clearly overcome prior art. In order to give a visual reference in this regard, for each problem, we add a horizontal line that indicates the improvement obtained by recent approaches when they overcame preceding methods. Such lines are drawn by computing the difference in performance between the best performing method in Tables 6.13, 6.11 and 6.12, for Biwi, FLD and Parse, respectively, and the second best performing method in the corresponding original papers ([93], [9], and [166]). We do not suggest that, in a particular task, the precise adjustment of, for instance, FT will provide results superior to the state of the art. We only intend to give a reference of the magnitude of the potential performance improvement (or deterioration!) between different variants of the same network, and claim that this potential improvement is far from being negligible.

The aspect that attracts the most immediate attention is that the margin of improvement in VGG-16 is generally larger than in ResNet-50, as shown by the gap between the top of the green and red columns in Figure 6.4 (with the exceptions of PL in Biwi and data-preprocessing in Parse). Regarding the network variants, we conclude that VGG-16 obtains highly different results while, on the contrary, the behavior of ResNet-50 is generally more stable. From a practical point of view, a larger improvement can be expected from the careful configuration of the VGG-16 variants. With respect to the data pre-processing, the improvement may also be quite large, and which network is subject to larger improvement is highly dependent on the problem at hand.

Generally speaking, the most critical factors seem to be FT, RL and data-preprocessing. The only exception is VGG-16 in Parse, where PL offers a larger improvement than data pre-processing. As a consequence, it is preferable to invest time trying different FT and RL strategies than different variants of PL or DO, which appear to be the elements with less difference between the best and worst configurations.

Finally, we observe that the impact of a correct data pre-processing is crucial for the performance of vanilla deep regression networks. Moreover, the margin of improvement highly depends both on the problem at hand and on the architecture. In addition, given that for some datasets the optimal data-preprocessing strategy depends on the network or is undefined, we argue that the pre-processing can have huge impact on the results and extract two conclusions. First, at practical level, we strongly recommend to compare different data pre-processing strategies in order to improve the performance. Second, at a scientific level, our results illustrate that the employed data pre-processing procedures must be carefully detailed when it comes to reporting results. Different pre-processing techniques must be evaluated and carefully explained in scientific papers in order to obtain reliable conclusions and reproducible results.

6.9 CONCLUSION

This chapter presents the first comprehensive study on deep neural architectures for computer vision regression. Indeed, we aim to shade some light into how to use vailla deep regression models in computer vision and which parameters are more important and should be evaluated with more care, as well as how to avoid undesirable situations. To do that, we exploited two base architectures on three different regression computer vision problems. For every configuration tested, we train the network five times, and report confidence intervals of the median of the error. Additionally, we test these results for statistical significance. In this way we provide both an absolute and a relative measure of the performance. The extensive experimental evaluation carried on allowed us to extract some global recommendations (either positive or negative) as well as point the network/data configurations that need further comparison since conclusive evidence was not found.

In addition, we have shown that correctly fine-tuned deep regression networks compete with problem-specific methods in the literature that are entirely devoted to solve only one task. Importantly, the overall set of experiments clearly points out the need of a proper data-preprocessing strategy to obtain results that are meaningful and competitive with the state-of-the-art. This behavior should encourage the community to use very clean experimental protocols that include transparent and detailed descriptions of the different pre-processing strategies used. Ideally, papers should evaluate the impact of different pre-processing strategies when presenting a new dataset or a new method.

CHAPTER 7

CONCLUSION

7.1 SUMMARY

As explained in the Introduction, the robot needs both to detect elementary cues and perform higher level recognition tasks. In this Thesis, we present both high- and low-level recognition models that can be employed in the context of human-robot interaction. We present a high level model able to recognize activities that are performed in group. The proposed model can handle videos with either single or multiple group activities that occur simultaneously. However, in the context of human-robot interaction, a task of vital importance, and prior to recognizing activities, is to observe the people performing those activities. For this reason, we tackle the problem of robot gaze control. The agent is able to autonomously learn how to find people in the environment by maximizing the number of people present in its field of view and favoring people who speak. A synthetic environment is used for pretraining in order to perform transfer learning to the real environment. Neither external sensors nor human intervention are necessary to compute the reward.

In a second time, we focus on methods that solve low-level recognition tasks such as head or human pose. The study of regression problems is done according to three aspects. First, we propose to couple a Gaussian mixture of linear inverse regressions with a neural network. Therfore, we show that the proposed inverse regression model outperforms L_2-based regression models used by most of the state-of-the-art computer vision methods, at least in the case of head-pose estimation. Our model is evaluated on the on the most widely used head-pose dataset. Second, we present a model that is able to deal with noisy annotations at training time. We combine graphical models for robust regression with deep learning in a jointly trainable fashion. The derived optimization procedure alternates between solving the unsupervised task of outlier detection with the supervised task of inlier deep regression. Finally, we present the first comprehensive study on deep neural architectures for regression problems in computer vision. Indeed, we aim to shade some light into which parameters are more important and should be evaluated with more care, as well as how to avoid undesirable behaviors. We exploited two base architectures on three different regression computer vision problems. For every configuration tested,

we report confidence intervals and perform statistical tests. This extensive evaluation allowed us to extract practical recommendations (either positive or negative) when using deep neural architectures for regression problems.

7.2 FUTURE RESEARCH DIRECTIONS

In this final section, I want to present the future directions of research that could come out from my these. For the sake of clarity, I list them by chapter.

- Chapter 2: The proposed model have been implemented using dense trajectories features but the model can be implemented using any features including deep features. It would be interesting to evaluate the proposed model with deep features. It would require some time to select what are the appropriate networks for that task. More interestingly, we could finetune the network parameters using directly the training loss . Another direction of research would consist in changing the model such that the computation time does not explode when the number of activities increases. In the literature, most of works focus on a limited set of activities but this limitation may be problematic in real environment scenarios where the set of possible human activity is extremely large.
- Chapter 3: Our gaze control model could be extended by modeling better group behaviors including features such as speech turn-taking, or head-pose and gaze of the participants in a conversation or social meeting. In this scenario, we would need to automatically detect voice activity as well as to estimate head poses and gazes. We could then generate synthetic data following this group behavior model and consequently improve the decision taken by the robot. Another possible improvement of the existing approach could imply the consideration of each person's identity by combining our model with a multi-person tracker. More precisely, instead of considering simple joint heatmaps as observations, we could have a set of joint heatmaps (1 per person) with temporal consistency. This is a challenging issue since the dimension of the observations would be dependent on the number of people in the scene.
- Chapter 4: It would be interesting to evaluate our method on other computer vision problems, like facial keypoint detection or full body pose estimation, and extend the type of distributions used in our mixtures, as for example t-distributions to make the model more robust to outliers. It would be an interesting way of combining this work with the robust formulation of chapter 5.
- Chapter 5: We presented a robust formulation for deep regression that could be extended in two ways. First, it would be interesting to adapt this Gaussian-uniform mixture model to the classification problem. Consequently, we could detect mislabelled points and even points that do not belong to any class. That would be an interesting property since the robust classification models in the literature can only deal with permutation between labels. Secondly, in our proposal we alternate between

an outlier detection step and a weighted backpropagation step. For the first step, it would be interesting to compare different standard outlier detection algorithms to compare their performances in the context of deep learning.

- Chapter 6: From the experiments presented in this chapter, we extracted general guidelines for regression tasks. As future works, we could of course increase the number of parameters to adjust and add supplementary datasets to consolidate our recommendations even if it would be time consuming, computationally intensive, and tedious. We hope this paper will encourage people to perform similar study for other deep learning problems such as classification, segmentation, object detection or speech recognition. Moreover we encourage people to perform similar statistical analysis when proposing new approaches. Finally, this study confirms that many network behaviors are still unexplained and that the community should push towards a better theoretical understanding of deep networks in order to explain better the bevior we highlighted.

