UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PROIECT DE DIPLOMĂ

Mecanisme de fuziune pentru construcția de sisteme eficiente de recunoaștere a activităților umane

Ana-Cosmina Popescu

Coordonator științific:

Conf. dr. ing. Irina Mocanu

BUCUREȘTI
2019

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DIPLOMA PROJECT

Fusion Mechanisms for Building Efficient Human Activity Recognition Systems

Ana-Cosmina Popescu

Thesis advisor:

Associate Professor Dr. Eng. Irina Mocanu

BUCHAREST

2019

CONTENTS

SINOPSIS

Detecția activităților umane este de mulți ani un subiect de interes în domeniul vizualizării computerizate (engl. computer vision), datorită numeroaselor sale aplicații în medicină, supraveghere, divertisment, interacțiune om-calculator, pentru a menționa câteva. O dată cu progresele tehnologice actuale (privind în special echipamentele de achiziție de date și puterea de prelucrare), pot fi investigate noi soluții. Lucrarea prezintă un sistem pentru recunoașterea activităților umane bazat pe combinarea informației provenite din toate canalele unui videoclip 3D: fluxul RGB, cel de adâncime, scheletul și obiectele din jur sunt trecute independent prin rețele convoluționale 2D. Aceste rețele sunt obținute cu o metodă de învățare automată autonomă bazată pe tehnica NAS (engl. Neural Architecture Search). Toate ieșirile rețelelor sunt combinate într-o predicție finală. Mecanismele de fuziune captează informația relevantă dintr-un videoclip, totodată nefiind intensiv computaționale. Aplicat pe setul de date *MSR-DailyActivity3D*, sistemul conduce la o acuratețe de 98.43%. Pe setul de date realizat de autori în cadrul proiectului de cercetare, *PRECIS HAR Dataset*, se obține o acuratețe de 94.38%.

ABSTRACT

Human activity recognition has been a branch of interest in the field of computer vision for decades, due to its numerous applications in medicine, surveillance, entertainment, human-computer interaction, to mention a few. Along with the actual technological progress (especially regarding data acquisition hardware and computational power), new solutions can be investigated. We design a system for recognizing human activities based on merging information from all channels of a 3D video: RGB, depth, skeleton and context objects are passed independently through 2D convolutional neural networks. These networks are computed with an automated machine learning method based on NAS technique (Neural Architecture Search). All network outputs are combined in a final array of class scores. The fusion mechanisms are not computationally intensive, but reflect the meaningful information from a video. On *MSR-DailyActivity3D*, the system yields 98.43% accuracy. On the dataset acquired by the authors for this project, *PRECIS HAR Dataset*, we get an accuracy of 94.38%.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my thesis advisor, Associate Professor Dr. Eng. Irina Mocanu, for her expertise and help.

I would also like to thank to Mr. Bogdan Cramariuc and Mrs. Oana Cramariuc, for their advice and for providing me with necessary equipment for filming the dataset.

In the meantime, many thanks to everybody that helped me film the dataset.

On a personal note, I thank my family for their support and understanding.

LIST OF ABBREVIATIONS

ADL	Activities of Daily Living
AutoML	Automated Machine Learning
BMI	Binary Motion Image
CNN	Convolutional Neural Network
DMM	Depth Motion Map
fDMM	Front DMM
HAR	Human Activity Recognition
MHI	Motion History Image
ML	Machine Learning
NAS	Neural Architecture Search
RGB-D	Red, Green, Blue, Depth
RL	Reinforcement Learning
RNN	Recurrent Neural Network
sDMM	Side DMM
SVM	Support Vector Machine
tDMM	Top DMM

1 INTRODUCTION

In this chapter we introduce the ideas that led to the development of the system proposed in the paper: we start by giving a general context related to activity recognition, and then we state the problem that we identified. We continue with setting the objectives, thus making the transition towards the proposed solution. After briefly describing the solution, we summarize our results. We end this chapter by presenting the paper structure.

1.1 Context

Human activity recognition is one of the sub-domains of computer vision with the most practical applications that lead to life quality improvement. It aims to allow computers to understand the surrounding environment, and consequently become pro-active in necessary situations.

During the past years, the development of activity detection mechanisms has led to solutions such as fall detection systems [1, 2], health monitoring systems [3, 4], school surveillance [5], or even to more esoteric ones, like cooking assistants [6], or sports personal trainers [7, 8].

Even if a lot of effort has already been invested in this direction, solutions are still imperfect, having drawbacks like the lack of enough training data, the difficulty to be re-trained for solving custom problems, bad functioning when environmental conditions change, the difficulty to run in real-time or non-stop, or a need for expensive specialized hardware.

Given this context, we explore data fusion mechanisms that could help us in building an efficient human activity recognition, designed for indoor monitoring.

1.2 Problem

The project solves the problem of indoor activity detection, mainly with regard to assisting elderly and sick people. It proposes a system that is fast and lightweight (in terms of memory and data transfer), that can be used for real-time surveillance.

Even if nowadays there are several hardware devices (sensors, cameras, monitoring systems) that are built to assist the people in need, most of them require human intervention to interpret data. However, the human factor is not fully reliable, being prone to error, and also not accessible to everyone (human resources cost more than technological ones, like robots or cameras). In addition, for generating alerts in case of emergencies, or offering aid when

needed, the hardware devices have to be autonomous, to run uninterruptedly and to be able to reason on their own, in order to identify a certain activity that is performed.

Another problem that we are striving to solve is the lack of benchmarks for activity detection problems, based on RGB-D (red, green, blue, depth) videos. The available resources are quite limited, as far as the number and diversity of public datasets are concerned [9].

1.3 Objectives

To solve the previously stated problem, the human activity detection system has to achieve the following objectives:

- To detect at least 10 different activity types, with an accuracy of at least 80%.
- To require a classification time of less than two seconds per video (where video has a length of 2-10 seconds), including server-client communication.
- To have a lightweight client component, occupying less than 100MB.
- To provide an end-to-end solution with reduced financial costs.
- To provide a system that is easily portable on hardware devices.
- To offer a new dataset for activity detection with at least 10 activities and 30 subjects.

1.4 Solution

The approach proposed in this paper suggests an **innovative** and **effective** HAR (human action recognition) **system**, based on RGB-D videos. Its main purpose is to be integrated with surveillance cameras or assertive robots designed for assisting older people indoors. The system identifies dangerous situations in order to send fast emergency alerts.

The three key ideas on which the system relies are:

(i) A video (i.e. an ordered sequence of frames) can be compressed in a single image (called *motion history image* [10]); consequently, activity recognition in videos can be narrowed to the problem of classifying images.
(ii) Combining multiple results, not necessarily highly accurate, will most likely lead to a correct solution [11, 12].
(iii) All classifications performed on different data streams are useful and help in increasing the overall accuracy, if merged according to different coefficients of confidence [13, 14].

Given a 3D recording, video information can be decomposed into four different data streams: RGB data, depth data, skeleton coordinates, and contextual objects. The proposed solution consists of combining (with a *channel fusion* mechanism) the likelihood scores returned by passing each data stream to a specialized CNN. These scores are obtained by passing the

fused data of each stream (with a *temporal fusion* mechanism) as input to a 2D convolutional neural network. Finally, for extra accuracy, the combined scores can be weighted according to the surrounding objects identified in the video (with a *context fusion* mechanism).

The proposed system is highly customizable, being completely decomposable into independent modules. Plus, it is easy to install on different types of hardware and does not require expensive and large-sized data acquisition mechanisms.

Apart from the HAR system, we also introduce a **new dataset** for human activity detection: *PRECIS HAR Dataset*. This dataset contributes to solving the lack of training/testing data for designing activity recognition systems. It has a wide and diverse range of activities (16 activities) and a big number of subjects (50), compared to other public datasets [9].

1.5 Results

We trained the HAR system by using two datasets, each of them with 16 classes of activities. Upon evaluation, we obtained high accuracies (98.43% on *MSRDailyActivity3D* and 94.38% on *PRECIS HAR Dataset*). In addition to being accurate, we made the HAR system effective, as far as classification time and data transfer from the client to the server are concerned.

As another achievement, the *PRECIS HAR Dataset* is now publicly available, thus contributing to the advances in the domain of activity recognition.

1.6 Paper Structure

In chapter 2, we introduce the underlying motivation for building a human activity recognition system that is not only accurate but also efficient.

Then, in chapter 3, we show the current efforts that have been done towards activity recognition. We state the general context, present top approaches for solving the HAR problem, highlight the current research directions, and list the solutions that are going to be used in our system. We give pertinent arguments to support our choices.

Next, chapter 4 details upon our solution for human activity recognition: we describe the used input streams, the architecture, the data flow diagram, and the system modules. In chapter 5 we give technical details: used technologies, algorithms and configurations, and development process. We point out the encountered difficulties and used solutions.

After describing the implementation process, we analyze results in chapter 6. We describe used datasets, give different performance metrics and compare our results with top ones.

The paper ends with chapter 7, where we draw conclusions on the HAR system and outline possible future developments and improvements.

2 MOTIVATION

This chapter discusses the five main factors that represent the motivation for the work supported by this paper. Firstly, HAR has a key role in the currently emergent domain of Computer Vision. Secondly, the proposed system is potentially useful for many categories of stakeholders. Then, the system is easy to integrate on existing hardware (cameras, robots). Fourthly, we identified a lack of specialized HAR benchmarks. Finally, to our knowledge, the ideas that we present are original.

Key Role in an Emerging Domain

We are nowadays facing a rapid development of machine learning and computer vision. Technological advances support the emergence of new solutions and ideas in this direction. Thus, we believe that addressing the problem of human activity recognition, a central sub-branch of computer vision, is a direction that offers us many research possibilities, and the chance to improve our solution in the near future.

Fusion mechanisms in the context of HAR have also been investigated before; however, the increase of computational power that we are nowadays facing allows us to efficiently make use of them in practice, not having to consider them anymore as speed bottlenecks.

Numerous Potential Stakeholders

Designing a surveillance system based on human activity recognition has a wide range of possible applications, from which several categories of people could benefit.

The category of people that would have to benefit the most from using our HAR system is represented by older people. A study performed by the Department of Economic and Social Affairs of the United Nations [15] shows that *"globally, the number of older persons is growing faster than the numbers of people in any other age group"*. They also estimate that by 2030, people over the age of 60 will represent more than 25% of Europe's population. Thus, this category is numerous, and designing solutions with regard to it is justifiable.

While older people are the direct beneficiaries of such a system, there are also several indirect stakeholders: the families of older people are assured of the safety of their dear ones, also diminishing the costs of looking after.

The surveillance system can also be used in other cases, at home or in public institutions. For example, teachers can monitor young children in kindergartens, or hospital employees can monitor constantly patients from different salons. It can also be integrated into other activities of daily living, such as cooking, sports, or entertainment.

Then, recording a new public RGB-D dataset would be useful for people conducting research projects in computer vision, ranging from researchers to students and teachers.

Ease of Integration

A product difficult to understand/use/integrate with practical applications, no matter how good, might end up not being used. We intend to build a HAR system that is lightweight, easy to use, does not require big quantities of data transferred to the server, is not computationally expensive, and can be easily integrated with hardware solutions (cameras, robots, etc.). By building such a system, we expect to have a high rate of adoption of the product, because a user would not have difficulties in understanding or adopting our solution.

Furthermore, we investigate different fusion mechanisms, that can be applied to different types of input, because we want to provide solutions for training a HAR system with any available data stream (RGB, depth, or both). We develop a system that is intended to be used with RGB-D cameras but can also work with cameras that provide only one stream (RGB or depth).

Lack of Specialized Benchmarks

A survey conducted by Zhang et al. (2016) [9] shows that, by the year 2015, there were only 27 single-view public datasets, having between 7 to 27 actions (13 on average), performed by 1 to 40 subjects (15 on average), with a total of 14 to 1760 examples (506 on average). There are thus few datasets available for training/evaluation, and they are not large-scale.

Apart from the size, the existing datasets have also limited applicability: each dataset provides activities belonging to a certain type, allowing models to be trained to recognize only those activity classes. Filming a new dataset allows adding diversity to the existing activity range.

Originality

One last factor that motivates us is that, to the best of our knowledge, nobody has proposed any similar solution to the problem of human activity recognition so far. Some of the sub-ideas are found in technical literature, but how we assemble all system components is original.

As far as the training/testing part is concerned, we train our system on two different datasets, out of which one is entirely filmed, pre-processed and labeled by us.

In this chapter, we described the main factors that motivate us in conducting our work concerning HAR fusion mechanisms, and justify our efforts in this direction. We believe that our work could contribute to the advance of the domain, which would lead to a clear improvement in the quality of life.

3 EXISTING METHODS

In this chapter, we firstly present the general context that currently frames the problem of human activity recognition. Then, we detail some of the most performant HAR systems that have already been proposed. Thirdly, we group research efforts into research directions. Lastly, from all discussed solutions, we list the ones that are going to be used in our own HAR system.

3.1 State of the Art

Human activity recognition (HAR) represents an important sub-domain of computer vision, referring to the task of identifying (or correctly classifying) human actions, given a full video of a person performing that action. It is especially a topic of interest in computer vision, mainly because of its numerous applications: surveillance, medicine, entertainment, human-computer interaction, video retrieval (on the Internet), autonomous driving and robotics.

Although the domain of human activity recognition has been studied for more than four decades [16–18], there is still progress to be made. To give a concrete example, Kong and Fu (2018) [19] identify the following research challenges, as far as HAR is concerned: intra- (people perform the same actions in different ways) and inter- (several class activities are much alike) class variations, cluttered background (for example, indoor and outdoor environment) and camera motion, insufficient annotated data, and uneven predictability (some frames are more representative than others).

However, the current situation indicates that the domain of computer vision is about to make huge progress in the next few years.

The most important reason is the astonishing evolution of computational power and computer architectures. Theis and Wong (2017) [20] present the opportunities created by the end of Moore's law: they see the physical limitations attained by the silicone transistor as an opportunity for developing new devices (like molecular or DNA transistors), technologies and computer architectures. Not to mention the efforts towards building accessible quantum computers [21] (that would replace the personal computers that we use today), which would change the perspective in which we solve computational problems nowadays.

Remarkable progress is also being made as far as hardware is concerned. Strictly speaking about hardware for performing costly computations, efforts towards widely offering computational power are made. An illustrative example is represented by the online-hosted Graphics

Processing Units (like NVIDIA GPU Cloud[1] or Google Cloud GPU[2]), or the more recent Tensorflow Processing Units [22]. The continuous development of specialized hardware for data acquisition should also be noted. As 3D cameras and real-time sensors become more powerful and accessible[3], developing HAR systems based on video streams becomes easier and cheaper.

3.2 Related Work

In this section we present some of the most performant HAR systems from technical literature, that take videos (with or without the depth component) as input. The criteria upon which we analyze the systems are chosen with regard to the purpose of our paper. Thus, we discuss the datasets that they use, the technologies that they use for activity recognition, and the accuracy of the method. For each method, we outline the advantages and disadvantages. We keep in mind the strong points and integrate them into our system and we try to avoid ending up with the same weak points.

Some of the discussed systems have the corresponding code available in public repositories. In those cases, we could reproduce these methods, and re-train them on our computer for supplementary analysis. We tested using a six-core Intel Xeon W-2135 processor with an NVIDIA GP107GL Quadro P1000 graphics card.

Ji, Xu, and Yang (2013) [23] design a real-world surveillance system, based on 3D CNNs (convolutional neural networks). The system architecture consists of stacking together the following layers: one pre-initialized layer with hard-coded kernels, three convolutional layers, followed by two subsampling layers, and one fully connected layer, that maps network outputs to activity classes. Their system manages to work in real-time, as well as to achieve 90.2% accuracy on the KTH public dataset [24]. It is fully automated and does not rely on hand-crafted features as input for the CNN. However, the method requires a long training time and a big dataset. This happens because it only relies on learning from one stream, namely the RGB stream.

Karpathy et al. (2014) [25] explore time fusion mechanisms for identifying human activities with 2D CNNs in large-scale RGB video datasets. They propose a system that is composed of two parallel CNNs, but having different inputs. They call these two inputs *fovea stream* and *context stream*. The first one consists of video frames downsampled to half of their original size, and the second is obtained by cropping only the middle region of the video frames. Combining with transfer learning, their system scores 63.3% on UCF-101 dataset [26]. However, their time fusion model is not very effective, compared to zero time fusion (the accuracy growth is of only 1.6%), which suggests that the used CNN architecture is not optimal.

[1] https://www.nvidia.com/en-us/data-center/gpu-cloud-computing/. Last accessed: 22 June 2019.
[2] https://cloud.google.com/gpu/. Last accessed: 22 June 2019.
[3] https://rosindustrial.org/3d-camera-survey. Last accessed: 22 June 2019.

Further on, Khaire et al. (2018) [27] make use of fusion mechanisms applied on RGB-D stream, and design a HAR system that uses five parallel CNNs, each functioning on one compressed image, obtained from a video. They pre-train the CNNs on VGG-16 [28], thus reducing training time, by transfer learning. Their system is highly accurate, scoring 95.38% accuracy on UTD-MHAD dataset [29]. They also investigate different mechanisms of combining the results obtained from the 5 CNNs (average or product rule). The product rule appears to be more accurate (yielding an improvement of 1.56%) One of the key points in their system is to combine RGB with the available depth information, thus making use of all available channels. However, the method for compressing skeleton data is computationally expensive. We note its advantages, but try to think of a more lightweight approach for compressing skeletal joints.

