2020

Fall and activity detection framework on a robotic platform for older person care

Belmonte Klein, Frederico

UNIVERSITY OF
PLYMOUTH

Fall and activity detection framework on a robotic platform for older person care

by

Frederico B. Klein

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Computing, Electronics and Mathematics

November, 2020

Acknowledgements

First I would like thank my lovely family that stayed back home in Brazil and supported me over all those years before I could finally be able to do my Ph.D. I want to thank my father and my mother for imbuing in me the love for science and my brother Júnior for sharing with me his love for technology and sister Natália for sharing with me her love for human sciences. Without them I do not think I would have followed this path.

I would also like to express my gratitude to my director of studies, Angelo Cangelosi for the support and guidance and the whole CRNS team for all their help and for pushing me forward to complete this work. Additionally I would like to thank Ricardo de Azambuja, Karla Štěpánová, Michael Klein and Samantha Adams, for helping me getting started in my PhD and Anna Kharko, Alexandre Antunes, Leszek Pecyna, Oksana Hagen and Pontus Loviken for their support during the final years.

Finally, I must thank the Conselho Nacional de Pesquisa e Tecnologia do Brazil (CNPq) for the support given to me in the form of a full scholarship. This work was supported by CNPq Brazil (scholarship 232590/2014-1).

Authors Declaration

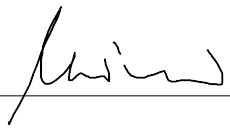
At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award, without prior agreement of the Doctoral College Quality Sub Committee.

No work submitted for a research degree at University of Plymouth may form part of any other degree for the candidate either at the University or at another establishment.

This study was financed with the aid of a studentship form the Ministry of Education of Brazil, awarded by the agency CNPq (scholarship 232590/2014-1).

Relevant scientific seminars and conferences were regularly attended at which work was presented. A conference paper has been accepted and presented, related to the work with fall detection. Another journal paper and four other conference papers were published as collaborations with other researchers.

Word count for the main body of this thesis: **35307**

Signed: 

Date: 2020-11-05

Publications related to PhD work:

KLEIN, FREDERICO B. and Štěpánová; Karla and Cangelosi, Angelo. *Implementation of a Modular Growing When Required Neural Gas Architecture for Recognition of Falls.*, DOI: 10.1007/978-3-319-46687-3_58 Lecture Notes in Computer Science. 2016, pgs. 526–534.

Other publications:

F. B. KLEIN; A. Wilmot; V. F. de Tejada; B.L. Rodriguez; I. Requena; S. Busch; A. Rondepierre; T. Auzeeri; T. Sauerwald; W.F.P. Andrews; H. Rihan; M.P. Fuller; M. F. Stoelen. *Proof-of-concept Modular Robot Platform for Cauliflower Harvesting*, 12th European Conference on Precision Agriculture (ECPA), conference paper and oral presentation. 2019, Montpellier.

Stepanova, Karla; KLEIN, FREDERICO BELMONTE ; Cangelosi, Angelo; Vavrecka, Michal. *Mapping Language to Vision in a Real-World Robotic Scenario*, DOI: 10.1109/TCDS.2018.2819359 IEEE Transactions on Cognitive and Developmental Systems. 2018. pgs. 784–794.

Stepanova, Karla; Hoffmann, M.; Straka, Z.; KLEIN, FREDERICO B.; Cangelosi, A.; Vavrecka, Michal. *Where is my forearm? Clustering of body parts from simultaneous tactile and linguistic input using sequential mapping.*, arXiv. 2017.

de Azambuja, R.; KLEIN, F. B.; Adams, S. V.; Stoelen, M. F.; Cangelosi, A. *Short-term plasticity in a liquid state machine biomimetic robot arm controller*, DOI: 10.1109/IJCNN.2017.7966283 2017 International Joint Conference on Neural Networks (IJCNN) 2017, p. 3399, Anchorage.

de Azambuja, Ricardo; KLEIN, FREDERICO B.; Stoelen, Martin F.; Adams, Samantha V.; Cangelosi, Angelo. *Graceful Degradation Under Noise on Brain Inspired Robot Controllers.*, DOI: 10.1007/978-3-319-46687-3_21 Lecture Notes in Computer Science. 2016, pgs. 195-204.

Abstract

This thesis describes the classification human activity for the purpose of detecting falls, later extended to multiple activities. This was done with the final intention of implementing a robotic companion for older persons that can provide a certain level of automated care in case of some sort of emergency. The complexity of this work, combined with restrictions of the robot, motivated a creation of an infrastructure abstraction to allow deferred (decentralized) processing. The initial work was done by implementing classifiers that work use pre-processed skeleton data extracted from RGB plus Depth (RGB-D) sensors and implements some steps in order to make classification robust to changes. RGB-D classification first focused on falling detection and then extended into general activities which could be classified from skeleton data. A later attempt used Convolutional Neural Networks (CNNs) for classification of video footage of activities. All of those algorithms were modified to output classifications in real time. Results achieved were around 90% in accuracy for a simple fall vs. not fall activity in the TST fall v2 dataset, 70% global combined accuracy for the 12 actions of the Cornell activity dataset (CAD60) using skeleton data and 75% accuracy for the 51 actions on the Human Motion recognition Database with 51 actions as described in [97] (HMDB51), all of those showing close to state-of-the-art performance on datasets. On new activity data based on skeletons and video, however, results were less encouraging with 33.5% accuracy on skeleton data and 37.9% accuracy based on video.

While these results do not allow for a robotic platform to perform action detection currently, the overarching structure of systems necessary to execute it was demonstrated and used successfully, opening up doors for future research using more complex systems.

Contents

List of Figures

List of Tables

Acronyms

ADL activity of daily living.

ANN Artificial Neural Network.

CAD60 Cornell activity dataset.

CNN Convolutional Neural Network.

COPD Chronic Obstructive Pulmonary Disease.

CPR Cardiopulmonary Resuscitation.

DOF Degrees of Freedom.

GNG Growing Neural Gas.

GWR Growing when required Neural Gas.

HAR Human Activity Recognition.

HMDB14 Human Motion recognition Database with 14 actions.

HMDB51 Human Motion recognition Database with 51 actions as described in [97].

KNN K-Nearest Neighbour.

ML Machine Learning.

MYSET My own acquired set of 14 activities acquired in the modes described under section 5.2.1.

NG Neural Gas.

NN Neural Network.

PTU PanTilt Unit.

RGB Red Blue and Green, i.e. 3 channel colour image.

RGB-D RGB plus Depth.

ROS Robotic Operating System.

SAR Socially Assistive Robot.

SDK Software Development Kit.

SIFT Scale Invariant Image Transform.

SOM Self-Organizing Map.

SoW Skype on Wheels.

SVD Singular Value Decomposition.

SVM Support Vector Machines.

TM Telemedicine.

TSN Temporal Segment Network.

VPN Virtual Private Network.

Introduction

I was interested in investigating social assistive robots to track the habits of people and how could this information be used to foster wellbeing. The impact of healthy life-style choices, such as not smoking, a healthy diet, moderate alcohol consumption and physical exercise are associated with an overall lower mortality rate [51]. However, accurate long-term information is difficult to gather [156] and health scientists are still not certain on what are best strategies to foster healthy behaviour.

It is reasonable to assume that, if one wishes to intervene on those variables, first they need to be measured properly. Socially Assistive Robots (SARs), were they to be present in people's homes, would be the ideal platform to solve this problem.

As populations across the globe get older, and gradually lose their independence, there is a growing concern about who will take care of this population. SARs has been proposed as a possible solution for this issue. As the possibility of having robots in older person's homes increases, more interest is put on studying how to make use of this new interesting data, one of those is activity related information. In this context, I explored what I assumed is a desirable feature of a SAR: distress detection. Acute distress can have very negative outcomes if the person can't, themselves, ask for help. While acute distress can manifest itself in a number of different ways, the prototype for such a situation would be a fall event.

From my personal experience, a recurrent concern reported on doctor's appointments by older persons living alone is that "they could become sick and no one will be there to help them". This fear alone can be enough to motivate an otherwise healthy and independent adult to move into a retirement facility; a detrimental occurrence, as it

is known that changing the surrounding environment of an older person can cause disorientation and compromise cognitive functions. Institutionalisation itself has negative effects in terms of mortality rate and perceived quality of life of elderly persons [148]. While research has tried to address this issue [23], most current extant technology for elderly care consists on gadgets that each offers only limited functionality and often require considerable changes to a person's house and lifestyle. A robot has the potential of condensing the capabilities of multiple different devices, however, to be useful in this context, it should (among other things) be able to "understand" what the user is doing.

With these considerations in mind, fall detection poses an interesting problem as it is a hallmark of distress and could add great value to a robotic care platform.

Falling events are, however, very uncommon; in the domain of Human Activity Recognition (HAR), it would be most advantageous if robots could detect many different actions from a human user and react accordingly. These are the subjects that will be examined and explored in this thesis.

Thesis Structure

This document starts with a literature review of the context of elderly care (see Chapter 1). It explores medical aspects (see Section 1.1) and elderly care technologies (see Section 1.2 and Section 1.3), as a background that justifies the study of fall detection and activity detection. This chapter also reviews assistive robotics (see Section 1.4) with an overall view of general assistive robots (see Section 1.4) with a focus on assistive robotics for the elderly (see Section 1.4.2) and the progression into more specific uses cases that could be deployed on a robot, with a review of fall detection (see Section 1.5) and generalising this idea to more general actions (see Section 1.6). In the end of this chapter a brief summary of the expected

In Chapter 2, we present the infrastructure used for the present study, namely the design implementation structure chosen (see Section 2.1), an overview of Active Vision and Face Detection implementations (see Section 2.3).

Subsequent chapters 3, 4 and 5, represent different study cases of the evolution of the approach to classifiers of human activity, starting with fall detection using skeletons a falling stick model (Chapter 3). This structure is used and developed further to classify activities in general, which is described in chapter 4. Chapter 5 describes the deep learning implementation and my attempts at classifying real data obtained with the robot.

In Chapter 6, a general discussion is presented with reflections on the challenges related to the work presented in chapters 3, 4 and 5. In that chapter I also present ideas for future work to improve and build upon the activity classifiers, and how this could hope to improve the classification.

Chapter 1

Current state of research

In this chapter I aim to answer the question of how can one use a socially assistive robot to help promote health of older persons, by monitoring habits related to health, namely activities and among others occurrence of falls. I aim to demonstrate is that health monitoring is an important aspect of this task, both in monitoring acute ailments as well as state of chronic conditions. As such it can provide a safety net that promotes independent living in old age (see Section 1.1). In this context, a robot can provide additional gains over other types of devices by providing a privileged monitoring point which can substantiate adequate prompt interventions to improve quality of life for older persons.

Sections are constructed including an overview, a discussion and closed with a a conclusion about how knowing specifically the state of the art of this particular field has influenced me and the specifications of the platform design.

The starting point is reviewing the general outline of technical advances made in older person care (see section 1.1) and consider what kinds of functionality can potentially be condensed into a single device, a robot, and such topics will be investigated further (see section 1.2).

The first major technology investigated was Telemedicine (TM) (see section 1.3). TM serves as an important paradigm, because it uses very little infrastructure, it has been widely studied and it attempts at not only detecting conditions, but also

treating them, providing a useful ground comparison for a more complex system I wish to build. Additionally I consider that one can potentially include TM in a robotic platform as a way of combining additional gains from TM into other functionalities SARs have.

Socially Assistive Robot are reviewed on its current state of the art (see Section 1.4), both in general (see Section 1.4.1) and more specifically for elderly care (see Section 1.4.2). Adding functionality to SARs would increase its usefulness for the end-user. Possible additions I considered in this thesis were fall detection (see section 1.5) with Red Blue and Green, i.e. 3 channel colour image (RGB) and RGB-D sensor data, later generalized to activity detection (see section 1.6).

A final section 1.7, will propose expected contribution to the knowledge of this study.

1.1 Independent Living in Old Age

The concern of older persons living alone is not unfounded: quick access to health care saves lives. It is a medical fact [74] that diseases that can present themselves as a loss of conscience, such as strokes and heart infarctions - those two, the leading causes of death worldwide - can have an excellent prognosis if treated within 3 hours. Particularly cardiac arrests present a survival rate of about one in each three subjects, if Cardiopulmonary Resuscitation (CPR) and defibrillation are initiated in less than 5 minutes, whereas the probability of survival without any help is virtually zero [171]. Also other diseases such as pneumonia or Chronic Obstructive Pulmonary Disease (COPD) exacerbations do tend to have a faster recovery [183] if treated promptly. One must take care as to not make bold assumptions, even more in the scope of major reviews [48] are yet to reveal clear benefits of TM. Some interesting recent results [45] [46] [143] [70], however, hint COPD as a likely candidate to benefit from remote monitoring, as it could mean the provision of this fast access. Accurate, robust health monitoring of acute events is certainly a desirable feature to enable independent living in old age.

Not only acute events could benefit from persistent health monitoring; also chronic conditions and their progression is certainly of interest. One must note the high prevalence of chronic illness in older adults. Multimorbidity, that is, having 2 or more chronic health conditions, is reported to prevalent in 40%-80% of older adults. One study [106] from Sweden reported 55% prevalence of multiple comorbidities of the older adults population enrolled in its Kungsholmen Project in Stockholm, and an additional 30% having one chronic illness. Another study prepared from data of the Statistics Canada Research Data Centre at McMaster University, reported that 79.3% of persons in the age group 50-64 had at least one chronic condition, 90% of persons in the age group 65-79 and 93.3% of persons over 80 years of age.

Moreover, not only illnesses could benefit from careful health monitoring, but also lifestyle. With ageing, considerable changes occur in eating habits. Irregular nutritional status has been recognised as having important effects in the outcomes of morbid conditions, such as cancer, heart disease and dementia [181]. Malnutrition in older patients is presumed to be underdiagnosed due to normal loss of muscle mass associated with ageing and combined obesity and protein malnutrition which is common in this population, and is in itself a harder condition to evaluate in medical practice. The prevalence of malnutrition in persons over 65, when described as either deficits in protein, calories, vitamins or micronutrients, can be as high as 35% [181]. In elderly living in nursing homes this value may be even higher, with a study from Finland reporting 86% of 267 nursing patients from 3 different cites presenting malnutrition or being at risk [79]. It is also reported that polypharmacy (taking multiple medications), declining cognitive skill (evaluated as a Mini Mental State Examination score of under 20) and depression (evaluated with the Geriatric Depression Scale) were independent factors that were correlated to malnutrition risk. A reasonable hypothesis would be to assume that regular evaluation of eating habits of older persons (evaluated in terms of quantity and frequency), may make it easier to detect dysfunctional eating habits even before malnutrition may present itself.

Perhaps more importantly is to note that evidence shows that many of the so-called

age-associated illnesses can be prevented or even reversed with adequate life-style changes [50].

With these considerations in mind, the motivation for this Ph.D. thesis was broadly defined as: "How can one use automatic systems to evaluate the well-being of elderly persons", and then, upon detecting that something is wrong, implement another system to take the appropriate actions to mitigate negative consequences. Several assumptions were made a priori, so as to limit the work and try to make sure it could be realised in a reasonable amount of time by a single person, namely:

- Early detection of acute events: the prototype of these events is a fall. My focus on acute events has a simple rationale to enable a study to be carried out on a short amount of time as this would be a part of a Ph.D. thesis, which is limited in its duration. More scientifically, chronic issues are also more likely to be caught by physicians or family members, so adding a system to focus on those would probably not be as effective in promoting well-being.

- Should enable long term detection and logging of multiple events of interest. This refers to the extension of fall detection to multiple activities and should be the basis of a detector of appropriate and inappropriate lifestyle so as to thwart development of chronic conditions. As suggested by my literature review, I wish to track eating and drinking habits, exercise patterns, mobility, taking medications and smoking habits, among others.

- The proposed system does not rely on wearable devices. Patients tend to not use devices or medication when they consider themselves to be fine. This is a very serious issue and one of the reasons why high blood pressure or diabetes are so difficult to control. This would also be an issue for fall detection, if it relied on a wearable sensor that was not being worn at the time of falling. This is a strong concern as most falls tend to occur at night [14, 187] and it would be very likely that a patient would not be wearing the sensor in this case.

- A system should not majorly modify the person's living environment. Modific-

ations to someone's house are quite effective, but difficult to implement, since they depend on the person's house layout, habits and current health status. It would be more convenient to come up with a system that does not rely on such changes to facilitate adoption.

A robotic system, with appropriate sensors, that could position those sensors appropriately to gather data of interest seem to solve those issues. This is the kind of system I wish to investigate in the context of evaluating and promoting well-being. As this task, as stated, was too broad, I limited the explorations to detection and limited detection to initially of falls (see Section 1.5 for review and Chapter 3 for experiment and implementation) with an attempt at extending this investigation to a classifier of different activities (see Section 1.6 for a review and Chapters 4 and 5 for implementations of a skeleton based activity classifier and a deep learning based activity classifier respectively).

1.2 Assistive Technology for Older Person Care

With the ageing of populations around the world, elderly care is a field of growing concern. Many different technological aids [23] are being developed specifically for this population and robotics has emerged as a possible solution, as the mobilisation of human caretakers for such a large amount of persons seems unfeasible. While robots - regarding human-robot-interaction - are yet to find a particular field in which they are undeniably useful, an interesting approach [132] to their use is finding newer areas in which they can nothing but excel, simply because there are neither persons nor other technology available to perform that task.

A literature review was done in the topic of assistive technology for elderly care, which organised and described ad hoc roughly as follows (slightly modified version of Katchouie's paper classification [78]):

- **Telemedicine** (TM)
  - **Human Supervised TM**

  - Automatic TM

- Smart Sensors: Internet of Things (IoT) [1,2]
  - Ambience Intelligence
  - Wearable Devices
- Electromechanical Assistive Devices
  - Orthopaedic (Adnexal) devices: wheelchairs, walking aids, exoskeletons, reaching arms
  - Embodied systems, viz., **Robots**
    * Service Robots
    * **Socially Assistive Robot**

The classification presented here had the initial aim to be broad, yielded a vast number of papers and poses quite a difficult task for a single person to review properly. A decision was made to narrow down on its scope to focus mostly tasks that could be migrated and combined into a single robotic platform. I marked in bold letters the items which I will investigate further, namely Human Supervised TM and SARs[2].

Ideally I would also review different sensors and their use in an elderly care scenario, so as to determine what kind of information would not be obtainable without those specific sensors, but upon reviewing the subject, it was recognised that this task would be too broad and it could be integrated in a future step. A smaller more focused overview of wearables and smart sensors will be presented on the context of fall detection in section 1.5.1 . An additional remark must be made as there is quite a great degree of overlap in some of those technologies, with many studies doing interventions with one or more of those modalities combined.

[1] Considerable overlap with TM applications, many commercial products.

[2] I will also review indirectly some functionalities which are currently proposed to be implemented by smart sensors when I review fall detection.

1.3 Human Supervised Telemedicine

Telemedicine (TM) is an obvious choice of application that could be implemented in a robot [89, 147, 152], but itself does not necessarily require such a complex infrastructure as the one that a robot implements. The rationale here is very simple: assuming that, for whatever reason, there is already a service robot inside someone's residence, then deploying an algorithm that increases functionality adds very little cost with enormous potential additional gain to the user. Depending on how good a set of algorithms the scientific community may develop, this collection alone could justify a robot's existence in people's residences and be a basis for this technological ecosystem.

An excellent review article was found in the subject of TM done by the Cochrane Collaboration [48]. This systematic review's outcomes still hold true today[49, 52] and will hence be examined in greater detail. it listed around 10000 studies, but included only 93 studies with around 22k participants. The focus was on sick individuals and not elderly per se - however one must note the high prevalence of chronic illness in this population (see Sec. 1.1), which, in my view makes this topic relevant.

The intervention types were divided by technical outline and scope of intervention (note that some of them overlap):

- Technical outline:
  - Remote telemonitoring (55 studies)
  - Real-time conferencing (38 studies)
- Scope:
  - Early detection of a chronic condition (41 studies)
  - Provide treatment/rehabilitation (12 studies) [3]
  - Education/self-management (23 studies) [4]

[3] E. g., delivering Cognitive Behavioural Therapy (CBT)
[4] E. g. education for diabetics

- Specialist consultation for diagnosis/ treatment (8 studies)
- Real time assessment of health status (8 studies)
- Screening (1 study)

This review presented **no benefit** for hard outcomes such as all cause mortality (n = 5239), risk ratio (95% confidence interval): 0.89 (0.76-1.03, p=0.12 for a 6 month follow up) and admission to hospital (n=4529) risk ratio with range 0.36-1.60.

In some soft outcomes, however, improvements were statistically significant, such as improving quality of life of patients with heart failure (n = 482, follow-up range 3-6 months, mean difference = -4,39, QoL measured by the MLHFQ questionnaire), improving diabetes control, cholesterol levels and blood pressure.

Some considerations must be made about the Cochrane review on TM, which curtails the extent on which conclusions can be drawn based on this review. Those are related to study size, heterogeneity, failure reporting and Machine Learning (ML) use.

Size and heterogeneity influence conclusions to be drawn from a meta-analysis [69], as a large number of very small studies with varied implementations and outcomes measurements make results difficult to compare and may have even hide desirable outcomes. The most likely possibility is that the differences between in person interventions and TM are most likely small, and that would require a larger studies with a very precise methodology to demonstrate. This is of special importance, since conditions in scientific studies are prepared to be as close to ideal as possible and one must expect a real world implementation of a system to fall short from the study design. A conservative interpretation of these results is that one may not expect to demonstrate any large effect benefit with TM unless considerable methodology changes are made over what has already been experimented. Another issue brought on by this great number of small studies is that even positive results end up being disputed, as this design indirectly enables multiple comparisons [138].

Another issue is that most studies did not report system failures and the few that do, report numbers as high as 30% system communication failures. Finally an additional

issue is that no ML was used in these studies (one study used automatic reasoning for diagnosis and was excluded because of that).

As specifications for an effective TM system, it seems reasonable that a common framework would produce more homogeneous results and thus enable to demonstrate smaller size effects. Such a system also should be simple, to show cost-benefits over traditional care and it should be robust, to be dependable and trusted by professionals and patients.

We Need Better TM Studies To avoid some of the issues that were brought by the Cochrane review[5] of TM, a sound TM paradigm should aim to have an infrastructure that is (1) common to all, so as to facilitate analysis and reduce heterogeneity of interventions; (2) Simple in its structure (to be also cost-effective), as it is known cost-effectiveness is a major metric used by policy makers and dictates adoption of a technology in the majority of medical interventions by the health sector; and (3) robust, which is a requirement of most, if not all, medical devices, when one considers that a failure in some scenarios could cause disastrous consequences.

With such an infrastructure, more studies should be performed that enrol more patients (to demonstrate more subtle effects) with a high quality layout (double-blind randomised controlled trials) and over a longer time scale (longitudinal studies). Those studies should probably focus on some of the effects that were shown to be improved by TM, like symptom control in Congestive Heart Disease, which, with a higher population sample, could demonstrate cost-effectiveness by avoiding unnecessary doctor visits. The same goes for other indirect metrics, such as tighter control of blood sugar levels in diabetic patients, cholesterol levels and blood pressure, which should, on longer term studies demonstrate benefits of TM in preventing deaths from vascular disease. Important unknown variables here are (1) the novelty effect, which could have improved the results for some metrics more than they would in a longer study (2) adoption rate and dropout, as perhaps this type of intervention

[5]The Cochrane board itself acknowledges, based on their work, the difficulty in reviewing the huge number of academic productions on the field of TM [133], indicating it would wish there was a way to automate the review process so that state of the art information about TM could be available constantly.

is considered much too intrusive and could cause patients to stop using the system, even though it remain theoretically effective.

A more important realisation would be that using a robot solely for TM is probably not an economically justifiable idea, unless the robotic platform used is very inexpensive, unlike pure TM interventions shown on some cases to be more cost-effective than traditional approaches [41]. The reason behind this is simple: the cost of the robot's hardware makes it a less favourable alternative if all you need is a much more affordable smart sensor or communication platform. However this may change depending on robots already being deployed for whatever other reason, if, say, a points of failure analysis may show that a robot is more reliable than having multiple sensors around the environment, or the price of robots changes or if some perception modalities are shown to be much better perceived with a moving robot than with ambient intelligence. Additional aspects may influence one or the other approach, such as, low adherence to the usage of wearable devices, perceived higher invasion of privacy with a distributed home camera surveillance system when compared to a single robot, individual preferences, etc.

Roughly, one can estimate the economical gains of this implementation modality by examining an example study from 2015 done on the prediction of acute exacerbations of COPD [46] (not on the Cochrane review) that used ML to achieve a 5 +/- 1.9 days prediction of exacerbation by evaluating respiratory sounds. Assuming that on average 22% of exacerbations require hospitalization [1] in moderate or severe COPD, assuming no errors on the system this saves 1.1 hospitalisation days, which is around £300 pounds ($\sim$ 270£/day in COPD [65]) per year.

COPD is of course an extreme case, with only around 15.3% of older adults having one or more hospital stays (percentage from the CDC/US in 2016 [20]) and with an average time of hospitalisation for persons older than 65 being around 5.4 days [21], yielding a total of 41,680k days of hospital care. As in 2010, 12.972% of the US population was 65 and over [10] with a total population in 2010 of 308,745,538, one can estimate the number of elders to be around 40 million. This results in an average elderly person from the US spending around 1 day in hospital care per year. Similar

data from Germany [96] puts the number of average around 3.3 hospital days per year. This study also reports in this age bracket, doctor visits for men at 14.6 per year, 15.1 visits per year for women, costs that could be offset if a TM robot, at least in some cases, were able to correctly examine a patient.

Cost of a day of hospital care varies considerably between countries [114], estimated in 2015 to be from US$ 424 in Spain to US$ 5,220 in the United states. This would put the expected return from saving an average of one day of hospital care somewhere between £ 270/day to US$ 5,220/day depending on the country. With rising costs of hospital treatment as well as life expectancy, there is definite potential for TM to prove its worth in this scenario, however one must recognise that expected cost savings alone are not enough to justify a £ 5,000 Nao Robot or a US$ 400,000 PR2 [161]. Those robots may not be the most appropriate for the job of TM, but they represent the average cost of a simpler models (Nao) to more capable models (PR2). As more functionality would be required from the robot, the price would gravitate more towards the Willow's Garage 2010 PR2 robot - a 20 Degrees of Freedom (DOF) prototype research robot - which would be in turn offset by the price reductions from mass production. Simpler Skype on Wheels (SoW) robots [189], such as the US$ 3,999 Double 3 [72].

This review of TM thus brings me to the conclusion that **investigating TM is probably outside the scope of these studies and would require a much larger project with mature infrastructure**. However it also gives us useful information about what requirements my Socially Assistive Robot (SAR) implementation should provide, name my item 1 of expected contributions in Section 1.7.

1.4 Assistive Robotics

1.4.1 General Overview of Assistive Robotics

The current uses of assistive robots and robotic companions for the elderly was also reviewed (see Sec. 1.4.2), with particular mention to cornerstone papers of Rabbitt [132] that review and propose a new paradigm in which to view the role of

socially assistive robotics.

The focus of this particular aspect of the literature review is to examine the potential of robots to improve and act on mental health related issues. It is perhaps worth stressing the difference in focus when discussing service robots and assistive robots [78]. While both are intended to provide assistance, the former is a kind of robot that would not only be used in homes, but also office environments and would tend to perform more manual tasks. The focus of its study would be how to perform tasks together with a human, manipulate tools, receive directions, navigate cluttered environments, pick and place objects and related activities. Assistive robots would tend to focus more on human robot interaction from a psychological aspect and evaluate positive outcomes that could come from this interaction [132]. It embodies the idea of a robotic companion, more than just that of an augmented tool to carry on activities. While the end goal is to have both of those systems hopefully working on the same platform, currently most of these tasks are research topics and for this reason the separation here will be made. This section of the review will focus on the interaction aspects, rather than the service aspects.

It is perhaps worth mentioning that TM alone (without the use of embodied systems, that is, robots) was analysed for mental health outcomes by the TM Cochrane review [48]. In total it evaluated 7 studies showing no difference in outcomes, quality of life or costs, however there is no information on particulars of the design of the studies or whether they were non-inferiority type studies.

A key paper identified on this topic is a non-systematic review of SARs for mental health done by Rabbitt, Kazdin and Scassellati [132]. This paper suggests not only (1) adapting robots to existing interventions, but also (2) use robots for specific novel research that would not be possible to be implemented by anything else other than a robot. This paper evaluates usage of robots as a companion, as a therapeutic play partner and as a coach. While Rabbitt et al. fails to demonstrate any unquestionable or promising use of robots in this context, it mentions that this may be to the small size of studies performed.

A special mention should be given here to coaching, where 2 different studies seemed to show promising results for robotic coaches. More particularly, a study coaching elderly to do physical exercise [44] (with 33 participants) found that a robotic exercise coach is more effective than a computer screen coach, where participants found the robot "more helpful and attractive" than its screen version, also rating the activity more "enjoyable and more useful", however not detecting any difference between performance among groups. Additionally another study evaluating a weight loss coach [84] (with 45 participants) found that training with robots was engaging for longer (50.6 days usage with robot on average, 36.2 days with a computer, and 26.7 days with a paper log). It however failed to show differences in weight loss, possibly due its short duration. Caution should be exercised in evaluating this result as the studies were small, the samples were compose mostly of women participants (27 females and 6 males on Fasola and Mataric's study and 36 females and 9 males for the Kidd and Breazeal's study), and very likely dependent on how good the screen interface was done as well as other effects that could be at play, such as novelty bias and social desirability bias. On the other hand, it may as well be that the perceived embodiement of the robot helps users relate to the task and works by creating a deeper sense of responsibility and engagement towards the exercise or weight loss program.

Another important paper in this context was the 2011 Sharkey and Sharkey's review paper on the ethics of using robots as companions for elderly and younger children [152]. In those contexts the authors seem to believe that older adults were better emotionally equipped to deal with robots and draw benefits from this interaction as drawing from benefits in healthcare and welfare, while younger children would possibly suffer mostly negative consequences due to an emotional attachment that would be based on deceit and anthropomorphization of robots and ascribing to them characteristics that they do not have. Note that the authors mention both healthcare and welfare, which does imply that the robotic companions in question would not only be providing social assistance, but also service robots functionality, which drives those conclusions further from actual state of research, as they imply

robotic companions would be quite capable in general to justify their existence in people's homes.

1.4.2 Assistive Robotics for Older Person Care

A very good overview of assistive robotics in the context of elderly care is provided by Kachouie's paper from 2014 [78], a mixed methods Cochrane library style review. The author tries to answer the question of what can currently SARs do. Kachouie's review is thorough, but limited by the methodologic limitations of the original research, which calls out to more strict experimental conditions in studies in the field of robotics for elderly care.

Katchouie finds around 1000 studies, including 34 of those in its analysis. The studies referred are small (number of participants from 1 to 67) and divided into:

- Physically assistive non-embodied: wheelchair, exoskeleton, artificial limb
- Embodied assistance:
  - Service robots: independent living, bathing, mobility, navigation
  - Socially Assistive Robot: companion

What is related in the reviewed literature seems to demonstrate that the effects of interacting with these SARs resemble (but were not limited to) those of antidepressants [78, 145]: improved mood, reduced stress, increased activity, calmer and richer expressions.

Although the authors intentions and methodology are sound, it is limited by the studies which it included, namely their small size, but mostly the fact that most of the studies had a predominant population of women and that the participants had multi-factorial dementia. What this review indirectly points out is that if SARs are in fact useful in this context, adequate studies to confirm this hypothesis are still lacking. Some important biases could be noted when the table of included studies was analysed, to name a few (as highlighted by Katchouie et al. [78]):

- Cultural background affects results and attitude towards robots, however most

robot studies included are from Japan;

- Uneven amount of men in the samples - and gender was recognised as affecting robot acceptance;
- Many studies included participants with dementia (without specifying cause) - dangerous to generalise that these results would hold for elderly without cognitive impairment;
- Most robots used animal looking robots and studies indicate that appearance influences its acceptance;
- Studies were done mostly in health care facilities, not on subjects' houses;
- No control for biases (such as the possibility of Hawthorne effect [111]), not randomised, blinded, experimental, outcomes not standardised or unclear;
- Most studies were short duration with small sample size (most less than a year, except of Wada et al. [174])
- Weak methodologies [78] makes most studies outcomes possibly very hard to reproduce.

My Conclusion: Proposed Improvements for SARs Studies The simplest solution to getting more results in the field of SARs would be bigger (larger number of participants to reduce random effects and get better estimation of effect sizes, and reduce Type I and Type II errors), multi-centric studies (to account for different countries' perception of robotics) with longer duration (eliminates novelty effect) and a high quality methodology: clear interventions, randomised and controlled and blinded if possible, with clear measurable outcomes (assessed by validated tests or clinically meaningful measures).

Those studies do raise interesting research questions, such as:

- The educational level is inversely associated with overall robot rating: is this because more educated persons can perceive more of the robots' shortcomings or are other dimensions affecting this?

- Also, people can become attached to very simple robots (e.g Roomba): is this a global capacity a specific type of person has?
- Is it even ethical to use a machine to provide support for people by exploring this trait [153]?
- What are the effects on elderly without dementia and what are their effects on men?

While those studies did raise interesting research questions, I did not find any particular one that could be tested by the infrastructure the University provided (most of these questions could be solved with longer and larger studies with many participants and robots). Those were similar issues that were presented by the TM paradigm. What instead seemed to be within the realm of possibilities - and this would be this thesis' contribution - is to add more capabilities to SARs, to increase their potential use (item 1 of my expected contribution in Section 1.7). An additional item is added to that list, inspired by my desire to have reproducible strong results, namely item 4. I proceeded with my initial idea of using the robot as a mobile sensing unit and to investigate how well could distress or absence of well-being be detected by a robotic platform.

1.5 Fall Detection

A vast whilst incipient literature was found in fall detection, particularly oriented in detecting falls of older individuals [24, 29, 47, 58, 56, 108, 119, 186].

This realisation, together with the fact that unimodal systems could have great accuracy [120] motivated further studies to initially involve only RGB-D data (imagining a static optimally positioned observer) and consider multi-sensor or sensor fusion implementations of fall detection more of engineering challenges than scientific barriers. It is expected that multiple modalities might be necessary on a more advanced stage after closing an experimental loop and evaluating real-world scenario causes for failures in detection, however at this stage, for all the aforementioned reasons, I chose to focus on a single sensor implementation.

This is an a priori requirement, item 2 on my list of expected contributions in Section 1.7.

1.5.1 Using Smart Sensors

Detecting falls can be done with very high accuracy with the use of sensor fusion and wearable devices [58, 108, 186], being already implemented in many consumer products [168, 22, 128, 150]. These implementations have, however the disadvantage of requiring installation or to be worn by the subject all times, which greatly reduces compliance and overall effectiveness.

The specific task of fall detection has recently attracted a lot of research, with a lot of focus on smart home environments. Most fall detection systems [186] involve wearing special sensors device with accelerometers or detectors built on the floor or a combination of video and wearable devices with a sensor fusion approach in order to increase the accuracy of detection even further. These approaches although simpler (and therefore robust) have however the limitation of needing either a sensor to be worn at all times, or that the person's house to be adapted for this, which in practice will vastly limit its adherence. I see coding fall detection into some sort of a multi-functional robotic companion - that could have as one of its many functionalities: fall detection - as a reasonable solution to this problem. A robot can follow the user in a different environment, position itself in order to prevent image occlusion and this avoids the need to renovate someone's house or remember always to wear a sensor. Moreover the robot has only to add, as it could serve also as an additional sensor for a sensor fusion approach in case of a smart home environment.

1.5.2 Remote Assessment of Falls

My intention is to detect falls remotely, that is, without a specific sensor that needs to be used by the person, for the reasons mentioned in Section 1.5.1. This is a narrower scope than the detection of falls with any type of sensor, but which already presents a number of interesting publications. A review of Cippitelli et al. [30], did a great job in reviewing fall detection with both radar and RGB-D sensors. Although

radar implementations would fulfil the requirements of remote assessment of falls, an additional sensor would also incur in additional cost.

I chose to focus my approach on RGB-D sensor data classifiers, as those have been made affordable and common with the release of Microsoft Kinect, and reserve the possibility of adding additional sensors to a platform in the future, if this information would be proven necessary.

Cippitelli et al. relates 26 different papers that focus on classification of fall X non-fall events, under 8 different datasets, namely , Falling Event Detection [190], ATC4^2 [28], TST Fall Detection v2 [56], Falling Detection [193], UR Fall Detection [98], SDUFall [6], EDF (itself introduced in the review article "A Survey on Vision-based Fall Detection" by Zhang et al.) [195] and OCCU [194]. An additional dataset that includes falls was mentioned by Zhang et al. [191], the UWA3D Multiview [134].

Some of those datasets are more challenging than others. I will focus the discussion on the dataset I used for the first study, the TST Fall Detection v2. Although this dataset was classified with very high accuracy by the authors (100% accuracy) [56], it had done so using IMU data. Rougier et al.'s paper "Fall Detection from Depth Map Video Sequences" [139], using centroid's velocity threshold for classification of falls, combined with Parisi and Wermter's "Hierarchical SOM-Based Detection of Novel Behavior for 3D Human Tracking" [120], indicated that this was likely a consistent approach, which coincided with the physical intuition of a falling event and motivated my first experiment.

1.6 Activity Detection

The accurate detection of human poses and actions has many uses from robotics, to gaming, security, human-computer interaction, human-robot interaction, telepresence and healthcare just to name a few, with this list getting larger each time accuracy improves. In the field of robotics, activity detection [93] is a fundamental concept if the robots are used in a setting where they are expected to cooperate

with humans. The initial approaches to the detection of human actions involved the processing of RGB images, and as such activity detection was shown to be a hard problem due to (1) the difficulty in segmenting the human body from the background, (2) estimating pose and (3) accurately processing pose information. Recently however this task was made considerably easier with the introduction of skeleton tracking based on depth-sensing cameras such as implemented by the Microsoft Kinect, which implement task (1) and (2). As this technology steadily improves, with the introduction of novel algorithms [157], it also allowed for more complex perception tasks that depend accurate pose, such as activity recognition, to be investigated.

While falls are rare, activities happen all the time. In this sense, when compared to fall detection HAR is perhaps much more widely useful goal (see item 3 in Section 1.7). It is a very important intermediate goal if one is aiming to develop a capable and useful automatic assistant and/or service robot, especially in healthcare applications. In the special case of elderly care, it is fundamental that a helper system would work with as little as possible input from the user, in a virtually autonomous way, as one of the goals for such an assistant is that it would be able to provide care when the user is not at their full capacity. In this sense, recognition of human activity plays a fundamental role and it is desirable for this activity recognition to be based on as little as possible actively gathered information, and in this context, RGB-D, by adding another dimension to video data provides a much simpler to analyse paradigm when compared with the challenged of normal 2D colour video. One of the many interesting features of RGB-D data is the ease with which one may extract useful skeleton poses, which enable works such as the present one and those of many others [24, 191, 63, 121, 151, 42, 31] to implement activity recognition based on these time skeleton sequences (see subsection 1.6.1).

More recently, due to great advances done in classifying images using deep network architectures, more complex activity classification algorithms using RGB data were developed and seem a promising approach to solve this class of problems (see subsection 1.6.2).

1.6.1 Skeleton Based Activity Detection

Specifically in this case, I chose to extend the work on fall detection to multiple activities detection as this is a more complex work, but also enables to test the fall detection, or the fall detection principles as a specific action, that is, falling, with other types of activities as proxies for that, as to get people to do realistic falls is hard and it would be difficult to conceive of a practical way to test such a classifier in a more realistic scenario, given the rarity of falls.

A vast literature already exists in activity recognition through RGB-D data as it has in normal video (RGB) data [24]. A high quality, thorough and lengthy review was done by Zhang et. al. [191], and relates the most widely used datasets for activity detection as well as the benchmark holder algorithms for such datasets. As different activities have different levels of interaction between subjects, objects and the environment as well as different datasets offer varying complexity with different number of different classes, wider (kicking, running) or more finely grained tasks (hand movements), I will focus on methods that use skeletons and try to classify whole body postures - mostly related to my initial goal task of classifying falls.

I will review the state of the art results of CAD-60 in chronological order. Initially this paper along with the dataset was published in 2012 and it utilises a two-layered maximum entropy Markov model with a on-the-fly graph structure selection [162] presenting a (excerpt from the paper)"precision/recall of 84.7%/83.2% in detecting the correct activity when the person was seen before in the training set and 67.9%/55.5% when the person was not seen before". A comprehensive list of precision and recall is available at Cornell's website [136] for conference.

To name a few, Gupta et al. in 2013 [63] classified this dataset without using the skeleton information (using only the depth maps), using depth information for better segmentation and code descriptors to feed an ensemble discriminator achieving 78.1% precision and 75.4% recall. Shan and Akella using skeleton information, in 2014 [151] implemented a classifier that estimates key poses based on estimation of kinetic energy and a support vector machine to achieve a global precision of 93.8%

and 94.5% recall, Faria et al. in 2014 [42] used a dynamic bayesian network model to assign weights to multiple classifiers and implement an ensemble learning technique to achieve 91.1% precision and 91.9% recall overall. Particularly the architecture that I was trying to replicate from Parisi et al. 2015 [121] uses a chained growing when required neural gas classifier to achieve a global 91.9% precision and 90.2% recall on this dataset. Cippitelli in 2016 [31] uses K-Means and a multiclass support vector machine with a radial basis function kernel to achieve a global 93.9% precision and 93.5% recall on the CAD-60 dataset. More recently in 2017, Manzi [105], using a combined approach (X-Means and Support Vector Machines (SVM)) very similar to Cippitelli, achieved perfect recognition on this dataset.

1.6.2 Image Based Activity Detection

The works on image based activity detection, that is, the recognition of a human activity based on a sequence of RGB images can be approached as an extension of classification of still images so as to accommodate for time variations. As such, the recognition of human actions in videos is currently still a very challenging research task as it provides and requires additional cues in the motion sequence when compared to image classification [158].

Earlier attempts at human activity detection employed different complex hand crafted features to better encode spatiotemporal attributes [101, 184, 38, 76] of a motion sequence. Many different works explore the advantage of those custom features, usually based on extrapolations of classic image features to deal with temporally changing data, such as 3D-SIFT [149] (an extension of the Scale Invariant Image Transform (SIFT) algorithm [104]), histogram of 3D gradient orientations (HOG3D) [85] (an extension of HOG for activity detection [36]), histogram of optical flow (HOF), Fisher Vectors [142] among many others [179].

As expected from the development of CNN (see subsection 5.2.2 for more details) and their overall performance in image recognition, a similar process took place in video recognition, with extensions of 2D CNNs to deal with image sequences (3D CNNs) [167].

The "Two-Stream Convolutional Networks for Action Recognition in Videos" paper from Simonyan and Zisserman Another common approach employed to improve recognition of action sequences was to use not only the raw RGB images but also perform time dependent operations to that datastream and use this data combined with the still images to better capture the time changing aspect of image sequences. A prominent work that combined those two modalities and used CNNs for classification was the "Two-Stream Convolutional Networks for Action Recognition in Videos" from Simonyan and Zisserman [158] to classify the UFC101 and the HMDB-51 datasets. Perhaps most importantly in this work was establishing a way of implementing a CNN based network that could take advantage of temporal information as it is often the case that image based classifiers extended to deal with temporal changes use mostly the information just from still images, citing the work of Karpathy et al. [80], in which a network working on single frames performs similarly to a network using stacks of frames and was 20% less accurate than hand-crafted state-of-the-art trajectory based representation classifiers [124] (65.4% accuracy vs 87.9% on UCF-101).

The key idea for the temporal stream is the use of a multi-frame dense warping optical flow to construct a frame sequence such as described by Brox et al. [18]. In order to implement this structure which is used side by side with the RGB frame classifier, different options of stacking are evaluated (single frame optical flow, that is, stacking is not used; optical flow stacking with different window lengths, L=5 and L=10, trajectory stacking or optical flow stacking, unidirectional vs. bi-directional flows and mean subtraction vs. no mean subtraction) with its best performance achieved on the UCF-101 dataset with optical flow stacking, stack size L =10, with mean subtraction and a bi-directional flow of 81.2% when evaluated on split 1 of UCF-101. The results when combined via a fusion layer using SVM match state-of-the art recognition performance on the UCF-101 dataset with 88.0% average accuracy over 3 splits, just slightly outperforming Peng et al. [124] on this dataset, but not on HMDB51 (accuracy of 59.4% vs 66.79% state of the art at the time,

"Action Recognition with Stacked Fisher Vectors" from Peng et al. [125])[6].

The network used for this work was the CNN-M-2048 architecture of [25] and it is conceivable that a network with superior image recognition capabilities would also improve action detection. This possibility was tested and successfully developed by the TSN and served as the starting point for explorations.

Temporal Segment Network (TSN): the first state-of-art Convolutional Neural Network for action recognition on HMDB51 dataset The "Temporal Segment Networks: Towards Good Practices for Deep Action Recognition" by Wang et al. paper from 2016 [178] is a recent important benchmark algorithm when discussing classification of human actions from datasets such as the UCF101 and the HMDB51.

Several small improvements were added to the TSN classifier so as to enable it to beat state of the art hand-crafted feature based classifiers. Important is to note its title as it refers to a temporal subsampling of videos, which is considered by the authors as its main contributing feature, as this is said to prevent overfitting, as currently available video actions datasets that have proper annotations are still fairly small in size.

Additional improvements are the use of other modalities of dense flow computation methods (aside from Brox [18]) such as Farneback's dense flow [43] and $TV-L^1$ [188]. More implementation details and considerations are presented in section 5.2.3.

Newest developments and current state of the art classifiers for HMDB51 From the publication of the TSN paper in 2016, 32 other papers have achieved better accuracy on HMDB51 see Fig. 1.1.

According to the HMDB51 website, as of September 2019, the current highest accuracy obtained on the HMDB51 dataset is 82.48% from "Hallucinating IDT Descriptors and I3D Optical Flow Features for Action Recognition with CNNs" by Wang et al. [176]. This algorithm employs Improved Dense Trajectory descriptors

[6]For a good overview on Fisher Vectors see [142].

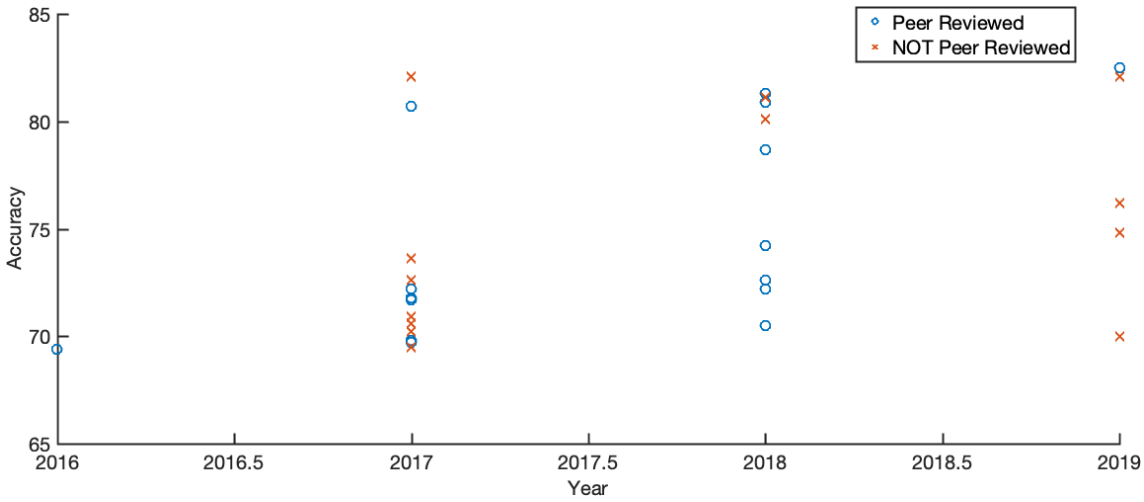


Figure 1.1: All the 32 new publications from 2016 until today indexed by HMDB51 website as having higher accuracy than 69.5% TSN. Marked with a blue circle are the peer-reviewed submissions and with a red cross are the not peer-reviewed ones. Results are grouped by year and do not represent actual dates in which those were published.

(IDT) [175], which is a combination of either Bag of Words or Fisher Vectors with CNNs, more specifically employing the I3D model "Two-Stream Inflated 3D ConvNet (I3D)" [19].

Perhaps also worth mentioning is the second best result, the 2017 article "End-to-end Video-level Representation Learning for Action Recognition" from Zhu et al. [196] built with the Deep networks with Temporal Pyramid Pooling (DTPP) architecture which makes use of a dual flow and RGB acquiring sparse samples and the aggregated both in space and time to achieve a video-level representation on multiple timescales. To note is that this structure is a descendant of many of the ideas behind TSN, however using a more fine-tuned pre-trained models from `http://yjxiong.me/others/kinetics_action/#transfer`; it uses sparse frames to avoid overfitting, 2 streams design (flow +RGB) and has a model based on the BN-inception network. Also worth noting is that the code is available for direct testing in `https://github.com/zhujiagang/DTPP`.