CHAPTER A

APPENDIX

A.1 CHAPTER 3: SIMULATED ENVIRONMENT

We consider that the robot can cover the field $[-1,1]^2$ by moving its head, but can only visually observe the people within a small rectangular region $\mathcal{F}_t \subset [-1,1]^2$ centered in position vector $\boldsymbol{\Theta}_t$. The audio observations cover the whole reachable region $[-1,1]^2$. However, the actual robot we use is only able to locate the yaw angle of the sound sources, therefore we decided to solely provide sound observations on the horizontal axis $[-1,1]$. On each episode, we simulate one or two persons moving with random speeds and accelerations within a field $[-\xi,\xi]^2$ where $\xi > 1$. In other words, people can go to regions that are unreachable for the robot. For each simulated person in the current episode, we consider the position and velocity of their head at time t, $\boldsymbol{h}_t = (u_t^h, v_t^h) \in [-\xi,\xi]^2$ and $\dot{h} = (\dot{u}_t^h, \dot{v}_t^h) \in \mathbb{R}^2$, respectively. At each frame, the person can keep moving, stay without moving, or choose another random direction. The details of the simulated environment generator are given in Algorithm 3. In a real scenario, people can leave the scene so, in order to simulate this phenomenon, we consider two equally probable cases when a person is going out horizontally of the field ($v_t^h \notin [-\xi,\xi]$). In the first case, the person is deleted and instantly recreated on the other side of the field ($v_{t+1}^h = -v_t^h$) keeping the same velocity ($\dot{v}_{t+1}^h = \dot{v}_t^h$). In the second case, the person is going back towards the center ($v_{t+1}^h = v_t^h$ and ($\dot{v}_{t+1}^h = -\dot{v}_t^h$)). A similar approach is used when a person is going out vertically except that we do not create new persons on top of the field because that would imply the unrealistic sudden appearance of new legs within the field. Figure A.1 displays a visual representation of the different fields (or areas) defined in our synthetic environment, and Figure 3.3 shows an example of a sequence of frames taken from the synthetic environment and used during training.

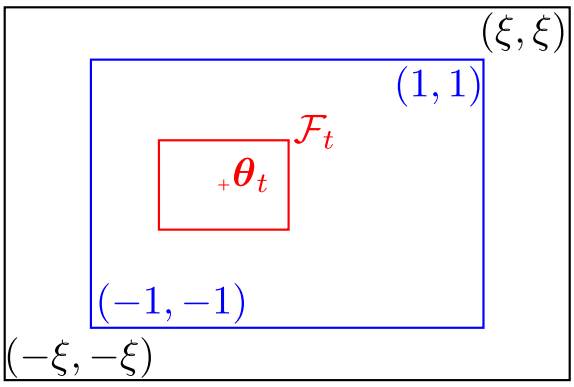


Figure A.1: Diagram showing all fields used in the proposed synthetic environment. The robot's field of view (in red) can move within the reachable field (in blue), whereas the participants can freely move within a larger field (in black).

Moreover, in order to favor tracking abilities, we bias the person motion probabilities such that a person that is faraway from the robot head orientation has a low probability to move, and a person within the field of view has a high probability to move. Thus, when there is nobody in the field of view, the robot cannot simply wait for a person to come in. On the contrary, the robot needs to track the persons that are visible. More precisely,

we consider 4 different cases. First, when a person has never been seen by the robot, the person does not move. Second, when a person is in the robot field of view ($\boldsymbol{h}_t \in \mathcal{F}_t$), they move with a probability of 95%. Third, when the person is further than a threshold $\tau \in \mathbb{R}$ from the field of view ($||\boldsymbol{h}_t - \boldsymbol{\Theta}_t||_2 > \tau$), the probability of moving is only 25%. Finally, when the person is not visible but close to the field of view ($||\boldsymbol{h}_t - \boldsymbol{\Theta}_t||_2 < \tau$ and $\boldsymbol{h}_t \notin \mathcal{F}_t$), or when the person is unreachable ($\boldsymbol{h}_t \in [-\xi, \xi] \backslash [-1, 1]$), this probability is 85%. Regarding the simulation of missing detections, we randomly ignore some faces (following a Bernoulli distribution) when computing the face features. Concerning the sound modality, we randomly choose between the following cases: 1 person speaking, 2 persons speaking, and nobody speaking. We use a Markov model to enforce continuity in the speaking status of the persons, and, similarly to visual observations, we simulate wrong SSL observations employing a Bernoulli distribution.

From, the head position, we need to generate the position of all body joints. To do so, we propose to collect a set $\mathcal{P}$ of poses from an external dataset (the AVDIAR dataset [56]). We use a multiple person pose estimator on this dataset and use the detected poses for our synthetic environment. This task is not trivial since we need to simulate a realistic and consistent sequence of poses. Applying tracking to the AVDIAR videos could provide good pose sequences, but we would suffer from three major drawbacks. First, we would have a tracking error that could affect the quality of the generated sequences. Second, each sequence would have a different and constant size, whereas we would like to simulate sequences without size constraints. Finally, the number of sequences would be relatively limited. In order to tackle these three concerns, we first standardize the output coordinates obtained on AVDIAR. Considering the pose p_t^n of the nth person, we sample a subset $\mathcal{P}_t^M \subset \mathcal{P}$ of M poses. Then, we select the closest pose to the current pose: $p_{t+1}^n = \underset{p \in \Pi}{\operatorname{argmin}}\, d(p, p_t^n)$ where

$$d\left(\begin{pmatrix} u_1 \\ v_1 \\ s_1 \end{pmatrix}, \begin{pmatrix} u_2 \\ v_2 \\ s_2 \end{pmatrix}\right) = \frac{1}{\sum_{j=1}^{J} s_1^j s_2^j} \sum_{j=1}^{J} (s_1^j s_2^j) \sqrt{(u_1^j - u_2^j)^2 + (v_1^j - v_2^j)^2} \tag{A.1}$$

This distance is designed to face poses with different number of detected joints. It can be interpreted as an L_2 distance weighted by the number of visible joints in common. The intuition behind this sampling process is that when the size M of $\mathcal{P}_t^M$ increases, the probability of obtaining a pose closer to p_t^n increases. Consequently, the motion variability can be adjusted with the parameter M in order to obtain a natural motion. With this method we can obtain diverse sequences of any size.

Data: $\mathcal{P}$: a set of poses, δ: time-step
σ: velocity variance, M: pose continuity parameter

Randomly chose N in $[1..3]$.
for $n \in [1..N]$ **do**
  Initialize $(\boldsymbol{h}_0^n, \dot{\boldsymbol{h}}_0^n) \sim \mathcal{U}([-1,1])^2 \times \mathcal{U}([-1,-0.5] \bigcup [0.5,1])^2$.
  Randomly chose p_0^n in $\mathcal{P}$.
end
for $t \in [1..T-1]$ **do**
  for $n \in [1..N]$ **do**
    Randomly chose $motion \in \{Stay, Move\}$
    if $motion = Move$ **then**
      if $\boldsymbol{h}_t^n \notin [-\xi, \xi]^2$ **then**
        The person is leaving the scene.
        See section 3.5.
      else
        $\boldsymbol{h}_{t+1}^n \leftarrow \boldsymbol{h}_t^n + \delta(\dot{\boldsymbol{h}}_t^n + \mathcal{N}((0,0), \sigma))$.
        $\dot{\boldsymbol{h}}_{t+1}^n \leftarrow \frac{1}{\delta}(\boldsymbol{h}_{t+1}^n - \boldsymbol{h}_t^n)$
      end
    else
      $\boldsymbol{h}_{t+1}^n \leftarrow \boldsymbol{h}_t^n$
      $\dot{\boldsymbol{h}}_{t+1}^n \sim \mathcal{U}([-1,-0.5] \bigcup [0.5,1])^2$
    end
    Draw $\mathcal{P}_t^M$, a random set of M elements of $\mathcal{P}$
    $p_{t+1}^n \leftarrow \underset{p \in \mathcal{P}_t^M}{\operatorname{argmin}}\, d(p, p_t^n)$
  end
end

Algorithm 3: Generation of simulated moving poses for our synthetic environment.

A.2 CHAPTER 5: FASHION LANDMARK DETECTION

In section 5.4.1, we presented experiments on the fashion landmark detection problem. In Figures A.2 to A.4, we show training examples containing at least one landmark that DeepGUM considers as outlier. These landmarks correspond to three different scenarios:

- Figure A.2 shows images containing (i) either wrong annotations (e.g. last two images of the last row), (ii) ill-posed cases such as more than one clothe per image or (iii) challenging images (i.e. unusual clothing items like the third and last images of the fourth row).
- Figure A.3 shows images in which one or more landmarks are visually occluded.
- Figure A.4 shows images containing inlier landmarks wrongly classified as outliers by DeepGUM.

A.3 CHAPTER 5: AGE ESTIMATION

In section 5.4.4, we presented experiments on the age estimation task. Figures from A.5 to A.7 display three different groups of images depending on the probability of being inlier that DeepGUM assigns to each age annotation:

- Figure A.5 shows randomly selected images with a high probability of being outliers according to DeepGUM ($r_n < 0.33$). Even if some of the results could be debatable, we argue that most of the annotations (displayed below the image) are incorrect. DeepGUM correctly performs the task for which it was designed.
- Figure A.6 displays randomly selected images for which the network has trouble deciding between inlier and outlier ($0.33 < r_n < 0.66$). Even more, for most of these images it is quite hard to decide whether the annotation is correct or not.
- Figure A.7 shows randomly selected images that are considered by DeepGUM as inliers ($0.66 < r_n$). Indeed, the annotation below each image looks correct in most of the cases.