Chéron, Laptev, and Schmid (2015) [30] propose a method of summarizing video information through a pose-derived method. They gather together information related to movement, body parts and time in a so-called *"pose-based convolutional neural network descriptor"*. They extract these features from the RGB stream and from the optical flow of a video. Then, they use the P-CNNs as input to train a SVM (Support Vector Machine) classifier. The reported accuracies are 73.4% on the JHMDB-GT [31] dataset, and 60.8% on the MPII Cooking-Pose dataset [32]. After shuffling the dataset split lists, we also tested the corresponding code for the HAR system [4] on our personal computer. We obtained an accuracy of 72.8% on the first dataset, and one of 61.1% on the second one. Our results correspond to the ones indicated in the paper. However, even while using transfer learning, the training time was long (seven hours), mainly because of the expensive computation of the optical flow.

The work of Wu et al. (2017) [33] proves the enormous efficiency advantage of compressing videos into images, before passing them through CNN mechanisms. Instead of training the neural network on raw videos, they train it directly on compressed videos (with a compression mechanism like H.264 or HEVC). Their method has high accuracy (while using optical flow, 94.9% on UCF-101 dataset [26] and 70.2% on HMDB-41 [34], and 90.4%, respectively 59.1% without using optical flow). Then, even more importantly, it has high efficiency: 4.6 times faster than the popular Res3D [35]. Using the public code available[5] and removing optical flow computation, we shuffled datasets and performed similar experiments (91.3% on the first dataset and 58.4% on the second one). This HAR system proves the importance of fusion mechanisms and the speedup that is obtained if we classify images instead of videos.

To sum up, using convolutional neural networks as a mechanism for detecting class activities is used in the majority of papers. Then, using several information streams (RGB, depth, and skeleton) helps in improving overall accuracy. In addition, making use of transfer learning can reduce training time considerably. Finally, using compression/fusion mechanism to reduce the dimensionality of a video from 3D to 2D, turning it into an image, helps in attaining a good efficiency rate, as far as time and memory are concerned.

[4] https://github.com/gcheron/P-CNN. Last accessed: 21 June 2019.
[5] https://github.com/chaoyuaw/pytorch-coviar. Last accessed: 21 June 2019.

3.3 Research Directions

While analyzing technical literature, we identified three fundamental research directions, as far as human activity detection in videos is concerned: mechanisms for summarizing information present in videos, the use of convolutional neural networks for image/video analysis, and the shift towards the automated machine learning paradigm. We present each in the following subsections.

3.3.1 Fusion Mechanisms

We define a *fusion mechanism* as a procedure of combining multiple pieces of information of the same type, to obtain a compressed (way smaller), but as comprehensive as possible (ideally, no information is lost after the compression), representation of that information. Fusion mechanisms are required in CNN architectures because they help in reducing the size of the input (either removing one dimension or shortening the sizes of the existing dimensions).

Next, we describe fusion mechanisms that are popular in technical literature, in the context of activity detection. We group them based on the nature of the data that they combine.

3.3.1.1 Temporal Fusion

The concept that makes videos different from images is time. A video represents a sequence of chronologically stacked images (frames). Thus, in order to preserve its nature, video classification methods have to include time in the classification process.

According to Karpathy et al. (2014) [25], we distinguish the following temporal fusion mechanisms:

(a) **Single frame**: As the title suggests, this method does not take into account the temporal dimension, ignoring it throughout the classification process. It consists of extracting only one frame, f_k, from the whole video, and then identifying the action only based on that image.
As the video is represented only by an image, the classification is done with image mechanisms, like classical training classifiers (Support Vector Machines [36], k-Nearest Neighbour [37], Bayesian models [38]) or deep learning techniques [39].
The frame f_k is selected either in a simple manner (ex: $k = \text{rand}(1, n)$, or $k = \text{ceil}(n/2)$), or following a complex heuristics (Grycuk, Knop, and Mandalfind (2015) [40] describe a method for selecting the most relevant keyframes in a video).
Table 1 lists the strong and weak points of the mechanism.

(b) **Late fusion**: The idea behind the late fusion model is to select a couple of frames and to pass them through the same network, one after another. Thus, the selected frames

Table 1: Advantages and disadvantages of zero temporal fusion

Advantages	**Disadvantages**
Short video pre-processing time	Does not take into account the temporal dimension of the videos
Short computational time	Strongly dependent on the frame selection heuristics
Efficient if the frame is conveniently selected	

are treated independently, until almost the end of the classification process, when the network results are combined.

This approach is designed to overcome the problem of getting classifications only based on a single video frame: the intuition is that several results might converge to a correct one. Similarly to the previous method, the frames can be chosen either randomly, or by finding the key frames of a video. Strong and weak points of this approach are listed in Table 2.

Table 2: Advantages and disadvantages of late fusion method

Advantages	**Disadvantages**
Takes into account the temporal dimension	Long computational time (several consequent network passes are needed)
Short video pre-processing time	Dependent on the frame selection heuristics
Efficient if the frames are conveniently selected	In case of transfer learning, it requires changing the architecture of the pre-trained model (i.e. ending FC layers)

(c) **Early fusion**: As the name suggests, the fusion of temporal information with spatial information is done as soon as possible, this being directly on the pixel level. The method offers a way of encoding all temporal information, throughout the whole video, if needed, or throughout the whole selected sequence. If a video is too long, we might want to crop only the middle part, or only the most relevant part, or compute the synopsis of that video. An example of a method for computing the synopsis of a video is propounded by Panagiotakis, Ovsepian, and Michael (2015) [41].

Depending on the type of data that is being merged, different pixel-level fusion algorithms can be used:

- *Motion history images* [10] (MHI) are numerical representations capable of encoding the movements of silhouettes in videos. They do not depict the trajectory, but the type of movement. For example, by analyzing MHIs it is possible to distinguish between actions with the same start and end coordinates, such as opening and closing a door. They are widely used in computer vision, and more precisely in activity detection [27, 42, 43].
- *Binary motion images* (BMI) are developed by Dubhal et al. (2015) [44] and represent a way of compressing a video into a 2D image, by representing motion change after background removal of every frame. This processing is done by

applying a Gaussian Mixture Model. It is noise resistant but requires more time to be computed than the previously mentioned MHI.

- *Optical flow* [45] identifies a pattern of apparent motion of people and objects from a video, considering relative motion having an observer and a scene. It is widely used in human activity detection [46,47], being accurate, but quite computationally expensive, and environmental conditions dependant.
- *Depth motion maps* (DMM), firstly introduced by Yang, Zhang, and Tian (2012) [48], are a way of encapsulating motion information extracted from depth data.
- *Skeleton images* represent a visual summary of skeleton information present throughout a video. They are introduced by Khaire et al. (2018) [27].

Advantages and disadvantages are listed in Table 3.

Table 3: Advantages and disadvantages of early fusion method

Advantages	**Disadvantages**
Takes into account the temporal dimension	Video pre-processing time is required; depending on the complexity of the chosen method, this can produce overhead.
Not dependent on frame selection	
Can use a pre-trained network model	

(d) **Slow fusion**: This method consists if starting to classify using several independent networks, and then gradually merging them, by connecting outputs from different networks to the same layer.
Advantages and disadvantages are listed in Table 4.

Table 4: Advantages and disadvantages of slow fusion method

Advantages	**Disadvantages**
Takes into account the temporal dimension	Hard to use a pre-trained model (the architecture is complicated, has to be specific, designed according to the problem.).
Not dependent on frame selection	Computationally expensive (some frames are considered as part of the input of multiple networks)

The previous classification is represented in Figure 1.

An alternative to using a temporal fusion mechanism would be to use a 3D CNN on a stack of frames, or a long-short-term-memory mechanism. For example, Baccouche et al. (2011) [49] design a HAR system using a sequential deep learning mechanism: the first component is a 3D CNN that learns spatiotemporal features present in videos, and the second one is an RNN (recurrent neural network) trained to classify smaller sequences of frames. The RNN contains a hidden layer of LSTM (long-short-term-memory) cells. Another example is provided by Arunnehru, Chamundeeswari, and Prasanna Bharathi (2018) [50]: they classify human

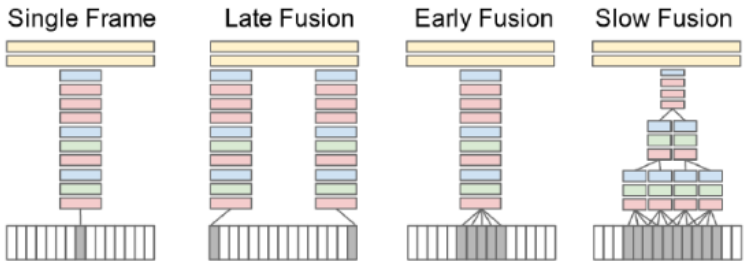


Figure 1: Karpathy et al. (2014) [25]. Time fusion mechanisms

activities by using a 3D CNN with a 3D motion cuboid (stores temporal difference among several consecutive frames).

3.3.1.2 Channel Fusion

In RGB-D videos, it is obvious that multiple information channels are involved: these channels form the spatial dimension of the video and are most often identified as RGB, depth and skeleton data. When performing activity recognition tasks on RGB-D videos, it is natural to apply different classification strategies on them, and then combine the scores. Merging multiple channels of information yields a richer, more accurate final result because they have different natures and illustrate different features.

Channel fusion mechanisms can be regarded as *decision-level* fusion mechanisms, because they are applied in the final steps of the classification process. Roughly speaking, a channel fusion mechanism is a black box that takes several arrays of class scores as inputs and returns a unifying array illustrating them all.

Let $C = \{c_1, ...c_n\}$ be the set of classes used in training the model and $\mathbb{S}$ the set of likelihood scores produced by the different streams used as input for the model. A class probabilities array $S_k \in \mathbb{S}$ contains a corresponding score for each class:

$$S_k = \begin{bmatrix} score^k_{c_1} \\ \vdots \\ score^k_{c_n} \end{bmatrix} \tag{1}$$

The goal of the channel fusion mechanism is to combine all elements of $\mathbb{S}$ in only one array of scores, S_f. A convenient function φ is chosen and used in merging the scores corresponding to each class.

$$S_f = \begin{bmatrix} \varphi(c_1) \\ \vdots \\ \varphi(c_n) \end{bmatrix} \tag{2}$$

The problem of making an only decision from multiple results has been analyzed for decades [51–53]. We further analyze some of the popular methods that can be used for combining the

outputs of multiple classifiers (i.e. for defining function φ), along with their advantages and drawbacks:

Ho, Hull, and Srihari (1994) [51] present the following methods for combining decision made by several classifiers. These solutions can be regarded as voting mechanisms, because they work on yes-no decisions, and not on arrays of probabilities.

- *The Highest Rank Method*: maybe the most intuitive combining mechanism, it consists of counting votes for each class, and choosing the one with the highest number of votes. It is recommended for merging a small number of classifiers, classifying in a big number of classes, each of them being specialized on a certain data type. Its drawback is that it can result in many ties.
- *The Borda Count Method*: it is a generalization of the above method, that measures the amount of agreement between classifiers. It treats all classifiers as equal, but assumes that they are independent and that they contribute in an equal manner. It consists of assigning a Borda count score to each class, and then arranging the classes in descending order, according to their Borda count scores.

$$Borda\_score(c) = \sum_{S_k \in \mathbb{S}} |x \in C | score_x^k < score_c^k| \tag{3}$$

Another interesting idea presented by Srihari et al. (1994) [51] is that different sets of classifiers can cooperate. They call this concept *"multistage combination"*, meaning that classifiers can be grouped, and different channel fusion mechanisms can be applied to each group, depending on their nature.

For combining arrays of class scores instead of decision, Duin (2002)[52] proposes several fixed combining rules:

- *The Product Rule*: used only for combining the outputs of independent classifiers, with the same importance in the classification process; assumes accurate likelihood scores, with no noise. Its drawback is that it fails for small (or zero) values.

$$\varphi(c) = \prod_{S_k \in \mathbb{S}} score_c^k \tag{4}$$

- *The Sum Rule*: used only for combining the outputs of independent classifiers, with the same importance in the classification process; can help in noise reduction.

$$\varphi(c) = \sum_{S_k \in \mathbb{S}} score_c^k \tag{5}$$

- *The Maximum Rule*: usually used in merging a small number of classifiers, defined over a big range of classes, each classifying a different type of inputs. To mention its weak points, it may result in many equal scores, and it also not accurate when mixing

over-trained classifiers with weak ones.

$$\varphi(c) = \max_{S_k \in \mathbb{S}} (score_c^k) \tag{6}$$

- *The Median (Average) Rule*: behaves in the same manner as the summation rule but may generate more accurate results. Can be used for classifiers with the same importance, being independent or not.

$$\varphi(c) = \frac{\sum\limits_{S_k \in \mathbb{S}} score_c^k}{|\mathbb{S}|} \tag{7}$$

Then, Triantaphyllou, Shu, Nieto Sanchez, and Ray (1998) [53] extend the combining rules, by allowing more customization through weights usage:

- *The Weighted Sum Model*: same as above, but allowing each classifier to have a different influence on the result (classifiers that are thought to have a greater accuracy will have a bigger weight).

$$\varphi(c) = \sum_{S_k \in \mathbb{S}} score_c^k \cdot weight_k \tag{8}$$

- *The Weighted Product Model*: same as above, but allowing each classifier to have a different influence on the result (classifiers that are thought to have a greater accuracy will have a bigger weight).

$$\varphi(c) = \prod_{S_k \in \mathbb{S}} (score_c^k)^{weight_k} \tag{9}$$

3.3.1.3 Context Fusion

Previously described methods only analyze the motion induced by a certain action. Although this is generally an efficient method on its own for classifying actions, it cannot distinguish between different actions that are based on the same movement patterns. For example, throwing a ball or throwing a tossed paper rely on the same motion, as well as holding a laptop, a book or a piece of paper.

The above-mentioned problem can be solved by including information about the surrounding objects in the classification model. For example, if a guitar is being identified in most of the frames of a video, the score for class "playing the guitar" would be increased.

For example, Ramoly et al. (2017) [54] make use of semantics in the reasoning process, by building an ontology with terms that might point to a certain human activity. Further on, Baradel et al. (2018) [55] design a model that is capable of reasoning on actions happening in a certain video, producing semantically coherent output. It does not only take into account

the spatial dimensionality, but also the temporal one. Also worth mentioning is the work of Ihianle et al. (2018) [56], where a key idea in recognizing human activities is the association of object usage to daily activities. Thus, their system uses object identification and object interaction, in order to identify activities.

A notable remark is, however, that this mechanism diminishes the degree of autonomy of the HAR system. Choosing the rule for checking for the *logical correlation* between a certain object and a class requires human intervention, at least in the first stage. Of course, even these correlations can be learned with artificial intelligence techniques (ex: linear regression, decision trees), but additional effort is required. In practice, one can skip the context fusion mechanism completely, or can pre-establish a list of class-object associations.

3.3.2 Convolutional Neural Networks

With the growing popularity of neural architectures, a large number of human activity detection systems are based on convolutional neural networks [23, 25, 27, 49, 50, 57].

Convolutional neural networks are deep learning algorithms widely used in image (video) analysis. According to Haykin (2009) [58], CNNs are *"designed specifically to recognize two-dimensional shapes with a high degree of invariance to translation, scaling, skewing, and other forms of distortion"*.

During the training process, a 2D CNN takes as input training images, assigns some parameters (i.e. weights w and biases b) to different characteristics of the images (low-level ones, like lines or dot patterns, or high-level ones, like entire objects), and then adjusts these parameters over several epochs. After training is over, a new input is fed in the network and the output is obtained using the pre-computed parameters.

Using the same notations as Ji, Xu, Yang, and Yuhe (2013) [57] and including biases and ReLU (rectified linear unit) activation function, the formula for computing the value of a convolution in an image becomes:

$$v_{ij}^{xy} = \max(0,\ b_{ij} + \sum_{k} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} w_{ijk}^{pq} \cdot v_{(i-1)k}^{(x+p)(y+q)}) \tag{10}$$

where:

v_{ij}^{xy} = value at position (x, y) in the j^{th} map from the i^{th} layer
b_{ij} = bias for the j^{th} map from the i^{th} layer
k = index over the maps from the $(i-1)^{th}$ layer
w_{ijk}^{pq} = weight at (p, q) in kernel (of size $P_i \times Q_i$) of the j^{th} map from the i^{th} layer

3D CNNs differ from 2D CNNs in the sense that they do not only capture the spatial dimension, but also the temporal one. Thus, network input is now represented by a video, i.e. a sequence

of images distributed over time, instead of a single image. The mathematical formula, obtained similarly to formula 10, is:

$$v_{ij}^{xyz} = \max(0,\ b_{ij} + \sum_{k} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijk}^{pqr} \cdot v_{(i-1)k}^{(x+p)(y+q)(z+r)}) \tag{11}$$

Notations for formula 11 are similar to the ones used in 10, with the difference that this time, the kernel size is $P \times Q \times R$.

To sum up, the main advantages of CNNs over other types of deep learning models are:

- minimal pre-processing required
- automatic feature extraction
- through the pooling operation, the number of parameters is reduced, thus achieving a lower risk to encounter over-fitting, and reducing the dimensionality of the input
- a good ability to generalize.

3.3.3 Automated Machine Learning

As previously shown, neural network architectures are nowadays acknowledged as being more powerful than classical machine learning techniques, and extremely useful in activity detection problems. However, designing neural networks is hard and requires a lot of effort. The main problems are:

(i) Optimal design: designing a neural network consists of trying several configurations (i.e. changing the number and types of layers), training the network every time, and then deciding which architecture behaves the best according to some performance metric (e.g. accuracy, loss).