My Conclusion: Design Specifications for a Robotic Classifier These observations substantiated my approach on how to design a reasonably generic infrastructure for classifiers, more specifically, action classifiers, that would be based on the idea of improving general performance by reuse and combining different existing

algorithms. Moreover, it is expected that more heterogeneous algorithms (by having low error correlations) would give the most increase in performance possible.

This is the basis of choosing a skeleton based, topological GWR classifier (see sections 3.2.4 to 3.2.7 and chapter 3) as a starting point for this investigations, as I suppose should have a low error correlation with classifier responses from promising structures involving image-based CNN (see chapter 5).

Additionally, the need for multiple algorithm deployment (see item 1) is also the basis for implementing the ROS-docker encapsulation infrastructure (see section 2.2).

1.7 Expected Advancements to the Science and Technologies

Considering the landscape of research presented with the current state of TM, HAR, fall detection and activity detection and progress in the latest years, we aim for our work to bridge this gap and investigate the implementation of those systems in a mobile robotic platform.

As an expected contribution to current scientific knowledge, I expect from this work:

1. Devise an efficient way of deploying a ML algorithm for fall and action detection on a robotic platform;
2. Implement one or more fall detection algorithm that is testable on a robotic platform;
3. Implement one or more activity detection algorithm that is testable on a robotic platform;
4. Test the implemented algorithms with a strict cross-validation method, either cross-dataset validation or real-time classification of data obtained from a robotic platform.

As we want to work with deployment and implementation of many algorithms to

hopefully implement ensemble learning and fusion strategies, we see that the infrastructure needs to be built to allow multiple instances and training modalities at once. With the idea of ensembles in mind, it is known that algorithms with greater variability produce better ensembles in the end. The GWR was chosen because of this feature: it may produce a classifier that excels in situations where others do not and should improve the overall accuracy of a combined system. CNN was used as it is the state-of-the-art today in classifiers and as it is a technology that simply must be used when classifying images, we see (as does the literature) as it being paramount to classification of moving pictures.

Initial plans and necessary changes During this work, we planned for ensembles from the beginning and having multiple algorithms working concurrently was one of the necessary building blocks of this plan. These implementations, however, turned out to be much more involved than we estimated and this part of our plan had to be amended. Another issue was also execution times and processing loads of running many algorithms at once. Clustering and parallelism were needed to achieve reasonable performance and structures also needed to be simplified to reduce network load and latency. The necessity of a solid infrastructure (see Chapter 2) to isolate classifiers and provide proper networking made itself necessary and a considerable amount of time was spend in devising a proper ROS nvidia-docker interface and having algorithms work in a distributed way. As I see this work now, it is more suitable for a team than for a single Ph.D. student, as there are many independent parts to optimize at once.

All things considered, I am glad with the overall result of this work. The infrastructure built, although complex, seems evident now if a cluster-like robotic platform is to be attempted. I believe the ideas presented here should serve as a good stepping stone to bigger and more challenging robotic implementations of perception systems.

Chapter 2

Infrastructure

2.1 Implementation Overview

My initial implementation (see Chapter 3) of the classifier algorithm for fall detection did not focus on deployment on the robot. Its main goal was to serve as proof of concept for a later implementation. As such it was undertaken using Matlab in a Windows system, so as to be able to conduct tests using the Kinect for Windows[1] version of the sensor and its Matlab Software Development Kit (SDK). Most of the work, however, was done using either the TSTv2 dataset [56][2] or the CAD60 dataset [162][3] . The initial plan was to wrap the classifier and use Matlab's ROS interface (and perhaps a simple embedded computer with Matlab and the Kinect for Windows drivers) and ROS networking capabilities to deploy and test this system, however the classifier's performance (see Tab. 4.4) was not considered good enough to warrant this implementation and newer structure was adopted to be described below. An overview of the present work implemented to easily deploy docker-encapsulated deep-learning classifiers in a robotic platform can be seen in Fig. 2.1.

Data is acquired from an ASUS Xtion sensor (see Fig. 2.3a) [7] , whose image frames are published as a ROS topic and compressed to be sent to the remote ROS network (see Section 2.2). Frames from this image topic are resized (to a smaller and constant

[1]Uses Kinect for Windows type skeletons, that is, skeletons defined by 20 joints.
[2]Uses Kinect v2 type skeletons, that is, skeletons defined by 25 joints.
[3]Uses Kinect v1 type skeletons, that is, skeletons defined by 15 joints.

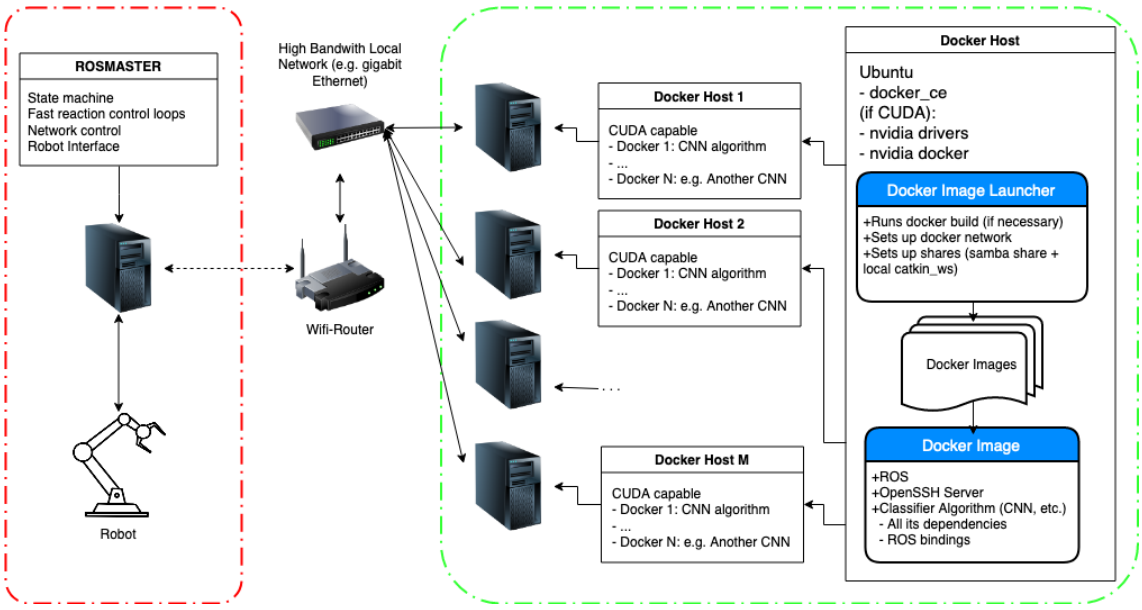


Figure 2.1: Diagram of a general ROS network architecture with fast loop (red) and slow computationally intensive remote loop (green).

height and width) and used to calculate dense optical flows by the deepflow node which is used by the classifier. A full resolution version is fed to face detection node, which will be used by the active vision node.

The active vision node uses the last known tracked face position and image flows to estimate the current position of the head in the picture frame. This (x,y) position is used for feedback to a controller node that positions the pan-tilt unit so as to have the face within a certain portion of the image being acquired by the sensor.

I also implemented a classifier, which is the deployment of a pre-trained model of the Temporal Segment Network (see section 5.2.3 for overview and chapter 5 for implementation) using features from a model trained on the HMDB51 dataset [97], stripped down from its last layer in a transfer learning strategy, feeding theses intermediate values to another network (implemented in pyTorch) which was trained on and enlarged dataset containing both HMDB51 data samples and my own acquired dataset (MYSET).

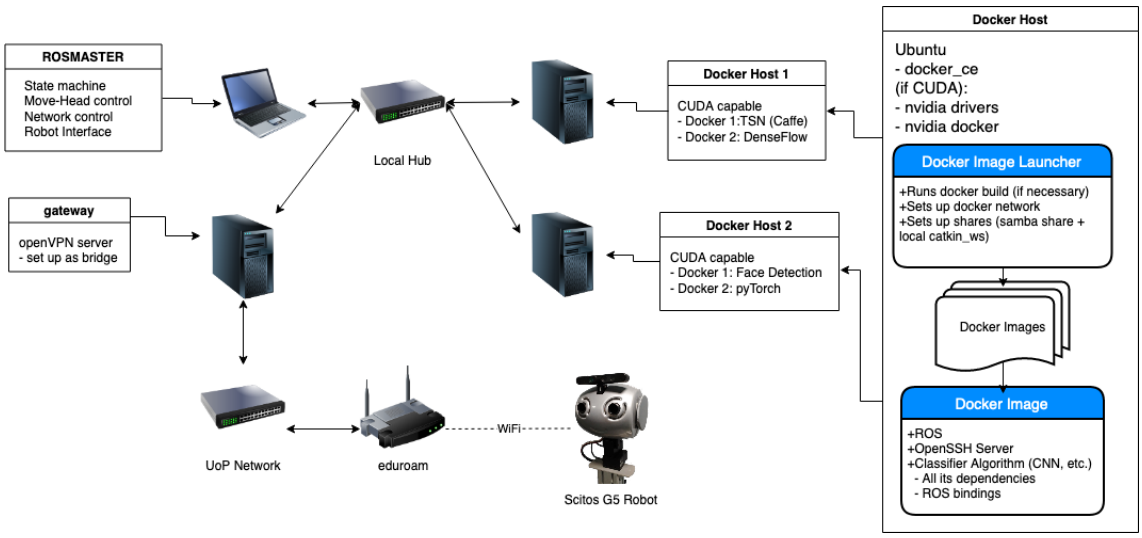


Figure 2.2: Diagram of the ROS network architecture with Scitos G5 robot as built with VPN.

2.2 Infrastructure

As the Robot Scitos G5 is not fitted with a CUDA capable graphics card - necessary for the algorithms I wanted to test - it was decided that a deep learning classifier would be implemented using a ROS network. The simplest way of doing this would be to install the necessary packages and repositories from the provided TSN GitHub page, however, as it was foreseen that maybe more than one implementation of Deep Learning classifiers would be tried, for portability, I chose to implement a more complex system that ran inside a docker image. The basic network structure outline can be seen in Fig. 2.2.

Using docker allowed me to avoid at least partially dependency hell and having to install multiple versions of software in the same machine as different algorithms were configured to use slightly different libraries. Docker allows for a more responsive system when compared to a virtual machine and prevents tainting of the main system by trying over and over to set up dependencies, failing and have to do a clean install. Each image can be generated from a Dockerfile script on-the-fly and that script can be later deployed on a different host, in case I need to restructure the network for any reason.

The biggest advantage of this system is empowering simple robots in combination with high-processing capabilities by using workstations that were designed to run

General-purpose computing on graphics processing units (GPGPU), e. g., CUDA applications, with minimal changes to the host system.

I believe that this is the most natural deployment of a ROS network in which a robot is required to use to multiple concurrent deep-learning algorithms and that, since it relies on ROS and docker, it can be easily extendable and highly modular. Those are features I want, as they aid in developing speed and help with fast prototyping, characteristics which are highly desirable for testing multiple different architectures.

The setup presented here could and should probably be implemented using a docker compose and docker swarm, but due to a foreseen complexity of starting off with these models, these were not used. I implemented a simpler version of these systems which was adequate to run the desired structure, however I advise strongly that the correct way of running this infrastructure (and make use of more advanced load-balancing, for instance), should make use of docker compose and docker swarm (see Future work, Section 6.9.1).

The idea implemented here was heavily inspired by Aquila [126], incorporating ideas from Elastic Thought [102] and the new paradigm of Edge Computing [155].

2.2.1 Docker Images

As I was interested in running code that required CUDA, each particular docker image was instantiated from a cuda-ubuntu16.04 with cudnn5 already installed and then adjusted to have ROS installed.

So as to keep the images to a manageable state, ROS was compiled from source and it was used mostly to implement the communication protocols, that is, to publish and subscribe to topics, read and create custom messages, have access to ImageTransport and openCV-bridge. So as to enable remote deployment from a ROSMASTER, an ssh server was also installed in all docker images.

The common structure of each image is to have also a docker network setup (one per host) in a start-up script, along with a ssh-fs mount that enabled the code within

the docker image to be edited from the host computer. This mount had the catkin workspace for that image, which would be setup each time the docker image is compiled, and had the additional packages necessary for that docker-image to do its tasks. This setup was not strictly necessary, but it ensured I had a single version of the most current code and it allowed for easier version control (using git for version control, in this case).

Additionally, as I did not use docker compose, it was necessary to keep track of all the host images' IP addresses and their names, if any of the ROS nodes were to directly communicate with a ROS node contained within that docker image.

2.2.2 Docker Hosts

The common setup necessary on the docker host was kept quite slim, as I intended for the code to be straightforward and the complexity to be dealt within the docker images and ROSMASTER. Strictly, the docker hosts only needed to have docker_ce installed and be capable of retrieving docker images. But as I aimed to use CUDA optimized versions of algorithms, NVIDIA drivers and nvidia_docker2 were also needed to be installed.

Each host also contained a folder in which the docker image packages would be contained, and was setup to have an openssh-server, so that docker image packages startup scripts could be called remotely.

2.2.3 Docker Image Package

So as to launch each docker image, a git package was organized containing scripts to set up the docker image and build it, as well as the catkin workspace (with its packages as submodules) that would be shared between host and docker image.

Each docker host was set-up as to do forwarding (which was setup automatically if needed by a startup script to be run on the docker host), so that the docker machines that it would spawn could communicate with the ROSMASTER. The docker startup script also set up manually docker network bridges that were set on host basis, that

is, each host had its own docker bridge and provided the docker images with an IP in its range.

Additionally the script would have each host have access to a shared samba mount, for common file sharing among all ROS network nodes.

2.2.4 Rosmaster

As I did not use docker compose, the ROSMASTER needed to keep track of all docker images' IP addresses so it would adequately set up communications back and forward. An adequate ip routing table also needed to be set, as each docker host was also forwarding packages to each of its containing docker images. This was done manually by a script that would add all the hosts in my network and create the adequate routing table. This package would also be responsible with setting up 'ssh-rsa' keys with docker images so that ROS's paramiko could ssh into docker images and launch nodes remotely (using launch files <machine>tags).

2.2.5 Virtual Private Network (VPN)

Another issue I needed to solve when deploying this structure is that, although the robot is mobile, my ROS network is not and the robot needs to connect to it somehow.

My initial idea was to have the robot physically near the network and have a network cable, so that neither security nor bandwidth issues would arise. This would also remove worries about time delays and package loss, especially when streaming multiple video channels.

An idea that would provide some bandwidth and network limitations was then to evolve the structure a little bit and connect the robot via WiFi to our ROS network, however, none of those approaches was possible, since I did not have a testing space near the ROS network computers or within WiFi range. The solution I adopted for this issue was to use eduroam with a Virtual Private Network (VPN) tunnel and to stream compressed video between the robot and the network. Inside local

the network, video topics would be streamed uncompressed, for speed and ease of use. An encrypted tunnel was used, because otherwise the robot's software interface would be exposed to the outside world. Not only could the video stream be watched by anyone else, which would raise privacy concerns, but also commands to move the robot could be issued, which could pose serious physical harm.

This approach has the advantage of enabling use of the robot with remote processing at any point within coverage of eduroam, however it creates a bottleneck in terms of bitrate, which requires the use of compressed Video or Theora encoded image transport topics. Unexpectedly to me, this is more of an issue than I assumed initially, as it places more restrictions on what kind of processing can be done remotely. Compressed video using a resolution of 640x480@30fps is forwarded through a VPN usually quite well, however transient artefacts are occasionally present in the video, possibly due to spikes in network usage within the university's infrastructure. Additionally, I also can't use caching strategies, due to small delay requirements from nodes such as the active vision implementation (see Section 2.3). Moreover, I noticed that quick movements of the subject or quick camera movements would generate lots of artefacts, which are detrimental for video capture and those would most likely degrade the classifier's performance.

In my tests, so as to mitigate this issue, I thus, decided to try to avoid quick camera movements and position the subject slightly further away from the robot than I would believe optimal, so as to minimise frame changes, which is perhaps a sub-optimal solution, but that still produces videos visually similar to the ones in the base dataset, that is, the HMDB51, which includes lower frame-rate videos, and some compression artefacts (see Future Work Section 6.9 for ideas on how to deal with this issue).

2.3 Active Vision

Active vision, as described by Aloimonos et al. [3], is defined by having an active observer which tries to control some geometric parameter of the sensory apparatus.

It stresses that human perception is not passive, but active, with humans observing scenes while adjusting eyes for the level of illumination, focusing on objects of interest and adjusting the angle of their eyes and head so as to observe a scene better. One of the main motivations of using active vision is to mitigate some of the issues that passive vision have and thus observing images that are hopefully easier for the classifier to process. Ballard describes a similar concept which he refers to as animate vision [9], in which the goal is to control the direction of gaze and thus get a better representation of the environment. The paper also bases its justification on biologically inspired works that link perception with action. One particularly interesting example comes from seminal research from Yarbus [2] that shows, among other things, how different instructions affect how a participant views a particular image. This finding is taken as a cornerstone for eye tracking research [165]. Ballard's paper also describes gaze control algorithms that although focus on moving stereo camera systems, do apply to some of the issues I was facing when using a static camera paradigm.

More specifically, in order to detect an action using a video-feed, one needs to track a subject within its environment, to make sure the action is being captured on camera. For this I sought to devise an active vision strategy that would allow me to keep the subject within the frame all of the time.

As this was not the main focus of the investigations, I tried to deploy a system that was only good enough for the task, and with that a simplified strategy was sought. The strategy chosen was not the detection of the whole person, but mainly focusing on the subject's head (using for this a face detection algorithm). Later a simple controller would move the head so as to keep the person in the frame.

The code necessary to run this is available in (face detection docker image launcher: `https://github.com/mysablehats/FT`, from ROSMASTER repo, use the move-head package `https://github.com/mysablehats/wholerobot_act` and denseflow docker image launcher, nested branch: `https://github.com/mysablehats/dt/tree/nested`).

2.3.1 Face Detection and Fast Feature Tracking using Optical Flow

Face detection is considered a mature image classification problem and I chose to make use of available of the available computational resources and went for the most accurate CNN classifier open source face detection algorithm I could find. In doing so I hoped to not have this subsystem in any way degrade the performance of the other challenging tasks I needed to perform.

For face detection I used the algorithm provided with full code from `https://github.com/ageitgey/face_recognition` [57] based on the dlib library with a .9938 mean accuracy on the Labelled faces in the Wild dataset [170] - note human mean accuracy was measured to be 0.9920. While dlib is not state of the art, many of the highly performing systems are commercial software, while dlib has freely accessible code. Dlib allows face detection, using a 640x480 resolution up to 5 meters away, which was suitable for my experiments. However, deploying such algorithm with this required accuracy, would run at maximum frame-rate of 2-6fps, which was too slow to allow proper PanTilt Unit (PTU) control.

In order to improve the speed of accurate face-detection, another strategy was used, based on the fact that I am already calculating optical flows for the classifier. Namely, with dense optical I have dense motion vectors for a whole image, so integrating these vectors over time allows me to estimate where the face might have moved during the time with which I don't have any response from the face detection algorithm. A better estimate of face's position is thus calculated using both response from the neural network and the estimated position from the sum of vectors from the whole area where the face was last seen in the picture, namely, if I define M to be the motion vector at instant t, f to be a function proportional to the flow vector of each pixel within the image and A as the area of interest constituted by ΔA_i pixel elements, for a generic mask I have:

$$M(t) = \sum_{i=1}^{n} f(t)(x_i, y_i)\Delta A_i$$

where:

$$(x_i, y_i) \in A_i.$$

and k is a constant specific to the flow algorithm being used.

For a simpler case of a not using a custom mask, but instead considering a rectangular approximation, with time dependent limits $[x_0, x_1][y_0, y_1]$, I have a more easily computable formula as:

$$M(t) = k \sum_{x=x_0}^{x_1} \sum_{y=y_0}^{y_1} f(t)(x, y)$$

With the updated position of the object's centre C from t_0 until a time t being given by:

$$C(t) = C(t_0) + k \sum_{\tau=t_0}^{t} M(\tau)$$

This idea mimics behaviours of the human eye saccades and could probably be extended to tracking of any particular object, with a reduction in Neural Network (NN) computational costs, if I assume that calculating dense optical flows is less complex than object identification.

2.3.2 PanTilt Unit (PTU) Control

The PTU used in the Scitos G5 robot's head is the PTU-D46 from direct perception (see figure 2.3). Drivers for this unit were available for ROS in `https://wiki.ros.org/flir_ptu_driver`, however since the robot was using an older version of ROS, due to its operating system being 32 bits, and older version had to be used which caused additional issues. Moreover faulty hardware (a serial port problem) presented as an intermittent problem was causing issues with control and limited the amount of tests I could perform.

The control strategy used was a simple proportional gain control, with values measure experimentally so as to avoid overshooting. Those were set to be $kx = 0.0011°/pixel$

(a) (b)

Figure 2.3: Scitos G5 head (a) with Xtion sensor and 2 firewire cameras (not used) and a detail of its pan-tilt unit (b), the PTU-D46-17.5.

and $ky = 0.0012°/pixel$ and set velocity to a constant $0.9°/s$. An attempt to implement indirect velocity control was attempted, but due to low baud rate (9600 bps), unnecessary data sent through the serial port by the old ROS driver implementation (specifications form the unit recommend Encoded Mode (binary format for higher bandwidth computer control) and the driver was implemented with Interactive Mode (ASCII command set) and the option for velocity control was not implemented), the fastest I could get information to and from the PTU was 6hz (surprisingly much slower than the sped up face recognition algorithm running at 30fps, possibly limited by camera frame-rate). A proper implementation of very fast head tracking would also entail coding the correct driver options (encoded mode and velocity control), or using another PTU (or even designing another proper PTU, as it is not such a complex unit, with say, dynamixel motors and 3D printed parts).

This speed limitation was not too much of an issue, however it did make it so that fast movements, especially near the robot, would cause the face tracking to be lost.

For the controller set-point, I chose not to position the head in the middle of the frame, but rather have it a bit up, so as to make sure that most of the body of the subject would be present within the captured frame. In the end, experimental testing showed that the optimal position so as to have the head not leave the frame

too often, was to position it in the middle of the first third (from top to bottom) of the picture frame.

So as to minimise the need for constant camera movement (and avoid image compression artefacts), a deadband was also implemented, comprising 30% of picture size in the x axis (horizontal axis) and 20% in the y axis (vertical axis).

2.4 TSN docker specific implementation details

The choice for classifier that guided this whole structure was for the python-Caffe TSN `https://github.com/yjxiong/temporal-segment-networks`. As in the paper that described it, I used both RGB and flow trained networks to attempt classification of actions. In my implementation, the classifier's GitHub package itself was divided into 2 docker image loader repositories, namely one that provided the dense optical flows (`https://github.com/mysablehats/dt/tree/nested`) and another that implemented the caffe wrapper (`https://github.com/mysablehats/CT/tree/nested`).

To use the pyTorch architecture as well, the images were grouped into another repository, available in `https://github.com/mysablehats/whole_f1/`. A pre-experimental setup of the robot with already all of the individual components working concurrently can be seen in Fig. 2.4.

2.4.1 Dataset Loader

My initial experiments were in implementing this algorithm were to make sure one could do real-time classification using this neural network, as the paper is focused mainly in the classification of the datasets UCF101 [160] and HMDB51 [97] and not its possible deployment.

In order to test whether I implemented a proper classifier, a dataset loader module for ROS was implemented. This was done by using a video publisher to act as if it were a camera connected to a robot. The advantage of doing this is that once the structure is finished, loading datasets or image from an external sensor would be

Figure 2.4: Pre-experimental setup of the robot with face tracking already implemented. For the data gathering, the console was occluded and a simple word representing an action would be shown on the screen, together with pre-recorded synthetic audio instructions. Initially the plan was to gather a dataset with multiple subjects, but in the end the set was simplified to a single subject, me.

seamless.

I implemented a loader node based on ROS services controls, so as to be able to play and stop playback, reload splits and choose only some portions of the dataset to be published, limiting the actions to a smaller subset. Reducing to a smaller set is useful since this classifier was planned to be used in a robot to be operated inside one's house, so many outdoor activities could be completely disregarded.

Further reduction of the scope of the classifier was done as to make data acquisition of testing test simpler, as some actions involved more than one person or were not performed indoors.

The end activity list that was used for most tests was composed of 14 actions, namely: 'brush hair', 'chew', 'clap', 'drink', 'eat', 'jump', 'pick', 'pour', 'sit', 'smile', 'stand', 'talk', 'walk' and 'wave'. This subset of the HMDB51 dataset will be referred to as HMDB14 from now on in this text. More details about this subset can be read in section 5.2.1.

An outline of the dataflow structure can be seen in Fig. 2.5. I chose this slightly more elaborated, ROS network based, using docker containers, so as to abstract learning from any image dataset (given a proper structure with classes and split definitions) in the same way I would learn from a robot camera. Additionally, I chose to allow for preprocessed data to be saveable in a samba share, so as to avoid having to run the whole state machine all the time. The structure chosen seems to me to be fairly general and makes deploying a classifier straightforward, a useful feature if many different classifier structures want to be tested. Additionally, it enforces real-time classification strategies, as using data such as action lengths and ending events becomes evident.

2.4.2 Denseflow

The denseflow is originally a wrapper of dense flow algorithms from a GitHub repository implementation from Limin Wang [177], with a slightly modified version implemented in the TSN as an additional modality to still RGB, so as to

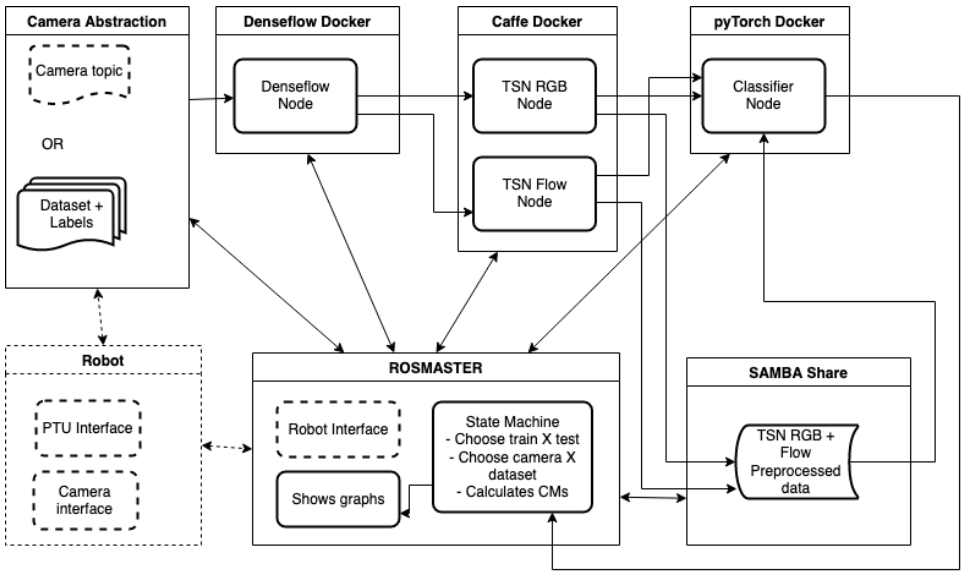


Figure 2.5: A block diagram structure of the dataflow of ROS nodes used. The camera is abstracted and allows for seamlessly loading of either a dataset (as a fake camera node) or the camera and other robot functions (marked with dashed lines).

increase the overall classifier performance. It basically consists of a C++ wrapper of openCV functions to run on video files which can be compiled to be executed with CUDA.

As my initial idea was to simply implement a real-time ROS version of this package, little was changed in its overall structure. It was just made into a ROS package, reading and writing ROS image topics using CVbridge and deployed in a docker container with a bare-bones ROS kinetic distribution.

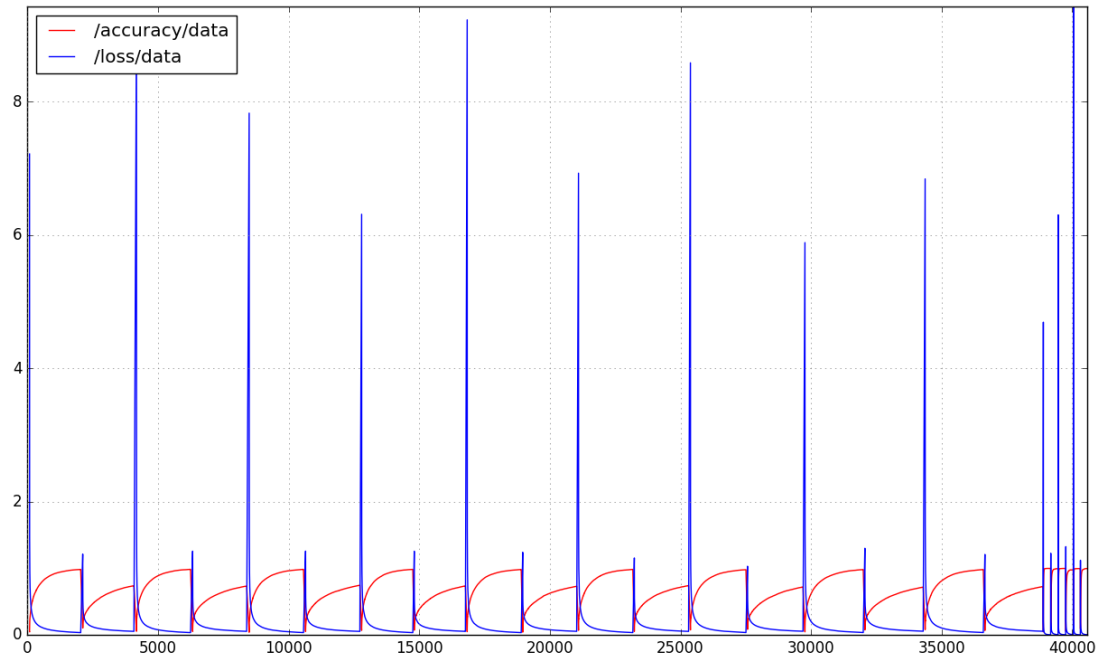
Alterations were intended not to change function, but to facilitate integration with ROS, and as such, instead of writing video files, the video frames are now video-topics (and thus streamed), the flow that was decomposed into X and Y components of grey images was composed into a Blue-Green image as to avoid the need to use message filters and having to deal with message synchronisation between topics and finally a cast that was used to convert flow floating point images to grayscale was substituted by the adequate openCV version of such function.

My latest denseflow docker image is available in full in: `https://github.com/mysablehats/dt/tree/nested` if a docker deployment is to be attempted. In case one may choose to install OpenCV with CUDA bindings and run the algorithm on the host's OS, one may use the repository: `https://github.com/mysablehats/`

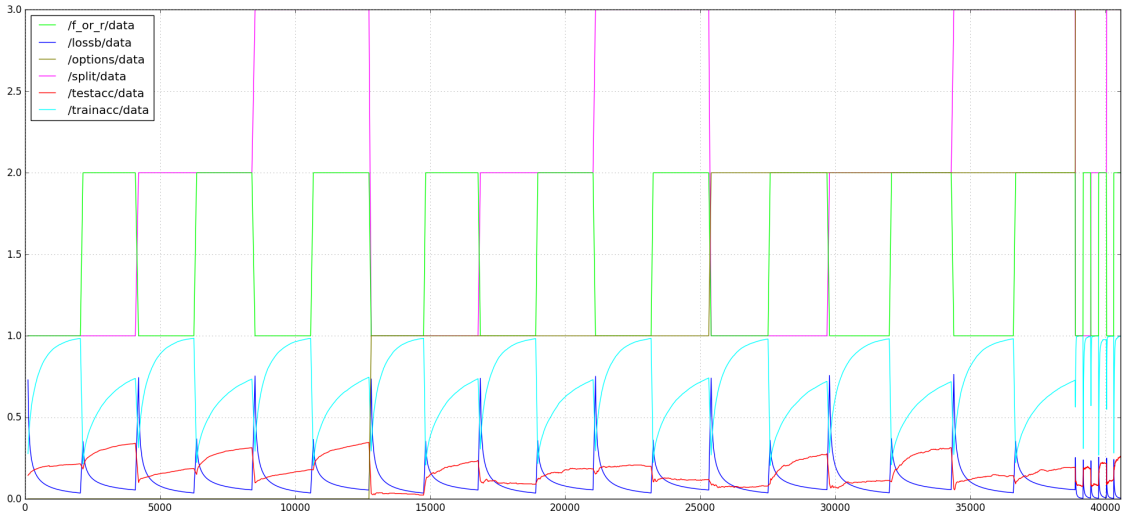
`dense_flow` [88].

2.4.3 State Machine

A simple state machine to follow through the steps of learning procedure was created as a python script. This was necessary because with the proposed implementation, systems run on multiple machines, forming effectively a cluster that needs to have its behaviour synchronized. This state machine ran on the same host responsible for active vision nodes, but their implementations were separate as we ideally would have the active vision controlled as close to the robot as possible. Old legacy software was an issue and some nodes could not be run in the Scitos robot, and hence the structure presented here. The signals from the docker image were also published which allowed for control within this state machine as can be seen in Fig. 2.6.



(a) Loss and accuracy of training set on the whole 4 testing conditions over all splits and 2 different modalities.



(b) Same graph with added control signals and testing accuracy that show clear demarcations of splits, modality and test options.

Figure 2.6: Graph of both loss and accuracy with and without control signals. I can see that the network gets close to 100% accuracy on training set, however the testing data shows it is beneath 40%, clearly showing overfitting.

Chapter 3

Fall Detection

3.1 Introduction

In this chapter I present an extended methods and results section from the work done on fall and - to a limited extent - activity detection with some extended explanation about the structure of my classifier and details of my falling stick model, as it was presented in ICONIP 2016 [86]. I also provide the full source code (this version of the code is available in in www.github.com/frederico-klein/ICONIP2016) of the Growing When Required Neural Gas implementation (in Matlab and Julia), as well as the full classification architecture and the inverted pendulum model (only in Matlab).

3.2 Materials and Methods for Fall Detection

The methodology is described first with a general overview of the whole architecture, followed by a description of the datasets used (the TST v2 and the Falling Stick Model) and the classifier's most key components. I will focus on the differences from my implementation and Parisi's - that is, my implementation of the GWR Neural gas, followed by sliding window implementation, the hierarchical aggregation, preconditioning and ending with labelling.

3.2.1 Justification for the Chosen Architecture: a Chained GWR Sliding Window Topological Classifier

GWR Neural Gas

Perhaps the simplest way to describe the Growing When Required Neural Gas is as an evolution of Self Organising Maps (SOMs) that aims to fix some of its shortcomings for specific tasks. SOMs form a description of data by adjusting a fixed grid (usually a 2 dimensional rectangular or hexagonal lattice) of points (or neurones) based on their relationships with their neighbours in higher dimensional space. One of the limitations of a SOM is that with a fixed grid adapting to data with more complex representations (say a complex topology with disjoint sets) renders less than optimal solutions. In order to tackle this limitation, Martinetz presented the Neural Gas [110], in which the neurones would not have a fixed topology, but could roam the space in order to describe data better. The Neural Gas itself had limitations and many others have made improvements [130] of the basic algorithm, the most relevant to me is the Growing Neural gas [53] which adds neurones to the set in order to give a better representation of the set, and the Growing When Required neural gas [109] that by setting an activation parameter threshold a priori limits the addition of neurones. For a more in depth understanding of these methods one should refer to the original papers which implemented them (in References).

Without knowing the literature (that says this method is effective for this particular task), one may reason that the use of a topological unsupervised learning method for topologies is adequate for this task, as, in a sense, an action sequence is a topological space, as a person's limbs must follow continuous trajectories, that is, there are no jumps, with limbs disappearing in one place and appearing on another and also, the particular velocities or positions are not exactly the most relevant descriptors of an action sequence, but their relations. The justification of using multiple chained gases (as opposed to one) is, first the biological plausibility reviewed extensively by Parisi but secondly probably due to necessity regarding the way too long execution time of a gas with a high number of dimensions. Finally one must add that, although neural

gases, due to their nature, adjust to data that changes over time, this feature does not seem useful in tracking movement. For this function a sliding window scheme was used.

3.2.2 Classifier Methods

In this section, I will review the non-linear classifier methods (K-Nearest Neighbour (KNN) and Neural Gas (NG) family of methods) and the general structure of the BN-Inception network used by the TSN classifier.

A review of the KNN algorithm is in order, as it is a building block for the usage of topological descriptor algorithms (SOM and NG), in essence unsupervised classifiers, and assigning labels. This is done by relying on the idea of distance function as representative of similarity[1].

I will review the progression of topological descriptor algorithms from the Neural Gas family, briefly explain their relationship with SOMs, with each other and general behaviour as classifiers and the chosen classifier structure for chapters 3 and 4.

3.2.3 K-Nearest Neighbours Classifier

A KNN [35] is arguably one of the simplest non-linear classifiers possible. In a sense it implements an extension of a table look-up with the addition of a similarity measure, known as distance metric [4]. For a classification attempt of a point P of dimensionality N, a set of labelled observations (from your training data) could be imagined as a cloud of points represented in a hyperspace of size N and uses a parameter K to consider the K-nearest labelled points to our point P, i.e., have the smallest value of the distance metric to that observation, and computes what is the most common class label ascribed to those observations and returns the result.

This supervised classification method was successfully used both as a way to apply labels to the unsupervised Growing when Required Neural Gas and to classify skeletons for my work on Fall Detection [86] and used solely to classify actions with good

[1]This labelling does not need to be non-linear and can also be done by other algorithms, say an SVM with a linear or custom kernel.

results for my attempt at an activity classifier (see Chapter 4). Different values of K were used, but it was found that generally considering K=1 or 1-NN, that is, only the label of the closest neighbour, is a sound strategy [121, 35]. Also additional common distance metrics were evaluated (such as cab-drive, generalised Minkowski distances and Malahanobis) as well as my own custom distance function (see section 4.2.6 for more details), however my testing indicated that the Euclidean distance performed better.

Slight Alterations to KNN to Deal with Time Sequences The use of KNN in the context of classifying skeletons is based on the assumption that a particular skeleton pose represents an action. A natural extension of this use, implemented by others[121] is to use not only the pose but a sliding-window pose sequence (see section 3.2 for more details). This slice of skeletons should represent an instance that only happens on a particular action, that is, it does not account for hidden states or transitions. So as to mitigate that, a sliding-window was used and often in my tests combined with a custom low-pass filter, more specifically a moving mode based filter, which is a small extension of the idea of a low-pass filter to handle labelled classes. The way this mode based low pass filter works is, for an observation $P(t_0)$ made in time t_0, if I have a parameter called the mode length M, by ascribing to this point $P(t_0)$ the most common observation label from the set of $\{P(t_0 - M), ..., P(t_0)\}$ classified skeleton poses (or sliding pose sequence). This type of filtering was used in my Fall Detection (see section 3.3.3) to improve classification considerably, however it did not seem to be particularly useful in a more general action classifier (see Chapter 4).

3.2.4 Neural Gas

A NG is described by Martinetz [110] as a data compression technique. The basis of this approach is to represent a manifold of data by employing a finite number of samples, i. e., a "codebook", to represent the whole manifold. This is done by matching each observation data to a specific code in the codebook, which is found by a "distortion measure", which I, for practical reasons describe as a distance

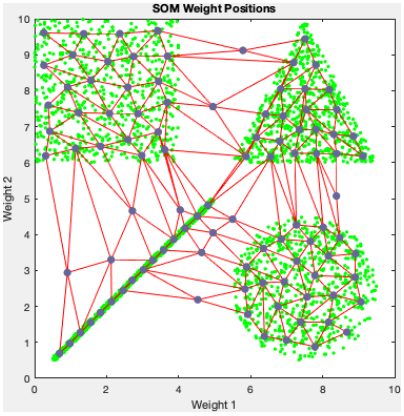


Figure 3.1: A sample example of a SOM from MATLAB's nctool being fit with the test 4 shapes clusters (square, triangle, line and circle). Note some points in between shapes (extraneous); those could, in my implementation of a topological based classifier represent missclassifications if the datapoint being presented is between a shape and the extraneous point, but the extraneous point itself is near another shape. Using 10x10 grid, i.e. 100 points.

metric.

One may draw a parallel here with the situation of the 1-NN(see section 3.2.3), in which I use for the codebook the whole training dataset. This way one may view the neural gas as one of many compression techniques used to make a KNN search set more manageable, that is, limit the number of distance functions that need to be calculated to identify the closest match. Note that the compression algorithm itself does account for labelling and as such its core functionality is unsupervised in nature.

One of the most widely used compression algorithms of this type is the Kohonen's Self-Organizing Map (SOM) [91], however a SOM has itself a predefined topology, that is, the spacial relationships between the codebook units is fixed and the points or codes or neurons merely adapt their positions so as to represent the training data. Moreover, the number of code units in a SOM is fixed. One of the main motivations of the Neural Gas was to define a type of codebook compression that would allow for topologies themselves, that is, the relations between the points in their neighbourhoods to be discovered as well (see Fig. 3.1).

The basis of the working of this algorithm is to implement an adaptation rule where the adaptation steps from presenting data causes the points to jitter and converge

to the topology being presented by using a Hebbian rule - hence the name neural gas.

A simplified description of the algorithm would be that for each new point from a training set being presented to the gas, for each gas unit I find the set of points which are closer to it, order those points using a distance metric and use an exponential adaptation rule, as well as set the index ij of a binary matrix C (initially zeroed) to 1, so as to represent the digraph which is the representation of the topological structure of the presented data. Age of connections are also tracked (as when neurons from the gas move, their connectivity patterns may change) and are set to zero if the point is closer to the gas unit than the unit being matched, or if it is further from it, the age of connections is increased. After this age reaches a certain threshold, the connection is dropped, that is, the index ij of matrix C is set to zero.

For a more complete an formal description of the Neural Gas algorithm, I suggest referencing to the original paper from Martinetz [110].

3.2.5 Growing Neural Gas

The GNG was described by a 1995 paper by Fritzke [53]. It presents an important extension of the basic NG compression algorithm by extending a later version of the algorithm that employs competitive Hebbian learning. The major improvement, however, done by Fritzke was to provide a rule to allow for increasing neural units, as the NG algorithms and its improvement, all seem to predefine a certain number of points and just adapt their positions (much like a traditional SOM would). This, however, was a limitation to its use, as it was difficult a priori to know how many units would be enough to describe a particular topological structure.

The paper shifts focus from describing neural units to point nodes or reference vectors and it also cements the idea of creating a topological map by just finding the closest two units, which limits the number of edge updates as well as restricting the created topology to a fixed dimensionality (usually 2 or 3).

This method implements considerable improvements in comparisons with the ori-

ginal NG in terms of usability, but still presents many different parameters, such as the insertion parameter λ, adjustable movement fractions ϵ_b and ϵ_n, maximum age a_{max}, error decay d as well as not indicate clearly a criterion to decide when to stop the gas growth - it mentions using maximum net size as a parameter or a performance measure, however no clear suggestion is made on to what such criterion should be. This is an important aspect of training the GNG which was attempted to be solved by the GWR.

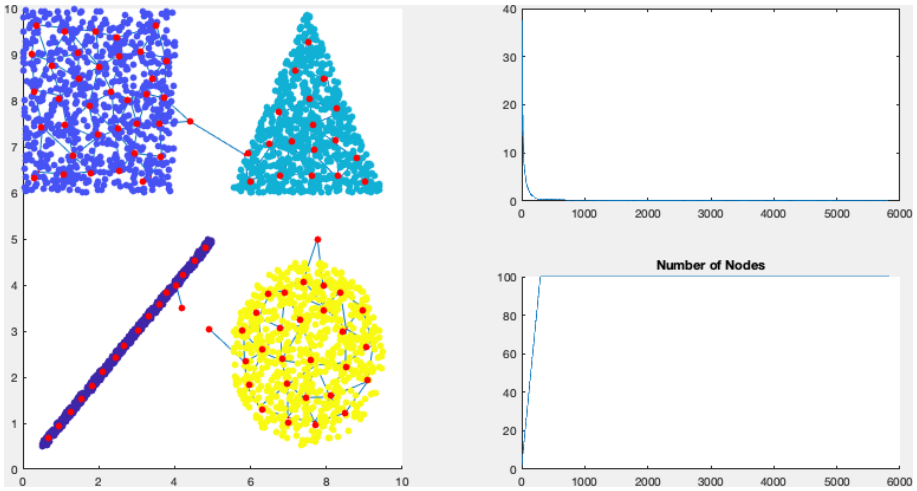
As a starting point of this implementation of the Growing when required Neural Gas (GWR), I used a Matlab implementation available from Matlab FileExchange [92]. A simple example of the use of a GNG on 2D input data (graphed as x,y coordinates) as a simple classifier of clusters with different shapes, using 100 nodes and running for 2 epochs can be seen in Fig. 3.2.

3.2.6 Growing When Required Neural Gas

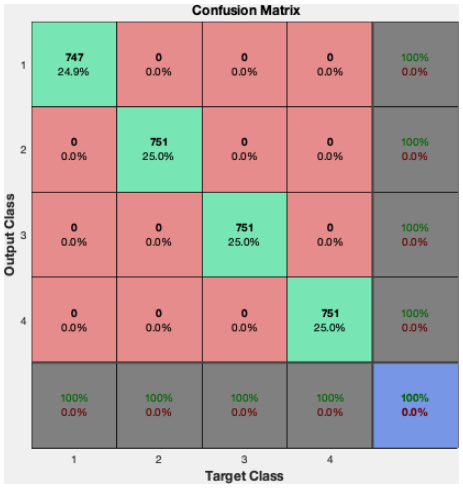
The GWR was proposed by Marsland et al. in a paper from 2002 [109] and it aimed at solving some of the issues with the GNG, namely how to manage the gas growth. I mention in section 3.2.5 the insertion parameter λ, which works by adding with a fixed step frame a new node to the digraph for the unit with the highest accumulated error with its neighbour, which required to keep track of unit errors and limited the growth speed of the topological map, particularly in the initial steps when the network is small.

Marsland suggestion on improving node creation was to use a function to add nodes whenever the existing network did not sufficiently match the input. This had the effect of allowing the network to increase in size very quickly when new data was added and allowing for the network to stop growing naturally when the matching was considered to be of a sufficient quality, addressing two important issues of the NG creating procedure. Additionally, this feature enabled the topological distribution to change through time and also enables the NG to adapt to a changing topology.

This is done by defining an activation measure a and an insertion threshold a_T,



(a)



(b)

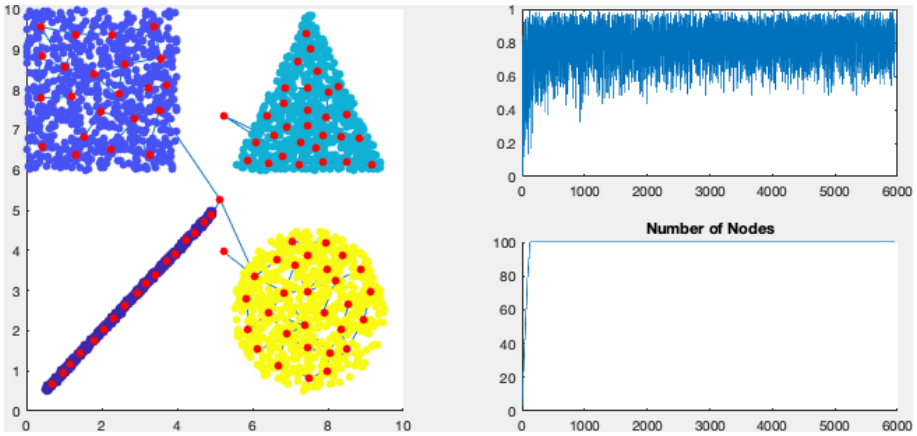
Figure 3.2: In (a) left: 4 shapes to show topology learned by the gas (blue lines) and (points in red) the gas units; on (a) top right the error, (a) bottom right the number of nodes reached. In (b) we can see the confusion matrix, which in this case (GNG) is a perfect classifier (shapes are simple). Note some points are not completely inside the figure, this is due to low number of epochs. Parameters for this test are: $a_{max} = 500$, $\epsilon_n = 0.006$, $\epsilon_b = 0.2$, $age_{inc} = 1$, $\lambda = 0.5$, $d = 0.99$.

which are values between 0 and 1 that represents how well a particular observation is accounted for by gas units. A value of 1 represents that a particular node and the presented data are the same and zero means they are completely different. The measure of similarity between nodes in the gas and the presented data is defined by a distance metric that compares the observation to the existing NG nodes. This activation a is defined as $a = e^{-\|\xi - w_s\|}$, where ξ is the observed data sample and w_s is the weight of the best matching neuron node and $\|P\|$ is the distance function, the norm of P in a given metric (usually Euclidean). If this activation a is considered below the threshold a_T, then the network grows and a new node is added in a similar fashion to the node adding from the GNG. An important addition of this paper is, as well, presenting a use for NG in anomaly detection in robotics, showing that only the GWR (when compared to a GNG and a Reduced Coulomb Energy network [135]) was able to present a steady progression of presentations of novel and not novel stimuli during different runs of the experiment.