A.4 CHAPTER 5: HEAD POSE ESTIMATION

In section 5.4.3, we presented experiments on the head pose estimation task. We illustrated the benefit of our propoal that robustly detect outliers at training time, Figure A.8 shows images from the McGill dataset the DeepGUM considered as outlier. In these examples, many clear outliers appear. For some images, it is difficult to say if the annotation is good even for a human annotator.

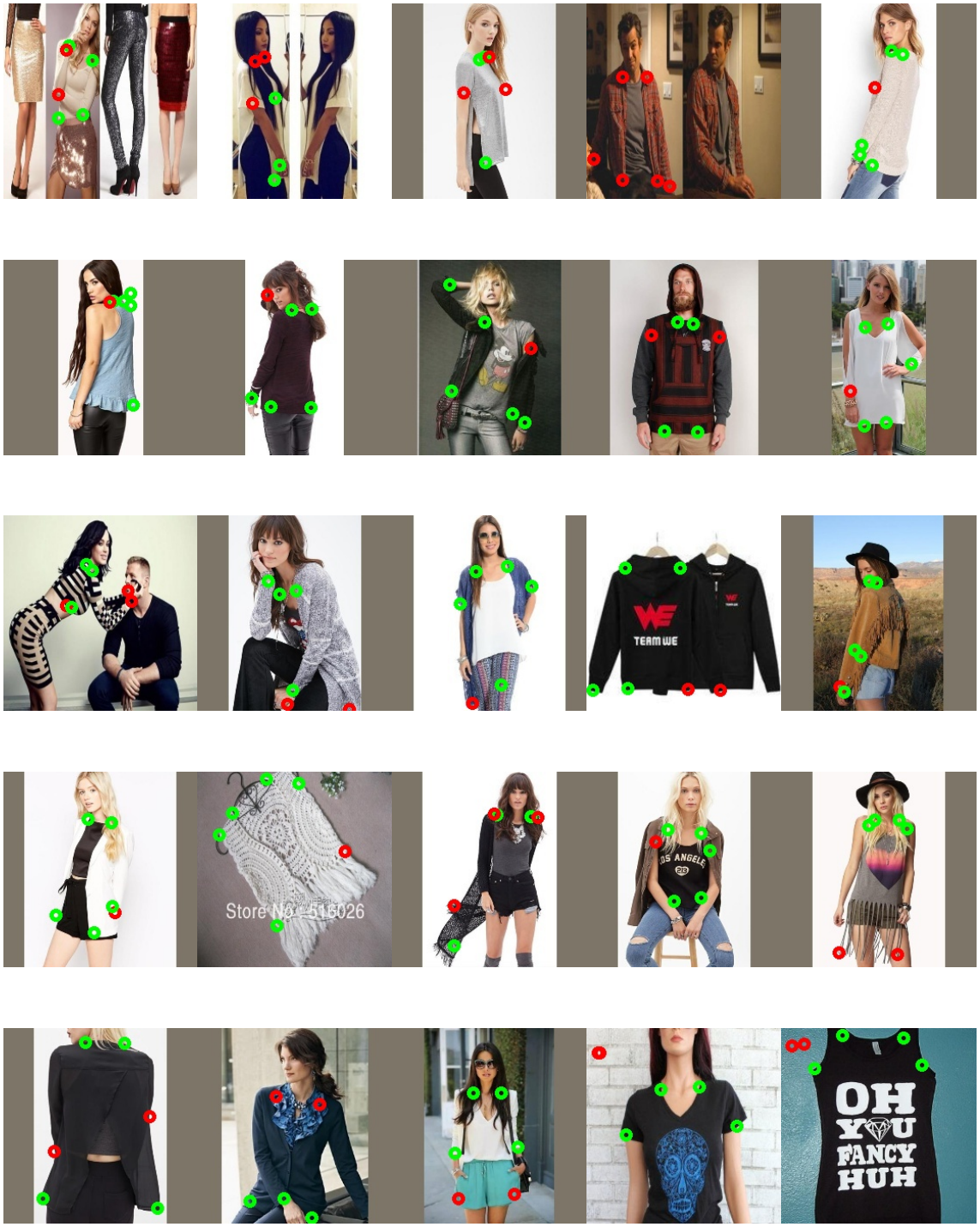


Figure A.2: Example of images from the Fashion Landmark Dataset: landmarks detected as outliers by DeepGUM are shown in red, while inliers are shown in green.

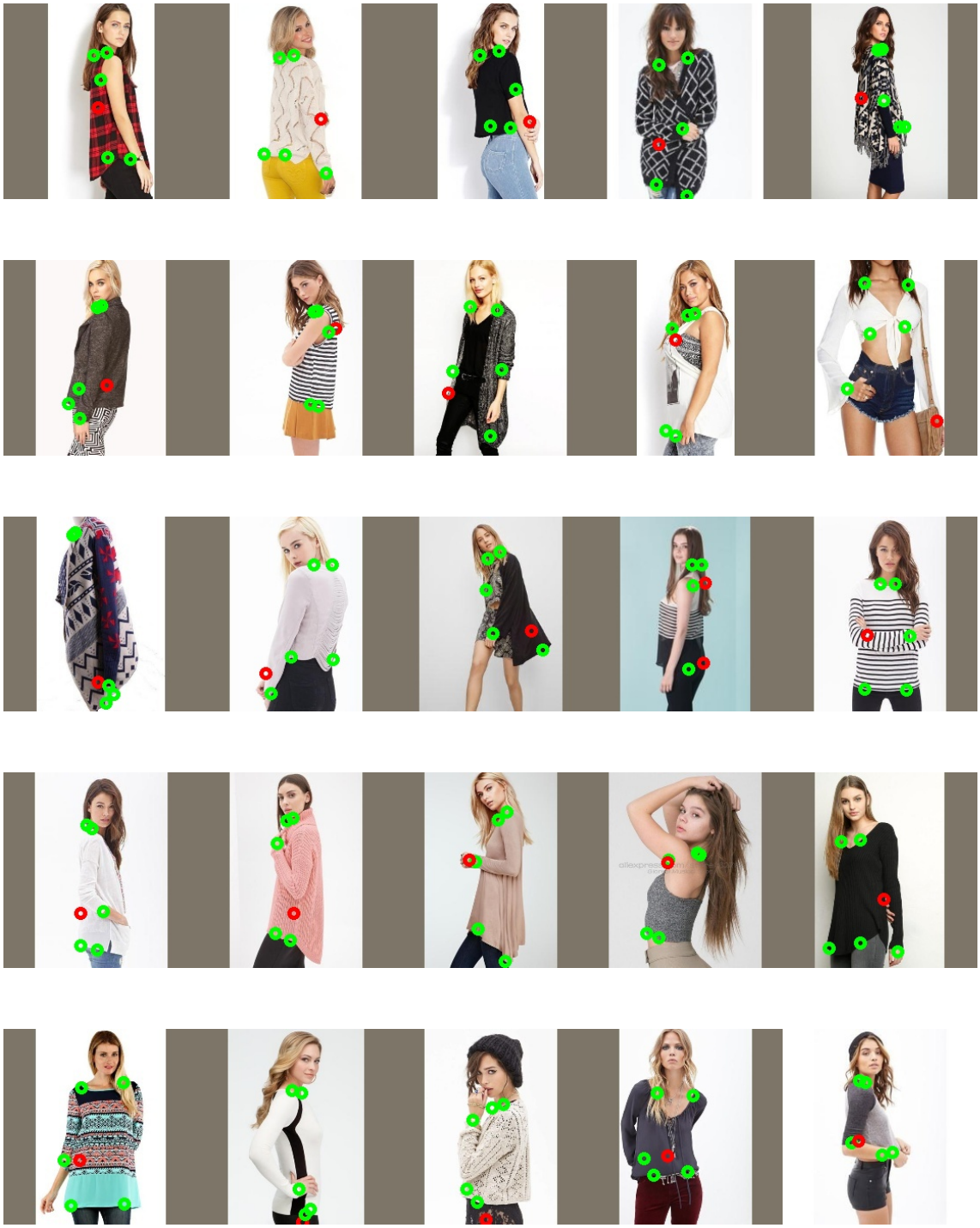


Figure A.3: Example of images from the Fashion Landmark Dataset: landmarks detected as outliers by DeepGUM are shown in red, while inliers are shown in green. In all these images, the detected outliers correspond to occluded landmarks.