(ii) Hyperparameter tuning: apart from the architectural configuration, neural network components have hyperparameters (like the number of filters, their dimensions, their corresponding strides) that also have to be found, either once again by trial and error, or by inspecting similar architectures from the specialized literature.

The paradigm called *automated machine learning* tries to solve these problems. Its underlying idea is simple: make use of machine learning to automatically generate the optimal machine learning model. Putting this into perspective, the need for machine learning expertise is replaced with greater computational power: as hardware resources become more powerful, it looks like a natural solution to automatize the tedious process of finding ML models by trial and error.

Neural architecture search (NAS) is the process of automatically finding optimal architectures and hyperparameters for a given problem, by trying several solutions from a pre-defined search space.

Developing architectures using the NAS method has started gaining popularity recently: according to Wistuba, Rawat, and Pedapati (2019) [59], the first that started putting efforts into this method are Zoph and Le (2017) [60]. Further on, efforts have been done in this direction, and nowadays, especially in image classification and object detection, architectures found using NAS methods outperform handcrafted ones (Elksen, Metzen, & Hutter, 2018 [61]).

Figure 2 contains a self-explanatory visual representation of the concept: a search strategy is repeatedly chosen from a certain search space, trained, and tested using a performance metric.

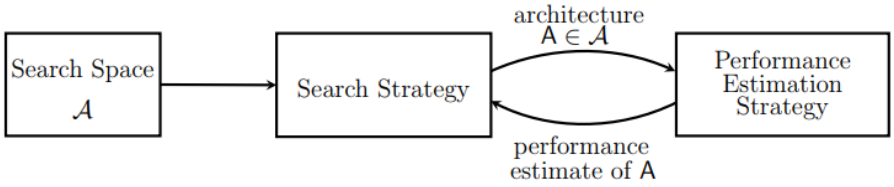


Figure 2: Elksen et al. (2018) [61]. Abstract illustration of NAS methods

The performance metric is used by an optimization algorithm, that allows the NAS mechanism to converge to an optimal network architecture. Some of the most popular examples are:

- Reinforcement learning: Baker, Gupta, Naik, and Raskar (2018) [62] and Zhong, Yan, Wu, Shao, and Liu (2017) [63] use a Q-learning algorithm, while Zoph, Vasudevan, Shlens, and Le (2017) [64] and Cai, Chen, Zhang, Yu, and Wang (2017) [65] use a policy iteration algorithm, with gradient descent.
- Genetic algorithms (Wistuba, 2018 [66], Liu, Simonyan, Vinyals, Fernando, and Kavukcuoglu (2017) [67]).

3.4 Chosen Solutions

This section describes the solutions that are going to be used in the development of our HAR system, along with arguments that motivate our choices. We decided to use fusion mechanisms (time, channel and context), 2D CNNs trained on images and an automated machine learning mechanism: Google's AutoML.

3.4.1 Fusion Mechanisms

We use fusion mechanisms, mainly because they help in avoiding networks with large inputs; this is considered an advantage, because large inputs yield networks with more parameters. Consequently, these networks require more training data and more computation time. Plus, bigger inputs do not necessarily determine better results. This happens because important features of the dataset can also be captured in compressed formats if computed efficiently.

Given the previous analysis, as well as the purpose of the HAR system, we chose to use the following fusion mechanisms.

The **temporal fusion** mechanism that was chosen for the HAR system is the *early fusion mechanism*. In other words, time is considered as soon as possible in the classification process, i.e. at pixel level; We believe that the temporal component is vital in HAR problems, because it represents the main difference between images and videos. Consequently, we believe that it would not be sufficient to be considered in later stages (during or after the video is run through the neural networks). Plus, this method is not dependent on any frame selection heuristics (like the single/late frame approach, for example), so it ensures more confidence, without restricting the system designer.

More specifically, we build the temporal fusion mechanism by using: MHI for compressing RGB data, DMM for depth data, and skeleton images for skeletal information.

We use **MHIs** as RGB information synthesis method because they:

- represent video sequences in a global manner (compared to computing the synthesis only from a couple of frames)
- do not take into account only the spatial dimension, but also the temporal one
- are computationally simple and fast
- do not rely on assumptions (ex: an alternative to MHI is optical flow; when computing it, one has to assume that video brightness is constant, and that motion changes are slow [45]).

We use **DMMs** as depth information synthesis method because they:

- represent video sequences in a global manner
- take into account the temporal dimension
- are computationally simple and fast
- are environment invariant (ex: do not depend on lighting conditions)
- are useful when subjects are facing the camera (else, projecting the 3D points on 2D planes might yield to inaccurate results).

Finally, **skeleton images** are a suitable way of summarizing skeletal information because they:

- represent video sequences in a global manner
- take into account the temporal dimension
- are computationally simple and fast
- can be customized easily according to needs.

Apart from the temporal fusion mechanism, we use a **channel fusion** mechanism with a mathematical rule, and a **context fusion** mechanism, where we use contextual objects to make logical assumptions about the performed activities.

3.4.2 2D CNNs

The proposed HAR system uses convolutional neural networks to classify images (illustrating a whole video sequence). As the previous subsections showed, these are the most popular mechanisms for identifying visual elements in a robust unsupervised manner.

Even if 3D CNNs allow more accurate results, they are computationally expensive. Only by comparing formula 10 with formula 11, the computational overhead is visible: there is summation over an extra dimension (R). Because 3D CNNs allow input with one extra dimension (time, in case of videos), they consequently have way more parameters to learn (weights and biases). Training time, as well as classification time, increases considerably. The extra dimension is also reflected in the input taken by the neural network: we have to pass 3D input (i.e. a sequence of frames), instead of 2D input (an image, i.e. a matrix of pixels).

We remind the reader that the purpose of this paper is to build an *effective* HAR system. Consequently, our HAR system uses 2D CNNs and classifies computed images, instead of the full video. We argue that using fusion mechanisms and 2D CNNs is more efficient, and almost as (if not even more) accurate than using raw input and 3D CNNs.

3.4.3 Google's AutoML

In this paper, we use the neural networks generated by Google's Auto ML to compute individual arrays of class scores on each of the four data streams (RGB, depth, skeleton and context). The idea behind this automated network generating algorithm is firstly described by Zoph and Le (2016) [60], and is based on a neural architecture search mechanism with a policy-oriented optimization algorithm.

A recurrent neural network (RNN)—the *controller*—is used to generate candidate models (i.e. choose the hyperparameters) of a neural network. After each model is generated, a *child network* is trained, and the accuracy of the just trained model is returned to the controller. The controller updates its policy of generating architectures via reinforcement learning (based on the accuracy received from the child). Of course, the goal is to maximize the expected accuracy.

This process of trying different architectures until the best is found is accelerated with parallelism and asynchronous updates but is still extremely computationally expensive. It is made practical by Zoph et al. (2017) [64], who combine it with transfer learning, and also design a restricted search space for the controller (called *NASNet search space*). Plus, it has integrated data augmentation techniques.

NASNet yields outstanding performances. Figure 3 illustrates its accuracy vs. the computational power required, compared to other State of the Art methods. For the same amount of resources, NASNet outperforms all other methods.

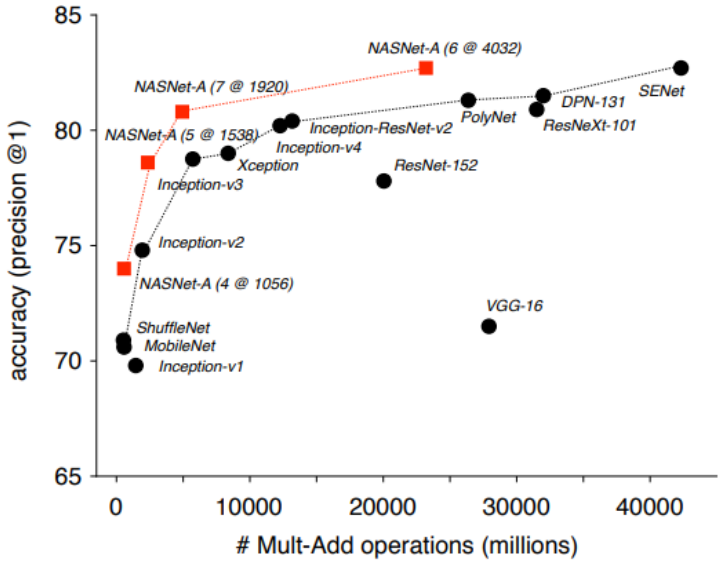


Figure 3: Zoph et al. (2017) [64]. Accuracy versus computational demand across top performing published CNN architectures on ImageNet dataset [68]

We decide to use automated machine learning to generate and train the needed convolutional neural networks for the HAR system, because of its outstanding performances (see Figure 3), as well as the convenience of using an AutoML technique instead of a hand-crafted one.

3.5 Chapter Summary

In this chapter we firstly identified the general context of the activity detection problem nowadays in section 3.1, then named performant existing mechanisms section 3.2, and outlined three of the most important research directions in this domain, in section 3.3.

Finally, in section 3.4 we name the concepts that we are selecting as appropriate for our own HAR system: the system contains all types of fusion mechanisms (temporal, represented by MHI, DMM and skeleton image, channel and context), its core is represented by 2D CNNs and we use an automated machine learning technique for training, namely Google's AutoML.

4 DESCRIPTION OF THE PROPOSED SYSTEM

In this chapter we present an overview of the activity recognition system that we designed. Firstly, the input data streams schema is conferred. Then, the architecture is detailed. Thirdly, we depict the system data flow. Finally, we emphasize its modularity.

4.1 Input Data Streams

The system uses an RGB-D camera as data acquisition hardware, so the two raw streams that come from the camera are a sequence of RGB frames and a sequence of depth frames. Out of the RGB frames, contextual data (i.e. a list with objects identified in certain frames of the video) and skeletal data (i.e. positions of the body joints of people identified in each frame) are computed. Then, out of the depth frames, we extract three projections (front, side, and top) for each frame. This way, we obtain all the streams on which the HAR system relies: RGB data, depth data, skeletal data, and context data.

Figure 4 shows the logical factorization of video information, namely the data streams.

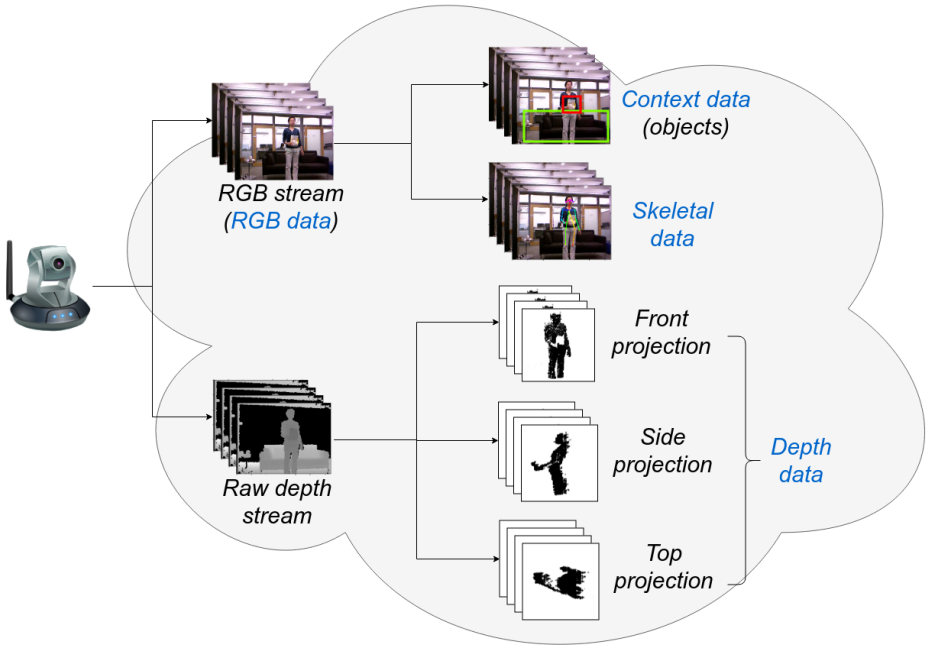


Figure 4: Input data streams extracted by the data stream manager component

4.2 Architecture

Roughly speaking, the system consists of applying temporal fusion mechanisms at pixel-level, then passing each fused stream through one specialized CNN, combining the predictions with channel fusion mechanisms, and finally adding relevant context via a context fusion mechanism.

The architecture is firstly presented from a high-level perspective, in Figure 5, and then each component is described in a detailed manner, following the data flow order, in Figure 4, Figure 6, Figure 7 and Figure 8.

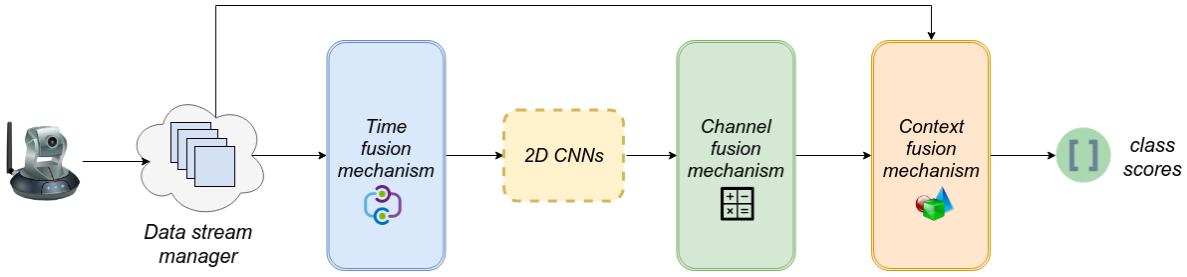


Figure 5: Full architecture of the proposed activity recognition system

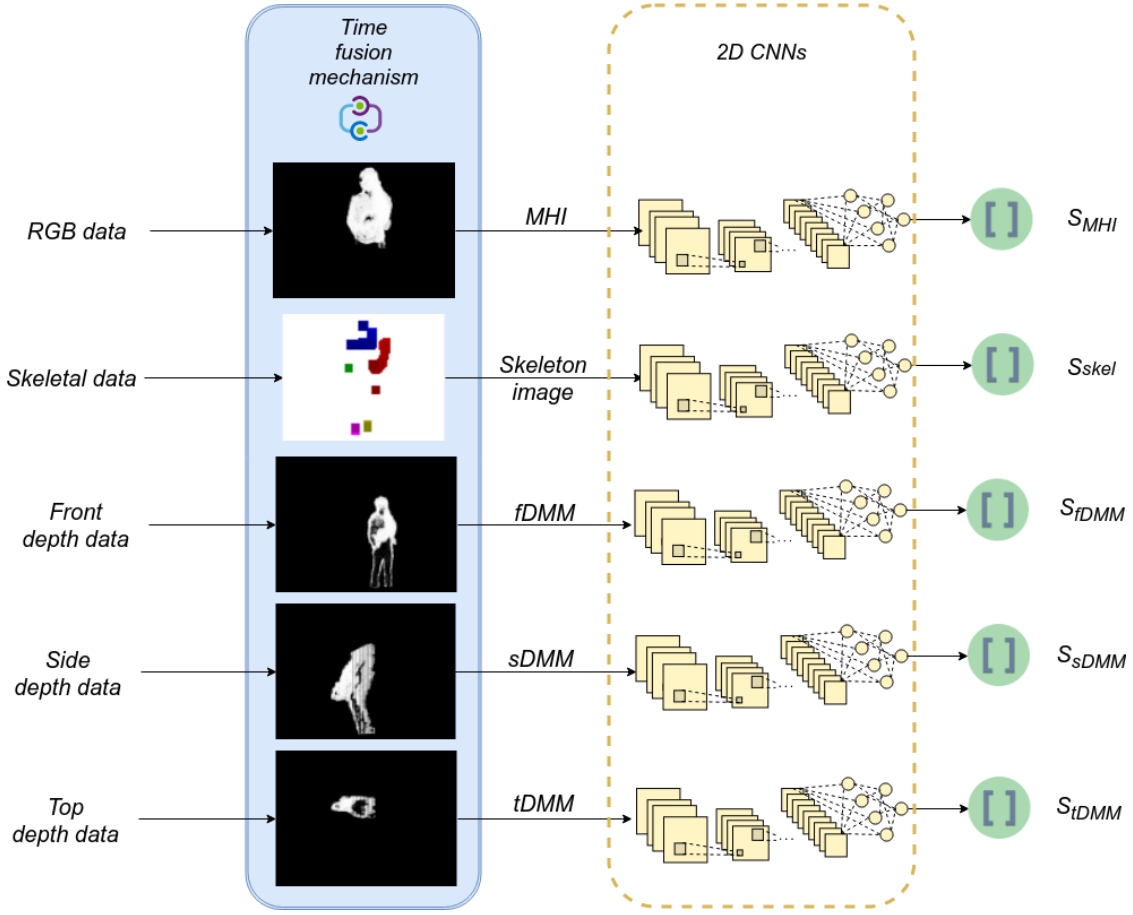


Figure 6: Detailed architecture of the time fusion mechanism and linkage with the CNNs

The main components of the system (i.e. the time fusion mechanism and the 2D CNNs) are depicted in Figure 6. The data streams from Figure 4 are each "compressed" with an appropriate time fusion mechanism (MHI [10] for RGB data, skeleton image [27] for skeletal data, and DMMs [48] for depth data), and then passed through a specially trained CNN. This means that each of the CNNs is trained with compressed images of a specific type (for example, only one CNN is trained to identify MHI images). The output of these two interconnected modules is represented by several class probabilities arrays (five arrays, if all available data is used).