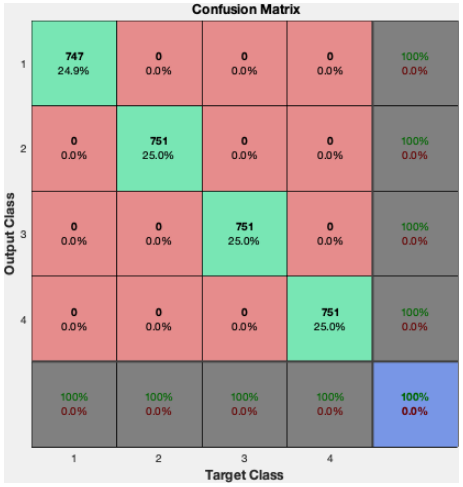
For a more complete and formal description of the whole GWR algorithm one may refer to the original paper from Marsland et al. [109] which also presents a good review of different compression and topological descriptor algorithms based neural gases.

My initial implementation of the GWR, built with direct comparison with the GNG is available in its Matlab version in Matlab File Exchange [11] and on GitHub in `https://github.com/frederico-klein/GWR_GNG_classifier`. An Octave compatible and a Julia implementation of just the GWR algorithm are also available under `https://github.com/mysablehats/gwr`.

As I did for the GNG, a simple example of the use of a GWR on 2D input data as a simple classifier of clusters with different shapes, also using 100 nodes and running for 2 epochs can be seen in Fig. 3.3. Note a perhaps slightly better representation with the GWR when compared with GNG; results however depend on proper gas parameters being set and on particular random seed from MATLAB's Random Number Generator function.



(a)



(b)

Figure 3.3: In (a) left: 4 shapes to show topology learned by the gas (blue lines) and (points in red) the gas units; on (a) top right the activation of the closest node, (a) bottom right the number of nodes reached. In (b) we can see the confusion matrix, which in this case (GWR) is a perfect classifier (shapes are simple). Note some points are not completely inside the figure, this is due to low number of epochs. Parameters for this test are: $a_{max} = 500$, $\epsilon_n = 0.006$, $\epsilon_b = 0.2$, $a_t = 0.80$, $h_0 = 1$, $a_b = 0.95$, $a_n = 0.95$, $t_b = 3.33$, $t_n = 3.33$ (Parameters were chosen to output a similar node distribution to Fig. 3.2).

3.2.7 Multilayer Growing When Required Classifier

Using an unsupervised method for topological description [55] of tasks is not a new idea, since these methods have many possible advantages such as the ability to "operate autonomously, on-line or life-long, and in a non-stationary environment".

The basis for using a NG type is to use as inputs the skeletons, treated as a multi-dimensional vector, that is, a 3*N vector, where N is the number of joints under a particular type of representation, and obtain a set of prototype nodes with a gas that describe possible configurations of skeletons in a 3D space. What I expect is that, by averaging nearby skeletons, as a NG does, I get a more stable representation of that particular pose, that is less noisy than the original data, with the additional compression which is provided by having a number of nodes smaller than the number of data-points in the original dataset.

I chose to replicate the infrastructure implemented by [121] for its overall performance in the CAD60 database and the theoretical generality of the method. From my personal understanding, the GWR as a GNG or a Self-Organizing Map (SOM), implements a data compression and filtering algorithm that uses later a KNN for classification. In this sense, the performance should be similar with the KNN (the best or second best method in the work of Akella et. al [151]), to a degree in which it should be slightly worse - as it loses some information - or better - if it manages to filter out noise and thus increase its generalisation properties. This is inspired by the idea that most of the learning could be on the feature selection as Akella tried to demonstrate.

A complete implementation of this hierarchical design, a dataset loader with many preconditioning and post-conditioning functions and custom distance measures, as well as visualisations for multidimensional gases, gas fitness and utilities for estimating model fitness are available in GitHub under `https://github.com/frederico-klein/classifier`.

3.2.8 Skeleton Data

More thorough descriptions [129] of the data obtained from the depth sensor should be referenced, but basically it is basically a set of J points (where J is the number of joints) with x, y and z coordinates, each representing a landmark on the body in time [33] in a 3D space. I represent thus a particular pose as the concatenation of these J points, such as that for each time frame k I have a pose p represented by the matrix:

$$p(k) = \begin{bmatrix} j_{1x}(k) & j_{1y}(k) & j_{1z}(k) \\ j_{2x}(k) & j_{2y}(k) & j_{2z}(k) \\ \dots & & \\ j_{Jx}(k) & j_{Jy}(k) & j_{Jz}(k) \end{bmatrix} \tag{3.1}$$

An action sequence represented on discrete time steps $1..K$ could therefore be represented as the multidimensional array resulting from the sequential concatenation of the k-th pose matrices. To use the pose information with a gas I change the representation of the pose matrix p(k) into a vector size $3 * J$ and the action sequence is the horizontal concatenation of the all the k-th, p(k) matrices. One may thus understand the pose vector as a single point in a high dimensional space and an action sequence as a necessarily continuous trajectory in that space.

3.2.9 Sliding Window

The sliding window is a simple solution to a hard machine learning problem which is temporal change representation. With say a window of 2 samples, one records the data from the previous time step, say a vector of length w, and concatenates it with the present data sample, so to have a vector size 2w as its input vector. More formally, in a k time step, with an input $w(k)$, a sliding window size of q, I will have concatenated input as:

$$W(k) = \begin{bmatrix} w(k) & w(k-1) & \dots & w(k-q) \end{bmatrix} \tag{3.2}$$

In my case, I construct input for the next layer not from the raw data, but by doing the concatenation of best-matching neurons from the mapped gas. Let A be a matrix with the set of prototypical neurons, then let the definition of a remapping function M as a minimizer of a simple Euclidean distance be:

$$M(A, w) = \left\{ w \mid M(A, w) \min_{w' \in A} \sum_{j=1}^{J} (w_j - w'_j)^2 \right\} \tag{3.3}$$

Then, the response from the gas as the best matching neuron of an input pose w(k) is defined as:

$$\hat{w} = M(A, w) \tag{3.4}$$

Using as an example the next concatenation step $q = 3$, for the first layer ($l = 1$) I would have the sliding window of matched neuron units:

$$W_{layer1}(k) = [\hat{w}(k)\hat{w}(k-1)\hat{w}(k-2)] \tag{3.5}$$

The concatenated vector for the second layer (l=2) would be:

$$W_{layer2}(k) = \left[\widehat{W_{layer1}}(k)\widehat{W_{layer1}}(k-1)\widehat{W_{layer1}}(k-2)\right] \tag{3.6}$$

Which expands into:

$$W_{layer2}(k) = \Bigg[\widehat{[\hat{w}(k)\hat{w}(k-1)\hat{w}(k-2)]}\,\widehat{[\hat{w}(k-1)\hat{w}(k-2)\hat{w}(k-3)]} \\ \widehat{[\hat{w}(k-2)\hat{w}(k-3)\hat{w}(k-4)]}\Bigg] \tag{3.7}$$

Which does not agree with the presented on my reference paper as it was described that the whole algorithm takes 9 sample steps to produce a result. I introduced an additional sampling parameter p, which is the number of time samples to jump after constructing a concatenated long vector. It implements undersampling by discarding

the p interleaving samples from the full sliding window, defined as:

$$W_{q,p}(k) = \begin{cases} \hat{w}(k)\hat{w}(k-1)\dots\hat{w}(k-q) \text{ if } k \pmod{p+1} = 1 \\ \text{not defined otherwise} \end{cases}$$

In my case, I use the parameters ($q = 3$, $p = 2$) in most simulations, the first layer concatenation stays the same long vector $W_{layer1}(k)$, except that the p interleaving samples are missing and I have for my second concatenation the resulting best matching neuron would be:

$$W_{layer2}(k) = \Big[[\hat{w}(k)\hat{w}(\widehat{k-1})\hat{w}(k-2)][\hat{w}(k-3)\widehat{\hat{w}(k-4)}\hat{w}(k-5)] \\ [\hat{w}(k-6)\widehat{\hat{w}(k-7)}\hat{w}(k-8)]\Big] \quad (3.8)$$

Which requires the 8 previous data samples as well as the current value (so 9 samples in total) as I wanted.

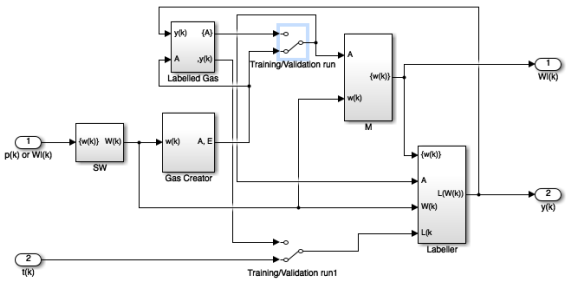
3.2.10 Classifier Architecture

The classifier was implemented as a serial chaining of gas subunits. This was done to enable different structures to be tried with minimal effort. Although more architectures, with just 1 or even 7 chained gases, were tried, those did not seem to outperform the version with 5 gas subunits implemented by Parisi's [121] paper (see Fig. 3.4). All classification attempts in this text were thus done using the same version as Parisi's, that is, 2 parallel sets of 2 gas subunits in series, each stream dealing with either pose positions or pose velocities and the last gas that integrates both.

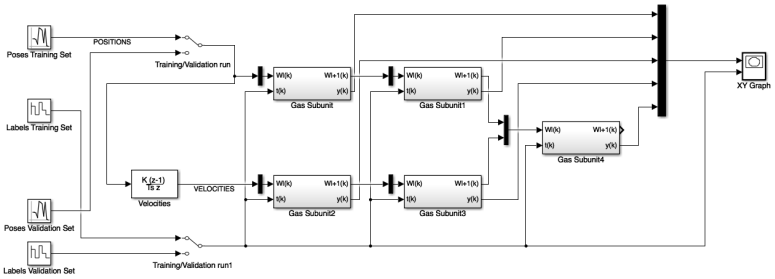
3.2.11 Gas Subunits

For each gas there are 5 main chained elements that are responsible for implementation estimation and classification:

- Sliding Window: implements the temporal concatenation of sample (also im-



(a)



(b)

Figure 3.4: (a) diagram for each gas subunit. (b) diagram of end architecture after linking all gas subunits. This particular structure was chosen to replicate Parisi's [121].

plements concatenation of multiple streams in case they exist).

- Gas Creator: receives data samples $p(k)$ or concatenated poses $W_l(k)$ and implements the learning algorithm for either the Growing When Required neural gas or the Growing Neural Gas.
- Mapping: finds the best matching pose from the nodes matrix A corresponding to each sample from the dataset.
- Labeller: simple labelling function that assigns the label of the estimated concatenated pose as the same as the label of the pose to which it best matches.
- Activation checker: during training, checks to see if points are able to be well represented by the gas, and if not removes them from the sample.

3.2.12 Whole Implemented Architecture Overview

A simple description of the gas classifier is as follows. I start start with a dataset with a set of action sequences (sequence of points in high dimensional space),

calculate velocities from it and construct 2 distinct vectors from the data that are processed in parallel. One for positions and another for velocities. Afterwards I separate it into training and validation set. For the training set I take the parallel streams of positions and velocities are then presented to their respective GWR neural gas (one may also choose a GNG for comparison). The gas function constructs 2 matrices as a result: a matrix with a set of prototypical neurones (or in my case skeleton poses) that best represent my dataset and an Edges matrix that represents the neighbourhood relationships of these neurones. Currently I disregard their neighbourhoods, and consider only the prototypical neurones. After this, I use the prototype neurones to reconstruct my initial dataset using instead of the original data, these prototypical neurones, in the following manner: for each point in my dataset I calculate the distance between itself and all prototypical neurones from the gas and I consider the best matching neuron as the one with the smallest distance. The best-matching neuron is then used to construct a new input dataset for the next layer. These best-matching units (or neurones or prototypical poses) are then temporally concatenated (usually with parameter $q = 3$ and $p = 2$) and are fed to the next layer of gas. After all gases are obtained, a simple labelling procedure is performed, where the label of each prototypical neuron is taken as the label of the point in the dataset to which it is closest. For the validation set the procedure is slightly simpler: I merely find the best-matching unit from the trained gas, label it and do the temporal concatenation in the same structure as dictated by the architecture network. The sequence of gases and their interconnections is defined by simple graph network model that accept some variation in the construction of the layers. What was most commonly used was to separate velocities from positions in parallel streams, process a gas for each, then concatenate q poses (usually $q = 3$) and create the input from the next layer, again separating positions and velocities, concatenate again q times and combine both poses and velocities into a longer vector to create the input from last layer.

3.2.13 Construction and Randomisation of Validation and Training Sets

The implemented architecture allows currently for loading 3 different datasets, the TST v2 fall dataset, the CAD60 dataset and an artificially generated simplified Falling Stick model Dataset for testing purposes. The datasets in question present a set of N subjects, each of them performing M actions. The data from each particular datasets are randomised between validation and training sets in 2 distinct manners:

- Type 1 (known subjects): all the actions are put in one group and validation and training sets are picked from $M \times N$ possible actions;

- Type 2 (unknown subjects): each subject is assigned to be a part of the validation or the training set, from a N set of subjects.

As I understand, the type 2 randomisation is a more realistic test scenario, as perhaps the heights or limb lengths or mannerisms from the subjects I trained my algorithm on maybe be different from the actual end user. Unfortunately I do not have all participants performing all tasks on the CAD60 database, so this test is not possible on that set. The dataset was separated into Validation and Training sets containing 80% and 20% of data respectively, before each training. They were separated by subject, so that each subject had all of its actions belonging exclusively to one set. Since actions did not last exactly the same amount of time, the percentage of this separation was not necessarily accurate.

3.2.14 Preconditioning

The GWR algorithm is not translation invariant, so the first action performed on the data was to select a joint - based on my reference algorithm I used the hips and subtracted the offset from the hips joint in both the z and x coordinates from that from all other joint vectors. Secondly, I normalised (scaled) the data so that after scaling variance of the data would be equal to 1. The final step was to implement a centroid generating function, so to generate a smaller dimensionality representation

of the skeleton poses, in a similar fashion to the function tested by Parisi. I created a model of 3 centroids that were the average position of the skeleton points such that the upper centroid was composed by the joints: head, neck, left shoulder, right shoulder, left elbow, right elbow; middle centroid corresponded to torso and lower centroid: left knee, right knee, left hip, right hip. Many other preconditioning functions are available on the supplied code and maybe be tried by the interested reader.

Gas Chained Network

The architecture I implemented is a reproduction of the modular gas architecture implemented by Parisi. Basically the skeleton poses are used to construct a velocities matrix $v(k) = (w(k) - w(k-1))/\Delta t$ (Δt here being the time-step between samples, although its importance may be limited as the sample rate is supposed to be constant and I use normalisation procedures) separated into pose and treats poses and velocities independently. The algorithm implemented can deal with an arbitrarily long chain of linked gas classifiers and other structures may be tried. Most tests use 2 sequential gases for position and velocity and a final gas that is fed the concatenated data from the previous layers. If all dimensions of pose vector were used (for the TST database, that means 25 joints $\times 3$), the first layer would have poses described by 75 dimension vectors for both poses and velocities, the second layer (as it does a temporal concatenation with sliding window q=3) would have length 225 for both position and velocities and the final aggregation layer has input vector size of 1350 (225 from each position and velocity layers concatenated by 3).

3.2.15 Dataset Loader

A generic dataset loader was devised to create suitable validation and training sets from the following datasets, in a way that they could be easily compared with Matlab's standard neural networks classifier implemented by the nnstart wizard. This was done in such a way that the datasets would be separated between training and validation sets (80% of data samples being used for training and 20% for valida-

tion[23]) in two different ways: you could have either known subjects, that is, both validation and training set had all the subjects, but different actions performed by them, or with unknown subjects, where the subjects in the validation set where completely different from the ones in the training set.

3.2.16 Datasets Used

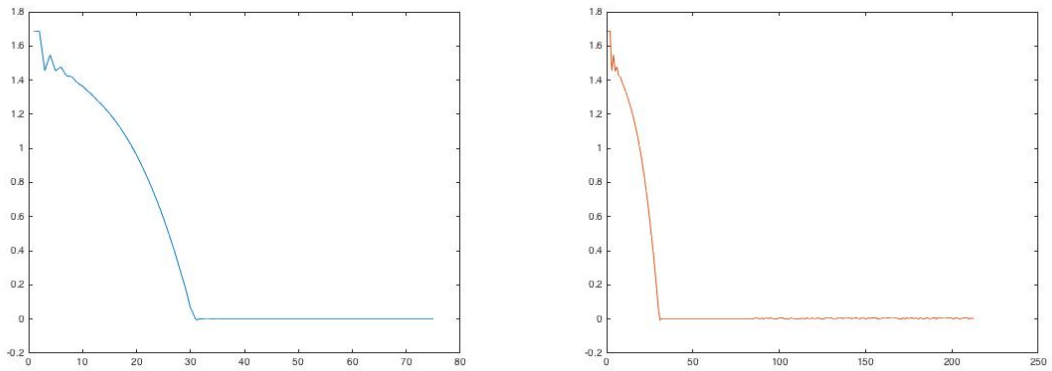
The skeleton datasets present actions as a sequence of skeleton poses that change over time. Skeleton descriptions vary in the number of points that describe them, depending on version of KINECT algorithm employed to generate them, with KINECT v1 having 15 points per skeleton, KINECT v2 having 25 points to define a skeleton (with more points used to describe hands and feet) and KINECT for Windows using 20 points to describe a skeleton.

As such they often contain RGB-D data (and in the case of the TST v2, IMU data as well), but this information was not used for the work presented here. I will present, from this class the Tstv2 dataset (see Section 3.2.16), the falling stick model (see Section 3.2.16), the CAD60 (see Section 4.2.1) and my own collected version of CAD60 actions, Ourset (see Section 4.2.1).

TST Fall Detection ver.2 Dataset The TST v2 dataset contains skeleton positions Microsoft Kinect v2 and IMU data for 11 subjects performing either ADLs (activities daily living) and simulated falls. The subjects were between 22 and 39 years old, with different height (1.62-1.97 m) and build. Each of the two main groups (ADLs or Falls) contains 4 activities is repeated three times by each subject [56]. For the present study only the skeleton joints in depth and skeleton space and time information were used. The accelerometer, as well as other data, were not used for

[2]This proportion was chosen as it is commonly used in the literature, however a more precise estimation of the best validation training ratio can be done using the formula $\frac{f^*}{g^*} = \sqrt{\frac{Cln(N/\alpha^2)}{h_{max}}}$, where N is the number of families of recognizers, in our case the number of nodes used, e. g. 1000, α, the risk of being wrong, chosen to be 0.05 as in the paper from Guyon that proposed this formula [64], $C = 1.5$, the constant of Chernoff bound, the largest complexity $h_{max} = 405$ to be considered on the last gas with 9 samples long with 15 joints in 3D space, f^* the fraction of t that should be reserved for validation and g^* the fraction of t reserved for training. The results give us 0.2186, very close to our 20% used empirically.

[3]Note that since actions are kept as a whole, depending on the randomization chosen, these split ratios will vary.



(a) (b)

Figure 3.5: Two typical results (a) and (b) from my mathematical model. Both of them generated with the equations 3.9 and 3.10.

this algorithm.

Falling Stick Model A simple stick model was developed to test the ability of a Growing When Required multilayer with a sliding window classifier to discriminate between action sequences that included a fall. I modelled 2 different activities, a fall and a walk as the movement of a stick in a 3D space and then simply substituted the stick for a typical skeleton.

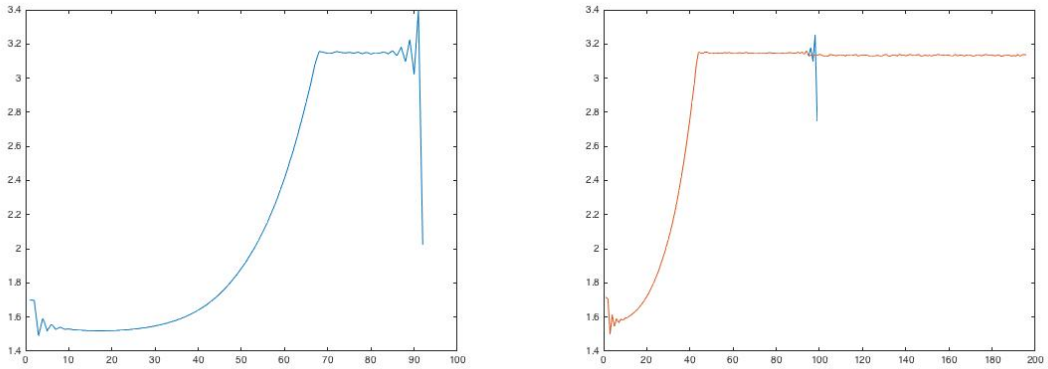
Fall

I simulated the fall of a person by a free falling inverted pendulum rod with a random initial pitch angular velocity θ and perfect slippage. It can be shown [122] that the kinematic differential equations that describe angle and position changes for a rod are:

$$-mg\frac{L}{2}cos\theta = \left(I_c + \frac{mL^2}{4}cos^2\theta\right)\ddot{\theta} - \frac{mL^2}{4}cos\theta sin\dot{\theta}^2 \tag{3.9}$$

$$0 = m\ddot{x_c} \tag{3.10}$$

The equation 3.10 basically means that the perfect slippage does not alter the x position of rod's centre of mass. And, approximating a person by a slender rod, one has $I_c = \frac{mL^2}{3}$. Some alterations were done to this model, so that when reaching a



(a) (b)

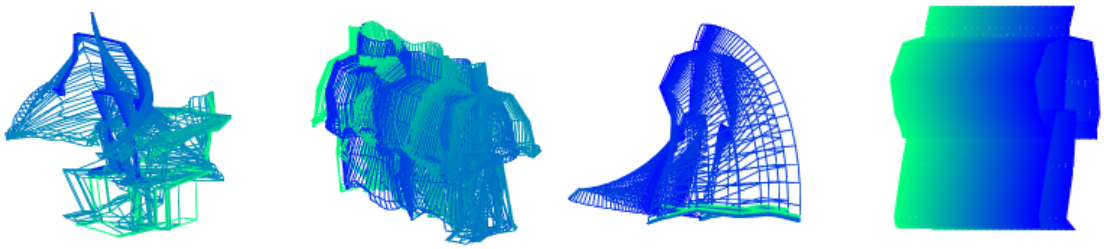
Figure 3.6: (a) Pendulum ODE45 solver reaching an unstable point due to numerical approximations. (b) The corrected version stabilises the angle output with a random noise after reaching a resting position within a certain range of resting positions.

resting position, it would stay there, as the ODE solver can sometimes yield unstable solutions (see Fig. 3.6).

The model also was given simulation parameters to add random noise in the variables of height (1.6 - 1.9m), initial position (within a square area) and any initial yaw angle.

Walk

The simulation of a person's walk was done by simply doing a linear space of displacements inside the area that would be covered by the Kinect sensor, with random initial positions and walking angle. The end results after a skeleton is overlaid on top of the simple stick model of both the non-linear differential equation model and the simple walk model (and their comparison with the dataset data) can be seen in Fig. 3.7.



(a) (b) (c) (d)

Figure 3.7: (a) a typical fall from TST v2 dataset. (b) one of the ADLs from the TST v2 dataset, a walk. (c) a typical fall from this model. (d) a "walk" from this model.

3.3 Results[4]

For all the results here presented, the simulation parameters for the GWR neural gas are the same as in the Parisi reference paper[121]. One of my main concerns while evaluating the performance of such an algorithm was to avoid cherry picking, a sample of different learned gases was created, by randomising data presentation to the gas after temporal concatenation.

For both of my models I present the results of a set of gases with the same parameters only differing random initialisations. To avoid cherry-picking, I choose the best performing gas under training measures and show the measure of its validation counterpart.

3.3.1 Cornell CAD60 Dataset

As a means of comparing my implementation with that of Parisi, I also tested my architecture on the CAD60 dataset. I used the same simulation parameters described in his paper (preconditioning was also different, with only the removal of feet joints and no normalisation), only a smaller number of epochs (10 epochs). Apparently my implementation does a lot of overfitting, as it reaches 99.6% accuracy on the

[4]Initially I had reported the variations (in form of ± 1 standard deviation) of most gas results, however under a deeper scrutiny, those were removed, as their behaviour is still debatable and dependant on both the specific model used as classifier and the specific dataset [12, 90, 192]. While I still believe that the estimation of variations were adequate under the published study [86], a proper evaluation would require a more strict analysis of the specific NG algorithm and require more data, which is a limitation of our studies (relatively small datasets) to demonstrate stability [15, 83]. Under these circumstances it is uncertain what kind of information would a statistic such as a binomial test really represent. Most of the variation results are still available for conference however, under bound copies of relevant papers chapter of this thesis.

training set (average: 99.38% for 8 trials) but only reaches 71.7% on the validation set. It is interesting to notice that the misclassifications were limited to some specific actions , with others having the same accuracy (greater than 90%) in both training and validation set (see chapter 4 for more details).

3.3.2 Falling Stick Model

My algorithm, even with a much smaller network (100 nodes), seems to be quite consistently capable of classifying my faux fall/walk model. I simulated 20 subjects with varying height and starting orientations performing either a fall or a walk. The peak accuracy on the validation set of my implementation was 98.33% (average: 97.12% for 8 trials).

3.3.3 TST Fall Detection ver.2 Dataset

As a means of comparing the performance of my implementation, I also classified the TST v2 dataset using a Recurrent Self-Organising Map (RSOM) implementation. The RSOM used the same preconditioning as I did for the chained gas classifier and a set of 3 consecutive poses $(p(k), p(k-1), p(k-2))$. The simulation parameters were: 900 nodes, 30 epochs, method 'RSOMHebbV01', chosen to be as close as possible to the implemented gas classifier. The peak accuracy on the validation set of the RSOM with these parameters was 78.76% (average: 77.67% for 5 trials). The accuracy on the validation set of my implementation will be discussed in detail in the next sections.

Effects of Increasing Epochs

Upon visual inspection, no clear real pattern emerges from increasing the number of epochs on the gas architectures with the parameters tested. Some tests with small number of nodes (from 20-100 nodes) had smaller effect on changing number of epochs, while more epochs were required to have a gas express units from more than some actions (results not shown). The gas has a tendency of populating units with very similar ones, if parameters are not set correctly and increasing number of

Epochs	Validation Set	Training Set	Maximum
E = 10	73.99%	88.35%	75.6%
E = 20	72.74%	88.06%	74.9%
E = 30	77.3%	90.8%	77.3%

Table 3.1: Change in accuracy for TST v2 dataset with varying number of epochs for chained GWR classifier with 1000 nodes with N = 8 trials except for 30 epochs, where N = 1.

epochs fixes those issues naturally, but at cost of unreasonable running times.

This effect is consistent with what was expected from the system, that is, as the datasets in question were several hundred times larger than the gas I wanted to train, there is a similar effect on changing gas unit as does training over many epochs.

Additional steps were taken to make sure unit representation was evenly distributed among gas units (see Chapter 4), like making sure α insertion parameter would increase progressively and as such we would not have the issue of having too many very similar skeletons added as nodes an a single "more common" skeleton that would be updated most often. Again, due to chaotic behaviour of the NG, statistical analysis of variability of the results would entail verifying not only classifier stability, but also cross-validation stability, which is hard to demonstrate generally and would be of limited use. We refrained from using statistical analyses here as this is still a contentious point in literature and was not clear exactly how such results could be interpreted.

Learning across layers

In order to understand how learning happens across layers I analyse the output classification from 5 gases with 1000 nodes run over 10 epochs (see Table 3.2) the results reflect what I would expect: there is a steady increase as I progressed through the layers and there is a gain in accuracy.

Table 3.2: Progression of classification accuracy within different layers for chained GWR Neural Gas classifier with 1000 nodes and run over 10 epochs (for 8 trials).

Gas Element	Validation Set	Training Set
GWR gas 1 Pos	67.69%	93.04%
GWR gas 2 Vel	64.80%	69.43%
GWR gas 3 Pos	66.93%	90.45%
GWR gas 4 Vel	65.14%	70.66%
GWR gas 5 STS	73.99%	88.35%

Table 3.3: Accuracies (in %) after applying the moving mode filter on classification results of the final gas unit of the classifier (for 8, 8 and 1 trials respectively).

Epochs	10		20		30	
Filter length	Val	Train	Val	Train	Val	Train
5	81.25	95.05	79.72	79.72	82.7	94.9
10	85.95	95.74	83.87	96.00	86.9	95.5
15	88.57	95.31	85.91	95.46	88.6	97.0
20	89.95	95.34	86.70	95.26	88.6	95.0
25	90.16	94.32	87.37	94.47	89.2	95.4
35	90.20	92.29	88.49	92.44	91.5	93.3
40	89.28	91.23	87.75	91.33	91.5	92.2
50	87.39	88.84	85.35	88.80	91.3	89.9

Mode Filter

With the intention of performing some sort of temporal filtering, I implemented a moving mode filter. The moving mode filter had very interesting effect on the classification results of the TST v2 database (see table 3.3.3). Highest classification accuracy achieved (See table 3.4) was 94.2% on the validation set by 3^{rd} gas with 1000 nodes and 10 epochs with mode filter length of 35 data samples.

Table 3.4: Confusion matrix for my most accurate gas classifier. The calculated accuracy for the validation set is 94.2%, higher than the 92.0% training set.

Validation Set		Target 1	Target 2
Output	1	1532	36
	2	124	1054

Training Set		Target 1	Target 2
Output	1	4006	258
	2	328	2746

3.4 Conclusion and Discussion

The resulting classification scheme does the task which I want, that is, discriminate falls within the TST v2 dataset, it does it better than the RSOM and it does it consistently with around 90.2% accuracy while using the mode filter. One must note that adding a moving mode filter of size 35 means a delay of 11.67 s (since I need $(9 + 1) \times 35$ samples @30Hz) much more than the 0.6s Parisi reported, since he achieves high accuracy without needing a filter. Another possibility for this is could be due to my decision of implementing noise detection only on the training run, but there may be other causes for this issue. Nonetheless, the delay we have of 11 seconds is still adequate for the application and as such, I believed I achieved my goal and I have now a classifier of falls with openly accessible code.

Another important discussion is related to the convergence or not of classifier responses to a "good" classifier and how the GWR and other NGs behave. I had initially reported the variations of classification results of most gas classifiers, however under a deeper investigation motivated by review of this work, I decided those would better not be included in the final version of this document. The behaviour of a classifier under variation of cross-validation folds alone is still debatable and can be shown to decrease (which is expected, under higher number of folds) but also increase, depending on classifier stability [83] [15] and interactions with a specific dataset [12, 90, 192]. While I consider that the estimation of variations were adequate enough for the published study [86], a more proper evaluation of these effects would require a much more stringent analysis of the specific NG algorithms used, should also include more data (hard to achieve in our studies as we have relatively small datasets) and different datasets entirely, to study stability and convergence. Considering all of those circumstances it is unclear what information would normal statistic analysis confer and as such those could not be interpreted.

A final remark refer to the extension of the results and analysis presented here. This is a sequential work and will be examined further in the chapter 4.

Chapter 4

Activity Detection Using Skeleton Data

4.1 Introduction

In this chapter, I present an extension of the architecture from ICONIP2016[86] paper to attempt to better classify the CAD60 and check for generalisation properties by testing it on my own acquired set (see Section 4.2.1). I also show best results so far and their comparisons with KNN and SVM with a linear kernel as well as present some details about changes done to the Marsland GWR algorithm, necessary for speed-up and to adequate it as a classifier (initially a topological learning method). This algorithm was implemented fully in Matlab[1] and in its latest version is available in full in `https://github.com/frederico-klein/cad-gas/`. A condensed version of this updated implementation and results is available in arXiv.org [87]. As this chapter is heavily based of the work done on fall detection (see Chapter 3), a justification (see Section 3.2.1) and explanations about the general behaviour of NGs can be read in Section 3.2.2.

[1] Matlab R2017a.

4.2 Materials and Methods for Activity Detection using Skeletons

The main structure of the classifier used in this work is described in the following algorithm (Alg. 1): The description of this methods section follows the order with which each function is called within the classifier algorithm, with the referring section added as a comment in the pseudocode table Alg. 1. I also added most of the optional variations that were implemented in the following text for completion purposes - it may not make much sense to correct for rotation variation twice, or to use different distance metrics for finding neighbouring points in the gas and another one to find the best matching unit for creating input for next levels and labelling, but the implemented algorithm allows this to be possible and those options explorable.

4.2.1 Datasets Used

For the presented work I intended to classify the data from the CAD60 and its 4 subjects and 12 actions (see subsection 4.2.1 for more details). I trained mostly on this set and aimed to extrapolate the model obtained from this data to classify my own captured set to hopefully cross-validate my classifier method. For this extended testing, I used a smaller dataset captured (Ourset) by my own MATLAB implementation using 2 subjects in an office environment (see section 4.2.1 for more details). Training and validation splits will thus refer to splits of the CAD60. For the training of the model and classification of Ourset I used the whole CAD60 and tested on Ourset.

A special mention should be done involving the representation of skeletons used in Ourset when compared to the representation used in CAD60, since considerable differences exist. This is due to Kinect v1 and Kinect for Windows having different representations of 3D space (with different units being used for lengths), so skeletons needed to be normalised for their sizes to be comparable and different skeleton

[2]Later expanded to 'compressors' structure which allowed also for GNG and SOMs to be used.

Algorithm 1 Main Classifier structure

```
1: simvar ← simulation variables
2: params ← parameters for gases
3: data ← dataset                                                    ▷ See Sec. 5.2.1
4: for all Subjects (or Actions depending on validation type) do
5:
6:     function RandomizeDataset(data,simvar)                        ▷ See Sec. 4.2.2
7:         According to simvar:
8:         Choose validation type
9:         Choose ordering type
10:        Divide data into validation and training sets
11:        return trainingDataset
12:        return validationDataset
13:    Preconditioning(trainingDataset,simvar)                       ▷ See Sec. 4.2.3
14:    Preconditioning(validationDataset,simvar)
15:    SaveDataset(trainingDataset,validationDataset,simvar)         ▷ For
   replicability
16:    for All layers do
17:        for trainingDataset,validationDataset do
18:
19:            function GasTraining(trainingDataset,params)          ▷ See Sec. 4.2.4
20:                longInput ← SetInput(trainingDataset,params)      ▷ See Sec. 4.2.5
21:                if trainingSet then
22:                    gas ← CreateGWR^2(longInput,params)           ▷ See Sec. 4.2.6
23:                else
24:                    gas ← alreadyTrainedGas
25:                bestMatchingUnits ← GenerateBestMatching(longInput,gas,metric)
26:                if trainingSet then                               ▷ See Sec. 4.2.8
27:                    Labeller ← CreateSomeLabeller(gas,targets,metric)
28:                else
29:                    Labeller ← LabellerFromTraining
30:                output ← Labelling(longInput,Labeller,bestMatchingUnits,params)
31: Calculate Metrics
32: Save Current Trial
```

joint description, with the Kinect v1 using 15 joints and the Kinect for Windows describing a skeleton with 20 joints (see Fig. 4.1). [3]

So as to be able to compare those 2 different skeleton definitions, a simple conversion table was used (see Appendix A.1 for details), however the matching was not precise as can be seen in Fig. 4.1.

CAD-60 For this work I chose to classify the activities from the CAD-60 available at [162] and my own dataset (Ourset see subsection 4.2.1) created in the mould of CAD60. The CAD60 is a dataset containing 4 subjects performing 12 different actions, captured by a Kinect version 1. The dataset contains 320x240 pixels RGB-D (RGB plus Depth) motion sequences and skeletal information acquired at a constant frame rate of 30fps [34]. The skeletal information is composed of 15 points per skeleton with x,y and z coordinates corresponding to the best estimate location of joint positions in a 3D space as extracted by its algorithm from the depth data. For the present work the RGB-D will be disregarded and only these joint positions will be used.

The 14 actions were: 'brushing teeth', 'cooking (chopping)', 'cooking (stirring)', 'drinking water', 'opening pill container', 'random', 'relaxing on couch', 'rinsing mouth with water', 'still', 'talking on couch' , 'talking on the phone', 'wearing contact lenses', 'working on computer', 'writing on whiteboard', with 'still' and 'random' generally not used for analysis. Some actions, such as 'wearing contact lenses' and 'opening pill container' tended to be shorter than others (around 500 samples for 'wearing contact lenses' and 300 samples long for opening pill container'; normal actions had between 1000 and 2000 samples) and so as to have a more balanced dataset, those actions were repeated twice in case of 'wearing contact lenses' and 3 times in case of 'opening pill container', resulting in 15 samples per subject, and hence the name CAD60 - as there are 60 'valid' actions in total.

[3]Review of literature and resources from Microsoft did not report how differently joint values would change from sensor to sensor, nor could we find any information regarding average errors. As subjects used to gather different datasets vary, we did not see any possible to get an estimate of the error without physically testing the devices under same conditions. This investigation was initiated, but was dropped as our work shifted towards deep learning and networks such as OpenPose [169] would render it obsolete.

These actions were also separated by room (as some actions would be more likely to be done in a specific environment) and often accuracies under this smaller sets are reported, namely: bathroom ('brushing teeth' , 'rinsing mouth with water', 'wearing contact lenses'), bedroom ('drinking water' , 'opening pill container', 'talking on the phone'), kitchen ('cooking (chopping)' , 'cooking (stirring)' , 'drinking water' , 'opening pill container'), living room ('drinking water' , 'relaxing on couch' , 'talking on couch' , 'talking on the phone') and office ('drinking water' , 'talking on the phone' , 'working on computer' , 'writing on whiteboard').

In addition to the skeleton poses, joint orientations were also provided and as well RGB and depth frames for every action collected, however those were also not used in this work.

My Version of the CAD-60 Actions: Ourset So as to enable testing of the classifiers obtained from training on the CAD60 for generalisation properties, I captured activities from 2 subjects (one male and one female) in the moulds of the CAD60 (see subsection 4.2.1). This dataset was created with a KINECT for Windows - a slightly different sensor from KINECT version 1; it was constructed using the same actions producing 640x480 RGB-D motion sequences and a 20 points per skeleton corresponding to locations of 20 different joints. Additional differences include different scaling of skeletons, capacity to track up to 6 simultaneous skeletons among others.

The sequences were chosen to be roughly the same length in duration as the ones from CAD60. However, unlike CAD60, the lengths of the actions were not manually cropped, with the length being specified before each action was captured.

In Ourset, the 14 different actions (12 plus 'random' and 'still') are performed by subjects in a fairly standardised manner and with a simplified office environment. Depth and Image frames were also captured for completion purposes, however those were not used in this work. A detailed table of the actions acquired by subject and their frame lengths are provided in Tab. 4.1.

	Subject 1		Subject 2		Total	
Action	Actions	Frames	Actions	Frames	Actions	Frames
1. brushing teeth	1	1500	1	1750	2	3250
2. cooking (chopping)	1	1500	1	1500	2	3000
3. cooking (stirring)	1	1400	1	1400	2	2800
4. drinking water	1	1500	1	1500	2	3000
5. opening pill container	3	300	3	250	6	1650
6. random	1	1900	1	1900	2	3800
7. relaxing on couch	0	-	1	1450	1	1450
8. rinsing mouth with water	1	1700	0	-	1	1700
9. still	1	1000	1	1000	2	2000
10. talking on couch	1	1700	1	1700	2	3400
11. talking on the phone	1	1400	1	1200	2	2600
12. wearing contact lenses	1	440	2	440	3	1320
13. working on computer	1	1900	1	1800	2	3700
14. writing on whiteboard	1	1800	1	1800	2	3600
Total	15	18640	16	18630	31	37270

Table 4.1: Actions performed by subject and length of one or multiple actions and total included in dataset.

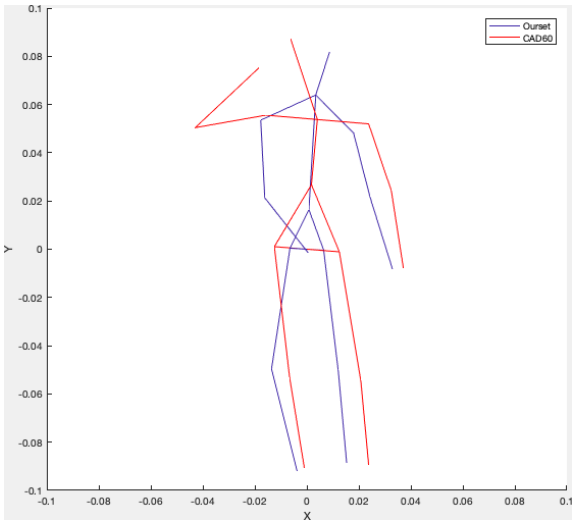


Figure 4.1: In red a skeleton CAD60 and in blue a skeleton from Ourset, both seen with hips facing the sensor. Even though the skeletons correspond to different persons in different poses, note the seemingly different definitions for spine and torso (higher in Kinect v1) as well as wider hips and difference in neck joints. These differences are due to a combination of: (1) different person with different limb lengths and position (2) possible different algorithm implementation.

4.2.2 Building Training and Validation Dataset Splits[4]

Segmentation

The separation of validation and training data was a cornerstone step while training this classifier, as improper division might present an oversimplified problem and present a nonworking classifier as a functional one. Conversely, an overly stringent cross-validation strategy, might not present enough data for the algorithm to generalise its response and make it unable to learn the desired problem and/or be too time consuming.

Leave-one-subject-out The cross-validation method used was a leave one subject out approach. This was chosen as I see it as a more realistic approach, in which a trained classifier is provided as is, without any data being required as input or in a setup phase from the user what will employ it. For most trials, a randomised subject from the four was excluded from the training set and used to calculate accuracy, and for my end result the gas accuracies were calculated 4 times, each time excluding one of the subjects from the set and then averaging the results. According to Zhang [191], the de-facto validation scheme is leave one subject out, perhaps for its simplicity in implementation but also as it seems to be a robust and sensible way to perform cross-validation. One may consider unlikely -or at least cumbersome to implement - to have to train a learning algorithm using the data from the very subject I am analysing (although this was not an impediment for commercial software to be deployed, such as with voice recognition and OCR of handwritten data).

Leave-One-Action-Out for All Subjects[5] The dataset was separated by action, in a leave one action out cross-validation strategy, namely on my case this means using 47 actions for training $4 \times 12 - 1$ and the remaining action for validation. The classifier was then run 48 times, each time with one action per subject being excluded from the training set. This validation method was used initially and

[4]This would ideally be performed by our docker based dataset loader as explained in Chapter Chapter 2, however, since the HMDB51 dataset has fixed training set splits (to avoid this issue and make results more easily comparable with each other), this was only implemented in the Matlab version of the code.

would, in practice mean that before being able to classify an action, an initial setup would be necessary for a new subject. It was deprecated due to my algorithm of choice not making use of this information and the time increase in testing with this partitioning.

Leave-One-Action-Out for Random Subjects[5] The dataset was separated by action, in a leave one action out cross-validation strategy, namely on my case this means using 47 actions for training $4 \times 12 - 1$ and the remaining action for validation, making sure to pick only one of each action from random subjects. As I have 12 actions, the classifier was then run 12 times, each time with one action per subject being excluded from the training set. Faster than "Leave one action out for all subjects", but also deprecated in favour of Leave-one-out, due to low performance.

80% Training, 20% Validation The dataset was separated randomly, having 80% of the data being used for training and the remaining 20% used for validation. The classifier was then run multiple times but there is no guarantee that all actions will be present on the training and the validation dataset. Common validation strategy, however, the accuracy of classifiers oscillated too much depending on the particular partition chosen, which complicated analysis.

Dataset Ordering

Another aspect of data presentation was its order on the set. The Growing when required Neural Gas (GWR), does present a rather chaotic behaviour, with the end resulting gas varying immensely based on initial gas nodes and the order with which the data is presented for the gas to learn it.

[5] As we have a peculiar dataset with same subjects executing multiple actions, it may be the case that we learn information from those subjects (such as limb lengths) or leg positions, or even other examples of them doing such an action, that would make it easier for us to classify actions from this subject, given that it is a person whose information we already have in our training dataset. The leave-one-out strategy here would give us an inflated better performance, which we are trying to avoid by having other partition modalities. This is not technically an issue, but it just would mean that training the classifier would be done in a slightly different way. Such aspect is mentioned briefly in the [162] paper, but no suitable generic solution is presented, we aim here to provide some rational which was tested by this work.

Sequential Ordering The data is fed into the gas in the sequence which is read from the dataset as subjects are labelled 1,2,3 and 4 and the actions are alphabetically ordered. This is mainly a choice for its repeatability, but there is no reason to believe that this is the best way to present data for learning, as different wording on labels or subjects would change the resulting trained gas.

Random Ordering After the dataset is constructed in an ordered fashion, a random permutation of the total size of training and validation actions - respectively - is built, and the activities are learned in an arbitrary sequence, as each next action can be any action from any subject.

Random Organised Ordering It may be the case that the gas learns better if it learns all actions of a specified type one after the other, that is, all subjects performing the same action, say brushing teeth, to then change into learning the next action. This ordering was constructed based on the random ordering from above, only that after randomisation, the labels are also randomised (can be any of the 12 possible actions on the CAD60 dataset) and the dataset is ran through to find all the actions with that label. These is done sequentially and results in a dataset in which all the actions are shuffled and the subjects are shuffled, but the actions have the examples of all 3 participants (for the training set) condensed. For the validation set this procedure only reshuffles the set. This idea was born from inspiration in Curriculum Learning concepts [13, 40].

4.2.3 Preconditioning: Achieving Translational, Mirror, Scale and Rotational Invariance

Data preconditioning was implemented in order to achieve invariance in regards to translation - as an action should be classified the same way, no matter what displacement it has in regards to the origin. As in my last implementation of a growing when required neural gas classifier [86], translation invariance was obtained by centring the skeleton around the hips, that is, the average between left and right hip joints was taken and subtracted from all other joints removing any displacement

the skeleton pose may have accumulated.

Mirroring

Additionally, to correctly classify actions done with the non-dominant hand, the data was doubled with an inverted horizontal axis version of itself, so as to achieve the correct classification of chiral skeleton action sequences (mirror symmetry). Initially this procedure was done as a pre-conditioning step, however testing showed that a more generic option was to modify the distance metric and correct for this effect there (see section 4.2.7).

Normalisation

A normalisation procedure was undertaken so as to make the set of all the skeletons joints have each joint a variance equal to one. Even though this procedure was effective for my presented classifier, some problems arise when I try to use this approach for smaller sets without much variation - as it would happen in a real-time implementation of such classifier. A method simpler to implement in a real-time scenario was proposed for achieving scale invariance, based on the fact that during each action sequence limb sizes are constant - even though measurement from Kinect for such lengths might vary due to noise (see Fig. 4.2). The present normalisation procedure then calculates the sum of all limb lengths, that is, the summation of the norm of all connected joint points during the whole action sequence and then calculates the median of that measure, making it a normalisation factor for all joint points, so that the end total limb length would equal one.