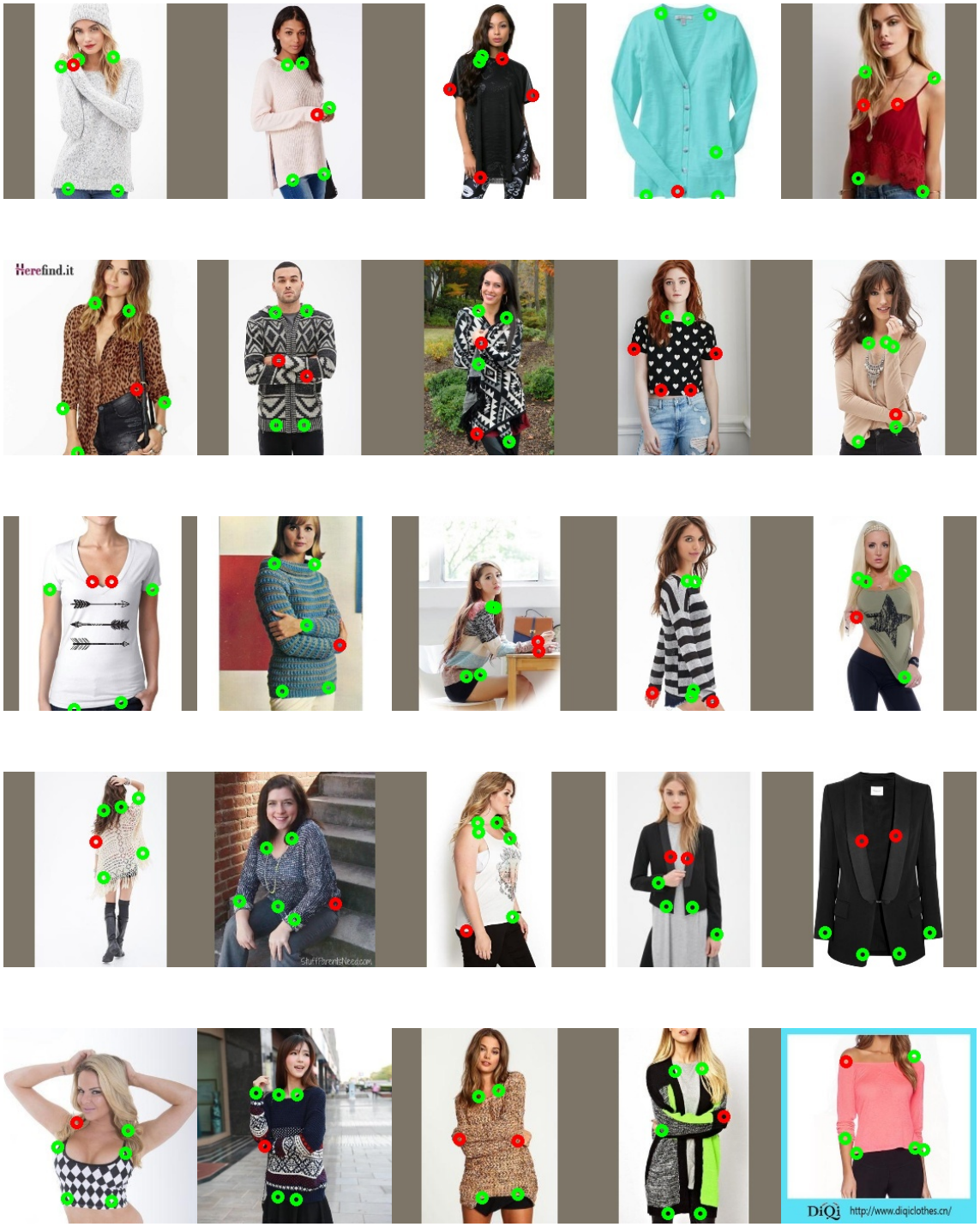


Figure A.4: Example of images from the Fashion Landmark Dataset: landmarks detected as outliers by DeepGUM are shown in red, while inliers are shown in green. The red landmarks correspond to inliers wrongly classified as outliers.

Figure A.5: Sample images of the CACD dataset estimated as outliers during training ($r_n < 0.33$). The label below each image is the annotated age together with the r_n at the end of the training of DeepGUM.

Figure A.6: Sample images of the CACD dataset with high outlier uncertainty ($0.33 < r_n < 0.66$). The label below each image is the annotated age together with the r_n at the end of the training of DeepGUM.

Figure A.7: Sample images of the CACD dataset estimated as inliers during training ($r_n > 0.66$). The label below each image is the annotated age together with the r_n at the end of the DeepGUM training.

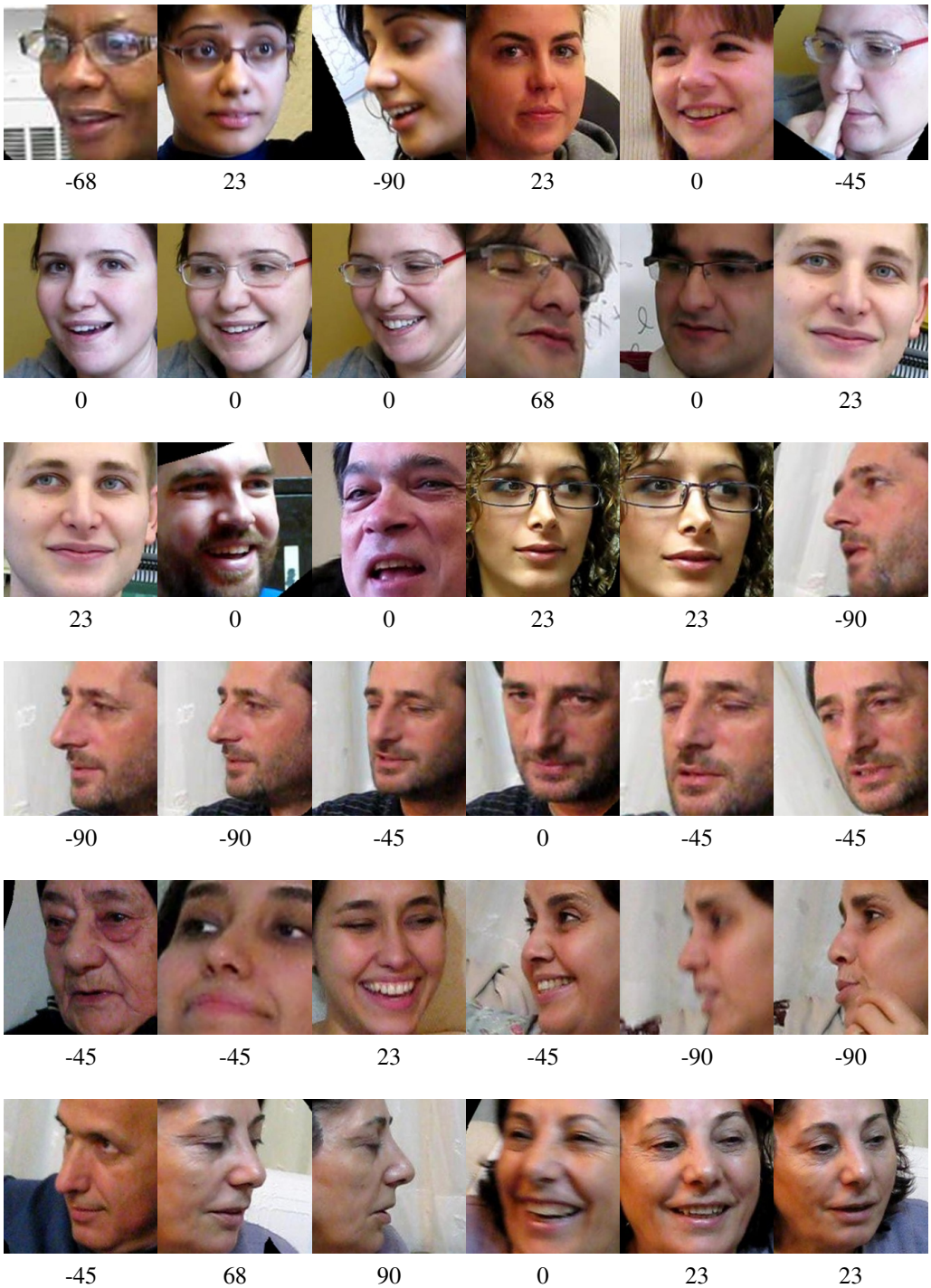


Figure A.8: Sample images from the McGill dataset considered as outliers during training ($r_n < 10^{-4}$). The label below each image is the angle included in the annotation.