Figure 7 shows the channel fusion mechanism for which we opted. The purpose of the channel fusion mechanism is to combine several different arrays with prediction scores into a relevant one.

Firstly, front, side and top DMMs are merged by averaging the arrays of scores element-wise. An average mechanism has been chosen, because all three motion maps are computed from the same information source (i.e. depth data), and with the same procedure (the only difference is made by the Cartesian planes on which we project the depth point cloud).

Then, the resulting scores are merged with the ones from the MHI and skeleton arrays, by using a weighted average formula. Weights allow setting a different relevance to each type of stream. We prefer applying a weighted average instead of a normalized weighted product, to avoid obtaining (almost) null values.

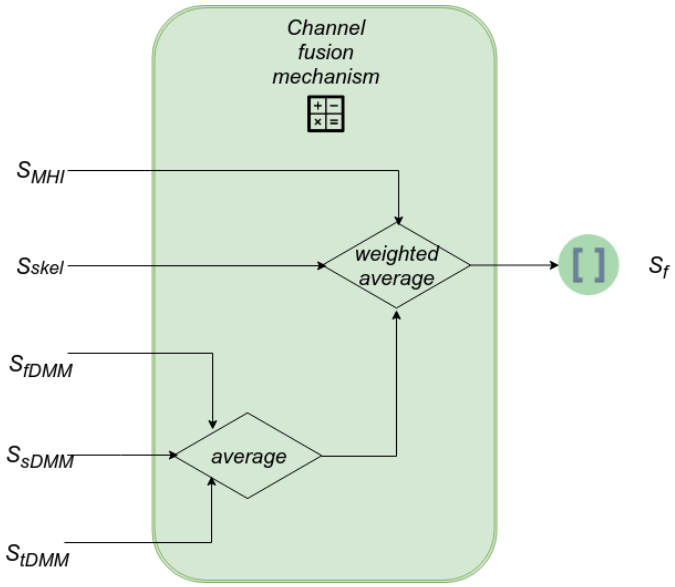


Figure 7: Detailed architecture of the channel fusion mechanism

The last step of the HAR system is represented by merging contextual data to the so far obtained result. More specifically, the objects that have been identified in a given video modify the scores outputted by the 2D CNNs. See Figure 8 for the block scheme.

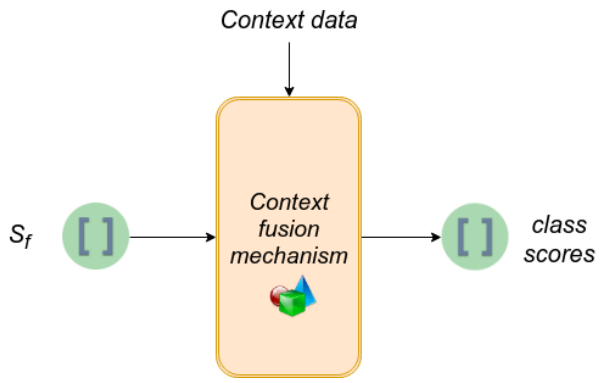


Figure 8: Detailed architecture of the context fusion mechanism

4.3 Data Flow Diagram

Figure 9 depicts the data flow of the HAR system when all its components are connected. The origin of one data flow is encapsulated into its color in the scheme, meaning that the same colors belong to the same flow in the system. In addition, colors that are close to each other represent data flows emerged from the same informational source.

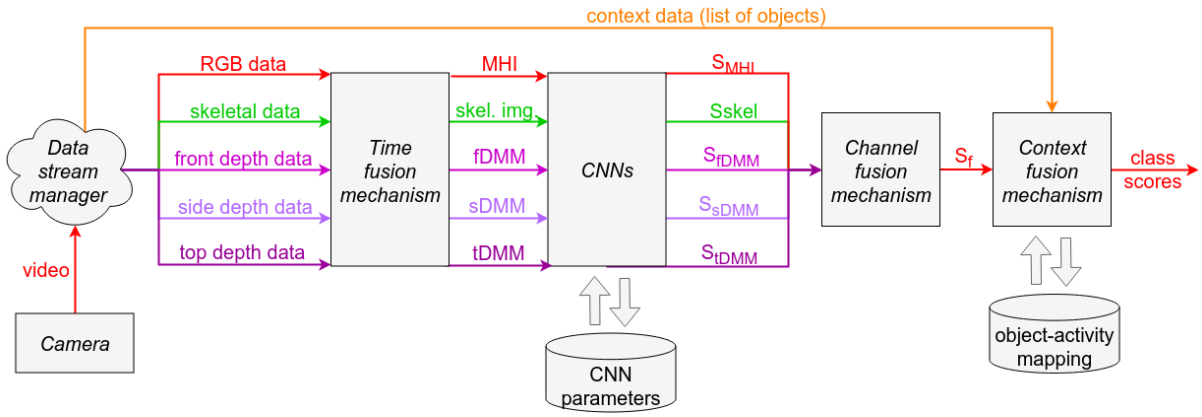


Figure 9: Data flow diagram of the HAR system. Each color represents a different informational source for the system. Colors derived from the same base color correspond to information streams that are obtained from the same raw stream.

The trained convolutional neural networks have their parameters stored in a dedicated database: the CNN parameters database. Similarly, the context fusion mechanism stores the pre-learned mapping of objects and activities in a database: the object-activity mapping database. For example, this is how the system knows that object "guitar" points to activity "playing the guitar". During the training phase, the information stored in the two databases is queried, but also updated. Afterward, while testing the system or while integrating it in practical applications, the databases are only queried.

4.4 System Modularity

As the architectural schema points out, the system is fully modularised, meaning that all sub-systems are isolated, and allow custom connectivity and usage. According to the degree of accuracy that one needs for the system, the following combinations can be built:

(i) *Minimal configuration*: use only the RGB stream and a time fusion mechanism that produces an MHI. Do not use any channel (there is only one channel considered) or context fusion mechanism. This solution (red data flow in Figure 9) has lower accuracy, but works only with RGB data, and has a shorter computation time.

(ii) *Custom configuration*: according to the purpose of the HAR system (i.e. types of activities that are intended to be classified), one can make different module combinations: MHI with skeleton image, with or without context data, MHI with DMMs, or only with front DMM, with or without context data, etc. This corresponds to any combination of differently-colored flows from Figure 9.

(iii) *Full configuration*: use all three fusion mechanisms, and all four data streams. This configuration is depicted in Figure 9, where every component is represented.

Apart from offering flexibility, the separate modules allow parallel computation. All five differently colored data flows from the middle of Figure 9 can be computed in parallel. Parallel computation helps increases the efficiency of the system in terms of speed.

4.5 Chapter Summary

In this chapter, we started by listing the data streams that represent the input of the HAR system, in section 4.1. Then, we described the architecture in section 4.2, and presented the data flow diagram in section 4.3. We ended the chapter by emphasizing the modularity and thus flexibility of the system, in section 4.4.

5 IMPLEMENTATION DETAILS

In this chapter we firstly introduce the technologies that have been used throughout the development of the HAR system. Then, we detail the algorithms used while implementing the fusion mechanisms. Lastly, we briefly describe the development process of the activity detection system.

5.1 Technologies

This section briefly describes all technologies used while developing the HAR system. We also present some of their advantages and disadvantages, thus motivating why we chose to use them.

As far as the programming language is concerned, all video computations have been written in **Python**. For video manipulation, we used OpenCV's `cv2`. For testing, we used `unittest` unit testing framework. Python was chosen due to the easiness of usage, reduced amount of code, prior knowledge, and range of available documentation.

For video data acquisition and manipulation, we used the following development kits:

- **OpenNI**[1]: is an open source framework that provides APIs for developing applications that use Natural Interaction. It forms a standard API for communicating with video and audio sensors, as well as with dedicated middleware (software that analyzes and understands this data). In particular, we used this framework for saving depth information, while filming our dataset, and for manipulating depth data from public datasets.

- **Orbbec Astra SDK**[2]: is the official software development kit for Orbbec cameras, provided by the manufacturers. It provides code snippets with examples on how to establish communication with Orbbec hardware, as well as some pre-compiled resources (color stream viewer, depth stream viewer, skeleton viewer, hand viewer, etc.). We used this resource while filming our dataset.

Then, for the development of the system, we used the following platforms:

- **Cloud AutoML**[3]: this Google product aims to make the design, training, evaluation, and utilization of custom machine learning models accessible for everybody. They pro-

[1] https://github.com/OpenNI/OpenNI. Last accessed: 16 June 2019.
[2] https://orbbec3d.com/develop/. Last accessed: 16 June 2019.
[3] https://cloud.google.com/automl/. Last accessed: 16 June 2019.

vide an end-to-end solution for solving supervised learning problems with different types of input (natural language, text in different foreign languages, video, images or tables). See Figure 10 for a schematic description.

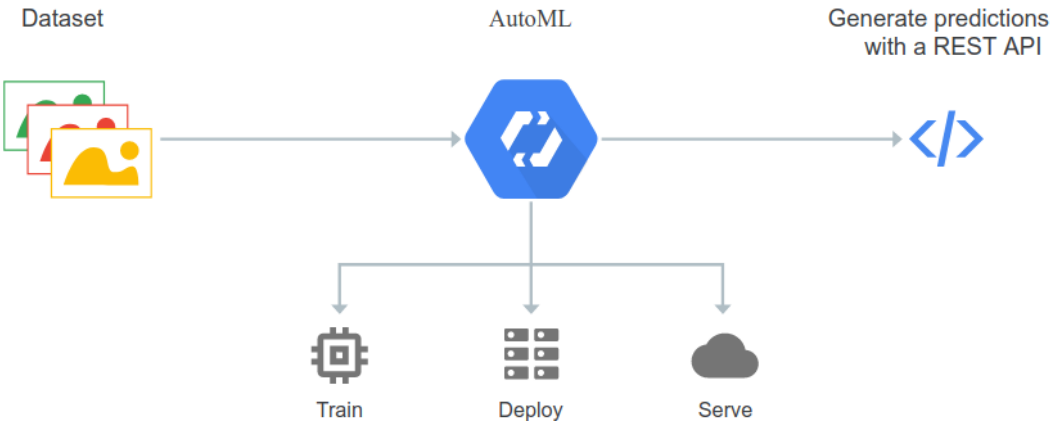


Figure 10: Functioning of Google Cloud AutoML[4]

The sub-products that we used are AutoML Vision (for training 2D CNNs for image detection) and AutoML Video Intelligence (for training 3D CNNs for video detection). For integration, we used the Python Vision API[5].
We chose this solution because of its convenience (offers buckets for data storage, specialized hardware for training the neural network, automatic mechanisms for finding an optimal network configuration, testing mechanisms, and a solution for hosting the model), as well as for its high accuracy. For finding the optimal architecture, we also tried Auto-Keras[6], its competitor, but the results weren't satisfactory. However, Auto-Keras outputs the found network architecture.
For our dataset dimensions, these solutions are free, but for large-scale datasets and prolonged training time, the product has to be paid. Another disadvantage is the fact that the architecture of the neural network cannot be inspected. However, the trained model can be downloaded, or hosted directly in the cloud.

- **Google Colaboratory**[7]: often shortened Google Colab, it is a free programming environment based on Jupyter notebook. Its main advantages are the following: it requires no setup, offers flexibility (additional resources can be installed just like on a Linux machine), runs everything on cloud and offers free (with the limit of 12 hours per day) GPUs and TPUs. Plus, it offers the possibility of presenting code interactively, along with explanations and interactive UI elements (sliders, bars, drop-down menus, etc.). Given all the above advantages, we considered Google Colab as the most suitable environment for development, testing, and deployment.

[4]© https://cloud.google.com/automl/. Last accessed: 17 June 2019.
[5]https://cloud.google.com/vision/docs/apis. Last accessed: 18 June 2019.
[6]https://autokeras.com/. Last accessed: 17 June 2019.
[7]https://colab.research.google.com/. Last accessed: 16 June 2019.

- **Google Drive**[8]: while using Colab, the most convenient solution for storing input data is naturally by using one of Google's products, namely Google Drive. We chose this product as a storage solution because it can be easily integrated with Google Colab, can be shared with collaborators, and it can be used free of charge (up to 15GB per account). However, when implementing a HAR system where data privacy matters, one must consider another storage solution, where user data is not available to the storage providers.
  To integrate Drive with Colab, one has to install a Drive FUSE wrapper (we used `google-drive-ocamlfuse`[9]), then generate authentication tokens for Colab and credentials for the Drive FUSE library. See Figure 11 for the full configuration.

```
# Generate authentication tokens for Colab.
from google.colab import auth
auth.authenticate_user()
```

```
# Generate credentials for the Drive FUSE library.
from oauth2client.client import GoogleCredentials
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} \
                        -secret={creds.client_secret} < /dev/null 2>&1 | grep URL
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} \
                        -secret={creds.client_secret}
```

Figure 11: Generation of authentication information while mounting Drive to Colab

Finally, we create a directory on Colab's local storage and mount Google Drive using that directory. Python code is available in Figure 12.

```
# Create a directory and mount Google Drive using that directory.
!mkdir -p /content/drive
!fusermount -u /content/drive
!google-drive-ocamlfuse /content/drive
```

Figure 12: Script for mounting Google Drive to Google Colab

We also used a series of libraries that perform sub-tasks required by the components of the HAR system:

- **YOLOv3** [69] for object detection: it represents the improved version of YOLO [70], a real-time object detection library. It offers high performance (in terms of accuracy and speed), and is easy to configure and use.
  Another important aspect that made us choose this library was the range of objects that it can identify. YOLO has modules trained on different datasets. The dataset that is of interest for out human activity detection system is COCO [71] dataset. Based on it,

[8]https://drive.google.com/. Last accessed: 16 June 2019.
[9]https://github.com/astrada/google-drive-ocamlfuse. Last accessed: 16 June 2019.

YOLO can detect 80 object classes. See Figure 13 for an example. Plus, in case other objects are needed, YOLO provides an easy mechanism for custom object detection (re-training the last layers of their model).

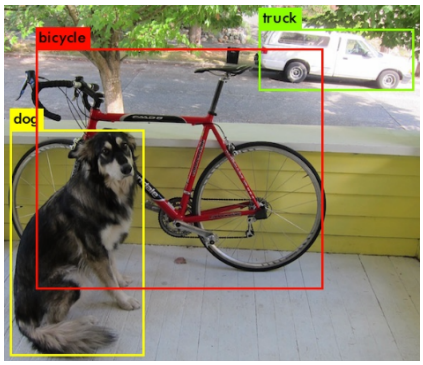


Figure 13: Example of some objects identified by the YOLO library[10]

As another significant advantage, it is open-source and can be used free of charge. This is a plus, compared to other object detection APIs, like Scale[11] or Vision API[12].

- **OpenPose** [72] for skeleton identification: it is a library that detects human body joints on images, in real time. Plus, it works with multiple people present in one image, and can also perform more complicated tasks, like hand or facial recognition.
  Being fast, open source and offering an API for integration in several programming languages (including Python), it represents the perfect solution for identifying skeletal joints used in computing skeleton images by the HAR system.

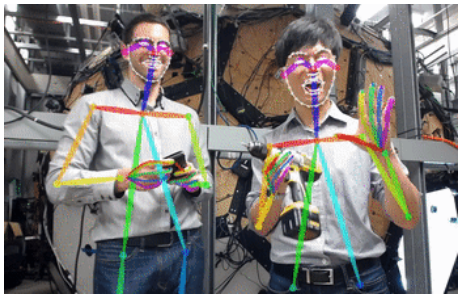


Figure 14: Example of body joints identified by the OpenPose library[13]

An alternative to using a library for inferring skeleton information would be to use skeleton information extracted directly from the camera/from its SDK. However, even if it might yield slightly higher accuracy, this restricts the range of cameras that can be used for the system, so we consequently did not choose this option.

[10] © https://pjreddie.com/darknet/yolo/. Last accessed: 16 June 2019.
[11] https://scale.ai/. Last accessed: 16 June 2019.
[12] https://cloud.google.com/vision/. Last accessed: 16 June 2019.
[13] © https://github.com/CMU-Perceptual-Computing-Lab/openpose/. Last accessed: 16 June 2019.

5.2 Fusion Mechanisms

While performing a classification task (in this case, activity detection on videos), all used fusion mechanisms must be combined to produce an array of final scores:

$$S_f = \begin{bmatrix} score^f_{c_1} \\ \vdots \\ score^f_{c_n} \end{bmatrix} \tag{12}$$

where $C = \{c_1, ...c_n\}$ represents the classes distinguished by the HAR system. After having computed S_f, the activity class of the input video is obtained by selecting the class that has the highest score:

$$C(video) = \underset{c \in C}{\operatorname{argmax}}\, score^f_c \tag{13}$$

In this section, we present the actual implementation of the fusion mechanisms used by the HAR system. We firstly present theoretical concepts and algorithms, and then, where needed, code snippets with relevant function calls.

5.2.1 Temporal Fusion

The following subsection briefly presents the mechanisms used for implementing the temporal fusion, i.e. MHI on RGB data, DMMs on depth data and skeleton images on skeleton joints.

5.2.1.1 MHI Computation

While computing the MHI, each pixel intensity is the result of a function of the motion that took place recently at that location. In Figure 15, a lighter value indicates more recent motion.