Rotation Invariance

Lastly a procedure to rotate the skeleton to a canonical position - facing the sensor - was implemented. My first naive approach was to rotate the skeleton so as to make sure that the hips would face the sensor, that is, let $O(k)_{Hip}$ be the old hip position for a pose $p(k)$ on an instant k; $J(k)_{LeftHip}$ and $J(k)_{RightHip}$ be 3D (x,y,z) vector coordinates of the positions of the left and right hip joints respectively, components from $p(k)$. I want to find a 3x3 instant rotation matrix R(k) that ensures that the

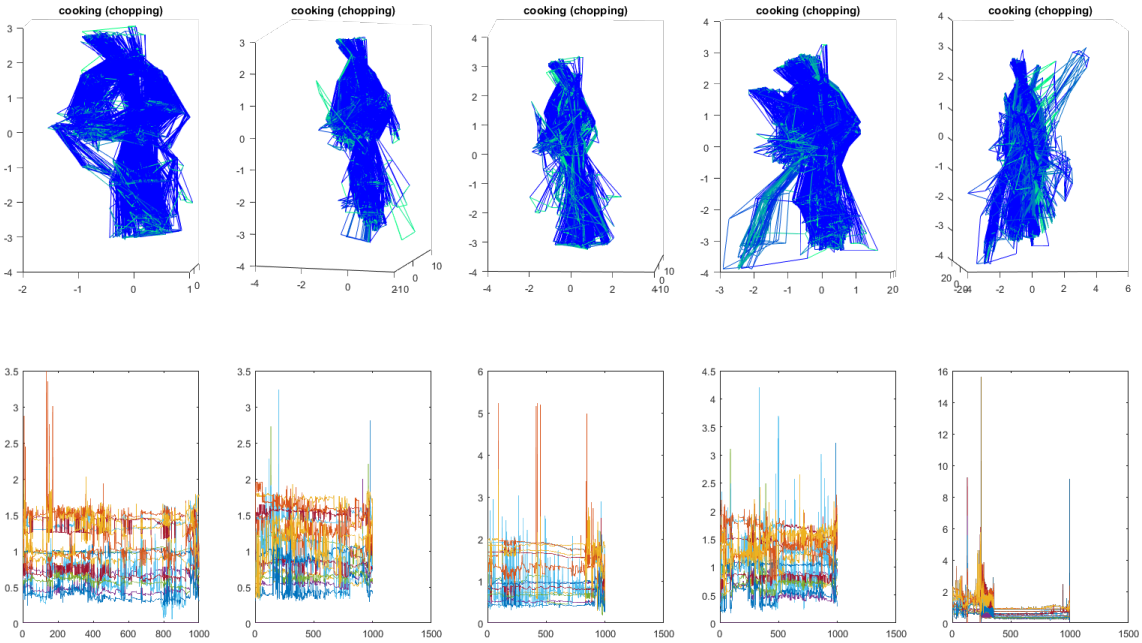


Figure 4.2: An example of the normalisation for cooking(chopping) action. On top 5 action sequence skeleton plots, coded from blue to teal (winter colormap); on bottom the limb lengths (y axis) by frame (x axis). Note that limb lengths are noisy, but rather stable for same subject (4 first plots, from left to right). When there is considerable noise, limb lengths are no longer reliable (rightmost sequence).

new joint hip vector N_{hip} has only components along the x axis ($N_{hip}^T = [1, 0, 0]$), that is:

$$R(k)O(k)_{Hip} = cN_{hip} \tag{4.1}$$

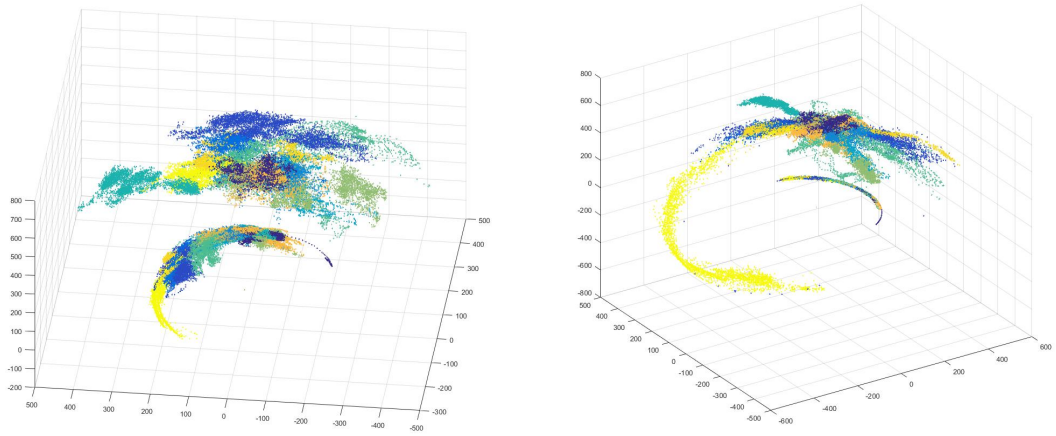
Where c is the magnitude of the hip vector and:

$$O(k)_{Hip} = J(k)_{LeftHip} - J(k)_{RightHip} \text{ and } N_{hip} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{4.2}$$

I can remove c by normalising:

$$R(k)\frac{J(k)_{LeftHip} - J(k)_{RightHip}}{\|J(k)_{LeftHip} - J(k)_{RightHip}\|} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{4.3}$$

This problem can be seen as optimisation problem and solved numerically, however since this procedure needs to be performed many times (once per skeleton in the



(a) (b)

Figure 4.3: (a): Colour coded by activity, the outer shell represents head joint positions vector in 3D space and the inner smaller shell the $J_{LeftHip} - J_{RightHip}$ hip vector. (b): After the correction 2 effects can be seen: first that the head positions tend to clump together, making the dataset harder to classify; second, the rotation procedure alters skeleton rotation in undesired ways, which can be seen by the incorrect orientation of the head on one of the actions from the set (yellow cluster).

action sequence), I chose to use a more computationally efficient algorithm [115]. This procedure however was proven to be inadequate to correct rotations as I can see from Fig. 4.3 as it rotates for the correction around the vector $O(k) \times N_{hip}$.

A seemingly more adequate procedure as to how to optimally rotate a skeleton as to match a specific was seen to be similar to the the orthogonal Procrustes problem [146] or Wahba's problem [107] and as such I decided to implement the Kabsch algorithm [77]. This algorithm is familiar to roboticists, as a more complete version of 3D point matching, the Iterative Closest Point(ICP) [27] is often used to piece together point-cloud data to generate map terrains. Wahba's algorithm is sufficient for my needs, as I have one-to-one correspondence of points (provided by my skeleton definition).

First correlation matrices of an k-th skeleton of my set, described by $p(k)$ was calculated against a sample template most common standing skeleton M (a skeleton from the dataset that has the smallest sum of distances when compared to all others), then a cross-covariance matrix H was calculated between that template and the pose

$p(k)$ as:

$$H(k) = M^T p(k) \tag{4.4}$$

And calculate the Singular Value Decomposition (SVD) of H, such as:

$$H = U(k)S(k)V(k)^T \tag{4.5}$$

and define the the rotation matrix $R(k)$ to be:

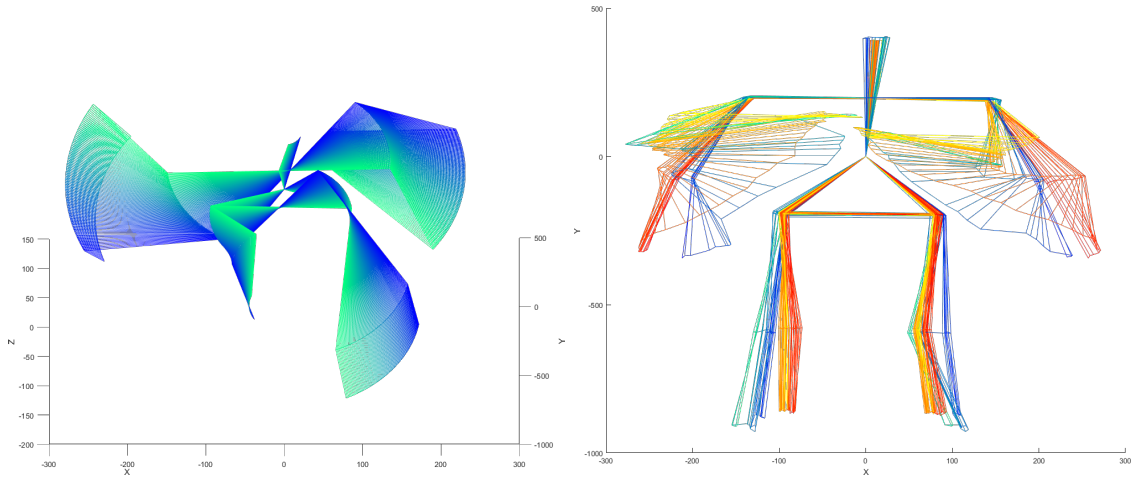
$$R(k) = V(k)U(k)^T \tag{4.6}$$

Note that I deliberately chose not to correct rotations matrices with negative determinants, an improper rotation[6], as this rotation and mirroring is also desirable, as for my problem, an action performed in a mirrored fashion is also supposed to be the same action. Lastly from this rotation matrix, I calculated the Euler angles and selected only the yaw angle to generate a new rotation matrix that would multiply every joint point and move the k-th to be facing the sensor, but with pitch and roll angles preserved (see Fig. 4.4).

4.2.4 Training the Neural Gas

In order to minimise comparison between many poses and to achieve a filtration of skeleton poses, each pose (or concatenation of poses) is matched to prototypical poses that are constructed by the Growing when required Neural Gas (GWR) (see section 3.2.7 for more details). In my implementation, the relationships between neighbouring points are not utilized and the neural gas can be seen as a compression algorithm or a filter of sorts that either (a) averages similar poses (or concatenation of poses), if they are similar enough, given a distance metric and a threshold parameter, or if the pose is considerably different from all the other poses (or concatenation of

[6]An improper rotation is a rotation that has rotations on the three axes combined with a flipping of one of these axes (mirroring operation), giving it a negative determinant [116].



(a) (b)

Figure 4.4: (a): An example of rotation correction for a single skeleton (not an action sequence). The skeleton starts with quite a deal of rotation along the Z axis (yaw, seen in blue) and is corrected until its final zero yaw rotation (teal). The process is divided into 100 equal steps to show better the operation; in the algorithm only the last skeleton (lightest teal) is used. (b): A fully corrected sequence (opening pill container in blue-teal) with a correctly matched template (red-yellow).

poses, which are points in a hyperspace or nodes of the neural gas) (b) adds a new point (or node) to my ensemble of points (the neural gas). By averaging points that are similar enough (pass the threshold), the GWR performs compression, not only reducing the dataset with which a naive k-nearest neighbours would need to go through (as a condensed nearest neighbour would do [66]), but also implements filtering.

Some small alterations to the structure of the GWR were implemented, namely to either optimise speed or make it more appropriate to solving my particular problem, namely:

- time invariant: although the GWR can deal with time sequences, I used the algorithm in its static version, using a sliding window scheme to deal with time changes in my data (see 3.2.9)
- the number of nodes limit: as it is described by [109], the GWR doesn't stop growing until the points being presented to the gas approach the limit imposed by the alpha activation threshold. As in the seminal paper [121] implementation, I impose a restriction to this growth by stopping adding points as the gas reaches that limit. The end result is having an effective mean activation

on the whole gas that is smaller than I would expect when the gas reaches its learned state, but with the advantages of being able to pre-allocate matrices and gain time during training. A further improvement here to reduce the number of epochs a gas needs to run is increase the alpha progressively until its endpoint, so as to have better unit representation with less total epochs. This was implemented with defined alpha increments $\alpha_i = \alpha_i + 1$, which increase a_t progressively from $a_{t0} = 0$ based on the rule $a_t = 1 - e^{-\alpha_i}$ until a threshold defined to be our old a_t. This increased performance of the algorithm enormously while guaranteeing more random point insertion and unit representation, lowering the necessity of running the algorithm for many epochs.

- uncommon sample removal: the noise from the skeleton tracking data behaves in a complex manner with joint tracking failure - for one or multiple joints - producing sometimes very unexpected skeleton positions. As we expect the sensor to work properly most of the time, a simple algorithm based on the fact that we expect for errors to be few[7] and present low activation from the gas, enables removal of points based on the activation value provided by the gas and a measure of mean activation and its standard deviation. These values are calculated for the whole trained set after each epoch and the points are then compared to this mean subtracted from a γ parameter times the standard deviation, that is, if the point is gamma times the standard deviation too off from the expected value it will be cut off from further calculations and not be a part of the set. This was implemented in the Parisi et al. paper [121] and set to 4 as we are aiming here to remove long tails, were a small number of values have a large error.

4.2.5 Gas Chaining

To detect time changes with the classifier, multiple gases were chained together, composing higher hierarchical structures of the matched data. More specifically,

[7] In unexpected body positions or in tricky situations for the depth sensor (e. g. where there is not a lot of depth difference between the subject and the background), this assumption does not hold.

across different layers, sequences of skeleton poses were concatenated together using not the initial skeleton poses, but their best matched nodes from the gas of the previous layer. This was done using a sliding window scheme for both velocities and poses. For a more detailed explanation about how the sliding window was implemented I refer to Sec. 3.2.9 .

Parameters

Differential parameters were used in each layer as changing the amount to dimensions of the neural gas (a concatenation of 3 skeleton poses increases dimensionality by a factor of 3) will affect the measured distance threshold. Moreover the velocities layer has quite different dimensions from the start. The parameters used for each layer[8] (for the CAD60 dataset unless otherwise noted) were $a_1 = 0.99983; a_2 = 0.99999; a_3 = 0.99995; a_4 = 0.999998; a_5 = 0.9999$. These were chosen based on visual inspection of plot data and inclusion behaviour to suit my normalised skeletons.

4.2.6 Gas Distance Kernels

Simple Kernel

The simplest kernel I can use to calculate distance is a euclidean metric in which each of the joint 3d coordinates is concatenated into a long vector and distance is the euclidean norm from this point in a higher dimensional space and all other nodes from the gas. That is, for a pose matrix p(k) of a k time-step represented as:

$$p(k) = \begin{bmatrix} j_{1x}(k) & j_{1y}(k) & j_{1z}(k) \\ j_{2x}(k) & j_{2y}(k) & j_{2z}(k) \\ \dots & & \\ j_{Jx}(k) & j_{Jy}(k) & j_{Jz}(k) \end{bmatrix} \tag{4.7}$$

[8]Note that here those thresholds are limit thresholds that the algorithm gets to eventually using our sped up implementation based on alpha increments.

I have it's vector representation $\xi(k)$:

$$\xi(k) = \Big[[j_{1x}(k)j_{2x}(k)\dots j_{Jx}(k)][j_{1y}(k)j_{2y}(k)\dots j_{Jy}(k)][j_{1z}(k)j_{2z}(k)\dots j_{Jz}(k)]\Big] \tag{4.8}$$

It's distance from a node n^i that compose the gas matrix A, will be:

$$d(n^i, \xi(k)) = \sqrt{\sum_{j=1}^{3J}(n_j^i - \xi(k)_j)^2} \tag{4.9}$$

Sum of Euclidean Distances

A perhaps more intuitive distance measure would be to consider the distance difference from each joint and add all of those distances to obtain a similarity metric. As such, the total distance is calculated as the sum of the individual joint to joint distances between the gas node and the current presented skeleton pose or concatenation of poses. In this case the I represent the node n^i in its matrix representation with the respective x, y and z coordinates and I have the distance as:

$$d(n^i, p(k)) = \sum_{j=1}^{J}\sqrt{(n_{jx}^i - j(k)_{jx})^2 + (n_{jy}^i - j(k)_{jy})^2 + (n_{jz}^i - j(k)_{jz})^2} \tag{4.10}$$

4.2.7 Gas Distance Conditioning

At first glance, this step seems to be the same as the procedure in section 4.2.3, with the main difference in being whether the data itself was changed or not. This allows for more than one distance to be tested at a time and to choose the smallest one as opposed to changing the data in the dataset itself and assuming that transformation was a rightful correction. This was necessary since Mirroring and Rotation Correction were implemented on visual inspection alone and unfortunately I could not guarantee these procedures will necessarily create a better matching skeleton. However, keeping a normal dataset and applying corrections on each skeleton as needed was a much more time consuming procedure.

Mirroring

The adequate procedure to mirror a skeleton involves mirroring it around its axis of symmetry, that is, a simple inversion of the x, y or z axis is not sufficient as they also affect its rotation. To preserve rotation and just flip left-right, the procedure used was to rotate the skeleton so as it would have their hips parallel to x axis with a rotation matrix R, then invert x coordinates, change the positions of left joints (for a version 1 Kinect joint enumeration that is J4: left shoulder, J5: left elbow, J12: left left hand, J8: left hip, J9: left knee and J:14 left foot) for the respective position of right joints (J6: right shoulder, J7: right elbow, J13: right hand, J10 right hip, J11:right knee and J15: right foot), then apply the inverse rotation R^T (as the inverse of a rotation matrix is also its transpose) to recover a skeleton with the same orientation as the original one, only with left and right handedness flipped. The advantage of using this procedure when calculating a distance function, as opposed to use it to just double the training set, is that it allows for me to keep the number of units in the gas a stable number (as I would 2 different templates to match either correct skeleton if it was used as a preconditioning).

This procedure was not very computationally expensive as the simple rotation procedure (not Wahbas' algorithm) from subsection 4.2.3 could be used, as it is enough that the axis of symmetry coincides with the plane that is being mirrored, the matrix rotation inversion is trivial and the same for all points in the skeleton pose $p(k)$ and it did not increase the overall running time, as doing it while the gas is comparing points and check 2 values (either normal or mirrored).

Rotation Correction

It uses Wahba's algorithm to calculate the best possible rotation that minimises distance from the current presented skeleton pose and the gas node, updating the current pose to the canonically rotated one. As this rotation correction procedure is done to each node from the gas, although inherently parallel, this procedure in my current implementation is very time consuming. I intentionally did not correct the negative determinant case of Wahba's algorithm (that can produce mirrored

images), so that it could produce on demand fix for mirrored images (in case of left-handed action), but as this procedure is not precise for unequal skeleton poses, this may also be a source of error. Additionally it may also be used together with mirroring function, e. g., for testing purposes.

4.2.8 Labelling

Labelling is done by identifying the "nearest" pose and assigning a label according to that prototype's; in a sense it is a k-nearest neighbour algorithm with k set to one. First, to each node from the resulting gas (representing a pose or concatenation of poses), a label is applied from the training dataset using the methodologies described below. Then to classify data, using some metric I compare distances from each node to each data point and find for each pose from the dataset, the node to which it most closely matches. The classifier node receives then this label.

Node Labelling - Best Matching Pose

I label the nodes from the gas finding the pose from the training dataset that has the minimal distance to the current node and assign its label to the gas node. This is the strategy used by the Parisi's paper [121]. From this I construct a zero vector, whose length is the number of possible labels, with a one on the corresponding dimension to the matching label.

Node Labelling - Effective Node Match[9]

Effective node matching can be seen as a backwards matching of sorts. It is inspired on the idea that some poses are common and not specific to a certain pose (shared positions), while others only occur in a certain action. Using the best matching pose will not account for this. In this way, the whole dataset is matched with their best matching node points - that is, the node with which they have a smallest distance

[9]Note that these implementations make most sense when trying to classify actions as a sequence of skeletons, with the correct label being the weighted sum of the classifier's response over time. For individual frames, for the classifiers attempted there is virtually no change in results. For this reason, the "Combined Effective node match and best matching pose" combined with a argmax was used, instead of the original "Best matching pose", as it allowed for possibly more accurate time filtering.

according to the chosen metric - and count for each node point, to which action labels it was mapped. This is then normalized[10]; it is divided by the sum of matched points from the dataset, so that the sum of vector components is one. The node label will now instead of a single number, be a vector with the number of dimensions equal to the number of possible labels, with fractional values, corresponding to how specifically that vector will match to each action.

Node Labelling - Combined Effective Node Match and Best Matching Pose[9]

As the effective node matching can produce nodes that are not matched to any element in the training set, it is necessary an additional step to ensure that every point will have at least a single match. This is achieved by first labelling nodes with their best matching pose and counting that as a first match for the effective node match - although this is arbitrary and there is no reason to believe a priori that counting the best matching pose as a single effective node match is adequate.

4.3 Results

Currently best results from the GWR neural gas are around 70% (see Tab. 4.2) within the CAD60, with simple euclidean gas metric kernel, 1000 nodes, 10 epochs, without normalisation, mirror or rotation correction, using only position - no temporal concatenation - best matching pose labelling, without any gas distance conditioning under a leave-one-subject-out for all subjects cross validation scheme. Results can be seen discriminated per action type in Tab. 4.5.

Although described in methods (see section 4.2) are many more options for the classifier configuration, not all possible combinations were tested and only some options that seemed of interest and worthy of discussion will be presented here in more detail. The high variability in results from the gas depending on the number

[10]Here perhaps a better option would have been to use a softmax function. However since back-propagation was not attempted, I considered the exponential superfluous and used a simple normalization instead. Moreover, the exponentiation would bring the result closer to argmax which was the exact issue this function was trying to mitigate.

of epochs used and order of presentation of skeletons required many runs to detect small improvements and testing all possible permutations is not feasible with its current implementation[11]. Some of those address similar issues with only slight improvements with negative effect on performance (like pre-condition mirroring vs. distance metric mirroring) and a detailed breakdown of all its effects would be a rather lengthy enterprise. These features however, possibly have their use in an algorithm fusion approach, as they are seen to capture different information about the set [117], which may contribute to higher accuracy.

The best results were achieved with Random organised ordering (where each action is learned with all data from that action being presented at the same time), which was used for presenting the following results. In general, it presented similar results to a sequential ordering and around 0.5% superior on neural gases (however results were variable due to random nature of NGs) than testing with truly random order. However, as I theoretically see this presentation as more general than presenting always the same sequence, Random Organised Ordering - our simplified curriculum learning attempt - was thus used.

The calculation of measures was done following the recommendations of Sokolova's paper [159] and as such we report global accuracy (A_{Global}) as it is a robust measure for the overall performance of an algorithm and has low sensitivity to extreme performances on a specific class [159]. One may see the confusion matrices (see Fig. 4.5 and A.1) from the classifier for all actions combined together, as the left side showing the labelling done on training data, and right below the same results discriminated by subject, that is, from subjects 1 through 4 (Fig. A.1), also with the left displaying training data and the right validation data.

If $C_{n\times n}$ be the confusion matrix of the multiple class classifier with n classes, I can define a global accuracy A_{Global} as:

$$A_{Global} = \frac{\sum_{i=1}^{n} C_{ii}}{\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}} \tag{4.11}$$

[11]Each run with 10 epochs with 5 gases with 1000 nodes would require roughly 3.5 hours, which was unfortunately very variable depending on the preconditioning options and distance function used.

Table 4.2: Overall global accuracy (in %) for all actions and all subjects on CAD60

	SVM	KNN	GNG	GWR
No preconditioning	55.43%	53.76%	46.44%	47.27%
Centring and mirroring	66.36%	83.02%	75.97%	75.2%
Centring, mirroring and normalizing	67.27%	82.48%	78.37%	79%

The first thing that may be observed from such matrices is the discrepancy in the global accuracy from the results in the training and validation data, where most of the training data is correctly classified (accuracy around 90% - results not shown), whilst the accuracy on validation data is around 70%, showing a great deal of overfitting and not correct generalisation of the current presented classifier structure.

Something else that may be apparent is that for instance for subject 2 and 4 (Fig. A.1c and A.1g), not even the training data is correctly classified as for instance for subject 2 in actions 4 and 6 (corresponding to actions 'drinking water' and 'relaxing on couch' respectively) or in subject 4 on action 6.

4.3.1 Comparisons Between Algorithms

The classifiers were run and compared in a "by scene" manner, in which the available actions to be classified were not all the 12 different actions, but given a particular scene, say bathroom or bedroom, a smaller set containing between 3 and 4 classes of actions was examined.

The results from each of the 4 different methods, namely SVM[12], KNN, GNG and GWR (with simple preconditioning) and the effects of varying that preconditioning can be seen in table 4.2. This was done to separate the effects of the pre-processing stage from the learning algorithms efficacy. For a simpler comparison on which was the best method of all, the overall global accuracies were used, here defined as the sum of the traces of the confusion matrices of all scenes and all subjects divided by the total number of poses (see eq. 4.11). As the method KNN has shown to be the more accurate one, details of its accuracy by activity type can be seen in

[12]Support Vector Machine (SVM) with linear kernel trained for multiple classes using Error-Correcting Output Codes (ECOC) [103].

Table 4.3: Precision and recall of the best-found algorithm (1NN, centred and mirrored skeletons) in the different environments of CAD-60 for all subjects combined.

		"New-person"	
Location	Activity	Precision	Recall
Bathroom	Brushing teeth	96.46%	93.11%
	Rinsing mouth	87.68%	100.00%
	Wearing contact lens	99.11%	88.82%
	Average	94.42%	93.98%
Bedroom	Talking on phone	63.12%	60.44%
	Drinking water	55.47%	65.81%
	Opening pill container	98.83%	77.85%
	Average	72.47%	68.03%
Kitchen	Cooking-chopping	96.86%	75.97%
	Cooking-stirring	56.35%	81.43%
	Drinking water	73.41%	74.46%
	Opening pill container	98.83%	81.06%
	Average	81.36%	78.23%
Living room	Talking on phone	63.12%	60.44%
	Drinking water	81.28%	79.30%
	Talking on couch	100.00%	99.42%
	Relaxing on couch	100.00%	100.00%
	Average	86.10%	84.79%
Office	Talking on phone	63.12%	60.44%
	Writing on whiteboard	89.64%	100.00%
	Drinking water	82.06%	76.45%
	Working on computer	100.00%	100.00%
	Average	83.71%	84.22%
	Global average	94.42%	81.95%

table 4.3[13].

Regarding the optimal value for k on the KNN classifier: this value seemed to vary considerably. Depending on how many classes there were to be chosen from, its optimal value ranged from around 320 (when all tasks are possible classes) to the optimal value 8 (when there are only 3-4 possible tasks using only centring and mirroring preconditioning) and to the optimal $k = 2$ (when centring, mirroring and normalizing was used). As the difference between the maximum value achieved and

[13]Initially I had reported the variations of results upon a cross-validation strategy chosen of Leave-One-Subject-Out for All Subjects, however under a deeper analysis, those were removed, as their behaviour is still a debatable and dependant on the specific dataset [12, 90, 192]. The original table for reference is still present for conference in the appendix (see Table A.5).

the value for $k = 1$ was usually around 0.5% or less (83.65% vs. 83.02% and 82.91% vs 82.48% respectively) in those particular cases, I chose to report the outcomes with $k = 1$, which seem to yield a simpler classifier with accuracy very close to the optimal.

The table 4.2 shows clearly that disregarding the translation of the skeleton pose in regard to the origin is advantageous in every method, as well as indicate that the idea to normalise the skeleton based on neck to torso distance is in most cases helpful - however not in the best performing method only by a small margin (83.02% against 82.48%). The interesting results when analyzed by scene, where clearly in some environments as "bedroom" I have a lower than expected precision and recall, were not only occurring in the KNN approach, but in both the gas implementations as well as in the SVM.

I consider negative results of also being of interest in this context, as one considers options so as to improve current classifiers. The tests were done in a greedy manner, that is, each alteration was tested on a base GWR implementation with 1000 nodes and running for 10 epochs, for consistency, with only one of the tested properties changed.

Preconditioning So as to achieve translation invariance, I centre the skeletons around the hips, however, I wondered if that was the optimal place. I also tested centring around the shoulders, that is, considering the middle point between shoulders as a zero. Another alternative was to consider the middle point between the hips, but not remove the height, that is, centring around the projection of the hips on the ground. Of those, centring around the hips seems to yield the best results.

Perhaps of most interest in this section is the influence of removing rotations. This task has proven to be slightly more complex than anticipated, mostly due to the fact that Wahba's algorithm expects an object to be known a priori and only present alterations in its rotation. This is not the case for skeletons as the configuration of the skeleton is not know, therefore I have no good skeleton to match it to and detect

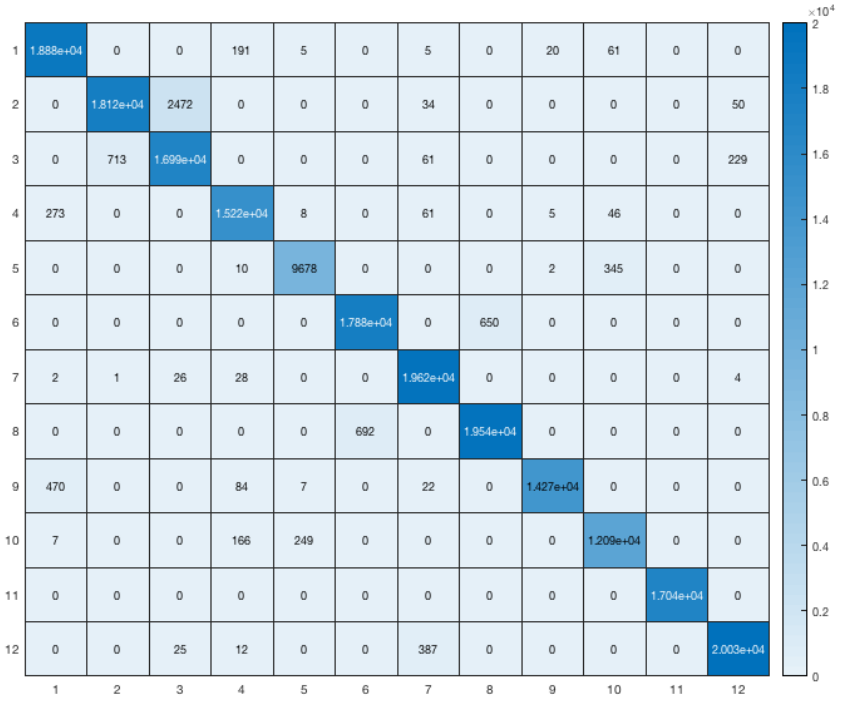
Table 4.4: Results for the CAD60 dataset for all subjects leave-one-out with a chained GWR classifier (results for last layer) with 600 nodes with N = 4 trials

	% Overall result for all actions
$\text{Precision}_{\text{M}}$	69.45%
Recall_{M}	70.85%
F1_{M}	70.15%

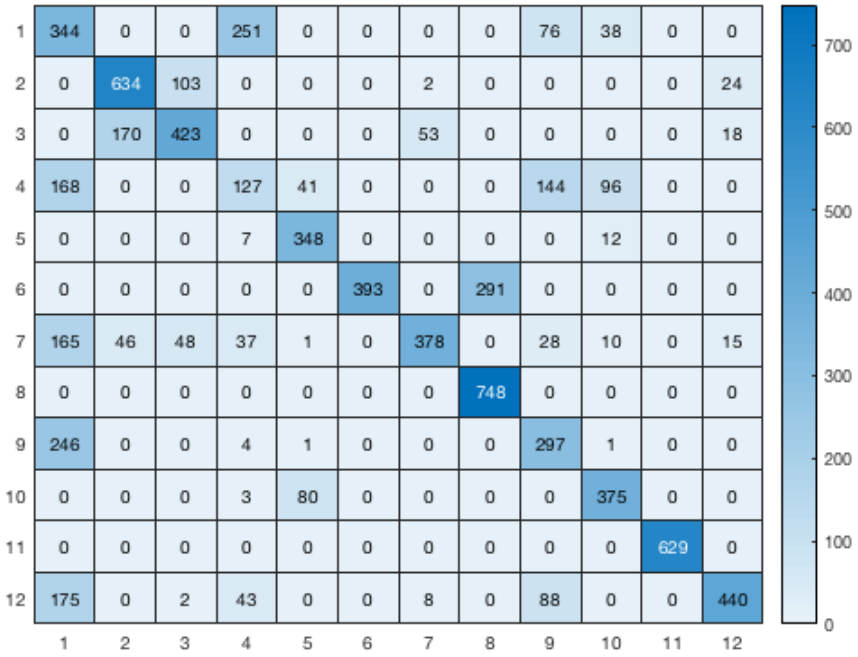
rotations. The initial approach tried to solve this first by matching it against "the most common skeleton", that is, the skeleton that had the smallest distance to all others. This works reasonably well if the skeletons being matched are quite alike, but if the template and the tested skeleton are on very different configurations, the rotation correction obtained was often very poor. Moreover, it had the bad property of putting two similar amongst themselves, but very different from the template into very different configurations. An attempt to fix this was to test among all gas nodes, i. e., prototypes, but this idea was abandoned due to high computational costs.

Moreover, the rotation angles themselves were shown to have some information about the class membership in this dataset (as can be seen in head positions from Fig. 4.3). One may argue that skeleton pose orientation does not correlate with the actions being performed in the general case and only shows a quirk of my dataset and how it was acquired (the Kinect has requirements in terms of occlusions) and should not be considered at all, but this small dataset does not allow for such an assumption to be tested.

Distance Functions Euclidean metric was considered superior on most of my tests among those tested (tested also Malahanobis, sum of euclidean distances, conversions to polar and spherical representations and use of Euclidean distances on those), with other alternatives performing worse or equal (but with increase in computation times), but with loss in performance.



(a)



(b)

Figure 4.5: Confusion matrix for the combined data for subjects 1,2,3 and 4. Figure (a) shows training data and (b) validation data. Indices 1-12 correspond to the actions on the dataset: 1- 'brushing teeth' , 2- 'cooking (chopping)', 3- 'cooking (stirring)', 4- 'drinking water', 5- 'opening pill container' , 6- 'relaxing on couch', 7- 'rinsing mouth with water', 8- 'talking on couch', 9- 'talking on the phone', 10- 'wearing contact lenses' , 11- 'working on computer', 12- 'writing on whiteboard'.

Table 4.5: Classification of each action sequence in a leave one subject out validation type with optimal parameters for (1000 nodes) GWR without filtering

Activity index	Activity label	Precision	Recall
1	'brushing teeth'	69.87%	39.15%
2	'cooking (chopping)'	79.60%	76.71%
3	'cooking (stirring)'	69.22%	64.84%
4	'drinking water'	20.50%	33.82%
5	'opening pill container'	92.41%	87.91%
6	'relaxing on couch'	60.44%	68.43%
7	'rinsing mouth with water'	55.08%	82.64%
8	'talking on couch'	74.47%	67.27%
9	'talking on the phone'	52.21%	65.91%
10	'wearing contact lenses'	95.01%	69.00%
11	'working on computer'	100.00%	100.00%
12	'writing on whiteboard'	64.62%	94.53%

4.3.2 Across real world application: Results on the dataset Ourset

So as to test the capacity of my implemented approach to generalise on data that would be similar on that captured by a robot under optimal conditions, a dataset was constructed and tested on the trained models using CAD60 skeletons. The acquisition of this data was performed under controlled conditions to mimic as close as possible data that would be acquired real application.

The results were not encouraging and thus a wider range of different approaches were attempted and are reported in the confusion table 4.6, namely a single hidden layer neural network with either 10 (10-NN) or 100 (100-NN) hidden units trained with the scaled conjugate gradient backpropagation[14] , the default configuration for a neural network model from Matlab's nntool. Note that since the data collected from Ourset dataset has a slightly different skeleton definition (20 joints from Ourset, obtained with a Kinect for Windows sensor vs. 15 joints from CAD60, obtained from a v1 Kinect), with different numerical representation (units of length differ), the normalisation step was used for all tests, as it was necessary. The rotation correction chosen used was the single rotation correction on distance function with

[14]This optimizer was used as it is the default in Matlab. Other more modern methods, such as ADAM (Adaptive Moment Estimation) were only introduced in newer versions, namely R2018a.

Table 4.6: Overall global accuracy (in %) for all actions and all subjects on Ourset

	10-NN	100-NN	SVM	KNN	GNG	GWR
Centring, mirroring and normalising	19.1%	22.6%	26.2%	25.1%	32.6%	33.5%
Centring, mirroring, normalising and rotation correction	23.4%	20.5%	15.2%	25.1%	20.5%	18.8%

Wahba's algorithm (more time consuming version) as it was considered a superior choice.

4.4 Conclusion and Discussion

From the analysis I performed there seems to be conflicting results when comparing the classifiers used under the CAD60 dataset alone when compared with results from Ourset.

One may reason that Ourset was not constructed adequately, mostly because the sensor used was of a different kind and is thus not appropriate. Although this is perhaps a fair conclusion, it does not come naturally from visual inspection of the dataset. As visual inspection of the skeletons shows, the differences from the model are not obvious and may be only related to different body proportions (longer/shorter limbs, head, feet hands, wider/narrower hips and shoulders) and body poses. An algorithm that aims to classify skeleton poses should deal gracefully with such differences. My understanding is that the dataset Ourset is a challenging, but still valid test for an activity detection algorithm.

Another point that needs to be addressed is the small size of these datasets. This is an important issue, but due to the difficulty in getting adequate full body skeletons (no missing data) without an active system (active vision), this is an issue that will affect all work done in this way. The small size of Ourset (N = 2) is not considered an issue, since it was used only for validation of the results and not for training. This can be an issue, since having more data from the Kinect for Windows sensor would allow us to test the extent to which the sensor and set-up affects the results. The interpretation we have is that these 31 actions with 37k frames is enough to evaluate performance.

I most likely need to run additional tests using other and larger datasets with additional types of action, but as it appears, even though the simple KNN algorithm with just centring the skeletons gives the best accuracy, it seems to fail to generalise with that representation.

A detail about Random organised ordering to be evaluated is the tendency of the NG algorithms to fuse points that may represent different actions, which I could mitigate if a not truly random order was presented to the gas. This, combined with the random nature of NGs, made trying different configuration of the NG architectures, very time consuming as I needed to test the NGs algorithms under many iterations to make sure the effects observed were significant. [15]

I should also mention the reasons why I believe that solving rotation issues seemed to do more damage than good in my classifiers. It may be that I am exploiting a characteristic of the set, that for a particular action, only a certain type of angles would give you a complete skeleton with no occlusions and as such the act of acquiring a dataset with properly defined skeletons creates an inherent bias that will not be present if data is capture in "wild" conditions. If this is the case, a real working classifier needs in fact to do rotation correction, such as the Wahba's algorithm and I would be in fact less close than first assumed in coming up with a classifier structure that can really generalise. An opposing idea would be to assume that I need in fact a person to be in a particular angle to understand what they are doing, as sometimes it is even difficult for a human to understand what is happening when viewing a situation from a certain unexpected angle. [16]

An additional issue that must be pondered is the use of different skeleton representations from the learning dataset and the testing dataset, which is not ideal and could be responsible for a substantial amount of the mismatches encountered.

[15] This was motivated by our desire to get a better more shuffled gas unit representation without needing to increase number of epochs, which added up runtime, as it was mentioned in the results section from Chapter 3.

[16] In the literature, these are the so called mental rotation tasks [154, 172] and can be themselves quite challenging. The fact the people must think and have a hard time in rotating objects to different orientation are a slight indication to me that object appearances - and probably actions as well - are learned in a certain orientation and rotation is not performed automatically. Perhaps it is even possible to test this given that men and women typically display different levels of aptitude for these tasks.

These conflicting findings, point in the direction of using larger datasets with an algorithm that of which I have full control, not based on proprietary software such as Kinect's toolkit, so that these conflicts can be solved.

Additionally, perhaps combining skeletons with other strategies to limit the number of possible classes such as using a state machine that encodes knowledge about the task, or other information, such as the detection of objects that are being grasped or sound (talking on the phone and drinking water would be much more easily discriminated in that manner, as for with pose detection, considerations of where the hand is when the head is modelled as a single point, can be tricky for this approach) and this multi-modal information could be integrated in an additional layer to correct inaccuracies inherent to only performing HAR with pose detection.

An additional important observation about this task, based on this work, was concerning the difficulty testing multiple similar conditions and the recognition that this task is, as Herlihy et al. puts it [68], "embarrassingly" parallel[17], that is, I can test different architectures of classifiers very easily if I have more CPUs. It became clear to me that to solve this sort of problem, I needed a system that would allow me to test many different scenarios simultaneously, with plenty of data, and possibly recognise that for slightly different actions, or context, perhaps one or another type of classifier is superior and somehow choose the most appropriate one, or combine their outputs somehow (see section 6.8).

These realisations motivated me into implementing my docker based system (see section 2.2), to hopefully implement systematic training of multiple architectures (not implemented due to time constraints) and the extension of my investigations into deep learning, so as to be able to cope with larger datasets and faster training.

[17] An "embarrassingly" parallel problem is one that can be easily divided into parallel processing units and those can be executed in parallel with no crosstalk between processes.

Chapter 5

Deep Learning Implementation

5.1 Introduction

In this Chapter, I explore a deep learning implementation of action recognition, namely the one from the paper "Temporal Segment Networks: Towards Good Practices for Deep Action Recognition" by Wang et al. [178], which I altered to make it real-time[1], and my attempt to deploy such classifier in the robotic platform Scitos G5 using ROS. This particular implementation was chosen for its high accuracy on the HMDB51 dataset [97], which made it a common benchmark for other works to compare against and - perhaps most importantly - because the source code was openly available, which would aid in quick deployment.

5.2 Materials and Methods for Activity Detection using Video Sequences

5.2.1 Datasets Used

The datasets used for the following investigations were the HMDB51 [97] with a restricted version of it to include actions that can be performed indoors by a single person (often referred to as Human Motion recognition Database with 14 actions

[1]Real-time being interpreted in the sense of having a time constraint, as opposed to waiting for an event, in this case, for the action to end, to output a result.

(HMDB14)) and my own acquired data using the same 14 actions (often referred to as MYSET). One of the reasons for using this dataset was that it was already successfully classified with good performance with the TSN algorithm, which reduced the uncertainty about whether I can actually classify activity data well with this architecture and enables me to focus on other aspects of the implementation. The second main reason for this dataset to be used for most investigations was that the set of activities it contains is more indicative of actions that can be performed indoors, when compared to UCF101.

More details on the datasets used can be seen in subsections 5.2.1 and 5.2.1.

The image-only datasets do not contain depth information, are thus easier to gather, and with that present wider collection of data samples. I will present the dataset HMDB51 (see Section 5.2.1), the HMDB14 (see Section 5.2.1) and my own version of the HMDB14 actions, the Myset dataset (see Section 5.2.1).

Human Motion Database: HMDB51 The HMDB dataset was introduced by Kuehne et al. in the IEEE paper from 2011 "HMDB: A Large Video Database for Human Motion Recognition" [97]. It introduced a challenging annotated dataset of action videos based containing 6'766 video clips in 51 distinct actions, each action having at least 101 video clips, obtained from the internet. Moreover, it tackles some of the difficulties of classifying videos by employing video stabilisation preprocessing step (using the RANSAC algorithm).

Actions sequences were divided into 5 categories namely:

- General facial actions: smile, laugh, chew, talk;
- Facial actions with object manipulation: smoke, eat, drink;
- General body movement: cartwheel, clap hands, climb, climb stairs, dive, fall on the floor, backhand flip, handstand, jump, pull up, push up, run, sit down, sit up, somersault, stand up, turn, walk, wave;
- Body movements with object interaction: brush hair, catch, draw sword, dribble, golf, hit something, kick ball, pick, pour, push something, ride bike,

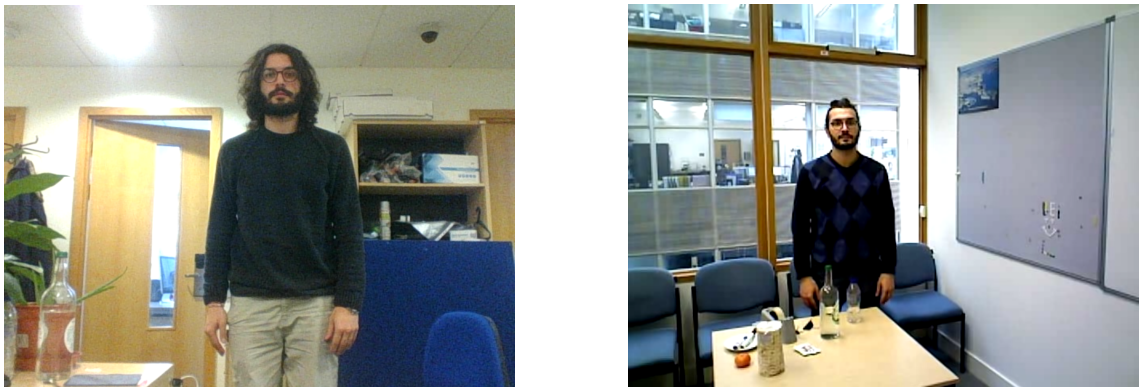
ride horse, shoot ball, shoot bow, shoot gun, swing baseball bat, sword exercise, throw;

- Body movements for human interaction: fencing, hug, kick someone, kiss, punch, shake hands, sword fight.

Note that these activities do describe many common actions that I might expect to be observed in a SAR use case, however many of those would be quite rare occurrences in a home environment, especially in an elderly house (such as cartwheel, golf, handstand, kick, draw sword or sword fight). This motivated me to use a subset of this dataset for my studies (see subsection 5.2.1).

Human Motion recognition Database with 14 actions (HMDB14) In short, the HMDB14 dataset is a sub-set of HMDB51 containing 14 actions that are performed indoors, with a single agent; an idea that shares some elements with the creation of the J-HMDB dataset [75], mostly on implementation idea, which was motivated by the facts that most action recognition algorithms fail when presented with real world data. As in the J-HMDB dataset, in which a subset was chosen from HMDB51, I chose to limit the set, but not limit the dataset to the same actions and also, since I am not currently using Joint information, I did not particularly care for videos that shown full body poses. Moreover, J-HMDB included outdoor activities such as golf, kick ball, pull-up, shoot ball, shoot bow, shoot gun, swing baseball and throw - actions I cannot reasonable expect to happen often in an indoors setting of a robot being used for elderly care - and removed some actions which seemed important to me, such as chew, drink, eat, pick, smile, stand and talk - actions that seem to be important to describe the behaviour of the elderly in their homes - and which I expect a CNN based network to be able to classify; a fact that also I aimed to check.

For those reasons I chose a different subset of 14 actions, namely: 'brush hair', 'chew', 'clap', 'drink', 'eat', 'jump', 'pick', 'pour', 'sit', 'smile', 'stand', 'talk', 'walk' and 'wave'. However, unlike the J-HMDB dataset, no additional joint information or annotation of puppet masks as was done as those were not planned to be needed



(a) (b)

Figure 5.1: A picture of the end of a standing up action both in Ph.D. common office lab (a) and empty office (b). Both actions were shot while no other person was in close proximity.

for the architecture used.

Own Acquired Data: the MYSET Dataset The MYSET dataset was acquired using Scitos G5's Xtion head mounted sensor, recorded with 640x480 on both RGB and Depth channels (although depth was not used for the work presented in this section). So as to emulate different recording styles, videos were recorded with both a moving pan-tilt unit (active vision) and static camera, using both close-ups and distant subjects. So as to simplify data collection (and data analysis), videos were collected of only one subject, with a fixed duration of either 4 or 5 seconds in 2 different rooms (this PhD's office lab and an empty office - see Fig. 5.1).

Videos were recorded from compressed video streams (so some compression artefacts are present). In order to make sure the dataset was classifiable, videos with too many recording artefacts, as well as videos that were improperly labelled, or were improperly executed were manually removed from the set. Data acquisition was performed 14 times for all actions, 11 times in the PhD office lab and three times in the empty office. The total playback duration of the acquired set is 8 minutes and 7 seconds.

Note that, unlike the HMDB51, the video streams acquired were not stabilised. I assume this to be not of major importance, due many of the MYSET videos being recorded with a static camera and the simple movement of the active vision

Action	Office Lab	Empty Office	Total Included	Removed
1. Brush Hair	8	2	10	4
2. Chew	9	2	11	3
3. Clap	7	2	9	5
4. Drink	6	2	8	6
5. Eat	3	1	4	10
6. Jump	6	2	8	6
7. Pick	6	2	8	6
8. Pour	5	2	7	7
9. Sit	6	3	9	5
10. Smile	8	1	9	5
11. Stand	4	1	5	9
12. Talk	7	2	9	5
13. Walk	3	3	6	8
14. Wave	8	2	10	4
Total	86	27	113	83

Table 5.1: Actions performed by action, by room and total included in the dataset. Under the label 'Removed' is the number of actions that were not considered well executed or had video artifacts and did not make it into MYSET.

node.

A summary of the dataset acquired can be seen in Tab. 5.1.

5.2.2 Artificial Neural Networks (ANNs) and Deep Convolutional Neural Networks (CNNs)

The underpinning ideas behind Artificial Neural Networks (ANNs) date back to 1940s and 1950s with the works of D.O. Hebb with the introduction of the notion of Hebbian Learning and Rosenblatt that coined the term perceptron [137] when trying to build a model of the human brain [140]. Slowly ideas were added to this simple model, such as the addition of multiple layers, so that a network could be fitted to learn more complex functions such as the XOR and proper training procedures with the proposal from Werbos in 1974 [182] and Rumelhart et al. in 1986 [141] of back-propagation training.

More recently great progress has been achieved in using ANNs to classify images by using deep neural networks, i. e., networks with a larger number of layers and efficient GPU based computing methods to train them [95]. This result may come as a bit of a surprise when I consider that a single hidden layer ANN with a limited finite

number of nodes using a sigmoid activation functions is capable of approximating any function on a compact subset of R^n or conversely, that the error of a decision function based on an ANN can be made arbitrarily small. This came to be known as the Universal Approximation Theorem [113].

One of the reasons for this more intricate structure is that, even though simple single hidden layer ANN are universal approximators, this result does not specify how one would go about finding the weights that would enable this architecture to fit a continuous function. A CNN[2] (see Eq. (5.1)) makes the process of training an artificial neural network much faster, by having a much smaller number of connections and parameters that need to be set for a network of similar size layer.

It achieves this functionality by sharing weights between many units that are close to each other [95]. This is done by employing multidimensional discrete convolution filter weights, instead of considering each individual data sample as an isolated entity.