A.5 CHAPTER 5: FACIAL LANDMARK DETECTION

Section 5.4.4 reports experiments on the facial landmark detection (FLD) task. We showed the benefit of using DeepGUM, a robust regression approach able to detect outliers at training time, under the presence of different kinds of corrupting outliers. Figure A.9 shows images from FLD corrupted with the *l-UGO* strategy and 30% of outliers (thus these images correspond to the point of the red curve in Figure 5.a in the main manuscript with x-axis value equal to 30%). Superposed to the images we can see circles and crosses for DeepGUM and BiWeight respectively, located at the (corrupted) annotations. The color indicates whether each method detects the annotated landmark as an outlier (red) or inliner (green). In the case of BiWeight, since the method is naturally coordinate-wise, there is a vertical and horizontal lines, denoting whether the vertical and horizontal coordinates respectively are detected as outliers. First of all we remark that almost all of the uncorrupted landmarks are detected as inliers by both methods. This corresponds to the 100% outlier precision (i.e. no inliers are classified as outliers) in the curves of Figure 5.a. Regarding the detection of outliers, we can see that BiWeight classifies many outliers as inliers (lower outlier recall with respect to DeepGUM, as in Figure 5.a). We can also observe that, because BiWeight works coordinate-wise, some of the landmarks are detected as outliers horizontally and not vertically and vice-versa. For instance in the first row fifth column, we can see that the nose landmark is wrongly annotated as close to the eyebrow. Horizontally the error is not big, and therefore BiWeight classifies this as a horizontal inlier and vertical outlier. However, this is wrong, because ideally we would not want to use the eyebrow as a nose samples. Other examples confirm this behavior and explain, not only why the recall of BiWeight is lower than DeepGUM, but also uncover one of the reasons of the difference in performance.

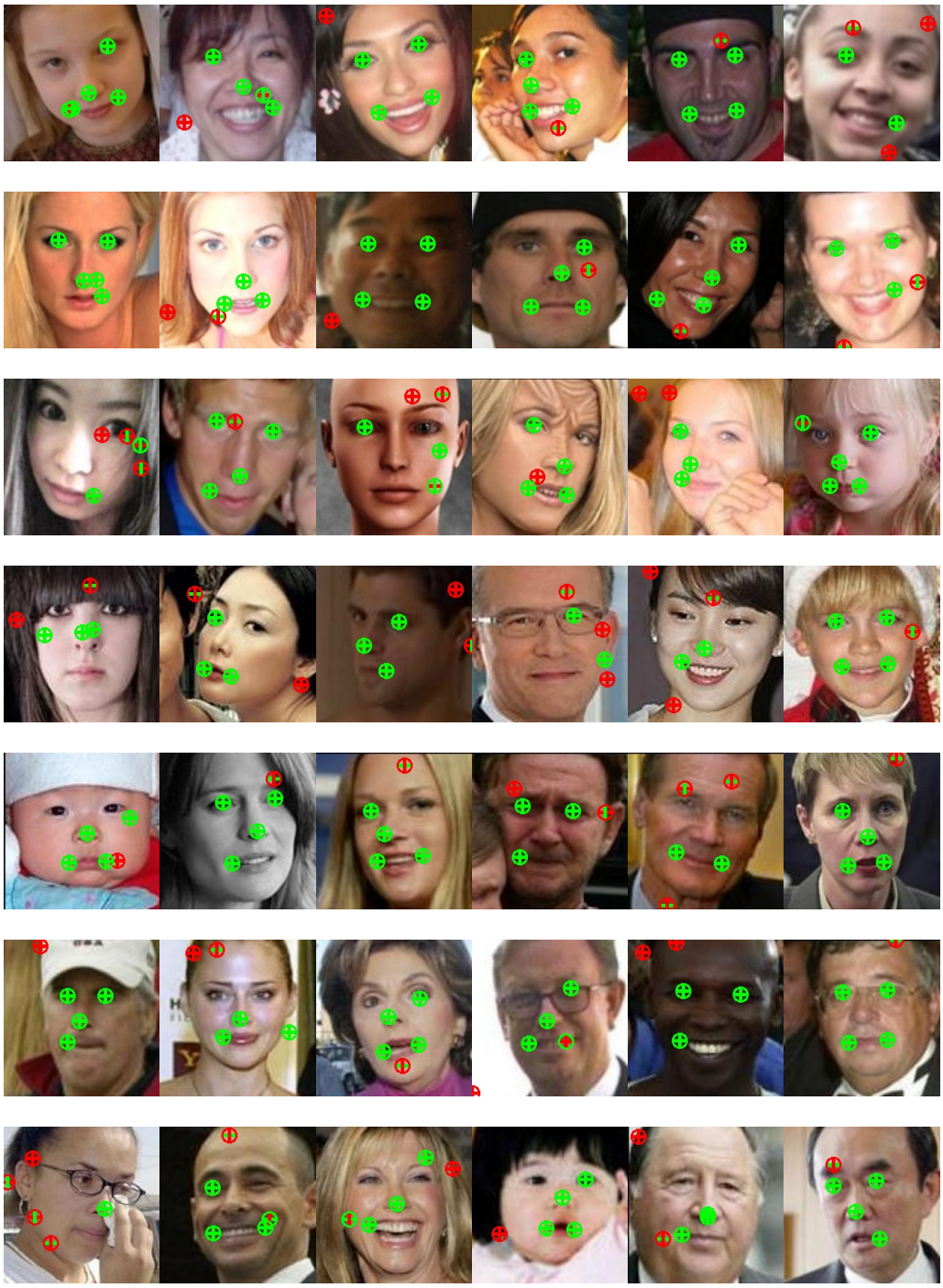


Figure A.9: Sample images of the facial landmark detection problem: landmarks that are considered as outliers during the training of DeepGUM are displayed in red while the ones classified as inliers are displayed in green.

A.6 Teaching, Internship and Collaborations

During the first year of my PhD, I taught at UFRAPS (Sport Department of University of Grenoble-Alpes (UGA)). I had to organize a two-semester course about basic computer tools for first year students. This training should also give the possibility to volunteers to prepare the C2I examination (a certificate attesting the computer related skills required for the continuation of studies and the professional integration). During this period, I was supervised by Gal Mermet. Importantly, since the number of students was very large (800 in Grenoble + 300 Valence) and as the maximal capacity of the UFRAPS amphitheaters is 100 places, I had to teach 8 times each lecture. The courses in Valence were given by a substitute teacher. Furthermore, I set up an online platform to help the students to work from home. In this task, I benefited enormously from the advices and technical support provided by Pierre Gillois who is the Grenoble C2I manager.

During the second year, I taught in the Science and Technology Department (DLST) at UGA. My courses were part of the Physics, Mathematics and Mechanics course (PMM). This course is intended for students who want to have a solid training in both Physics and Mathematics. My teaching, entitled Introduction to Unix and to Programing (Inf 111) aimed to present the architecture of a computing environment (Unix and Bash), imperative programming with C language and basics of algorithmics, coding and testing.

During the last year, I helped the Perception team to organize a data challenge for the UGA master students. The goal of this challenge was to develop an audio-visual diarization model. The code I wrote for this challenge is available on gitHub[1].

Two master internships in the Perception Team were related to my Thesis and co-advised between Prof. Radu Horaud, Pablo Mesejo and myself. First, Rémi Juge helped me working on the deep mixture model for regression detailed in Chapter 4. Then, in a second time, I co-supervised Sylvain Guy that worked on the problem on visual voice activity detection. In conjunction with Sylvain, I performed deep learning experiments on that problem. Sylvain is nowadays a PhD candidate in the team and we already obtained interesting results that should be ready for publication soon.

In summer 2016, Rafael Muñoz Salinas visited the perception team for two months. After interesting discussions and preleminary experiments, I started to work on the deep mixture model for regression we finally published in [94]. In summer 2017, I visited the MHUG team (Multimedia and Human Understanding Group) headed by Prof. Nicu Sebe. From this visit came out two submitted papers that are not included in this manuscript [156, 162].

A.7 Publications and Submissions

Here are is the list of papers that have been published or submitted during my PhD.

[1] `https://github.com/Stephlat/dataChallengePerception`

ARTICLES INCLUDED IN THIS MANUSCRIPT:

- [92] **Stéphane Lathuilière**, Georgios Evangelidis, and Radu Horaud, Recognition of group activities in videos based on single-and two-person descriptors, In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.
- [94] **Stéphane Lathuilière**, Rémi Juge, Pablo Mesejo, Rafael Munoz-Salinas, and Radu Horaud, Deep mixture of linear inverse regressions applied to head-pose estimation, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [96] **Stéphane Lathuilière**, Benoît Massé, Pablo Mesejo, and Radu Horaud, Deep Reinforcement Learning for Audio-Visual Servoing in Human-Robot Interaction, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [95] **Stéphane Lathuilière**, Benoît Massé, Pablo Mesejo, and Radu Horaud, Neural Network-based Reinforcement Learning for Audio-Visual Gaze Control in Human-Robot Interaction, *Pattern Recognition letters*, 2018.
- [97] **Stéphane Lathuilière**, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud,DeepGUM: Deep Robust Regression with Gaussian-Uniform Mixtures , In *IEEE European Conference of Computer Vision (ECCV)*, 2018.
- [91] **Stéphane Lathuilière**, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud, A Comprehensive Analysis of Deep Regression, *Submitted to TPAMI*, 2018.