An MHI image, H^I_τ, has the same size as the input images, I. It is computed so:

$$H^I_\tau(x,y,t) = \begin{cases} \tau, & |I(x,y,t) - I(x,y,t-1)| > \delta I_{th} \\ \max(0, H^I_\tau(x,y,t-1) - 1), & \text{else} \end{cases} \tag{14}$$

where:

$I(x,y,t)$ = RGB value at position (x,y), at time point t
τ = the maximal time frame to be considered
δI_{th} = threshold value, empirically set

MHIs have been computed using `cv2`'s built-in function, `updateMotionHistory`. Thus, Figure 15 shows the procedure that is called for every frame, to encode motion information (in

(a) Couple of video frames

(b) MHI with τ=2 seconds, δI_{th}=32

Figure 15: Correlation between video frames and MHI for a person calling a cellphone

our case, in variable `vis`). Parameters `MHI_DURATION` and `DEFAULT_THRESHOLD` are set empirically, depending on the length of the video sequence that has to be compressed in an MHI.

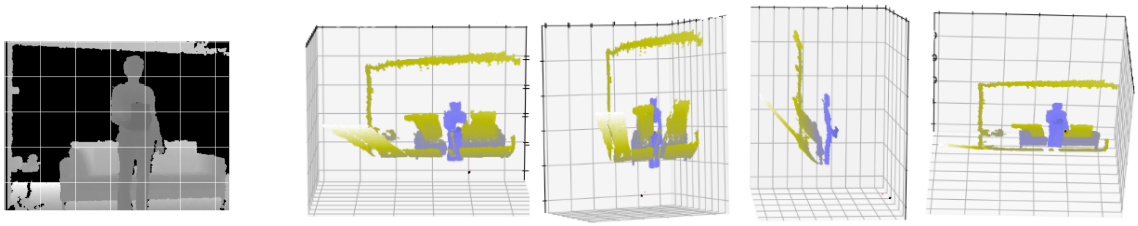
```
import numpy as np
import cv2 as cv

MHI_DURATION = 4
DEFAULT_THRESHOLD = 32

def compute_mhi(video_src):
    cam = cv.VideoCapture(video_src)
    _, frame = cam.read()

    h, w = frame.shape[:2]
    prev_frame = frame.copy()
    motion_history = np.zeros((h, w), np.float32)

    while cam.isOpened():
        _, frame = cam.read()
        frame_diff = cv.absdiff(frame, prev_frame)
        gray_diff = cv.cvtColor(frame_diff, cv.COLOR_BGR2GRAY)
        _, motion_mask = cv.threshold(gray_diff, DEFAULT_THRESHOLD, 1,
                                      cv.THRESH_BINARY)
        timestamp = cv.getTickCount() / cv.getTickFrequency()
        cv.motempl.updateMotionHistory(motion_mask, motion_history, timestamp,
                                       MHI_DURATION)
        vis = np.uint8(np.clip((motion_history - (timestamp - MHI_DURATION)) /
                               MHI_DURATION, 0, 1) * 255)
        prev_frame = frame.copy()
    cam.release()

    return vis
```

Figure 16: Python function for computing an MHI for an image. Code adapted from OpenCV documentation.[14]

5.2.1.2 DMM Computation

The first step in computing the DMM of a video is to extract the *depth point cloud* out of raw depth data. Raw depth data is data that comes directly from the depth camera: a corresponding float value for each pixel of each frame of the video. The point cloud represents

[14]https://github.com/opencv/opencv_contrib/blob/master/modules/optflow/samples/motempl.cpp. Last accessed: 16 June 2019.



(a) Raw depth data (b) Depth data point cloud, viewed from different angles

Figure 17: Visual comparison between 2D raw depth data and 3D depth point cloud

(a) Front view (b) Side view (c) Top view

Figure 18: Projections on the three Cartesian planes of the point cloud depicted in Figure 17

the conversion of raw data into corresponding 3D points. Figure 17 shows the conversion of raw values to a point cloud.

Given a raw depth value $d[x][y]$, the cloud point p_{xyz} is obtained as it follows:

$$p_{xyz} = d[x][y] \begin{bmatrix} (x - c_x)/f_x \\ (y - c_y)/f_y \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{15}$$

where:

f_x = focal length in the direction of X-axis
f_y = focal length in the direction of Y-axis
(c_x, c_y) = center point of the depth camera

At this moment, various effects can be applied to the point cloud. For example, depending on the input video and classifier purpose, one could want to normalize the cloud, remove background elements (i.e. remove from the cloud all points behind a certain threshold), or select only points from the central region, or from a bounding box isolating the person.

Next, as suggested by Yang et al. (2012) [48], we extract three 2D maps, being the projections on the three Cartesian planes: $proj_f$ on plane XOY (front view), $proj_s$, on plane XOZ (side view), $proj_t$, on plane YOZ (top view). See Figure 18 for a visual representation of these maps.

Projected points are obtained by multiplying 3D points with the corresponding projection

(a) Front view (b) Side view (c) Top view

Figure 19: Front, side and top DMMs for a person eating

matrix. For example, given cloud point p_{xyz}, the front view point p_{xy} is obtained as follows:

$$p_{xy} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} p_{xyz} = \begin{bmatrix} p_x \\ p_y \\ 0 \end{bmatrix} \tag{16}$$

Projection maps are stored in binary format. For instance, the XOY projection map is:

$$proj_f[x][y] = \begin{cases} 1, & \exists\, p_{xy} \\ 0, & \text{else} \end{cases} \tag{17}$$

Depth motion maps and are computed in a similar way to the MHIs for RGB data. However, instead of applying H_τ function on an RGB image, we apply it on the three projection maps. To reiterate the concept, DMMs are obtained by merging all frames of a projection map, according to consecutive changes in the projection maps. In practice, DMMs are computed using the following formula:

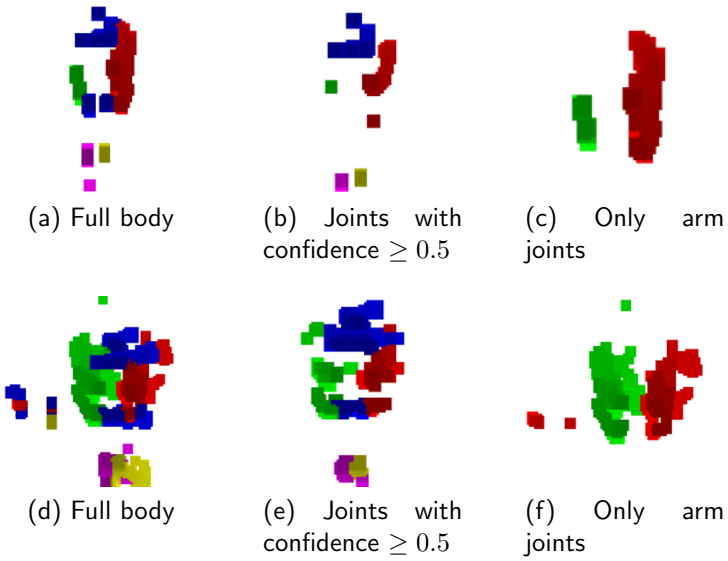
$$dmm_v = \sum_{k=2}^{nr.frames} (map_v^k \neq map_v^{k-1}), \quad v \in \{f, s, t\} \tag{18}$$

Figure 19 illustrates the three DMMs for a video of a person eating. We can see that the hand movement has been captured by the DMMs (light gray traces), as well as the silhouette slight movement, generated just by standing up (dark gray traces).

5.2.1.3 Skeleton Image Computation

A *skeleton image* represents the compression of the skeletal joints identified throughout all video frames. We encode the passage of time by decreasing the color intensity (i.e. by darkening it), and the skeletal information by depicting groups of joints that belong to the same body part with different colors.

We now propose a simple method for computing a skeleton image, inspired by the work of Khaire et al. (2018) [27]. This method can be applied on 2D skeletal joints and is not



(a) Full body (b) Joints with confidence ≥ 0.5 (c) Only arm joints

(d) Full body (e) Joints with confidence ≥ 0.5 (f) Only arm joints

Figure 20: Skeleton images of a person calling a cellphone (a-c), or cheering up (d-f)

computationally expensive. Other more elaborate methods like the one described by Liu, Akhtar, and Mian (2017) [73] or by Khaire et al. (2018) [27] might yield better results. However, using such a complicated method is beyond the scope of this paper.

Before generating the skeleton image, skeletal joints have to be divided into k groups, based on neighborhood and anatomy. To give an example, knee joint, ankle joint and toes joint could belong to the same group, which could be further divided based on whether the body part belongs to the left or right side. It is not mandatory to include all joints; the groups can contain only joints that are thought to be useful for the need of the application. See Figure 20, pictures (a) and (d), respectively (c) and (f). In practice, we use (c) and (f), if we know that all actions from the dataset are performed by users that only move their hands, or (a) and (d), if we expect the user to move their whole body.

Then, body joints identified in each frame are mapped to their group corresponding color, and a little region centered in that body joint is colored. As time passes, the color's intensity decreases. The size of the region, S and the intensity decrease factor, α are determined empirically. In our tests, α is usually set to 0.5.

A confidence factor, ζ, can be included if the method of identifying body joints is not highly trustworthy. Thus, only joints that are identified with high confidence will be taken into account while computing the skeleton image. In Figure 20, images (a) and (d), use a joint no matter what its confidence scores is, while (b) and (e) only use joints with a score $\geq 50\%$. Moreover, in (d) and (f), we can spot joint points on the right side that do not belong to the person performing the action. These points represent noise (another person moving in the background, or badly identified joints), that is removed with ζ.

For simplicity, the obtained skeletal image is of size 100×100. If a more accurate representation is required, the size of the image can be increased.

Algorithm 1 describes the pseudo-code for obtaining skeleton images.

Algorithm 1: Compute skeleton image

Data: $frames$, // Array of images representing the input video.
$G = \{g_1, \dots g_k\}$, // Groups that correlate chosen joints.
$colors = \{color_{bg}, color_1, \dots color_k\}$, // Color map for the new image.
α, // Controls how fast the color intensities decrease.
ζ, // Threshold for joint confidence.
S // Size of region that is colored by a single joint.

Result: $image$ // Matrix of size 100×100 representing the skeleton image.

```
1  begin
2      image ← image colored in color_bg;
3      for i ← 1 to |frames| do
4          for human ∈ get_humans(frames[i]) do
5              δ ← α · i/|frames|;
6              for joint ∈ get_body_joints(human) ∩ G do
7                  if confidence(joint) ≥ ζ then
8                      region ← region of pixels of size S × S, centered in joint;
9                      color ← colors[joint] · (1 − δ);
10                     set region to color;
```

5.2.2 Channel Fusion

The channel fusion mechanism is applied to the arrays of probabilities outputted by the temporal fusion mechanisms. We apply a weighted average between data obtained out of different types of sources and an average function over sources that are related. Thus, the rule for merging all the channels becomes:

$$\varphi(c) = \frac{1}{3} \cdot (s_c^{MHI} w_{MHI} + s_c^{skel\_img} w_{skel\_img} + \frac{s_c^{fDMM} + s_c^{sDMM} + s_c^{tDMM}}{3} w_{DMM}) \quad (19)$$

where:

φ = function that we apply on each component of the array of class scores
s = score
w = weight, expressed as probability; $\sum_x w_x = 1$
c = one of the classes that the HAR system is trained to classify

In practice, we implemented this mechanism in Python, using the `numpy.average` function.

5.2.3 Context Fusion

The idea behind the context fusion mechanism is the following: detect scene objects by selecting a couple of frames, and then feeding each of them into a 2D CNN trained for object detection. Each frame would produce a list of identified objects, which would yield to a total list of objects. Depending on the importance that we want to give to the context fusion mechanism, the final list object is obtained by reunion (for an aggressive mechanism) or intersection (for a light mechanism) between the two lists. Figure 21 depicts the mechanism.

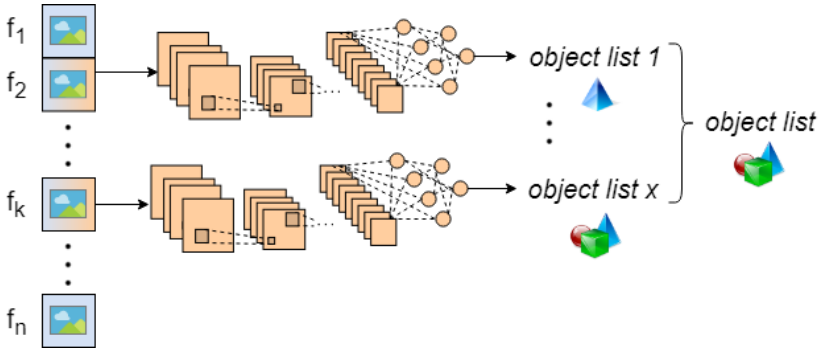


Figure 21: Process of obtaining the list of surrounding objects for applying an aggressive context fusion mechanism

The frame selection heuristics is not vitally important, mainly because an object defining an action is thought to be used throughout the action video. Thus, we select the frames using simple logic: out of a video, we extract 5 random frames.

The first approach that we tried when implementing the object detection mechanism was to re-train the last layers of a famous CNN for object detection (like Inception [74] and VGG [28]), to recognize objects that can appear in the videos from the dataset. The obtained results weren't satisfactory.

The following approach was to use a pre-existing library for object detection. We decided to use YOLOv3 [69]. Out of the pre-trained classes, we selected the following classes as relevant for the HAR system: person, bottle, cup, sofa, laptop, cell phone, book.

After getting the list of objects that appear throughout the video sequence, we correlate them with the scores obtained from the other components of the activity detection classification system. The correlation is done by assigning higher weights to the scores of activity classes that would logically need the identified objects. We define ω as an empirically chosen weighting factor.

The pseudo-code from 2 describes the algorithm for merging context information.

Algorithm 2: Context fusion mechanism

Data: $score_c$, // Class scores computed so far.
$object\_list$, // Objects that appear during the video.
C, // Set of classes.
ω // Weighting factor.

Result: $score_c$ // Modified.

```
1 begin
2     for obj ∈ object_list do
3         for c ∈ C do
4             if ∃ logical correlation between obj and c then
5                 score_c ← score_c · (1 + ω);
```

5.3 Development of the System

The HAR system has been implemented in several stages. As a general guideline, we implemented and tested each module separately, and wired them together in the end.

We started by writing functions for data manipulation (extracting frames from RGB videos and raw depth data from depth sensor data). We then applied needed pre-computations to data streams: we identified objects with object detection mechanisms, and we tracked body joints. We continued with implementing the pixel-level algorithms: MHI on RGB data, DMMs on depth data and skeletal data on body joints. We wired everything together with the channel fusion mechanism and finally added the context fusion mechanism.

The last stage of the project consisted of filming our dataset. Inspired by the results obtained while testing the system on a public dataset, we felt the need to produce a new dataset, of greater size, and with more relevant activities for the developed system. More details about the dataset can be found in section 6.1.

5.4 Chapter Summary

In this chapter, we detailed the implementation of the HAR system. Firstly, we named the technologies that we used. Then, we gave more details on the actual implementation of the fusion mechanisms: idea, pseudo-code, configurations, problems that we encountered and solutions that we found. Finally, we presented the development process from a chronological view, ordering the HAR system components by how we implemented them.

6 EVALUATION AND TESTING

This chapter presents a detailed analysis of the HAR system, regarding tests executed on its modules, and evaluations performed on its possible configurations.

Firstly, we introduce the two datasets that we used for evaluation: a public RGB-D dataset, and another one, filmed by us, with regard to the purpose of our HAR system. Then, we describe the mechanism used for module testing and perform a system evaluation. We finally compare our method with other existing methods, detailed in chapter 3.

6.1 Datasets

In order to prove the effectiveness of the proposed solution, experiments were performed on a public RGB-D dataset (*MSRDailyActivity3D*), as well as on own recordings(*PRECIS HAR Dataset*), captured with a 3D camera, namely ORBBEC Astra Pro[1].

MSRDailyActivity3D [75]
This dataset contains 16 different activities (drink, eat, read book, call cellphone, write on a paper, use laptop, use vacuum cleaner, cheer up, sit still, toss paper, play game, lay down on sofa, walk, play guitar, stand up, sit down), performed twice (once standing up, once sitting down) by 10 subjects, in an office. There are 320 videos in total. The activities were chosen as being typical ADLs that can be performed in a living room.

People interact with different objects (bottle, bag of food, cellphone, paper, laptop, vacuum cleaner, ball of paper, gaming console, game controller, guitar). This, along with the fact that depth data is somewhat noisy and subjects sometimes do not respect intuitive motion patterns, make the dataset challenging.

During the experiments, the dataset has been split into training set (subjects 1-6), validation set (subjects 7 and 8) and test set (subjects 9 and 10). We removed the scene background, including other people than the main subject that were appearing in the videos.

PRECIS HAR Dataset[2]
This is the dataset that we filmed while working on the HAR system. It has two purposes: firstly, it is intended to help us prove the efficiency and utility of the system for detecting

[1]https://orbbec3d.com/product-astra-pro/. Last accessed: 16 June 2019.
[2]http://bit.do/precis-har-dataset. Last accessed: 22 June 2019.

human activities, and secondly, we made it public, so that it can become a benchmark in finding future HAR solutions.

The dataset consists of 16 different activities (stand up, sit down, sit still, read, write, cheer up, walk, throw paper, drink from a bottle, drink from a mug, move hands in front of the body, move hands close to the body, raise one hand up, raise one leg up, fall from bed and faint), performed by 50 subjects. There are thus 800 videos in total, having both RGB and depth recordings.