Simply put, the convolution $(f * h)(x, y)$ of two 2D signals [26] $f(x, y)$ and $h(x, y)$ is defined by the equation:

$$(f * h)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i, j)h(m - i, n - j) \tag{5.1}$$

Where usually one of the signals is the image and the other signal is the kernel. In the case of image convolutions, the domain is bound to the size of the image (so not infinity) and we usually effect the transformation over three channels (Red, Blue and Green), so the expression is slightly altered. Another important realization about this process is that a convolution can be broken apart into chunks and is therefore an "embarrassingly" parallel problem. This feature enables convolutions to be performed extremely fast in graphics hardware.

[2]A proper explanation of what a 2D discrete convolution is and how it can be used to limit the number of learnable parameters in a neural network is much too broad and thus outside the scope of this text. For a good review of discrete convolutions we recommend reading [67]. Neocognitron [54] is probably the earliest work using a CNN and presents a good explanation of the use of convolutions for image analysis, which is the basis of the work that is presented here.

A simplified way of considering a CNN would thus be the problem of optimizing a set of 2D kernels so as to have such a set of kernel output different convolution results to images of different classes. Kernel values are usually initialized with white noise and numerical methods are used to improve classification accuracy upon a series of iterations. These kernels are often stacked on top of each other for mathematical efficiency in computing the convolutions and with all of these tricks combined, a great number of images can be used to train a network in a reasonable amount of time.

The convolution process has the additional characteristic of being translation invariant, which is a property that many physical processes have, such as speech and image data, that is, for example, a phone is a phone, no matter where it is present in a picture and its pixel value representation will locally be the same. Finally, one must add that one of the main advantages of such a network type is that it was demonstrated to be very proficient in the task of image classification and coping with very large datasets and network models of this type can be trained efficiently by current existing computer hardware, i. e., namely GPUs, in a reasonable amount of time[3][4].

5.2.3 The Temporal Segment Network Architecture

In this section I aim to briefly review the deep neural network design classifier used in the TSN used in chapter 5, its overall structure and some of the network layers and methods employed by this structure.

The basic structure of the TSN is to use a combination of static image classification combined with classification of optical flows. This idea was introduced by Simonyan and Zisserman's 2014 paper "Two-stream convolutional networks for action recognition in videos" [158] and in short reduces a video classification problem to two

[3]A single CNN layer is not enough to produce a classifier and usually a single layer fully connected NN is used as the last layer to output classes.

[4]A more efficient CNN is a deep convolutional neural network. It uses not a single layer convolution, but many stacked on top of each other so as to be able to compose features in an image. A process called back-propagation is used as the numerical optimization method to determine the weights, that is, the kernel values that will be used for classification.

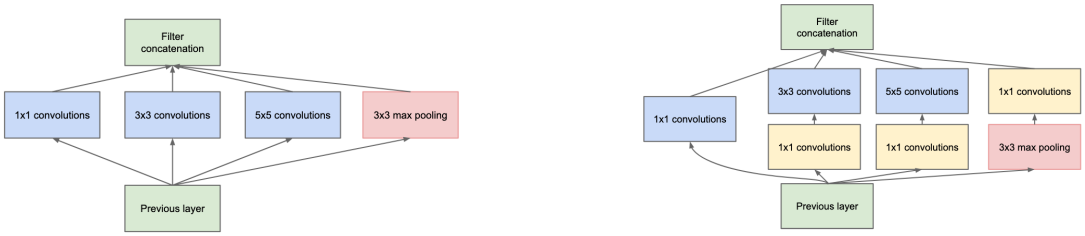
image classification problems, one on static images and another done on optical flows. This allows us to reuse successful image classification architectures to classify actions as well as a lot of the improvements that were done to those image classifiers over the years.

More specifically, the architecture for the CNN used in the TSN classifier was Inception with Batch Normalization (BN-Inception), described in more detail in [73].

Inception Modules Worth describing is the use of inception modules for the CNN. These were first introduced in the context of the ImageNet Large-Scale Visual Recognition Challenge 2014(ILSVRC14), on a paper from Google, with the GoogLeNet architecture [164]. The justification for the introduction of such architecture is suggested as an intermediate approach between pure sparse matrix multiplications (which are hard to optimise under current hardware) and dense models, which in effect is an indirect way of increasing the network size.

The paper introduces the idea of the inception module (see Fig. 5.2) as a combination of multiple convolution filter sizes that are stacked together to create a higher depth output, by just concatenating the outputs of those convolutions together. More specifically it uses 1x1, 3x3 and 5x5 convolution kernels; additionally it adds a 3x3 max_pooling kernel and the outputs from all those functions are fed into a concatenation filter. One may easily see how the dimensionality of the output would grow enormously, especially in a network architecture with many layers. The complete design of the inception module uses compression, implemented as a 1x1 convolution layer applied before higher length 3x3 and 5x5 convolutions.

As important modifications to inception modules, one must note the implementation of factorisation for larger convolutions, which would be explained in more detail as a principle in a December paper of the same year [163] - more details below. One must also point out the introduction of a new approach to training so as to allow increase learning rates, which is using normalisation each mini-batch, so as to deal with a problem of having outputs distributions from layers change too much during



(a) (b)

Figure 5.2: Naive (a) and complete (b) inception v1 module diagram with dimension reduction (from GoogLeNet [164]).

learning, a problem named **internal covariance shift**.

Later, the work with inception style networks continued and additional improvements in the modules were implemented in versions 2 and 3 described by Szegedy et al. in "Rethinking the Inception Architecture for Computer Vision" paper from 2015 [163]. In this paper, focus is being put into optimising computationally inefficient large convolutions and make better use of currently available hardware by, for example, (1) avoiding representation bottlenecks, especially in the first layers of the network, which limits representation and (2) factorising large convolution, which entails replacing such large convolutions, say a 5x5 convolution by 2 serial 3x3 convolutions without losing much representational power [163]. I see fitting mentioning this work here, because it seems to go more in depth on explaining the reasons for some of the alterations that were already introduced in the BN-Inception model [73]. Additional design optimisations and considerations are made and I refer the original paper [163] for a more careful explanation of all the design principles used to optimise those inception modules.

More generally, the TSN implementation of the BN-Inception model using this architecture has a number of stacked inception modules, with occasionally interposed max_pooling layers to reduce grid size, commonly employing more traditional convolution layers first and using the inception modules only at the end so as to limit the issue of dimensionality growth. A diagram of the full network can be seen in Appendix B.

Pre-training The datasets used in Two-Stream architecture [158] and the Temporal Segment Network architecture [178] both decided to use the UCF101 [160] and the HMDB51 [97] datasets for their exploration. Those datasets, although rather large (with around 9.5K and 3.7K videos) are still too small to allow training of very deep models such as the ones that are being employed currently and require a very large number of images to train. The risk here is to have your particular network overfit too much, which is an issue.

So as to mitigate this, pre-training strategies were employed in both of those studies. For the static image classifier (spatial stream), this endeavour seems more straightforward as typical cross-dataset fashion, employing a pre-trained network fed with images from the much larger ImageNet dataset [37] with 3.2M images in total.

For the optical flow, however, since this input depends on time adjacent frame sequences, the ImageNet dataset pre-training was not adequate and Two Streams [158] and TSN [178] diverge on setting up initial weights for those networks. The idea from Two Streams was to train optical flows with both UCF101 and HMDB51 together and use two different soft-max layers on top of the final fully connected layer and their own separate loss function as it was suggested by Collobert and Weston [32] in the multi-task learning framework model. The overall training loss is then computed by summation of training losses from the 2 different tasks (i.e. learning either HMDB51 or UCF101) and the weights' derivatives found by using back-propagation.

TSN uses a much simpler approach which is grounded on using the same general architecture for optical flows as it is used for static frames. The calculated flows, which are represented as a floating point matrix, are discretised and converted to a 0 to 255 interval so as to be represented as a grey-scale image, and as such the sizes of images of both RGB and flow channels match. Now, to initialise the weights, the weights of the pre-trained static image are averaged over the RGB channels and those are replicated to match the depth, i.e. the number of frames, of the flow channel. The TSN implementation paper suggests that this initialisation tends to work well to reduce the effect of overfitting.

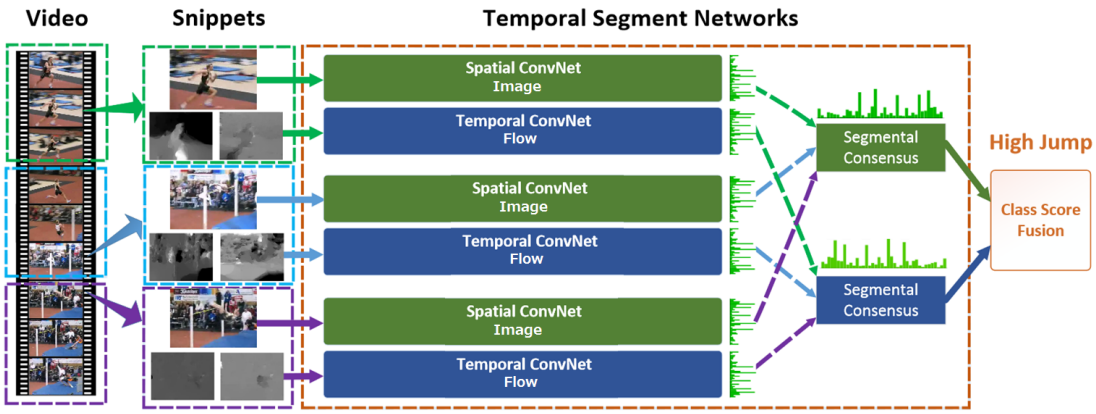


Figure 5.3: TSN structure model (adapted from [178]). Note special and temporal ConvNets, i.e., image frames and optical flow respective inferences. On vertical axis there is time progression. Segment consensus over time can be as simple as an averaging function.

5.2.4 Classifier Architecture

The classifier is a wrapper for a caffeNet in python, which was modified to load from streams instead as loading from video files. Again not much was changed from its original structure (see Fig. 5.3), but mostly the portions pertaining loading of the datasets.

One important change that needs to be mentioned concerns a hard problem in action recognition, which is, how long does an action last. The TSN architecture tries to optimise training by regularly sampling frames in a rate which is smaller than the real frame rate, going by the assumption that, in a well trained network, similar frames will have similar output and as such, neighbouring frames can often be skipped without much impact on overall performance. Moreover, it reads more often short duration activities than longer activities, so as to try to not miss important stages that can help a classifier grasp the underlying structure of an action.

This variable sampling requires that one must know the video duration before going through the frames, so as to classify only the smaller subset that would present the novel information.

In a real-time implementation, this is unfortunately not possible and other strategies must be used.

As with my initial tests, I was still relying on playing back recorded videos of the HMDB51 dataset and then my own videos, this could still be done, but later for deployment, it would mean that I would have a classifier working in a different way from which it was trained, which could negatively affect performance in an unpredictable way.

The way I chose to solve this issue was to remove this feature and simply skip a fixed number of frames. Unless otherwise noted, the results contained in this document were elaborated with this frame skipping constant set to 5 frames, that is, in a 30fps video, I would examine 6fps, which I view as a reasonable compromise between performance and speed.

The algorithm was implemented with 3 different outputs:

- one that uses services to start and stop the classifier and consider the whole action sequence in order to make the classification, published as /action_own;
- an instant classifier that uses only the current frame (or in case of flow, the current last 5 frames from a stack), published as /action;
- a compromise between those two which uses a moving window to classify the N last frames, with N generally chosen to be 50 frames, published as /action_fw;

And each of those outputs was published in 3 different ways so as to allow visualisation and further processing if needed:

- The most probable class output from the classifier, std_msgs.msg.String output, published with the base topic's name, that is, either /action, /action_fw or /action_own;
- The most probable class with the confidence level, a custom caffe_tsn_ros.msg.Action message, published under base topic's name with _label appended to it;
- A dictionary of labels and their respective confidence levels, a custom caffe_tsn_ros-.msg.ActionDic message, published under base topic's name with _label_dic appended to it;

A further modification implemented was to remove the last fully connected aggregation layer [scores] that gets results from the convolutional layers and outputs class memberships. The layer was removed and instead of returning classification results and confidence levels, the full 10240 array defining response from the algorithm was published as a ROS multiarray. This node was initially deployed using only instant frame responses (corresponding to the /action topic), due to difficulties in storing and stacking the output from many frames (variable size array) and loading them. However, due to my hypothesis that this could be negatively affecting results too greatly, multi-array stacks were also implemented.

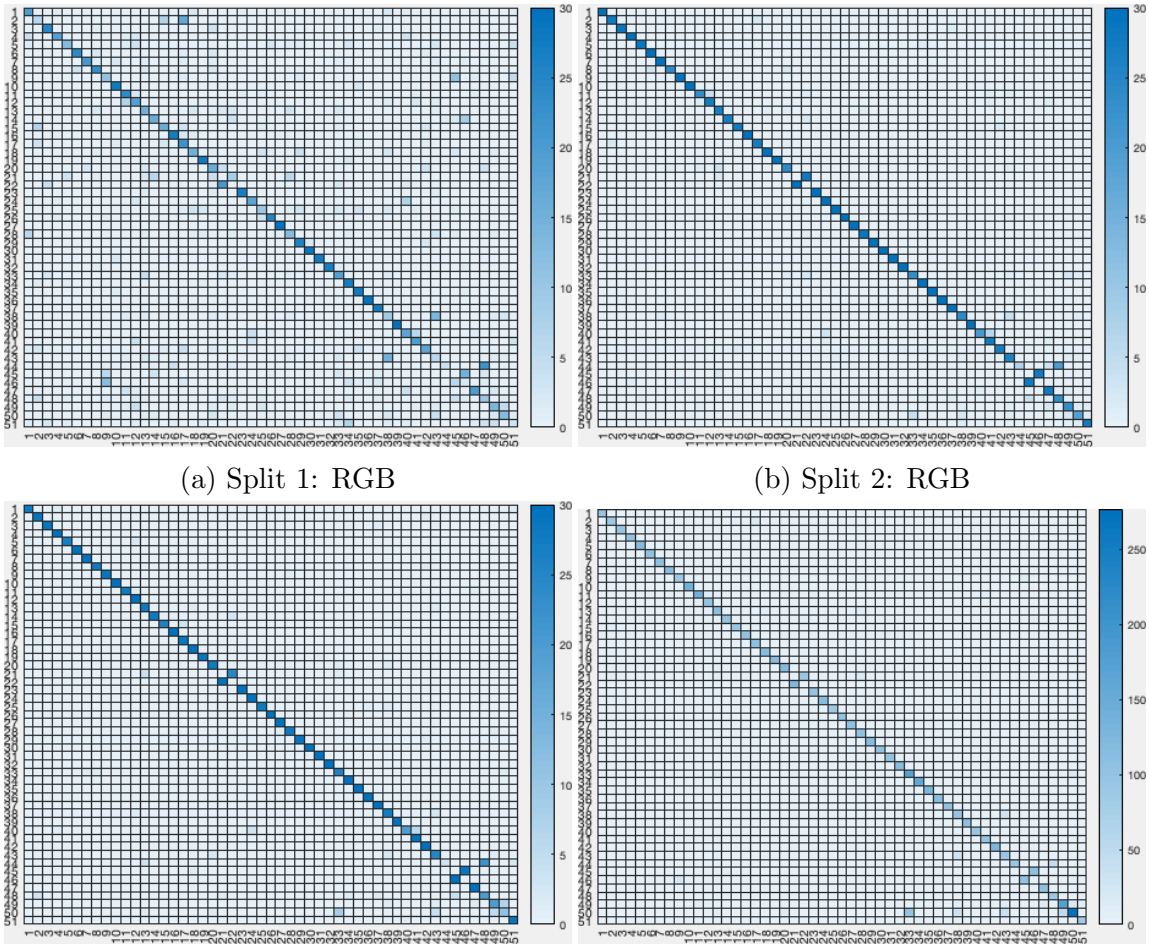
The idea behind this splitting of networks was that of hopefully generalising the process of transfer-learning, in which a network's last layer is removed to allow similar classifiers to be used without extensive training of the whole network in cases only a small dataset is available (see Section 5.5).

This node was as well deployed in its own Docker image, with its own catkin workspace. The full code for the ROS package is available at: `https://github.com/mysablehats/caffe_tsn_ros`.

5.3 First Experiment: Replication of Temporal Segment Network Results

The first experiment implemented was the validation of whether or not I successfully could replicate the results from TSN in a real-time classification.

Results from my implementation, running a real-time classifier on a dataset, using the pre-trained model provided yielded the resulting confusion matrices in Fig. 5.4. This was trained using the whole movie segment (/action_own topic) and classifying the most probable action it represented considering the set of samples for that segment using the RGB data only without re-sizing, that is, without reading the modified frame data from denseflow, but using the dataset frame data itself. The accuracies for splits 1-3 and average overall were respectively: 0.60131, 0.84248,



(a) Split 1: RGB

(b) Split 2: RGB

(c) Split 3: RGB

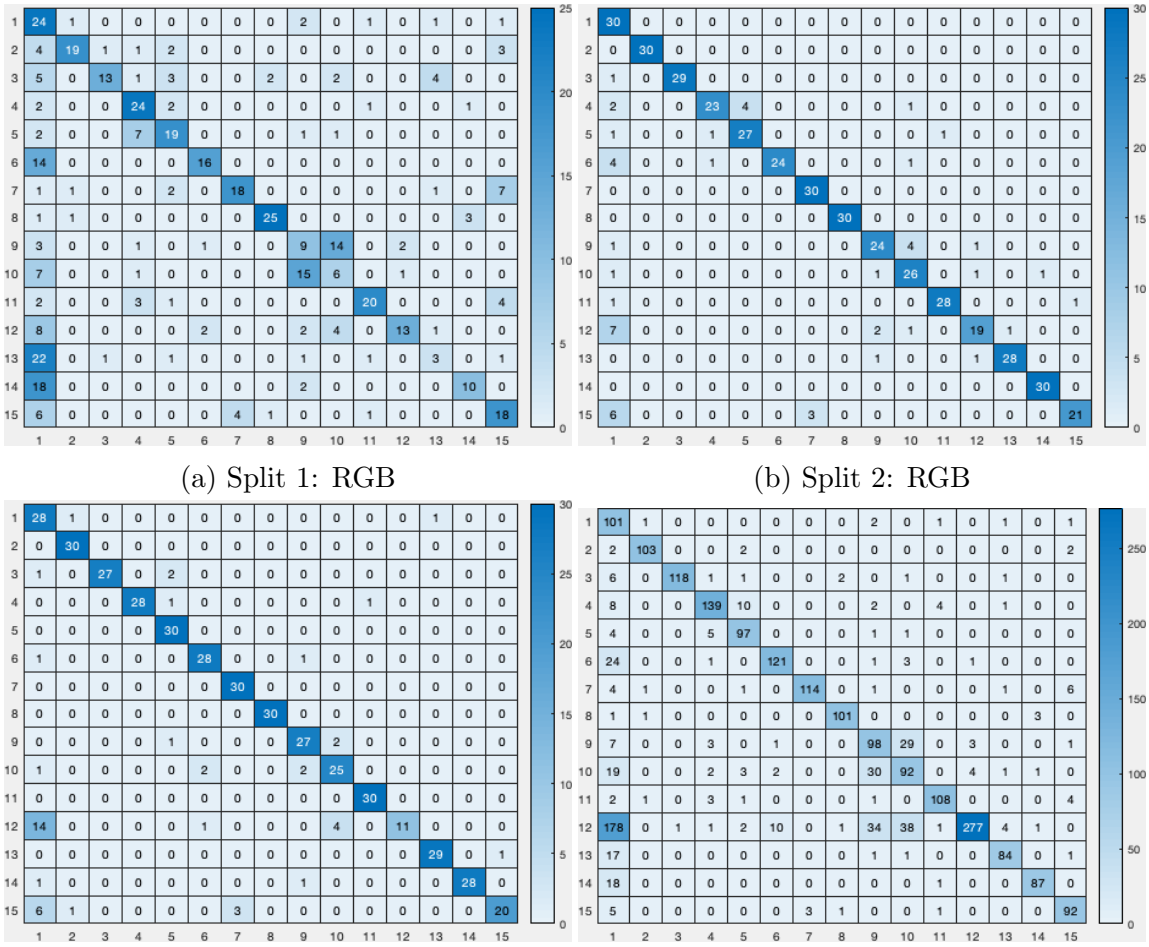
(d) All splits combined: RGB

Figure 5.4: Confusion matrix responses from the vanilla TSN classifier ran on test splits of HMDB51, by split with RGB modality, running on full actions .

0.845 and 0.78303.

I am also interested in the responses from the indoors single person smaller set of actions, the HMDB14, which are plotted in Fig. 5.5. The accuracies for splits 1-3 and average overall were respectively: 0.51778, 0.88667, 0.89111 and 0.75823. The responses from the moving window with window size N=50 (topic /action_fw) were also analysed and are presented in the Fig. 5.8. The accuracy of this configuration was for splits 1-3 and global, respectively: 0.6171, 0.8614, 0.8519 and 0.7955. For the HMDB14 the results were respectively 0.5477, 0.8904, 0.8701 (splits 1-3) and 0.7589 globally (confusion matrices not shown).

An additional test that was performed (and used for the following examinations) was to determine how good were classifications when I considered not each action as a whole, but only one frame of each action (topic /action). These instant responses



(a) Split 1: RGB

(b) Split 2: RGB

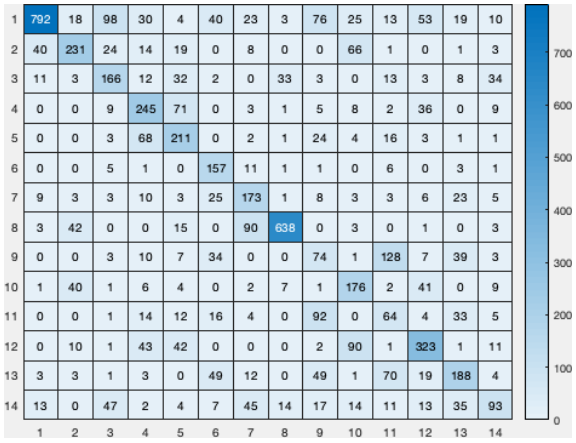
(c) Split 3: RGB

(d) All splits combined: RGB

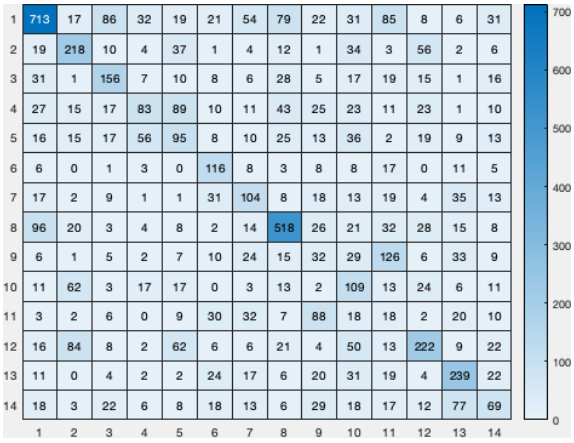
Figure 5.5: Confusion matrix responses from the vanilla TSN classifier ran on test splits of HMDB14, by split with RGB modality, running on full actions - responses from /action_own topic.

from each time frame (as opposed to best label per action video) can be seen in Fig. 5.6 and Fig. 5.7. Average accuracy over all splits of the restricted set with instant responses (HMDB14): flow network: 0.432894; RGB network: 0.580788.

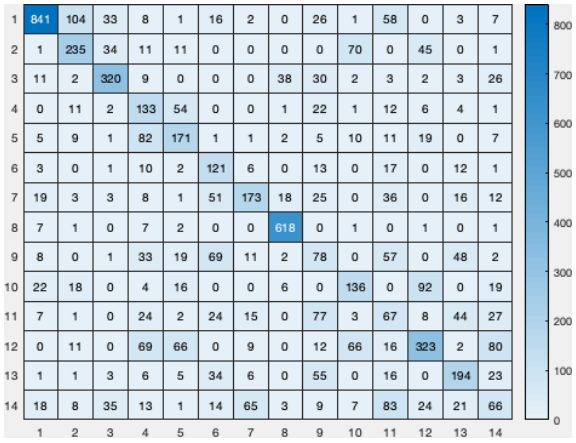
As one can see, using a fixed time step did not negatively affect the overall performance, with my implementation achieving very similar results to the ones from the paper that introduced TSN for both /action_own and /action_fw implementations. Using only instant frame classifications, that is /action topic results, however seemed to have bit more of an impact on decreasing accuracy.



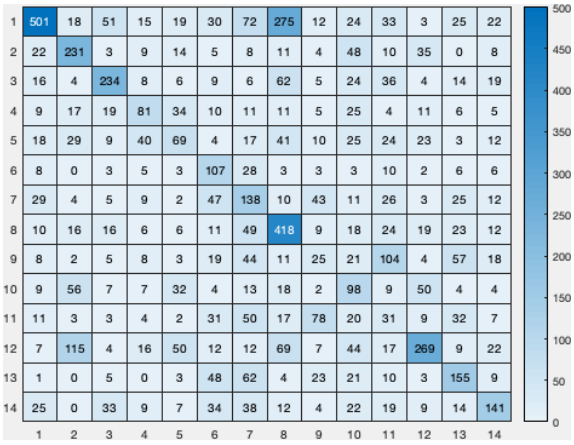
(a) Split 1: RGB



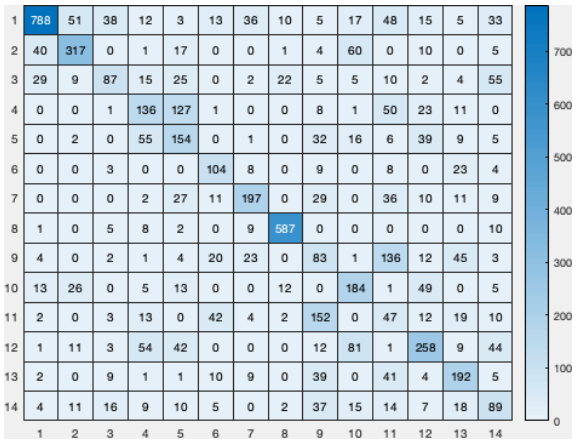
(b) Split 1: flow



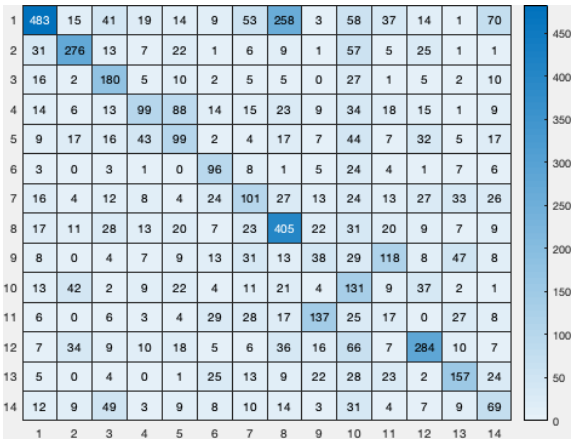
(c) Split 2: RGB



(d) Split 2: flow

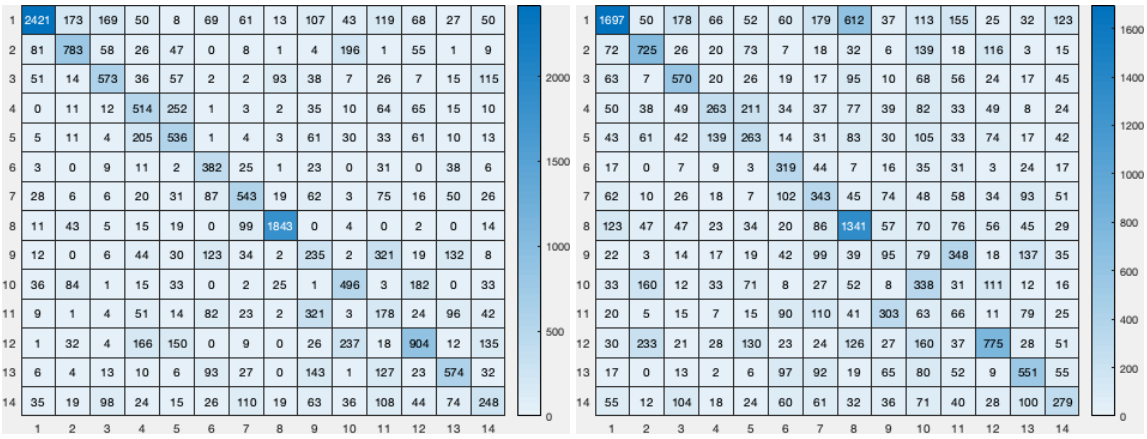


(e) Split 3: RGB



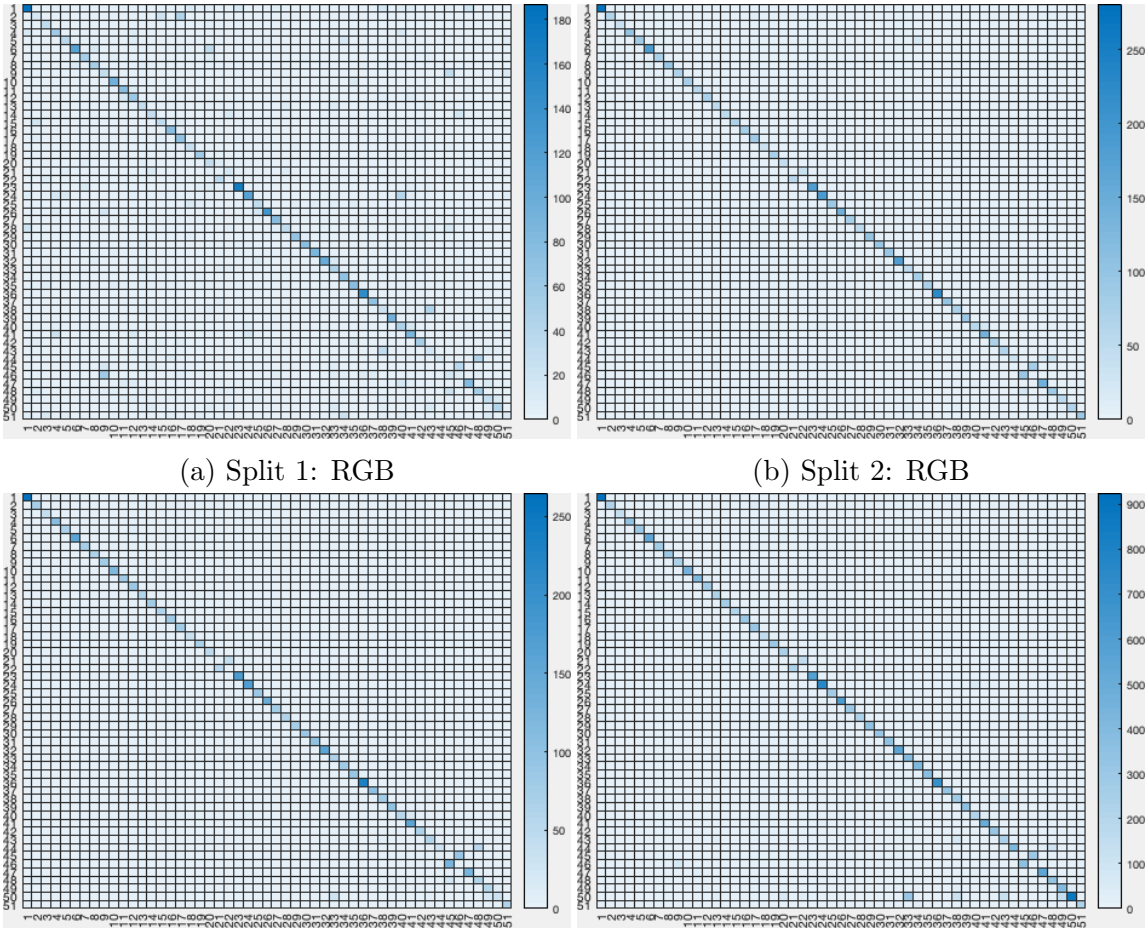
(f) Split 3: flow

Figure 5.6: Confusion matrix responses from the vanilla TSN classifier ran on test splits of HMDB14, by split and modality, e.g., flow or RGB, running on instant actions - responses from /action topic.



(a) All splits combined: RGB (b) All splits combined: flow

Figure 5.7: Combined confusion matrix responses from the vanilla TSN classifier ran on HMDB14, by split and modality, running on instant actions - responses from /action topic.



(a) Split 1: RGB (b) Split 2: RGB

(c) Split 3: RGB (d) All splits combined: RGB

Figure 5.8: Confusion matrix responses from the vanilla TSN classifier ran on test splits of HMDB51, by split with RGB modality, running on moving window size 50 - responses from /action_fw topic.

5.4 Experiment 2: Real Time Classification on Scitos G5

The next experiment was to test whether the classifier as it is would be capable of classifying real-time data from an unknown origin not from the dataset.

This is necessary as there is likely to be a difference between dataset (which is a restricted set) and real file captured actions.

The experiment design consisted of a single room with the robot positioned stationary, only moving a pan-tilt unit in which the kinect sensor was mounted, running the active vision algorithms and the ROS infrastructure necessary to run the pre-trained TSN. A simple script was implemented to say and show to a subject on video screen what sort of action he should perform from the list of restricted actions. I replaced the dataset loader topic for the kinetic videocamera and both recorded the video and depth channels for later use and did a real time classification.

Results of such single first experiment yielded the following confusion matrix which can be seen in Fig. 5.9. For this experiment, the more accurate version of the classifier's output was used, namely the topic /action_own. The experiment was performed twice with similar responses both times, yielding a global accuracy of 21.43%.

Surprisingly the algorithm here (ran with RGB input only) made most mistakes by marking actions as being 'pour', not a common error in subsequent runs for RGB (where most actions were considered 'smiles').

Subsequent runs yielded similar results. The confusion matrices for the combined results can be seen in Fig. 5.10. Accuracy for each classifier considered in isolation was cf1_rgb : 0.13428, cf1_flow : 0.29204, cf2_rgb : 0.23941, cf2_flow : 0.26519, cf3_rgb : 0.1434, cf3_flow : 0.19057. Their average accuracy of all RGB classifiers was 0.17206 and all flow classifiers was 0.25031.

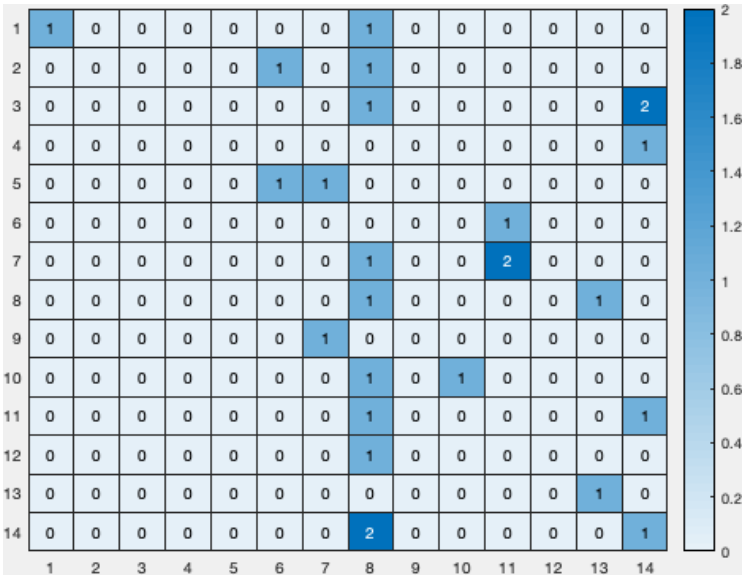
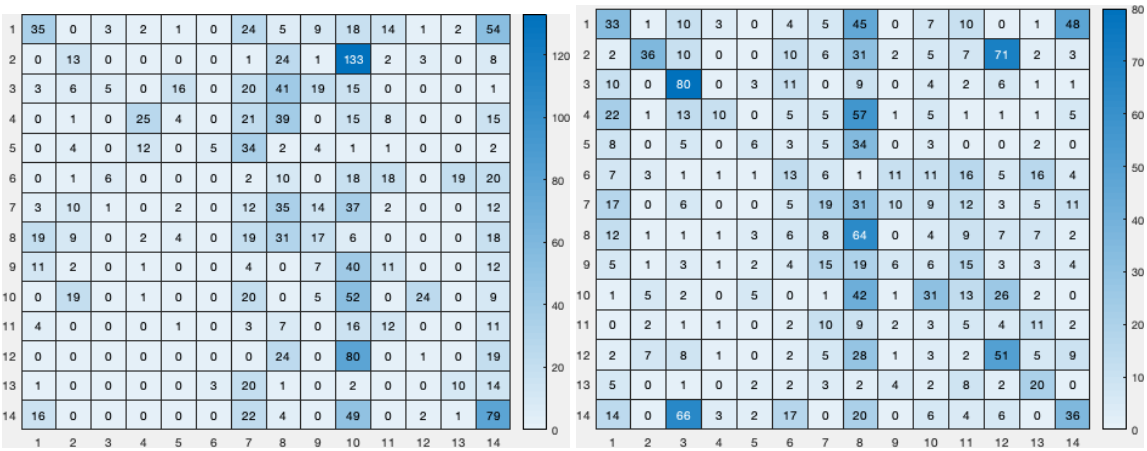


Figure 5.9: Response for first experiment with robot and full loop with vanilla TSN classifier with only RGB channel. Experiment was performed twice with 3 actions correctly classified in the first try and 3 actions more on second try.



(a) All splits combined: RGB (b) All splits combined: flow

Figure 5.10: Combined confusion matrix responses from the vanilla TSN classifier ran of MYSET, by split and modality.

5.5 Experiment 3: Transfer Learning

The next investigation attempt was to try to identify whether I could train only the last layer of the model to deal with unseen data and increase its overall performance.

In a usage case, this would be akin to a robot having to be calibrated first before use, so as to know the subjects house and how he performs actions, which is a reasonable requirement.

5.5.1 Sklearn Explorations

For this experiment I wanted to use any suitable model to replace the last layer of the network so as to try to optimise the response for the data I gathered with the robot. The idea is to perhaps optimise the response by adding more features, so as to enable a classifier to be able to also optimise the response from the flow and RGB layers (as opposed to just using the argmax for the combination of those two). Initially, I just tried to fit a model with the combined output of RGB and flow models, namely a one-to-one matching of RGB and flow responses form global layers. This was done so as to be able to use the data I already gathered with the robot implementation and allow for further explorations using it.

Frame matching was done in an approximate manner as the data from the 2 networks was neither synchronised nor time-stamped. I only removed the samples which didn't line up based on real class differences when compared to the last classes for each of those streams, that is, if the responses from the i-th element where, say for the RGB 'dive', last action 'dive' and for the flow 'clap', last action 'dive'; then the flow network was already on a newer class, so the RGB samples will be dumped until they line up again. Note that the reading of the datasets is not randomised in order, all of the instances from each action being lumped together, so some of the matchings were also not lining up, however they would remain undetected. I hope that those, however are few, and should not compromise the end results that much, considering the number of samples I have.

	Train			Test		
	RGB	flow	combined	RGB	flow	combined
Split 1	13463	13463	13461	6215	6215	6206
Split 2	14213	14214	14208	6365	6365	6360
Split 3	14290	14289	14286	6023	6023	6017

Table 5.2: The complete table of loading frames with RGB and flow channels combined frame by frame for HMDB51.

	Train			Test		
	RGB	flow	combined	RGB	flow	combined
Split 1	1197	1198	1190	509	509	509
Split 2	1231	1231	1228	478	478	478
Split 3	1084	1084	1084	506	506	506

Table 5.3: The complete table of loading frames with RGB and flow channels combined frame by frame for MYSET.

Trying this algorithm with my own initially acquired data for the first training split caused only 8 frames from over 1197 not to line up, 4 from flow and 4 from RGB, dropping from a total of 1197 samples to 1190 samples - these being the worst results on MYSET. The complete alignment results for both sets initially used can be seen in the tables 5.2 and 5.3 [5,6].

It should be noted that these numbers are underestimating alignment, since the loading of the datasets is done class by class and some misalignment could be hidden. This is a known issue and should have been solved using message filters in ROS and direct saving of the matched data, however creating these sets is rather time consuming and I expect the number of transitions to be small in comparison with the lengths of the videos. That is, for my reduced split 3 from HMDB51 there are 420 included in the test which resulted in 6023 samples. If I assume 50% of those will be improperly aligned for at most 1 frame, I get roughly 200 frames out of 6000, which is around 3% error being introduced by this procedure. I expect this not to be

[5]Note that these frames were subsampled from the video input, with a step = 6, that is, only 1 every 6 frames was captured

[6]These values are unfortunately also not a conservative estimate of the number of misaligned frames, since I am only considering class differences as misalignments and I have 100 video changes per class and only 14 class changes - a more conservative estimate would be to estimate the misalignment error as 7.2 times larger than 8/1190 so around 4.8%; using time-stamped topics, message filters and a time_synchronizer, this initial implementation was simplified and thus the usage of time-stamped synchronized topics was skipped due to time constraints.

Estimators	10		30		100*		200*	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
split1	0.99925	0.52304	1	0.57638	1	0.62053	1	0.61731
split2	0.99944	0.50031	1	0.56038	1	0.61006	1	0.61918
split3	0.99965	0.53714	1	0.61359	1	0.63038	1	0.6347
average	0.9994	0.5202	1	0.5835	1	0.6203	1	0.6237

Table 5.4: Performance of different classifiers on training and testing splits of HMDB51 with combined RGB and flow channels by number of estimators. * random state=0 for reproducibility.

a major issue, but it will be kept in mind as one of the points where improvements can be made.

Initial Tests: sklearn.ensemble.RandomTreeClassifier It has come to my attention that my dataset has a lot of features, that is 10240 per modality, totalling 20480 for both RGB and flow channels. I have in my restricted set from HMDB51 only 13000 samples, which would allow for immense amounts of overfitting. In order to try to go around it, I tried a robust classical method to try to go around this problem.

The first attempt was to try to fit a Random Tree Classifier and see what kind of performance that would give when compared with a NN last layer. I wanted to know if it was also feasible to test multiple parameters of such a classifier giving that the size of my dataset is quite large. Using both flow and RGB combined stream features the test finished in under a minute with 10 estimators with responses below.

Another reasoning for using a Random Tree Classifier is its overall high performance with few parameters to tune. This would hopefully provide a reasonably good classifier that is fast enough and can serve as a ball park for some refinements. A table with the results obtained from trying to classify the features extracted from HMDB51 dataset with different number of classifiers is presented in Tab. 5.4.

A graph of the overall behaviour can be seen below (see Fig. 5.11):

As a next test, so as to try to reduce overfitting I tried reducing the maximum number of features used by each classifier - the default is sqrt(n_features), which in this case would be 143 features for only 14 classes. I tried another common setting

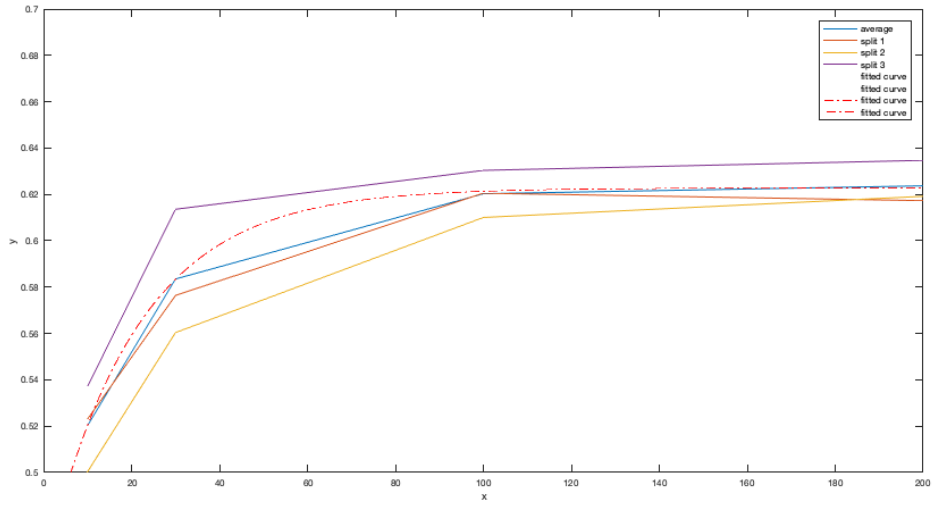


Figure 5.11: Accuracy responses from classifier using RGB+Flow with random forest classifiers with varying number of estimators (n_estimators = [10, 30, 100, 200]). Dashed red line is a fitted curve of type $a+(b-a)e^{-kn}$, with a = 0.6228, b = 0.4575, k = 0.048. *Note that graph was zoomed to 0.50-0.70 to highlight differences.

	Training	Testing
split 1	1	0.61086
split 2	1	0.60613
split 3	1	0.63767
average	1	0.6182

Table 5.5: Reduced number of features to log2 for combined RGB and Flow features from HMDB51 dataset.

which is 'log2', which in this case would yield 14 max_features. Results from this test can be seen in Tab. 5.5. Actually with 1 single feature and 200 estimators I can already overfit as shown in table 5.6.

This algorithm is clearly overfitting and other attempts to prevent it (such as reducing max_features) were not successful. However, it may be interesting to know whether it can classify the initial captured dataset MYSET and whether it does

	Training	Testing
split 1	1	0.58153
split 2	1	0.53884
split 3	1	0.59914
average	1	0.5732

Table 5.6: Reduced number of features to 1 for combined RGB and Flow features from HMDB51 dataset.

	Training	Testing
split 1	1	0.178782
split 2	1	0.267782
split 3	1	0.201581
average	1	0.2160

Table 5.7: Combined RGB and Flow features trained on HMDB14 dataset data (training) and tested on MYSET (testing).

	RGB	FLOW
Split 1	0.11002	0.29666
Split 2	0.22176	0.24686
Split 3	0.17787	0.17787
Average	0.16879	0.24046

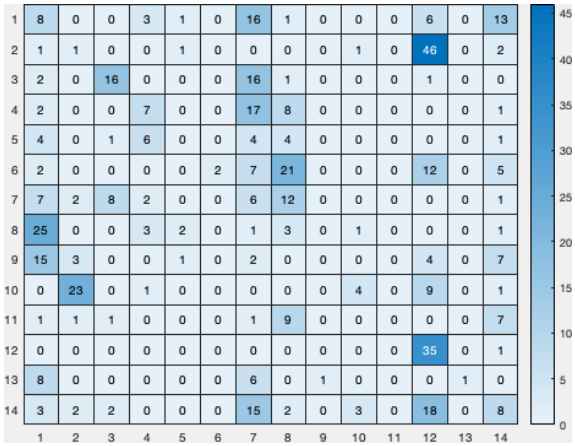
Table 5.8: Accuracy (testing) for RGB and flow modalities for Random Forest classifier, trained on HMDB14, tested on MYSET.

improve on just using the normal fully connected last layer from the already trained Caffe network.

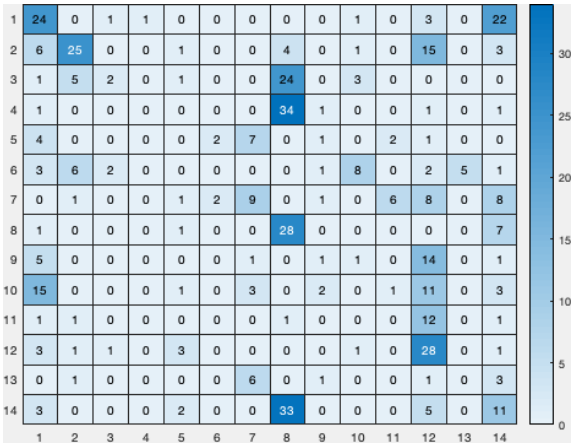
Using sklearn.ensemble.RandomTreeClassifier without additional training data As a first test, I want to be able to compare how well this replaced last layer would perform when compared to the original algorithm with no help from training it on the acquired data, that is: train on HMDB14 and test on MYSET. I chose as frame of comparison a random tree classifier with 100 estimators and the normal number of max_features, only using random_state=0 for repeatability (results in Table 5.7).

The confusion matrices for combined flow and RGB can be seen in Fig. 5.12. For comparison purposes, also the isolated flow and RGB results are presented in Fig. 5.13 and Fig. 5.14 and Tab. 5.7.