OTHER ARTICLES:

- [156] Aliaksandr Siarohin, Enver Sangineto, **Stéphane Lathuilière**, and Nicu Sebe, Deformable gans for pose-based human image generation, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [162] Evgeny Stepanov, **Stéphane Lathuilière**, Shammur Absar Chowdhury, Arindam Ghosh, Radu-Laurentiu Vieriu, Nicu Sebe, and Giuseppe Riccardi, Depression severity estimation from multiple modalities, *IEEE International Conference on E-health Networking, Application & Services (Healthcom)*, 2018.

OTHER RESOURCES:

- The code of [94] is publicly available at `https://team.inria.fr/perception/research/dmlir/`.
- The code of [96], [97] and [91] is available at `https://team.inria.fr/perception/team-members/stephane-lathuiliere/`
- A video showing additional experiments of [96] is available at `https://team.inria.fr/perception/research/neural-reinforcement-learning-for-human-robot-interaction/`.

REFERENCES

[1] Byungtae Ahn, Jaesik Park, and In So Kweon. Real-time head orientation from a monocular camera using deep neural network. In *ACCV*, 2014.

[2] Mohamed R Amer and Sinisa Todorovic. A chains model for localizing participants of group activities in videos. In *ICCV*, 2011.

[3] Mohamed Rabie Amer, Peng Lei, and Sinisa Todorovic. HIRF: Hierarchical random field for collective activity recognition in videos. In *ECCV*, 2014.

[4] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *CVPR*, pages 1014–1021, 2009.

[5] Fabien Badeig, Quentin Pelorson, Soraya Arias, Vincent Drouard, Israel Gebru, Xiaofei Li, Georgios Evangelidis, and Radu Horaud. A distributed architecture for interacting with nao. In *ACM ICMI*, 2015.

[6] Yutong Ban, Xavier Alameda-Pineda, Fabien Badeig, Sileye Ba, and Radu Horaud. Tracking a varying number of people with a visually-controlled robotic head. In *IEEE/RSJ IROS*, 2017.

[7] Jeffrey D Banfield and Adrian E Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 1993.

[8] A. J. Bekker and J. Goldberger. Training deep neural-networks based on unreliable labels. In *ICASSP*, 2016.

[9] Vasileios Belagiannis, Christian Rupprecht, Gustavo Carneiro, and Nassir Navab. Robust optimization for deep regression. In *ICCV*, 2015.

[10] Gleb Beliakov, Andrei V. Kelarev, and John Yearwood. Robust artificial neural networks and outlier detection. Technical report. *CoRR*, 2011.

[11] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[12] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.*, 1994.

[13] M. Bennewitz, F. Faber, D. Joho, M. Schreiber, and S. Behnke. Towards a humanoid museum guide robot that interacts with multiple persons. In *IEEE-RAS*, pages 418–423, 2005.

[14] Michael J Black and Anand Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *IJCV*, 1996.

[15] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *ICCV*, 2017.

[16] Xavier P Burgos-Artizzu, Pietro Perona, and Piotr Dollár. Robust face landmark estimation under occlusion. In *ICCV*, pages 1513–1520, 2013.

[17] Gabriel Bustamante, Patrick Danés, Thomas Forgue, and Ariel Podlubne. Towards information-based feedback control for binaural active localization. In *IEEE ICASSP*, 2016.

[18] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *IEEE CVPR*, 2017.

[19] Vijay Chandrasekhar, Jie Lin, Olivier Morère, Hanlin Goh, and Antoine Veillard. A practical guide to CNNs and Fisher Vectors for image instance retrieval. *Signal Processing*, 128:426–439, 2016.

[20] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *JAIR*, 2002.

[21] Bor-Chun Chen, Chu-Song Chen, and Winston H. Hsu. Cross-age reference coding for age-invariant face recognition and retrieval. In *ECCV*, 2014.

[22] W. Choi and S. Savarese. A unified framework for multi-target tracking and collective activity recognition. In *ECCV*, 2012.

[23] Wongun Choi, Yu-Wei Chao, Caroline Pantofaru, and Silvio Savarese. Discovering groups of people in images. In *ECCV*, 2014.

[24] Wongun Choi and Silvio Savarese. Understanding collective activities of people from videos. *IEEE TPAMI*, 2013.

[25] Wongun Choi, Khuram Shahid, and Silvio Savarese. What are they doing?: Collective activity classification using spatio-temporal relationship among people. In *ICCV Workshops*, 2009.

[26] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[27] W. Conover. *Practical Nonparametric Statistics*. Kirjastus: John Wiley and Sons (WIE), 1998.

[28] Pietro Coretto and Christian Hennig. Robust improper maximum likelihood: tuning, computation, and a comparison with other methods for robust Gaussian clustering. *JASA*, 2016.

[29] Armel Cretual and François Chaumette. Application of motion-based visual servoing to target tracking. *IJRR*, 2001.

[30] Francisco Cruz, German I Parisi, Johannes Twiefel, and Stefan Wermter. Multimodal integration of dynamic audiovisual patterns for an interactive reinforcement learning scenario. In *IEEE/RSJ IROS*, 2016.

[31] Frédéric Cupillard, François Brémond, and Monique Thonnat. Group behavior recognition with multiple cameras. In *WACV*, 2002.

[32] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[33] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[34] Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Real-time facial feature detection using conditional regression forests. In *CVPR*, pages 2578–2585, 2012.

[35] A. Deleforge, R. Horaud, Y. Y. Schechner, and L. Girin. Co-localization of audio sources in images using binaural features and locally-linear regression. *IEEE TASLP*, 2015.

[36] Antoine Deleforge, Florence Forbes, Silèye Ba, and Radu Horaud. Hyper-Spectral Image Analysis with Partially-Latent Regression and Spatial Markov Dependencies. *IEEE STSP*, 2015.

[37] Antoine Deleforge, Florence Forbes, and Radu Horaud. High-Dimensional Regression with Gaussian Mixtures and Partially-Latent Response Variables. *Stat Comput*, 2015.

[38] Meltem Demirkus, Doina Precup, James J. Clark, and Tal Arbel. Hierarchical temporal graphical model for head pose estimation and subsequent attribute classification in real-world videos. *CVIU*, 2015.

[39] Zhiwei Deng, Mengyao Zhai, Lei Chen, Yuhao Liu, Srikanth Muralidharan, Mehrsan Javan Roshtkhari, and Greg Mori. Deep structured models for group activity recognition. In *BMVC*, 2015.

[40] Joaqun Derrac, Salvador Garca, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.

[41] Vincent Drouard, Sileye Ba, Georgios Evangelidis, Antoine Deleforge, and Radu Horaud. Head pose estimation via probabilistic high-dimensional regression. In *ICIP*, 2015.

[42] Vincent Drouard, Radu Horaud, Antoine Deleforge, Silèye Ba, and Georgios Evangelidis. Robust head-pose estimation based on partially-latent mixture of linear regressions. *IEEE TIP*, 2016.

[43] Vincent Drouard, Radu Horaud, Antoine Deleforge, Silèye Ba, and Georgios Evangelidis. Robust head-pose estimation based on partially-latent mixture of linear regressions. *IEEE TIP*, 26(3):1428–1440, 2017.

[44] Vincent Drouard, Radu Horaud, Antoine Deleforge, Silèye Ba, and Georgios Evangelidis. Robust head-pose estimation based on partially-latent mixture of linear regressions. *IEEE TIP*, 26(3):1428 – 1440, January 2017.

[45] Vincent Drouard, Radu Horaud, Antoine Deleforge, Silx00E8ye Ba, and Georgios Evangelidis. Robust head-pose estimation based on partially-latent mixture of linear regressions. *TIP*, 2017.

[46] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(7):2121–2159, 2011.

[47] O. J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.

[48] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why Does Unsupervised Pre-training Help Deep Learning? *JMLR*, 11:625–660, 2010.

[49] Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Gool. Random Forests for Real Time 3D Face Analysis. *IJCV*, 2013.

[50] Gabriele Fanelli, Juergen Gall, and Luc Van Gool. Real time head pose estimation with random regression forests. In *CVPR*, pages 617–624, 2011.

[51] R.A. Fisher. *Statistical methods for research workers*. Edinburgh Oliver & Boyd, 1925.