The first nine classes of activities are similar to the ones from *MSRDailyActivity3D*. We chose to reproduce the activities that we consider having the most relevance for the purpose of our HAR system (i.e. indoor surveillance). Thus, we enlarged the amount of data available for training the system on those classes of activities, because our dataset is filmed in the same manner as the public dataset (same angle, distance, and field of view). A comparison between the two datasets, from the point of view of similar activities, can be found in Appendix A.

We also introduced several classes that we found useful for training the HAR system, or for its purpose. For example, classes "fall from bed" and "faint" help in designing HAR systems capable of sending emergency alerts. Keyframes from videos, along with a short description and the motivation for introducing them in the dataset can be examined in Appendix B.

The dataset does not contain other people than the video subject that appear as environmental noise. In addition, the background does not change. However, its difficulties are:

- **Different lighting conditions**: the source light is variable, sometimes having scenes with pale light and other times with sun shining directly on scene parts. These lighting variations are however only visible in the RGB stream. The depth stream is not affected by environmental changes, as far as the light is concerned. We thus want to emphasize the utility of also examining the depth stream, while classifying videos from RGB-D cameras.
- **Small camera occlusions**: from time to time, we added small occlusions to the RGB stream (we move the mouse from a part of the screen to the other), that simulate the presence of external, uncontrolled environment factors; for example, in a real scenario, a fly could pass in front of the camera.
- **No stage directions**: while filming the dataset, we did not instruct participants on how to perform each action. We simply told them the name of the action. Thus, they all interpreted them slightly differently, closer to the natural way in which a person would behave.
- **Variable video length**: because we did not impose constraints on how the participants had to perform each action, dataset videos have different lengths.

For evaluation, we divided the dataset into three parts: training (subjects 1-30), validation (subjects 31-40) and testing (subjects 41-50). We removed the background, removing information that is not of interest to the classification problem.

We also performed training and evaluation on videos from both datasets, for classes of activities that are similar. In that case, we splitted the 60 available subjects (10 from the public dataset and 50 from our own) in the following manner: 36 for training, 12 for validation and 12 for testing. We shuffled the list of videos before splitting because we wanted to have videos from both datasets in all three splits.

6.2 Module Testing

As described in section 5.3, we implemented the HAR system module by module. We wrote unit tests for each module. This helped us detect errors early, and make sure that the modules function correctly upon wiring them together.

The temporal fusion module could not be tested automatically, because its output is represented by an image summarizing a video. We tested it by visually comparing output images with input images. All performed tests were successful.

For testing the channel fusion module, we wrote unit tests in Python using the `unittest` unit testing framework. We tested the operations performed on arrays of probability results, used to combine all results in one. All written tests were passed.

The context fusion module was tested (also by writing unit tests in Python) in two steps. First, we wrote tests that check if all objects from an image have been identified by the object detection algorithm. Then, we verified if the input array of probability scores is modified correctly, accordingly to the list of identified objects. All written tests were passed.

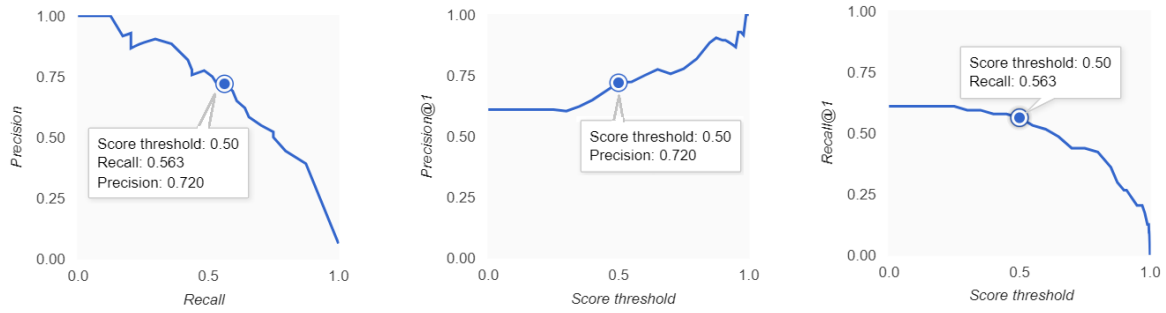
6.3 System Evaluation

In this section, we evaluate the HAR system in terms of accuracy and efficiency. We analyze each CNN independently, then we evaluate the system as a whole. We end with an analysis of the time and memory requirements of our system, compared to a 3D HAR CNN.

6.3.1 MHI-based CNN

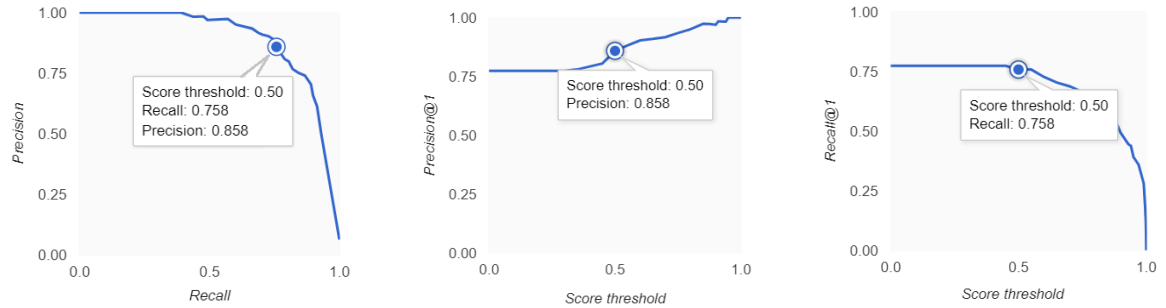
We evaluate the MHI-based CNN by performing three experiments: train a network on all activities of *MSRDailyActivity3d*, then one on all activities of *PRECIS HAR Dataset*, and finally train a third one by using common activities of the two datasets.

Train and test on MSRDailyActivity3D
Firstly, we trained a CNN based on MHIs, using the *MSRDailyActivity3D* dataset. We obtained an accuracy of 70.3%. The training time was equal to 11 minutes.



(a) Precision vs recall (b) Precision@1 vs score threshold (c) Recall@1 vs score threshold

Figure 22: MHI-based 2D CNN on *MSRDailyActivity3D*: Representations of the precision-recall tradeoff curves. @1 means that we only consider the class with the highest score.



(a) Precision vs recall (b) Precision@1 vs score threshold (c) Recall@1 vs score threshold

Figure 23: MHI-based 2D CNN on *MSRDailyActivity3D*, with data augmentation: Representations of the precision-recall tradeoff curves. @1 means that we only consider the class with the highest score.

Figure 22 shows the corresponding precision and recall curves for the MHI-based CNN without data augmentation.

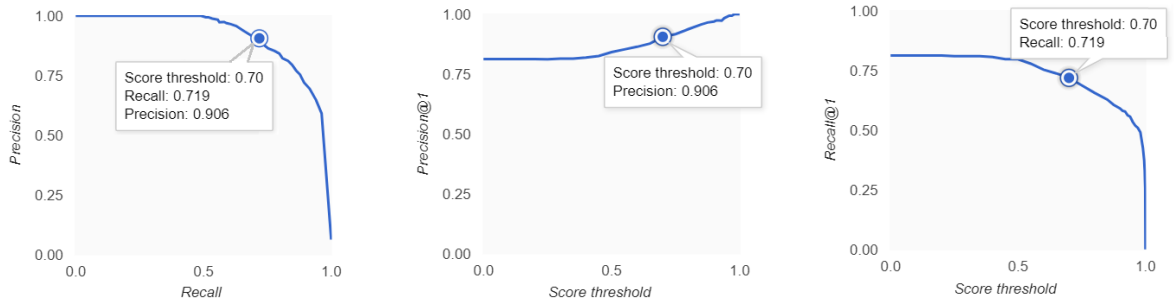
Because the accuracy was not satisfactory, we decided to augment the dataset by performing vertical flips on each MHI. Thus, from 320 samples, we obtained 640. The training time increased to 13 minutes, but we also noticed a remarkable accuracy increase: 90.63%. See Figure 23 for plots representing precision and recall curves. By analyzing the curves, we conclude that a threshold equal to 0.5 works best in case the system needs both precision and recall.

Then, Figure 24 shows the confusion matrix. On this dataset, only by training one CNN on the RGB stream, we can identify correctly almost all activity classes. However, activities that consist of similar movements are sometimes confused: calling a cellphone, drinking or playing a game are confused to eating, reading a book is confused to writing a paper and sitting still to using a laptop.

For a visual explanation of these confusions, as well as for the class scores that each wrongly classified example obtained, see Appendix C.

	call cellphone	cheer up	drink	eat	lay down on sofa	play game	play guitar	read book	sit down	sit still	stand up	toss paper	use laptop	use vaccum cleaner	walking	write on a paper
call cellphone	0,75			0,25												
cheer up		1														
drink			0,75	0,25												
eat				1												
lay down on sofa					1											
play game				0,25		0,75										
play guitar							1									
read book								0,75								0,25
sit down									1							
sit still										1						
stand up									0,25		0,75					
toss paper												1				
use laptop										0,25			0,75			
use vacuum cleaner														1		
walking															1	
write on a paper																1

Figure 24: Confusion matrix for MHI-based CNN on *MSRDailyActivity3D*, with data augmentation (vertical flipping of MHIs).



(a) Precision vs recall (b) Precision@1 vs score threshold (c) Recall@1 vs score threshold

Figure 25: MHI-based 2D CNN on *PRECIS HAR Dataset*, with data augmentation: Representations of the precision-recall tradeoff curves.

Train and test on PRECIS HAR Dataset

When training an MHI-based CNN on *PRECIS HAR Dataset*, with data augmentation, we get an accuracy of 85.63%, in 12 minutes. Figure 25 shows the precision an recall curves. From analyzing the graphs, we set the threshold to 0.7.

Figure 26 shows the confusion matrix for this experiment. We observe that the classes with most wrongly classified examples are "drink from a bottle" and "drink from a mug". This is normal because the motion paths of these classes are similar.

Other errors appear due to different ways of performing the same action, or due to environmental conditions. For example, one MHI was classified as "read book", instead of "sit still", because the subject was wearing a T-shirt with a square shape on it, that looked like a book.

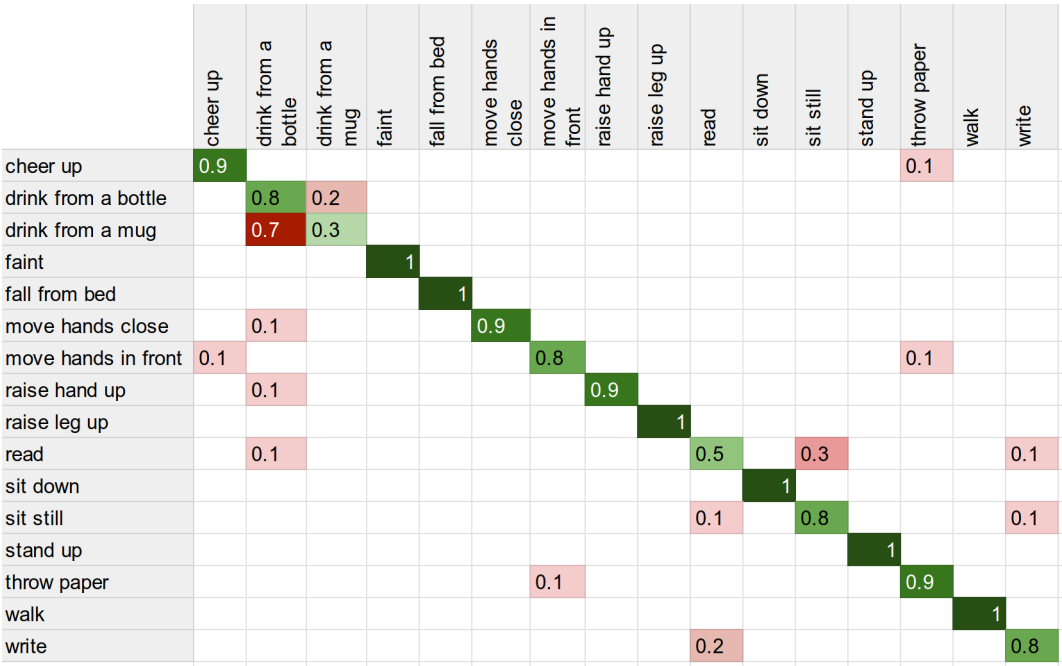


Figure 26: Confusion matrix for MHI-based CNN on *PRECIS HAR Dataset*, with data augmentation (vertical flipping of MHIs).

Train on PRECIS HAR Dataset, test on MSRDailyActivity3D

We selected the nine common activities from both datasets and used one as training set, and the other for testing. Even with data augmentation, we reach an accuracy of only 64%. After analyzing the output of the network, we concluded that the causes for this discrepancy are: a slightly different manner of filming the dataset, different performing of the same actions, different positioning of the subjects in front of the camera.

From all experiments above, we can conclude that CNNs trained on MHIs obtained from RBG stream are powerful, even when used independently. However, for actions with similar movements, we need other mechanisms. In addition, augmentation techniques can help in increasing accuracy visibly (in our case, with 20.33%), when the size of the dataset is reduced.

Last but not least, we believe that RGB stream yields to accurate results independently, because our datasets are filmed in artificial conditions (the subjects are usually facing the camera, there is little background noise). Thus, for datasets that are not synthetic, MHI-based CNNs may result in worse results.

6.3.2 DMM-based CNNs

In order to test the three types of DMM-based CNNs (front, side and top DMM-based CNNs), we performed experiments a subset of 11 activity classes from *MSRDailyActivity3D*: call cellphone, cheer up, drink, eat, play game, read book, sit down, sit still, stand up, use laptop and write on a paper. We reduced the dimensionality of the dataset to fasten the process. However, we only removed five of the activity classes that were not problematic in the previous

subsection (i.e. that were correctly classified by the RGB MHI-based CNN).

The accuracies are around 50% (47.73% for fDMM CNN, 43.18% for sDMM CNN, 43.18% for tDMM CNN). The required training time is of around 30 minutes (26 minutes for fDMM, 27 minutes for sDMM and 26 minutes for tDMM). The corresponding confusion matrices are in Appendix E.

Thus, we can conclude that on this dataset, DMMs alone are not capable of correctly identifying action categories. However, we can use their class scores to improve RGB predictions. From the confusion matrices, we observe that each CNN makes different mistakes, so the idea of combining their outputs is justifiable.

6.3.3 Skeleton Image-based CNN

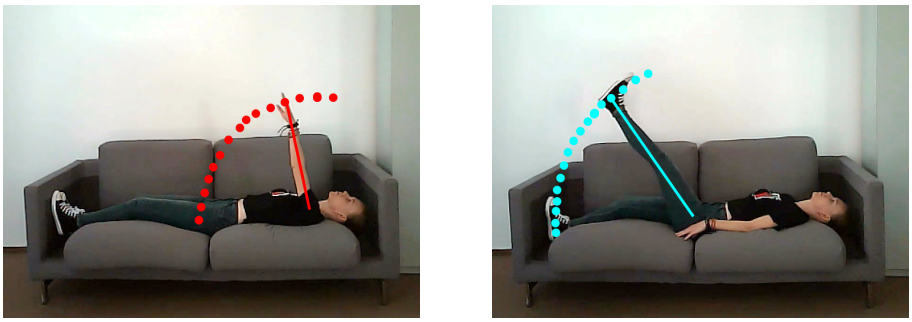
We train a CNN using skeleton images, obtained from the following three body members categories: head and shoulders, left arm and hand, right arm and hand. We do not consider other body joints, because activities performed in *MSRDailyActivity3D* dataset do not involve feet (apart from walking, which can also be detected with the three selected categories).

Training the CNN on the whole *MSRDailyActivity3D* dataset, and using data augmentation by vertical flipping, we end up with an accuracy of only 49.5%. However, the training time is small (9 minutes). The confusion matrix can be found in Figure 27.

	call cellphone	cheer up	drink	eat	lay down on sofa	play game	play guitar	read book	sit down	sit still	stand up	toss paper	use laptop	use vac. cleaner	walking	write on a paper
call cellphone	0,5			0,25								0,25				
cheer up		1														
drink	0,25		0,75													
eat	0,25		0,25					0,25								0,25
lay down on sofa					0,5						0,5					
play game						0,25	0,25	0,25					0,25			
play guitar		0,25					0,75									
read book				0,5			0,25	0,25								
sit down				0,25					0,5		0,25					
sit still										0,75			0,25			
stand up									0,25		0,75					
toss paper	0,5	0,25										0,25				
use laptop						0,25		0,5								0,25
use vac. cleaner							0,5							0,5		
walking															1	
write on a paper							0,5	0,25								

Figure 27: Confusion matrix for Skeleton image-based CNN on *MSRDailyActivity3D*, with data augmentation (vertical flipping of MHIs).

The low accuracy is expected. This happens because we do not encounter considerable human pose differences in the activities present in the dataset. In general, subjects don't change their



(a) Raise one hand up (b) Raise one leg up

Figure 28: Body parts identification in frames from *PRECIS HAR Dataset*: By depicting lower and upper body parts in different colors in a skeleton image, we can distinguish actions with the same motion trajectory, but that are executed with different body parts.

poses significantly. There are only a few classes (the ones that were correctly identified, having high scores in the confusion matrix) for which the skeleton image-based CNN is useful. See Appendix F for a concrete example.