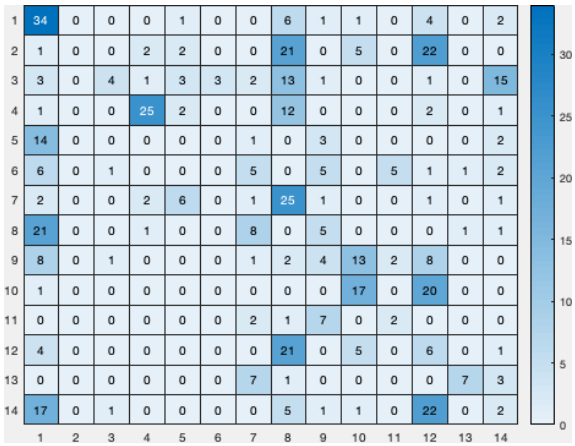
These results are very similar to the Neural Network model, which may indicate that MYSET does not have actions matching those of the HMDB14, that is, sit on HMDB14 is not the same sit as in MYSET. Another possibility is that the features themselves that were acquired are basically different from those in an office environment. Finally it may be that the structure itself of TSN is adequate to train HMDB51 (and datasets acquired in similar fashion), but does not perform well on



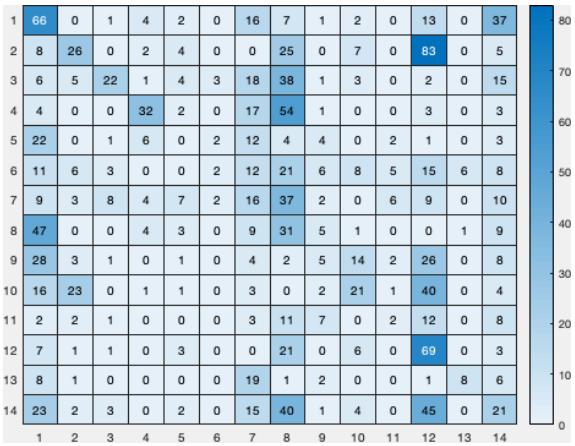
(a) Split 1: combined flow and RGB



(b) Split 2: combined flow and RGB

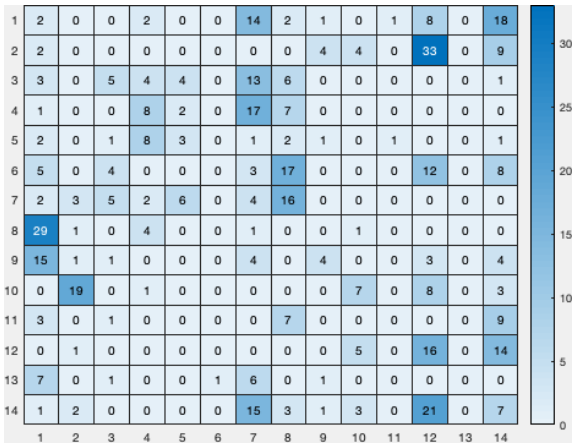


(c) Split 3: combined flow and RGB

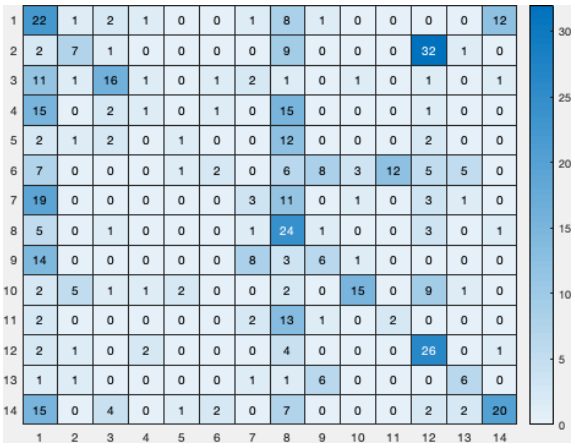


(d) All splits combined: combined flow and RGB

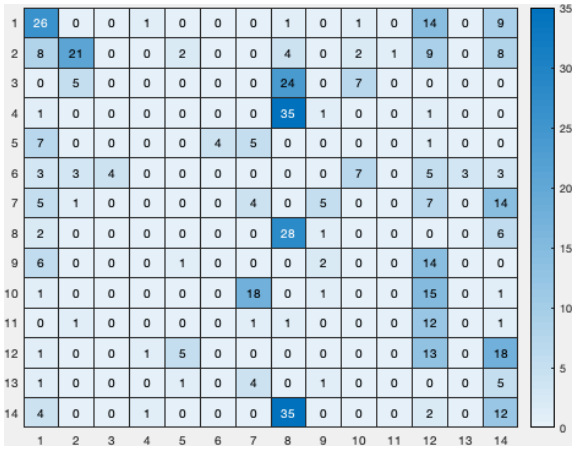
Figure 5.12: Confusion matrix responses from the vanilla TSN classifier ran on test splits of HMDB14, by split with combined flow and RGB modality, running on instant actions - responses from /scores topic.



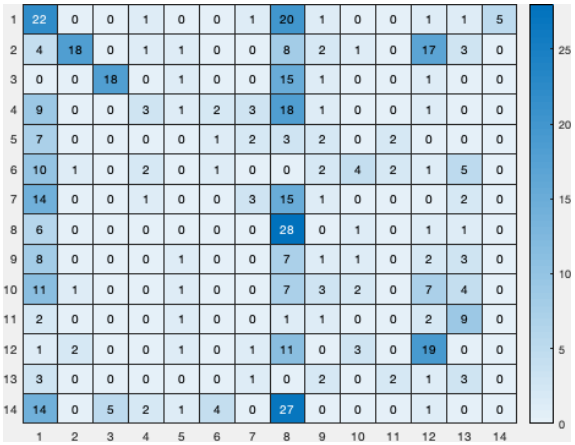
(a) Split 1: RGB



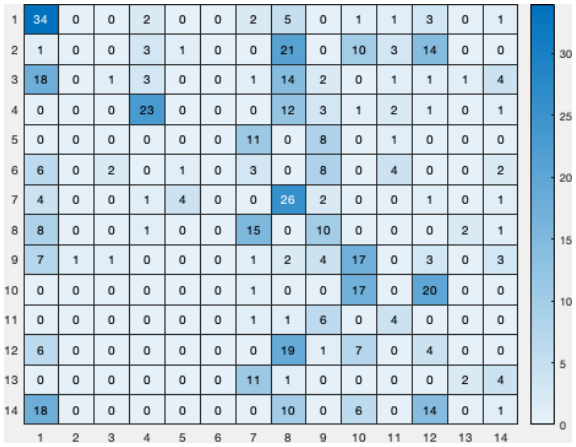
(b) Split 1: flow



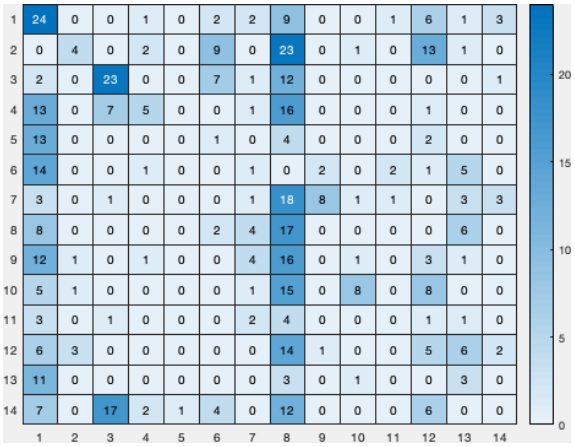
(c) Split 2: RGB



(d) Split 2: flow

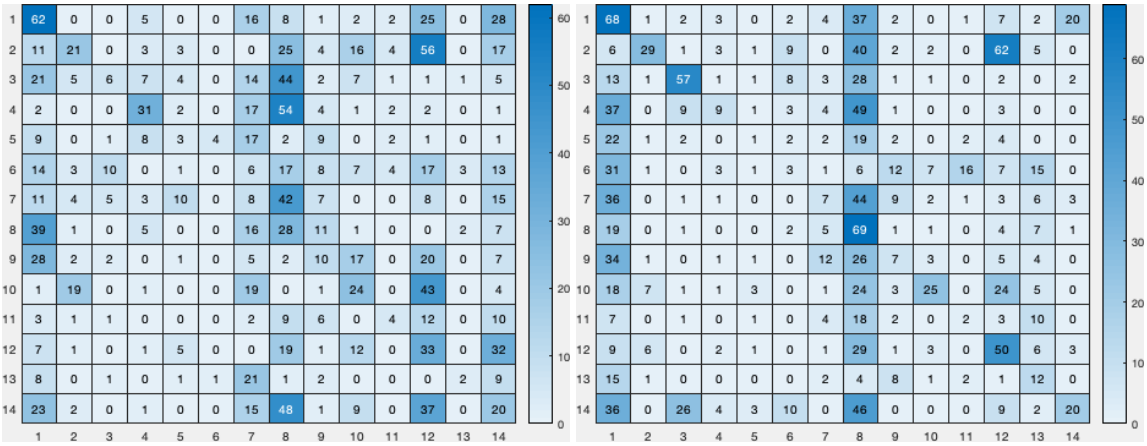


(e) Split 3: RGB



(f) Split 3: flow

Figure 5.13: Confusion matrix responses from the Random Forest Classifier with TSN features ran on test splits of MYSET, by split and modality, e.g., flow or RGB.



(a) All splits combined: RGB (b) All splits combined: flow

Figure 5.14: Combined confusion matrix responses from the Random Forest Classifier with TSN features ran on test splits of MYSET, by split and modality.

a set of actions in the wild.

If the first or second case is true, then acquiring enough data to train the last layer model on only the dataset should resolve the problem. Finally, if it is the case that TSN structure itself cannot be used, perhaps it can be extended (with some ensemble method) or other architectures could be used (see section 1.6.2) instead.

Using sklearn.ensemble.RandomTreeClassifier with additional training data A next test was done to see if adding my own acquired data and training on larger dataset would improve those results. Two different scenarios were tried, namely:

- use my dataset MYSET and HMDB14 combined to train the classifier, and test on MYSET, so as to test whether I need to add new data to the original dataset HMDB14 to make it more complete and descriptive of those set of actions;

- use my set MYSET and its appropriate splits to both train and test, so as to check whether the action set, as it was captured, was describing considerably different actions from the original set and would award it having complete new categories so as to classify them.

This was the best results I obtained so far in the testing set MYSET, which can

be seen with confusion matrices in Fig. 5.15 and Fig. 5.16. For the RGB classifier, testing accuracy for splits 1-3 and combined are 0.18271, 0.10251, 0.2253 and 0.1702, however for the flow classifier, accuracy was considerably higher, for splits 1-3 and combined respectively: 0.34185, 0.39749, 0.39921 and 0.3795.

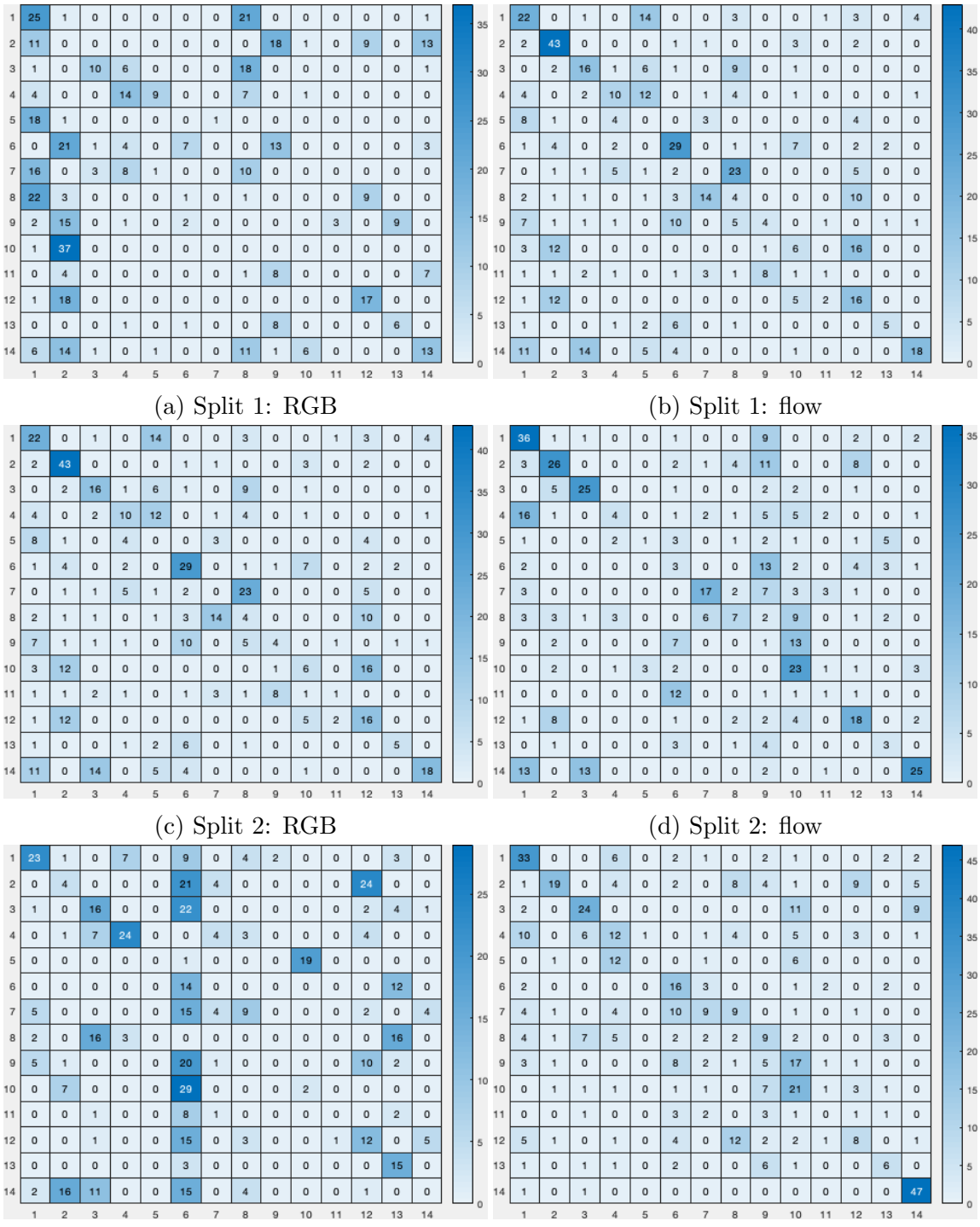
It may be that due to the number of tests performed the results obtained here are only due to cherry-picking and the results here do not represent a step forward to solving the problem, however this will only be known with further tests with a larger dataset. For the moment, however, I will assume that these results point to something about the dataset I hasn't been considered before.

I suspect that these results together with the similarity in responses from both Fig. 5.10 and Fig. 5.14 point to the fact that these actions, as captured, do not constitute a group that is contained within HMDB14. Or better said, the resulting classifier that is generated from using HMDB14 does not generalise towards the set of actions I gathered on my own set. It may be that it does not even capture the correct features necessary for it to happen, or simply that the data sources are too different (either from the cinematography choices or from a different understanding of what 'smile' or 'sit' means).

It may also be the case that not considering the action as a whole and only classifying it frame by frame is too strict a requirement from this classifier (that hypothesis is corroborated by the fact that flows - which use consecutive frames - yield better results than RGB images). Those hypothesis guided the future investigations.

The results from Fig. 5.16 motivated me to try and see if I could improve it further with a combination of both RGB and Flow features. The responses for the Random Forest classifier for this approach can be seen in Fig. 5.17. The recorded accuracy for splits 1-3 and average were respectively: 0.210216, 0.184100, 0.332016 and 0.2431, not better than just using FLOW modality alone.

I can see that this combination approach, at least as it was implemented, did not increase the accuracy of the random tree classifier, most probably due to the high number of features present in comparison to the size of the dataset. As a matter of



(a) Split 1: RGB

(b) Split 1: flow

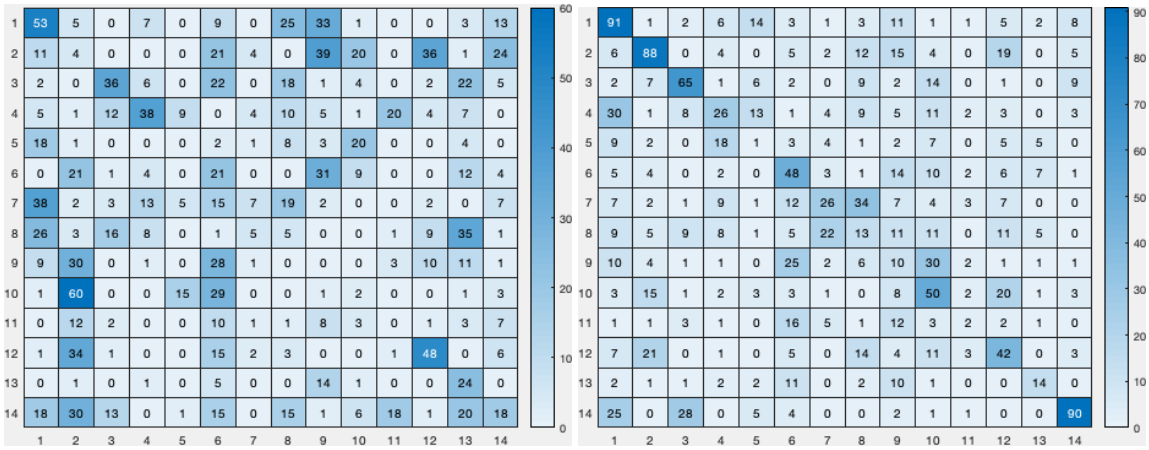
(c) Split 2: RGB

(d) Split 2: flow

(e) Split 3: RGB

(f) Split 3: flow

Figure 5.15: Confusion matrix responses from the Random Forest Classifier with TSN features trained and tested on splits of MYSET, by split and modality, e.g., flow or RGB.



(a) All splits combined: RGB (b) All splits combined: flow

Figure 5.16: Combined confusion matrix responses from the Random Forest Classifier with TSN features trained and tested on splits of MYSET, by split and modality.

fact, this was investigated before during the second experiment while classifying the CAD-60 dataset. Then I found that for datasets with high feature dimensionality, methods such as SVM and especially KNN were considered superior[7] [5].

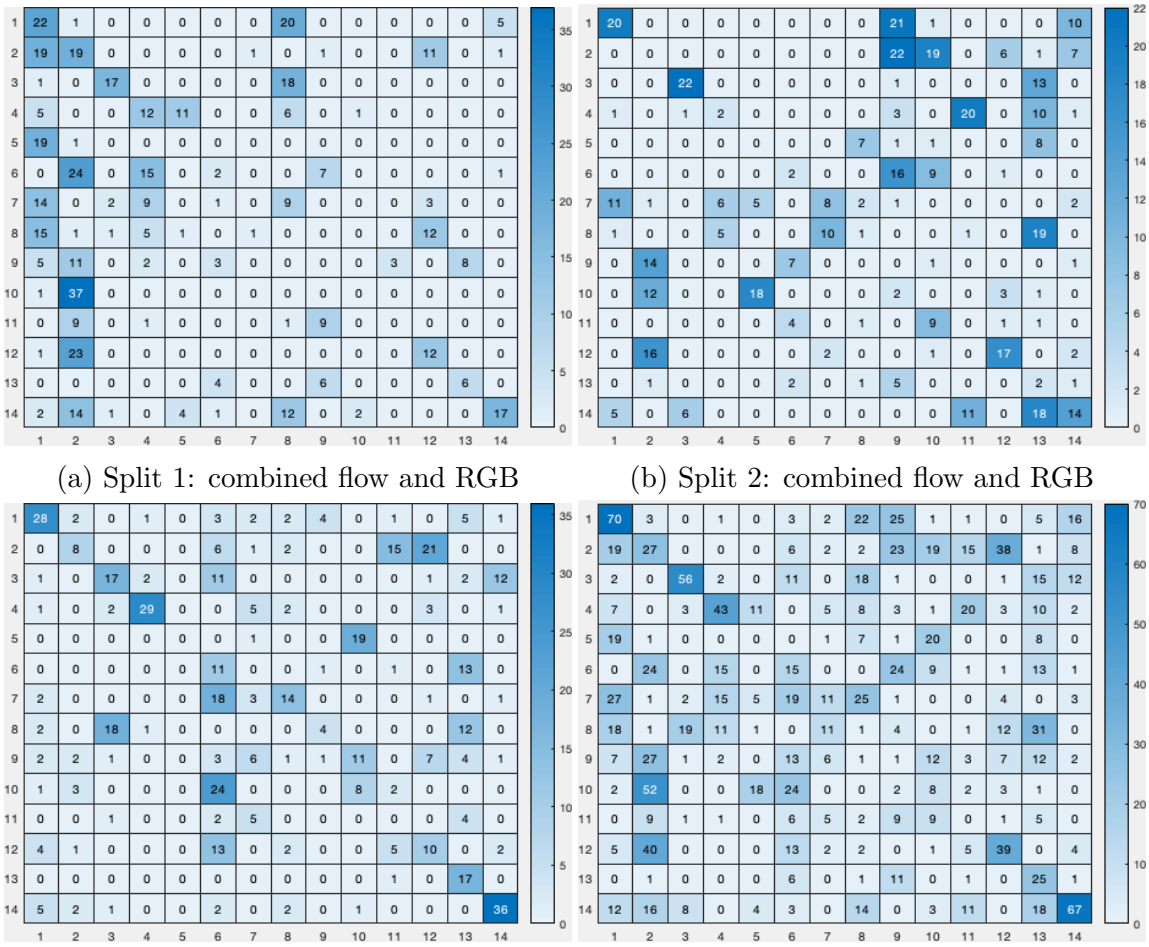
5.5.2 Pytorch Explorations

As a first attempt, I wanted to check if I could retrain the last layers of a NN to the same level of accuracy as the originally trained algorithm using a pyTorch implementation instead of Caffe. This problem, although seemingly trivial, should allow us to automate transfer learning. Given the high dimensionality of the data received as input, and that the original network form TSN has a single layer fully connected layer as input, I decided first to try to fit a net without any hidden layers. For loss function I used the mean square error, using a learning rate of 5E-6 with 100 batches of 1000 epochs.

I performed the same procedure as before, first testing the HMDB14 set with its training and testing splits, then training on HMDB14, testing on the splits from MYSET, then training on HMDB14+MYSET testing on splits of MYSET and finally training and testing on MYSET splits.

The loss function and accuracy over time can be seen in Fig. 2.6, as well as the

[7]Under the set of supervised classifiers and datasets tested by Amancio et al., that is kNN, Perceptron, Random Forests, Naive Bayes, SVM, Logistic, Simple Cart, C4.5, Bayes Net.



(a) Split 1: combined flow and RGB

(b) Split 2: combined flow and RGB

(c) Split 3: combined flow and RGB

(d) All splits combined: combined flow and RGB

Figure 5.17: Confusion matrix responses from a random forest classifier with features from TSN, trained on MYSET, ran on test splits of MYSET, by split with combined flow and RGB modality, running on instant actions - responses from /scores topic.

control signals that show which test was performed. I can see from those graphs that the network is clearly overfitting the data, probably due to the relatively high number of features when compared to the size of the dataset (10240 features vs. 13000 images on the whole HMDB14). The problem is even more pronounced when just training on MYSET due to greater disparity in dataset size and number of features.

This is a known issue when using BN-Inception to classify the HMDB51 dataset and has been addressed by using heavy dropout (0.8) [196]. The paper [127] suggests that in a similar scenario of using a single hidden layer, that dropout may be superior to L2 regularisation and other works also seem to use heavy dropout when classifying HMDB51 [80].

5.6 Conclusion and Discussion

It was interesting to us that flow seemed to give better predictions than RGB frames. Perhaps one of the reasons for this was that the environment in which I recorded the actions was really unknown to the classifier, or that RGB classifier relies too much on queues from the environment to correctly classify actions.

I succeeded in constructing a valid framework to deploy quite generally learning algorithms within a ROS system. I also managed to adapt the TSN classifier to a real-time implementation (with varying delays, depending on whether a long window, instant results or whole action classification was chosen) using the previous knowledge from skeleton classifiers to enable good classification (79.55% accuracy on the HMDB51 set using N=50, topic /action_fw) without relying on knowing when exactly an action finishes.

I was interested in the fact that responses for HMDB14 were quite similar to the ones from HMDB51. These were quite similar to the responses from the whole set, which is not necessarily the case: as I am dealing with a smaller set, one would expect it to be easier to classify. This effect in turn could be counterbalanced by having a smaller set concentrating more similar tasks (which is certainly the case

here, due to our interest in indoor only, single person tasks, which most likely offer lower quality visual cues), or it shows that the "resolution" we have for binning tasks is such that any grouping may be possible. This effect interests me because it may point to ways our implementation may be lacking in training data as well as ways in which the model could be improved to get more accurate descriptors (and not just linearly independent noise that would cause our model to overfit).

However, it seems as if the results obtained do not translate well into a real-world, in the wild, activity classifier. The optimisation of this framework is a laborious endeavour and issues with generalisation are reported in the pertaining literature [75]. One of the reasons for this may be that the classifier relies on cues from the environment and particular framing, as I noticed upon visual inspection of the HMDB51 dataset that certain actions such as smiling, tended to be mostly close-ups, which was not the case from data from MYSET.

My aim with the structure I chosen was to have a robot actively data-gathering from long periods and thus provide us with the larger amount of data that I believe necessary to train a proper activity classifier, as my chosen dataset, the HMDB51, most likely does not have sufficient examples (on average only 8 repetitions of each action) for this to be possible.

While the training of a neural network in a transfer learning style was not adequate, other methods such as random-trees should have given a moderately good estimate of what the lower bound of a functioning activity recognition classifier's accuracy should be under the conditions I presented it. The moderate results I achieved seem to indicate that I need to improve upon these bases (larger datasets, more descriptive models), rather than trying to get the optimal structure of the algorithm as it is.

I presented a real-time classifier implementation of the TSN algorithm completely built under ROS that allows multiple machines and algorithms to be used simultaneously, achieving a near state of the art performance for HAR (79.5% accuracy on the HMDB51 dataset), successfully replicating results from the literature.

For generalisation and real deployment in robotic platforms, however, more work is necessary to validate the approach attempted in this particular study case. I believe to have validated the infra-structure used so as to facilitate immensely future works with more descriptive models, i.e. better neural network structures, and that allow for more relevant data to be captured - in the very likely case that observer positioning and framing of actions play a role in describing adequately HAR.

Chapter 6

Lessons learned and potential improvements

Most of my discussion was done under the pertaining chapters, however some general discussion seems in order as it involves the totality of this work.

The goal with this project was initially to solve falling detection for elderly people in home environments. I hand-crafted a model based approach (see Chapter: 3) to this problem using a known skeleton classifier structure with over 90% accuracy on the TSTv2 dataset(a falling action vs activity of daily livings (ADLs) dataset). Due to the difficulties in testing and validating such a system and the availability of cheap commercial accelerometer based alternatives, the goal was extended to classify actions in general, as such classifier could be more easily validated.

More broadly, it is my current understanding that HAR is a more challenging problem than it seems at first. It seems to rely intensely on abstracted information from the environment and proper statement of the problem is not only hard, but also did not seem to conduct to any groundbreaking work, at least so far as the literature review showed.

Moreover, its uses were for it to work properly would be of huge gain to many different applications. From simpler use cases of those of describing a video, to more ambitious ones such as crime detection or even robotic learning of actions and

its autonomous execution.

This goal seems to be, given the current state of technology, a far one. With the here presented work, at least in its structure and approach, I expect to have contributed to the development of Human Activity Recognition.

6.1 Active Vision

My work suggests that the problem of active vision may be coupled with activity recognition. Only keeping the subject in frame is not enough to reproduce video sequences of the same style as those that are captured by humans with cameras, which influences recognition. This is a particularly important issue as, not only there are different movement styles based on the type of activity being captured, but also differences in zooming and framing that are likely correlated to the type of activity, which likely makes HAR using videos captured by people, i.e. videos from YouTube, an easier problem than classifying activities in the wild (with no zooming or proper framing). Examples from the HMDB51 are 'smile' or 'chew' sequences, that have a clear face focus, while 'jump' or 'walk' usually show whole body poses.

6.2 Skeletons

The most basic problem when attempting HAR based on skeletons, and that was dealing with noise. It is conceivably harder to identify an action, if the range of movement that describes it is with the same order of magnitude of sensor imprecision. This is especially troublesome if this noise is unpredictable and depends on the skeleton's pose and the environment in a way that cannot be easily modelled.

Velocities streams suffered more from this effect, which resulted in very noisy output for the classes under this layer. More accurate joint positions would likely benefit the skeleton classifier.

Second, the work on activity detection by classifying skeleton positions has some limitations as many actions cannot be interpreted as a simple sequence of skeleton

positions. To illustrate this, one may think of a picking action. Depending on where the object is, its size, where the person is and the constraints of the environment, the picking action may present itself as a completely different set of skeleton poses. One may circumvent this by having an enormous set with many different picking actions, however it is difficult to guarantee that every possible picking action will be present in the training set. Moreover it does not account for possible compositions of actions, such as picking something while sitting on a chair, or even laying down (as a mechanic would do while working on a car engine, for instance).

A more accurate description of an action seems to be necessary in order to solve these sorts of problems.

6.3 Deep Learning

I believe I only scratched the surface of the advantages that deep learning can offer us in terms of activity recognition. The fact that many architectures rely heavily upon models used for image recognitionis perhaps a reflection of this.

It may as well be the case that HAR relies on object recognition and I am indirectly using this fact on the proposed network structures. Although there is reason to believe that perhaps those two (object recognition and activity recognition) are very different problems, requiring different NN structures I am unaware of any work that has done so and shown better results than using custom features (the sometimes called "handcrafted features" [25], see Section 1.6.2). It is probable that Deep Learning techniques need to be used in conjunction with recurrence (say RNNs and LSTMs) or more complex models with hidden states. The organization of such a network is not yet known and may require biologically inspiration from neuroscience studies to narrow down its structure.

If, on the other hand, Deep Learning merely requires more data, or concurrent data from multiple modalities (say 3D information, flow information, position of objects and limbs), than we believe we are on the correct path with our proposed modular docker based ROS infrastructure.

6.4 Infrastructure

Much of my effort was put into creating an infra-structure necessary to be able to test the algorithms with which I was working. Problems with replicability gave rise to more commonly scientists publishing not only their papers, but also the code with which those were created. Moreover, I noted that many recent works were developed using available open-source implementations. I believed to have made a correct choice with the ROS implementation, even though it took considerable time to develop, as it allowed deployment of many more different architectures than would be possible if I wasn't using a docker encapsulated, multi-computer infrastructure.

Even with the computing power I had available, frequently I would run into problems with efficiency when deploying real-time computations and having more than one system to do load balancing was certainly beneficial.

Although I did not manage to completely implement my initial idea (of a skeleton classifier in ensemble with a deep-learning classifier), I believe that further work in this direction would have given me positive results, especially considering how the classifiers behaved under my own acquired datasets (Ourset and Myset).

The implementation we used is a natural one and has been used for a number of other applications. The slight complexity it has on the robotics side, due to having multiple components, should be alleviated in the future by creation of more and better adapter layers as this technology solidifies itself. Such adoption should simplify concurrent execution of multiple complex networks and allow for better testing as well as meta-learning [173], which today is a challenge to researchers without access to extensive resources.

6.5 Future prospects for TSN and the Deep Learning Implementations

As this work on activity detection of a robotic platform did span over a varied number of topics, my deep-learning implementation suffered from not being explored

well enough.

As future work, I would hope to re-implement the learning of the last layer of the network using my ROS infrastructure. Literature review indicates that the implementation of heavy drop-out (80-90%) with adequate training times should be sufficient to improve the results we have to match TSNs expected performance. This was not our focus in the beginning of this work and was thus relegated to background as merely an oversight, and is an area which certainly can be improved.

Additionally, I also want to systematically try out different approaches to fusion layers. In the literature not much attention seemed to be testing different modes of fusion, with most employing a weighted average, or on some cases SVM, without visible improvements in performance. I believe additional gains in accuracy can be made by employing a cleverer fusion modality with the use of ensembles (see Section 6.8).

Given the complexity of action recognition, it is expected that only a multi-modal approach will be sufficient to accurately describe actions in a way in which results can be generalized. Our very capable infrastructure (see Chapter 2) can be leveraged and used in its full potential for such tests. For that, object recognition with useful descriptions must play a role. With this objective in mind image classifiers focused on regions such as R-CNN [59] and Facebook's Detectron [60] should also be a useful addition to an ensemble model. Additional useful information may be available by considering object 6D information [71, 185, 123] and even concurrently with CNN skeleton pose data [169].

Additional work needs to be done to consider how would be the best way to take advantage of this information.

6.6 Extending Active Vision

Obtaining data that is as close as possible from the data your system was trained in seems to be a cornerstone of achieving an accurate classifier.

As most current activity datasets were captured by humans, if one wishes to use this kind of data, two main approaches seems to present themselves: (1) emulate as close as possible video capturing as a human would or (2) abstract camera motion and position, as well as other factors present in videos captured by people. Some methods, such as Improved Dense Trajectories (IDT) [175] compensate for camera motion, so using a method based on IDT, such as currently state-of-the-art for the HMDB51, the classifier from Wang et. al. [176], could potentially alleviate this issue.

Another possibility is however, to use data captured "in the wild", so as to have more representative data, or even, make use of my ROS implementation and record data on the very same way that it will be used for classification - full robot loop. The basis for this was already set, it merely depends on having the computers close to a room where the robot could be placed, and acquire data over a long period of time.

6.7 Better Skeleton Classification

I believe that skeleton representations are not only biologically plausible [8], but also seem to have promise in recognition of higher level scene representations (such as action recognition). The continuation of this work should be based on extracting more accurate and useful activity skeleton data. I started on this endeavour with the use of the CMU's OpenPose Caffe based CNN classifier [169]. Some issues involving speed and not having complete skeletons or joint information from the dataset videos prevented the skeleton classifier from being fused with the CNN classifier, but I believe they are solvable by using a cleverer skeleton representation (that allows comparison of incomplete skeletons) with optical-flows to increase performance. I am halfway in this implementation and hope to publish results from this study, upon solving these issues.

So as to have access to larger datasets I believe I can use a combination of RGB-D datasets, skeleton estimation from 2D images [180] and depth estimation from 2D

images [61, 62, 99] and obtain a very large set of actions with skeletons. Some of the hurdles involving skeleton stabilization and missing limbs across frames as well as performance enhancement with optical flows (an extension of section 2.3.1) were already implemented (solved with a similar idea as employed) as it was my initial idea to deploy this algorithm (which was put aside due to time constraints).

Moreover, perhaps a better description of skeleton poses, based on limb prototype motions, such as Sanzari's approach [144] show promise and should be a part of my proposed ensemble approach.

6.8 Ensembles

The idea of increasing the accuracy of algorithms by implementing a set of "weak learners" and combining their output to produce what Kearns and Valiant [81] refer to as group learning, now referred to by "strong learner" [82]. The idea, presented by Kearns as a collection of similar algorithms collectively trained (group learning) to outperform a single algorithm in the concept of boosting, was extended to multiple methodologies and now describes a meta algorithm that can be implemented in multiple ways, but with the same underlying statistical basis, that is, that using algorithms that are weakly correlated to the output one wishes to obtain, but not correlated with each other, enables to produce an Ensemble that has itself a higher accuracy than each of those basis algorithms used in isolation. While this is not always the case and depends on the dataset, what kinds of algorithms are used as a basis for the ensemble, as well as the number of weak algorithm used , the basic idea still holds, with the resulting classifier generally being more accurate than each of the individual classifiers making up the ensemble [118].

Opitz [118] shows that for Arcing [17], Bagging [16] and Boosting methods, that the initial addition of weak learners seem to be more important than having a very large number of learners (e.g. more than 100), with an almost linear fast decrease up to 20 classifiers and a much slower decrease after that - compare to my own attempt at a random tree forest classifier ensemble (See Fig. 5.11) implemented in section 5.5.1.

An additional important idea substantiated by empirical research is that a good ensemble is one that has individual algorithms that are accurate in different parts of the input space.

Moreover, the ensemble idea is also useful when mixing heterogeneous strong algorithms, which was demonstrated among others, by Dzeroski's 2004 paper [39] with the introduction of a meta-learning algorithm for an optimised tree stacking method, showing that with an adequate strategy, it is possible to improve results from simply choosing the best algorithm by cross-validation strategies. It also substantiates the idea further that error correlation between algorithms is inversely correlated to the performance gain.

Perhaps it is worth mentioning the 2009 Netflix competition, which has its awarded winner group Bellkor using a combination of methods (blending) and gradient boosted decision trees(GBDT) an ensemble method to achieve top performance [94]. More currently I can cite "WatsonPaths: Scenario-based Question Answering and Inference over Unstructured Information" from Lally et al., a IBM's Watson architecture, which uses ensembles [100].

I mention ensembles many times in this document, however the only attempt done with an ensemble method used was the random forest classifier, used in order to gauge the expected lower bound of a proper NN in the context of transfer learning with TSN on my own dataset Myset. This implementation did not use the whole potential of ensembles, which was my initial intention and could not be completed due to time constraints.

Ensembles, and in particular stacking, should be carefully tested for fusion layers, as it has the promise of improving recognition even above the recognition accuracy of the most accurate classifier for a particular action. This possibility should motivate the use of many algorithms simultaneously and even justify the implementation of not state-of-the-art algorithms with the hope that overall, I can still boost the performance of the overall system, or perhaps to, at least, cover gaps in other approaches. Moreover, for very particular cases, perhaps a hand-made solution may be

available in the literature and for that, I believe that with my docker encapsulation implementation, such algorithms will be easier to port and interface.

6.9 Improving Infrastructure: Remote Processing

The ideal structure for remote processing that allows sharing of resources without hurting performance was sought, but considerable lag occurred when fast camera motions were performed that created a lot of artefacts in the video feed when using the VPN structure. This most likely occurred because sending new frames with the fast changing background was causing a spike in data throughput that could not be delivered to the networked nodes. Some optimisation may be necessary on what is processed locally and what can be sent for remote processing. Even though these obstacles arose when using a remote processing strategy, I believe these can be overcome just with optimising engineering aspects of my network setup. Those could include using more modern codecs, improve network physical structure (either by requiring a direct WiFi link from robot to ROS network) or splitting tasks that require low-latency video with high-frame rate from other tasks that can handle more delay and perhaps having some computationally heavy processes run in the robot after all, but most in the remote ROS network.

I believe that possibly the dense flow computations and should be performed on the robot (say by just adding a JETSON[1] connected to the robot via Gigabit Ethernet), while object recognition, face-tracking or processing of action sequences can still be performed with deferred, i.e. remote, processing. This way we perform complicated image analysis with much lower rates (say 1-2 fps) and interleave the unknown frames with estimations from optical flow. With adequate time-stamping and synchronization of nodes in the network there is no reason to assume this system would not work. There is good reason to believe that both those types of processing could be beneficial to optimise learning strategies (see section 6.9.1). I envision that for a

[1]Lucas-Kanade optical flow is not that computationally intensive, with its computational load adjustable by changing image resolution. It is also highly parallelizable, but it needs to be computed with the lowest latency possible. Dividing computations here is a reasonable compromise if adequate bandwidth is available.

simple camera pan-tilt, background out of the field of view of the camera could be stored and only frame differences could be sent over the network which should allow for sufficient speed-up to keep the image feed adequate even with fast panning or tilting motions. Most importantly is to implement adequate compression without reasonable increase in lag (i.e. prevent caching), as delays can be hard to deal with (fast tracking control systems will easily become unstable) and would force too much processing power to be located within the robot, fact that I hope to avoid. The best structure that allows this is possibly very task dependent and deserving of future research and engineering efforts.

6.9.1 Docker Improvements

As mentioned in Section 2.2, a more elegant version of my docker deployment would perhaps use both docker compose and docker swarm, to facilitate use and to do more advanced load-balancing, for more complex future applications.

There is likely additional gain in using such a networked, multi-system infrastructure for learning algorithms, as the sharing of resources could allow for a swarm to learn a task faster than a single agent could. In the case of activity detection, multiple agents observing the same scene (with different viewpoints, light exposure and camera movements), is likely to provide a natural dataset augmentation. Additionally, multiple systems deployed at different locations might have exposure to a larger range of actions and by sharing their learning algorithms, could learn from experiences they themselves, never experienced. Another more general use case would be for reinforcement learning on robots trying to perform similar tasks, where you would also get hopefully faster learning, by sharing experiences. This encapsulation should also enable for meta-learning strategies to be attempted; if CUDA capabilities are not needed, with the configuration being abstracted, this structure should still allow for a simpler clustering deployment on academic settings. Currently commercial products exist to automatically deploy an adequate classifier for a particular dataset, but as these suits are proprietary (such as those provided by Microsoft AZURE or IBM's Watson), results obtained are not adequate for scientific

studies. I can envision a simple tree search algorithm that should allow for a similar functionality, e.g. start with Clustering, proceed to single hidden layer NN with varying number of nodes, then KNN, SVMs with various kernels, GMMs, K-Means systematically; if it has order, proceed to recurrent types, such as RNNs, MTRNNs, LSTMs and so on. Base on recent research, this systematic should be the starting point for any study on any particular dataset.

6.10 Beyond the basics: Datasets May Not Be Enough

There might be limitations with studies dependent on datasets I am just slowly uncovering as learning algorithms become more sophisticated and more complex tasks are attempted [166]. Activity recognition seems to be a complex problem and it is easy to envision that perhaps no dataset will ever be complete enough to account for all variations of possible actions. It seems that activity detection lacks a proper basing theory behind it. Simply stating that activities are state changes with common properties is much too broad a definition to guide this approach.

The research indicates that, depending on the type of action, very different symbol groundings may be used. The action of picking, for example, seems to be related with "attaching an object's frame of reference to a frame of reference that is under control of an agent". It does not seem to matter which hand is used, if you use both hands, if you used your mouth (say to get a coffee cup from a table), your elbow or an instrument, such as tongs. It also does not matter the position in which other limbs are or anything else. The complete description of this action using a dataset seems would be very inefficient and even ideally would reach only a plateau of accuracy, after more common picking actions were presented.

I believe that a model used to describe actions should have representativity for this kind of structure (in the picking example, attach a frame of reference) and that a HAR algorithm will thus need to infer the intention (or planning) consistent with that of attaching a frame of reference, to classify this action correctly. More research

needs to be done so as to describe this problem generally, as not for all actions this descriptive grounding is evident. Actions such as open, or close, seem to be, in this sense, very complex, as they would depend on an understanding of the topology of different objects (as in open a door, or open a bag seem to be almost different actions).

Perhaps the act of trying to replicate an action that is learned is necessary to generate the adequate representation of the action and merely observing it, is not sufficient, which would point towards a merger of activity detection with reinforcement learning strategies. Interesting work has been done that integrates planning and visual feedback under the field of execution monitoring [131], with a robotic implementation of Google's Deepmind paper "Asynchronous Methods for Deep Reinforcement Learning" from Mnih et al. [112] [2]. This implementation, however, uses a set of predefined actions. It would be highly advantageous to close the loop and have learned actions be able to be performed by a robotic agent and it is likely that both execution and recognition could be improved by having a closed loop system.

I hope that with an implementation of encapsulated docker containers for CNN networks, that the deployment of such testing scenarios would be expedited and that more time can be spent testing pertaining science, instead of setting-up the software for robotic platform.

[2]Based on the Torch implementation of this algorithm from `https://github.com/miyosuda/async_deep_reinforce`.

Conclusion

In this work I approach two different promising approaches to Human Activity Recognition (HAR), namely using skeleton classifiers and convolutional neural networks. My work on falling detection [86] was validated under the TST v2 dataset to have over 90% accuracy (see chapter 3); with my activity detection with skeletons under the CAD60 dataset showing over 75% accuracy (see chapter 4) and my TSN, ROS-docker real-time implementation (see chapter 5) showing also over 75% accuracy on the HMDB51 dataset.

I also present a software infrastructure (see Chapter 2) proposal on ROS network, multi-node system (see section 2.1), capable of deploying multiple CUDA nodes and use several different networks simultaneously, to train and test both on datasets and on real robot data, the accuracy of my classifiers. I also implemented an active vision system so as to obtain proper video input with a moving subject (see section 2.3), along-side an approach to optimise computation load of a CNN to get objects position by using optical-flow (see section: 2.3.1).

The proposed classifiers however, did not perform as well as expected considering my validation strategies, but especially on acquired data. I considered some approaches to improve on those results (see chapter: 6) by both exploring the capabilities of my ROS-Docker network implementation: in allowing deployment of multiple classifiers at once (ensembles), make use of the clustered structure to do parameter and network structure optimisation (meta-learning) and speeding up deployment of new classifiers (by using containerized images whenever available) as well on ways to extend the work on each classifier structure attempted.

On the question of how much did I achieve from my planned contribution to science from Section 1.7, perhaps the item I went the most far was Item 1, where I have an infrastructure that can deploy generic algorithms in a fairly clean way, if code is provided. Item 2 was implemented partially, resulting in my conference publication [86], however it lacked a proper robot implementation as it was considered difficult to test, as it is uncertain how representative are staged falls into a mattress in a lab environment, when compared to real falls in a cluttered home environment. For Item 3, I moved a bit further with 2 different implementations, one using classic ML techniques another one using more modern CNN. As for Item 4, I believed to have done an appropriate job, gathering new data that was challenging for the classifiers, a validation strategy which is, perhaps as good as cross-dataset validation, if not better, as it would give a good estimate of how well they would perform in a more realistic deployment.

While still being short from state of the art classifiers, I still believe there is a contribution to science in this work, namely in establishing a systematic way to deploy and train those algorithms under a challenging problem. We believe that our makeshift ROSnvidia-docker cluster should inspire more complex architectures and hopefully a group will be established to curate such type of implementation. The adoption of these measures should streamline the study of classifiers and that of action recognition, by allowing more focus to be put on the task of creating a novel classifier structure, than on other perfunctory tasks, such as solving dependencies or trying to replicate code from often unclear paper descriptions. Containerised deployments make these problems obsolete, while adding only a slight overhead in terms of initial complexity. Performance and dependency issues were a serious concern and this work would not have gone as far as it has without containerized deployment. As an example, the sale of GitHub to Microsoft changed ROS prefixes and one of my implementations would not compile any more after the latest updates were applied. If I hadn't implemented it in Docker and could just load the saved old image, I would have no choice but to solve this issue. It may seem like a simple occurrence, but those little snags add up and slow down research.

I hope to help future researchers in this topic, by providing not only the ideas on how to make easily shareable classifiers, but also providing the source code of the whole implementations presented here in full in GitHub under my personal repositories. If anything it provides a base level that can be used as a frame of comparison and hopefully easily extended and improved. It is not expected that a host will be able to carry around all the computational power it needs to execute many possible computations required by some of the current state-of-the-art learning algorithms. This fact, combined with the knowledge that environment does not change very fast, should allow us to share resources an implement decentralized deferred processing nodes using the tools exposed in this thesis. This area has a lot of possibilities to be investigated, in particular when it comes to using multiple systems concurrently in cooperation. Unfortunately, getting to the stage where those ideas can be tested can be quite arduous as we demonstrated with our explorations, but surely the structure and approach used here should serve as an adequate stepping stone for future works done in this area.

Appendix A

Additional materials for GWR based activity detection

A.1 Joint enumeration

The joint enumeration for the Kinect v1 (used in the CAD60 dataset) consists of 15 joints as can be seen in table A.1.

The joint enumeration for the Kinect for Windows (used in the Ourset dataset) consists of 20 joints as can be seen in table A.2.

The joint enumeration for the Kinect v2 (used in the tstv2 dataset) consists of 25 joints as can be seen in table A.3.

So as to be able to train one dataset and test on another (or even combine sets) we used a simple conversion table based on the text description of what each joint should be. As there are 3 different sets of skeletons being used, 6 different conversions were possible, but we will describe here (see Table) the one that was necessary to convert skeletons from 20 joints to 15 joints (Kinect for Windows skeletons into Kinect v1 skeleton definitions).

Ideally we would use a validated procedure (either from literature or provided by Microsoft) to convert those skeleton definitions or have data sample skeletons of the same subject with the same pose collected with the 3 sensors so that a better

Table A.1: Joint enumeration for Kinect v1.

HEAD	1
NECK	2
TORSO	3
LEFT_SHOULDER	4
LEFT_ELBOW	5
RIGHT_SHOULDER	6
RIGHT_ELBOW	7
LEFT_HIP	8
LEFT_KNEE	9
RIGHT_HIP	10
RIGHT_KNEE	11
LEFT_HAND	12
RIGHT_HAND	13
LEFT_FOOT	14
RIGHT_FOOT	15

Table A.2: Joint enumeration for Kinect for Windows.

hip_center	1
spine	2
shoulder_center	3
head	4
shoulder_left	5
elbow_left	6
wrist_left	7
hand_left	8
shoulder_right	9
elbow_right	10
wrist_right	11
hand_right	12
hip_left	13
knee_left	14
ankle_left	15
foot_left	16
hip_right	17
knee_right	18
ankle_right	19
foot_right	20

Table A.3: Joint enumeration for Kinect v2.