[52] Florence Forbes and Darren Wraith. A new family of multivariate heavy-tailed distributions with variable marginal amounts of tailweight: application to robust clustering. *Statistics and Computing*, 2014.

[53] Alfiia Galimzianova, Franjo Pernus, Bostjan Likar, and Ziga Spiclin. Robust estimation of unbalanced mixture models on samples with outliers. *TPAMI*, 2015.

[54] Junying Gan, Lichen Li, Yikui Zhai, and Yinhua Liu. Deep self-taught learning for facial beauty prediction. *Neurocomputing*, 2014.

[55] Chris Gaskett, Luke Fletcher, Alexander Zelinsky, et al. Reinforcement learning for visual servoing of a mobile robot. In *Australian Conference on Robotics and Automation*, 2000.

[56] Israel Gebru, Sileye Ba, Xiaofei Li, and Radu Horaud. Audio-visual speaker diarization based on spatiotemporal bayesian fusion. *IEEE TPAMI*, 2017.

[57] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2003.

[58] Ali Ghadirzadeh, Judith Bütepage, Atsuto Maki, Danica Kragic, and Mårten Björkman. A sensorimotor reinforcement learning framework for physical Human-Robot Interaction. In *IEEE/RSJ IROS*, 2016.

[59] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *CVPR*, 2014.

[60] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[61] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[62] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE TNNLS*, 28(10):2222–2232, 2017.

[63] Guodong Guo, Yun Fu, Charles R Dyer, and Thomas S Huang. Image-based human age estimation by manifold learning and locally adjusted robust regression. *IEEE TIP*, 17(7):1178–1188, 2008.

[64] Abhinav Gupta, Aniruddha Kembhavi, and Larry S Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE TPAMI*, 2009.

[65] Hossein Hajimirsadeghi, Wang Yan, Arash Vahdat, and Greg Mori. Visual recognition by counting instances: A multi-instance cardinality potential kernel. In *CVPR*, 2015.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[67] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[68] Yosef Hochberg. A Sharper Bonferroni Procedure for Multiple Tests of Significance. *Biometrika*, 75(4):800–802, 1988.

[69] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[70] Derek Hoiem, Alexei A Efros, and Martial Hebert. Putting objects in perspective. *IJCV*, 2008.

[71] BS Holland. An improved sequentially rejective Bonferroni test procedure. *Biometrics*, 43:417–423, 1987.

[72] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

[73] G. Hommel. A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika*, 75(2):383–386, 1988.

[74] P.J. Huber. *Robust Statistics*. Wiley, 2004.

[75] Moustafa Ibrahim, Srikanth Muralidharan, Zhiwei Deng, Arash Vahdat, and Greg Mori. A hierarchical deep temporal model for group activity recognition. In *CVPR*, 2016.

[76] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[77] Vamsi K. Ithapu, Sathya N. Ravi, and Vikas Singh. On architectural choices in deep learning: From network structure to gradient convergence and parameter estimation. *CoRR*, abs/1702.08670, 2017.

[78] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *IJCV*, 2016.

[79] Mihir Jain, Jan C. van Gemert, and Cees G. M. Snoek. What do 15,000 object categories tell us about classifying and localizing actions? In *CVPR*, 2015.

[80] Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *BMVC*, 2010.

[81] Sameh Khamis, Vlad I Morariu, and Larry S Davis. Combining per-frame and per-track cues for multi-person action recognition. In *ECCV*, 2012.

[82] Hyunwoo J. Kim, Brandon M. Smith, Nagesh Adluru, Charles R. Dyer, Sterling C. Johnson, and Vikas Singh. Abundant Inverse Regression Using Sufficient Reduction and Its Applications. In *ECCV*, 2016.

[83] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[84] Hedvig Kjellström, Javier Romero, David Martínez, and Danica Kragić. Simultaneous visual recognition of manipulation actions and manipulated objects. In *ECCV*, 2008.

[85] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *IJRR*, 2013.

[86] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[87] Tian Lan, Yang Wang, Greg Mori, and Stephen N Robinovitch. Retrieving actions in group contexts. In *Trends and Topics in Computer Vision*, 2010.

[88] Tian Lan, Yang Wang, Weilong Yang, and Greg Mori. Beyond actions: Discriminative models for contextual group activities. In *NIPS*, 2010.

[89] Tian Lan, Yang Wang, Weilong Yang, Stephen N Robinovitch, and Greg Mori. Discriminative latent models for recognizing contextual group activities. *IEEE TPAMI*, 2012.

[90] Zhengzhong Lan, Ming Lin, Xuanchong Li, Alex G. Hauptmann, and Bhiksha Raj. Beyond Gaussian pyramid: Multi-skip feature stacking for action recognition. In *CVPR*, 2015.

[91] Stéphane Lathuilière, , Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. A comprehensive analysis of deep regression. *Submitted to TPAMI*, 2018.

[92] Stéphane Lathuilière, Georgios Evangelidis, and Radu Horaud. Recognition of group activities in videos based on single-and two-person descriptors. In *IEEE WACV*, 2017.

[93] Stéphane Lathuilière, Rémi Juge, Pablo Mesejo, Rafael Muñoz Salinas, and Radu Horaud. Deep Mixture of Linear Inverse Regressions Applied to Head-Pose Estimation. In *CVPR*, 2017.

[94] Stéphane Lathuilière, Rémi Juge, Pablo Mesejo, Rafael Munoz-Salinas, and Radu Horaud. Deep mixture of linear inverse regressions applied to head-pose estimation. In *IEEE CVPR*, 2017.

[95] Stéphane Lathuilière, Benoît Massé, Pablo Mesejo, and Radu Horaud. Deep reinforcement learning for audio-visual servoing in human-robot interaction. 2017.

[96] Stéphane Lathuilière, Benoît Massé, Pablo Mesejo, and Radu Horaud. Neural network-based reinforcement learning for audio-visual gaze control in human-robot interaction. *Pattern recognition letters*, 2017.

[97] Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. Deepgum: Deep robust regression with gaussian-uniform mixtures. In *IEEE ECCV*, 2018.

[98] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Effiicient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–50. 1998.

[99] Ker-Chau Li. Sliced inverse regression for dimension reduction. *J Am Stat Assoc*, 1991.

[100] Ruonan Li, Rama Chellappa, and S Kevin Zhou. Learning multi-modal densities on discriminative temporal interaction manifold for group activity recognition. In *CVPR*, 2009.

[101] X. Li, L. Zhao, L. Wei, M.-H. Yang, F. Wu, Y. Zhuang, H. Ling, and J. Wang. Deepsaliency: Multi-task deep neural network model for salient object detection. *IEEE TIP*, 2016.

[102] Xiaofei Li, Laurent Girin, Fabien Badeig, and Radu Horaud. Reverberant sound localization with a robot head based on direct-path relative transfer function. In *IEEE/RSJ IROS*, 2016.

[103] Xiaofei Li, Laurent Girin, Radu Horaud, and Sharon Gannot. Multiple-speaker localization based on direct-path features and likelihood maximization with spatial sparsity regularization. *IEEE/ACM TASLP*, 2017.

[104] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Jia Li. Learning from Noisy Labels with Distillation. *arXiv preprint arXiv:1703.02391*, 2017.

[105] X. Liu, W. Liang, Y. Wang, S. Li, and M. Pei. 3D head pose estimation with convolutional neural network trained on synthetic images. In *ICIP*, 2016.

[106] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016.

[107] Ziwei Liu, Sijie Yan, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Fashion Landmark Detection in the Wild. In *ECCV*, 2016.

[108] Aly Magassouba, Nancy Bertin, and François Chaumette. Aural servo: sensor-based control from robot audition. *IEEE TRO*, 2018.

[109] Ricardo A Maronna, Douglas R Martin, and Victor J Yohai. *Robust statistics*. John Wiley & Sons, 2006.

[110] Michael Marszalek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *CVPR*, 2009.

[111] Benoît Massé, Silèye Ba, and Radu Horaud. Tracking gaze and visual focus of attention of people involved in social interaction. *IEEE TPAMI*, 2017.

[112] Peter Meer, Doron Mintz, Azriel Rosenfeld, and Dong Yoon Kim. Robust regression methods for computer vision: A review. *IJCV*, 1991.

[113] Pablo Mesejo, Oscar Ibánez, Enrique Fernández-Blanco, Francisco Cedrón, Alejandro Pazos, and Ana B Porto-Pazos. Artificial neuron–glia networks learning approach based on cooperative coevolution. *International journal of neural systems*, 25(04), 2015.

[114] Shun Miao, Z. Jane Wang, and Rui Liao. A CNN Regression Approach for Real-Time 2D/3D Registration. *IEEE Trans. Med. Imag.*, 2016.