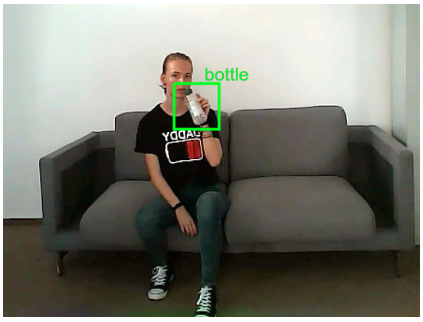
We also test the skeleton image-based CNN on two classes from *PRECIS HAR Dataset*, that were introduced especially for this purpose: "raise one hand up" and "raise one leg up". Even if we already got accuracy 100% for these classes with the MHi-based CNN, we observe that the skeleton image-based CNN yields higher confidence scores. For example, for the actions presented in Figure 28, the MHI-based CNN indicated "raise one hand up" with confidence 0.22 and "raise one leg up" with confidence 0.26, while the skeleton-based CNN scored confidence scores of 0.75, respectively 0.82.

6.3.4 Full Configuration

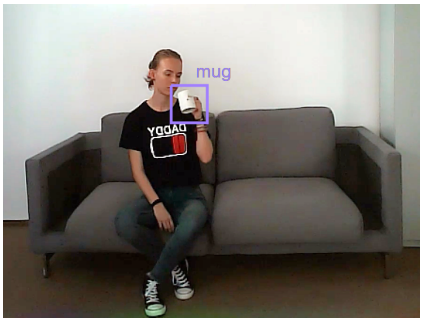
On *MSRDailyActivity3D*, using the full HAR system, with all fusion mechanisms (temporal, channel, context), we manage to attain 98.43% accuracy, thus exceeding all independent results. We can correct most of the wrong classifications from Appendix C via the context fusion mechanism (we identify with object detection a cellphone, a bottle, a game console, a book, a laptop, and a vacuum cleaner). The remaining ones (corresponding to activities stand up and sit down) are corrected via the channel fusion mechanism, by mixing information from all five CNNs. The only example that we are not able to correct is one from the category "write on a paper".

If we do no limit the autonomy of the system by using the context fusion mechanism, we obtain an accuracy of 93.75%.

In *PRECIS HAR Dataset*, the channel fusion and context fusion mechanisms also lead to an increase of accuracy: from 85.63%, we get to 94.38%. The context fusion mechanism has the most significant impact because it helps the HAR system distinguish between two of the most problematic classes: "drink from a bottle" and "drink from a mug". Analyzing only



(a) Drink from a bottle



(b) Drink from a mug

Figure 29: Contextual objects identification in frames from *PRECIS HAR Dataset*: By identifying objects present in certain frames of a video, we can distinguish actions with the same motion trajectory, but that involve different objects.

MHIs of a video is almost impossible to distinguish between the two (see Figure 26). However, identifying those objects leads to the correct classification. Figure 29 shows how the mug and bottle are identified during the video.

6.3.5 Efficiency Analysis

We tested the efficiency of the proposed HAR system against predictions made with a 3D CNN. We used a minimal configuration of the system (only RGB stream, encapsulated in an MHI-based CNN), with data augmentation. We trained a 3D CNN using Google Video Intelligence API[3] and the videos from *MSRDailyActivity3D* dataset as input.

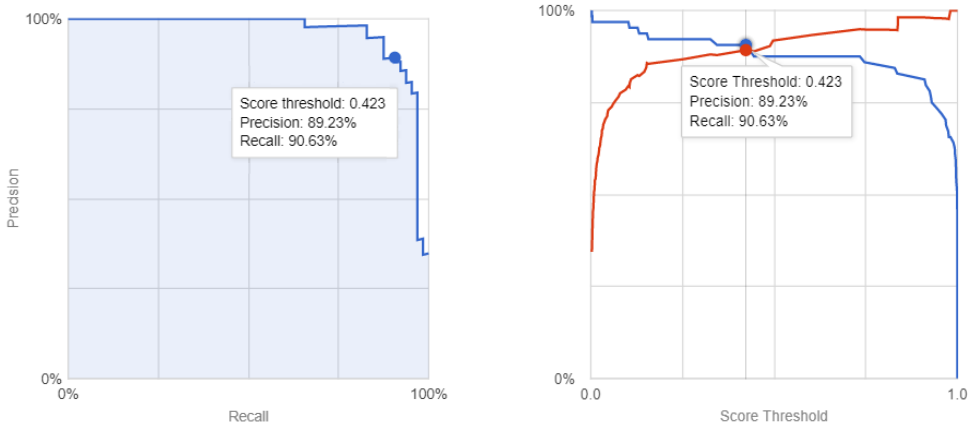
By using the 3D CNN, we attain an accuracy of 89%, with a training time of 64 minutes. The precision and recall curves are illustrated in Figure 30.

Comparing these results with the corresponding ones from subsection 6.3.1, we observe the following: while the precision-recall curves and accuracy are similar (90.63% for the 2D CNN and 89% for the 3D CNN), the training time differs a lot: for the 2D CNN, 13 minutes are required, whereas for the 3D CNN, training takes 64 minutes. Plus, the amount of data that had to be uploaded to the server for training the 3D CNN is way bigger, than in the case of the 2D CNN: 2.23 GB instead of 13.5 MB.

We thus prove experimentally that using fusion mechanisms, along with 2D CNNs is more efficient than using a 3D CNN.

As far as the total classification time is concerned, the necessary time ranges between 0.1 and 1.8 seconds, depending on the duration of the video. In our datasets, no video is longer than 12 seconds. For connecting to the server that holds the trained model, we used an Internet connection with a download speed of 96.86 Mbps and upload speed of 31.21 Mbps[4].

[3] https://cloud.google.com/video-intelligence/. Last accessed: 23 June 2019.
[4] According to https://www.speedtest.net. Last accessed: 24 June 2019.



(a) Precision vs recall

(b) Precision@1 (red) and Recall@1 (blue) vs score threshold

Figure 30: 3D CNN trained on *MSRDailyActivity3D*: Representations of the precision-recall tradeoff curves. @1 means that we only consider the class with the highest score.

6.4 Comparison with Existing Methods

Table 5 shows an accuracy comparison on the *MSRDailyActivity3D* dataset, considering other State of the Art activity recognition systems. We note that our system has one of the top scores (even without context fusion mechanism).

Table 5: Performance comparison on *MSRDailyActivity3D* dataset

Method	Accuracy(%)
Proposed system	**98.43%**
Lu, Jia & Tang (2014) [76]	95.63%
Jalal et al. (2017) [77]	94.1%
Proposed system without context fusion	**93.75%**
Althloothi et al. (2014) [78]	93.1%
Ramanathan, Kochanowicz & Thalmann (2019) [79]	90.84%
Wang et al. (2012) [75]	85.75%
Asadi-Aghbolaghi et al. (2017) [80]	82.50%

6.5 Chapter Summary

In this chapter, we firstly evaluated the HAR system on a public dataset, namely *MSRDailyActivity3D*, and then on our dataset, *PRECIS HAR Dataset*. The experiments proved its efficiency, regarding the accuracy, training time and data transferred to the server, and its relevance among other HAR systems from technical literature.

It has to be noted that the used datasets are quite small, and even with data augmentation techniques, results are affected by their sizes. Training the system on bigger datasets is expected to yield even higher results.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we implemented an efficient mechanism for human activity recognition, designed to be mainly used for indoor assistance for older people. In particular:

- We analyzed the technical literature and compared State of the Art methods from the domain of human activity recognition.
- We investigated different fusion mechanisms (temporal, channel, context) and their corresponding methods.
- We designed and implemented a HAR system that is modular and easily portable on hardware devices.
- We filmed a dataset, *PRECIS HAR Dataset*, with 50 subjects, each performing 16 actions chosen such that they illustrate the relevance of all fusion mechanisms.
- We trained and evaluated the system on two datasets, each with 16 classes of activities. We obtained high accuracies (98.43% on *MSRDailyActivity3D* and 94.38% on *PRECIS HAR Dataset*).
- We made the HAR system effective: a video classification requires between 0.1 and 1.8 seconds. The traffic necessary for training or classifying is reduced.

We conclude that the hypotheses that we had in mind while starting to develop the HAR system were proven. From the high accuracy scored by the MHi-CNNs, we affirm that a video can be efficiently compressed as an image, by using a pixel-fusion mechanism (in our case, the so-called motion history image).

Then, we established evidence for the fact that multiple predictions yield to better results. By individually training several CNNs and then combining their results, we obtained a slightly more accurate classification.

In addition to that, our experiments prove the importance of each information stream (RGB, depth, skeleton images, list with surrounding objects), as they all help differently in identifying actions. It was observed that:

- MHI-based CNNs perform well in identifying dynamic activities (walking, waving, cheering up, clapping hands) that take place in good lighting conditions.
- DMM-based CNNs can identify activities that take place in bad (or nonexistent) lighting conditions (falling from the bed, sleepwalking, fainting); they are also useful in activities that cannot be identified only based on the front view (read book, sit still).
- Skeleton image-based CNNs help in identifying dynamic activities and can sometimes act as corrections for the MHIs; however, their trust coefficient is low. Next, they help

in identifying actions with the same motion curve but performed with different body parts (raise one hand or one leg up).
- Surrounding objects help the system decide when activities that involve the same movement (eat, drink, call cellphone), or objects with an almost similar shape (drink from a bottle or a mug) are encountered.

Our work is relevant for the domain of human activity recognition because it brings innovation, meanwhile being tangible (i.e. we do not only present concepts theoretically, but we also implement and evaluate them). Plus, we introduced a new RGB-D dataset, that will help the computer vision community find new solutions for activity detection.

Future Work

In the future, we plan to train and test the HAR system on more datasets. For now, we can detect a limited number of activity classes (out of which some are also just for scientific purposes, and do not represent ADLs of an older person). The system would be trained on more activities that are common to older people. Ideally, we should train the system on a dataset that is not synthetic.

We also intend to integrate the HAR system with a programmable robot (like TIAGo[1]), to integrate surveillance capabilities with assertive ones. The robot could assist people in doing certain activities. In addition, an emergency system (for example, one that fires an alarm and calls relatives in case someone faints/falls) has to be integrated. A robot could follow someone everywhere in their house.

Then, the module has to be trained to work even if people are not facing the camera. In this paper, all experiments had subjects positioned in this way. It would also be useful to integrate multi-person activity detection.

[1] https://tiago.pal-robotics.com/. Last accessed: 24 June 2019.

BIBLIOGRAPHY

[1] M. B. Rasheed, N. Javaid, T. A. Alghamdi, S. Mukhtar, U. Qasim, Z. A. Khan, and M. H. B. Raja. Evaluation of human activity recognition and fall detection using android phone. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 163–170, March 2015.

[2] Simon Kozina, Hristijan Gjoreski, Matjaz Gams, and Mitja Lustrek. Efficient activity recognition and fall detection using accelerometers. volume 386, 09 2013.

[3] Henry Nweke, Teh Wah, Ghulam Mujtaba, and Mohammed Ali Al-garadi. Data fusion and multiple classifier systems for human activity detection and health monitoring: Review and open research directions. *Information Fusion*, 46, 06 2018.

[4] Ahmad Jalal, Shaharyar Kamal, and Daijin Kim. A depth video-based human detection and activity recognition using multi-features and embedded hidden markov models for health care monitoring systems. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4:54, 01 2017.

[5] M. Adil, R. Simon, and S. K. Khatri. Automated invigilation system for detection of suspicious activities during examination. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 361–366, Feb 2019.

[6] J. Monteiro, R. Granada, R. C. Barros, and F. Meneguzzi. Deep neural networks for kitchen activity recognition. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2048–2055, May 2017.

[7] Georg Waltner, Thomas Mauthner, and Horst Bischof. Indoor activity detection and recognition for sport games analysis. 05 2014.

[8] Swati Dewan, Shubham Agarwal, and Navjyoti Singh. A deep learning pipeline for indian dance style classification. page 7, 04 2018.

[9] Jing Zhang, Wanqing Li, Philip O. Ogunbona, Pichao Wang, and Chang Tang. Rgb-d-based action recognition datasets: A survey. *CoRR*, abs/1601.05511, 2016.

[10] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:257–267, 2001.

[11] Dean Abbott. Combining models to improve classifier accuracy and robustness. *Information Fusion - INFFUS*, 01 1999.

[12] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[13] Ajantha Atukorale and Ponnuthurai Suganthan. Combining classifiers based on confidence values. pages 37–40, 01 1999.

[14] Kesar Singh, Minge Xie, and William E. Strawderman. Combining information from independent sources through confidence distributions. *Ann. Statist.*, 33(1):159–183, 02 2005.

[15] United Nations. *World Population Ageing 2015*. 2017.

[16] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73:428–440, 1999.

[17] J. K. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images-a review. *Proceedings of the IEEE*, 76(8):917–935, Aug 1988.

[18] Koichiro Akita. Image sequence analysis of real world human motion. *Pattern Recognition*, 17(1):73–83, 1984.

[19] Yu Kong and Yun Fu. Human action recognition and prediction: A survey. *CoRR*, abs/1806.11230, 2018.

[20] T. N. Theis and H. . P. Wong. The end of moore's law: A new beginning for information technology. *Computing in Science Engineering*, 19(2):41–50, Mar 2017.

[21] S. Jain. Quantum computer architectures: A survey. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2165–2169, March 2015.

[22] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[23] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, Jan 2013.

[24] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03*, ICPR '04, pages 32–36, Washington, DC, USA, 2004. IEEE Computer Society.

[25] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, June 2014.

[26] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[27] Pushpajit Khaire, Javed Imran, and Praveen Kumar. *Human Activity Recognition by Fusion of RGB, Depth, and Skeletal Data*, pages 409–421. 01 2018.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[29] Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 168–172, 2015.

[30] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. P-CNN: pose-based CNN features for action recognition. *CoRR*, abs/1506.03607, 2015.

[31] 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, Dec 2013.

[32] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1194–1201, June 2012.

[33] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. Compressed video action recognition. *CoRR*, abs/1712.00636, 2017.

[34] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[35] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. *CoRR*, abs/1708.05038, 2017.

[36] King-Shy Goh, Edward Chang, and Kwang-Ting Cheng. Svm binary classifier ensembles for image classification. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 395–402, New York, NY, USA, 2001. ACM.

[37] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[38] Mark Berthod, Zoltan Kato, Shan Yu, and Josiane Zerubia. Bayesian image classification using Markov random fields. *Image and Vision Computing*, 14:285–295, 1996.

[39] P. N. Druzhkov and V. D. Kustikova. A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, 26(1):9–15, Jan 2016.

[40] Rafał Grycuk, Michał Knop, and Sayantan Mandal. Video key frame detection based on surf algorithm. In Leszek Rutkowski, Marcin Korytkowski, Rafal Scherer, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek M. Zurada, editors, *Artificial Intelligence and Soft Computing*, pages 566–576, Cham, 2015. Springer International Publishing.

[41] Costas Panagiotakis, Nelly Ovsepian, and Elena Michael. Video synopsis based on a sequential distortion minimization method. In *Proceedings, Part I, of the 15th International Conference on Computer Analysis of Images and Patterns - Volume 8047*, CAIP 2013, pages 94–101, New York, NY, USA, 2013. Springer-Verlag New York, Inc.

[42] Md. Atiqur Rahman Ahad, J K. Tan, Haekwon Kim, and S Ishikawa. Motion history image: Its variants and applications. *Machine Vision and Applications*, 23:255–281, 03 2010.

[43] Allah Bux Sargana, Plamen Angelov, and Zulfiqar Habib. *Vision Based Human Activity Recognition: A Review*, volume 513, pages 341–371. 01 2017.

[44] Tushar Dobhal, Vivswan Shitole, Gabriel Thomas, and Girisha Navada. Human activity recognition using binary motion image and deep learning. *Procedia Computer Science*, 58:178 – 185, 2015. Second International Symposium on Computer Vision and the Internet (VisionNet'15).

[45] Deqing Sun, Stefan Roth, J.P. Lewis, and Michael Black. Learning optical flow. volume 1, pages 83–97, 10 2008.

[46] Laura Sevilla-Lara, Yiyi Liao, Fatma Güney, Varun Jampani, Andreas Geiger, and Michael J. Black. On the integration of optical flow and action recognition. *CoRR*, abs/1712.08416, 2017.

[47] Anurag Ranjan, Javier Romero, and Michael J. Black. Learning human optical flow. *CoRR*, abs/1806.05666, 2018.

[48] Xiaodong Yang, Chenyang Zhang, and YingLi Tian. Recognizing actions using depth motion maps-based histograms of oriented gradients. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 1057–1060, New York, NY, USA, 2012. ACM.

[49] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *Proceedings of*

the Second International Conference on Human Behavior Unterstanding, HBU'11, pages 29–39, Berlin, Heidelberg, 2011. Springer-Verlag.

[50] J. Arunnehru, G. Chamundeeswari, and S. Prasanna Bharathi. Human action recognition using 3d convolutional neural networks with 3d motion cuboids in surveillance videos. *Procedia Computer Science*, 133:471 – 477, 2018. International Conference on Robotics and Smart Manufacturing (RoSMa2018).

[51] Tin Kam Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, Jan 1994.

[52] R. P. W. Duin. The combining classifier: to train or not to train? In *Object recognition supported by user interaction for service robots*, volume 2, pages 765–770 vol.2, Aug 2002.

[53] E. Triantaphyllou, B. Shu, S. Nieto Sanchez, and T. Ray. Multi-criteria decision making: An operations research approach, 1998.