SpineBase	1
SpineMid	2
Neck	3
Head	4
ShoulderLeft	5
ElbowLeft	6
WristLeft	7
HandLeft	8
ShoulderRight	9
ElbowRight	10
WristRight	11
HandRight	12
HipLeft	13
KneeLeft	14
AnkleLeft	15
FootLeft	16
HipRight	17
KneeRight	18
AnkleRight	19
FootRight	20
SpineShoulder	21
HandTipLeft	22
ThumbLeft	23
HandTipRight	24
ThumbRight	25

Table A.4: Joint conversion table of Kinect for Windows skeletons into for Kinect v1 skeletons.

HEAD	$head$
NECK	$spine$
TORSO	$shoulder\_center$
LEFT_SHOULDER	$shoulder\_left$
LEFT_ELBOW	$elbow\_left$
RIGHT_SHOULDER	$shoulder\_right$
RIGHT_ELBOW	$elbow\_right$
LEFT_HIP	hip_left
LEFT_KNEE	$knee\_left$
RIGHT_HIP	hip_right
RIGHT_KNEE	$knee\_right$
LEFT_HAND	$(wrist\_left + hand\_left)/2$
RIGHT_HAND	$(wrist\_right + hand\_right)/2$
LEFT_FOOT	$(ankle\_left + foot\_left)/2$
RIGHT_FOOT	$(ankle\_right + foot\_right)/2$

conversion strategy could be tried (eg. multivariate regression or a neural network), but such data was not easily found. So as to have some baseline we proposed to simply use a table (see Table A.4) and average the results of some redundant joints (as the skeleton we captured had more information than the one on which we tested). This procedure was visually inspected and it seemed to output reasonable usable and comparable skeletons, however we cannot rule out that this imprecise matching is a cause of classifier mismatches.

A.2 Additional results skeleton action classifier

Table A.5: Precision and recall of the best-found algorithm (1NN, centred and mirrored skeletons) in the different environments of CAD-60 for all subjects combined. ± signs indicate 1 standard deviation considering leave one subject out (LOO) cross-validation strategy.

		"New-person"	
Location	Activity	Precision	Recall
Bathroom	Brushing teeth	96.46% ± 4.29%	93.11% ± 6.84%
	Rinsing mouth	87.68% ± 18.55%	100.00% ± 0.00%
	Wearing contact lens	99.11% ± 0.61%	88.82% ± 11.67%
	Average	94.42% ± 11.18%	93.98% ± 8.55%
Bedroom	Talking on phone	63.12% ± 46.84%	60.44% ± 41.24%
	Drinking water	55.47% ± 39.08%	65.81% ± 45.57%
	Opening pill container	98.83% ± 2.34%	77.85% ± 39.14%
	Average	72.47% ± 37.50%	68.03% ± 38.80%
Kitchen	Cooking-chopping	96.86% ± 1.94%	75.97% ± 18.23%
	Cooking-stirring	56.35% ± 39.66%	81.43% ± 29.14%
	Drinking water	73.41% ± 48.97%	74.46% ± 49.65%
	Opening pill container	98.83% ± 2.34%	81.06% ± 32.72%
	Average	81.36% ± 33.54%	78.23% ± 30.88%
Living room	Talking on phone	63.12% ± 46.84%	60.44% ± 41.24%
	Drinking water	81.28% ± 17.82%	79.30% ± 23.43%
	Talking on couch	100.00% ± 0.00%	99.42% ± 1.17%
	Relaxing on couch	100.00% ± 0.00%	100.00% ± 0.00%
	Average	86.10% ± 27.43%	84.79% ± 27.11%
Office	Talking on phone	63.12% ± 46.84%	60.44% ± 41.24%
	Writing on whiteboard	89.64% ± 20.23%	100.00% ± 0.00%
	Drinking water	82.06% ± 18.78%	76.45% ± 26.94%
	Working on computer	100.00% ± 0.00%	100.00% ± 0.00%
	Average	83.71% ± 28.02%	84.22% ± 28.02%
	Global average	94.42% ± 11.18%	81.95% ± 28.82%

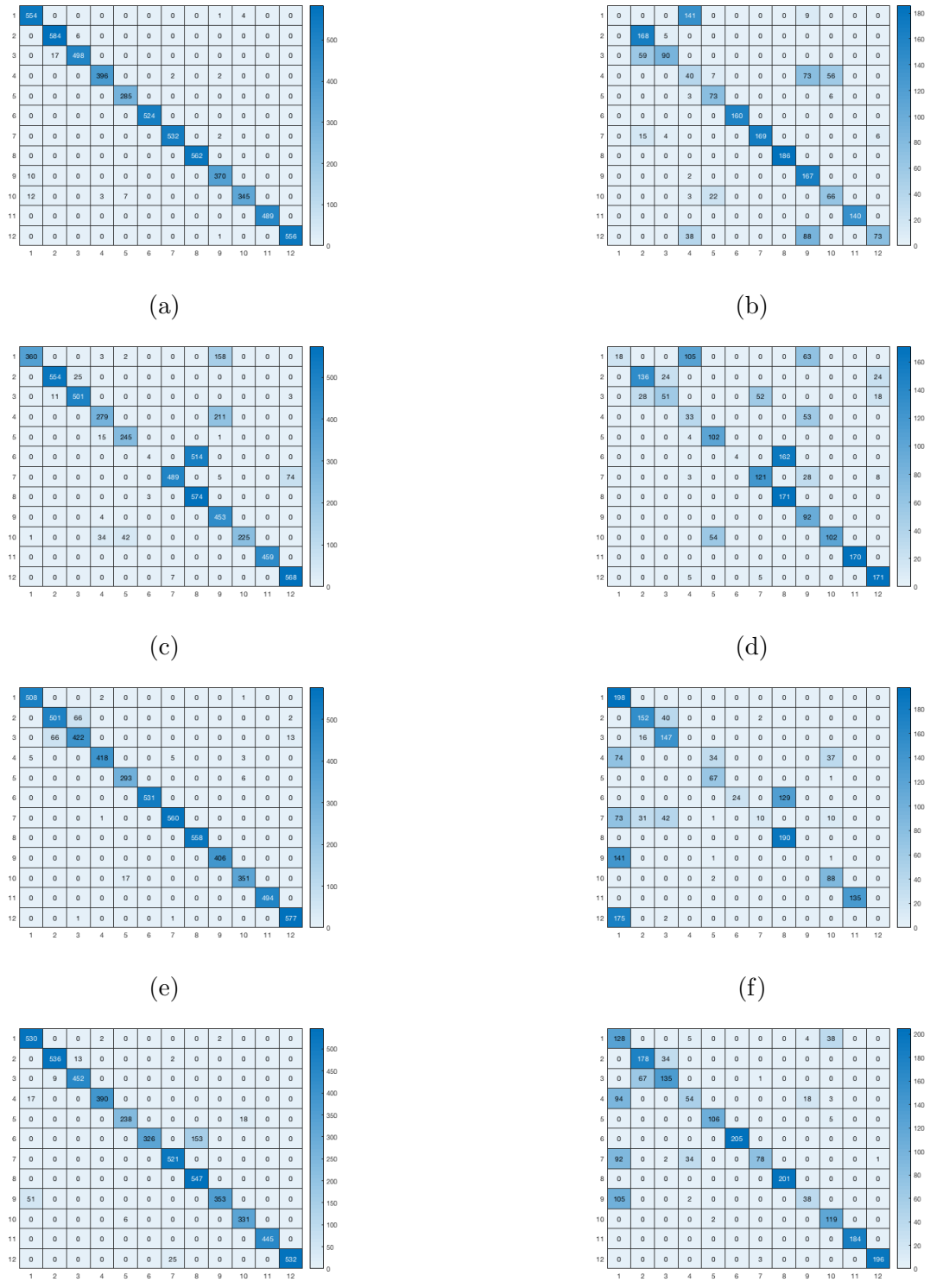


Figure A.1: Confusion matrix for training (a, c, e and g) and validation (b, d, f and h) for subjects 1,2,3 and 4 respectively being assigned to validation set. Labels are the same as in Fig. 4.5

Appendix B

Diagram of the BN-Inception network[1]

I've used the provided Caffe python script draw_net.py and chose to divide it in 10 pages so as to be able to show at least the overall connectivity. The diagram is printed top-to-bottom and shows the validation and training version of the RGB network with its initial full convolution layers (figure B.2, top), followed by 10 inception modules (figures B.2 until Fig. B.11 and its fully connected layer and output (figure B.11). The flow network has the same overall structure. The complete network structure is available in `https://imgur.com/a/sRdR1BN` and can also be accessed with the QR-code in Fig. B.1.

[1]The BN-Inception network is quite large and it is a challenge to display it in a sensible way within the constrains of A4 size papers. We understand the difficulty in reading the network structure from the given pictures. It is perhaps worth considering if it is possible to continuously draw network structures upon increasing complexity.

Figure B.1: QR-code for imgur link of the complete structure of TSN image classification network branch.

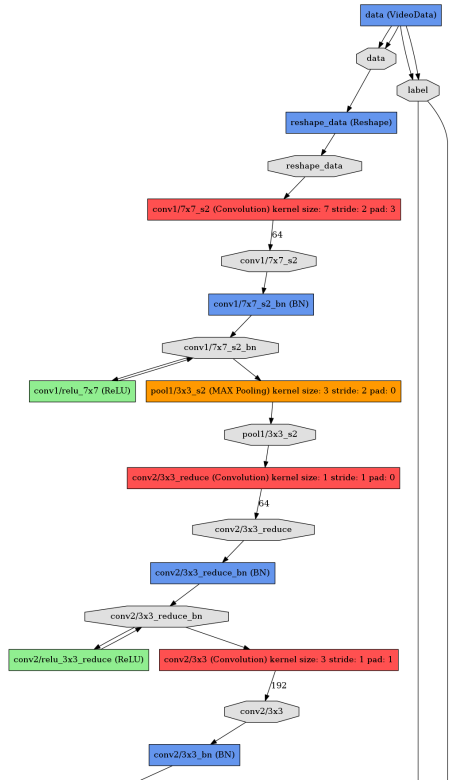


Figure B.2: Complete diagram of the BN-Inception network used in Temporal Segments Networks classifier architecture. Page 1/10

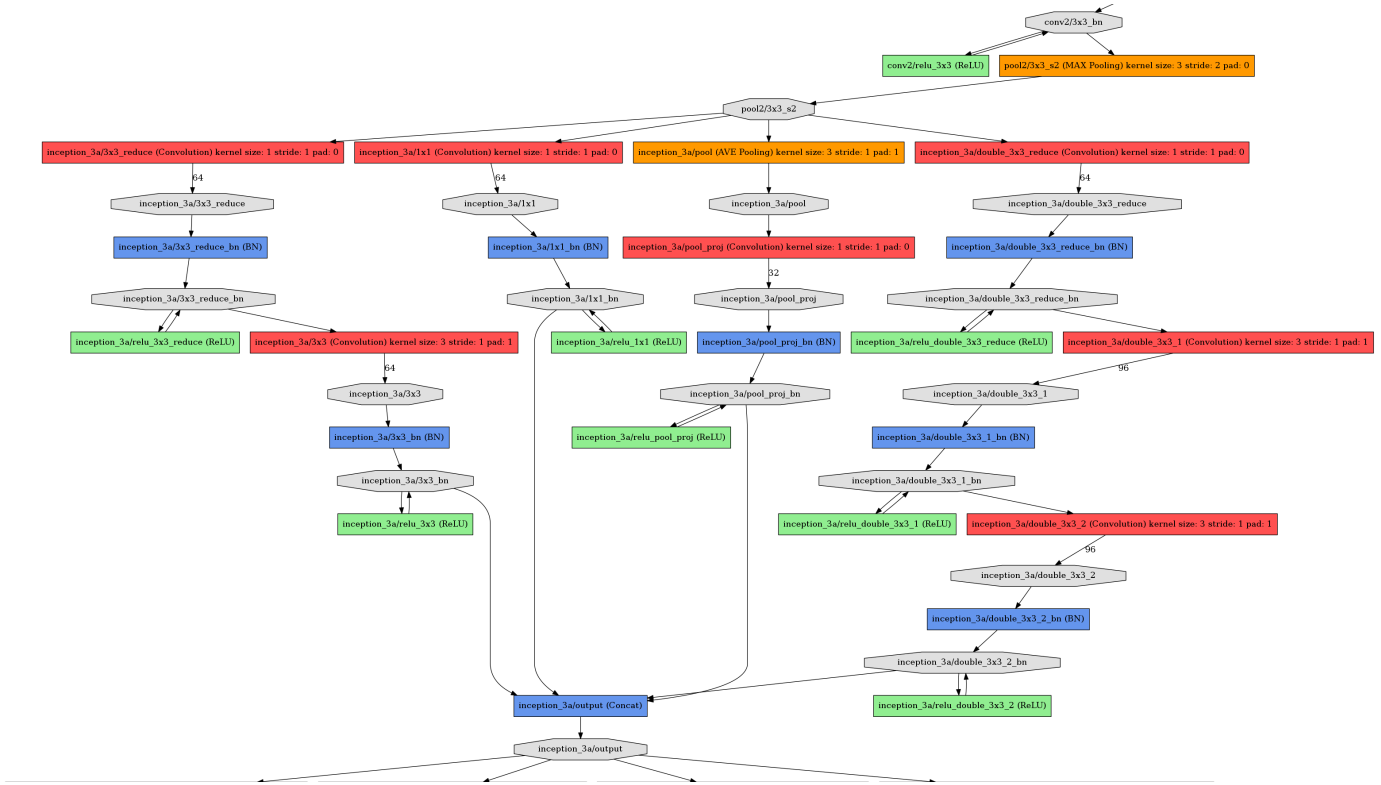


Figure B.3: Complete diagram of the BN-Inception network used in Temporal Segments Networks classifier architecture. Page 2/10

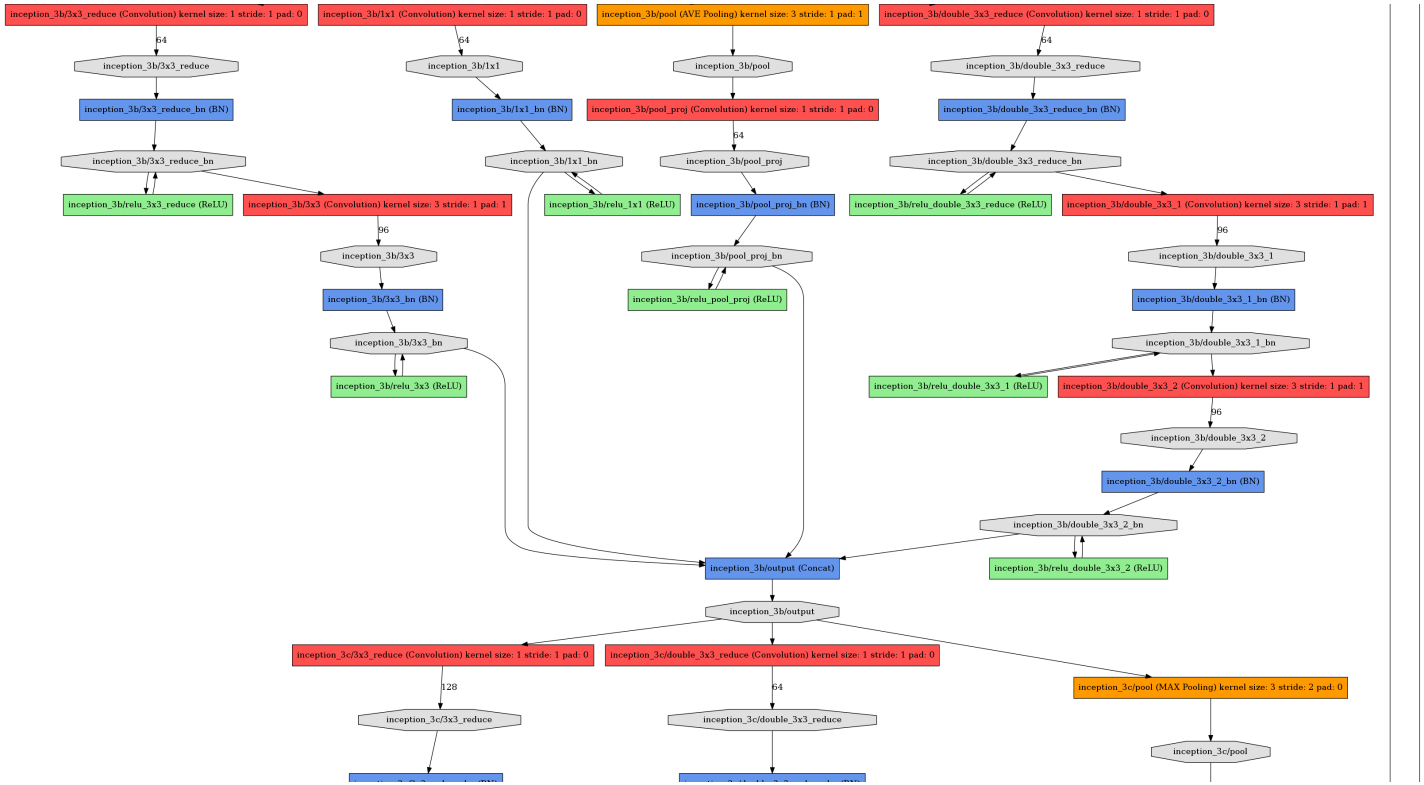


Figure B.4: Complete diagram of the BN-Inception network used in Temporal Segments Networks classifier architecture. Page 3/10

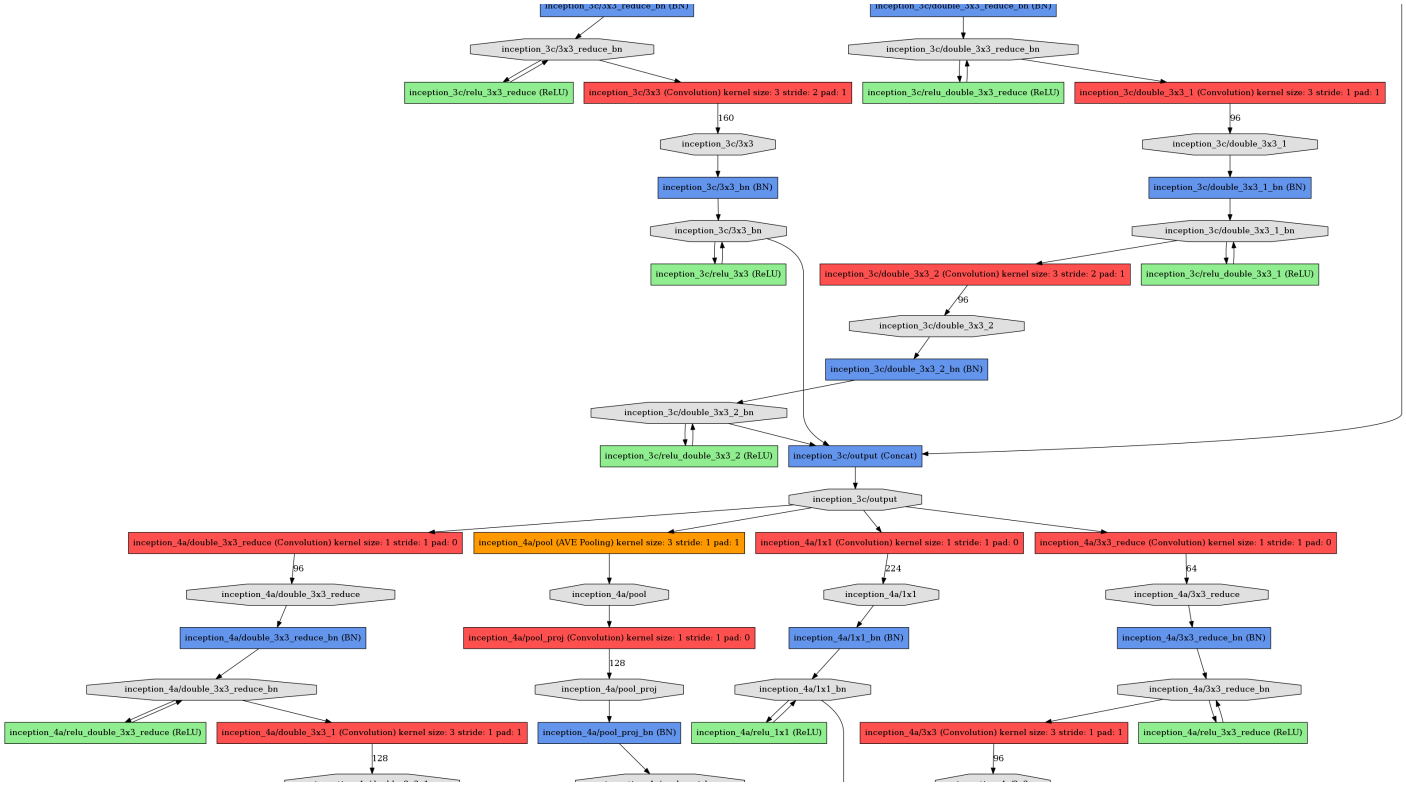


Figure B.5: Complete diagram of the BN-Inception network used in Temporal Segments Networks classifier architecture. Page 4/10

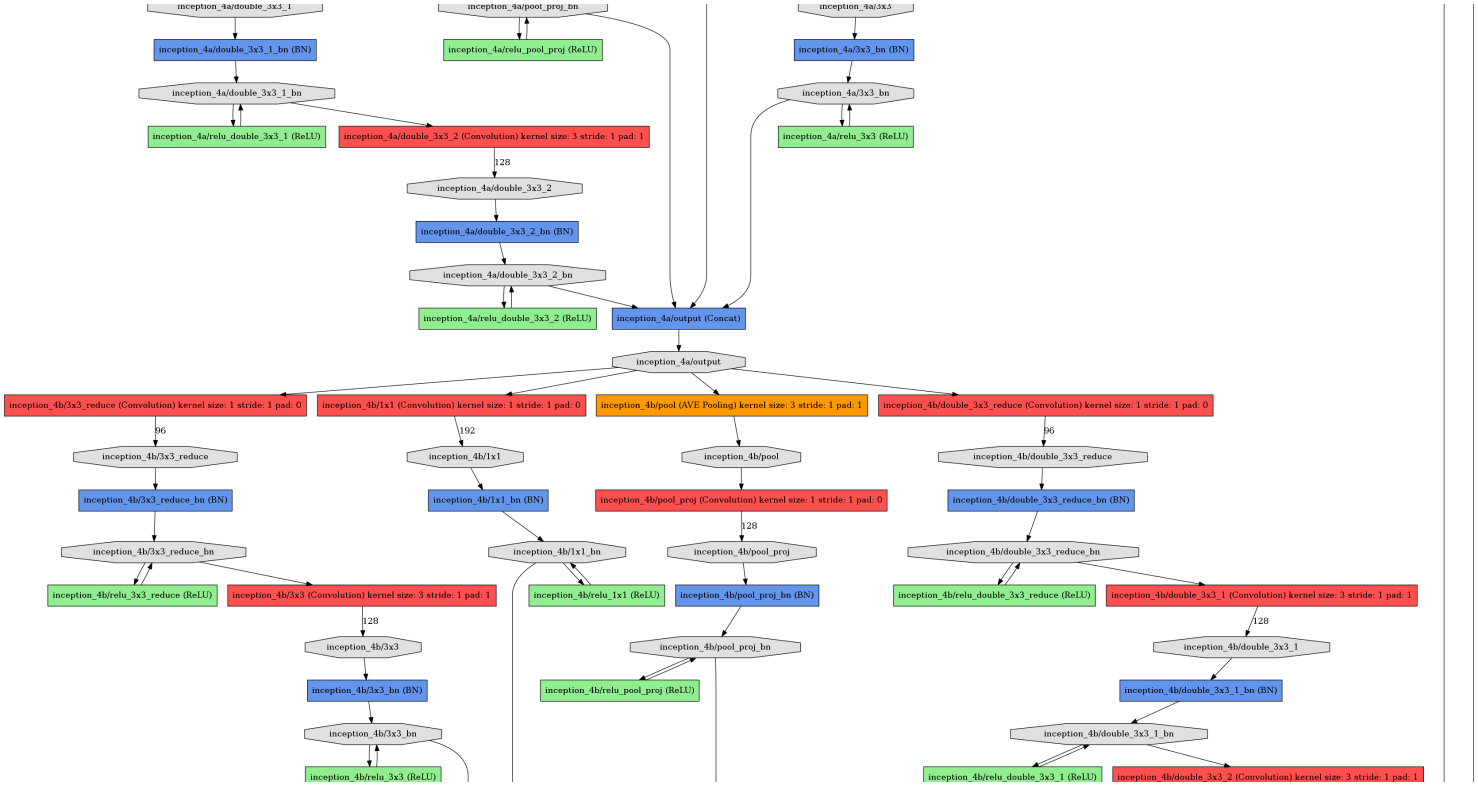


Figure B.6: Complete diagram of the BN-Inception network used in Temporal Segments Networks classifier architecture. Page 5/10

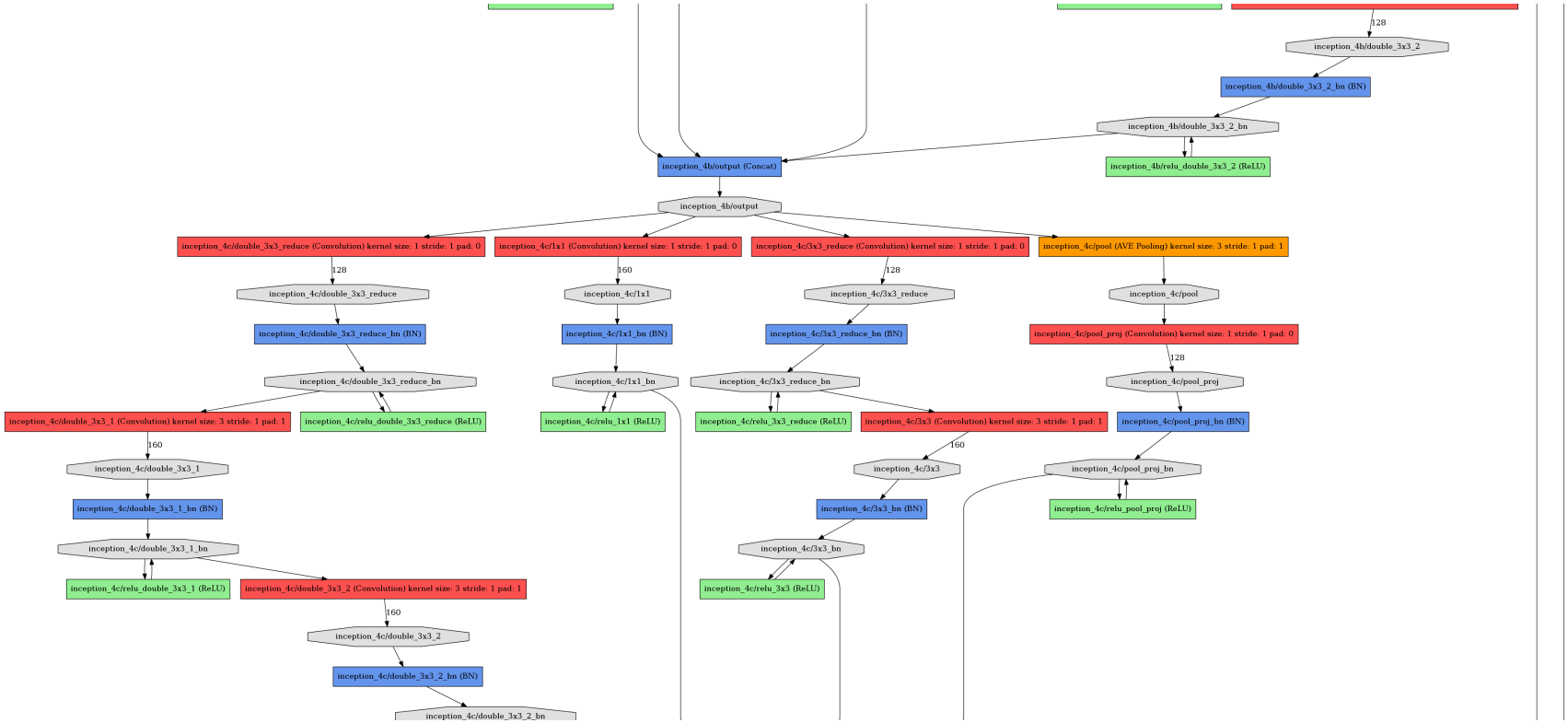


Figure B.7: Complete diagram of the BN-Inception network used in Temporal Segments Networks classifier architecture. Page 6/10

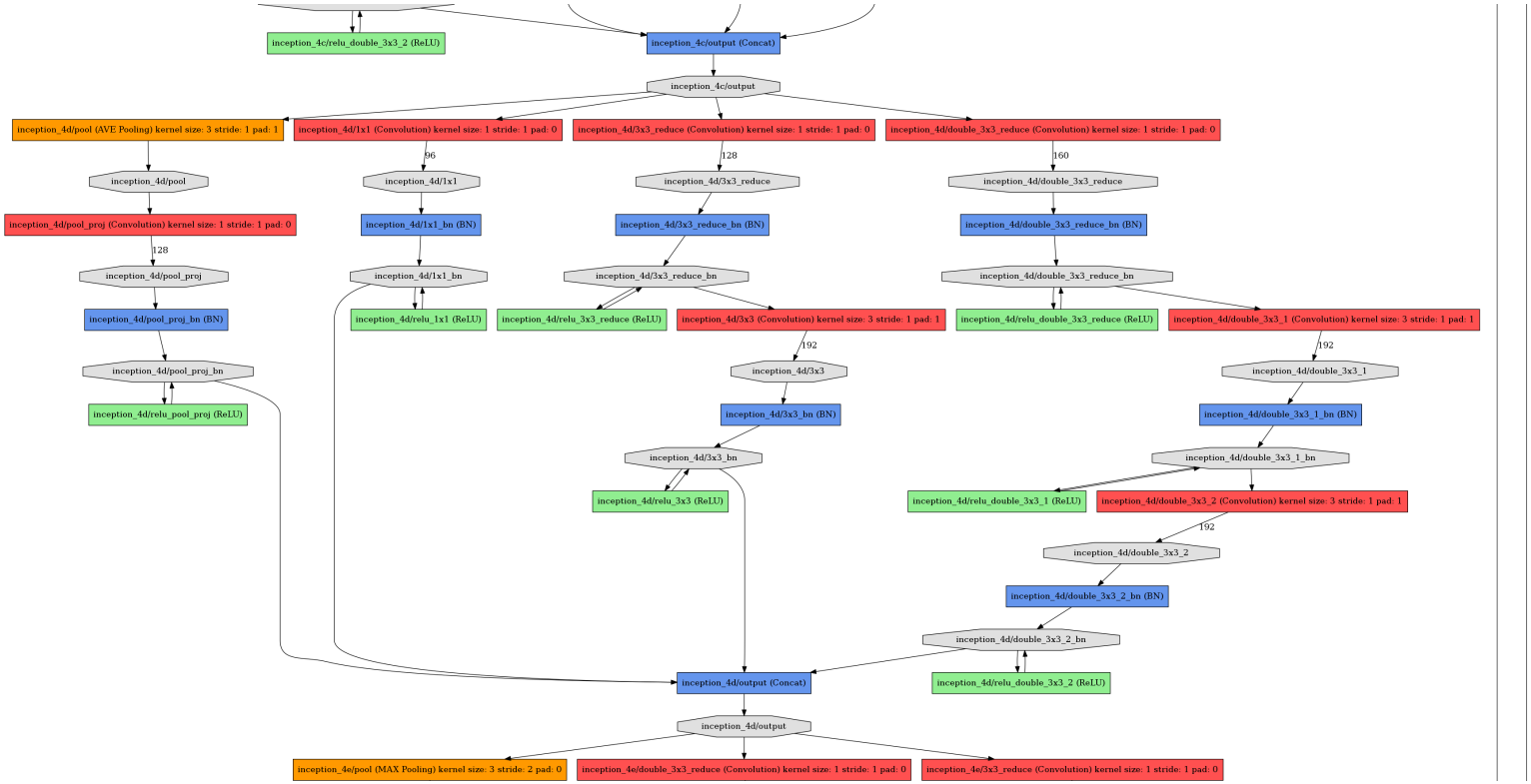


Figure B.8: Complete diagram of the BN-Inception network used in Temporal Segments Networks classifier architecture. Page 7/10

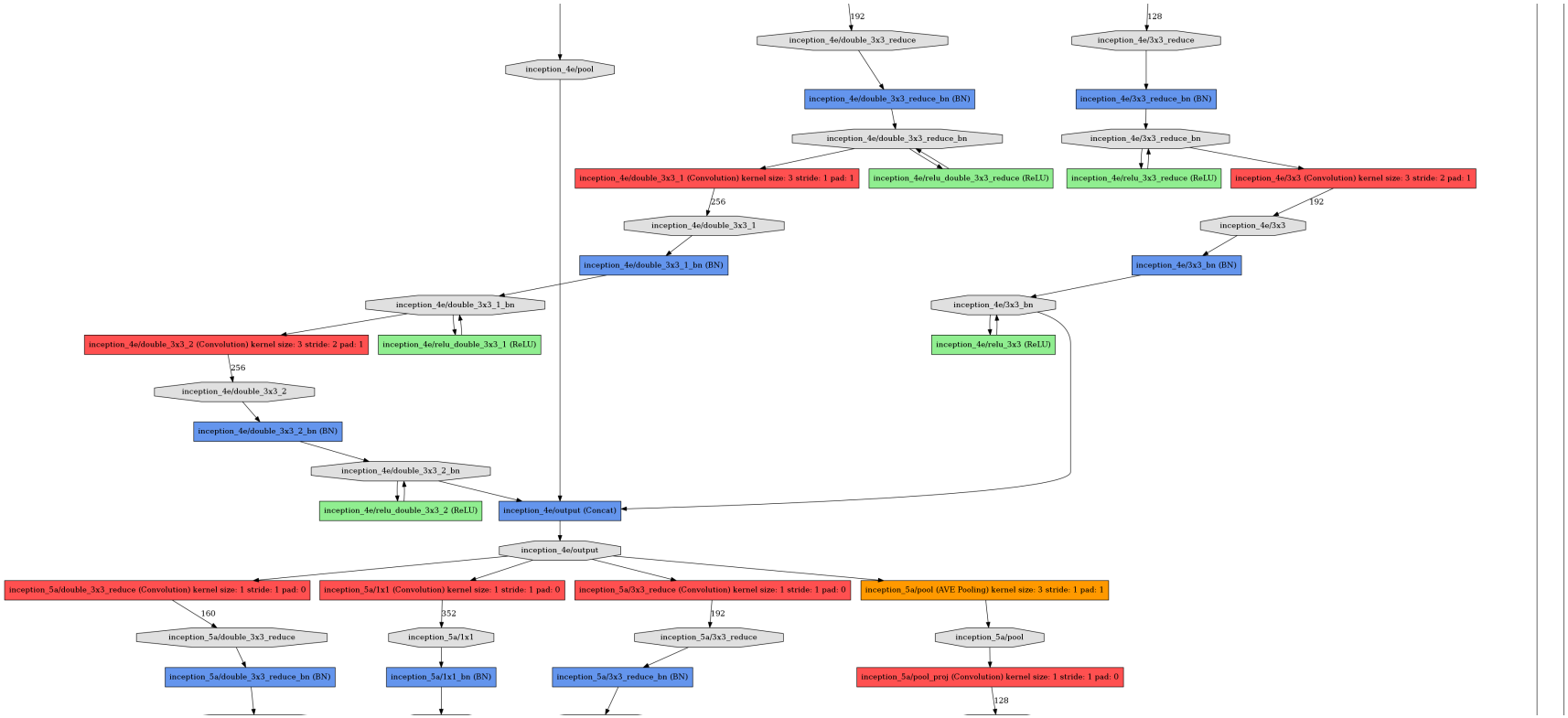


Figure B.9: Complete diagram of the BN-Inception network used in Temporal Segments Networks classifier architecture. Page 8/10

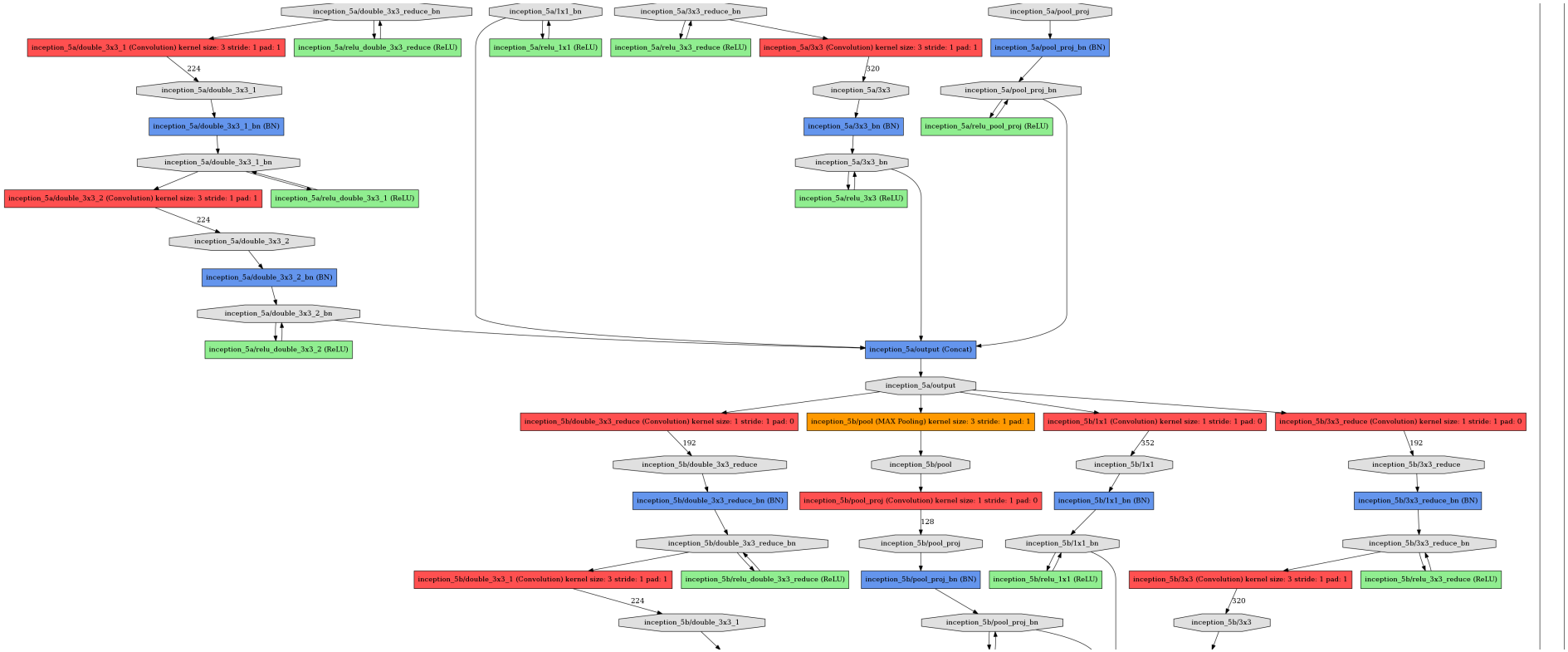


Figure B.10: Complete diagram of the BN-Inception network used in Temporal Segments Networks classifier architecture. Page 9/10

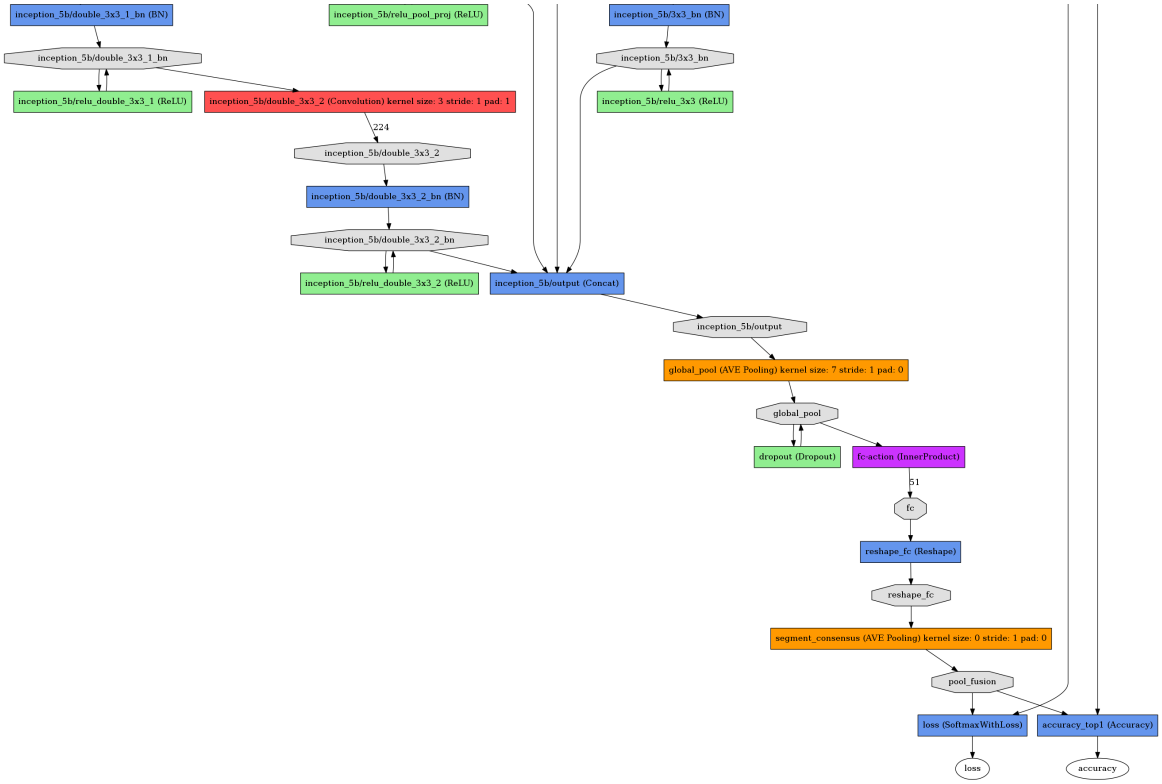


Figure B.11: Complete diagram of the BN-Inception network used in Temporal Segments Networks classifier architecture. Page 10/10

Bibliography

[1] Shawn D. Aaron, Katherine L. Vandemheen, Dean Fergusson, François Maltais, Jean Bourbeau, Roger Goldstein, Meyer Balter, Denis O'Donnell, Andrew McIvor, Sat Sharma, Graham Bishop, John Anthony, Robert Cowie, Stephen Field, Andrew Hirsch, Paul Hernandez, Robert Rivington, Jeremy Road, Victor Hoffstein, Richard Hodder, Darcy Marciniuk, David McCormack, George Fox, Gerard Cox, Henry B. Prins, Gordon Ford, Dominique Bleskie, Steve Doucette, Irvin Mayers, Kenneth Chapman, Noe Zamel, and Mark FitzGerald. Tiotropium in Combination with Placebo, Salmeterol, or Fluticasone–Salmeterol for Treatment of Chronic Obstructive Pulmonary DiseaseA Randomized Trial. *Annals of Internal Medicine*, 146(8):545–555, 2007.

[2] Alfred L. Yarbus. *Eye movements and vision.* Plenum Press, New York, NY, USA, 1967.

[3] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356, January 1988.

[4] Ethem Alpaydin. *Introduction to Machine Learning.* MIT Press, 2014. Google-Books-ID: 7f5bBAAAQBAJ.

[5] Diego Raphael Amancio, Cesar Henrique Comin, Dalcimar Casanova, Gonzalo Travieso, Odemir Martinez Bruno, Francisco Aparecido Rodrigues, and Luciano da Fontoura Costa. A Systematic Comparison of Supervised Classifiers. *PLOS ONE*, 9(4):e94137, 24-Apr-2014.

[6] Muzaffer Aslan, Abdulkadir Sengur, Yang Xiao, Haibo Wang, M. Cevdet Ince, and Xin Ma. Shape feature encoding via Fisher Vector for efficient fall de-

tection in depth-videos. *Applied Soft Computing*, 37:1023–1028, December 2015.

[7] ASUS. Xtion — 3D Sensor. https://www.asus.com/3D-Sensor/Xtion/, 2012.

[8] Vladislav Ayzenberg and Stella F. Lourenco. Skeletal descriptions of shape provide unique perceptual information for object recognition. *Sci Rep*, 9(1):1–13, June 2019.

[9] Dana H. Ballard. Animate vision. *Artificial Intelligence*, 48(1):57–86, February 1991.

[10] The World Bank. Population ages 65 and above (% of total population) | Data.

[11] Frederico Belmonte Klein. GWR and GNG Classifier - File Exchange - MATLAB Central, June 2016.

[12] Yoshua Bengio and Yves Grandvalet. No Unbiased Estimator of the Variance of K-Fold Cross-Validation. page 17.

[13] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, Montreal, Quebec, Canada, 2009. ACM Press.

[14] Sarah D. Berry and Ram Miller. Falls: Epidemiology, Pathophysiology, and Relationship to Fracture. *Current osteoporosis reports*, 6(4):149–154, December 2008.

[15] Olivier Bousquet and Andre Elisseeff. Stability and Generalization. page 28.

[16] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[17] Leo Breiman. BIAS, VARIANCE , AND ARCING CLASSIFIERS. page 22, 1996.

[18] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High Accuracy Optical Flow Estimation Based on a Theory for Warping. In Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Tomás Pajdla, and Jiří Matas, editors, *Computer Vision - ECCV 2004*, volume 3024, pages 25–36. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[19] Joao Carreira and Andrew Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. *arXiv:1705.07750 [cs]*, May 2017.

[20] CDC-US. Table 81. Persons with hospital stays in the past year, by selected characteristics: United States, selected years 1997–2016. page 4, 1997.

[21] CDC-US. Number, percent distribution, rate, days of care with average length of stay, and standard error of discharges from short-stay hospitals, by sex and age: United States, 2010. page 3, 2010.

[22] Cerna. Onhand Assistance Watch - Health monitoring smart watch.

[23] Marie Chan, Daniel Estève, Christophe Escriba, and Eric Campo. A review of smart homes—present state and future challenges. *Computer Methods and Programs in Biomedicine*, 91(1):55–81, 2008.

[24] Jose M. Chaquet, Enrique J. Carmona, and Antonio Fernández-Caballero. A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding*, 117(6):633–659, June 2013.

[25] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. *arXiv:1405.3531 [cs]*, May 2014.

[26] Siddhartha Chaudhuri. Image Processing. page 7, 2010.

[27] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, April 1992.

[28] Zhongwei Cheng, Lei Qin, Yituo Ye, Qingming Huang, and Qi Tian. Human Daily Action Analysis with Multi-view and Color-Depth Data. In Andrea Fusiello, Vittorio Murino, and Rita Cucchiara, editors, *Computer Vision – ECCV 2012. Workshops and Demonstrations*, Lecture Notes in Computer Science, pages 52–61. Springer Berlin Heidelberg, 2012.

[29] W. J. Choi, J. M. Wakeling, and S. N. Robinovitch. Kinematic analysis of video-captured falls experienced by older adults in long-term care. *Journal of Biomechanics*, 48(6):911–920, April 2015.

[30] E. Cippitelli, F. Fioranelli, E. Gambi, and S. Spinsante. Radar and RGB-Depth Sensors for Fall Detection: A Review. *IEEE Sensors Journal*, 17(12):3585–3604, June 2017.

[31] Enea Cippitelli, Samuele Gasparrini, Ennio Gambi, and Susanna Spinsante. A human activity recognition system using skeleton data from RGBD sensors. *Computational Intelligence and Neuroscience*, 2016:1–14, 2016.

[32] Ronan Collobert and Jason Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. page 8, 2008.

[33] Microsoft Corporation. JointType enumeration, 2016. Available at `https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx`, accessed in 2016-05-14.

[34] Microsoft Corporation. Kinect for Windows Sensor Components and Specifications, 2017.

[35] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[36] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human Detection Using Oriented Histograms of Flow and Appearance. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3952, pages 428–441. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[37] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. page 8, 2009.

[38] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior Recognition via Sparse Spatio-Temporal Features. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72, Beijing, China, 2005. IEEE.

[39] Saso Dzeroski and Bernard Zenko. Is Combining Classifiers Better than Selecting the Best One? In *Machine Learning*, pages 255–273. Morgan Kaufmann, 2004.

[40] Jeffrey L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, July 1993.

[41] Hiro Farabi, Aziz Rezapour, Reza Jahangiri, Abdosaleh Jafari, Asma Rashki Kemmak, and Shima Nikjoo. Economic evaluation of the utilization of telemedicine for patients with cardiovascular disease: A systematic review. *Heart Fail Rev*, November 2019.

[42] D. R. Faria, C. Premebida, and U. Nunes. A probabilistic approach for human everyday activities recognition using body motion from RGB-d images. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 732–737, 2014.