[115] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of convolution neural network advances on the imagenet. *CVIU*, 161:11–19, 2017.

[116] Noriaki Mitsunaga, Christian Smith, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning. *JRSJ*, 2006.

[117] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari With Deep Reinforcement Learning. In *NIPS Deep Learning Workshop*, 2013.

[118] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[119] S.S. Mukherjee and N.M. Robertson. Deep Head Pose: Gaze-Direction Estimation in Multimodal Video. *IEEE MM*, 2015.

[120] Kevin Murphy, Antonio Torralba, and William Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. In *NIPS*, 2003.

[121] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[122] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. Head pose estimation in computer vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2009.

[123] Moin Nabi, Alessio Del Bue, and Vittorio Murino. Temporal poselets for collective activity detection and recognition. In *ICCVW Workshops*, 2013.

[124] C. Nebauer. Evaluation of convolutional neural networks for visual recognition. *IEEE TNN*, 9(4):685–696, 1998.

[125] Peter Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.

[126] Ralph Neuneier and Hans Georg Zimmermann. How to train neural networks. In *Neural Networks: Tricks of the Trade*. Springer, 1998.

[127] N Neykov, Peter Filzmoser, R Dimova, and P Neytchev. Robust fitting of mixtures using the trimmed likelihood estimator. *CSDA*, 2007.

[128] Regina Nuzzo. Scientific method: Statistical errors. *Nature*, 506(7487):150–152, 2014.

[129] Shigeyuki Odashima, Masamichi Shimosaka, Takuhiro Kaneko, Rui Fukui, and Tomomasa Sato. Collective activity localization with contextual spatial pyramid. In *ECCV*, 2012.

[130] Margarita Osadchy, Yann Le Cun, and Matthew L. Miller. Synergistic face detection and pose estimation with energy-based models. *JMLR*, 2007.

[131] Wanli Ouyang, Xiao Chu, and Xiaogang Wang. Multi-source deep learning for human pose estimation. In *CVPR*, pages 2329–2336, 2014.

[132] Emeline Perthame, Florence Forbes, and Antoine Deleforge. Inverse regression approach to robust non-linear high-to-low dimensional mapping. Technical report, INRIA, 2016.

[133] Leonid Pishchulin, Arjun Jain, Mykhaylo Andriluka, Thorsten Thormählen, and Bernt Schiele. Articulated people detection and pose estimation: Reshaping the future. In *CVPR*, pages 3178–3185. IEEE, 2012.

[134] Ronald Poppe. A survey on vision-based human action recognition. *IVC*, 2010.

[135] A. H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro. Robot gains social intelligence through multimodal deep reinforcement learning. In *IEEE Humanoids*, 2016.

[136] Ahmed Hussain Qureshi, Yutaka Nakamura, Yuichiro Yoshikawa, and Hiroshi Ishiguro. Show, attend and interact: Perceivable human-robot social interaction through neural attention Q-network. In *IEEE ICRA*, 2017.

[137] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *ICCV*, 2007.

[138] Deva Ramanan. Learning to parse images of articulated bodies. In *NIPS*, 2007.

[139] Rajeev Ranjan, Vishal M. Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *CoRR*, 2016.

[140] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE TPAMI*, 39:1137–1149, 2015.

[141] Gernot Riegler, David Ferstl, Matthias Ruther, and Horst Bischof. Hough Networks for Head Pose Estimation and Facial Feature Localization. In *BMVC*, 2014.

[142] Gregory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. Lcr-net: Localization-classification-regression for human pose. In *CVPR*, July 2017.

[143] D Rom. A sequentially rejective test procedure based on a modified Bonferroni inequality. *Biometrika*, 77:663–665, 1990.

[144] M. Rothbucher, C. Denk, and K. Diepold. Robotic gaze control using reinforcement learning. In *IEEE HAVE*, 2012.

[145] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *IJCV*, 2016.

[146] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. John Wiley & sons, 2005.

[147] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[148] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[149] Michael S Ryoo and Jake K Aggarwal. Recognition of composite human activities through context-free grammar based representation. In *CVPR*, 2006.

[150] Michael S Ryoo and Jake K Aggarwal. Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In *ICCV*, 2009.

[151] MS Ryoo and JK Aggarwal. Stochastic representation and recognition of high-level group activities. *IJCV*, 2011.

[152] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 2013.

[153] Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. In *NIPS*, pages 4053–4061, 2016.

[154] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local SVM approach. In *ICPR*, 2004.

[155] Pierre Sermanet, David Eigen, Xiang Zhang, Michal Mathieu, Robert Fergus, and Yann Lecun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[156] Aliaksandr Siarohin, Enver Sangineto, Stephane Lathuiliere, and Nicu Sebe. Deformable gans for pose-based human image generation. In *IEEE CVPR*, 2018.

[157] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[158] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[159] Leslie N. Smith and Nicholay Topin. Deep Convolutional Neural Network Design Patterns. *CoRR*, abs/1611.00847, 2016.

[160] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Stat Comput*, 2004.

[161] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958, 2014.

[162] Evgeny Stepanov, Stephane Lathuiliere, Shammur Absar Chowdhury, Arindam Ghosh, Radu-Laurentiu Vieriu, Nicu Sebe, and Giuseppe Riccardi. Depression severity estimation from multiple modalities. *IEEE Healthcom*, 2018.

[163] Jonathan A C Sterne, D R Cox, and George Davey Smith. Sifting the evidence—what's wrong with significance tests?Another comment on the role of statistical methods. *BMJ*, 322(7280):226–231, 2001.

[164] Charles V Stewart. Robust parameter estimation in computer vision. *SIAM Review*, 1999.

[165] Lei Sun, Haizhou Ai, and Shihong Lao. Activity group localization by modeling the relations among participants. In *ECCV*, 2014.

[166] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, 2013.

[167] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, pages 1139–1147, 2013.

[168] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1st edition, 1998.

[169] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[170] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, pages 2818–2826, 2016.

[171] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? *IEEE TMI*, 35(5):1299–1312, 2016.

[172] Andrea L Thomaz, Guy Hoffman, and Cynthia Breazeal. Reinforcement learning with human teachers: Understanding how people want to teach robots. In *IEEE ROMAN*, 2006.

[173] Tijmen Tieleman and Geoffrey Hinton. Rrmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[174] A. Toshev and C. Szegedy. DeepPose: Human Pose Estimation via Deep Neural Networks. In *CVPR*, 2014.

[175] Pavan Turaga, Rama Chellappa, Venkatramana S Subrahmanian, and Octavian Udrea. Machine recognition of human activities: A survey. *IEEE TCSVT*, 2008.

[176] M. Vázquez, A. Steinfeld, and S. E. Hudson. Maintaining awareness of the focus of attention of a conversation: A robot-centric reinforcement learning approach. In *IEEE ROMAN*, 2016.

[177] Bertie Vidgen and Taha Yasseri. P-values: Misunderstood and misused. *Frontiers in Physics*, 4:6, 2016.

[178] Bingjie Wang, Wei Liang, Yucheng Wang, and Yan Liang. Head pose estimation with combined 2D SIFT and 3D HOG features. In *ICIG*, 2013.

[179] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[180] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 1992.

[181] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, pages 80–83, 1945.

[182] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

[183] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015.

[184] Lingxi Xie and Alan L. Yuille. Genetic CNN. In *CVPR*, 2017.

[185] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *CVPR*, 2013.

[186] S. Yan, H. Wang, X. Tang, and T. S. Huang. Learning auto-structured regressor from uncertain nonnegative labels. In *CVPR*, pages 1–8, 2007.

[187] Heng Yang, Wenxuan Mou, Yichi Zhang, Ioannis Patras, Hatice Gunes, and Peter Robinson. Face alignment assisted by head pose estimation. In *BMVC*, 2015.

[188] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE TPAMI*, 35(12):2878–2890, 2013.

[189] Bangpeng Yao and Li Fei-Fei. Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses. *IEEE TPAMI*, 2012.

[190] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[191] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How Transferable Are Features in Deep Neural Networks? In *NIPS*, pages 3320–3328, 2014.

[192] Stella X Yu and Jianbo Shi. Multiclass spectral clustering. In *ICCV*, 2003.

[193] Sang-Seok Yun. A gaze control of socially interactive robots in multiple-person interaction. *Robotica*, 35(11):2122–2138, 2017.

[194] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[195] Xiantong Zhen, Zhijie Wang, Ali Islam, Mousumi Bhaduri, Ian Chan, and Shuo Li. Multi-scale deep networks and regression forests for direct bi-ventricular volume estimation. *Med Image Anal*, 2016.

[196] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, pages 2879–2886, 2012.