[54] N. Ramoly, V. Vassout, A. Bouzeghoub, M. A. E. Yacoubi, and M. Hariz. Refining visual activity recognition with semantic reasoning. In *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pages 720–727, March 2017.

[55] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos. *CoRR*, abs/1806.06157, 2018.

[56] Isibor Ihianle, Usman Naeem, Syed Islam, and Abdel-Rahman Tawil. A hybrid approach to recognising activities of daily living from object use in the home environment. *Informatics*, 5:6, 01 2018.

[57] S. Ji, W. Xu, M. Yang, and K. Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, Jan 2013.

[58] Simon S. Haykin. *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition, 2009.

[59] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. A survey on neural architecture search, 2019.

[60] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016.

[61] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20:55:1–55:21, 2018.

[62] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *CoRR*, abs/1611.02167, 2016.

[63] Zhao Zhong, Junjie Yan, and Cheng-Lin Liu. Practical network blocks design with q-learning. *CoRR*, abs/1708.05552, 2017.

[64] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.

[65] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Reinforcement learning for architecture search by network transformation. *CoRR*, abs/1707.04873, 2017.

[66] Martin Wistuba. Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations. In *ECML/PKDD*, 2018.

[67] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *CoRR*, abs/1711.00436, 2017.

[68] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[69] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[70] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[71] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[72] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[73] Jian Liu, Naveed Akhtar, and Ajmal Mian. Skepxels: Spatio-temporal image representation of human skeleton joints for action recognition. *CoRR*, abs/1711.05941, 2017.

[74] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[75] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1290–1297, June 2012.

[76] C. Lu, J. Jia, and C. Tang. Range-sample depth feature for action recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 772–779, June 2014.

[77] Ahmad Jalal, Yeon-Ho Kim, Yong-Joong Kim, Shaharyar Kamal, and Daijin Kim. Robust human activity recognition from depth video using spatiotemporal multi-fused features. *Pattern Recognition*, 61:295 – 308, 2017.

[78] Salah Althloothi, Mohammad H. Mahoor, Xiao Zhang, and Richard M. Voyles. Human activity recognition using multi-features and multiple kernel learning. *Pattern Recognition*, 47(5):1800 – 1812, 2014.

[79] Manoj Ramanathan, Jaroslaw Kochanowicz, and Nadia Thalmann. Combining pose-invariant kinematic features and object context features for rgb-d action recognition. *International Journal of Machine Learning and Computing*, 9:44–50, 02 2019.

[80] Maryam Asadi-Aghbolaghi, Hugo Bertiche, Vicent Roig, Shohreh Kasaei, and Sergio Escalera. Action recognition from rgb-d data: Comparison and fusion of spatio-temporal handcrafted features and deep strategies. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 3179–3188, 2017.

A CORRELATION BETWEEN DATASETS

Table 6 parallelly displays keyframes from the two datasets used for evaluation of the HAR system: MSRDailyActivity3D [75] and PRECIS HAR Dataset. Each dataset has 16 activities, out of which 9 are similar.

Table 6: Frames corresponding to similar activities from the two datasets used for evaluation

Activity	MSRDailyActivity3D Dataset	PRECIS HAR Dataset
stand up		
sit down		
sit still		
read		

Table 6 (continued)

Activity	MSRDailyActivity3D Dataset	PRECIS HAR Dataset
write		
cheer up		
walk		
throw paper		
drink		

B ACTIVITIES INTRODUCED BY PRECIS HAR DATASET

Table 7 briefly describes the activities introduced by our dataset, along with the motivation for including them. We display one illustrative frame for each activity.

Table 7: Frames corresponding to activities introduced by PRECIS HAR Dataset

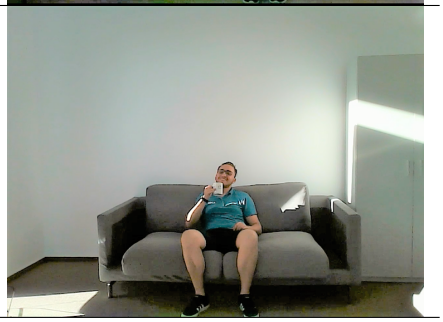
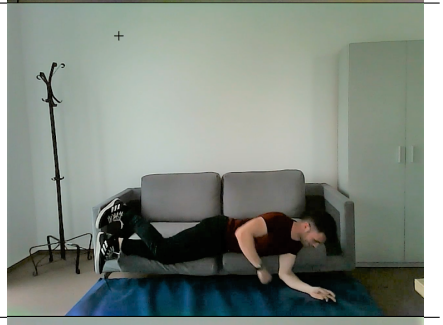
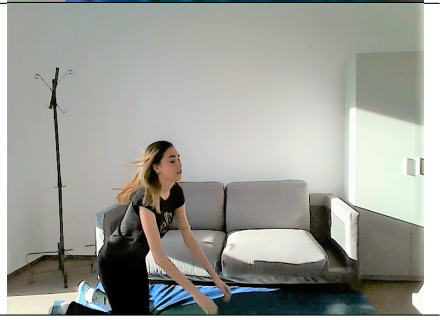
Activity	**Description**	**PRECIS HAR Dataset**
drink from a bottle	The subject has to drink from a bottle of water, sitting down or standing up. The purpose of this action is to be distinguished from an action with similar movements, but involving a different object (drink from a mug).	
drink from a mug	The subject has to drink from a mug, sitting down or standing up. The purpose of this action is to be distinguished from an action with similar movements, but involving a different object (drink from a bottle).	
move hands in front of the body	Facing the camera, the subject has to move their hands alternatively, keeping them stretched. This action has to be distinguished from an action that looks similar if viewed frontally, but different if viewed laterally (move hands close to the body).	
move hands close to the body	Facing the camera, the subject has to move their hands alternatively, keeping them near their body. This action has to be distinguished from an action that looks similar if viewed frontally, but different if viewed laterally (move hands in front of the body).	

Table 7 (continued)

Activity	Description	PRECIS HAR Dataset
raise one hand up	Lying on the couch, the subject has to raise one hand. This action has to be distinguished from one involving a similar movement, but of another body part (raise one leg up).	
raise one leg up	Lying on the couch, the subject has to raise one leg. This action has to be distinguished from one involving a similar movement, but of another body part (raise one hand up).	
fall from bed	Simulate falling from bed while sleeping. It helps in designing HAR systems that detect anomalies and emergencies.	
faint	Simulate fainting or stumbling. It helps in designing HAR systems that detect anomalies and emergencies.	

C INCORRECT MHI-BASED CNN RESULTS

Table 8 presents wrongly (weakly) classified videos from the evaluation set. We use all 16 videos from the *MSRDailyActivity3D* dataset and vertical flipping as augmentation technique. Apart from the 11 videos that are not correctly or strongly classified, all others are.

Table 8: Wrong classifications and weak classifications (class score is under a threshold, set at 0.7) resulted from the MHI-based CNN results of *MSRDailyActivity3D*. We list the first three classes and scores, out of five.

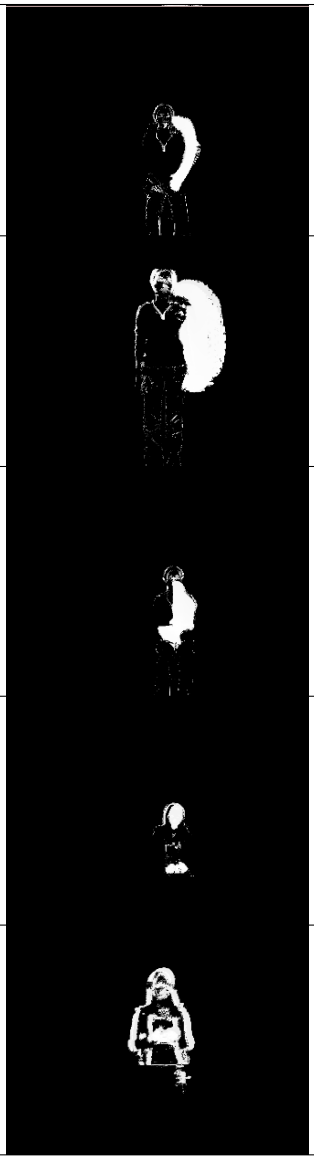
Real class	**MHI**	**C1**	**S1**	**C2**	**S2**	**C3**	**S3**
call cell-phone		eat	0.626	drink	0.170	**call cell-phone**	0.107
call cell-phone		call cell-phone	**0.630**	drink	0.253	eat	0.086
drink		eat	0.654	**drink**	0.163	call cell-phone	0.057
play game		play game	**0.540**	write on a paper	0.306	read book	0.077
play game		eat	0.622	**play game**	0.227	call cell-phone	0.050

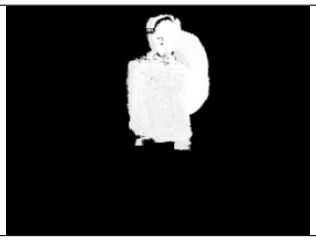
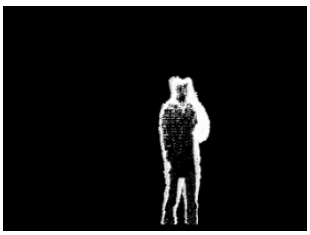
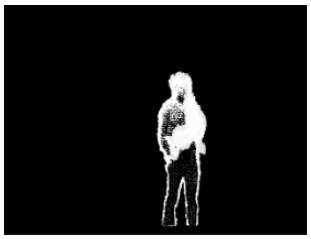
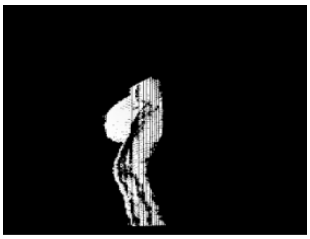
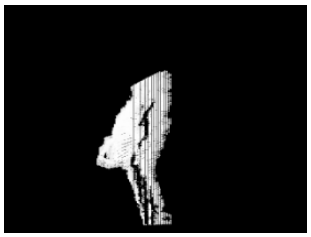
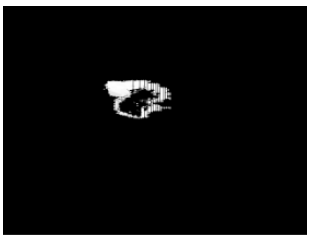
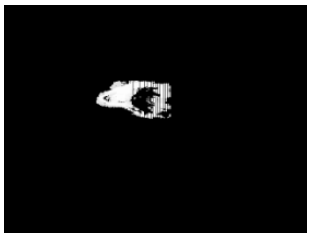
Table 8 (continued)

Real class	MHI	C1	S1	C2	S2	C3	S3
read book		write on a paper	0.753	**read book**	0.219	drink	0.017
sit down		sit down	**0.597**	stand up	0.400	play game	0.002
stand up		sit down	0.834	**stand up**	0.162	sit still	0.001
use laptop		sit still	0.596	**use laptop**	0.266	write on a paper	0.136
use vac-uum cleaner		use vac-uum cleaner	**0.572**	toss paper	0.321	stand up	0.028
write on a paper		write on a paper	**0.558**	call cell-phone	0.290	read book	0.081

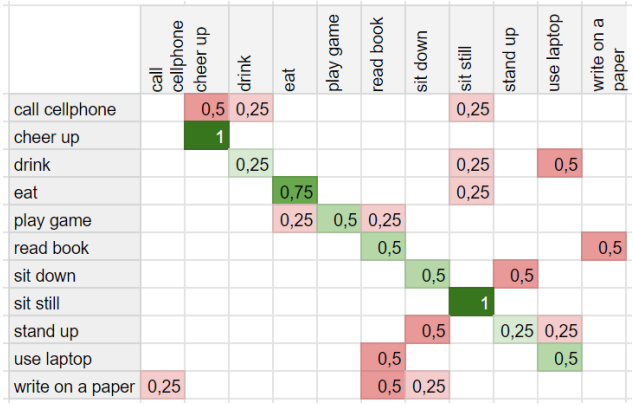
D ILLUSTRATION OF DMM-BASED CNN UTILITY

Table 9 presents two examples where the utility of considering DMMs is evident. These activities look almost similar, when examining their MHIs, but are a lot different, when considering their DMMs.

Table 9: Visual example that justifies the utility of the DMM-based CNNs

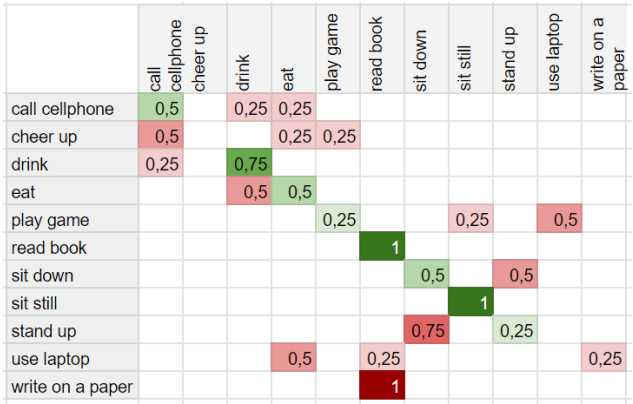
CNN base	**Activity 1: drink**	**Activity 2: eat**	**Explanation**
MHI			These two activities have a resembling MHI. By only looking at the MHI, it is hard to notice significant differences between the two movement maps.
Front DMM			The front DMM represents the frontal movement, like the MHI, but from a slightly different angle, and from a point of observation that is placed further. Differences start to be noticeable.
Side DMM			Even more than in the front DMM, the side DMM perfectly illustrates the nature of movement: we can now easily distinguish between the person drinking and eating.
Top DMM			The top DMM supports the distinction made by the two previous DMMs: we observe that activity 1 is performed only with one hand (drink), while activity 2 involves both habds (eat).

E CONFUSION MATRICES FOR DMM-BASED CNNS



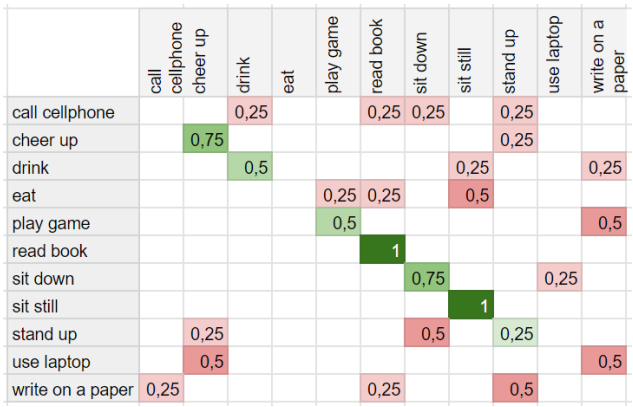
	call cellphone	cheer up	drink	eat	play game	read book	sit down	sit still	stand up	use laptop	write on a paper
call cellphone		0,5	0,25					0,25			
cheer up		1									
drink			0,25					0,25		0,5	
eat				0,75				0,25			
play game				0,25	0,5	0,25					
read book						0,5					0,5
sit down							0,5		0,5		
sit still								1			
stand up							0,5		0,25	0,25	
use laptop						0,5				0,5	
write on a paper	0,25					0,5	0,25				

(a) Front DMM-based CNN



	call cellphone	cheer up	drink	eat	play game	read book	sit down	sit still	stand up	use laptop	write on a paper
call cellphone	0,5		0,25	0,25							
cheer up	0,5			0,25	0,25						
drink	0,25		0,75								
eat			0,5	0,5							
play game					0,25			0,25		0,5	
read book						1					
sit down							0,5		0,5		
sit still								1			
stand up							0,75		0,25		
use laptop				0,5		0,25					0,25
write on a paper						1					

(b) Side DMM-based CNN



	call cellphone	cheer up	drink	eat	play game	read book	sit down	sit still	stand up	use laptop	write on a paper
call cellphone			0,25			0,25	0,25		0,25		
cheer up		0,75							0,25		
drink			0,5					0,25			0,25
eat					0,25	0,25		0,5			
play game					0,5						0,5
read book						1					
sit down							0,75			0,25	
sit still								1			
stand up		0,25					0,5		0,25		
use laptop		0,5									0,5
write on a paper	0,25					0,25			0,5		

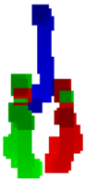
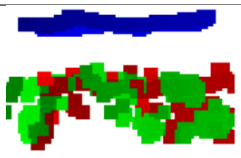
(c) Top DMM-based CNN

Figure 31: Confusion matrices for DMM-based CNNs on a subset of 11 activities from *MSR-DailyActivity3D*. We note that classifications are not extremely accurate (due to the small dataset without data augmentation and DMM complexity), but the different scores help in obtaining a combined, accurate score.

F CLASSES OF ACTIVITIES WITH VARIANT HUMAN POSE

Table 10 presents some examples of skeleton images resulted from videos of activities where the subject changes its pose, according to the activity that they perform.

Table 10: Correct classifications resulted from the skeleton image-based CNN trained on *MSRDailyActivity3D*. We list the first three classes and scores, out of five.

Real class	MHI	C1	S1	C2	S2	C3	S3
cheer up		cheer up	0.290	toss paper	0.275	play guitar	0.170
cheer up		cheer up	0.904	call cell-phone	0.040	use vac-uum cleaner	0.017
stand up		stand up	0.618	sit down	0.353	lay down on sofa	0.011
use vac-uum cleaner		use vac-uum cleaner	0.373	use laptop	0.169	sit still	0.074
walking		walking	0.983	cheer up	0.003	use laptop	0.002