[43] Gunnar Farnebäck. Two-Frame Motion Estimation Based on Polynomial Expansion. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Josef Bigun, and Tomas Gustavsson, editors, *Image Analysis*, volume 2749, pages 363–370. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[44] Juan Fasola and Maja Mataric. A Socially Assistive Robot Exercise Coach for the Elderly. *Journal of Human-Robot Interaction*, 2(2), June 2013.

[45] M. A. Fernández-Granero, D. Sánchez-Morillo, A. León-Jiménez, and L. F. Crespo. Automatic prediction of chronic obstructive pulmonary disease ex-

acerbations through home telemonitoring of symptoms. *Bio-Medical Materials and Engineering*, 24(6):3825–3832, 2014.

[46] Miguel Angel Fernandez-Granero, Daniel Sanchez-Morillo, and Antonio Leon-Jimenez. Computerised analysis of telemonitored respiratory sounds for predicting acute exacerbations of COPD. *Sensors (Basel, Switzerland)*, 15(10):26978–26996, 2015.

[47] David Fischinger, Peter Einramhof, Konstantinos Papoutsakis, Walter Wohlkinger, Peter Mayer, Paul Panek, Stefan Hofmann, Tobias Koertner, Astrid Weiss, Antonis Argyros, and Markus Vincze. Hobbit, a care robot supporting independent living at home: First prototype and lessons learned. *Robotics and Autonomous Systems*, 2016.

[48] Gerd Flodgren, Antoine Rachas, Andrew J Farmer, Marco Inzitari, and Sasha Shepperd. Interactive telemedicine: effects on professional practice and health care outcomes. In *Cochrane Database of Systematic Reviews*. John Wiley & Sons, Ltd, September 2015.

[49] Carolina Dutra Queiroz Flumignan, Aline Pereira da Rocha, Ana Carolina Pereira Nunes Pinto, Keilla Machado Martins Milby, Mayara Rodrigues Batista, Álvaro Nagib Atallah, and Humberto Saconato. What do Cochrane systematic reviews say about telemedicine for healthcare? *Sao Paulo Med J*, 137(2):184–192, July 2019.

[50] Luigi Fontana. Modulating Human Aging and Age-Associated Diseases. *Biochimica et biophysica acta*, 1790(10):1133–1138, October 2009.

[51] Earl S. Ford, Guixiang Zhao, James Tsai, and Chaoyang Li. Low-Risk Lifestyle Behaviors and All-Cause Mortality: Findings From the National Health and Nutrition Examination Survey III Mortality Study. *Am J Public Health*, 101(10):1922–1929, October 2011.

[52] James Freed, Charles Lowe, Gerd Flodgren, Rachel Binks, Kevin Doughty, and Jyrki Kolsi. Telemedicine: Is it really worth it? A perspective from evidence

and experience. *BMJ Health & Care Informatics*, 25(1), January 2018.

[53] Bernd Fritzke. A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, 7:625–632, 1995.

[54] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics*, 36(4):193–202, April 1980.

[55] Shen Furao and Osamu Hasegawa. An incremental network for on-line unsupervised classification and topology learning. *Neural Networks*, 19(1):90–106, 2006.

[56] Samuele Gasparrini, Enea Cippitelli, Ennio Gambi, Susanna Spinsante, Jonas Wåhslén, Ibrahim Orhan, and Thomas Lindh. Proposal and Experimental Evaluation of Fall Detection Solution Based on Wearable and Depth Data Fusion. In Suzana Loshkovska and Saso Koceski, editors, *ICT Innovations 2015*, volume 399 of *Advances in Intelligent Systems and Computing*, pages 99–108. Springer International Publishing, 2016.

[57] Adam Geitgey. Ageitgey/face_recognition, September 2019.

[58] Gillian Ward, Nikki Holliday, Simon Fielden, and Sue Williams. Fall detectors: a review of the literature. *Journal of Assistive Technologies*, 6(3):202–215, September 2012.

[59] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv:1311.2524 [cs]*, November 2013.

[60] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[61] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. *arXiv:1609.03677 [cs, stat]*, September 2016.

[62] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[63] Raj Gupta, Alex Yong-Sang Chia, and Deepu Rajan. Human activities recognition using depth images. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 283–292. ACM, 2013.

[64] Isabelle Guyon and T Bell Laboratories. A scaling law for the validation-set training-set size ratio. page 11, 1997.

[65] R. J. Halbert, J. L. Natoli, A. Gano, E. Badamgarav, A. S. Buist, and D. M. Mannino. Global burden of COPD: systematic review and meta-analysis. *The European Respiratory Journal*, 28(3):523–532, September 2006.

[66] P. Hart. The condensed nearest neighbor rule (corresp.). *IEEE Transactions on Information Theory*, 14(3):515–516, 1968.

[67] Simon S. Haykin and Barry Van Veen. *Signals and systems*. Wiley, New York, 2nd ed edition, 2002.

[68] Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming, Revised Reprint*. Elsevier, June 2012. Google-Books-ID: vfvPrSz7R7QC.

[69] Julian P. T. Higgins and Simon G. Thompson. Quantifying heterogeneity in a meta-analysis. *Statistics in Medicine*, 21(11):1539–1558, June 2002.

[70] Te-Wei Ho, Chun-Ta Huang, Herng-Chia Chiu, Sheng-Yuan Ruan, Yi-Ju Tsai, Chong-Jen Yu, and Feipei Lai. Effectiveness of telemonitoring in patients with chronic obstructive pulmonary disease in taiwan-a randomized controlled trial. *Scientific Reports*, 6, 2016.

[71] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-Driven 6D Object Pose Estimation. page 10.

[72] IEEE Spectrum. Double - ROBOTS: Your Guide to the World of Robotics. https://robots.ieee.org/robots/double/, 2020.

[73] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, February 2015. arXiv: 1502.03167.

[74] Edward C. Jauch, Jeffrey L. Saver, Bart M. Demaerschalk, Pooja Khatri, Paul W. McMullan Jr, Adnan I. Qureshi, Kenneth Rosenfield, Phillip A. Scott, Debbie R. Summers, and David Z. Wang. AHA/ASA guideline. *Stroke*, 2013.

[75] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards Understanding Action Recognition. In *2013 IEEE International Conference on Computer Vision*, pages 3192–3199, December 2013.

[76] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A Biologically Inspired System for Action Recognition. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, 2007. IEEE.

[77] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828, 1978.

[78] Reza Kachouie, Sima Sedighadeli, Rajiv Khosla, and Mei-Tai Chu. Socially Assistive Robots in Elderly Care: A Mixed-Method Systematic Literature Review. *International Journal of Human-Computer Interaction*, 30(5):369–393, May 2014.

[79] T. Kaipainen, M. Tiihonen, S. Hartikainen, and I. Nykänen. Prevalence of risk of malnutrition and associated factors in home care clients. *Journal of Nursing Home Research*, Jour Nursing Home Res 20151:47–51, March 2015.

[80] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-Scale Video Classification with Convolutional Neural Networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, Columbus, OH, USA, June 2014. IEEE.

[81] M. Kearns and L. G. Valiant. Cryptographic Limitations on Learning Boolean Formulae and Finite Automata. In *Proceedings of the Twenty-first Annual*

ACM Symposium on Theory of Computing, STOC '89, pages 433–444, New York, NY, USA, 1989. ACM.

[82] Michael Kearns. Thoughts on Hypothesis Boosting. Machine Learning class project. 1988.

[83] Michael Kearns, Murray Hill, and Dana Ron. Algorithmic Stability and Sanity-Check Bounds for Leave-One-Out Cross-Validation. page 21.

[84] C. D. Kidd and C. Breazeal. Robots at home: Understanding long-term human-robot interaction. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3230–3235, September 2008.

[85] A. Klaeser, M. Marszalek, and C. Schmid. A Spatio-Temporal Descriptor Based on 3D-Gradients. In *Procedings of the British Machine Vision Conference 2008*, pages 99.1–99.10, Leeds, 2008. British Machine Vision Association.

[86] Frederico B. Klein, Karla Štěpánová, and Angelo Cangelosi. Implementation of a modular growing when required neural gas architecture for recognition of falls. In Akira Hirose, Seiichi Ozawa, Kenji Doya, Kazushi Ikeda, Minho Lee, and Derong Liu, editors, *Neural Information Processing*, number 9947 in Lecture Notes in Computer Science, pages 526–534. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-46687-3_58.

[87] Frederico Belmonte Klein and Angelo Cangelosi. Human activity recognition from skeleton poses. *arXiv:1908.08928 [cs]*, August 2019.

[88] Klein, Frederico B. Mysablehats/dense_flow, June 2019.

[89] Natasa Koceska, Saso Koceski, Pierluigi Beomonte Zobel, Vladimir Trajkovik, and Nuno Garcia. A Telemedicine Robot System for Assisted and Independent Living. *Sensors*, 19(4):834, January 2019.

[90] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, Montreal, Quebec, Canada, August 1995. Morgan Kaufmann Publishers Inc.

[91] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[92] Ilias Konsoulas. Unsupervised Learning with Growing Neural Gas (GNG) Neural Network - File Exchange - MATLAB Central, September 2013.

[93] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):14–29, 2016.

[94] Yehuda Koren. The BellKor Solution to the Netflix Grand Prize. page 10, 2009.

[95] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[96] Lars Eric Kroll and Thomas Lampert. Direct costs of inequalities in health care utilization in Germany 1994 to 2009: A top-down projection. *BMC Health Serv Res*, 13(1):271, December 2013.

[97] H Kuehne, H Jhuang E Garrote, T Poggio, T Serre, and Brown University. HMDB: A Large Video Database for Human Motion Recognition. page 8, 2011.

[98] Bogdan Kwolek and Michal Kepski. Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer Methods and Programs in Biomedicine*, 117(3):489–501, December 2014.

[99] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[100] Adam Lally, Sugato Bagchi, Michael A. Barborak, David W. Buchanan, Jennifer Chu-Carroll, David A. Ferrucci, Michael R. Glass, Aditya Kalyanpur,

Erik T. Mueller, J. William Murdock, Siddharth Patwardhan, and John M. Prager. WatsonPaths: Scenario-Based Question Answering and Inference over Unstructured Information. *AI Magazine*, 38(2):59, July 2017.

[101] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Anchorage, AK, USA, June 2008. IEEE.

[102] Traun Leyden. Tleyden/elastic-thought, August 2019.

[103] Yang Liu. Using SVM and Error-Correcting Codes for Multiclass Dialog Act Classification in Meeting Corpus. page 4, 2006.

[104] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[105] Alessandro Manzi, Paolo Dario, and Filippo Cavallo. A Human Activity Recognition System Based on Dynamic Clustering of Skeleton Data. *Sensors (Basel)*, 17(5), May 2017.

[106] Alessandra Marengoni, Bengt Winblad, Anita Karp, and Laura Fratiglioni. Prevalence of Chronic Diseases and Multimorbidity Among the Elderly Population in Sweden. *American Journal of Public Health*, 98(7):1198–1200, July 2008.

[107] Landis Markley. Attitude determination using vector observations and the singular value decomposition. *J. Astronaut. Sci.*, 38, 1987.

[108] M. R. Marques, T. M. Raymundo, and C. S. Santana. Use of sensors systems to monitor the mobility of elderly. *Journal of Physics: Conference Series*, 477(1):012010, 2013.

[109] Stephen Marsland, Jonathan Shapiro, and Ulrich Nehmzow. A self-organising network that grows when required. *Neural Networks*, 15(8):1041–1058, 2002.

[110] Thomas M. Martinetz and Klaus J. Schulten. A "neural gas" network learns topologies. In Teuvo Kohonen, Kai Mäkisara, Olli Simula, and Jari Kangas, editors, *Proceedings of the International Conference on Artificial Neural Networks 1991 \rm (Espoo, Finland)*, pages 397–402. Amsterdam; New York: North-Holland, 1991.

[111] Rob McCarney, James Warner, Steve Iliffe, Robbert van Haselen, Mark Griffin, and Peter Fisher. The Hawthorne Effect: a randomised, controlled trial. *BMC Medical Research Methodology*, 7:30, July 2007.

[112] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *arXiv:1602.01783 [cs]*, February 2016.

[113] Mohamad H. Hassoun. *Fundamentals of Artificial Neural Networks MIT Press*. MIT Press, 1995.

[114] Mohamud A. Verjee. Home Visits and Home-Based Care: A Necessary, Impractical, or Humanitarian Primary Care Service? *J Family Med Prim Care Open Acc*, 3(1), January 2019.

[115] Tomas Möller and John F. Hughes. Efficiently Building a Matrix to Rotate One Vector to Another. *J. Graph. Tools*, 4(4):1–4, December 1999.

[116] Adam Morawiec. *Orientations and Rotations: Computations in Crystallographic Textures*. Springer Science & Business Media, 2004.

[117] Loris Nanni, Stefano Ghidoni, and Sheryl Brahnam. Handcrafted vs. non-handcrafted features for computer vision classification. *Pattern Recognition*, 71:158–172, November 2017.

[118] D. Opitz and R. Maclin. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, 11:169–198, August 1999.

[119] Filippo Palumbo, Jonas Ullberg, Ales Štimec, Francesco Furfari, Lars Karlsson, and Silvia Coradeschi. Sensor Network Infrastructure for a Home Care

Monitoring System. *Sensors*, 14(3):3833–3860, February 2014.

[120] German Parisi, Stefan Wermter, and others. Hierarchical SOM-based detection of novel behavior for 3d human tracking. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8. IEEE, 2013.

[121] German I. Parisi, Cornelius Weber, and Stefan Wermter. Self-organizing neural integration of pose-motion features for human action recognition. *Frontiers in Neurorobotics*, 9, June 2015.

[122] Thomas Peacock and Nicolas Hadjiconstantinou. course materials for 2.003j/1.053j dynamics and control i, spring 2007. mit opencourseware (http://ocw.mit.edu). lecture 10: 2d motion of rigid bodies: Falling stick example, work-energy principle.

[123] Sida Peng, Yuan Liu, Qixing Huang, Hujun Bao, and Xiaowei Zhou. PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation. *arXiv:1812.11788 [cs]*, December 2018.

[124] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of Visual Words and Fusion Methods for Action Recognition: Comprehensive Study and Good Practice. *arXiv:1405.4506 [cs]*, May 2014.

[125] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action Recognition with Stacked Fisher Vectors. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8693, pages 581–595. Springer International Publishing, Cham, 2014.

[126] Martin Peniak, Anthony Morse, Christopher Larcombe, Salomon Ramirez-Contla, and Angelo Cangelosi. Aquila: An Open-Source GPU-Accelerated Toolkit for Cognitive Robotics Research. page 8, 2011.

[127] Ekachai Phaisangittisagul. An Analysis of the Regularization Between L2 and Dropout in Single Hidden Layer Neural Network. In *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pages 174–179, Bangkok, Thailand, January 2016. IEEE.

[128] Philips. HomeSafe with AutoAlert — Philips Lifeline ®. https://www.lifeline.philips.com/medical-alert-systems/homesafe-autoalert.html.

[129] PrimeSense. Prime sensor™ NITE 1.3 framework programmer's guide - NITE.pdf.

[130] Yann Prudent and Abdellatif Ennaji. An incremental growing neural gas learns topologies. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 1211–1216. IEEE, 2005.

[131] Francesco Puja, Simone Grazioso, Antonio Tammaro, Valsmis Ntouskos, Marta Sanzari, and Fiora Pirri. Vision-based deep execution monitoring. *arXiv:1709.10507 [cs]*, September 2017.

[132] Sarah M. Rabbitt, Alan E. Kazdin, and Brian Scassellati. Integrating socially assistive robotics into mental healthcare interventions: Applications and recommendations for expanded use. *Clinical Psychology Review*, 35:35–46, 2015.

[133] Gabriel Rada. Telemedicine: Are we advancing the science? *Cochrane Database of Systematic Reviews*, (9), 2015.

[134] Hossein Rahmani, Arif Mahmood, Du Q Huynh, and Ajmal Mian. HOPC: Histogram of Oriented Principal Components of 3D Pointclouds for Action Recognition. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 742–757. Springer International Publishing, 2014.

[135] Douglas L. Reilly, Leon N. Cooper, and Charles Elbaum. A neural model for category learning. *Biological Cybernetics*, 45(1):35–41, August 1982.

[136] Cornell University Robot Learning Lab. Cornell activity dataset (CAD-60) results. available at: `http://pr.cs.cornell.edu/humanactivities/results.php` accessed in 28th july 2017, 2017.

[137] Frank. Rosenblatt. *Principles of neurodynamics; perceptrons and the theory of brain mechanisms.* Spartan Books, 1962.

[138] Kenneth J. Rothman. No Adjustments Are Needed for Multiple Comparisons. *Epidemiology*, 1(1):43–46, January 1990.

[139] Caroline Rougier, Edouard Auvinet, Jacqueline Rousseau, Max Mignotte, and Jean Meunier. Fall Detection from Depth Map Video Sequences. In Bessam Abdulrazak, Sylvain Giroux, Bruno Bouchard, Hélène Pigot, and Mounir Mokhtari, editors, *Toward Useful Services for Elderly and People with Disabilities*, Lecture Notes in Computer Science, pages 121–128. Springer Berlin Heidelberg, 2011.

[140] D.W. Ruck, S.K. Rogers, M. Kabrisky, P.S. Maybeck, and M.E. Oxley. Comparative analysis of backpropagation and the extended kalman filter for training multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):686–691, 1992.

[141] David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations.* MIT Press, 1986.

[142] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image Classification with the Fisher Vector: Theory and Practice. *International Journal of Computer Vision*, 105(3):222–245, December 2013.

[143] Daniel Sanchez-Morillo, Miguel Angel Fernandez-Granero, and Antonio León Jiménez. Detecting COPD exacerbations early using daily telemonitoring of symptoms and k-means clustering: a pilot study. *Medical & Biological Engineering & Computing*, 53(5):441–451, 2015.

[144] Marta Sanzari, Valsamis Ntouskos, and Fiora Pirri. Discovery and recognition of motion primitives in human activities. *arXiv:1709.10494 [cs]*, September 2017.

[145] Egemen Savaskan, Sandra E. Müller, Andreas Böhringer, André Schulz, and Hartmut Schächinger. Antidepressive therapy with escitalopram improves mood, cognitive symptoms, and identity memory for angry faces in elderly depressed patients. *Int J Neuropsychopharmacol*, 11(3):381–388, May 2008.

[146] Peter H. Schonemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1), 1966.

[147] C. Schroeter, S. Mueller, M. Volkhardt, E. Einhorn, C. Huijnen, H. van den Heuvel, A. van Berlo, A. Bley, and H. Gross. Realization and user evaluation of a companion robot for people with mild cognitive impairments. In *2013 IEEE International Conference on Robotics and Automation*, pages 1153–1159, May 2013.

[148] Paolo Scocco, Monica Rapattoni, and Giovanna Fantoni. Nursing home institutionalization: a source of eustress or distress for the elderly? *International Journal of Geriatric Psychiatry*, 21(3):281–287, 2006.

[149] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th International Conference on Multimedia - MULTIMEDIA '07*, page 357, Augsburg, Germany, 2007. ACM Press.

[150] Sense4Care. Angel4 Fall Detector- Sense4Care. https://www.sense4care.com/.

[151] Junjie Shan and Srinivas Akella. 3d human action segmentation and recognition using pose kinetic energy. In *Advanced Robotics and its Social Impacts (ARSO), 2014 IEEE Workshop on*, pages 69–75. IEEE, 2014.

[152] A. Sharkey and N. Sharkey. Children, the Elderly, and Interactive Robots. *IEEE Robotics Automation Magazine*, 18(1):32–38, March 2011.

[153] Amanda Sharkey and Noel Sharkey. Granny and the robots: ethical issues in robot care for the elderly. *Ethics and Information Technology*, 14(1):27–40, March 2012.

[154] Roger N. Shepard and Jacqueline Metzler. Mental Rotation of Three-Dimensional Objects. *Science*, 171(3972):701–703, February 1971.

[155] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646, October 2016.

[156] Jee-Seon Shim, Kyungwon Oh, and Hyeon Chang Kim. Dietary assessment methods in epidemiologic studies. *Epidemiol Health*, 36, July 2014.

[157] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116, 2013.

[158] Karen Simonyan and Andrew Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. page 9, 2014.

[159] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[160] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *arXiv:1212.0402 [cs]*, December 2012. arXiv: 1212.0402.

[161] IEEE Spectrum. PR2 - ROBOTS: Your Guide to the World of Robotics. https://robots.ieee.org/robots/pr2/, 2020.

[162] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Unstructured human activity detection from rgbd images. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 842–849. IEEE, 2012.

[163] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. *arXiv:1512.00567 [cs]*, December 2015. arXiv: 1512.00567.

[164] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9. IEEE, 2015.

[165] Benjamin W Tatler, Nicholas J Wade, Hoi Kwan, John M Findlay, and Boris M Velichkovsky. Yarbus, eye movements, and vision. *i-Perception*, 1(1):7–27, July 2010.

[166] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528, June 2011.

[167] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, Santiago, Chile, December 2015. IEEE.

[168] Apple (UK). Apple Watch Series 5. https://www.apple.com/uk/shop/buy-watch/apple-watch.

[169] Carnegie Mellon University. Openpose real-time multi-person keypoint detection. available at`https://github.com/CMU-Perceptual-Computing-Lab/openpose`. accessed in 28th of july 2017.

[170] University of Massachussetts. LFW : Results. http://vis-www.cs.umass.edu/lfw/results.html#dlib, 2019.

[171] Terence D. Valenzuela, Denise J. Roe, Shan Cretin, Daniel W. Spaite, and Mary P. Larsen. Estimating effectiveness of cardiac arrest interventions: A logistic regression survival model. *Circulation*, 96(10):3308–3313, 1997.

[172] S. G. Vandenberg and A. R. Kuse. Mental rotations, a group test of three-dimensional spatial visualization. *Percept Mot Skills*, 47(2):599–604, October 1978.

[173] Joaquin Vanschoren. Meta-Learning. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Automated Machine Learning: Methods, Systems,*

Challenges, The Springer Series on Challenges in Machine Learning, pages 35–61. Springer International Publishing, Cham, 2019.

[174] K. Wada, T. Shibata, and Y. Kawaguchi. Long-term robot therapy in a health service facility for the aged - A case study for 5 years -. In *2009 IEEE International Conference on Rehabilitation Robotics*, pages 930–933, June 2009.

[175] Heng Wang and Cordelia Schmid. Action Recognition with Improved Trajectories. In *2013 IEEE International Conference on Computer Vision*, pages 3551–3558, Sydney, Australia, December 2013. IEEE.

[176] Lei Wang, Piotr Koniusz, and Du Q. Huynh. Hallucinating IDT Descriptors and I3D Optical Flow Features for Action Recognition with CNNs. *arXiv:1906.05910 [cs]*, June 2019.

[177] Limin Wang. Wanglimin/dense_flow, September 2017.

[178] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. *arXiv:1608.00859 [cs]*, August 2016. arXiv: 1608.00859.

[179] X. Wang, L. Gao, P. Wang, X. Sun, and X. Liu. Two-Stream 3-D convNet Fusion for Action Recognition in Videos With Arbitrary Size and Length. *IEEE Transactions on Multimedia*, 20(3):634–644, March 2018.

[180] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[181] Jennie L Wells and Andrea C Dumbrell. Nutrition and Aging: Assessment and Treatment of Compromised Nutritional Status in Frail Elderly Patients. *Clinical Interventions in Aging*, 1(1):67–79, March 2006.

[182] Paul Werbos and Paul J. (Paul John. Beyond regression : new tools for prediction and analysis in the behavioral sciences /. 1974.

[183] Tom M. A. Wilkinson, Gavin C. Donaldson, John R. Hurst, Terence A. R. Seemungal, and Jadwiga A. Wedzicha. Early therapy improves outcomes of exacerbations of chronic obstructive pulmonary disease. *American Journal of Respiratory and Critical Care Medicine*, 169(12):1298–1303, 2004.

[184] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, volume 5303, pages 650–663. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[185] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *arXiv:1711.00199 [cs]*, November 2017.

[186] Goh Yongli, Ooi Shih Yin, and Pang Ying Han. State of the art: a study on fall detection. *World Academy Science, Engineering and Technology*, 62:294–298, 2012.

[187] Sachiyo Yoshida. A Global Report on Falls Prevention Epidemiology of Falls. *WHO*, page 40, 2007.

[188] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow. In Fred A. Hamprecht, Christoph Schnörr, and Bernd Jähne, editors, *Pattern Recognition*, Lecture Notes in Computer Science, pages 214–223. Springer Berlin Heidelberg, 2007.

[189] Sonam Zamir, Catherine Hagan Hennessy, Adrian H Taylor, and Ray B Jones. Video-calls to reduce loneliness and social isolation within care environments for older people: An implementation study using collaborative action research. *BMC Geriatr*, 18, March 2018.

[190] Chenyang Zhang. RGB-D Camera-based Daily Living Activity Recognition. page 7, 2012.

[191] Jing Zhang, Wanqing Li, Philip O. Ogunbona, Pichao Wang, and Chang Tang. RGB-D-based Action Recognition Datasets: A Survey. *arXiv preprint arXiv:1601.05511*, 2016.

[192] Yongli Zhang and Yuhong Yang. Cross-validation for selecting a model selection procedure. *Journal of Econometrics*, 187(1):95–112, July 2015.

[193] Z. Zhang, W. Liu, V. Metsis, and V. Athitsos. A viewpoint-independent statistical method for fall detection. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3626–3630, November 2012.

[194] Zhong Zhang, Christopher Conly, and Vassilis Athitsos. Evaluating Depth-Based Computer Vision Methods for Fall Detection under Occlusions. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Ryan McMahan, Jason Jerald, Hui Zhang, Steven M. Drucker, Chandra Kambhamettu, Maha El Choubassi, Zhigang Deng, and Mark Carlson, editors, *Advances in Visual Computing*, Lecture Notes in Computer Science, pages 196–207. Springer International Publishing, 2014.

[195] Zhong Zhang, Christopher Conly, and Vassilis Athitsos. A survey on vision-based fall detection. In *Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '15*, pages 1–7, Corfu, Greece, 2015. ACM Press.

[196] Jiagang Zhu, Wei Zou, and Zheng Zhu. End-to-end Video-level Representation Learning for Action Recognition. *arXiv:1711.04161 [cs]*, November 2017.

Bound Copies of Published Relevant Papers

Implementation of a Modular Growing When Required Neural Gas Architecture for Recognition of Falls

Frederico B. Klein[1(✉)], Karla Štěpánová[2], and Angelo Cangelosi[1]

[1] School of Computing, Electronics and Mathematics,
Plymouth University, Plymouth, UK
{frederico.klein,a.cangelosi}@plymouth.ac.uk
[2] Department of Cybernetics, Czech Technical University, Prague, Czech Republic
stepakar@fel.cvut.cz
http://www.plymouth.ac.uk, http://www.fel.cvut.cz

Abstract. In this paper we aim for the replication of a state of the art architecture for recognition of human actions using skeleton poses obtained from a depth sensor. We review the usefulness of accurate human action recognition in the field of robotic elderly care, focusing on fall detection. We attempt fall recognition using a chained Growing When Required neural gas classifier that is fed only skeleton joints data. We test this architecture against Recurrent SOMs (RSOMs) to classify the TST Fall detection database ver. 2, a specialised dataset for fall sequences. We also introduce a simplified mathematical model of falls for easier and faster bench-testing of classification algorithms for fall detection.

The outcome of classifying falls from our mathematical model was successful with an accuracy of $97.12 \pm 1.65\,\%$ and from the TST Fall detection database ver. 2 with an accuracy of $90.2 \pm 2.68\,\%$ when a filter was added.

Keywords: Action recognition · Falls · Neural networks · Neural gas · Topological classifiers · Socially assistive robotics

1 Introduction

In the field of robotics, activity detection [11] is a fundamental concept if the robots are used in a setting where they are expected to cooperate with humans. The initial approaches to the detection of human actions involved the processing of RGB images, and as such were proven to be a hard problem due to the difficulty in segmenting the human body from the background and accurately processing pose information. Recently however this task was made considerably easier with the introduction of skeleton tracking based on depth-sensing cameras as implemented by the Microsoft Kinect and as it steadily improves, it also allows for more serious tasks that depend on activity recognition to be tackled, such as activity detection. We will focus on its use in the context of socially assistive robotics for social elderly care.

A. Hirose et al. (Eds.): ICONIP 2016, Part I, LNCS 9947, pp. 1–9, 2016.
DOI: 10.1007/978-3-319-46687-3_58

1.1 Ageing Population

With the ageing of populations around the world, elderly care is a field of growing concern. Many different technological aids [3] are being developed specifically for this population and robotics has emerged as a possible solution as the mobilisation of human caretakers for such a large amount of persons seems infeasible. While robots in regards to human-robot-interaction are yet to find a particular field in which it is undeniably useful, an interesting approach [18] to their use is finding newer areas in which they can nothing but excel, simply because there are no persons nor other technology available to perform that task. One of such tasks is around the clock health monitoring for independent living.

1.2 Our Task of Interest: Fall Detection

It is medical fact [9] that diseases that can present themselves as a loss of consciousness, such as strokes and heart infarctions - those two, the leading causes of death world wide - can have excellent prognosis if treated within 3 h. Particularly cardiac arrests present a survival rate of about one in each three subjects, if CPR and defibrillation are initiated in less than 5 min, whereas the probability of survival without any help is virtually zero [19]. Also other diseases such as pneumonia or COPD exacerbations do tend to have better prognosis [20] if treated promptly. One must take care as to not make bold assumptions, even more under the light that major reviews [5] are yet to reveal clear benefits of telemedicine, but some interesting recent results [4,8] demonstrate COPD as a likely candidate to benefit from remote monitoring.

The specific task of fall detection has recently attracted a lot of research, with a primary focus on smart home environments. In fact most fall detection systems [21] involve wearing special sensors device with accelerometers or detectors built on the floor or a combination of video and wearable devices with a sensor fusion approach in order to increase the accuracy of detection even further. These approaches although more simple (and therefore robust) have however the limitation of needing either a sensor to be worn at all times, or that the person's house to be adapted for this, which in practice will vastly limit its adherence. We see coding fall detection into some sort of a multi-functional robotic companion-that could have as one of its many functionalities: fall detection-as a reasonable solution to this problem. A robot can follow the user in different environment, position itself in order to prevent image occlusion and we avoid the need to renovate someone's house or remember always to wear a sensor.

1.3 Our Approach: Use the Parisi's Multilayer GWR Classifier

Using an unsupervised method for topological description [6] of tasks is not a new idea, since these methods have many possible advantages such as the ability to "operate autonomously, on-line or life-long, and in a non-stationary environment". We chose to replicate the infrastructure implemented by [15] for it's overall performance in the CAD60 database and the theoretical generality of

the method. In this paper we describe our implementation of a classifier based on an unsupervised Growing When Required Neural Gas with a sliding window scheme for time integration and chained in multiple layers to implement noise removal.

Source code (available in www.github.com/frederico-klein/ICONIP2016) of the Growing When Required Neural Gas implementation (in Matlab and Julia) is provided, as well as the full classification architecture and the inverted pendulum model (only in Matlab).

2 Materials and Methods

2.1 Justification for the Chosen Architecture: A Chained GWR Sliding Window Topological Classifier

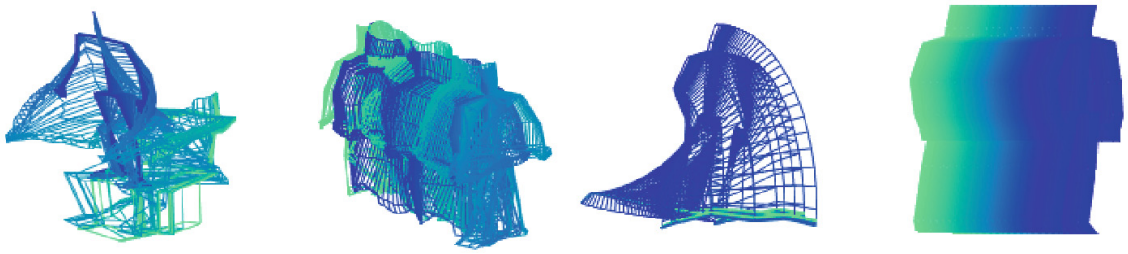
A detailed discussion of different types of neural gases is outside of the scope of this text. For a more in depth understanding one should probably first refer to Martinetz's paper [12] that implemented the first neural gas and later to Marsland's paper [13], that implemented the Growing When Required neural gas (GWR). The justification of using multiple chained gases (as opposed to one) is, first the biological plausibility reviewed extensively by Parisi but secondly probably due to necessity regarding the way too long execution time of a gas with a high number of dimensions. Finally one must add that, although neural gases, due to their nature, adjust to data that changes over time, this feature does not seem useful in tracking movement. For this function a sliding window scheme was used.

The dataset we chose to test our implementation was the TST v2 dataset contains skeleton positions Microsoft Kinect v2 and IMU data for 11 subjects performing either ADLs (activities daily living) and simulated falls. The subjects were between 22 and 39 years old, with different height (1.62–1.97 m) and build. Each of the two main groups (ADLs or Falls) contains 4 activities that are repeated three times by each subject [7]. For our present study only the skeleton joints in depth and skeleton space and time information were used. The accelerometer, as well as other data, were not used for our algorithm.

In addition to the real dataset a simplified stick model was developed to test the ability of a Growing When Required multilayer with a sliding window classifier to discriminate between action sequences that included a fall. We modelled 2 different activities, a fall and a walk as the movement of a stick in a 3D space and then simply substituted the stick for a typical skeleton.

Fall. We simulated fall of a person by a free falling inverted pendulum rod with a random initial pitch angular velocity θ and perfect slippage. It can be shown [16] that the kinematics differential equations that describe angle and position changes for a rod are:

$$-mg\frac{L}{2}cos\theta = \left(I_c + \frac{mL^2}{4}cos^2\theta\right)\ddot{\theta} - \frac{mL^2}{4}cos\theta sin\dot{\theta}^2 \tag{1}$$



(a) (b) (c) (d)

Fig. 1. (a) A typical fall from TST v2 dataset. (b) one of the ADLs from the TST v2 dataset, a walk. (c) a typical fall from our model. (d) a "walk" from our model.

$$0 = m\ddot{x}_c \tag{2}$$

And, approximating a person by a slender rod, one has $I_c = \frac{mL^2}{3}$. The model also was given simulation parameters to add random noise in the variables of height (1.6–1.9 m), initial position (within a square area) and any initial yaw angle.

Walk. The simulation of a person's walk was done by simply doing a linear space of displacements inside the area that would be covered by the Kinect sensor, with random initial positions and walking angle (Fig. 1).

2.2 Skeleton Data

The algorithm presented uses skeleton data and not RGB-D raw images. A more thorough descriptions [17] of the data obtained from the depth sensor should be referenced, but in short it is a set of J points (where J is the number of joints) with x, y and z coordinates, each representing a landmark on the body in time [10] in a 3D space. We represent thus a particular pose as the concatenation of these J points, such as that for each time frame k we have a pose p represented by the matrix:

$$p(k) = \begin{bmatrix} j_{1x}(k) & j_{1y}(k) & j_{1z}(k) \\ j_{2x}(k) & j_{2y}(k) & j_{2z}(k) \\ \cdots & & \\ j_{Jx}(k) & j_{Jy}(k) & j_{Jz}(k) \end{bmatrix} \tag{3}$$

An action sequence represented on discrete time steps $1...K$ could therefore represented as the multidimensional array resulting of the sequential concatenation of the k-th pose matrices. To use the pose information with a gas we change the representation of the pose matrix $p(k)$ into a vector size $3 * J$ and the action sequence is the horizontal concatenation of the all the k-th, $p(k)$ matrices. One may thus understand the pose vector as a single point in a high dimensional space and an action sequence as a necessarily continuous trajectory in that space.

2.3 Construction and Randomisation of Training and Validation Sets

The dataset was separated into training and validation sets containing 80 % and 20 % of data respectively, before each training similarly to a repeated learning-testing method [1,2]. They were separated by subject, so that each subject had all of its actions belonging exclusively to one set. This was done to describe a more realistic testing scenario, in which the subject performing the activities is completely new having no activity data of himself in the training set, preventing bias in accuracy estimation due to overfitting the training set.

2.4 Preconditioning

The GWR algorithm is not translation invariant, so the first action performed on the data was to select a joint - based on our reference algorithm we used the hips and subtracted the offset from the hips joint in both the z and x coordinates from all other joint vectors. Secondly, we normalised (scaled) the data so that after scaling variance of the data would be equal to 1. The final step was to implement a centroid generating function, so to generate a smaller dimensionality representation of the skeleton poses, in a similar fashion to the function tested by Parisi. We created a model of 3 centroids that were the average position of the skeleton points such that the upper centroid was composed by the joints: head, neck, left shoulder, right shoulder, left elbow, right elbow; middle centroid corresponded to torso and lower centroid: left knee, right knee, left hip, right hip. Many other preconditioning functions are available on the supplied code and maybe be tried by the interested reader.

2.5 Classifier Architecture

The classifier was implemented as a serial chaining of gas subunits. This was done to enable different structures to be tried with minimal effort. All classification attempts in this text were done using 5 gas subunits linked in manner as to implement the architecture in Parisi's [15] paper (see Fig. 2), that is, 2 parallel sets of 2 gas subunits in series, each stream dealing with either pose positions or pose velocities and a last gas that integrates both.

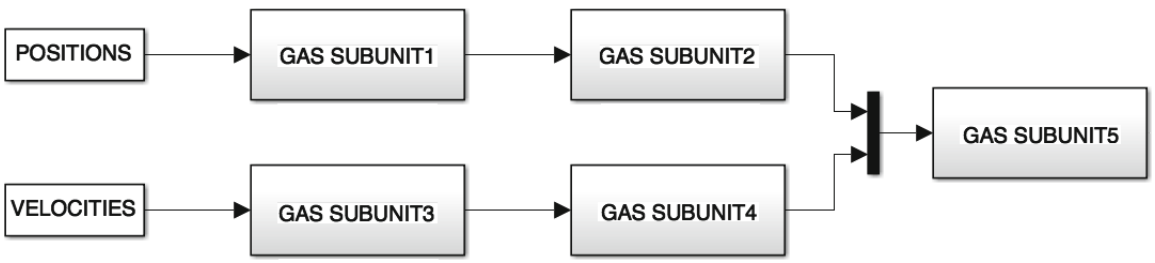


Fig. 2. Diagram of the classifier architecture.

For each gas subunit there are 5 main chained elements that are responsible for implementation estimation and classification:

- Sliding Window: implements the temporal concatenation of sample (also implements concatenation of multiple streams in case they exist).
- Gas Creator: receives data samples $p(k)$ or concatenated poses $W_l(k)$ and implements the learning algorithm for either the Growing When Required neural gas or the Growing Neural Gas.
- Mapping: finds the best matching pose from the nodes matrix A corresponding to each sample from the dataset.
- Labeller: simple labelling function that assigns the label of estimated concatenated pose as the same as the label of the pose to which it best matches.
- Activation checker: during training, checks to see if points are able to be well represented by the gas, and if not removes them from the sample.

3 Results and Discussion

For all the results here presented, the simulation parameters for the GWR neural gas are the same as in our reference paper [15].

3.1 Cornell CAD60 Dataset

As a means of comparing our implementation with that of Parisi, we also tested our architecture on the CAD60 dataset. Apparently our implementation does a lot of overfitting, as it reaches 99.6 % accuracy on the training set (average: 99.38 ± 0.2 % for 8 trials) but only reaches 71.7 % on the validation set. We noticed however that misclassifications were limited to some specific actions, with most having the same accuracy (greater than 90 %) in both sets. It is our conjecture that this difference reflects that the CAD60 dataset is too small to allow our stricter cross-validation method to produce generalization.

3.2 Falling Stick Model

Our algorithm, even with a much smaller network (100 nodes), seems to be quite consistently capable of classifying our faux fall/walk model. We simulated 20 subjects performing either a fall or a walk. The peak accuracy on the validation set of our implementation was 98.33 % (average: 97.12 ± 1.65 % for 8 trials).

3.3 TST Fall Detection ver.2 Dataset

Learning Across Layers. In order to understand how learning happens across layers we analyse the output classification from 5 gases with 1000 nodes run over 10 epochs (see Table 1) the results reflect what we would expect: there is a steady increase as we progress through the layers and there is a gain in accuracy.

Table 1. Progression of classification accuracy within different layers for chained GWR Neural Gas classifier with 1000 nodes and run over 10 epochs (for 8 trials).

Gas element	Validation set	Training set
GWR gas 1 Pos	$67.69 \pm 0.73\,\%$	$93.04 \pm 0.12\,\%$
GWR gas 2 Vel	$64.80 \pm 0.93\,\%$	$69.43 \pm 0.66\,\%$
GWR gas 3 Pos	$66.93 \pm 1.02\,\%$	$90.45 \pm 0.18\,\%$
GWR gas 4 Vel	$65.14 \pm 0.77\,\%$	$70.66 \pm 0.65\,\%$
GWR gas 5 STS	$73.99 \pm 1.16\,\%$	$88.35 \pm 0.41\,\%$

Mode Filter. With the intention of performing some sort of temporal filtering, we implemented a moving mode filter. The moving mode filter had an important positive effect on the classification results of the TST v2 database (see Table 2). Highest classification accuracy achieved (See Table 3) was 94.2 % on the validation set by 3^{rd} gas with 1000 nodes and 10 epochs with mode filter length of 35 data samples. One must note that adding a moving mode filter of size 35 means a delay of 11.67 s (since we need $(9+1) * 35$ samples @30 Hz) much more than the 0.6 s Parisi reported.

Table 2. Accuracies (in %) after applying the moving mode filter on classification results of the final gas unit of the classifier (for 8, 8 and 1 trials respectively).

Epochs	10		20		30	
Filter length	Val	Train	Val	Train	Val	Train
5	81.25 ± 1.44	95.05 ± 0.69	79.72 ± 2.09	79.72 ± 2.09	82.7	94.9
10	85.95 ± 2.13	95.74 ± 0.54	83.87 ± 2.93	96.00 ± 0.38	86.9	95.5
15	88.57 ± 2.48	95.31 ± 0.36	85.91 ± 2.93	95.46 ± 0.30	88.6	97.0
20	89.95 ± 2.36	95.34 ± 0.23	86.70 ± 4.08	95.26 ± 0.31	88.6	95.0
25	90.16 ± 2.28	94.32 ± 0.27	87.37 ± 3.79	94.47 ± 0.25	89.2	95.4
35	90.20 ± 2.68	92.29 ± 0.37	88.49 ± 3.83	92.44 ± 0.38	91.5	93.3
40	89.28 ± 2.68	91.23 ± 0.32	87.75 ± 3.82	91.33 ± 0.35	91.5	92.2
50	87.39 ± 2.06	88.84 ± 0.33	85.35 ± 3.39	88.80 ± 0.49	91.3	89.9

Comparison with RSOM. As a means of comparing the performance of our implementation, we also classified the TST v2 dataset using an RSOM implementation. The RSOM used the same preconditioning as we did for the chained gas classifier and a set of 3 consecutive poses $(p(k), p(k-1), p(k-2))$. The simulation parameters were: 900 nodes, 30 epochs, method 'RSOMHebbV01'. The peak accuracy on the validation set of the RSOM with these parameters was 78.76 % (average: $77.67 \pm 0.77\,\%$ for 5 trials).

Table 3. Confusion matrix for our most accurate gas classifier. The calculated accuracy for the validation set is 94.2 %, higher than the 92.0 % training set.

Validation set		Target		Training set		Target	
		1	2			1	2
Output	1	1532	36	Output	1	4006	258
	2	124	1054		2	328	2746

4 Conclusion

The resulting classification scheme does the task which we want, that is, discriminate falls within the TST v2 dataset, it does it better than the RSOM and it does it consistently with around 90.2 ± 2.68 % accuracy while using the mode filter. We believed we achieved our goal and we have now a classifier of falls with openly accessible code that will hopefully encourage persons into designing experiments using fall detection or using neural gases for classification of hard to classify data.

Acknowledgment. This work was partially supported by CNPq Brazil (scholarship 232590/2014-1) and by SGS grant No. 10/279/OHK3/3T/13, sponsored by the CTU in Prague, Czech Republic.

References

1. Arlot, S., Celisse, A.: A survey of cross-validation procedures for model selection. Stat. Surv. **4**, 40–79 (2010). doi:10.1214/09-SS054
2. Burman, P.: A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. Biometrika **76**(3), 503–514 (1989)
3. Chan, M., Estéve, D., Escriba, C., Campo, E.: A review of smart homes-present state and future challenges. Comput. Methods Prog. Biomed. **91**, 55–81 (2008). doi:10.1016/j.cmpb.2008.02.001
4. Fernandez-Granero, M.A., Sanchez-Morillo, D., Leon-Jimenez, A.: Computerised analysis of telemonitored respiratory sounds for predicting acute exacerbations of copd. Sensors (Basel) **15**, 26978–26996 (2015). doi:10.3390/s151026978
5. Flodgren, G., Rachas, A., Farmer, A.J., Inzitari, M., Shepperd, S.: Interactive telemedicine: effects on professional practice and health care outcomes. In: Cochrane Database of Systematic Reviews. Wiley (2015)
6. Furao, S., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. Neural Netw. **19**, 90–106 (2006). doi:10.1016/j.neunet.2005.04.006
7. Gasparrini, S., Cippitelli, E., Gambi, E., Spinsante, S., Wåhslén, J., Orhan, I., Lindh, T.: Proposal and experimental evaluation of fall detection solution basedon wearable and depth data fusion. In: Loshkovska, S., Koceski, S. (eds.) ICTInnovations 2015, Advances in Intelligent Systems and Computing, pp. 99–108. Springer International Publishing, Switzerland (2016)

8. Ho, T.-W., Huang, C.-T., Chiu, H.-C., Ruan, S.-Y., Tsai, Y.-J., Yu, C.-J., Lai, F.: Effectiveness of telemonitoring in patients with chronic obstructive pulmonary disease in Taiwan-a randomized controlled Trial. Sci. Rep. **6** (2016). doi:10.1038/srep23797
9. Jauch, E.C., Saver, J.L., Demaerschalk, B.M., Khatri, P., McMullan Jr., P.W., Qureshi, A.I., Rosenfield, K., Scott, P.A., Summers, D.R., Wang, D.Z.: AHA/ASA Guideline. Stroke (2013)
10. JointType enumeration [WWW Document], n.d. https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx. Accessed 14 May 16
11. Koppula, H.S., Saxena, A.: Anticipating human activities using object affordances for reactive robotic response. IEEE Trans. Pattern Anal. Mach. Intell. **38**, 14–29 (2016)
12. Martinetz, T.M., Schulten, K.J.: A "Neural Gas" networklearns topologies. In: Kohonen, T., Mäkisara, K., Simula, O., Kangas, J. (eds.) Proceedings of the International Conference on Artificial Neural Networks 1991, Espoo, Finland, pp. 397–402, Amsterdam, North-Holland, New York (1991)
13. Marsland, S., Shapiro, J., Nehmzow, U.: A self-organising network that grows when required. Neural Netw. **15**, 1041–1058 (2002)
14. Parisi, G., Wermter, S., others.: Hierarchical SOM-based detection of novel behavior for 3D human tracking. In: The 2013 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2011)
15. Parisi, G.I., Weber, C., Wermter, S.: Self-organizing neural integration of pose-motion features for human action recognition. Front. Neurorobotics **9**, (2015). doi:10.3389/fnbot.2015.00003
16. Peacock, T., Hadjiconstantinou, N.: Course materials for 2.003J/1.053J dynamics and control I, Spring (2007). MIT OpenCourseWare (http://ocw.mit.edu), Massachusetts Institute of Technology. Accessed 13 May 2016
17. Prime sensor^{TM} NITE 1.3 framework programmer's guide - NITE.pdf. http://pr.cs.cornell.edu/humanactivities/data/NITE.pdf. Accessed 14 May 2016
18. Rabbitt, S.M., Kazdin, A.E., Scassellati, B.: Applications and recommendations for expanded use. Clin. Psychol. Rev. 35, 35–46. doi:10.1016/j.cpr.2014.07.001
19. Valenzuela, T.D., Roe, D.J., Cretin, S., Spaite, D.W., Larsen, M.P.: Estimating effectiveness of cardiac arrest interventions: a logistic regression survival model. Circulation **96**, 3308–3313 (1997). doi:10.1161/01.CIR.96.10.3308
20. Wilkinson, T.M.A., Donaldson, G.C., Hurst, J.R., Seemungal, T.A.R., Wedzicha, J.A.: Early therapy improves outcomes of exacerbations of chronic obstructive pulmonary disease. Am. J. Respir. Crit. Care Med. **169**, 1298–1303 (2004). doi:10.1164/rccm.200310-1443OC
21. Yongli, G., Yin, O.S., Han, P.Y.: State of the art: a study on fall detection. World Acad. Sci. Eng. Technol. **62**, 294–298 (2012)